Universität Karlsruhe (TH)
Research University · founded 1825

Wireless Sensor Network Pattern Based Fault Isolation in Industrial Applications

Dissertation
Prof. Dr. Wilfried Juling
Fakultät für Informatik
Universität Karlsruhe (TH)

by

Luciana Moreira Sá de Souza

Advisors:
Prof. Dr. Wilfried Juling
Prof. Dr. Christian Becker

Defended on: 2009-06-05

Institut für Telematik TELEMATICS

Abstract

A next generation technology for environment awareness, called Wireless Sensor Network (WSN)s, is emerging with the potential to bring enterprise systems to a new level of business process management. Due to the tight integration of WSNs with the environment and the possibility to attach them to physical items, WSNs reduce the information gap that exists between the real world and its digital representation, enabling a new paradigm of business process execution.

However, regardless of high reliability of single elements present in WSNs, faults in the respective components always occur. Therefore, the adoption of WSNs in enterprise environments requires a constant supervision to ensure the reliability and availability of the provided services. In general and according to the scenario in which the WSN is applied, the effects of a failure may range from economic losses to contaminating the environment and risking human lives. Hence, the successful deployment of WSNs in business processes relies on efficient fault diagnostic systems, in order to reduce maintenance time and costs.

As enterprise business systems are required to work with a broad range of wireless sensor network platforms, integration mechanisms and fault diagnosis methods cannot rely on specific functionalities and information that may not be available at all platforms.

Hence, this thesis approaches the research question of how to integrate heterogeneous WSNs with enterprise business systems, and how to identify the malfunctioning components of these networks based only on a restricted information set that can be provided by most WSNs. Therefore, this restricted information set forms an appropriate WSN platform abstraction for this thesis. This restricted set of information is based on sensor node identification number, timestamp information and the time interval between messages (heart beat).

As a solution, we propose a framework that creates a hardware platform abstraction for enterprise applications. This framework enables the integration of heterogeneous WSNs with backend applications and defines the mechanisms for fault diagnosis and recovery.

As core contribution, this thesis proposes a fault isolation method based on the identification of system patterns. This approach has the ability to learn the behaviour of the network when specific failures occur and to combine known patterns. This reduces the need of expert system knowledge to determine the causes of misbehaviour in the network. The proposed solution proved to be efficient in identifying failures, and represents a new method for isolating failures in wireless sensor networks.

Zusammenfassung

Sensornetze bezeichnen eine aufkommende Technologie der nächsten Generation zur Über-wachung der Umgebung mittels Sensoren. Sie bergen das Potential, betriebliche Informati-onssysteme auf die nächste Stufe des Geschäftsprozessmanagements zu führen. Aufgrund ihrer festen Integration in der Umgebung und der Möglichkeit, sie an physischen Gegenständen zu befestigen, reduzieren Sensornetze die Informationslücke zwischen der Realität und ihrer digitalen Repräsentation und ermöglichen so völlig neue Möglichkeiten bei der Ausführung von Geschäftsprozessen.

Trotz der hohen Zuverlässigkeit einzelner Elemente eines Sensornetzes treten immer wieder Fehler in den jeweiligen Komponenten auf. Aus diesem Grund erfordert der Einsatz von Sensornetzen in Unternehmen deren permanente Überwachung, um die Zuverlässigkeit und Verfügbarkeit der bereitgestellten Dienste sicherzustellen. Derartige Fehler können finanzielle Verluste zu Folge haben, im Extremfall je nach Szenario aber auch Gefahren für Mensch und Umwelt. Daher ist der erfolgreiche Einsatz von Sensornetzen im Rahmen von Geschäftsprozessen auf effiziente Fehlerdiagnosesysteme angewiesen, um Wartungszeit und Wartungskosten zu reduzieren.

Da betriebliche Informationssysteme unterschiedlichste Sensornetzplattformen unterstützen müssen, können sich Integrationsmechanismen und Fehlerdiagnosemethoden nicht auf spezifische Funktionalitäten und Informationen stützen, die nicht von allen Plattformen unterstützt werden.

Aus diesem Grund geht diese Arbeit der Forschungsfrage nach, wie heterogene Sensornetze in betriebliche Informationssysteme integriert und fehlerhafte Komponenten innerhalb dieser Netze identifiziert werden können. Dabei darf nur auf grundlegende Informationen zugegriffen werden, die von jedem Sensornetz bereitgestellt werden können, um eine gemeinsame Abstraktionsebene zu schaffen. Diese grundlegenden Informationen setzen sich aus gelesenen Sensorwerten, Identifikationsnummern, Zeitstempeln und Intervallen zwischen gelesenen Sensorwerten zusammen.

Die Lösung besteht aus einem Framework. Es enthält eine Hardwareabstaktionschicht für betriebliche Informationssysteme, unterstützt dadurch die Intergration heterogener Sensornetzwerke und erlaubt Fehlerdiagnose und -beseitigung.

Der zentrale Beitrag dieser Arbeit ist eine Methode zur Fehlerdiagnose, die auf der Erkennung von Systemmustern beruht. Dieser Ansatz besitzt die Fähigkeit, das Verhalten des Netzwerks beim Auftritt von Fehlern zu erlernen und bekannte Muster zu kombinieren. Dadurch wird das zur Bestimmung der Ursachen eines Fehlverhaltens des Sensornetzes benötigte Expertenwissen verringert. Die vorgestellte Lösung hat sich hinsichtlich der Erkennung von Fehlern als effizient erwiesen und stellt eine neue Strategie zur Isolierung von Fehlern in Sensornetzen dar.

Acknowledgements

The path to achieve a doctoral degree is one of the hardest challenges I faced during my lifetime. It already became clear to me on the first year of my research that it is a lonely path, and yet one that I could not have gone through without the personal and practical support of several people.

Although the list of those that I wish to thank extends beyond the limits of this page, I would like to give a special token of my appreciation to the following people and institutions for their dedication, prayers, and support:

I would like to thank the University of Karlsruhe and SAP Research for giving me the opportunity to pursue this work in an inspiring and motivating environment.

My supervisors, Prof. Dr. Wilfried Juling, Prof. Dr. Christian Becker, and Christian Decker, the discussions and insights have significantly strengthened this study. It has been an honour to work with you and I will always be thankful for your wisdom.

Harald Vogt, thank you for the brain storming sessions and the valuable advices you gave me. Your help came at crucial moments, and I will always be thankful for that.

Zoltan Nochta, thank you for reading my thesis and providing valuable feedback that considerably improved my work.

Special thanks to my colleagues Jens, Patrik, Nina and Stamatis, I had the opportunity to share the office with some of you which was a really nice and exciting experience. Thank you for the nice evenings and the well known *"Wine of Friday"* event.

To my dear friend Asma, thank you for the long talks, the kind gestures and the friendship we developed throughout these years. The memories of the nice moments and difficult times we overcame together will always stay in my heart.

To my grandmother Dalva, for all your prayers, and for being all that a person can wish for in a grandmother.

To my sister Carla, thank you for the support and for bringing Gabriela into my life. Her pictures and videos made my life full of joy every day.

To my parents Otávio and Eliane, thank you for the guidance, education and love you always gave me during my life, without it I would not be where I am today.

To my love, words will never express all my gratitude for the support you gave me, the patience you had, and the sacrifices you made for me to finish this thesis. Thank you for always being there for me, constantly supporting and encoraging me, watching my steps, holding me never letting me fall. Ricardo, to you I dedicate this thesis.

Contents

1 Introduction: Thesis and Objective **1**

1.1 Problem Description 2

1.2 Thesis and Goal 4

1.3 Solution Approach 4

1.3.1 Fault Tolerant Framework for Wireless Sensor Networks . . . 4

1.3.2 Pattern Based Fault Isolation for WSNs Applied in Industrial Environments 5

1.4 Research Contribution 5

1.5 Structure and Contents 6

2 Problem Statement **9**

2.1 Enterprise Scenarios 11

2.1.1 Supply Chain Management 11

2.1.2 Environment, Health, and Safety 12

2.2 Effects of Wireless Sensor Network Failures in Business Processes . . 14

2.2.1 Environment Contamination and Risk of Human Life 15

2.2.2 Economic Losses 15

2.3 Scope of the Thesis 16

3 Research Foundations **19**

3.1 Models and Definitions 20

3.1.1 Wireless Sensor Networks 20

3.1.2 Fault Tolerant Industrial Systems 22

3.2 Wireless Sensor Network Faults in Industrial Applications 24

3.2.1 Sources of Faults in WSN Applications 25

3.2.1.1 Node Faults 26

3.2.1.2 Network Faults 27

3.2.1.3 Sink Faults . . . 28
3.2.2 Failure Types . . . 29
3.3 State of the Art in Fault Tolerance for Wireless Sensor Networks . . . 30
3.3.1 Fault Diagnosis Techniques . . . 30
3.3.1.1 Self Diagnosis . . . 31
3.3.1.2 Group Diagnosis . . . 32
3.3.1.3 Hierarchical Diagnosis . . . 34
3.3.2 Fault Recovery Techniques . . . 36
3.3.2.1 Active Replication in WSNs . . . 36
3.3.2.2 Passive Replication in WSNs . . . 37
3.3.3 Classification and Evaluation of Techniques . . . 40
3.4 State of the Art in Fault Diagnosis for Industrial Applications . . . 44
3.4.1 Binary Diagnostic Matrix . . . 44
3.4.2 Diagnostic Trees and Graphs . . . 45
3.4.3 Rules and Logic Functions . . . 46
3.4.4 Neural Networks . . . 47
3.4.5 Fuzzy Systems . . . 50
3.4.6 Classification and Evaluation of Techniques . . . 51
3.5 Summary . . . 52

4 Fault Tolerant Framework for Wireless Sensors Networks 55
4.1 Requirements . . . 56
4.2 Architecture . . . 59
4.2.1 Framework Layers . . . 61
4.2.1.1 Device Layer . . . 61
4.2.1.2 Platform Abstraction Layer . . . 61
4.2.1.3 Fault Management Layer . . . 62
4.2.2 Fault Diagnosis . . . 63
4.2.2.1 In-Network Fault Diagnosis . . . 63
4.2.2.2 Fault Detector . . . 63
4.2.2.3 Fault Isolator . . . 64
4.2.2.4 System State . . . 64
4.2.2.5 Component Interaction . . . 65

4.2.3 Fault Recovery Management 66
4.2.3.1 In-network Recovery 66
4.2.3.2 Decision Maker 67
4.2.3.3 Recovery 67
4.2.3.4 Mapper 68
4.2.3.5 Device Manager 69
4.2.3.6 Code Distribution Manager 69
4.2.3.7 Code Repository 69
4.2.3.8 Code Injector 70
4.2.3.9 Component Interaction 70
4.3 Summary 71

5 Pattern Fault Isolator for WSNs **75**
5.1 Failure Model 76
5.2 Crash Fault Analysis 76
5.3 Architecture 79
5.3.1 Choice of Techniques 80
5.3.2 Pattern Fault Isolation Process 81
5.3.3 Pattern Base 82
5.3.3.1 Message Throughput Distribution 84
5.3.3.2 Combination of Known Patterns 85
5.3.4 Membership Evaluator 89
5.3.4.1 Membership Functions 89
5.3.4.2 Threshold Definition: Gaussian Distribution Approximation 95
5.3.4.3 Considerations 96
5.3.5 Transient Analyzer 97
5.3.6 Failures Match Maker 99
5.3.7 Neural Classifier 101
5.4 Prediction Model 103
5.4.1 Membership Evaluator 103
5.4.2 Transient Analyzer 105
5.4.3 Failures Match Maker 106
5.4.4 Neural Classifier 106
5.5 Summary 106

6 Applications **109**

6.1 Application Trial: Research Lab Monitoring 110

6.1.1 Data Collection 111

6.1.2 Preliminary Data Analysis 112

6.2 Application II: Simulation 114

6.2.1 Existing Simulation Environments 114

6.2.2 FT-WiseNets Network Traffic Simulator 115

6.2.2.1 Topology Generator 116

6.2.2.2 Failures Generator 119

6.2.2.3 Traffic Generator 121

6.3 FT-WiseNets Monitoring Application 122

6.3.1 Fault Isolator Implementation 123

6.3.2 FT-WiseNets User Interface 125

6.3.3 Pattern Acquisition View 126

6.3.4 Pattern Combination View 127

6.3.5 Online Isolation Views 127

7 Pattern Based Fault Isolator Evaluation **131**

7.1 Simulation Procedures 131

7.1.1 Membership Evaluator 131

7.1.1.1 Threshold Definition 132

7.1.1.2 Timeframe 138

7.1.2 Pattern Combination 141

7.1.3 Transient Analyzer 142

7.1.4 Failures Match Maker 150

7.1.5 Neural classifier 154

7.2 Application trial results 156

7.2.1 Membership Evaluator 157

7.2.2 Failures Match Maker 158

7.2.3 Neural Classifier 162

7.3 Evaluation Results Discussion 164

8 Conclusions **167**

8.1 Future Work 169

A Message Exchange **171**

A.1 Message Handler 172

A.2 Notification Manager 173

A.3 Request Processor 174

A.4 Component Interaction 174

B Pattern Combination **179**

Bibliography **183**

List of Figures

1.1 Application scenario. 3

2.1 Potential value of introducing WSNs in industrial applications. 10
2.2 Wireless sensor nodes applied in quality control. 12
2.3 Application trial at a chemical plant where storage regulations of hazardous materials were monitored. 13
2.4 Effects of failures in WSNs applied in business processes. 14

3.1 Wireless sensor network model. 21
3.2 Fault diagnosis process for industrial control systems. 23
3.3 Failure caused by a loosely connected sensor. 25
3.4 Fault classification and propagation. 26
3.5 Self diagnosis in WSNs. 31
3.6 Group diagnosis in WSNs. 33
3.7 Hierarchical diagnosis in WSNs. 35
3.8 Steps to recover from a failure in a passive replication mode. 38
3.9 Graphical representation of a binary diagnostic matrix. 45
3.10 Diagnostic graph. 46
3.11 Failure diagnosis rules for an electric mining car. 47
3.12 Neural network model, from [128]. 49
3.13 Division of variable x into fuzzy regions and the corresponding membership functions, from [128]. 50

4.1 Application trial at a chemical plant. 57
4.2 FT-WiseNets: Fault Tolerant Framework for Wireless Sensors Networks. 60
4.3 Components interaction: fault diagnosis. 66
4.4 Component interaction: fault recovery. 71

5.1 Fault classification. 78

5.2 Pattern Fault Isolator. 80
5.3 Fault isolation process. 83
5.4 Message throughput histogram of a sensor node. 84
5.5 Sensor node message throughput distribution during normal operation and faulty state. 87
5.6 Influence including negative values. 88
5.7 Sensor node message throughput distribution when rounding the influence to zero. 88
5.8 Distribution of P. 90
5.9 Curve level view of P. 90
5.10 Normalized euclidean distance curves. 93
5.11 Curves level view of P and normalized euclidean distance curves. . . . 95
5.12 Normalized euclidean distance histogram of a fault pattern. 95
5.13 Cumulative density function of D_E. 96
5.14 Euclidean distances between the mean of sink failure patterns. 97
5.15 Transient state for a sudden crash failure. 98
5.16 Transient state for a link failure. 99
5.17 Diagnostic signal. 100
5.18 Minimum vectors of three different sink failures. 100
5.19 Unexpected observed failure vectors for sink failures S1, S2 and S3. . 101
5.20 Neural Classifier. 102

6.1 Nodes deployed in the research facility. 110
6.2 Placement of nodes during the application trial. 110
6.3 Number of events per node. 113
6.4 Fault balance for the period of 2 months. 113
6.5 Topology scheme in the FT-WiseNets Network Traffic Simulator. . . . 116
6.6 Topology generation view in the FT-WiseNets Network Traffic Simulator. 117
6.7 Failure percentage view in the FT-WiseNets Network Traffic Simulator. 120
6.8 Area of interference failure view in the FT-WiseNets Network Traffic Simulator. 121
6.9 Database scheme. 124
6.10 Pattern acquisition view. 126

6.11 Pattern combination view. 127

6.12 Online isolation view of the Membership Evaluator. 128

6.13 Online isolation view of the Failures Match Maker. 129

6.14 Online isolation view of the Neural Classifier. 129

7.1 Pre-processing procedure improves the normal distribution approximation. 134

7.2 Normalized euclidean distance distribution for each pattern analyzed. 135

7.3 Influence of threshold and number of pattern readings on fault isolation effectiveness. With more pattern readings the FIE approaches the "Ideal" value. 135

7.4 Influence of threshold and number of pattern readings on fault isolation effectiveness. Membership function behaves in the same manner for networks with and without overlap. 136

7.5 Diagnostic signal distribution for different topologies setups. There is an increase in the number of outlier readings as the overlap raises. . . 137

7.6 Node message throughput distribution with different timeframe setups. The distribution is better approximated by a Gaussian curve as the timeframe increases. 140

7.7 Diagnostic signal distribution for different topology setups. The number of outlier readings reduces as the timeframe increases. 140

7.8 Comparison between calculated and acquired mean values. 143

7.9 Comparison between calculated and acquired variance values. 143

7.10 Error between calculated and acquired mean and variance values. . . 143

7.11 Fault isolation effectiveness for link failures. The FIE increases as the interference raises, until the point where misdiagnosis start to occur. . 145

7.12 Misdiagnosis: link failure diagnosed as node failure. A higher number of misdiagnosis occur as the interference increases. 146

7.13 Link failure false positives. There is a high number of false positives for a threshold of 1std. 146

7.14 Fault isolation effectiveness for node failure. 148

7.15 Misdiagnosis: node failure diagnosed as link failure. 148

7.16 Node failure false positives. 149

7.17 Correct isolation of healthy nodes. 149

7.18 Percentage of total attempts where no matches were found by the Failures Match Maker. 151

7.19 Failures Match Maker results for sink failure isolation performance. Results confirm Prediction 5. 152

7.20 Failures Match Maker results for node failure isolation performance. . 152

7.21 Failures Match Maker results for link failure isolation performance. There is a high number of false positives with 1std as the threshold. . 153

7.22 Total number of neural classifications. 155

7.23 Neural Classification results. 156

7.24 Percentage of sink failures correctly identified by the Membership Evaluator. 158

7.25 Percentage of sink failures false positives accused by the Membership Evaluator. The number of FP decreases after reaching a "peak", because the diagnostic signal starts to be accepted by its own pattern. 159

7.26 Membership Evaluator results for overlapping sink failures. 159

7.27 Failures Match Maker results for sink isolation effectiveness. The results of this test confirm Prediction 5. 160

7.28 Failures Match Maker results for link failure false positives. There is a reduction on the number of false positives as the threshold value increases. 161

7.29 Failures Match Maker results for node failure false positives. After 5std the number of FP increases because the Failures Match Maker also starts to accept the diagnostic signal as part of incorrect sink failure patterns. 161

7.30 Neural Classifier effectiveness. Lower performance of the configuration TF=135s and TS=45s is due to stronger a overlap of a sink failure and the system in normal operation. 162

7.31 Incorrect classification of sink failures by the Neural Classifier. 163

7.32 Even classification of sink failures by the Neural Classifier. The diagnostic signals are not "recovered" for large thresholds, and get further away from the values used to train the network. 163

A.1 Components interaction: notification. 175

A.2 Components interaction: invocation. 176

B.1 Comparison between calculated and acquired mean values. 180

B.2 Comparison between calculated and acquired variance values. 180

B.3 Error between calculated and acquired mean and variance values. . . 180

B.4 Comparison between calculated and acquired mean values. 181

B.5 Comparison between calculated and acquired variance values. 181

B.6 Error between calculated and acquired mean and variance values. . . 181

Glossary

B2B Business to Business.

B2C Business to Customer.

CoBIs Collaborative Business Items.

EH & S Environment Health and Safety.

ERP Enterprise Resource Planning Systems.

FIE Fault Isolation Effectiveness.

FP False Positive.

FT-WiseNets Fault Tolerant Framework for Wireless Sensor Networks.

GUI Graphical User Interface.

KPI Key Performance Indicators.

LAN Local Area Network.

RFID Radio-Frequency IDentification.

RPC Remote Procedure Calls.

SOA Service Oriented Architectures.

SOAP Simple Object Access Protocol.

WSN Wireless Sensor Network.

1. Introduction: Thesis and Objective

Pushed by the market and competitors, modern enterprise information systems seek innovative solutions that can be adopted to drastically improve business processes. Currently, Radio-Frequency IDentification (RFID) is used to enhance business processes in a broad range of industrial applications [94, 138, 120], such as supply chain management, theft prevention and automatic payment. However, due to the hardware limitations of RFID, this technology is usually only applied to item tagging for identification and localization.

To overcome the restrictions of RFID, a next generation technology for environmental awareness is emerging to bring enterprise systems to a new level of business process management [4]. This new technological concept is called wireless sensor network (WSN), and is composed of wireless sensor nodes, which are devices that have sensors, computational capabilities, are battery powered and have the ability to communicate with surrounding nodes in a peer-to-peer fashion [41].

Due to their tight integration with the environment and the possibility of attaching them to physical items, WSNs reduce the information gap that exists between the real world and its digital representation, enabling a new paradigm of business process execution. In this new paradigm, the rules that govern the business process (called business logic) is distributed and executed on the devices, at the "point of action" [33].

The advantages that WSNs have on enterprise applications includes, for instance, increase in the scalability of systems, and reduction on the information load that must be processed by the back-end. A prime example of the use of WSNs applied

in industrial environments was introduced in the CoBIs project [26]. There, WSNs were applied to prevent drums containing hazardous chemicals from violating storage regulations. In this scenario business rules were pushed to wireless sensor nodes attached to the drums and were locally executed.

However, regardless of the reliability of single elements present in WSNs, faults in the respective components always occur. Hence, malfunctions and break-down states of WSNs applied in business processes must be efficiently diagnosed and recovered to reduce economic losses.

Our research is supported by several real world WSN deployments. We conducted an investigation on frequent faults that occur on those deployments [30]. As indicated by deployment reports [110, 163, 93, 168, 159], the installation of large-scale sensor networks for real world applications is not a trivial task and can lead to numerous failures. What works in theory does not always perform as expected in practice.

WSNs may fail due to various reasons, including radio interference, battery exhaustion, software bugs, or the dislocation of nodes. Such failures are caused by software and hardware faults, environmental conditions, malicious behaviour, or bad timing of a legitimate action. In some cases, a failure caused by a simple software bug can be propagated to become a failure of the entire WSN.

In general, the consequence of such an event is that nodes become unreachable or violate certain conditions that are essential for executing the business process. Hence, the successful deployment of WSNs in business processes relies on efficient fault diagnostic systems to support maintenance workers in the correct isolation of failures, in order to reduce maintenance time and costs.

In this thesis, we approach the problem of efficiently diagnosing faults in heterogeneous WSNs applied in business processes.

1.1 Problem Description

Enterprise business systems support a great variety of business processes and are therefore required to work with a broad range of wireless sensor network platforms. As a consequence, enterprise business systems cannot be restricted to one hardware platform. This also implies that specific functionalities may not be available on all platforms.

These restrictions also apply to the isolation of failures performed by the back-end system. Therefore, we tackle the research question of how to identify malfunctioning components in heterogeneous WSN based only on a basic information set that can

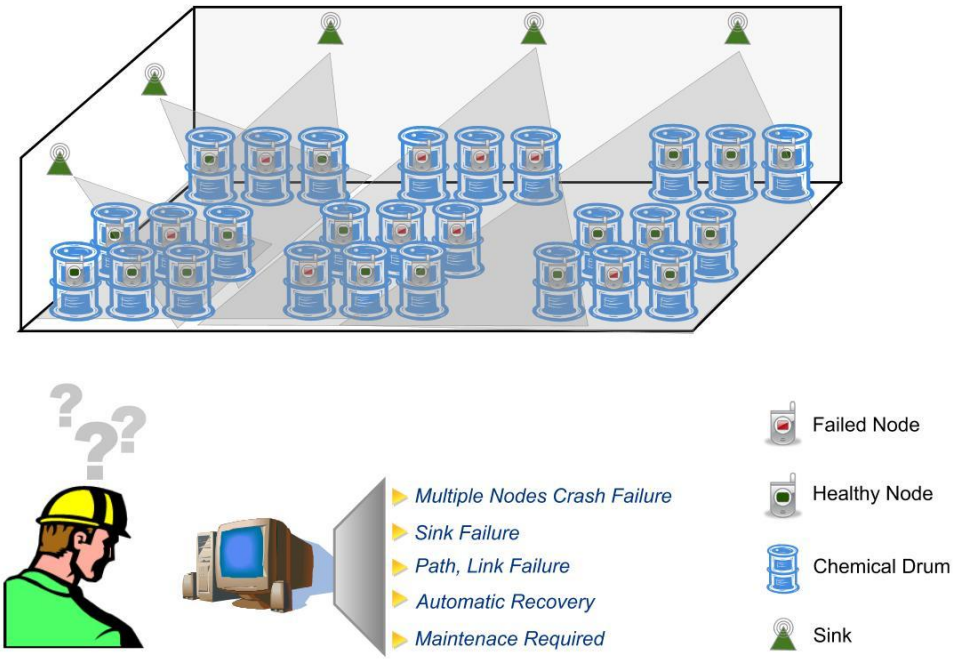


Figure 1.1: Application scenario.

be provided by any WSN regardless of its specific implementation. This inherently forms an appropriate WSN platform abstraction for enterprise business systems. This restricted information set is based on sensor node identification number, timestamp information, and the time interval between messages (heart beat).

Structural information, such as routing topology, is also linked to the functionality that a WSN may offer. Therefore it is hard to generically exploit structural information for fault diagnosis by enterprise systems. This restricted set of information makes the task of isolating failures extremely challenging.

Figure 1.1 depicts the CoBIs scenario, where several chemical drums have sensor nodes attached to them to monitor storage regulations. In this scenario, devices called sinks provide a coverage area, which enables them to gather the data generated by sensor nodes in their region. As demonstrated in this figure, the identification of failed components of a WSN, applied in a business process, based only on timeout evaluation will lead the system to masquerade failures of several components as only node failures. Sinks, when malfunctioning will not forward the data of several nodes to the back-end. Link and path failures can be confused with crash failures, and the composition of sink and node failures will only appear as node failures.

Additionally, due to the low cost associated with these devices, it is possible to imagine the deployment of large-scale WSNs with potentially thousands of nodes [41]. Without the support of a system that analyzes the great amount of data

generated by WSNs, the isolation process becomes cumbersome and time consuming. This results in increased effort to maintain the system and reduces the chances of mitigating the effects of the failure in the business process.

In this context, this thesis investigates the research question of how feasible it is to integrate heterogeneous WSNs platforms with enterprise business systems and perform fault isolation with a minimum, but for most platforms, generic information set.

1.2 Thesis and Goal

In order to address the problem of integrating heterogeneous hardware platforms with enterprise business systems, we propose:

Thesis: The isolation of failed components in heterogeneous WSNs platforms applied in business processes can be achieved based on a restricted information set: sensor node identification number, timestamp information, and heart beat interval.

Goal 1: To enable the integration of heterogeneous WSNs with enterprise applications.

Goal 2: To identify failures occurring in heterogeneous WSNs applied in business processes.

1.3 Solution Approach

Since WSNs are emerging as the next generation technology that will bring enterprise systems to a new level of business process management, and since they require the support of a broad spectrum of platforms, the integration of reliable heterogeneous WSNs with enterprise systems becomes a crucial point for enabling the deployment of WSNs in industrial applications.

In order to achieve the required integration, which takes fault management into account, this thesis will focus on addressing *(1)* the integration of heterogeneous hardware platforms with enterprise applications and *(2)* the isolation of failures in WSNs based on a restricted set of information.

1.3.1 Fault Tolerant Framework for Wireless Sensor Networks

For achieving an integration of heterogeneous WSN hardware platforms with enterprise business systems, this thesis proposes the Fault Tolerant Framework for

Wireless Sensor Networks (FT-WiseNets), a framework that embraces the tendencies defined by enterprise applications [89, 127, 82], such as Service Oriented Architectures (SOA) based on Web Services. The key idea of this framework is to define a layered architecture that creates a hardware platform abstraction for enterprise applications. As a result, new platforms can be easily integrated and accessed by business systems in a transparent way, achieving the goal of integrating heterogeneous hardware platforms.

This framework not only provides a transparent access for business systems to WSNs, but also defines mechanisms for fault diagnosis and recovery. These mechanisms aim at monitoring the network of devices and automatically recovering failures or triggering maintenance workflows when necessary. In addition, due to the utilization of web services and well defined interfaces, the framework is flexible and extendable. This enables the adoption of new fault diagnosis and recovery algorithms and techniques.

1.3.2 Pattern Based Fault Isolation for WSNs Applied in Industrial Environments

In order to demonstrate that fault isolation based on a restricted information set is achievable, we developed a novel approach that enables such analysis. This approach seeks to learn system patterns, during normal and faulty states. Once patterns have been learned by the system, pattern recognition techniques are applied during runtime to isolate system failures. Based on neural networks and statistics, this solution explores the benefits of each technique, which enables a better performance of the isolation process.

Using this technique, the faults that occur in WSNs can be efficiently identified, which increases the chances of mitigating their effects on the business process. For example, in a real application trial, the proposed approach correctly identified failures in 98% of the cases in a single-hop network. In addition, the proposed approach introduces a technique to reduce the number of patterns that must be acquired during the learning phase. This reduction of acquired patterns is due to the combination of already known system patterns.

1.4 Research Contribution

The work presented in this thesis is the result of experiences gathered in the CoBIs EU funded project [26], where real industrial scenarios made use of WSNs. In this project, fault management was not the main focus of the research; nevertheless, the

need for reliable WSNs became evident during the course of industrial application trials. Hence, we searched for mechanisms that can increase the reliability of WSNs applied in business processes, seeking to mitigate the effects of failures.

In a nutshell, this thesis makes the following contributions to fault management research applied in Wireless Sensor Networks and computer science:

1. FT-WiseNets, a framework for integrating WSNs with enterprise applications, including a flexible and extendable approach for fault management.

2. An analysis and classification of crash failures that are observed and their underlying causes.

3. A novel fault isolation method, based on pattern recognition techniques, which relies only on a restricted information set: sensor node identification number, timestamp information, and heart beat interval.

4. Application and evaluation of the FT-WiseNets framework, and of the proposed fault isolation approach.

1.5 Structure and Contents

One of the goals of this thesis is to integrate heterogeneous WSN platforms with enterprise applications to increase the value of business processes. By adopting fault management methods, enterprise systems can mitigate the effects of failures occurring in the networked devices.

In Chapter 1, we introduce the motivation of our thesis and present our contributions to fault management for WSNs applied in business processes.

In Chapter 2, we describe the benefits of integrating WSNs in industrial scenarios and analyze the effects of failures occurring in WSN.

The research foundations for this thesis are introduced in Chapter 3. This chapter presents models and definitions, and a discussion of the related work on fault tolerance in WSNs and fault diagnosis in industrial applications.

The FT-WiseNets framework is depicted in Chapter 4. The requirements for a framework that integrates WSNs with enterprise applications are discussed based on a real world application trial. Then all components of the framework and their interactions are presented.

In Chapter 5, a novel method for isolating failures in WSNs applied in business processes based only on a restricted information set is presented. A crash fault analysis is introduced to emphasize the challenge of isolating failures under the described restrictions.

The evaluation of the framework and the pattern fault isolator is described in Chapters 6 and 7. In these chapters, the setup of a real application trial, the lessons learned and the results are discussed.

Chapter 8 discusses the main contributions, the already visible impacts and the outlook of future research directions.

2. Problem Statement

Wireless sensor networks are emerging as a novel approach to support business processes in large-scale enterprise environments, which involves physical entities, such as goods and tools [5]. Software systems that provide various services for enterprise businesses are usually based on: highly decentralized, manual and often error-prone data collection, centralized data storage and business logic execution in so called "back-end" systems.

An important advancement that this new WSN technology provides is the possibility of distributing business logic functionality to sensor nodes. In this way, the virtual state of enterprises, as represented in business processes and in the supporting enterprise software systems, can reflect to a greater extent what is actually happening in the real world. Since wireless sensor nodes are considered to be much "smarter" than items tagged with RFID transponders, they can play a more active role in business processes [153].

Processes within a given operational environment can yield significantly better results, by relocating well-defined parts of business logic functionality (i.e. process execution), from resource intensive backend systems to relatively low cost networked embedded systems that run "where the action is". In our case, these systems use sensor network technology, which enables them to build collaborating "teams" that work together, for certain process relevant results [107, 153].

Relocating business logic to WSNs have the potential of adding value to industrial applications [33]. These aspects are illustrated in Figure 2.1 and are discussed below:

- Information gap reduction: Many benefits arise from improving the accuracy of data that represents the current status of a business process. These benefits

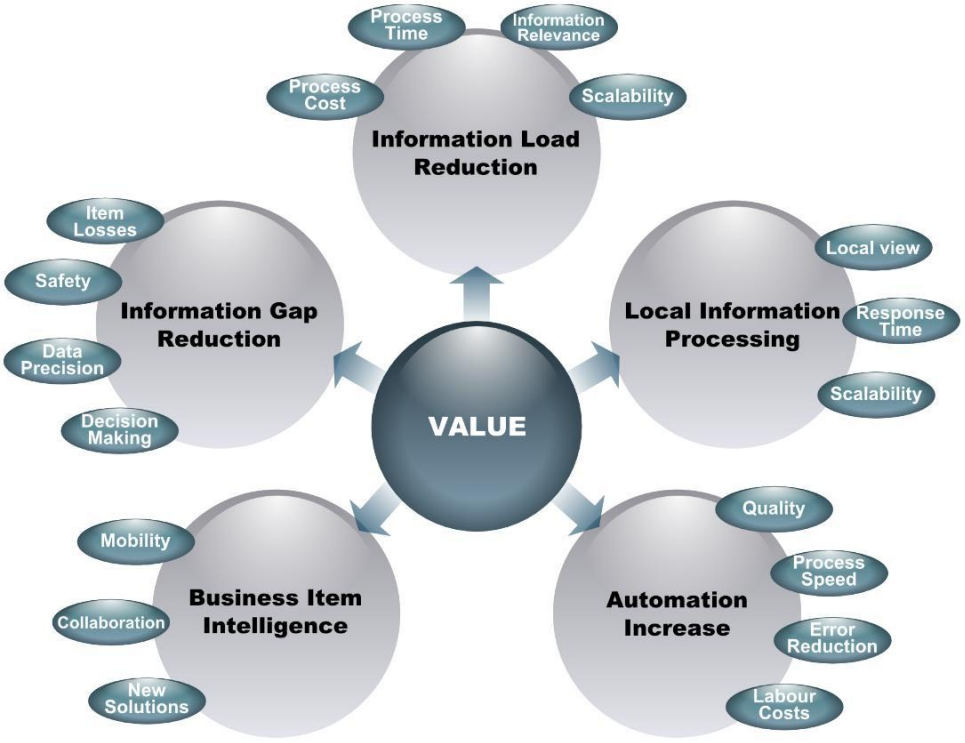


Figure 2.1: Potential value of introducing WSNs in industrial applications.

include the possibility of reducing losses of items, precise detection of hazardous situations (e.g. in safety-critical applications), and improving the decision making processes due to the availability of detailed information.

- Information load reduction: The use of WSNs enables low level processing of data, reducing the amount of data that has to be processed by the back-end to a small set of relevant information. The information load reduction has the potential to: reduce process execution and transactional costs, improve response times in business- and/or safety-critical situations, and to have a better scalability of the system.

- Automation increase: With devices capable of collaborating and executing business logic, it is possible to conceive processes with less human interaction. This leads to: error reductions, labour cost reduction, and increased process quality and speed.

- Business item intelligence: The use of WSNs in industrial applications enables a new paradigm of business process execution. With WSNs, processes become more flexible and mobile since nodes can collaborate with each other and the process can be executed anywhere the items are located.

- Local information processing: Wireless sensor nodes have the advantage that they are located at the "point of action", which enables them to process most events and information locally. This technology enables access to local information, which generates a local view of the current status of the system without the need of accessing a back-end system, i.e. by using a mobile device capable of capturing the wireless packages generated by sensor nodes. This leads to: more flexibility in the shop floor, better scalability of the system, and new approaches for handling business processes supervision.

In industrial scenarios, where WSNs technology is applied, the functionalities provided by WSNs must be reliable to avoid major losses in the business processes caused by failures in the sensor network. Therefore, it is necessary to analyze the applicability of WSNs in business processes, and the possible failures and effects that can result from the adoption of this technology [30].

2.1 Enterprise Scenarios

The benefits of applying WSNs in business processes make this technology attractive for industrial applications, including a variety of application domains (e.g. supply chain management [165], environment health and safety management) [107, 153]. Although the applicability of this technology covers a large spectrum of industrial domains, most of them share the need to improve their business processes, by integrating the information collected from the WSNs with Enterprise Resource Planning Systems (ERP). In this subchapter we will analyze two industrial scenarios where WSNs are applied in order to identify aspects where business processes can be improved.

2.1.1 Supply Chain Management

Supply chain management is an application domain that can profit from the adoption of WSNs. In [165] the authors propose an approach to use sensor nodes to monitor the products' storage conditions during transportation.

This scenario addresses Business to Business (B2B) and Business to Customer (B2C) transactions, where sensitive goods (e.g. food and chemicals) have to be delivered. In such scenarios, restrictions and handling requirements are key factors for quality control and for guaranteeing the quality properties of goods.

When the customer receives the goods, the customer must verify if the products have been properly handled and must define the validity period of the material. If

the restrictions are violated the validity period of the product has to be reduced, or, depending on the product and the violated condition, the product must be disposed.

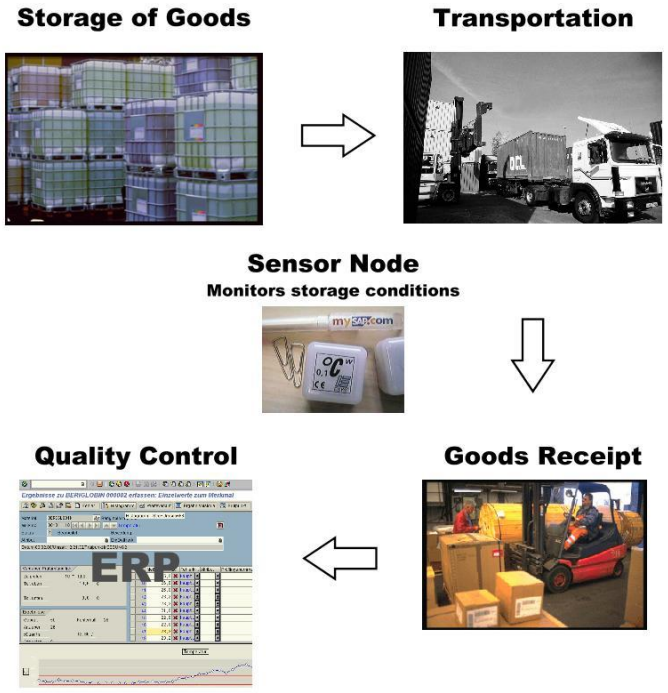


Figure 2.2: Wireless sensor nodes applied in quality control.

By inserting wireless sensor nodes in supply chain management, the storage and transport conditions of goods can be constantly monitored, as depicted in Figure 2.2. Goods are equipped with sensor nodes that constantly monitor the conditions in which the materials are stored and transported. Upon arrival the recorded data is automatically transferred to the ERP system where the quality of the material will be evaluated to verify if it is in a good condition to enter the manufacturing process or if it has to be discarded.

Many companies still apply old techniques for quality management like manual collection of data, based on analogous thermometers and paper plots. This procedure has several disadvantages, e.g. it is time consuming and prone to errors due to human interaction. Wireless sensor networks in this scenario can improve the automation of the process, while reducing errors and making the process more reliable and efficient. The use of business logic on the item can also help reduce the information overload by only propagating storage violation events to the back-end.

2.1.2 Environment, Health, and Safety

Every year, the U.S. Department of Labor [90] accounts for numerous occupational accidents in the field of oil and gas industry (e.g. in 2004, 52,830 nonfatal injuries

and 29 fatalities due to the exposure to harmful substances or environments have been recorded) [107].

WSNs represent a viable solution to these problems. Sensor nodes placed in chemical storage areas can collaboratively determine potential hazardous situations, and alert or take an action at the point of interest [156]. In addition, WSN technology can prevent errors in the manipulation and storage of chemical containers, thus leading to increased safety and reduced logistic costs.

These ideas are extended in the EU-funded project Collaborative Business Items (CoBIs) [26], by designing and implementing a distributed, service-oriented enterprise system, which incorporates the latest advances in WSN technology.

Figure 2.3: Application trial at a chemical plant where storage regulations of hazardous materials were monitored.

In the scenario investigated by this project, chemical handling regulations are stored and maintained in an ERP system and evaluated by wireless sensor nodes. Field tests have been carried out at the BP premises in Hull UK (see Fig.2.3), where the following use cases have been identified:

1. *Storage and manipulation of hazardous substances.* Chemical containers storing reactive substances must be handled according to a strict list of safety regulations. The following situations are to be avoided:

   (a) Incompatible substances in close proximity of each other

   (b) Total mass of a reactive substance stored in an area exceeding a maximum threshold

   (c) Hazardous substances stored in a protected area longer than a specified time period

2. *Continuous monitoring of environmental conditions.* In the case of hazardous substances, environmental parameters, such as temperature, humidity and light, should be continuously monitored and abnormal conditions should trigger immediate alarms and local actions.

In this scenario, storage regulations were modified on the ERP system and propagated in the form of rules to wireless sensor nodes. These nodes were attached to drums containing hazardous materials and programmed with information on their specific content, both chemical composition and volume, which enabled them to locally evaluate if storage regulations were being violated.

This scenario represents an application where WSNs not only improve safety of business processes by reducing the information gap between real world and digital information, but also make them more efficient by executing the business logic at the point of action.

2.2 Effects of Wireless Sensor Network Failures in Business Processes

The harsh environment in which WSNs are deployed make them susceptible to failures occurring in several layers of the system (analyzed in section 3.2.1). When applying this technology in industrial settings, special attention has to be taken on the reliability of these devices.

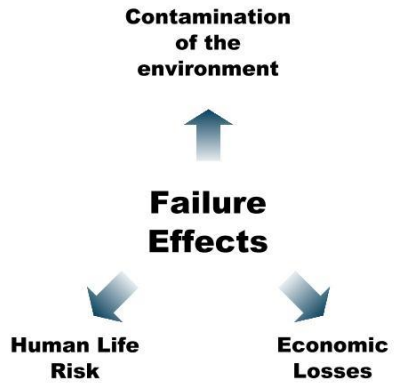


Figure 2.4: Effects of failures in WSNs applied in business processes.

Figure 2.4 presents the main set of effects that failures in WSNs can cause [84]. The effects of failures occurring within WSNs are closely related to the scenario in which it is applied and can range from financial losses to the contamination of the environment and the risk of human life.

2.2.1 Environment Contamination and Risk of Human Life

Safety in chemical industries aim at preventing notable accidents involving the contamination of the environment, as the ones listed below [8]:

- the 1975 Breek propylene release and refinery fire in which 14 people were killed.
- the 1976 Seveso accident in which highly toxic substances were released in the environment, causing the contamination of wide areas with consequent health implications for the surrounding population
- the chemical spill at Basel in 1986, where the Rhein River was severely contaminated.

Such accidents and others of lesser magnitude tend to repeat themselves due to the lack of information dissemination [121, 25]. These incidents have severe impacts on the industry. According to reports [38, 123], the most frequent accidents in the chemical industry are related to fire (41% frequency and 20% of financial loss) and explosions (23% frequency and 63% of financial loss).

In the CoBIs scenario presented in section 2.1.2, wireless sensor nodes should help reduce the risk of explosions and fire by preventing incompatible chemicals from being stored in the same location. Therefore, the use of WSNs in this scenario can reduce financial losses, human life risk and the contamination of the environment caused by accidents. Although this is a great improvement on the business process, this technology must be highly reliable when applied in safety-critical scenarios, since unidentified failures can lead to harsh accidents.

When a node becomes inactive due to a crash failure, it fails to advertise its content and to locally execute business logic. In such situations the node will not identify the presence of incompatible chemicals at the same location. A value failure on the chemical type message can also lead to an incorrect execution of the business logic. In this situation, the WSN will fail to recognize potentially hazardous situations or will trigger unnecessary alerts. Hence, failures in WSNs applied in safety-critical environments, as proposed by the CoBIs project, can lead to accidents where there is a potential for contamination of the environment and a risk to human life.

2.2.2 Economic Losses

In scenarios where safety-critical accidents are unlikely to occur, failures in the sensor nodes will usually lead to economic losses. While the cost of wireless sensor

nodes may be small, the economic losses caused by failures in these devices can be financially ruinous.

Losses related to maintenance can become very high, representing a major part of the total operational cost of all manufacturing or production plants. According to the industry sector, maintenance cost can represent between 15% and 60% of the cost of goods produced [117]. These costs are also related to the practices applied for maintenance. In [118], the authors present a comparative study performed with 250 small and medium manufacturers in Australia. There, different maintenance practices where adopted resulting in a difference of 53% in maintenance costs.

Failures usually require maintenance to bring the system back to normal operational status. Nevertheless, maintenance is not the only cost associated to malfunctions, which also consist of numerous factors including [117]:

- Halts in production
- Equipment repairs
- Product recalls
- Production time
- Reputation loss
- Material loss

In the transportation scenario described in section 2.1.1, a sensor node crash failure will prevent customers from verifying the quality of goods. This can lead to product quality reduction, material disposal, and even production halt until new material is provided by the supplier. Additionally, if a sensor node fails to record correct sensor reading, generating value failures, high quality material can be disposed, or worse, low quality material can be integrated in the manufacturing process of a customer, which can cause loss of business due to lost reputation.

2.3 Scope of the Thesis

Wireless sensor networks have a remarkable potential to improve business processes, as described in this chapter. Nevertheless, the integration of WSNs with real enterprise scenarios requires careful analysis of the impacts it can cause in the business process. Since it cannot be assumed that all sources of error can be eliminated,

even through careful engineering, the availability of the functionalities provided by a WSN depends to a large extent on fault diagnosis and recovery.

This PhD thesis is focused on providing mechanisms for reliably integrating WSNs with enterprise applications. These mechanisms include support of failure diagnosis to prevent failures occurring in the WSN from propagating to the back-end system. As core contribution this thesis proposes a mechanism for isolating crash failures in heterogeneous WSNs.

3. Research Foundations

The sensing capabilities of wireless sensor nodes together with software configurability, wireless communication and flexibility enable a new paradigm of solutions in many application domains, e.g. supply chain management [165], and environment health and safety management [107, 153], as described in section 2.1.

The characteristics that make WSNs interesting also make them vulnerable. As discussed in chapter 2, WSNs are subject to several failures, e.g. the wireless communication channel can suffer interference, software may contain errors and the failure of single components can masquerade malfunctions of other system parts.

Enterprise applications require a high degree of availability of subsystems in order to maintain a reliable and efficient operation of back-end services. In such environments, fault tolerance techniques must be applied to ensure a successful deployment of WSNs in business processes.

While fault diagnosis in industrial applications is a well established field of research, the existing work in the area of fault diagnosis and recovery in WSNs is scattered in different research domains, ranging from self-healing routing protocols to sensor fusion algorithms.

In this chapter we provide models and definitions used throughout this thesis. A systematic overview and assessment of known fault diagnosis and recovery techniques for WSNs and industrial applications is also presented. The purpose is to improve the understanding of the effectiveness of these techniques in the WSN domain, and to provide guidance for selecting appropriate measures when designing systems.

3.1 Models and Definitions

The contents of this section aims at facilitating the comprehension of the research contributions of this thesis and the current state of the art in fault tolerant systems. Therefore, it provides models and definitions for wireless sensor networks and fault tolerance in industrial systems.

3.1.1 Wireless Sensor Networks

A WSN is a system of *sinks* and small devices called *sensor nodes*. Sensor nodes have limited memory, processing power, energy resources, and also have the ability to communicate via wireless channels [1].

Each *sensor node* is equipped with multiple components [57]: a microprocessor, communication and storage subsystem, a battery supply, and sensors. Due to the low cost associated to these devices, it is possible to deploy large-scale WSNs with potentially thousands of nodes [41].

A *sink* is a device that has the capability to bridge the communication between wireless sensor nodes and the back-end systems.

Each node is in principle able to receive data from the sink and route data back to it. The routing is done based on multi-hop communication, i.e, using other sensor nodes to forward messages to the sink. In *continuous WSNs*, nodes periodically report their readings to the back-end on a regular time interval called *heart beat*. In contrast, *event-driven WSNs* only report their reading to the back-end when a monitored value has changed above a given threshold.

A WSN can be modeled in order to represent the main parts of the system that are subject to failures. Figure 3.1 depicts a generic model we propose for the analysis of such systems.

In this model the WSN is composed of a set $\{N\}$ of η nodes and of a set $\{B\}$ of w sinks.

Nodes directly connected (e.g. n3 and n4) are called neighbour nodes and are often used as a source of information in fault diagnosis methods for WSNs. The neighbourhood function NB indicates the nodes that are directly connected to a node, e.g. $NB\,(n9) = \{n7, n10, n11\}$.

The coverage area of a sink β_i is defined as $\delta_{\beta i}$ and indicates the set of nodes n that are under the reachable area of the sink, $\delta_{\beta i} = \{N\}_{\beta_i}$.

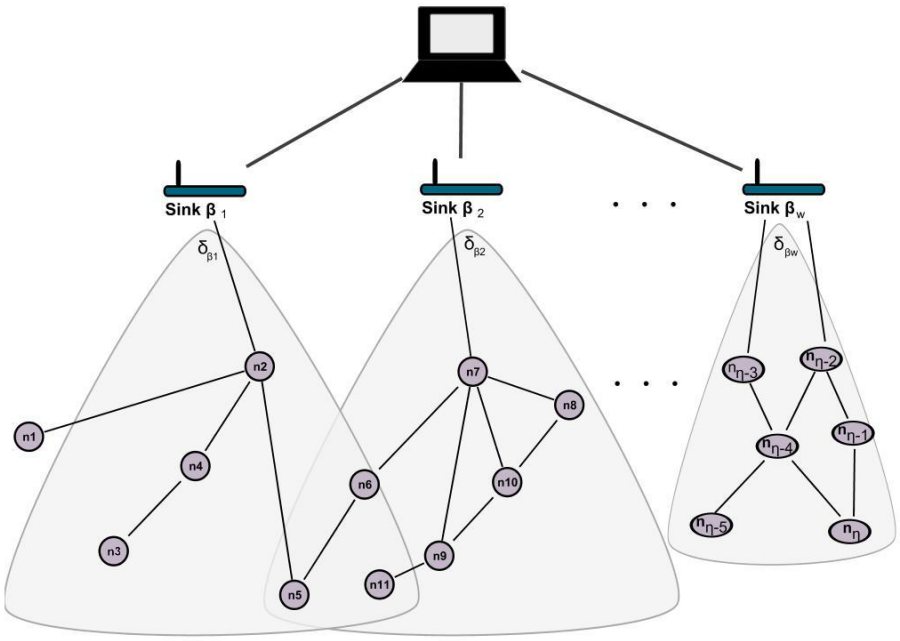


Figure 3.1: Wireless sensor network model.

Although the coverage of a sink ($\delta_{\beta i}$) can include several nodes, the set of nodes that actually report only to this sink ($\Phi_{\beta i}$) can differ from $\delta_{\beta i}$ due to overlaps between coverage areas and routing paths, as demonstrated by nodes $\{n_1, n_5, n_6, n_{11}\}$ in Figure 3.1.

The set of nodes reporting only to sink β_i is defined as $\Phi(\beta i)$, while the size of this set is defined as $\varphi(\beta i)$. In the example presented in Figure 3.1, $\Phi(\beta_1) = \{n_1, n_2, n_3, n_4\}$. The set of nodes reporting to sink β_i and β_j is defined as $\Phi(\beta_i, \beta_j)$. For sinks β_1 and β_2, $\Phi(\beta_1, \beta_2) = \{n_5\}$.

As different network topologies exist we use a multipath routing topology for our model. This choice is based on the idea that all other topologies can be considered as subsets of a fully connected network. Defining the topology of the network is especially important since, according to the available number of paths, the failure of a single component can cause several other nodes to also appear as failed. Hence, we define the set of paths that connects two points a and b as: $P_b^a = \{\rho 1_b^a, \rho 2_b^a, ..., \rho \xi_b^a\}$. Where ξ_b^a indicates the number of paths from point a to point b. The quality of path $\rho 1_b^a$ is defined as $\zeta 1_b^a$ and indicates the probability of a message reaching the destination. The path quality varies from a scale of 0 - 100 where 0 indicates a broken path while 100 indicates an end to end connection where no messages are lost.

It is also important to identify the number of nodes that rely on specific sensors as part of a unique route path (λ_{ni}), such as sensors n_2 and n_6 from Figure 3.1. In

this case the number of nodes relying on n_2 (λ_{n2}) is four (since n_5 has an alternative route), λ_{n6} is one and $\lambda\,(n2, n6)$ is six. The set of nodes that rely on specific sensors as part of a unique route path is defined as $\Lambda\,(n2, n6)$, in our example $\Lambda_{(n2,n6)} = \{n_1, n_2, n_3, n_4, n_5, n_6\}$. Following the same approach, the set of nodes that rely on specific routing paths is defined as $\Lambda\left(P^B_{n2}E, P^B_{n6}E\right) = \{n_1, n_2, n_3, n_4, n_5, n_6\}$.

3.1.2 Fault Tolerant Industrial Systems

To be considered *fault tolerant*, a system must provide resilience to faults, which prevents the system from reaching a state where failures are in place. Fault tolerant systems usually have a degree of faults that can be tolerated before the system reaches a failure state.

A fault tolerant system must perform two main tasks: *fault diagnosis* and *fault recovery*.

The process of enhancing the reliability of any system starts with the ability of identifying incorrect behaviours of the system's parts. This process is called 'Fault Diagnosis'. The diagnosis of processes can be defined as: *"... the method for determining the nature of a system disorder distinguishing it from other possible conditions"* [84]. The classification of models applied to the diagnostic processes, distinguishes models of systems applied to *fault detection* and those applied to *fault isolation*. Models used for fault detection allow detecting changes (symptoms) caused by faults. Models used for fault isolation map the symptoms to the failures that occurred in the system.

For example, a fault detection algorithm based on timeout can provide the information that numerous nodes have suddenly become silent, while a fault isolation method can analyze the data and indicate that a bridge is malfunctioning.

After the system has detected a fault, the next step is to recover from it. The main technique to achieve this goal is to *replicate* the components of the system that are vital for its correct operation. Through *replication*, a duplicate copy of a component is available in the system and is capable of performing the required tasks in case of a failure.

An overview of the standard fault diagnosis process in industrial control system is depicted in Figure 3.2 [84]. Here, the system model is the correct expected behaviour of the system to a given input signal *u(k)*, where k is a given time in either continuous or discrete space. The input signal to the system undergoes the process. The generated output *y1(k)* is compared with the desired output produced by the system model, resulting in the residual signal *r(k)*.

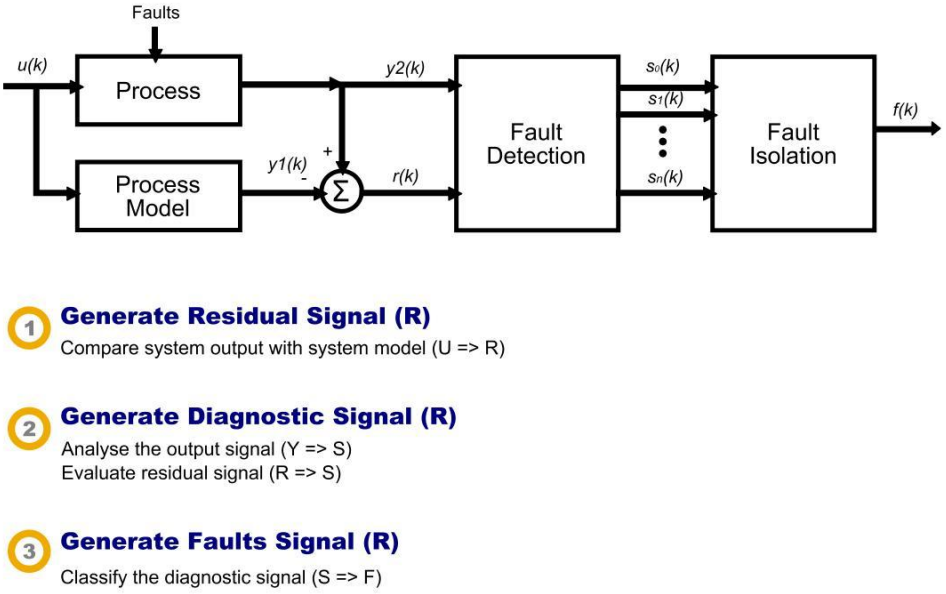


Figure 3.2: Fault diagnosis process for industrial control systems.

Based on the analysis of *r(k)* and on the direct analysis of additional output signals (e.g. *y2(k)*), diagnostic signals ($s_0(k)$, $s_1(k)$, ..., $s_n(k)$) are generated. These signals depend on the variable that is monitored. For example, the signal may contain information on timeout of sensor nodes, or indicate that a sensor reading is deviating from the expected value. These signals serve as input for a final analysis where the mapping of diagnostic signal $\overrightarrow{S}$ to the isolated failures $\overrightarrow{F}$ occurs.

This conceptual model can also be applied in the WSN domain. In this case the process is the WSN and *u(k)* are changes in the environment that could cause modifications on *y(k)*. The WSN's output signal that is monitored depends on the type of failures that is being diagnosed. For example, it can be the values acquired by the sensor nodes, in case that the system is evaluating value failures, or the number of messages generated by each node, if the system is monitoring crash failures.

The process model will generate an expected output value, which can be compared with *y(k)* to generate *r(k)*. An example of a process model is the expected number of messages from each node based on a known message heart beat, or an expected value for variables measured by sensors.

Fault detection (or residual evaluation) has the purpose of generating diagnostic signals for further analysis by the isolation process. For example, the fault detection process can analyze *r(k)* to determine which nodes have reached a timeout, and which nodes are generating outlier value readings.

Finally, the fault isolation will evaluate $s_0(k)$ and $s_1(k)$ to define the failures that have occurred in the system, e.g. the isolation process can identify a loose connection between a sensor and the node exists, or indicate whether a node in the routing path of several other nodes is malfunctioning.

3.2 Wireless Sensor Network Faults in Industrial Applications

The benefits of applying WSNs in a vast range of enterprise scenarios is evident, as described in section 2.1. Nevertheless, losses for the industrial application can occur when the availability of the functionality offered by this technology cannot be ensured.

By availability, we understand the probability with which a request to a WSN will lead to a valid and useful response. Availability (or point availability) is defined as:

$P(A) = \frac{MTTF}{MTTF+MTTR}$

Where $MTTF$ stands for *Mean Time To Failure* and $MTTR$ stands for *Mean Time To Repair* [12].

Systems that constantly fail and require long repair time will have very low availability. However, systems that have a high $MTTF$ and can be quickly repaired are considered highly available systems.

Availability and reliability are closely related concepts. In [13], Birolini defines reliability as "*... a characteristic of an item, expressed by the probability that the item will perform its required function under given conditions for a stated time interval. From a qualitative point of view, reliability can be defined as the ability of the item to remain functional. Quantitatively, reliability specifies the probability that no operational interruptions will occur during a stated time interval*".

Hence, to analyze the effects of unreliable WSNs applied in business processes it is necessary to investigate the sources of malfunctions of these devices. In this context it is important to point out the difference between faults, errors, and failures. Various definitions of these terms have been used [12, 29, 162]. This thesis refers to the definition given in [162]:

- A *fault* is any kind of defect that leads to an error.
- An *error* corresponds to an incorrect (undefined) system state. Such a state may lead to a failure.

- A *failure* is the (observable) manifestation of an error, which occurs when the system deviates from its specification and cannot deliver its intended functionality.

Figure 3.3 illustrates the difference between fault, error, and failure. A data acquisition program, running on *node A*, is expected to periodically send the measurements of its sensors to a data aggregation program, running on *node B*. However, *node A* suffers an impact causing a loose connection with one of its sensors. Since the code implementing *node A*'s program is not designed to detect and overcome such situations, an erroneous state is reached when the the program tries to acquire data from the sensor. Due to this state, the data acquisition program does not send sensor data to the data aggregation program within the specified time interval. This results in a crash or omission failure of *node A* observed by *node B*.

In the scenario explained above, the fault is the loose connection of the sensor. The error is the state of the data acquisition program after trying to read the sensor data. The failure occurs when the data acquisition program does not send the sensor data within the specified time interval.

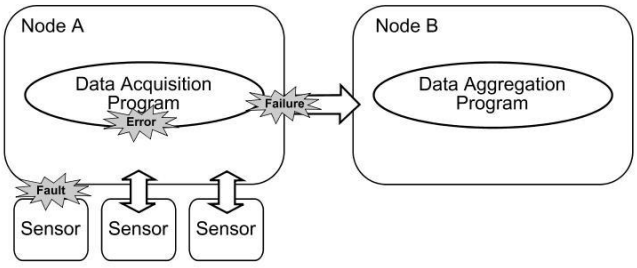


Figure 3.3: Failure caused by a loosely connected sensor.

3.2.1 Sources of Faults in WSN Applications

Wireless sensor networks are commonly deployed in harsh environments and are subject to faults in several layers of the system. To analyze the faults that can occur in real application scenarios, we performed a research on several application trial reports. The experience from these expeditions can be used as guidance for future application trials to avoid the same errors from happening.

Figure 3.4 presents a layered classification of components in a WSN that can suffer faults. A fault in each layer of the system has the possibility of being propagated to higher levels. For example, a power failure of a node $n1$ will cause the entire node to fail. If $n1$ is on a routing path, the messages of nodes $\Lambda_{(n1)}$ will not be delivered, making an entire region of the network silent until the routing path is restored.

Ultimately, if the application in the back-end, which presents the WSN data to the users, suffers a fault due to some software error or hardware failure, the entire system is considered faulty. In this thesis, however, we will concentrate on faults that can happen in the sensor nodes and the sink.

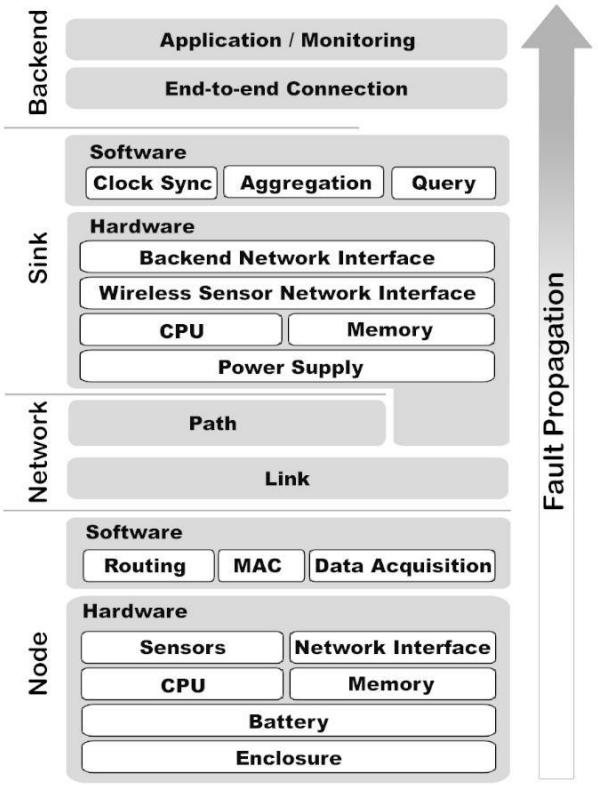


Figure 3.4: Fault classification and propagation.

3.2.1.1 Node Faults

Nodes have several hardware and software components that can produce malfunctions. For example, the enclosure can suffer impacts and expose the hardware of the sensor node to extreme conditions of the environment. In [110, 163, 159] due to stress from the environment and inadequate enclosures, the sensor nodes were exposed to direct contact with water,which caused short circuits. The report of a large-scale deployment in a potato field [93] indicated that the antennas from the nodes were quite fragile and would become loose when inserting the node into the packaging.

When the battery of a node reaches a certain stage, sensor readings may become incorrect. This has been observed in [168] where many outlier readings were generated in the network caused by imminent battery failure. As demonstrated in Figure 3.4, hardware failures will generally lead to software failures. A *Data Acquisition* program will not perform properly if the underlying sensors are providing incorrect

readings. Nevertheless, some hardware failures do not affect all the functionalities in a sensor node. In the example discussed, although the node cannot be used to provide correct sensor readings, it can still be used to route packets in the sensor network.

Software errors are a common source of failures in WSNs. In [174], the researchers reported that a software error caused the longest continuous network outage, taking the system offline for three days until the nodes could be manually reprogrammed.

Organizing a network in clusters is an approach used in many applications, for example to extend the lifetime of the network [58]. A small number of nodes are selected to become cluster-heads, e.g. $\{n_{c1}, n_{c2}, ..., n_{cn}\}$. They are responsible for coordinating the nodes in their clusters which consists of $\Lambda(n_{ci})$. The coordination can consist for instance of collecting data from them and forwarding it to the base station.

In case that a cluster-head fails, no messages of its cluster will be forwarded to the base station any longer. The cluster-head can also intentionally, or due to software errors, forward incorrect information. Depending on the application case, the impact of such a failure can vary from the quality degradation of measurements to alarm messages not being delivered to a back-end system.

While forwarding messages, nodes can aggregate data from multiple other nodes in order to reduce the amount of data sent to the base station. One common simple approach is to calculate the average of correlated measured values such as temperature, humidity and pressure, and send only one message to the back-end.

If a node generates incorrect data, the data aggregation results can suffer deviations from the real value. Also, if a node responsible for generating the aggregated data is subject to a value failure, the base station will receive incorrect information of an entire region of the network.

3.2.1.2 Network Faults

Routing is one of the fundamental building blocks in a WSN. It is essential for collecting sensor data, distributing software, configuring updates, and for coordinating nodes. Additionally, application-specific routing protocols may be required, for example for tracking and "following" moving objects. Faults on the routing layer can lead to dropped or misguided messages, or unacceptable delays.

In WSNs, communication paths P_b^a between nodes are highly volatile. WSNs do not always yield the same delivery rate of messages in field trials as in lab trials. For

instance, in [158] a delivery rate of only 58% of the messages was observed, and in [150] the instability of the link quality ζ_b^a between nodes lead to constant changes in the routing paths.

In several scenarios of sensor networks, nodes have a certain degree of mobility. In a glacial expedition [110] the experiment assumed a one hop network. The connection of nodes to the sink was calibrated during deployment with a reliable link connection ζ_b^a. Nevertheless, due to ice movement, after some time, nodes dislocated and one became unreachable, resulting in complete loss of data from this node, i.e. $\zeta_b^a = 0$.

Radio interference can also cause a reduction in ζ_b^a. For instance, in agricultural fields the placement of nodes must be carefully planned to in order to consider the link range reduction once plants start growing, as discussed in [166]. Another source of link failure is the collision of messages. In [159] researchers observed a potential for collision of messages of nodes in close proximity due to a phase change and overlap.

In other situations, however, nodes may have perfect link connections but the messages are not delivered to their destination due to path errors. A software error in the routing layer can generate circular paths or simply deliver messages to the incorrect destination.

3.2.1.3 Sink Faults

On a higher level of the network, sinks $\{B\}$ are also subject to faults of their components. When a sink β_i fails, unless fault tolerant measurements are present, a failure of several nodes occur given that the data from the sensor nodes cannot be accessed, i.e. the set of nodes $\Lambda(\beta_i)$ will not be able to communicate with the back-end.

Sinks can be deployed in areas where no permanent power supply is present, in such applications, batteries together with solar cells are commonly applied [110, 93, 168] to provide the necessary amount of energy. This traditional technique has proved to be inefficient in the glacial expedition reported in [110]. Although this worked perfectly for other expeditions, in this glacial environment the sink suffered a power failure due to snow covering the solar panels for several days.

Network infrastructure is usually also not available in the area where sinks are deployed, and therefore alternative solutions such as a satellite connection are used, which can cause fluctuations in the back-end network interface. In [104] researchers indicated that during periods of severe thunderstorm activity, the satellite connection would become unavailable.

Finally, the software that stores the data collected from the network, processes it and sends it to the back-end system, is subject to errors. The presence of these errors

can lead to data loss during the period when the fault occurred. For example, in the first application trial realized by the CoBIs project, the software of the gateway presented malfunctions that prevented the back-end application from receiving the data generated in the network.

3.2.2 Failure Types

As discussed in Section 3.2.1, several faults could lead to failures in wireless sensor networks: a node could be moved to a different region causing a link failure; nodes can suffer power failure and stop responding to requests, or they can start sending arbitrary values either intentionally (after a security breach) or due to a malfunction.

Here, we use the classification proposed in [29] to define the failures that a WSN is susceptible to: crash, omission, timing, value and arbitrary. These failures are the observable manifestation of underlying faults presented in Section 3.2.1.

Crash or omission : A failure by omission is determined by a node n_i sporadically not responding to requests, or sporadically not sending events at the scheduled time. A crash failure of n_i occurs when the node at some point stops responding to any request and stops sending events at the scheduled time. An omission degree f can be defined, which imposes a limit to the amount of omission failures n_i might have before being classified as crashed.

Timing: Nodes might fail due to a timeout in processing a request or event, or by providing data too early. Such timing failures occur when a node n_i responds to a request or sends an event with the correct value, but the message is received out of the time interval specified by the application. Timing failures will only occur when the application specifies timing constraints.

Value: A node n_i is considered to have failed due to an incorrect value when the node sends a timely response or event. However, the response contains inaccurate information. For instance, the aggregation of data generated by other nodes could forward a result value to a sink β_i that does not correctly reflect the input data. Such situations could be caused by malfunctioning software or hardware, corrupt messages, or even malicious nodes generating incorrect data.

Arbitrary: Arbitrary failures include all the types of failures that cannot be classified in previously described categories. In [92], Lamport introduced the Byzantine Generals Problem in the context of distributed systems. Recent work shows how to deal with this problem in the domain of wireless sensor networks [83]. Byzantine failures describe a type of arbitrary failures that are in general caused by a malicious node that not only behaves erroneously, but also fails to behave consistently when

interacting with other nodes and applications. Typical failures in WSNs can include for example, an aggregation program sending both incorrect and correct values to the sink, or the inability of a node to route a message, despite sending the acknowledge to the sender.

3.3 State of the Art in Fault Tolerance for Wireless Sensor Networks

Processing constraints, battery powered devices, harsh environments, and unreliable communication channels, are some of the characteristics that make WSNs a unique research field. In this domain, it is common to have a node providing functionality to its neighbours. Multi-hop routing is a simple example where nodes forward messages on behalf of each other [148, 54]. Nodes with stronger hardware capabilities can perform operations for other nodes that would either have to spend a significant amount of energy, or that are not capable of performing out their own operations [59, 151].

These functionalities, however, may fail due to various reasons. In general, the consequence of such an event is that a node becomes unreachable or violates certain conditions that are essential for providing functionality.

In some cases, a failure caused by a simple software error can be propagated to become a failure of the entire sensor network [174]. This results in application trials failing completely [93] and is not acceptable in safety critical applications. Hence, in this research domain, solutions are required in order to provide fault management of WSNs applied in enterprise applications. Our aim in this section is to investigate the state of the art in fault tolerance for WSNs.

Many tools and mechanisms to enhance the reliability of WSNs have been proposed, in spite of the fact that they have not been designed focusing on fault diagnosis and recovery [119, 46, 95]. We investigate these approaches and classify them according to the fault management functionalities they offer. First a study on fault diagnosis is presented, followed by a classification of existing fault recovery approaches.

3.3.1 Fault Diagnosis Techniques

The goal of fault diagnosis is to precisely identify failures and their causes, and in some cases to predict if certain functionalities will continue to work properly in the near future.

The simplest way to perform such a task is observing and isolating the malfunctioning parts by an expert. This technique has obvious drawbacks: high effort and low scalability.

Research in fault diagnosis for WSNs has not been deeply investigated yet. Nevertheless, a considerable amount of work already exists in the area of fault detection.

The techniques of automatic fault detection for WSNs achieve their goal through collaboration between nodes [88, 37, 108, 141], through self diagnosis performed by sensor nodes [57], or through the use of a more powerful device such as a computer in the back-end [139, 135].

We classified the investigated techniques according to the parties involved in the process. Through *self diagnosis*, the node itself can identify faults in its components. With *group diagnosis*, several nodes monitor the behaviour of another node. Finally, in *hierarchical diagnosis*, the fault detection is performed using a detection tree where a hierarchy is defined for the identification of failed nodes.

3.3.1.1 Self Diagnosis

Figure 3.5 presents the behaviour of nodes when self diagnosis is applied. In this approach the node itself performs routines to identify if its subparts are performing their tasks as expected.

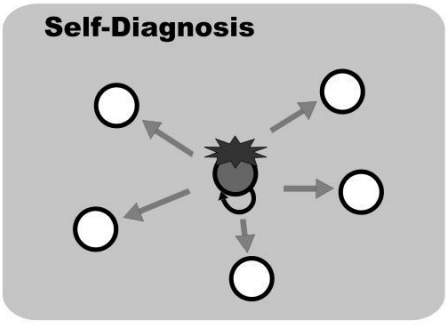


Figure 3.5: Self diagnosis in WSNs.

In [57], the authors propose an approach where nodes perform self diagnosis based on the measurements of accelerometers. This allows the nodes to determine if they suffered an impact that could lead to hardware malfunctions. Once the impact is detected by a node n_i, it advertises its current failure to $NB(n_i)$.

It has been observed that nodes with their battery close to exhaustion can generate incorrect sensor readings [168]. If the hardware allows the measurement of the current battery voltage, these failures can be predicted and an estimation of the time to death of the battery can be calculated [11, 133]. With this information

nodes have the possibility of announcing their status to prevent the propagation of erroneous data to the system.

Finally, a common routine to detect incorrect sensor readings is to identify values that cannot be generated under normal conditions, i.e. humidity above 100% or extremely high or low temperatures in a controlled environment. In [134], the authors apply this approach to detect sensor nodes that have value failures. In their deployment, they observed a sensor node that only generated data that was below its total detection range. The proposed approach also applies a small set of rules. For example, if the standard deviation calculated for some data points is beyond a certain threshold, then these points are identified as incorrect.

The authors in [96] have employed a similar approach of self diagnosis to identify incorrect sensor readings. They established a set of rules based on heuristics, phenomenological, and statistical methods. For example, using sanity levels, minimum and maximum environmental parameters, and statistic methods such as standard deviation. The correlation coefficient is introduced as a metric to calculate the level of correlation between different sensor readings. They calculate the correlation coefficient between the solar and temperature sensor readings for 24 hours time interval. If the correlation coefficient goes down below a certain threshold, the readings are identified as failed.

Performing in-network detection has the advantage of scaling well with the number of nodes due to the distributed nature of these techniques. Although self diagnosis has a great scalability potential, it relies on the fact that the sensor node is capable of correctly executing the diagnostic algorithm. In case of strong impacts and software errors this may not be achievable. Therefore, additional techniques must be in place to ensure a correct diagnosis of failures.

3.3.1.2 Group Diagnosis

Group diagnosis techniques seek to identify malfunctioning nodes through collaboration between nodes, as depicted in Figure 3.6. In this category of solutions, neighbour nodes monitor each other to detect value and crash failures.

The detection of nodes failing due to incorrectly generated values is only possible if a reference value is available. The authors in [88, 37, 24] propose fault detection algorithms for the task of determining event regions in the environment with a distinguishable characteristic. These algorithms are based on the idea that sensors from the same region should have similar values unless a node is at the boundary of the event-region.

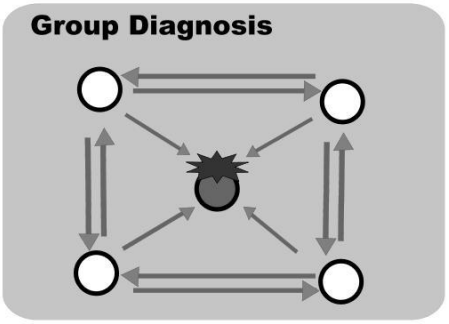


Figure 3.6: Group diagnosis in WSNs.

In [88], Krishnamachari and Iyengar propose a solution in the form of a Bayesian fault recognition algorithm. It exploits the notion that measurement errors due to faulty sensors are likely to be uncorrelated, while environment conditions are spatially correlated. The algorithm runs on each node n_i, making a binary decision about the correctness of its own sensor reading. The probability of $NB\,(n_i)$ reporting the same binary reading as n_i is then calculated. The Bayesian calculations estimate the probability of the node being faulty given the information about the correctness of its own sensor reading and the evidence regarding the readings of $NB\,(n_i)$.

The main drawback of the algorithm is that the sensors in the boundaries of an event region are likely to consider their reading as incorrect and therefore, mark themselves as faulty. Another drawback appears when nodes are at the edge of the deployed network. Such nodes have a reduced number of neighbours when compared to the nodes in the interior of the region. This can cause a higher number of erroneous fault detection. Nevertheless, the proposed algorithm has the advantage of being completely distributed and localized: each node only needs information from its neighbouring sensors.

In [24], the calculations and decisions regarding the correctness of the sensor readings are based on the readings of the sensor itself and its neighbours. The algorithm analyzes the history of the sensor data. During this process, it identifies sensors that generated outlier readings when compared to $NB\,(n_i)$. If the measurements (difference between sensor reading and readings of its neighbours) change significantly over the time, the authors assume that it is more likely that the sensor is faulty.

Similar to the approach explained in [88], the authors in [37] target the detection of faulty sensor nodes in a region where a certain event is observed (e.g. presence of a chemical). The main advantage of this algorithm compared to [88] is that it can accept any kind of scalar values as inputs and it does not only work with 0/1 predicates.

The algorithm gathers the sensor measurements of the neighbouring sensors and performs a comparison between the sensor measurement of the node and the mean of the measurements of the neighbouring nodes. If the difference between the sensor measurement and the mean of the measurements of the neighbouring nodes is greater than a certain threshold, it is assumed that the sensor reading is faulty. The outlier reading is identified by calculating the sample *mean* and *standard deviation* of the differences, then standardizing[1] the values and applying a predefined threshold.

This approach suffers from similar drawbacks as the method described in [88]. Its efficiency in the identification of failed nodes in event boundaries is reduced.

Another approach proposed in the literature is to let nodes observe whether the node providing functionality is in fact performing the operations that it is supposed to. In [108], a misbehaviour detection algorithm is used to aid the routing layer. Two main mechanisms are defined: a misbehaviour detection (*watchdog*) and a *pathrater*. The misbehaviour detection mechanism is based on the idea of eavesdropping the communication channel to verify whether messages are forwarded correctly and represents an extension of the Dynamic Source Routing (DSR) protocol [70]. The *pathrater* algorithm aids the routing protocol in the task of avoiding misbehaving nodes by calculating a rate to the routing paths based on their reliability.

A disadvantage of this approach is its need for bilateral communication and the limitation of encryption mechanisms adopted in the communication channel to one that enables eavesdropping the contents of packets content by neighbouring nodes.

Finally, focusing on providing a fault-tolerant approach for clusters in WSNs, the authors in [53] propose to support the dynamic recovery of failed sinks. The proposed protocol assumes that a sink has failed only if no sinks can communicate with it. The fault detection mechanism is based on constant status updates being exchanged between sinks and the further use of an algorithm to reach a consensus.

3.3.1.3 Hierarchical Diagnosis

When using hierarchical diagnosis techniques, nodes organize themselves in a way that a monitoring tree is formed, as depicted in Figure 3.7. Often in such systems, the monitoring duties are shifted to a more powerful device such as the sink to reduce the expenses of in-network resources.

The definition of a detection tree allows scalable fault detection in WSNs. Memento [141] proposes the usage of the network topology to forward fault detection results of child nodes to the parent nodes until it reaches the sinks. Each node forwards

[1] transfer the value to a standard normal distribution

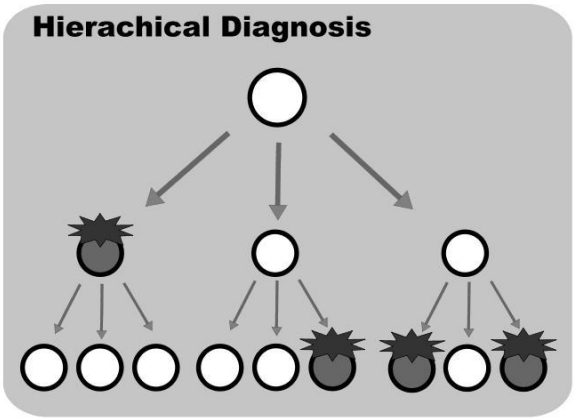


Figure 3.7: Hierarchical diagnosis in WSNs.

the status of the child nodes that it is monitoring to its parent node. Parent nodes perform an aggregation operation on their own result and on the results of the child nodes. The calculated value in then forwarded to the next level.

The approach proposed by Memento scales well with the network size; however, it consumes network resources. Shifting the fault detection task to a more powerful device is an alternative that can help increase the lifetime of the WSN.

In [155], the authors propose an algorithm that delegates the task of detecting and tracing failed nodes to the sink. At first, nodes learn the network topology and send their portion of the topology information to the sink. With this information the sink learns the complete network topology, which is used to send route updates as soon as the sink detects that nodes have become silent. These route updates transform the topology in a way that silent nodes are connected to healthy nodes. Using the updated route structure, the sink is able to determine which nodes have in fact failed, and which nodes were not responding due to a failed node in the routing path.

This approach is not applicable to event-driven WSNs. In [143], the authors propose a mechanism that uses a hierarchical network topology. In this network cluster-heads $\{n_{c1}, n_{c2}, ..., n_{cn}\}$ monitor ordinary nodes, and the sinks $\{B\}$ monitor the cluster heads. To monitor nodes, each sink β_i and each cluster-head n_{ci} constantly ping nodes that still have battery power left and that are under their direct supervision $\Lambda(\beta_i), \Lambda(n_{ci})$. If a node does not respond, it is marked as failed. One undesirable result of this approach is the incorrect indication that several nodes have failed when a cluster-head crashes.

Sympathy [135] is a debugging tool that also uses the hierarchical diagnosis approach. This tool instruments the WSN with monitoring software deployed on the sensor

nodes. This software generates metrics data that is forwarded to a centralized sink location for analysis. With this information, Sympathy applies a binary diagnostic tree to detect crash, timeout and omission failures and identifies the fault that generated the failure.

The major drawback of Sympathy is the need for specialized software running on the sensor nodes and the additional data that has to be sent to the sink. This requirement from Sympathy prevents its adoption in heterogeneous WSNs, making this approach unsuitable for enterprise business systems.

SNIF [139] is an example of a debugging tool, which aims at improving the Sympathy concepts. Contrary to Sympathy, this tool does not require additional traffic to be transported through the WSN. To automatically identify network failures, this tool proposes a binary decision tree. Although the authors claim that no additional traffic or software modification is required, the diagnostic tree makes use of routing topology information for deciding whether there is a failure and what kind of failure is occurring.

3.3.2 Fault Recovery Techniques

Fault recovery techniques seek to bring systems back to a stable operational state once failures have occurred. Several techniques have been proposed to increase the reliability of WSNs [49, 111, 35, 58, 46]. Here, we give an overview of this distributed efforts.

Recovery techniques are usually based on the method of replicating of components. This method ensures the continuity of an operation even if one of the replicated components is not working as expected. We classify recovery techniques for WSN into two major replication approaches: *Active* and *Passive*. Active replication means that all requests are processed by all replicas. Passive replication, a request is processed by a single instance (called primary replica) and only when this instance fails, another one takes over the tasks.

3.3.2.1 Active Replication in WSNs

Active replication in wireless sensor networks is naturally applied in scenarios where all, or several, nodes provide the same functionality. One example is nodes that periodically provide sensor data. Nodes that run this program activate their sensors and forward their readings to an aggregation node or to a sink. When some nodes fail to provide their sensor readings, the recipient still gets the results from other nodes, which is often sufficient.

Multipath routing

It is desirable to avoid that a single node crash partitions a entire network. Thus, a network should be k-connected, which allows $k-1$ nodes to fail while the network would still be operational [97]. Multipath routing [49] can be used to actively replicate routing paths. In [17], Bredin et al. propose an algorithm that calculates the minimum amount of additional nodes and their positions to guarantee k-connectivity between nodes.

Value aggregation

Value aggregation is a research area that provides mechanisms to merge the data generated by several nodes. This data can be the result of a specific request or simply the report of sensor value readings. Different approaches can be adopted for handling the differences in the received values.

Sensor value fusion [111, 35], an example of such an approach, is a research area that provides high-level information derived from a number of low-level sensor inputs. In this research domain, the inherent redundancy of sensor nodes can be used to provide fault-tolerant data aggregation. This is achieved through a trade off between the precision (the length) of the resulting sensor reading interval and the number of faulty sensors. This ensures that, despite node failures, the resulting reading interval contains the correct sensor reading of a region.

Another simple but efficient solution to prevent the propagation of a failure of one specific node to the entire network is to ignore the data that it is generating, as applied in [57]. The major challenge in this case is the identification of the malfunctioning nodes.

3.3.2.2 Passive Replication in WSNs

When passive replication is applied, one replica (called primary) receives and processes all requests. In the event of a failure of the primary replica, one of the backup replicas assumes the primary role and starts providing the requested functionality.

In order to maintain consistency between replicas, the state of the primary replica and the request information are transferred to the backup replicas. Given the constraints of WSNs, applications should be designed to be stateless, which minimizes the overhead for transferring state information between nodes.

The process of recovering from a fault when using passive replication is illustrated in Figure 3.8 and consists of three main steps: fault detection, primary selection and code distribution (or re-configuration).

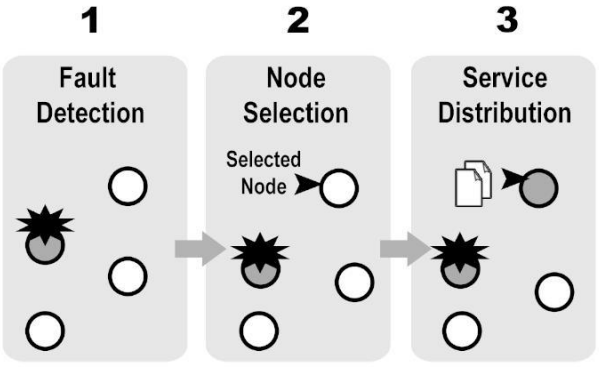


Figure 3.8: Steps to recover from a failure in a passive replication mode.

Phase 1: Fault Detection

The first step in a fault tolerant system, based on passive replication, is to detect malfunctioning parts. Fault detection approaches have been discussed in section 3.3.1. These techniques are classified as: self diagnosis, group diagnosis, and hierarchical diagnosis.

Once the failure is detected the next step is to select the new primary replica.

Phase 2: Node Selection

After determining that certain functionality in the primary replica is no longer available, a new provider must be selected. After this selection phase, a backup node becomes the new provider, assuming the primary replica role.

Several approaches on how the selection is performed have been proposed [58, 46, 53, 155]. Here we classify the different approaches according to the parties involved in the decision process.

Self selection - With *self-selection*, the node itself determines if it should become a primary replica. In LEACH [58], nodes periodically execute a probabilistic algorithm to determine whether they should take over the role of a cluster-head. In this probabilistic rotation system, nodes keep changing their role in the network. This approach enables the recovery of the system in case of a cluster-head failure. The time required to recover from such failure is the period of one role rotation.

Role assignment algorithms determine which role, such as coverage, clustering, and in-network aggregation, a node should be assigned to. In [46], a deterministic algorithm for autonomous role assignment is proposed. The assignment process considers the properties of the node itself, such as battery status and location, as well as the

neighbourhood and the roles chosen by neighbouring nodes. Role assignment mechanisms facilitate the localized self-configuration of a sensor network. This mechanism can re-establish provisioning of network functionalities if executed after node failures.

Group selection - In this selection method, a group of nodes selects the primary replica. An example of this technique is described in [53]. In this paper the authors propose the reallocation of nodes that were part of a cluster which suffered a cluster-head failure. The cluster-head, called sink, is considered to be a resourceful node. The solution assumes that all sinks in the network maintain a list of nodes that are currently in their cluster. Each sink also stores another backup list of nodes that could become part of their cluster. In the event of a sink failure, nodes that were allocated to the failed sink are transfered to new sinks. During the reallocation process, healthy sinks aggregate nodes if these nodes are in the backup list from the sink. If more than one sink has a specific node, a group selection process is triggered. The cluster-head selection is then based on the lowest communication cost of the nodes.

Hierarchical selection - In a hierarchical selection, a coordinator selects the new primary node. This applies to the rebuilding of routing paths [155], as well as the selection of a new cluster-head [55]. The former describes an algorithm to select the closest node to the sink. The latter approach applies fuzzy logic to the sink to select which node should become a cluster-head. This algorithm makes use of a fuzzy descriptor, the node concentration, energy level in each node and its centrality with respect to the entire cluster.

Phase 3: Code Distribution

During this phase, the nodes selected to become providers of specific functionalities must activate the program. In some cases the code is already available on the nodes and a simple change in the node's configuration is required. However, in some cases, e.g. when nodes do not have enough memory to store the code of all potential functionalities, it is necessary to inject code into the node. There are different techniques that can be used for code distribution: completely reprogramming the node, sending entire blocks of executable code, or sending small pieces of code such as scripts.

Pre-copy - As described in [46], this technique consists of making the code of all functionalities available on all nodes before deployment. This allows nodes to change their behaviour according to the role they are assigned to.

Code distribution - Several approaches have been proposed for disseminating code throughout the network. Maté [95] is an example for a byte code interpreter

for TinyOS where code is broken into capsules of 24 instructions. These capsules can be distributed through the network and installed on nodes. After installation the nodes start executing the new code. Agilla [44] is a Maté-based mobile agent middleware for programming wireless sensor networks. These mobile agents can be programmed to move through the network or replicate themselves to other nodes according to changes in the environment.

Impala [99] is a middleware for sensor networks that supports software updates and on-the-fly application adaptation. Unlike Maté, Impala focuses on networks that have a high degree of mobility, which can lead to long delays until an update is finished. While Maté stops the execution of an application until the update is finished, Impala processes ongoing software updates in parallel.

Remote execution - On the one hand code distribution is an approach that reduces the amount of memory required in the entire network, since not all nodes need to have the application pre-installed. On the other hand, code distribution consumes energy on the nodes exchanging the code and is susceptible to link failures. This can cause long delays until the code update is completed. Remote Execution [142, 140] is an alternative approach where low-power devices transfer tasks to more powerful devices without transferring the entire application code. Instead, only the required state information is transmitted. Such an approach is especially suited for heterogeneous sensor networks with at least some resourceful nodes.

A hybrid approach between code migration and remote execution is proposed in [106], where the application code is copied to another node when the battery level reaches a first threshold. As soon as the battery reaches a critical level, the execution state is transferred and control is handed to the remote node. This allows for the full usage of the available energy resources, since control is handed over before a node fails.

3.3.3 Classification and Evaluation of Techniques

As presented in sections 3.3.1 and 3.3.2, the existing work in the area of fault management in WSNs is distributed among several research domains. For the successful application of such techniques in enterprise systems, it is necessary to understand the cases in which they can be used considering their shortcomings. Table 3.1 provides an overview of the existing techniques for fault diagnosis in WSNs. We classify the investigated approaches according to: *(i)* the types of faults they are able to diagnose, *(ii)* the diagnosis mechanism applied, and *(iii)* the requirements of the approach.

	Diagnosed Fault Type			Fault Diagnosis Mechanism			Requirements	
	Crash	Value	Arbitrary	Self	Group	Hierarchical	Node Software	Information Set
[108]	X	X	X		X Eavesdropping		X	Packet content
[53]	X Sink Timeout				X Consensus			Sink Status
[106]	X Battery			X Lifetime			X	Battery Voltage
[57]			X Impacts	X Impacts			X	Acceleration
[44]	X Temperature			X Fire reaction			X	Temperature
[155]	X Timeout					X Sink	X	Topology
[88]		X				X Bayesian	X	Neighborhood
[143]	X Timeout					X	X Ping	Timestamp
[141]	X Timeout					X In-network	X	Topology, Timestamp
[135]	X Timeout					X Sink	X	Topology, Timestamp
[139]	X Timeout					X Sink		Topology, Timestamp

Table 3.1: Classification of fault diagnosis techniques.

	Fault Recovery						
	Active Replication	Passive Replication					
		Node Selection			Code Distribution		
	Mechanism	Self	Group	Hierarchical	Pre-Copy	Code Distribution	Remote Execution
[108]			X		X		
[106]						X	X
[57]	X Discard information from failed node				X		
[155]				X	X		
[44]				X Node with mobile code decides		X	
[53]			X Sink with minimum communication cost	X	X		
[46]		X Set of rules			X		
[49]	X Multipath routing						
[111, 35]	X Sensor Fusion						
[58]		X			X		
[95, 99]						X	

Table 3.2: Classification of fault recovery techniques.

Each of the diagnosis methods has its advantages and drawbacks: on the one hand self detection mechanisms, as proposed in [106], involve no communication costs except for announcing that a fault has been detected. On the other hand, sudden crash failures cannot be detected in this way. Group detection mechanisms, where nodes monitor each other, allow the detection of sudden crash failures. Such mechanisms impose higher energy costs due to the exchange of messages and the coordination of nodes. Additionally, the use of end-to-end encryption is often impracticable for value failure detection, since this would hamper other nodes observing the contents of messages [108]. Finally, in a hierarchical detection approach, such as [155], most of the communication and coordination costs can be shifted to a more powerful device. Nevertheless, hierarchical approaches require efficient techniques to handle the scale of the system.

Table 3.1 shows that most of the solutions discussed in the literature focus on individual types of faults. To our knowledge, the approach proposed in [108] is the only one that currently detects the three types of faults used in our classification, with the restriction that the detection of value faults is limited to forwarded messages. Content analysis, as described in [88], could extend its applicability, e.g. to aggregation.

This table also indicates that the modification of the node software is a common practice to diagnose node failures [108, 106, 57, 44, 155, 88, 143, 141, 135]. As described in chapter 1, in enterprise scenarios we cannot assume specific functionality available in the sensor nodes. Neither can we rely on information that may not be available in heterogeneous WSN platforms, such as network topology and neighbourhood analysis, as proposed in [135, 139, 141, 88].

Table 3.2 presents the classification of the solutions with respect to the fault recovery mechanism applied. The fault recovery mechanisms analyzed can be divided into two main branches: passive and active replication.

In the case of active replication the common approach is to remove the node from the route path or ignore its data as in [57, 49, 111, 35]. Nevertheless, this implies that nodes must run the same applications, which imposes higher energy and memory consumption on the network.

Passive replication on the other hand only initiates a backup copy when nodes suffer failures, thereby reducing the energy costs during normal operation. Nevertheless, this approach can result in high energy and communication costs if a code is updated, or if many messages have to be exchanged to select a new primary replica.

The presented classification and evaluation of WSN diagnosis and recovery techniques indicate that most solutions are platform centric. i.e. requires a different implementation for each platform. Additionally, the techniques proposed in this field do not focus on integration of WSNs with enterprise business systems, making them unsuitable for industrial applications. Therefore, business applications require additional research in fault management techniques to ensure the successful utilization of WSNs in real world deployments.

3.4 State of the Art in Fault Diagnosis for Industrial Applications

The constant increase in the complexity of technological installations such as power, chemical, metallurgical, nuclear and food industries determines the need for efficient automatic fault diagnosis techniques. These industrial systems make use of a large number of sensors to monitor and control different process variables [129].

The incorrect generation of sensor readings or even the lack of them can often lead to total system shutdown, hazards, and can even put human lives at risk. The outcome of such event can be significant economic losses, as described in chapter 2. Hence, fault diagnosis for industrial systems has been the focus of research in the past few decades, resulting in many efficient techniques and fundamental theories [84]. Yet, the ongoing research on fault diagnosis for WSNs usually does not rely on this pre-existing knowledge.

Seeking to overcome this gap, we provide a concise survey on existing techniques and theories in this section. Different techniques are applied in the fault isolation process to evaluate the diagnostic signals generated by fault detection and residual evaluation algorithms. Here we investigate the most important existing approaches for fault isolation: *(i)* binary diagnostic matrix, *(ii)* diagnostic trees and graphs, *(iii)* rules and logic functions, *(iv)* artificial neural networks, *(v)* fuzzy and *(vi)* neural-fuzzy fault diagnosis systems. In the following subsections, the fundamentals of these techniques are explained. Finally, this section is concluded with a discussion on the advantages and drawbacks of applying these methods in WSNs.

3.4.1 Binary Diagnostic Matrix

Several approaches exist to isolate failures based on binary diagnostic [84]. Binary diagnostic matrix is one of such methods. It creates a mapping between the diagnostic signals $\overrightarrow{S}$ and $\overrightarrow{F}$ based on the Cartesian product of these two sets.

S \ F	f_1	f_2	f_3	f_4	f_5	f_6	f_7
s_1	1		1		1	1	
s_2		1	1				1
s_3	1	1		1			1
s_4		1					
s_5	1		1				

Figure 3.9: Graphical representation of a binary diagnostic matrix.

Figure 3.9 depicts a binary diagnostic matrix. Each failure f_k from the set of failures F has a *"signature"* in the diagnostic signal. For instance the signature of failure f_1 is:

$s_1 = 1, s_2 = 0, s_3 = 1, s_4 = 0$, and $s_5 = 1$.

The diagnosability of a system is determined by the possibility of diagnosing the different failure types present in a system without ambiguity. When the signatures produced by the failures are not distinguishable (i.e. f_5 and f_6) the system is said to be *not diagnosable*.

The binary matrix depicted in Figure 3.9 can be created not only based on the grounds of system equations, but also based on expert knowledge of the system by analyzing the influence of particular failures on diagnostic signal values.

3.4.2 Diagnostic Trees and Graphs

Diagnostic trees and graphs is another approach for the isolation of failures based on binary diagnostic signals. This method is widely used in the industrial domain [116, 18, 19].

The diagnostic inference is performed through the execution of the logic represented in a tree, as presented in Figure 3.10. In this diagnostic tree, the failures and their symptoms are arranged in a graph, where the diagnostic signals correspond to the vertices. The leaves of the tree indicate the failure that is occurring. The branches that connect the vertices correspond to the value of the diagnostic signal of the parent vertex.

A similar approach called Ordered Binary Decision Diagram (OBDD) provides a symbolic representation for boolean functions in the form of directed acyclic graphs, and is a restricted, canonical form version of Binary Decision Diagrams (BDD) [116]. Algorithms that implement operations on boolean functions as graph algorithms on OBDDs are discussed in [18].

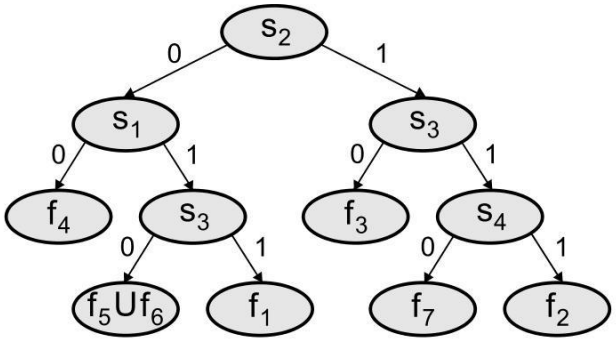


Figure 3.10: Diagnostic graph.

This approach has been developed as an attempt to optimize the search of the failure based on the symptoms detected by the system. As a result of efficient symbolic manipulations that OBDDs brings, a wide range of problems in hardware verification, testing, real-time systems, and mathematical logic have been solved. Most of these problems would have been otherwise impossible to solve due to the combinatorial explosion of system states [116].

3.4.3 Rules and Logic Functions

The use of rules and logic functions for the isolation of failures has been applied in several industrial application scenarios, such as petrochemical [136], mining [124] and power systems [78, 56].

These systems evaluate the current system state, diagnosing it based on sets of rules. These rules are the distillation of information and "rules of thumb" that one or more experts use when they assess the condition of a given system [100]. Figure 3.11 presents a set of rules extracted from [124], where the goal of the evaluation of the rules (inference) is the diagnosis of failures in an electric mining car.

The challenge of such systems, also called *Expert Systems*, lies in the generation of the rule base that is used to infer the diagnosis. In general, the rule base is generated based on the knowledge of experts in the system [100]. To facilitate the process of generating the rule base, studies investigate the possibility of automatically generating rules [101] and the use of tools that guide the experts through the rules creation process [68, 27, 91].

Some of these tools also called *Rule Engines*, provide frameworks for the evaluation of rules. Drools [27] is one example, where rules can be generated in a friendly way for both developers and business users. Another example, called Jess [91], is also

Rule 1

IF voltage between points 13 and 14 is normal

THEN control circuit transformer is ok

Rule 2

IF control circuit transformer is ok

AND control circuit breaker is closed

AND voltage between points 15 and 26 is zero

THEN problem is control circuit breaker

Figure 3.11: Failure diagnosis rules for an electric mining car.

based on the efficient reasoning Rete algorithm [45] and provides a fast light weight rule engine fully developed in Java.

3.4.4 Neural Networks

Artificial neural networks are information processing systems, which were developed as a generalization of mathematical models of human cognition or neural biology [42]. They have been applied in a broad spectrum of industrial applications for sensor fault detection [51, 112, 129, 179, 152]. Due to its performance properties, and ability to "learn" system patterns, artificial neural networks have their major impact when applied in complex and dynamic systems.

One prime example of the use of this technique was applied in a NASA (National Aeronautics and Space Administration) space shuttle main engine to validate sensor data [51, 112]. Another example, also investigated by NASA, describes an online health monitoring system based on neural networks [152]. This approach in used to detect malfunctions of an antenna pointing system of the NASA's Deep Space Network.

Neural networks have also been applied in other domains. In [179], the authors discussed the design of a neural network based sensor validation and fault detection system for a power generation system. In [129] a sensor fault detection and isolation technique using neural networks for dynamic systems with time delays was also proposed.

As described in [129] the motivation for using neural networks for these systems is the capability of neural networks to learn non-linear and complex dynamics of these systems using training examples.

In [42], artificial neural networks are described as systems that are based on the following assumptions:

1. Information processing that occurs at many simple elements called neurons.
2. Signals are passed between neurons over connection links.
3. Each connection link has an associated weight, which, in a typical neural net multiplies the signal transmitted.
4. Each neuron applies an activation function (usually nonlinear) to its net input (sum of weighted input signals) to determine its output signals.

Figure 3.12 presents a generic model of a multi-layered artificial neural network [128]. The topology of neural networks may considerably change from this model according to the number of existing layers and the connection between nodes. Nevertheless, this model represents a classical multi-layered network and serves as a basis for the explanation of more complex models. The depicted neural network is composed of the following layers:

- *Input layer:* The nodes in this layer are called *input units*. They represent the input to the network.
- *Hidden layer:* The nodes in this layer are called *hidden units*. They are called hidden since they are neither input nor output units.
- *Output layer:* The nodes in this layer are called *output units*. They represent possible concepts or values to be assigned to the input under consideration.

This network has three main underlying properties that describe the network topology (or connectivity) [48]: the types of connections, the order of connections, and weight range. The node properties describe the activation range and the activation function.

Each neuron has an activation function which determines its output depending on the weighted sum of its inputs. This can be a discrete or continuous function. Some of the well known activation functions as stated in [42] are: identity function, binary step function, binary sigmoid, and bipolar sigmoid. Sigmoid functions are the most commonly used activation functions for multi-layer neural networks.

When applied in fault diagnosis for the isolation of failures, neural networks are trained to match each pattern of the symptom vector to one of the known fault types or to the healthy state [84].

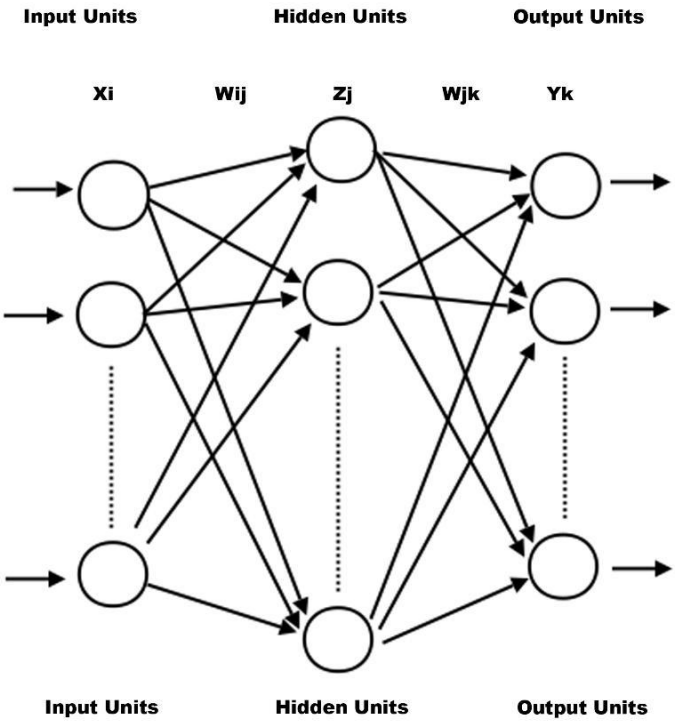


Figure 3.12: Neural network model, from [128].

To achieve this goal, the neural network undergoes a training phase, where a training algorithm based on a learning rule adjusts the connection weights in order to optimize the network performance [48].

Back-propagation is such an algorithm, which is usually adopted for training multi-layer feed-forward neural networks. The basic nature of this algorithm is simply a gradient descent method that minimizes the total squared error of the output computed by the neural network [42].

This algorithm performs a set of steps through the entire set of training data and re-adjusts the weights of the connections between the neurons. Such a cycle through the entire training set is called an epoch [42].

Normally, to train a neural network using back-propagation many numbers of epochs are required. The algorithm stops when a stopping condition is reached, for example if a certain number of epochs has been reached or if a local minimum of squared error was found.

At the end of the training phase, the neural network will provide as output the fault types according to the system state pattern provided.

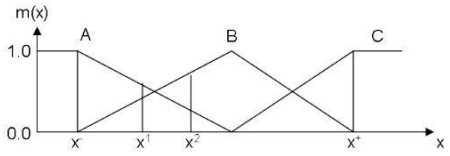


Figure 3.13: Division of variable x into fuzzy regions and the corresponding membership functions, from [128].

3.4.5 Fuzzy Systems

Artificial Intelligence (AI) has proved to be an efficient mechanism for solving problems in numerous domains, and has found a broad range of applications in diagnosis engineering [172]. In this research field, fuzzy logic emerges as a technique which is capable of dealing with sources of uncertainty, imprecision or incompleteness [181, 186, 182].

Fuzzy logic is particularly well suited for modelling non-linear systems [101], and has been successful in several applications where gradual adjustments are necessary [79, 137].

When compared to fuzzy logic, pure threshold techniques present several disadvantages. The isolation of failures based on threshold evaluation of residuals may be deceptive, lead to inference discrepancy and false diagnosis [84]. By introducing fuzzy logic, the uncertainties of diagnostic signals can be taken into account through linguistic variables that describe the state of the system.

Figure 3.13 depicts the fuzzy evaluation of a variable x. In fault isolation based on fuzzy classification, fault patterns are represented as fuzzy sets (or regions). The domain interval defines the most probable range of the variable, being divided into fuzzy sets according to the fault patterns known to the system (they can be equal or unequal in length) [172]. Each fuzzy set is assigned a fuzzy membership function (i.e. triangular, trapezoidal, and Gaussian).

In the example presented in Figure 3.13, the domain interval of x is divided into three regions A, B, and C, with trapezoidal and triangular membership functions.

Fuzzy systems have been successfully applied in numerous industrial domains. In the automotive industry, the authors in [101] used fuzzy logic to detect vacuum leak in the electronic engine controller as part of the end-of-line test at assembly plants. In [62], the uncertainty of diagnosing chemicals in a power transformer is discussed. There an approach based on fuzzy logic to model the uncertain boundaries

of chemical ratios is presented. In [50], fuzzy logic is applied together with other techniques to realize the diagnosis of a waste water treatment in France.

Finally, in [172] fuzzy logic is applied with the main focus on developing a fuzzy rule base considering two types of information: numerical information obtained from sensor measurements and linguistic information obtained from human experts. The numerical information are input and the corresponding output data pairs obtained by the system. The linguistic information are "IF-THEN" rules that are usually expressed from the experience of a human controller [172]. The key idea of their approach is to generate fuzzy rules from numerical data pairs, collect them into a common fuzzy rule base, and design a fault diagnosis system based on the combined fuzzy rules.

3.4.6 Classification and Evaluation of Techniques

Fault diagnosis for industrial applications has been widely investigated in the last decade. Nevertheless, most of these techniques have not been adopted so far for the diagnosis of failures in WSNs.

The gap that exists between fault diagnosis for industrial applications and fault diagnosis for WSNs is due to the difficulties to adapt the concepts developed for the industrial domain to the large scale of WSNs.

In order to analyze these difficulties, we classify and evaluate these techniques focusing on the characteristics that are mostly important for the WSN domain. Table 3.3 presents the investigated techniques and classifies them according to different Key Performance Indicators (KPI).

Although the *Binary Diagnostic Matrix* is extremely efficient on its deployment in small scale systems, its performance can become limited in WSNs scenarios. Due to the combinatory explosion of composite failures in WSNs, the set of failures ($\overrightarrow{F}$) will raise considerably with the number of failed components (sensor nodes, sinks and links).

Hence, solutions based purely on the analysis of the failed set such as *Binary Diagnostic Matrix*, *Diagnostic Graphs*, and *Expert Systems* are not adequate for WSNs due to the large scale of the system. Nevertheless, when *Expert Systems* are used not to identify each individual failure, but rather classes or simple special conditions, this approach has the potential to perform well.

Additionally, the effort required to adopt WSNs is a key factor in the decision process in business environments. Techniques such as *Binary Diagnostic Matrix*, *Expert*

Systems and *Diagnostic Trees and Graphs*, commonly require the analysis of the deployed system by an expert to determine the conditions of the diagnostic signals for each failure, therefore considerably increasing the adoption efforts.

Neural Networks present the great advantage of being able to learn system patterns based on a training set. The support of a system to gather the required data eliminates the need for an expert to analyze the system for each WSN deployment. Therefore, *Neural Networks* represent an interesting approach for business applications deployment due to adoption efforts reduction.

Experiments performed in [31] indicated drawbacks of this technique. With numerous failures and a high number of diagnostic signals, neural networks require long training periods. Additionally, similar to the human brain, neural networks can "forget" patterns when the training set is too large, which results in an incorrect diagnosis of failures.

Additionally, fuzzy systems do not present the same problem as neural networks where system patterns may be "forgotten". Nevertheless, the use of automatic fuzzy rule generation techniques may result in a large number of rules that must be evaluated during run-time. This leads to low performance as the system scales in the number of components.

As discussed, a solution for diagnosing failures in WSNs presents a major challenge regarding the scalability of the system. The adoption of each of these techniques in an isolated manner can lead to low performance of the system and incorrect diagnosis.

Nevertheless, through a careful selection and combination of techniques, it is possible to extract the best qualities of each technique and avoid their drawbacks. In Chapter 5, we present a hybrid approach that combines neural networks, statistical analysis and a rule based algorithm.

3.5 Summary

We studied the problem of fault diagnosis and recovery, surveying the different techniques currently applied in WSN research. A classification of the available fault tolerance techniques for wireless sensor networks has been proposed considering the various mechanisms adopted by the existing solutions. Seeking to bridge the gap between fault diagnosis for WSNs and industrial applications, we also investigated the existing solutions applied in control systems, focusing on their applicability in WSNs.

	Binary Diagnostic Matrix	Diagnostic Trees and Graphs	Rules and Logic Functions	Neural Networks	Fuzzy System
Input data	Expert knowledge	Expert knowledge	Expert knowledge	Training Data	Training Data/ Expert knowledge
Ability to Model complex and non-linear systems				X	X
Ability to Model uncertainty					X
Scalability -Setup -Diagnosis	 low (human interaction) low	 low (human interaction) high	 low (human interaction) medium/high	 low (large training sets) medium	 low (human interaction/ large training sets) medium
Deployment efforts -Human resource -Time	 High High	 High High	 High High	 Low Medium	 Low/High Medium/High

Table 3.3: Classification of fault recovery techniques.

Through the classification proposed, it is possible to compare the different solutions and identify the strong and weak points of each approach. This allows for a correct selection of the techniques that are more suitable to specific applications. By applying our classification we were able to verify that current approaches proposed for WSNs provide mechanisms for overcoming faults in sensor networks only in specific scenarios and applications. However, no approach provides fault diagnosis support to heterogeneous WSNs.

Finally, this research shows that the use of techniques and theories developed for industrial control systems can be beneficial for WSN domain. Nevertheless, due to the large scale of such systems, the isolated use of each of the surveyed techniques leads to low performance of the system, as discussed in section 3.4.6. Hence, to achieve good results in terms of fault diagnosis and recovery for this domain, new techniques which take the needs of business applications into consideration are required.

4. Fault Tolerant Framework for Wireless Sensors Networks

As the technology of WSNs evolves, the interest for integrating such devices with enterprise applications increases. These devices have the potential to improve business processes by reducing the gap between the real world and its virtual digital representation, as described in Chapter 2. Nevertheless, the diversity of hardware platforms imposes challenges in terms of integration with back-end enterprise systems.

The development of enterprise applications has recently embraced the concepts of service oriented architecture (SOA) based on web services [89, 127, 82]. This new concept delivers functionalities through well defined interfaces and protocols based on standards such as XML [16] and HTTP [43]. With SOA, the details of service implementation is hidden from enterprise applications. This new concept makes the process of switching service providers transparent.

Following the same vision, the functionality provided by WSNs must be accessible by enterprise applications through well defined interfaces and protocols, based on standards adopted by the industry [183]. Therefore the diversity of proprietary protocols that exists in WSNs must be hidden from enterprise applications to facilitate the integration and exchange of hardware platforms.

Although many routing protocols, especially in the sensor network area, propose different solutions to deliver collected data to the back-end [71, 77, 23, 66, 180], little work has been done to transparently couple sensor networks with business-oriented back-end systems.

Another major challenge faced for the integration of WSNs with business processes is the availability of services offered by the network of nodes. As described in Chap-

ter 2, WSNs are subject to frequent failures due to their tight integration with the environment. As a result of failures, false information (or none at all) can be generated and propagated to the back-end systems. The outcome may range from the lack of data of an item to incorrect business processes being triggered and even contamination of the environment, according to the scenario in which the WSNs are applied [84].

Regardless of the outcome caused by the failure, business processes will have to be corrected and maintenance will usually be required. This leads to additional costs associated with the adoption of WSNs by the industry. As one of the main goals of enterprise applications is to reduce the costs of business processes, we propose a fault tolerant framework for wireless sensor networks FT-WiseNets, which mitigates the propagation of failures to back-end systems. Our approach envisions a separation of the system in layers, allowing the implementation of different techniques for fault tolerance despite the diversity of hardware platforms currently available.

The requirements of the proposed framework are derived from previous experiences in a real world application trial in an enterprise environment [26]. These requirements have not yet been fully addressed by the current solutions proposed in the literature, as these solutions do not handle the integration of WSNs with back-end systems (discussed in Chapter 3).

The solution we propose in this Chapter, presents a novel approach for a structured management of failures, with extensibility and transparency, which are key requirements for enterprise applications. The adoption of this framework can reduce the threat of failures being propagated to upper layers and facilitates the integration of new techniques developed in the research community.

4.1 Requirements

The definition of requirements is the basis of every project. The analysis of the needs of stakeholders such as users, customers and the industry represents the first phase of any enterprise application's life-cycle and dictates the activities that will be required in the project [20, 64].

Since a thorough analysis of software requirements imposes costs in terms of time, man power, and resources, several software systems fail due to shortcomings in this phase [69, 105]. Nevertheless, research shows that the proper identification of information requirements early in the development process increases the possibilities of a successful project [60, 171].

To ensure a correct definition of the requirements for the proposed framework, we analyzed the application trial performed by the CoBIs project (described in Section 2.1.2). This trial was selected for the requirements analysis due to its focus on the use of WSNs integrated with business processes. Additionally, the lack of use of fault tolerant techniques represented a challenge during this trial as failures occurred in several layers of the system.

Figure 4.1 presents the setup of this application trial. In this scenario, sensor nodes were integrated with the Environment Health and Safety (EH & S) backend system [145]. The storage regulations were managed by the industry specialist directly on EH & S. Once modified, these storage regulations were propagated to the sensor nodes. The nodes would then react according to the new rules received and would generate alert messages if the rules were violated.

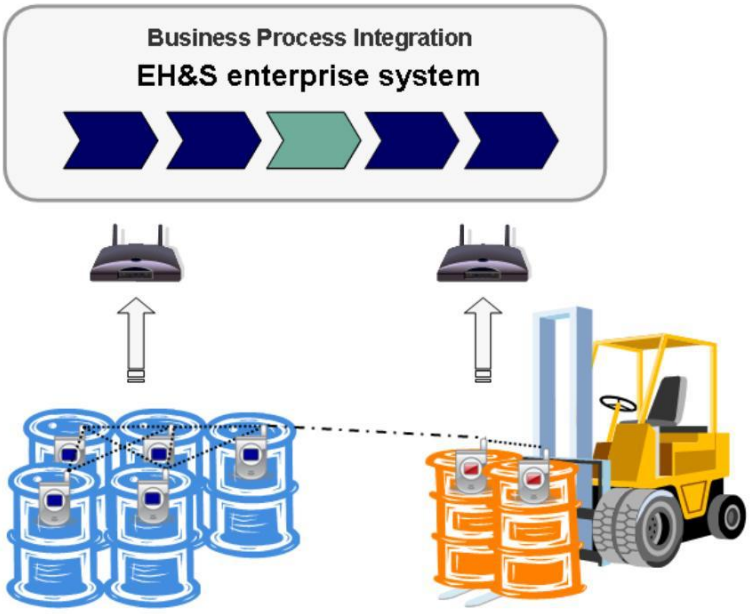


Figure 4.1: Application trial at a chemical plant.

During this deployment, the system stopped alerting the users several times about hazardous situations. This occurred due to failures in the sensor nodes and in the sinks, and bugs in the software responsible for storing the events generated by the WSN into the database. Although the trial did not use real chemicals (water was used instead), in such a scenario failures in the sensor network must be efficiently diagnosed and recovered in order to prevent accidents from happening.

Through the analysis of this scenario, we have derived the six major requirements of the FT-WiseNets framework, which reflects the need to provide highly available WSNs to business processes. These requirements represent the needs of the industry

and workers that are responsible for the maintenance of WSNs applied in business processes.

Automatic fault diagnosis:

WSNs generate a high amount of events per second, which makes the task of manual identification of malfunctioning components inefficient. Automatic isolation of failures overcomes this challenge through the use of algorithms that constantly monitor the WSN. This provides online information of the system behaviour and identifies the failed parts of a system.

The requirement of automatic isolation of failures originates from industry and maintenance workers. As WSNs are a new technology that is starting to be introduced in the industry, there is few professionals with know-how in this field. With efficient automatic fault isolation, workers can better perform their job and thereby reduce maintenance costs, which represents a key factor for the industry.

The automatic isolation of failures that considers the scalability of WSNs is also necessary for providing inputs for automatic recovery mechanisms.

Automatic recovery techniques:

Breakdown states, caused by failures occurring in WSNs applied to business processes, can result in high maintenance costs, delayed deliveries, unsatisfied customers and propagation of delays to additional business processes.

Some failures in WSNs do not require a physical interaction to be eliminated, for instance, outlier sensor readings can be eliminated through sensor fusion [111, 9], and coverage of a certain functionality can be guaranteed to a certain degree through the use of code-redeployment [22] even in the presence of node crash failures.

Automatic recovery techniques are a key requirement for the industry, given that they have direct impacts on the costs associated with the adoption of WSNs technology.

Extensibility:

Given the rapid development of new techniques for fault tolerance in WSNs, the framework must be extensible to allow new techniques and approaches to be easily integrated. The integration of new techniques should be enabled through the use of standards adopted by the industry. This improves the re-usability of any software component developed.

This requirement is also derived from the industry due to the dynamic growth of businesses and the need of rapidly improving processes with new techniques proposed in the research community.

Transparency:

SOA is a new trend in software engineering, which seeks to facilitate the development of enterprise applications [89, 127, 82]. One of the main focuses of this new paradigm is the transparency provided to applications in the sense of accessibility and development.

As business applications are programmed with a higher view of the business processes, the use of different fault tolerant techniques and the diversification of hardware platforms must be transparent to these back-end applications. Transparency is a key requirement demanded by the industry for successful development of new enterprise applications.

Support to heterogeneous WSNs:

The scenarios in which WSNs can be applied in industry can vary from inventory management to the supervision of storage regulations. Different scenarios require different support from hardware platforms. This leads to a great variety of protocols that have to be handled in order to allow applications to communicate with the WSN.

Following the transparency requirement, the framework must support the integration of heterogeneous hardware platforms in a transparent way to enterprise applications.

In this context, this framework investigates the research question of how feasible is to provide fault tolerance to WSNs with a restricted information set that can be provided by most hardware platforms. This framework should only rely on three basic information quanta: sensor node unique identification number, timestamp information and the time interval between messages (heart beat).

Delivery of online data:

In many application cases, back-end processes need a large amount of information about the current, or even past, status of business relevant items, and of their environment. In these cases, sensor nodes must collect and deliver online data to the backend systems.

4.2 Architecture

Today, a major drawback of 'smart technologies', such as sensor networks, is the lack of standards for both the lower network layers (i.e., physical, MAC, etc.) and

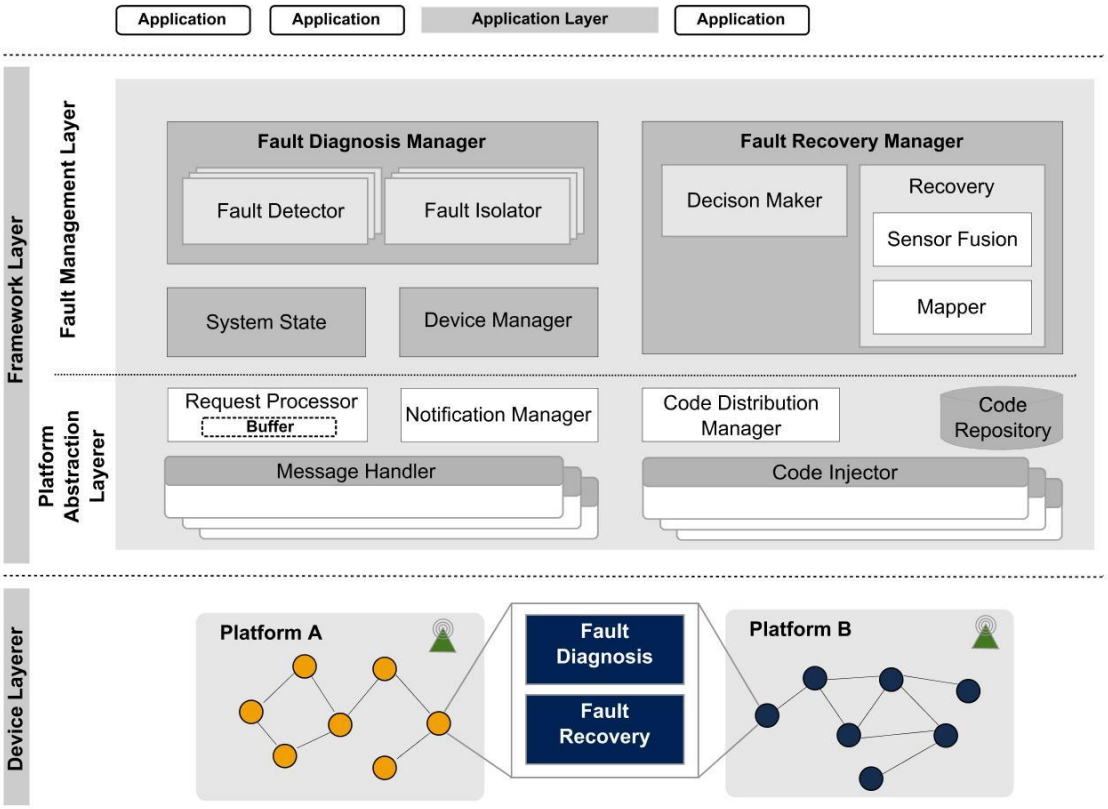


Figure 4.2: FT-WiseNets: Fault Tolerant Framework for Wireless Sensors Networks.

the higher application layers. The currently given heterogeneity of the technologies prevents their seamless integration into business applications which could profit from their functionality [122]. Therefore, we propose a generic mediating framework, called FT-WiseNets that provides the handling of different technologies on both the network and application side, and thus helps to overcome the current integration and reliability problems.

The overall architecture proposed is depicted in Figure 4.2. One of the major intentions of FT-WiseNets, and thus of the overall architecture, is to support business applications with reliable WSN services. To achieve this goal, the architecture was designed with specialized components that are responsible for enabling three main functionalities: Fault Diagnosis, Fault Recovery and Message Exchange.

The mechanisms of each of these main functionalities and the components involved in each process will be discussed in the following sections. This framework does not define specific algorithms, but rather defines the means for adopting different approaches in one integrated framework. To make the adoption of these different techniques transparent to the back-end, the framework makes use of web services since they are a standard well accepted by the industry and provide the means for the easy exchange of implementations.

4.2.1 Framework Layers

The architecture of this framework and all its components is depicted in Figure 4.2. This architecture is divided into three layers: *Device Layer*, *Framework Layer* and *Application Layer*.

In the *Device Layer*, different sensor network platforms operate and provide constant information, such as sensor readings and business processes related events.

The *Framework Layer* contains all components of the framework that run in the back-end and is sub-divided in two layers: *Platform Abstraction Layer* and *Fault Management Layer*. The former provides mechanisms for coupling heterogeneous WSNs with the back-end. The latter handles the failures generated in the WSN.

Finally, the *Application Layer* contains all business applications that make use of the functionality provided by the WSNs.

4.2.1.1 Device Layer

The device layer consists of all devices involved in the process of acquiring environmental data, and devices responsible for converting wireless sensor packages into packages that can be propagated into the LAN of the industry. In classical industrial WSNs, such as particle computers [169], these devices include wireless sensor nodes and bridges.

The logical separation of this layer from the other layers of the system is one technique applied in this framework to achieve a transparent integration of heterogeneous WSN platforms.

As some hardware platforms may provide in-network algorithms for fault management, these techniques are included in this layer. It must be pointed out that not all hardware platforms can support fault management in the *Device Layer* and therefore the framework cannot rely on specific techniques applied at this level. Nevertheless, when present, such techniques can improve the overall performance of the system. Furthermore, they also provide additional information to the *Framework Layer*, and thus, are considered as an optional part of the framework.

4.2.1.2 Platform Abstraction Layer

Due to the constraints imposed on WSNs (i.e., low processing power, restricted power supply, etc.) proprietary communication protocols that try to maximize the use of the available resources in the network are commonly applied. As a result, a great variety of protocols can be expected to be applied when adopting different hardware platforms in business scenarios.

Ziggbee [2] and Bluetooth [14] represent efforts to overcome the heterogeneity of protocols used in WSN. Although the need for these standards is evident from the industry's point of view, not many platforms have adopted them so far. Therefore, when applying WSNs in industrial environments, it can be expected that several protocols will have to be supported by the framework.

As a solution to this challenge we propose the use of a *Platform Abstraction Layer*, which together with the *Device Layer* provides the means for integrating heterogeneous WSNs with enterprise applications. In the *Platform Abstraction Layer* the components interact in order to perform the conversion between the protocol used in the back-end and the one used in the WSN.

In addition, the *Platform Abstraction Layer* provides transparent access to all functionality of the WSN. For instance, it provides the mechanisms for enabling message exchange between the WSN and the backend. The mechanisms applied include the distribution of events, and synchronous and asynchronous invocations. Synchronous invocations provides instant connection to sensor nodes, while asynchronous invocations enable the possibility of buffering requests for future invocations. This is especially interesting for scenarios where nodes can stay out of reach and be online again for some period of time.

The message exchange functionalities adopted in this thesis were developed during the CoBIs project [26] and further integrated with the FT-WiseNets framework. In Appendix A, we provide a concise description of this integration.

4.2.1.3 Fault Management Layer

WSNs applied in enterprise scenarios require a high level of availability to ensure the correct flow of business processes. Hence, to prevent failure occurring in the *Device Layer* from propagating to the *Application Layer*, the framework proposes the use of the *Fault Management Layer*.

The *Fault Management Layer* handles the failures that occur at the *Device Layer* level. The two main functionalities of the components in this layer are the diagnosis of failures and automatic failure recovery.

To support these functionalities the components provide the information about the network, maintain a database with all the events generated in the network and contain detailed information about the hardware of sensor nodes present in the network.

The information about the sensor node hardware is especially important for automatic fault recovery. This information is used when guaranteeing the coverage of

certain functionalities through the use of role assignment of nodes. Automatic recovery mechanisms also include the restructuring of the routing tree [155], and sensor fusion [35][111].

Not all failures can be automatically recovered. Nevertheless, the *Fault Management Layer* can trigger maintenance workflows for manual maintenance providing more detailed information for the maintenance workers. This can help reduce maintenance time and costs.

4.2.2 Fault Diagnosis

In this framework fault diagnosis is performed in two different layers: *Device* and *Fault Management*.

To diagnose failures in a platform independent manner, the architecture proposes the use of two components in the *Framework layer*: *Fault Detector* and *Fault Isolator*. Since different algorithms can be implemented for fault detection and isolation, the framework only defines the interfaces and intended interaction between components.

4.2.2.1 In-Network Fault Diagnosis

In-network fault diagnosis algorithms seek to analyze the behaviour of sensor nodes within the network. These approaches diagnose the malfunctioning parts through interaction between nodes and through self diagnosis, as described in 3.3.1.

An advantage of diagnosing failures at the *Device Layer* level is that failures are processed at the lowest layer of the system. This implies that there is a greater chance of mitigating misbehaviour before it reaches the back-end system, a potential for better scalability, and a better reaction time.

Nevertheless, the diagnosis of in-network failures relies on the fact that specific software will be available on the nodes. As this requirement cannot be imposed on the platforms adopted in industrial environments, the FT-WiseNets framework does not rely on this feature to diagnose failures. However, when available, the framework uses the information generated by the WSN to reduce the load of diagnosing malfunctioning parts in the back-end.

4.2.2.2 Fault Detector

Although diagnosing faults at the *Device Layer* has its advantages, it imposes a strong requirement on the hardware platforms that can be adopted, which is to support the fault detection algorithms. As this goes against the *"Support to heterogeneous WSNs"* requirement defined in section 4.1, in this framework we consider

that in-network fault diagnosis algorithms can be coupled with the framework to improve its performance. Its presence, however, is not a requirement for the diagnosis process.

When in-network fault diagnosis algorithms are not present, the *Fault Detector* component situated in the back-end is responsible for the constant evaluation of the data generated by the WSNs. This component generates symptom signals that serve as input for a second evaluation performed by the *Fault Isolator*, where the exact failure is identified.

4.2.2.3 Fault Isolator

The *Fault Isolator* component is responsible for evaluating symptom signals generated by the *Fault Detector* component. The result of this evaluation indicates the failures that probably occurred in the system. Different approaches can be applied for the isolation of failures, for example: rule inference, fuzzy evaluation of symptom signals, fuzzy diagnostic inference and pattern recognition based on neural networks.

Each approach has its advantages and ideal application scenarios where its performance overcomes the performance of other methods. Therefore this framework allows for a transparent integration of various implementations of this component. This is achieved by defining a web service interface and defining the interaction with the different components.

The isolation process relies on the symptom signals received. Therefore, the reception of specific signals is performed via event distribution. This ensures that one implementation of the *Fault Detector* will be able to provide signals to different implementations of the *Fault Isolator* component.

4.2.2.4 System State

WSNs applied in business processes have the potential to generate a high amount of events. These events contain information about the environment, the business process and also about the node itself.

In order to store all these events for further analysis, the *System State* component is registered at system start-up with the *Notification Broker* component in order to receive all events generated by the WSN. This information is then stored into a database.

The stored information comprises of:

- Node unique identification number

- Hardware platform
- Node status
- Events history
  - Timestamp
  - Event type
  - Value

Focusing on the support of different hardware platforms, the format of the sensor node ID depend on the hardware platform supported. We assume that the node id provided by the hardware platform is unique within that platform. Therefore, the unique identification of sensor nodes is based on its id and its hardware platform.

The status of a sensor node is a piece of information that is frequently requested by many applications. The *System State* component stores the status of all nodes, considers them as active at start-up, and updates their status based on events received. These events consist of failure reports generated by the *Fault Diagnosis Manager*, and recovery events generated by the *Fault Recovery manager*.

4.2.2.5 Component Interaction

The automatic diagnosis of malfunctions is a key requirement for successful deployment of WSNs in business processes. The FT-WiseNets framework proposes to enable this automatic diagnosis of failures through the use and interaction of components located in two layers of the system. The interaction of these components can be observed in Figure 4.3.

During start-up, the *System State* and *Fault Isolator* components register themselves at the *Notification Manager* (1). The *System State* registers for receiving all events that are generated by the WSN, while the *Fault Isolator* registers itself for receiving events generated by the *Fault Detectors*. This step makes use of the *Notification* message exchange pattern as described in section A. The events are generated by the WSN and stored in the *System State* component (2-4).

The diagnosis process starts with the detection of failures followed by a fault isolation process where the failure is precisely identified. The *Fault Detection* component analyzes the information available in the *System State* (5) and, when it identifies a symptom, it announces it as an event to the Notification Manager (6).

The *Fault Isolation* component based on the received information (7) determines the type of failure that has occurred and triggers a fault event (8).

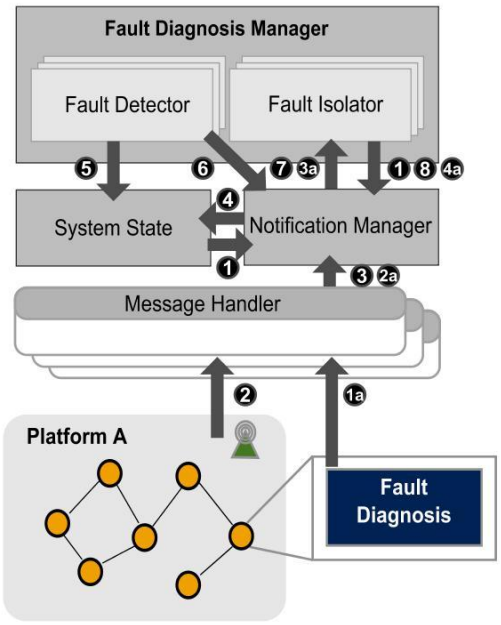


Figure 4.3: Components interaction: fault diagnosis.

When in-network fault diagnosis algorithms are available, once a failure is identified, this failure is propagated as an event directly to the *Fault Isolator* component (1a-3a). With this information, the *Fault Isolator* can define which failure has occurred and announce it to interested parties (4a).

4.2.3 Fault Recovery Management

WSNs are becoming an increasingly attractive solution for enterprise scenarios. The deployment of such devices in a running business process has to consider the risks involved in the adoption of this new technology. This includes the costs associated with breakdown states. Hence, the framework proposed provides efficient methods not only for failure diagnosis, but also for system recovery.

To reduce maintenance time, automatic recovery techniques are applied whenever possible. For a more efficient recovery time and a better integration with back-end systems, as soon as a failure is identified and automatic recovery techniques are not applicable, maintenance workflows are triggered to warn workers of occurring failures.

In this framework, the recovery process is distributed through three layers of the system: *Device*, *Platform Abstraction* and *Fault Management*.

4.2.3.1 In-network Recovery

In-network fault recovery methods running at the *Device Layer*, provide the means for WSNs to self-heal in the presence of failures. Fault recovery techniques in this

layer of the system include fault tolerant sensor fusion techniques [35][111], role assignment [46] and leader election as applied in LEECH [58]. These in-network can extend the life-time of the network and provide a precise sensor reading even if failures are present.

Most of these approaches are designed for specific routing structures. In some cases these approaches are dependent on the routing protocol applied. Nevertheless these techniques present high performance in terms of response time, since they operate directly at the point where the failure occurs.

4.2.3.2 Decision Maker

Self-healing WSNs are an attractive solution for handling fault management due to their performance and transparent approach. Nevertheless, when not present, alternative approaches have to be supported by the framework in order to ensure the availability of the services provided by the WSN. Therefore, the FT-WiseNets framework provides support to recovery mechanisms in the back-end.

The *Decision Maker* component is part of the group of components located at the back-end, which supports fault recovery. It receives information on the malfunctioning parts of the system and decides which action has to be taken in order to recover the failure.

Depending on the situation, the *Decision Maker* can opt for an automatic recovery, or can trigger a maintenance workflow. Maintenance workflows are triggered if no other node can take over the task of service provisioning. An example for such a case is a failed node responsible for reporting the room temperature, where no other node equipped with a temperature sensor is present in the same room.

4.2.3.3 Recovery

The main functionality of the *Recovery* component is to provide automatic mechanisms for bringing the system back to a stable state. This kind of automatic recovery is only possible in some specific cases:

- Another node is able to take over the service provisioning for the failed node. This is a typical recovery mechanism for ensuring the coverage of a given service within the WSN.
- Sensor fusion can be applied to generate a reliable sensor reading, even if some nodes provide outlier readings.

- Routing structure can be updated in cases where nodes become isolated but are still within the range of nodes that have a path to the sink.

Focusing on the above mentioned cases, in this framework we define the interface of this component in order to allow the implementation of mechanisms for automatic recovery. Since the coverage of services may require reprogramming of sensor nodes, the FT-WiseNets framework defines an additional set of components that together provide the required infrastructure for granting the coverage of services: *Mapper*, *Device Manager*, *Code Distribution Manager*, *Code Repository*, *Code Injector*.

4.2.3.4 Mapper

The main task of the service deployment process is to prepare the network to be used by different applications [161]. In this preparation special requirements need to be taken into account. These specific requirements are mainly related to the (spatial) dynamics of the scenarios [6] and to resource constraints.

Nodes can leave and join the network at a given location, such as storage areas, in an ad-hoc manner. When leaving the network, for instance due to the transportation of goods, the missing services have to be replaced, i.e. deployed to other still available nodes. In addition, due to the usage of (heavily) resource constrained devices, hardware capabilities must also be considered.

One necessary input is data that reflect the requirements (e.g., minimum available memory, requested bandwidth) that nodes have to fulfill to host services. Apart from the capabilities of single nodes, network related constraints, such as coverage, are also of interest.

The process step called Mapping is the most important building block within the automated service deployment. Based on the above service requirements and available network and node resources, the mapping process has to provide a set of decisions regarding which nodes have to run which service(s).

The selection of nodes is the main task of the *Mapper*. In the optimal case, during service mapping, a feasible configuration can be found that fulfills all the service and network level requirements by using available resources. However, in most cases, service mapping might be unsuccessful due to the lack of available resources. In some cases, even with the removal of services, it might be impossible to deploy the services on the required coverage specified. In that case the, *Mapper* may either deploy the service on the available coverage or not. This depends on the policy defined for the decision process.

4.2.3.5 Device Manager

As described in section 4.2.3.4, the mapping process requires information about the hardware platforms in order to make the appropriate selection of nodes for the deployment process.

The information necessary for this process is stored in a database, which is administered by the *Device Manager* component. This database contains the information about the sensor nodes and hardware platforms used by the system. This includes the ID of registered nodes, CPU type, available sensors, amount of memory, battery type, radio frequency and protocols used for communication. In this component, only static information is stored. Dynamic information has to be retrieved from the *System State*.

The *Device Manager* provides a foundation for applications, such as inventory tracking (which is located in the application layer). These applications are able to use the *Device Manager* to retrieve information about the nodes in use.

4.2.3.6 Code Distribution Manager

The *Code Distribution Manager* provides a uniform interface for the *Mapper* component to forward deployment descriptions. This component is the central contact point for service invocations regarding service lifecycle management. It provides an interface for platform independent service deployment and service lifecycle management.

The main functionality of this component is to select the appropriate *Code Injector* to perform the deployment requested by the *Mapper*. An invocation to this component includes the node address, from which the appropriate platform specific *Code Injector* is selected. Additionally, a deployment description generated by the *Mapper* component may request reprogramming several nodes from different platforms. It is the task of the *Code Distribution Manager* to ensure a proper execution of the deployment description among the hardware platforms.

4.2.3.7 Code Repository

The Service Repository contains a database holding descriptions of available services, including their description, deployment requirements, and implementations. It is intended to be used as the ”reference point” for the available services in an enterprise environment. This means, that every service being executed within the network of nodes is represented by an entry in the service repository.

4.2.3.8 Code Injector

After successful mapping of services to nodes and the allocation of nodes' resources, the service implementations have to be transferred to the corresponding nodes. This step can be implemented in different ways.

One option is to establish and use a centralised network-wide code injection component that acts as a master, by giving nodes the corresponding instructions (e.g., 'Remove Service A and deploy Service B.'), and sending the code (e.g. the implementation code of the new Service B) that has to be deployed and started.

In the case of peer-to-peer sensor networks, an optimization is to provide specific (code) dissemination services that autonomously run on nodes. Thereby, nodes could 'infect' all their relevant neighbours with the new service code.

Different approaches and algorithms can be used for this step of the deployment process. The selection of one method by the framework would impose a strong restriction on the hardware platforms supported. Therefore, different mechanisms are supported through the implementation of different *Code Injectors*.

4.2.3.9 Component Interaction

The constant monitoring and maintenance of WSNs applied in business processes is a key factor to ensure the availability of services offered by WSNs. As presented in Figure 4.4. failure recovery in this framework is achieved through the interaction of components distributed in three layers of the system: *Device*, *Platform Abstraction* and *Fault Management*.

The recovery process starts when failures are isolated by the framework. The *Decision Maker* receives the failure event (1), and evaluates whether automatically recovering the failure is possible (2), or if a workflow requesting maintenance has to be triggered (2a).

For automatic recovery, sensor fusion can be applied to recover the system from value failures (3b). Coverage can be guaranteed through the reassignment of sensor nodes roles. In such cases the *Mapper* evaluates the requirements of the services that need a guaranteed coverage in order to generate a deployment description. The *Code Repository* provides the meta-data that indicates the service requirements (3). The *Device Manager* supplies the information regarding node characteristics (4), while the *System State* component indicates the status of the sensor nodes (5). With this information the *Mapper* component produces a deployment description that indicates which nodes should have their role reassigned or re-programmed.

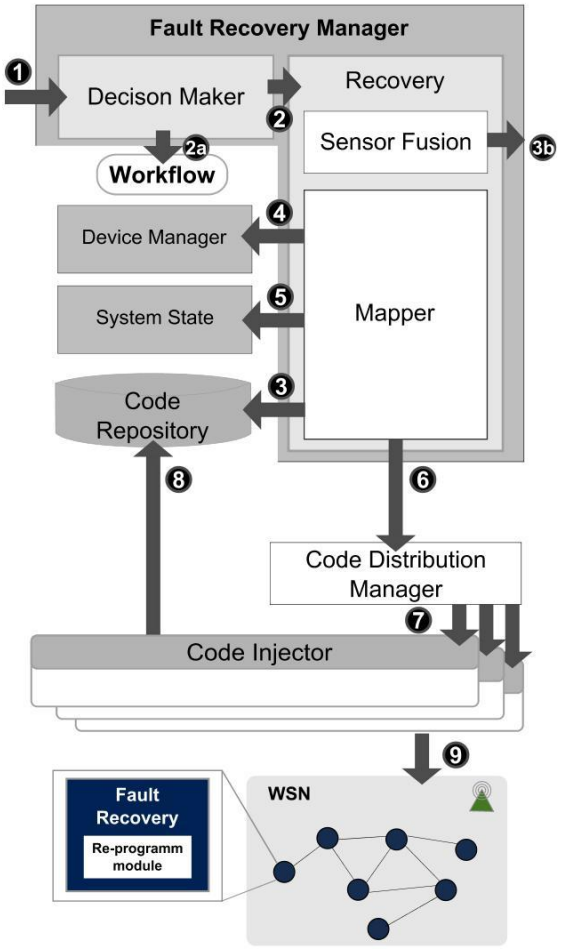


Figure 4.4: Component interaction: fault recovery.

The *Code Distribution Manager* receives the deployment description (6) and coordinates its execution by forwarding parts of it to designated platform specific *Code Injectors* (7).

Finally, when necessary, the *Code Injector* downloads the code implementation from the Code Repository (8) before executing the deployment (9).

4.3 Summary

As a result of the analysis of existing fault tolerant techniques investigated in Chapter 3, we have identified that none of the current solutions are suitable for enterprise environments. Hence, this work proposes an approach to overcome the challenge of WSNs heterogeneity and integration with backend systems. This framework was designed in order to satisfy the requirements defined in Section 4.1:

-Transparency: As discussed in Chapter 3, transparency has not been the focus of fault tolerance research for wireless sensor networks so far. Most approaches propose platform dependent solutions and fail to provide transparent access for applications

to heterogeneous hardware platforms. FT-WiseNets proposes a *Platform Abstraction Layer* where a set of components interact to offer transparent message exchange to heterogeneous WSNs, based on web services .

-Extensibility: Current solutions for fault management in WSNs only support specific algorithms for fault diagnosis and recovery. The use of web services as a standard communication among components allows for extensions on the framework. This extension is achieved through the implementation of new techniques for the already defined service interfaces (e.g. Fault Detector and Fault Isolator).

-Support to heterogeneous hardware platforms: The framework proposed has a separation of the system in layers, which enables the *Application Layer* to abstract the services offered by WSNs from the hardware platforms adopted. Enabling the adoption of heterogeneous platforms in a business scenario goes beyond supporting a variety of communication protocols. It is also necessary to ensure that the framework does not impose additional requirements on the wireless sensor nodes. Thus, special attention was given to the fault diagnosis and recovery mechanisms.

-Automatic fault diagnosis: The diagnosis of failures can be executed in two layers of the system: *Device* and *Fault Management*. In-network diagnosis of failures is one of the approaches supported by the framework. This solution, however, imposes additional requirements on hardware platforms, which go against the requirements defined. Therefore, the framework supports centralized fault diagnosis. Automatic fault diagnosis algorithms located at the back-end are still a challenge not yet completely solved for WSNs. Nevertheless, such an approach is valuable to correctly identify failed nodes without reducing the lifetime of the sensor network nor impose the requirement of in-network fault diagnosis and recovery algorithms to hardware platforms. In Chapter 5 we present an approach that overcomes this difficulty.

-Automatic fault recovery: Maintenance in business processes can become very costly especially when break-down situations occur. Through automatic recovery mechanisms executed in the *Device* and *Framework* layers, the architecture offers the means to reduce maintenance costs. Additionally, the coupling of fault management mechanisms with the back-end through the use of workflows can lead to reduced maintenance times.

-Delivery of online data: Business processes require online information about assets and business items that belong to the system. The proposed framework provides the mechanisms for accessing data from WSNs in three different manners: via request-response, notification and through the *System State* component. This

component contains all the events generated by the network of nodes and enables applications to access the latest data generated by wireless sensor nodes.

5. Pattern Fault Isolator for WSNs

As discussed in chapter 2, WSNs have many attractive benefits to industrial applications. Nevertheless, its malfunctions and breakdown states can result in many losses. Therefore the identification of malfunctioning parts of a WSN must be efficient to mitigate undesirable effects.

The variety of scenarios in which WSNs can be applied leads to a natural selection of different platforms that are more suitable for each application. This results in a broad range of WSN platforms adopted in an industrial environment. Hence, the process of isolating malfunctioning parts of a WSN should not only be efficient, but also independent from any hardware platform. For the same reason, enterprise systems cannot support specific on-node fault detection software functionality as proposed in approaches similar to [88][37][135]. Structural information, such as routing topology is also specifically linked to the functionality of a WSN and is therefore hard to generically exploit for fault diagnosis by enterprise systems.

In this context, this thesis investigates the research question on how feasible is fault isolation with a minimum, but for most platforms, generic set of information. We approach this question by proposing a fault isolator mechanism, which only relies on three basic information quanta: sensor node identification number, timestamp information, and heart beat interval. We believe that this information set could be provided by any WSN regardless of its specific implementation and therefore inherently forms an appropriate WSN platform abstraction for enterprise business systems.

The proposed solution is part of the FT-WiseNets framework [30], and fulfils the task of the fault isolator component. To evaluate our system, we performed an

application trial for a period of three months where 36 nodes and 5 sinks were constantly monitored. The results of this trial are discussed in chapter 7.

5.1 Failure Model

In an enterprise environment, it is important to support heterogeneous hardware platforms [30]. Figure 3.1 in Section 3.1.1 presents a generic model where the main components of the WSN are illustrated. We assume that any of the components of this model (nodes, sinks and back-end) can suffer crash failures and wireless links can suffer wireless interference.

In such scenarios where heterogeneous WSNs are in place, we cannot assume that all platforms will support pre-defined set of functionalities. Such functionalities include: complete route structure information available to the back-end system, sink addressed pings and diagnostic software running on sensor nodes. Therefore we assume the worst case scenario where the only information available is: sensor node identification number, timestamp information, and heart beat interval.

Although this is a very restricted set of data, several aspects of the WSN can be evaluated using this information as input. For instance, it is possible to detect outlier sensor readings by comparing the readings from different nodes and defining rules for the expected values. Strong reductions in the number of messages (message throughput) generated by sensor nodes can also be identified giving indications of problems in the routing path (i.e. crashed nodes and wireless interference).

The challenge appears when we try to identify failures of sinks and back-end only based on this information.

For our approach we have focused on the isolation of failures based on the symptoms detected by the message throughput of each wireless sensor node. We assume a static network where nodes are not mobile and the routing structure does not suffer modifications. Since timeouts can be recognized, we assume that the network will provide periodic information about sensor readings.

5.2 Crash Fault Analysis

By analyzing the model described in section 3.1, we can make a classification of the observed node failures F_o that occur in a WSN. It is important to differentiate between the set of node failures that are observed (F_o) and the set of real failures (F_r), because in many cases $F_o \neq F_r$. The set of real failures that occur in the system F_r indicates the components that are malfunctioning in the system.

As we can only identify crash and omission failures of nodes based on timeout, a sink failure (F_r) will appear as several node failures (F_o). In a single path system, if a node n_i suffers a crash failure (F_r) and $\lambda_{ni} > 1$, several parts of the network are observed as failed (F_o).

In this classification, the observed failures F_o are separated into three different groups:

Global: All nodes appear as failed. In this situation $F_o = \{N\}_{Fo} = \{N\}$.

Partial: Only parts of the system appear as failed while other parts continue their normal operation. In such case $F_o = \{N\}_{Fo} = \{n_{o1}, n_{o2}...n_{of_o}\}$, where f_o is the number of observed failed nodes and $f_o < \eta$.

Single: Only one node indicates malfunction. For this scenario $F_o = \{N\}_{Fo} = \{n_{o1}\}$.

In addition, we separate the real failures into *Isolated Failures* and *Composed Failures*. Isolated failures occur when only one component of the system fails, while composed failures occur when more than one component fails in conjunction. The latter case represents a set of cases which are harder to identify and that have not been deeply investigated by existing solutions so far.

The components of the system that are subject to failures can be divided in four categories [30]: Node, Network, Sink and Back-end.

The set of failed nodes in this context is defined as $F_{rN} = \{n_{f1}, n_{f2}, ..., n_{f\eta f}\}$, where η_f is the number of failed nodes. The set of failed sinks is defined as $F_{rB} = \{\beta_{f1}, \beta_{f2}, ..., \beta_{fwf}\}$ where w_f is the number of failed sinks. The set of failed paths is defined as $F_{r\rho} = \left\{P^B_{\rho f1}, P^B_{\rho f2}, ..., P^B_{\rho fp}\right\}$ where p is the number of failed routes and ρf_x is the starting point of the failed path. The failure caused by a back-end crash is defined as B.

Based on the model described in section 3.1.1 and the concepts provided in this section, it is possible to perform a static analysis of WSNs to define a possible source of failures (F_r) and their conditions, given the observed failures (F_o).

Figure 5.1 depicts this crash failure analysis. In this classification, observed failures are separated into: global, partial and single. Each box inside this table represents a possible real failure (F_r). The real failures are divided into two main groups: isolated and composite failures. Each real failure has two columns: one that expresses the source (the real failure F_r), and one that indicates the conditions that must be fulfilled.

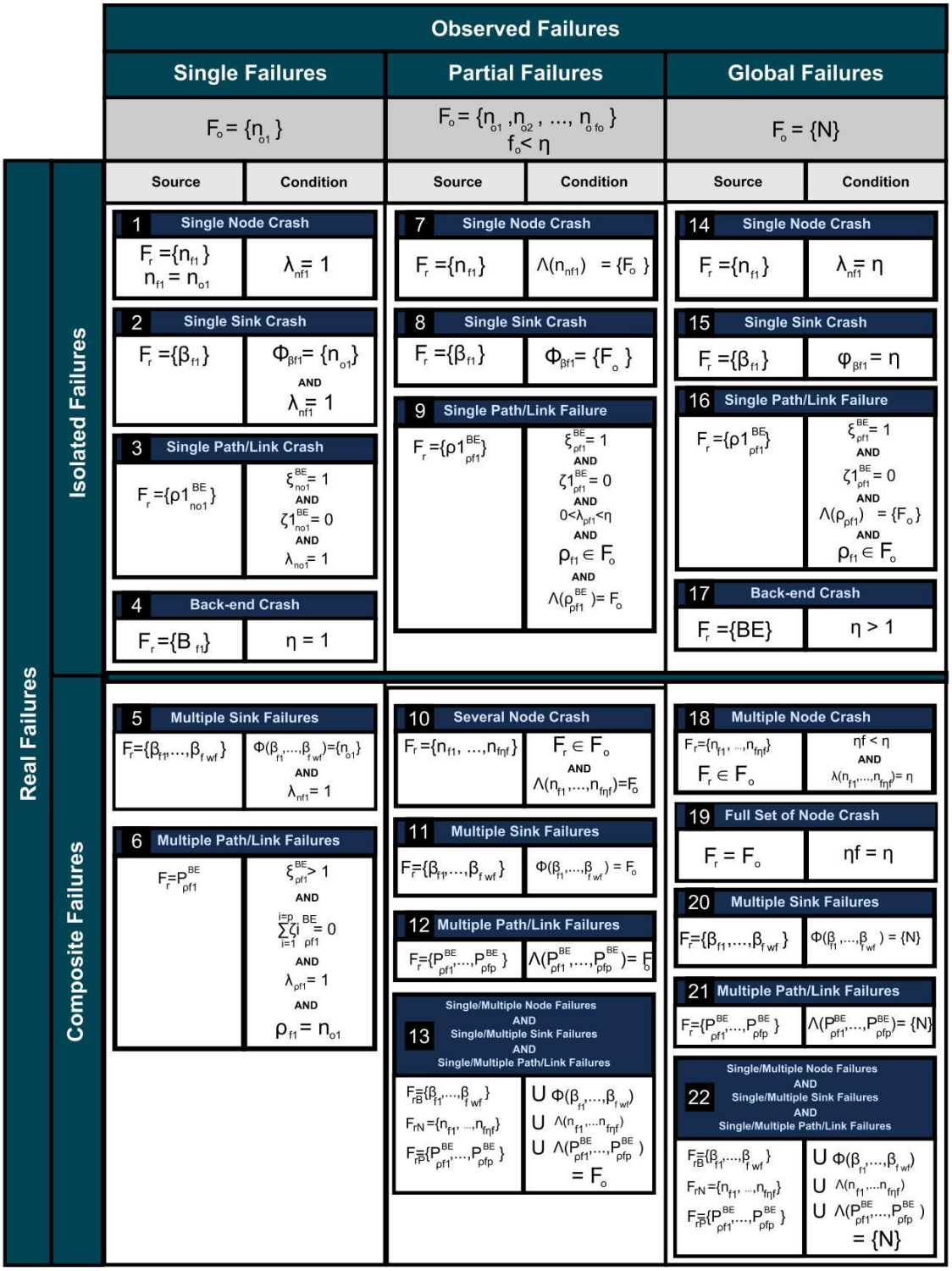


Figure 5.1: Fault classification.

These conditions need to be satisfied in order to co-relate (F_r) to (F_o). For example, when a node stops reporting its measurements to the back-end, one of the possibilities is that the node itself has crashed. In this case λ_{nf1} must be equal to one, otherwise the other nodes would also be observed as failed. This case represents the *"Single Node Crash"* box in the *"Single Failures"* column (box 1).

In some situations all the conditions of one or more real failures can be met. In the example of the single observed node crash, it is also possible that a sink failed. This case is represented by the *"Single Sink Crash"* box and with the condition $\Phi(\beta f1) = \{n_{o1}\}$ fulfilled (box 2).

In this case a pure rule based approach, as the one presented here, is not enough to isolate failures in WSNs and resolve ambiguity.

Additionally, this table demonstrates that the process of isolating failures, based on the static analysis of the system and its components, frequently requires the knowledge of the routing topology. As discussed in section 5.1, this information is not part of the restricted information set that is available to the back-end system. Therefore, an alternative solution is required to isolate failures in WSNs deployed in industrial environments.

5.3 Architecture

In some cases, it is sufficient to detect the crash of individual nodes based on timeout information. However, the information of how individual nodes fail may be important not only to help determine which part of the system is in fact malfunctioning but also to decide optimum actions to be taken for maintenance. For instance, the sudden failure of numerous nodes can indicate that an essential part of the system has failed (back-end, sink, node in the routing path), or a substantial reduction in the message throughput of sensor nodes from a region can give indications of wireless interference.

To distinguish between the different possible sources of failures, having only a restricted set of information available for the back-end system, we propose a solution based on failure pattern recognition. We selected the message throughput of each sensor node as the diagnostic signal ($\overrightarrow{S}$) for the isolation analysis. This information provides relevant information not only about the health state of components between the node and the back-end, but also provides indications of wireless interferences that occur in the routing path.

The goal of the proposed approach is to be able to identify single and partial failures originated from composite failures involving sinks, nodes, and links failures. Boxes

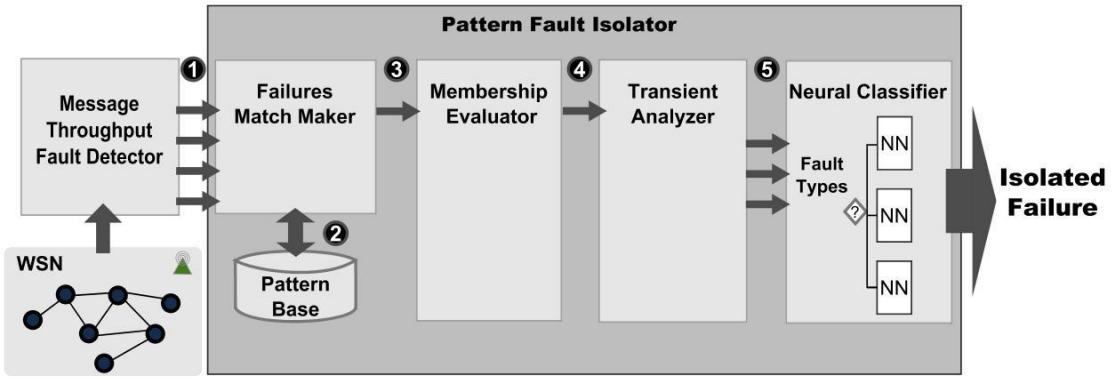


Figure 5.2: Pattern Fault Isolator.

1 to 13 in Figure 5.1 present the failures targeted by the fault isolation approach proposed in this thesis.

Figure 5.2 depicts the approach proposed in this thesis. During operation time, the WSN will generate events, which will be stored in the *System State* component of the FT-WiseNets framework. The first stage of the diagnosis process starts with the generation of the diagnostic signal ($\overrightarrow{S}$). This signal is the message throughput of each sensor node and is calculated by the *Message Throughput Fault Detector*. This value is calculated taking into consideration a specific time frame ($number of events/time frame$) and serves as input for the *Pattern Fault Isolator*.

Based on the diagnostic signal provided, the *Pattern Fault Isolator* will determine the source of the current failure occurring in the system. To achieve this, the *Pattern Fault Isolator* relies on six main components: *Rules Verifier*, *Transient Analyzer*, *Probability Evaluator*, *Failures Match Maker* and *Neural Classifier*.

5.3.1 Choice of Techniques

Different techniques exists for the process of recognizing patterns, as described in Chapter 3. Each technique has its advantages and drawbacks. Hence, the solution proposed here explores these approaches, applying them according to their strengths.

Rule based approaches, designed for a generic WSN model, are not able to differentiate between several failures that occur in the system based only on the message throughput of sensor nodes. This drawback occurs because the information about the network topology is vital to evaluate the conditions presented in section 5.2. Nevertheless, this approach is very efficient when simple rules must be verified. For instance, it is better to identify if the message throughput of a sensor node is greater than zero, in order to eliminate the possibility of a node crash, than to apply a neural network to identify the pattern.

Probabilistic approaches are very helpful to estimate the possibility of a diagnostic signal belonging to a known pattern. The drawback of this approach appears in the creation of a *Pattern Base* containing all possible states of the system. WSNs deployments in business scenarios can include thousands of nodes. Each node may fail generating a specific system state. Composite failures also consist in a different system state. This results in a combinatorial explosion of system states that should be avoided.

A similar problem occurs when we apply only neural networks for the recognition of all failure patterns. Although neural networks are extremely efficient for recognizing patterns, its efficiency considerably decreases according to the training set size and according to the number of patterns that must be identified [31].

The combination of these techniques, however, results in a high performing fault isolation solution (this is analyzed in Chapter 7).

5.3.2 Pattern Fault Isolation Process

The *Pattern Fault Isolation Process* executes the logic in order to correctly identify malfunctions in the system. Figure 5.3 presents the algorithm executed by this component.

The first step is to verify if the diagnostic signal $\overrightarrow{S}$ contains failures. Failures are identified if nodes present a message throughput equal to zero, or if the probability evaluator indicates that the readings do not belong to the pattern acquired for normal operation.

Since our system recognizes the back-end as one unique component, once the first step of the fault isolation indicates a massive failure of all sensor nodes in the system occurring within a short time frame, a failure of the back-end is identified and no further action is required. The other failure types, however, may have multiple components and it is necessary to further analyze the data to identify the malfunctioning part.

In the following step, the *Pattern Fault Isolation Process* forwards the diagnostic signal $\overrightarrow{S}$ to the *Failures Match Maker* component. This component analyzes the current diagnostic signal and returns a set of possible failure types. These failure types consist of patterns stored in the *Pattern Base*, combined with node crash and link failures. Known system patterns that represent a direct match with the diagnostic signal, without additional node crash and link failures, are also returned in the set of possible failure types.

In addition to the returned set, a vector is generated containing the observed failures. This information is important, since it serves as an indication to maintenance workers of possible malfunctioning system parts, in case of incorrect isolation of failures by the fault isolator.

The vector containing the observed failures is calculated through a simple comparison of the diagnostic signal $\overrightarrow{S}$ and the minimum expected message throughput of each node during normal operation. A final analysis is performed by the *Transient Analyzer* to indicate which failure occurred due to node crash or link failure.

According to the number of results returned by the *Failures Match Maker*, the *Pattern Fault Isolation Process* decides the next steps to be taken.

If only one result is returned by the *Failures Match Maker*, this will be the result of the isolation process. Otherwise, if multiple results are returned, the *Neural Classifier* performs a final classification giving a score to each of the possible failure matches.

Since the data available for training the neural networks only contains specific failure types acquired during the pattern acquisition phase, it is not possible to use the trained neural networks with the diagnostic signal, which contains combined failure types. Therefore, the *Failures Match Maker* generates a modified diagnostic signal. In this signal, the message throughput value from nodes that are considered as failed for the given pattern are replaced. The value used for replacement is the average message throughput of the sensor node for the given pattern. This procedure is used to "simulate" a recovery of the failed node, and is described in detail in Section 5.3.6.

Once the modified diagnostic signals are generated, they are forwarded to the *Neural Classifier*. Since the number of modified diagnostic signals is equal to the number of possible failure types, the *Neural Classifier* evaluates each modified diagnostic signal. The final result of the fault isolation process is the set of possible failures with a score that indicates the number of times the *Neural Classifier* has opted for each fault type. The failure match with the highest score is the final failure result selected by the system.

5.3.3 Pattern Base

The process of recognizing patterns relies on the fact that several patterns are known to the system beforehand. Acquiring failure patterns, however, can become a complex process, since each state (Δ) of the system potentially generates a different pattern. Additionally, each deployment has its own patterns, making it impossible to acquire a *Pattern Base* once, and redistribute this base to different applications.

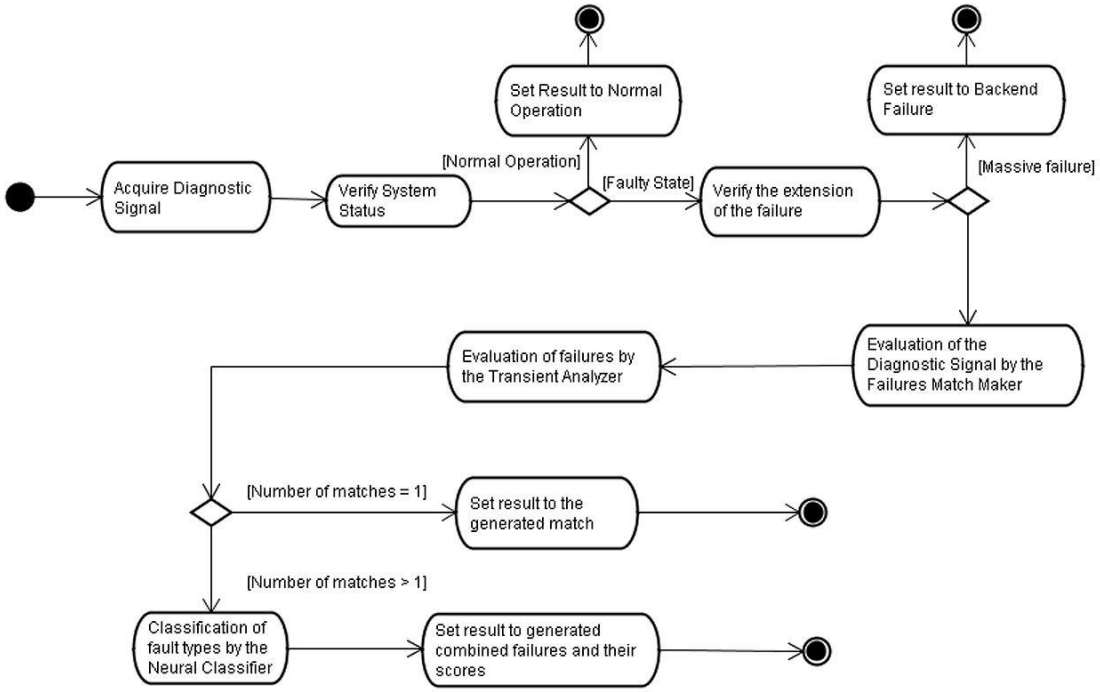


Figure 5.3: Fault isolation process.

To overcome this challenge, making this process more efficient, the *Pattern Base* only contains the system states during: normal operation, inducted sink failures, and failures that naturally occur when a business process uses the WSN. The acquisition of sink failure patterns is assisted by an application that guides the maintenance workers through the process.

Since pattern correctness directly impacts the performance of the *Pattern Fault Isolator*, it is important to ensure that only controlled failures occur in the system during data acquisition. The application offered by the FT-WiseNets framework supports maintenance workers during this process by verifying if all nodes provide a minimum level of message throughput before the sink failure is manually inserted in the system. The application also verifies if all nodes return to normal operation once the failure is recovered.

It is possible, however, that a failure occurs only during the failure pattern acquisition. Since it is not possible to distinguish this type of failure and the sink failure inserted in the system, we assume that the data acquired corresponds to the failure inserted in the system.

The patterns stored in the *Pattern Base* have the format:

- System State (Δ)
- Time frame (tf)

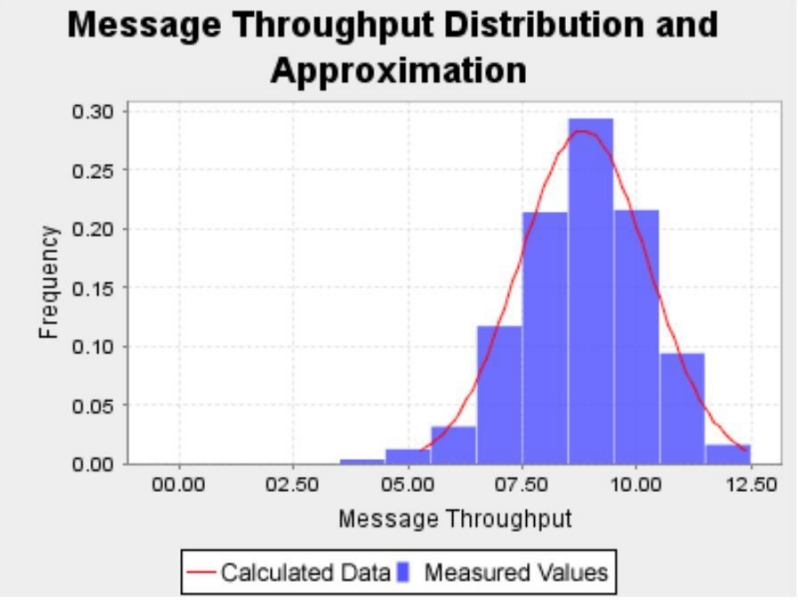


Figure 5.4: Message throughput histogram of a sensor node.

- Message throughput of each node ($m_{\Delta,ni}$)
- Standard deviation of the message throughput of each node

5.3.3.1 Message Throughput Distribution

To comprehend the probability distribution of the message throughput of each node it is important to investigate the rules that control the behaviour of the underlying system.

By analyzing the WSN model described in section 3.1.1, we can assume that based on the path quality that connects a node to the back-end, there is a probability p of the generated message reaching each sink in a given system state Δ.

Figure 5.4 depicts the distribution of the number of messages generated by a single node that reached the back-end. For this simulation we selected a network without overlap to observe the message throughput distribution for a single sink.

This discrete distribution can be approximated by a *Binomial* distribution [113]. This is justified by the nature of the system. In a given time interval the node will attempt to send a message to the backend $TimeInterval/Heartbeat$ times. The result of each attempt (the message reaching the sink or not) is independent of previous events, yielding success with a probability p.

The number of messages from a sensor node that reach the back-end, is the sum of messages from the same node that reach each individual sink. Considering that the distribution of the message throughput that reaches each sink can be approximated

by a normal distribution, the number of messages that reach the back-end will also follow a normal distribution.

This assumption is correct since the sum of two normal distributions is also a normal distribution, as presented in equation 5.1.

$$N(\mu, \sigma) + N(\nu, \tau) = N(\mu + \nu, \sigma^2 + \tau^2) \tag{5.1}$$

5.3.3.2 Combination of Known Patterns

The combination of existing patterns is based on the idea that failures will impact the amount of messages the WSN can generate.

By computing the influence of each failure and combining them, it is possible to generate a set of pattern readings that correspond to the combined failure. This computed set can then be used to isolate failures in the same manner as the acquired patterns.

Consider a j^{th} pattern of messages throughputs stored in the *Pattern Base* as a vector $\overrightarrow{M_{\Delta j}} = (m_{\Delta,n1,j}, m_{\Delta,n2,j}, ..., m_{\Delta,n\eta,j})$, where $m_{\Delta,ni,j}$ is the number of messages generated by sensor node n_i, during a time period t, for system state Δ. The matrix $\boldsymbol{M}_\Delta$ contains all the f pattern readings available in the system for the system state Δ and is represented as follows:

$$\boldsymbol{M}_\Delta = \begin{Bmatrix} m_{\Delta,n1,1} & m_{\Delta,n2,1} & ... & m_{\Delta,n\eta,1} \\ m_{\Delta,n1,2} & m_{\Delta,n2,2} & ... & m_{\Delta,n\eta,2} \\ \vdots & \vdots & \ddots & \vdots \\ m_{\Delta,n1,f} & m_{\Delta,n2,f} & ... & m_{\Delta,n\eta,f} \end{Bmatrix}$$

The mean of the available patterns provides an indication of the expected message throughput of sensor nodes for a given system state Δ (which can be a faulty state or normal operation) and can be calculated as:

$$\overline{\boldsymbol{M}_\Delta} = \frac{1}{f} \sum_{j=1}^{f} \boldsymbol{M}_{\Delta j}$$

Consider the mean of the pattern acquired during normal operation as:

$$\overline{M_{NO}} = (m_{NO,n1}, m_{NO,n2}, ..., m_{NO,n\eta})$$

The mean of the pattern acquired during a failure:

$$\overline{M_{Fi}} = (m_{Fi,n1}, m_{Fi,n2}, ..., m_{Fi,n\eta})$$

The mean influence $\overline{I_{Fi,j}}$ of a failure Fi acquired by the system is calculated as:

$$\overline{I_F} = \overline{M_{NO}} - \overline{M_{Fi}} \tag{5.2}$$

The variance of the influence of a failure Fi $\sigma^2_{I,Fi}$ is calculated as:

$$\sigma^2_{I,Fi} = \sigma^2_{NO} - \sigma^2_{Fi} \tag{5.3}$$

By summing the influences of the combined failure patterns ($CF = \{F1, F2, ..., Fn\}$), it is possible to generate a combined influence (CI_{CF}), where:

$$\overline{CI_{CF}} = \overline{I_{F1}} + \overline{I_{F2}} + ... + \overline{I_{Fn}}$$

$$\sigma^2_{CI} = \sigma^2_{I,F1} + \sigma^2_{I,F2} + ... + \sigma^2_{I,Fn}$$

Finally, the message throughput distribution of the combined pattern (CP) is calculated by subtracting the combined influence distribution from the normal operation distribution.

$$\overline{CP} = \overline{M_{NO}} - \overline{CI_{CF}}$$

$$\sigma^2_{CP} = \sigma^2_{NO} + \sigma^2_{CI}$$

Rounding Effect Analysis

The *Patter Combination Method* provides the mathematical tools necessary to generate the pattern of several failures occurring at the same time, based on the patterns of these failures occurring separately. The final combined pattern can result in negative values for the message throughput. As negative message throughput values do not reflect the real system, these values are rounded to zero.

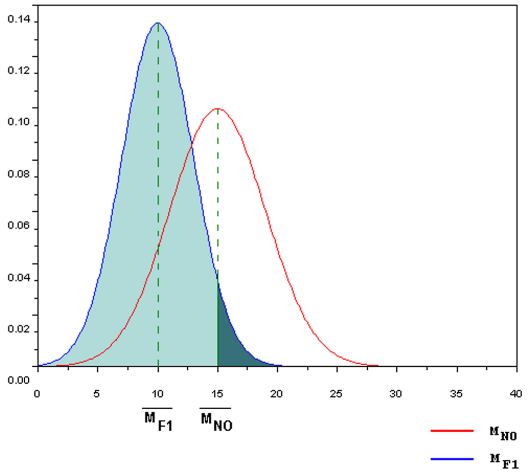


Figure 5.5: Sensor node message throughput distribution during normal operation and faulty state.

The influence of a failure in the WSN is calculated as presented in equation 5.2 and 5.3. The influence of a failure on one node will also follow a normal distribution as presented in equation 5.4.

$$I_{Fx,i} = N(\overline{M_{NO,i}} - \overline{M_{Fx,i}}, \sigma^2_{NO,i} - \sigma^2_{Fx,i}) \tag{5.4}$$

To evaluate the effect of rounding to zero negative values, we analyze the single case where only one failure is "combined". The result of such combination should be the original distribution of the message throughput during the faulty state. For this case the combined message throughput of a single node can be represented as indicated in equation 5.5.

$$CP_{Fx,i} = \overline{M_{NO,i}} - CI = \overline{M_{NO,i}} - \overline{M_{NO,i}} + M_{Fx,i} = M_{Fx,i} \tag{5.5}$$

In cases, where $M_{Fx,i}$ is close to $M_{NO,i}$, overlaps may occur as demonstrated in figure 5.5. In such cases, the distribution of the influence will also include negative values as presented in figure 5.6.

Although this may seem contradicting, it is only a reflex of the nodes message throughput during a faulty system state. Removing the negative values from the influence results in an incorrect distribution as presented in figure 5.7.

Therefore, when applying the *Pattern Combination Method* on multiple combined failures, the same principle discussed applies. Hence, the message throughput of the

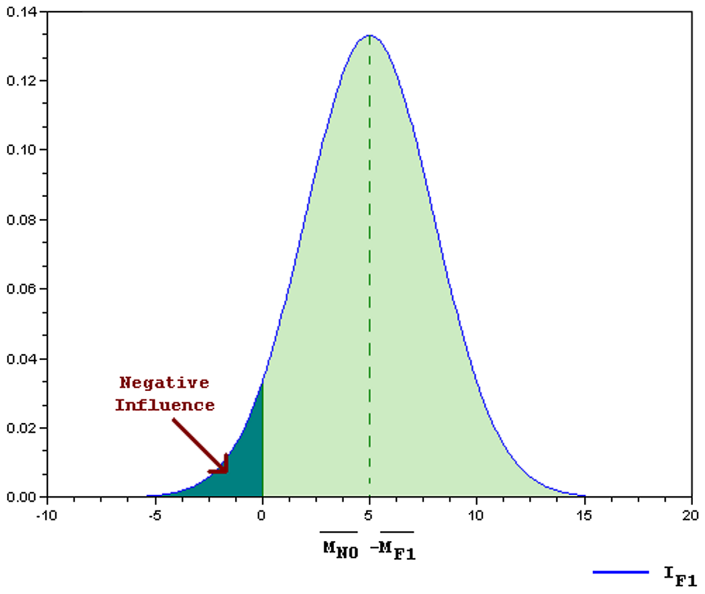


Figure 5.6: Influence including negative values.

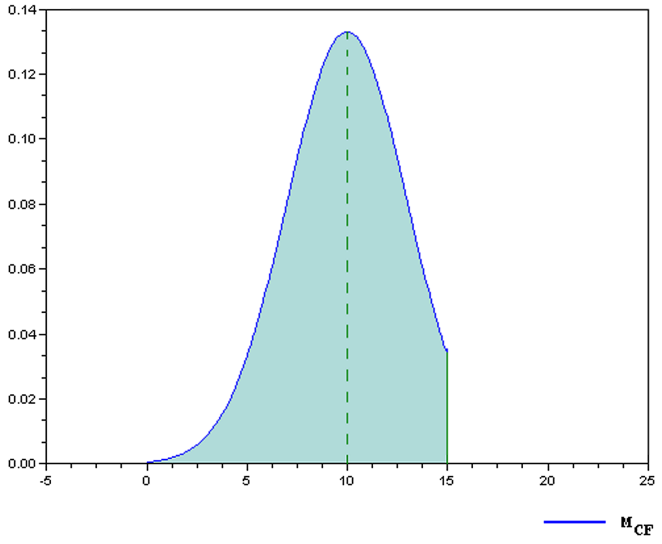


Figure 5.7: Sensor node message throughput distribution when rounding the influence to zero.

sensor node is rounded to zero if it contains negative values, only at the last step of the pattern combination technique.

5.3.4 Membership Evaluator

The *Membership Evaluator* component is responsible for evaluating if the current diagnostic signal $\overrightarrow{S}$ belongs to the patterns stored in the *Pattern Base*.

This component is the basis for the fault isolation process. It requires a metric that can be evaluated in a computationally efficient manner and that provides reliable results.

At first, the *Membership Evaluator* calculates the distribution of the patterns and the probable maximum distance a pattern can have from its mean pattern vector. Then, it evaluates the likelihood of $\overrightarrow{S}$ belonging to one of the patterns known to the system.

5.3.4.1 Membership Functions

A reliable way to determine if $\overrightarrow{S}$ belongs to a known pattern is to calculate the probability of it occurring.

Assuming a normal distribution for each node, the probability P of a given vector occurring given the normal distribution of each node's message throughput in a given pattern can be defined as presented in equation 5.6:

$$P = P_1(x_1) \times P_2(x_2) \times \ldots \times P_\eta(x_\eta) \tag{5.6}$$

Where P_i is the normal distribution of each variable x_i and has the formula presented in equation 5.7

$$P_i = \frac{1}{\sigma_i\sqrt{2\pi}} e^{-\frac{(x_i-\overline{x_i})^2}{2\sigma_i^2}} \tag{5.7}$$

The membership function based on probabilistic evaluation, defines a minimum probability P_T as the threshold to determine if a diagnostic signal belongs to a given pattern, as represented in equation 5.8.

$$MembershipFunction : P \geq P_T \tag{5.8}$$

Figure 5.8 presents the distribution of P for $\overline{x_1} = 10, \overline{x_2} = 15, \sigma_1 = 0.6, \sigma_2 = 1$.

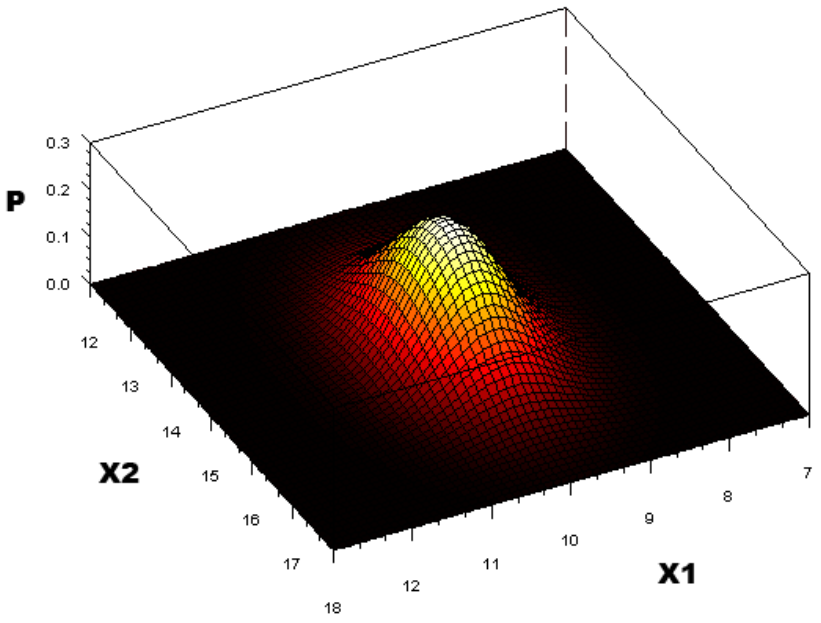


Figure 5.8: Distribution of P.

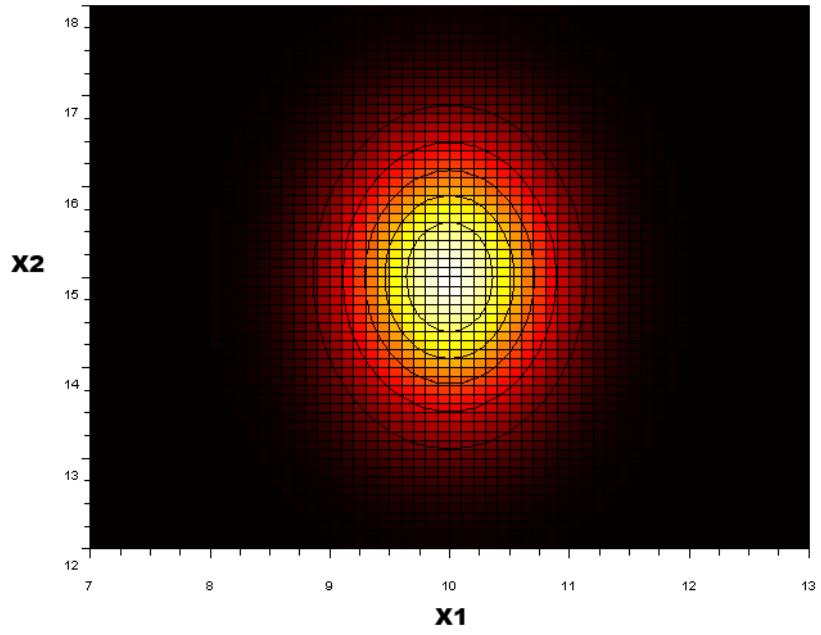


Figure 5.9: Curve level view of P.

Defining a threshold for P in Figure 5.8 results in an ellipse that defines an area on the $X_1 \times X_2$ plan where value pairs of X_1 and X_2 are accepted as part of the pattern. Figure 5.9 presents the curve levels that represent the membership areas delimited by such ellipses.

Although this metric provides reliable results, it is computationally intensive due to the exponential calculation. Therefore we approach the problem applying a different technique that does not require the exponential calculation but yields the same results.

This other technique is called *Normalized Euclidean Distance*, and is based on the Euclidean distance method [85] normalizing the values from $\overrightarrow{S}$. It defines if a vector of n variables belongs to a given pattern or not. This method defines a maximum normalized distance from the expected average to accept a diagnostic signal as part of the pattern.

Equation 5.9 presents the mathematical formula that defines this function, where x_{ni} is the normalized values of the observed variables and $\overline{x_n}$ is the expected average value of the normalized value of x_i.

$$DEN = \sqrt{(x_{n1} - \overline{x_n})^2 + (x_{n2} - \overline{x_n})^2 + \ldots + (x_{n\eta} - \overline{x_n})^2} \tag{5.9}$$

The normalization of the variables is performed based on the equality:

$$CDF(x_i) = CDF(x_n)$$

Where $CDF(x)$ is the cumulative density function. From this equality we can derive Equation 5.10

$$\frac{x_{ni} - \overline{x_n}}{x_i - \overline{x_i}} = \frac{\sigma_n}{\sigma_i} \tag{5.10}$$

Assuming the normalized form of the random variables as being the *standard normal distribution*:

$$\overline{x_n} = 0, \sigma_n = 1$$

$$\frac{x_{ni} - 0}{x_i - \overline{x_i}} = \frac{1}{\sigma_i}$$

$$x_{ni} = \frac{x_i - \overline{x_i}}{\sigma_i} \tag{5.11}$$

By applying 5.11 to 5.9, the normalized euclidean distance equation results in the formulation presented in equation 5.11.

$$DEN = \sqrt{(x_{n1} - 0)^2 + (x_{n2} - 0)^2 + \ldots + (x_{n\eta} - 0)^2}$$

$$DEN = \sqrt{(\frac{x_1 - \overline{x_1}}{\sigma_1})^2 + (\frac{x_2 - \overline{x_2}}{\sigma_2})^2 + \ldots + (\frac{x_\eta - \overline{x_\eta}}{\sigma_\eta})^2}$$

$$DEN^2 = \frac{(x_1 - \overline{x_1})^2}{\sigma_1^2} + \frac{(x_2 - \overline{x_2})^2}{\sigma_2^2} + \ldots + \frac{(x_\eta - \overline{x_\eta})^2}{\sigma_\eta^2} \tag{5.12}$$

It is important to note that DEN is only defined for values greater or equal to 0.

The *Normalized Euclidean Distance* membership function defines a threshold DEN_T, which determines if a diagnostic signal belongs to a given pattern or not. This membership function is presented in equation 5.3.4.1. As DEN is only composed of values greater or equal to zero, the equation can also be written as presented in 5.13.

$$DEN \leq DEN_T$$

$$MembershipFunction : DEN^2 \leq DEN_T^2 \tag{5.13}$$

Figure 5.10 presents the *Normalized Euclidean Distance* curves using the same values as the probabilistic approach for the normal distribution of X1 and X2 ($\overline{x_1} = 10, \overline{x_2} = 15, \sigma_1 = 0.6, \sigma_2 = 1.$).

Comparison Analysis:

To perform a mathematical comparison between the *Probabilistic* approach and the *Normalized Euclidean Distance* method, we apply equation 5.7 to equation 5.6:

$$P = \frac{1}{\sigma_1\sqrt{2\pi}} e^{-\frac{(x_1 - \overline{x_1})^2}{2\sigma_1^2}} \times \frac{1}{\sigma_2\sqrt{2\pi}} e^{-\frac{(x_2 - \overline{x_2})^2}{2\sigma_2^2}} \times \ldots \times \frac{1}{\sigma_\eta\sqrt{2\pi}} e^{-\frac{(x_\eta - \overline{x_\eta})^2}{2\sigma_\eta^2}}$$

$$P = \frac{1}{(\sqrt{2\pi})^\eta \times \sigma_1 \times \sigma_2 \times \ldots \times \sigma_\eta} e^{-\frac{(x_1 - \overline{x_1})^2}{2\sigma_1^2} - \frac{(x_2 - \overline{x_2})^2}{2\sigma_2^2} - \ldots - \frac{(x_\eta - \overline{x_\eta})^2}{2\sigma_\eta^2}} \tag{5.14}$$

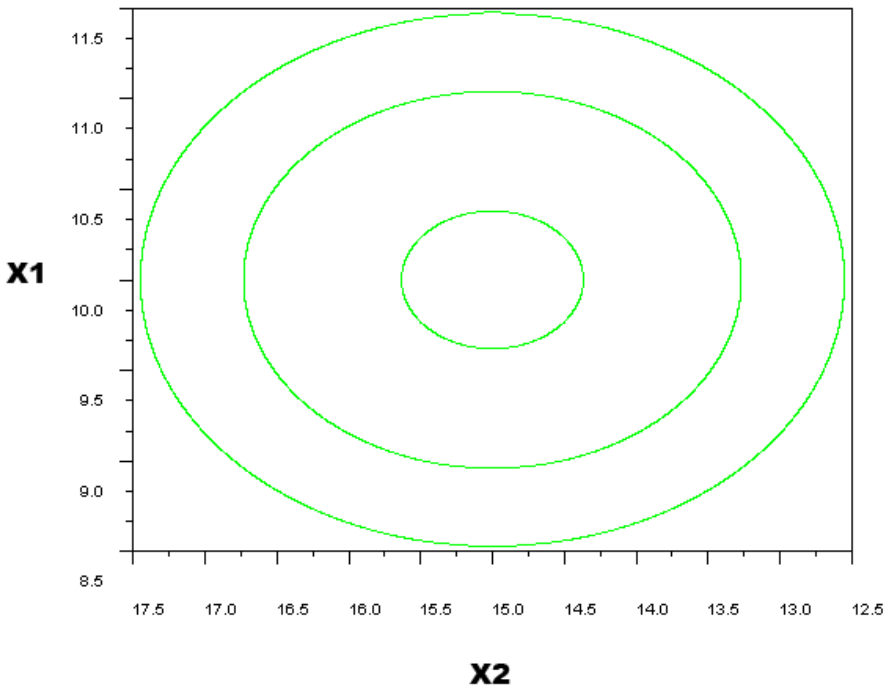


Figure 5.10: Normalized euclidean distance curves.

We define C as a constant for the given pattern with the following value:

$$C = (\sqrt{2\pi})^{\eta} \times \sigma_1 \times \sigma_2 \times \ldots \times \sigma_\eta \tag{5.15}$$

We can introduce DEN in equation 5.14 using the equality presented in equation 5.16.

$$-\frac{(x_1 - \overline{x_1})^2}{2\sigma_1^2} - \frac{(x_2 - \overline{x_2})^2}{2\sigma_2^2} - \ldots - \frac{(x_\eta - \overline{x_\eta})^2}{2\sigma_\eta^2} = \frac{-DEN^2}{2} \tag{5.16}$$

Replacing C and the equality defined in 5.16 to equation 5.14 we have:

$$P = \frac{1}{C} e^{\frac{-DEN^2}{2}} \tag{5.17}$$

Applying this formula on the membership function defined in equation 5.8 we have:

$$\frac{1}{C} e^{\frac{-DEN^2}{2}} \geq P_T$$

$$e^{\frac{-DEN^2}{2}} \geq P_T \times C$$

$$\frac{-DEN^2}{2} \geq ln(P_T \times C)$$

$$DEN^2 \leq -2 \times ln(P_T \times C) \tag{5.18}$$

For $-2 \times ln(P_T \times C)$ to be greater or equal to 0, $P_T \times C$ must be less or equal to 1.

$$P_T \times C \leq 1$$

$$P_T \times (\sqrt{2\pi})^{\eta} \times \sigma_1 \times \sigma_2 \times \ldots \times \sigma_{\eta} \leq 1$$

$$P_T \leq \frac{1}{(\sqrt{2\pi})^{\eta} \times \sigma_1 \times \sigma_2 \times \ldots \times \sigma_{\eta}}$$

$$P_T \leq P_1(\overline{x_1}) \times P_2(\overline{x_2}) \times \ldots \times P_{\eta}(\overline{x_{\eta}}) \tag{5.19}$$

Equation 5.18 imposes a condition on the maximum value of P_T, which is defined as P in the exact position of the average values (maximum value of P).

From equation 5.18 and 5.13 we can derive a relation between P_T and DEN_T as follows:

$$DEN_T^2 = -2 \times ln(P_T \times C)$$

$$DEN_T = \sqrt{-2 \times ln(P_T \times C)} \tag{5.20}$$

Equation 5.17 proves that the *Probabilistic* approach can also be represented as the *Normalized Euclidean Distance*, yielding the same result. The relation between the thresholds defined in each method is presented in equation 5.20. This means that for each threshold defined in the probabilistic method an equivalent threshold for the *Normalized Euclidean Distance* method that yields the same result exists.

Nevertheless, the *Normalized Euclidean Distance* method presents a better computational performance since the exponential calculation is eliminated from the equation.

In Figure 5.11, the curves level view of P and the *Normalized Euclidean Distance Curves* are plotted on the same graph to demonstrate that a value of DEN_T that defines the same subspace on X1 and X2 as P_T exists.

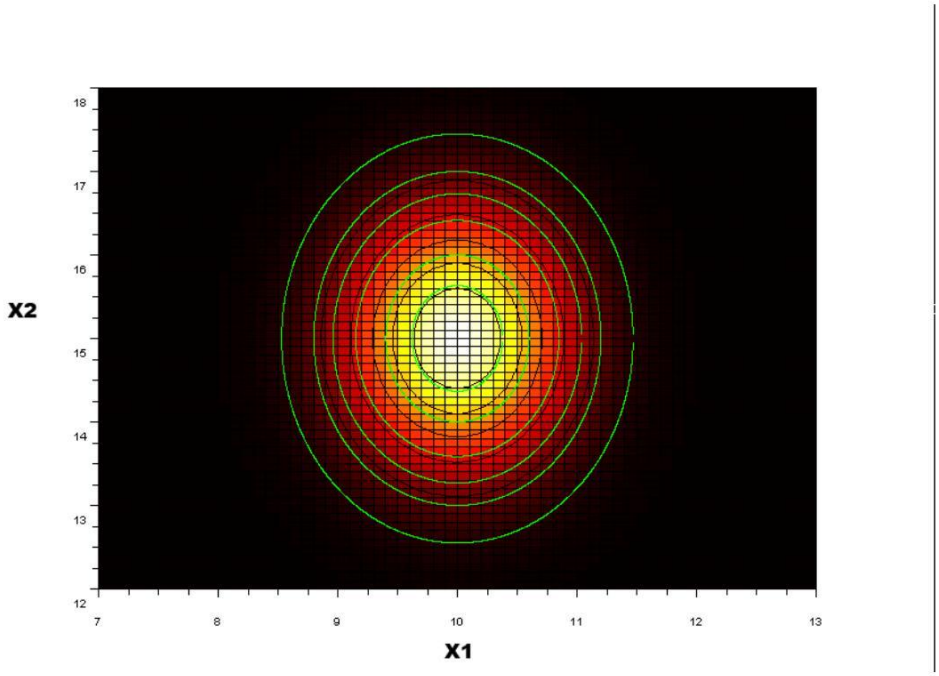


Figure 5.11: Curves level view of P and normalized euclidean distance curves.

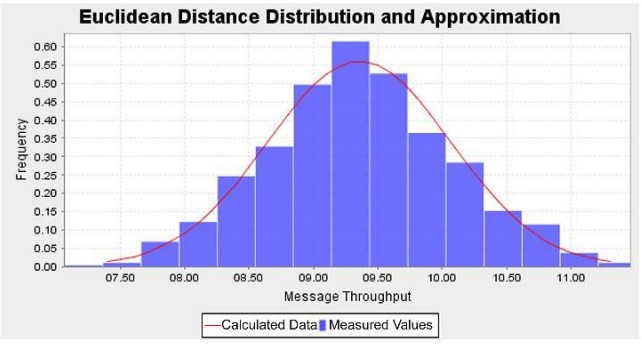


Figure 5.12: Normalized euclidean distance histogram of a fault pattern.

5.3.4.2 Threshold Definition: Gaussian Distribution Approximation

To estimate the threshold DEN_T of the *Normalized Euclidean Distance* method, we evaluate the histogram of the normalized euclidean distances between the patterns $\boldsymbol{M}_{\Delta j}$ and the estimated mean $\overline{\boldsymbol{M}_{\Delta}}$.

Figure 5.12 presents a histogram of the normalized distances between $\boldsymbol{M}_{\Delta j}$ and $\overline{\boldsymbol{M}_{\Delta}}$ applying the formula from equation 5.12. The values presented in this graph were obtained from a simulation with a randomly generated topology containing 100 nodes, 5 sinks and link quality ranging from 0.5 to 1.

In such conditions, it is possible to approximate the probability density function of the normalized distance between patterns and the mean to a Gaussian distribution, as also depicted in figure 5.12.

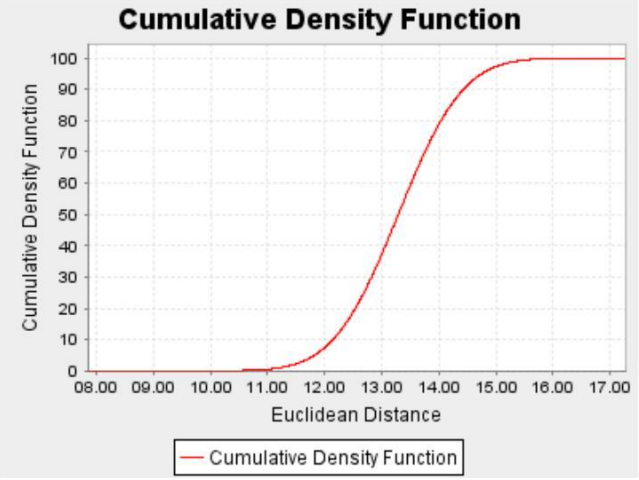


Figure 5.13: Cumulative density function of D_E.

Applying this approximation, we calculate the variance σ_D^2 and the mean μ_D for the distances vector $\overrightarrow{DEN} = d_1, d_2, ..., d_f$ based on the *Maximum Likelihood Estimation* [3, 130] method:

$$\mu_D = \frac{1}{f}\sum_{i=1}^{f} d_i$$

$$\sigma_D^2 = \frac{1}{f}\sum_{i=1}^{f}(d_i - \mu_D)^2$$

When the distances distribution can be approximated to a normal distribution, the probability density function P presented in Figure 5.12 is calculated as demonstrated in equation 5.7.

By integrating P_i we find the cumulative density function depicted in Figure 5.13. This function presents the percentage of the patterns $\boldsymbol{M}_{\Delta j}$ that have a maximum distance d. This allows us to determine a threshold distance DEN_T where a given percentage of the patterns will have a distance to $\overline{\boldsymbol{M}_\Delta}$ equal or smaller than DEN_T. In the presented diagrams this distance is 11.49 for a percentage of approximately 99.8%.

A diagnostic signal $\overrightarrow{S}$ is evaluated as belonging to a given system state Δ if its distance to $\overline{\boldsymbol{M}_\Delta}$ is equal or smaller than $d_{T\Delta}$.

5.3.4.3 Considerations

The membership function alone is only applicable when the distances between the patterns of different system states do not overlap. Figure 5.14 presents the dis-

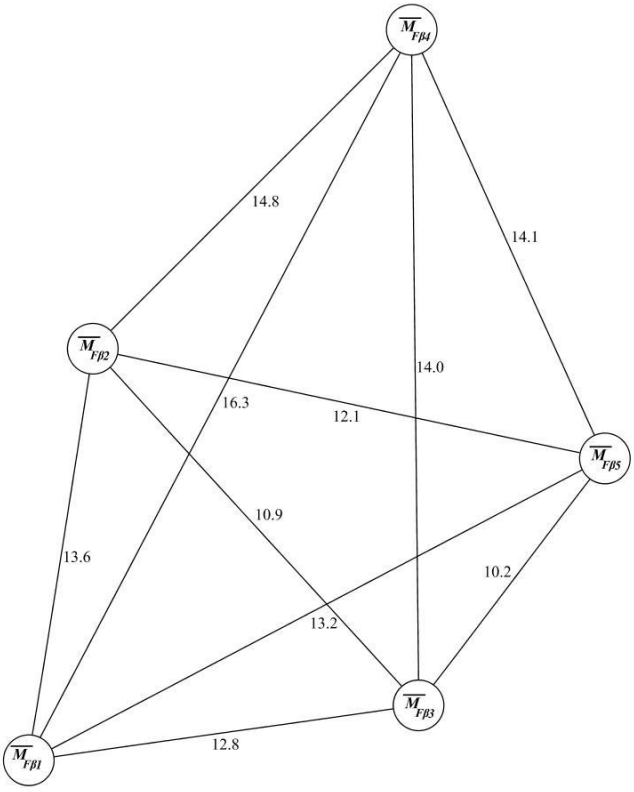


Figure 5.14: Euclidean distances between the mean of sink failure patterns.

tances of the mean of each sink failure pattern acquired in the first application trial performed for this research. This is described in chapter 6.

These patterns present a considerable distance between their mean values and serve as good example for a scenario where this approach is valid. Nevertheless, it is important to point out that, only isolated failures with minimum overlap between coverage areas are considered in this graph. Once composed failures occur, and major overlaps exist, a mechanism to distinguish them is required.

5.3.5 Transient Analyzer

When analyzing a current faulty state, the information of the transition from a healthy to a crashed state can help identify the cause of the failure. The *Transient Analyzer* component is responsible for performing this analysis and providing an estimation of the cause. This component attempts to differentiate two causes of failures:

1. Node crash

2. Link failure

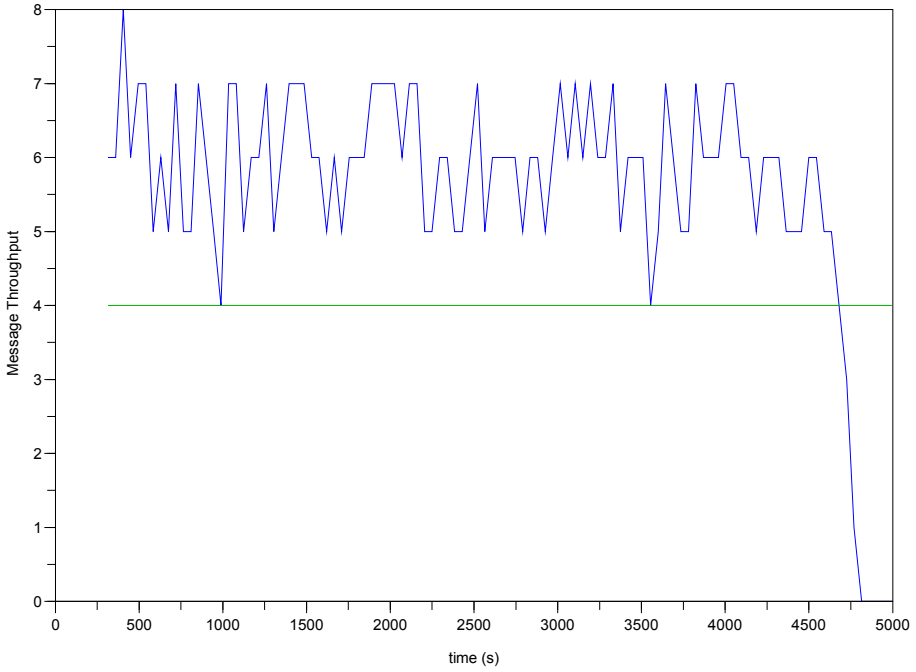


Figure 5.15: Transient state for a sudden crash failure.

It is important to point out that according to the link failure model, it is impossible for the back-end to distinguish between node crash and link failurea. For instance a strong wireless interference may suddenly occur causing the same effect as a crash failure. Nevertheless, since the back-end works with a very restricted set of information, any additional knowledge that can be extracted through analyzing the data has the potential to improve the isolation process.

In this component, a node crash is identified when the node suddenly stops sending events after a period of normal operation. A period of normal operation is defined as a period during which the node has a minimum message throughput. This minimum amount of messages is learned by the system and stored in the *Pattern Base*.

Figure 5.15 depicts the evolution of the calculated message throughput for a node that suffered a crash failure. This graph shows a linear decrease of the observed value over time, which is expected since the node stops sending messages the moment it crashes.

When links suffer interference, it is often the case that some messages still reach their destination. This causes the transient evolution of the message throughput to not follow a linear regression, as showed in Figure 5.16.

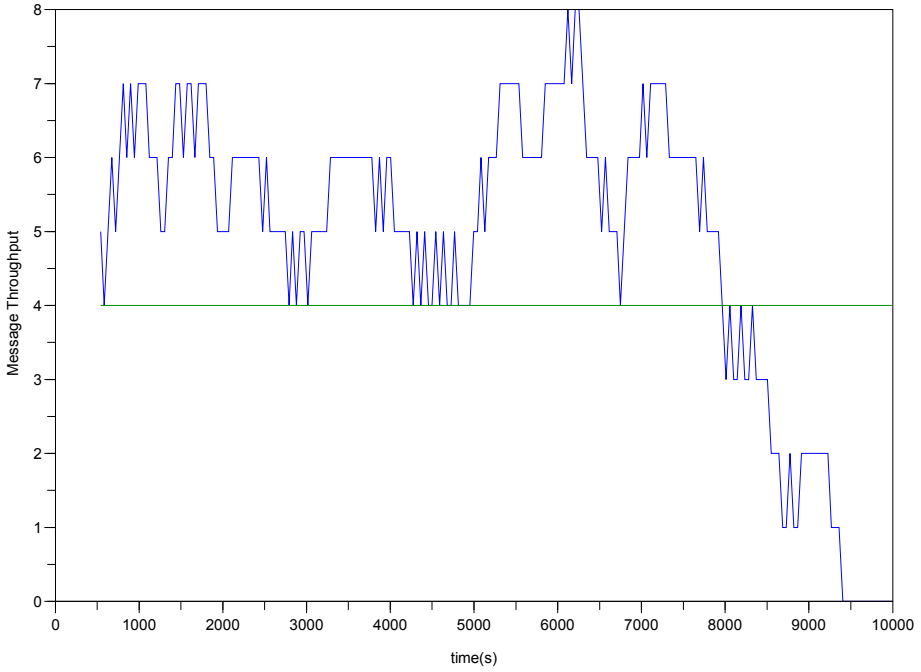


Figure 5.16: Transient state for a link failure.

A link failure is recognized by the system once the message throughput of a sensor node goes below its minimum, but the node still sends sporadic messages.

To define the minimum expected value of the message throughput of a given node during normal operation, it is necessary to evaluate its behaviour individually.

Due to the different link qualities of the paths connecting wireless sensor nodes with the back-end, a global minimum expected value of the nodes message throughput would lead to imprecise numbers that can either result in higher false positives or lower fault isolation effectiveness.

Therefore, the approach adopted evaluates the performance of individual nodes during normal operation, analyzing their message throughput. The minimum expected value of the nodes message throughput is selected based on a confidence interval. This interval defines the distance from the average in number of stand deviations (e.g 1σ, 2σ).

5.3.6 Failures Match Maker

The *Failures Match Maker* is a component which executes an algorithm to combine node crash failures and link failures with the patterns stored in the *Pattern Base*. This procedure is used to match the known patterns with the diagnostic signal $\overrightarrow{S}$

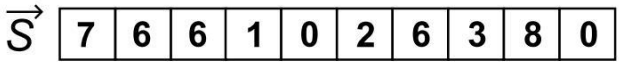


Figure 5.17: Diagnostic signal.

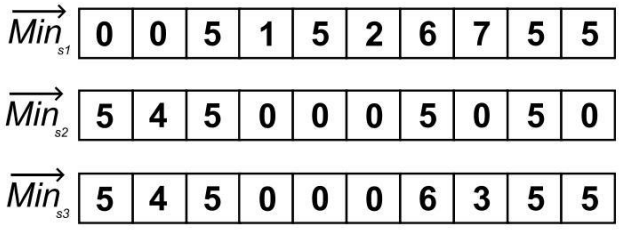


Figure 5.18: Minimum vectors of three different sink failures.

and determine the possible real failures in the system. This component enables the identification of failures represented by box 13 in Figure 5.1.

Consider a diagnostic signal $\overrightarrow{S}$ with a message throughput from ten nodes as presented in Figure 5.17. Given the heart beat of the nodes in this example and the timeframe analyzed, the maximum message throughput is eight.

As the fault isolator has no information about the network topology, a graph analysis to identify the failure is not possible. Therefore the match maker uses the patterns already known to the system to identify possible composed failures.

Figure 5.18 depicts the minimum ($\overrightarrow{Min_{\Delta}}$) vectors of three different sink failures. These vectors contain the minimum message throughput of each node. These values are calculated using as basis the message throughput distribution of each node for the known patters, and a confidence interval.

The minimum vectors allow the *Failures Match Maker* to generate an unexpected observed failure vector ($\overrightarrow{UF_{\Delta}}$) for each system state available in the pattern base.

The $\overrightarrow{UF_{\Delta}}$ indicates which components present a faulty state which was not expected for the given stored pattern (i.e. have a lower message throughput than expected for the pattern). This vector is calculated as follows:

$$UF_{\Delta,i} = \left\{ \begin{array}{ll} 0, & if \quad \overrightarrow{S_i} >= \overrightarrow{Min_{\Delta,i}} \\ 1, & if \quad \overrightarrow{S_i} < \overrightarrow{Min_{\Delta,i}} \end{array} \right\}$$

Figure 5.19 presents $\overrightarrow{UF_{\Delta}}$ applying the ($\overrightarrow{Min_{\Delta}}$) presented in Figure 5.18.

To ensure that the diagnostic signal $\overrightarrow{S}$ belong to a combination of a pattern known to the system state and additional node crash and link failures, the algorithm replaces

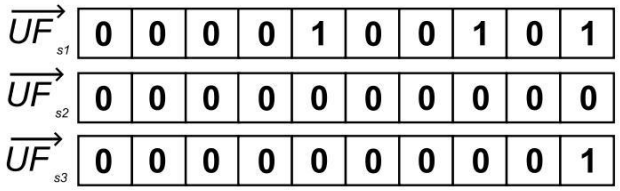


Figure 5.19: Unexpected observed failure vectors for sink failures S1, S2 and S3.

the values of the message throughput in $\overrightarrow{S}$ for the identified failures in $\overrightarrow{\overrightarrow{UF_{\Delta}}}$. The replaced value consist of the average message throughput of the node for the pattern. The resulting vector is then passed to the *Membership Evaluator* to verify if it belongs to the given pattern or not.

If the *Membership Evaluator* returns a negative response, the pattern is not considered as possible combined failures. Otherwise, the *Failures Match Maker* requests the *Transient Analyzer* to evaluate each node identified as unexpected failure.

The result of the *Failures Match Maker* is a set of possible failures, which consist of the identified system patterns and the result of the analysis of the *Transient Analyzer*.

5.3.7 Neural Classifier

The *Neural Classifier* is applied when a diagnostic signal $\overrightarrow{S}$ has a high probability of belonging to different patterns known to the system. This component is composed by several neural networks specially trained to distinguish "overlapping" system states. The overlaps between different system states occur when the *Normalized Euclidean Distance* of the diagnostic signal $\overrightarrow{S}$ to more than one pattern in the *Pattern Base* is small enough to be considered as a high probability match by the *Probability Evaluator*. Another overlapping case can occur. When combined failures happen in a way that the "recovered" diagnostic signal is accepted as a member of more than one pattern, multiple results are returned by the *Failures Match Maker*.

Neural Networks are suitable for the task of differentiating between the possible patterns, since they can achieve good results when classifying patterns of noisy signals. An example of noisy signals are the diagnostic signals applied in this approach [31]. Additionally, neural networks have a very high performance when compared with other pattern recognition techniques such as neural-fuzzy systems [36]. This makes them an attractive option for enterprise scenarios.

The behavior of the *Neural Classifier* follows two stages: the first stage selects and trains the pool of neural networks, in the second stage the classifier is executed during operation time, depicted in figure 5.20.

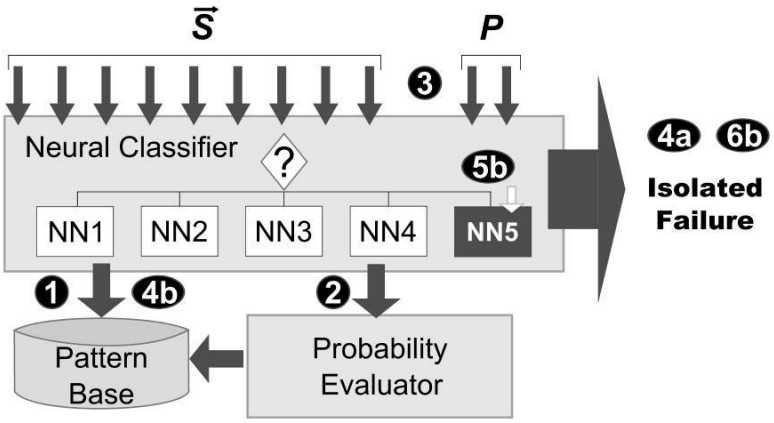


Figure 5.20: Neural Classifier.

The process of selecting specialized neural networks is based on an evaluation of possible overlaps between failures. By gathering the mean of each pattern stored in the *Pattern Base* (1) and requesting the *Membership Evaluator* to determine the system state that the pattern belongs to (2), a set of overlapping system states can be found. If overlaps are encountered, a neural network is trained to differentiate between the failures (e.g. NN1, NN2, NN3 and NN4).

Neural networks are trained using the data stored in the *Pattern Base* as training sets. To train the neural network we apply the *Perceptron* training algorithm with back propagation. This learning algorithm is chosen since it is proved to converge when there is a solution [84].

The specialized neural networks are composed of η input nodes and m outputs according to the specialization. The number of inputs is equal to the number of sensor nodes in the network. The number of outputs of the neural network is dependent on the number of system states that must be differentiated.

The output of the neural network is made up of values between zero and one. A value 1 indicates that the diagnostic signal belongs to the system state represented by the output. A value of 0 indicates that the diagnostic signal does not belong to the pattern. The value range of the neural network output requires an activation function of each node of the neural network to operate between [0,1]. The function selected is the logistic function:

$$f(v) = \frac{1}{1 - e^{-v}}$$

During operation time, the *Neural Classifier* receives a set of "recovered" diagnostic signals and a list of possible failures P (3) from the "Failures Match Maker" component. Based on the list of failures, this component selects the appropriate neural network to produce the final result of the *Pattern Fault Isolator* (4a). This final

result consists of a set of failures and a score. This score reflects the number of times that the *Neural Classifier* selected each failure as the probable current system failure.

When a request is received and an appropriate neural network is not available, the classifier automatically trains a neural network based on the data available in the *Pattern Base* (4b), and adds it to the neural networks pool (5b). Finally, the result is calculated using the new neural network (6b).

To improve the scalability of the system, the wireless sensor network can be divided in sectors. For each sector, a set of neural networks is trained which restricts the maximum number of inputs for the neural network and facilitates the maintenance of the system when the neural networks have to be retrained (e.g. a new sink is inserted in a location).

5.4 Prediction Model

In this section, we perform an analysis of the fault isolation method proposed and of the WSN model defined in Section 3.1.1. This analysis allows to predict the behaviour of the fault isolation method under specific conditions. These predictions serve as a basis for system evaluations. The evaluations of the system is described in Chapter 7 and proves the predictions made in this Section.

5.4.1 Membership Evaluator

Threshold Definition

By applying the *Gaussian distribution approximation*, the threshold is determined based on on the cumulative density function of the distance distribution, as discussed in Section 5.3.4.

With this approach, the threshold defines the percentage of the fault diagnosis signals that are in fact part of the pattern and that will be considered as such. This ideal percentage is calculated based on confidence intervals. These intervals define how much of the population will be within a certain number of standard deviation threshold. This values consider a cut on the normal distribution on both sides ($\pm n\sigma$). The threshold of the membership function applies a cut on only one side of the curve. Therefore, the threshold can be calculated as presented in the equation bellow. Table 5.1 presents these values.

$$IdealValue = 0.5 + \frac{confidence(n)}{2}$$

n	confidence	Ideal Value
1	0.682689492137	0.841344746
2	0.954499736104	0.977249868
3	0.997300203937	0.998650102
4	0.999936657516	0.999968329
5	0.999999426697	0.999999713

Table 5.1: Confidence Interval and ideal effectiveness values

As the average and standard deviation values are determined based on sampling, the higher the number of samples (pattern readings), the higher the chance of these calculated values being closer to the ideal distribution. Therefore it is possible to assume that:

Prediction 1: *"In networks where a Gaussian distribution approximation is suitable and no near overlaps with other patterns exist, the threshold from the membership function will determine its fault isolation effectiveness as the number of acquired pattern readings and analyzed diagnostic signals tend to infinity."*

As this is a distribution approximation, differences from the ideal value and the observed values may still occur and are further analyzed in Chapter 7.

Timeframe

The message throughput is the sum of the number of messages that successfully reach the back-end. Considering the heart beat constant, the timeframe directly influences the number of message sending attempts a node executes in each pattern reading.

In networks that contain nodes with a message throughput variance very close to zero, the approximation of node message throughput distribution to a *Gaussian* distribution becomes inaccurate, if the number of attempts to send messages is not large enough. As the message throughput operates in the discrete domain, the approximation will present a very high "peak" close to the mean.

As the calculations of DEN is based on the assumption that the distribution of the nodes throughput can be approximated to a normal distribution, an interesting phenomenon occurs when the system reaches the borders of this assumption.

The number of messages generated by a sensor node that reach each individual sink is a binomial distribution, as described in 5.3.3. In this distribution, the mean and

variance are calculated as demonstrated in equations 5.21 and 5.22, where p is the probability of success for each attempt and n is the number of attempts.

$$Mean = n * p \tag{5.21}$$

$$Var = n * p(1 - p) \tag{5.22}$$

In a WSN, p is equivalent to the path quality between the node and the backend ($\varsigma^{ni}_{Backend}$). The number of attempts is calculated based on the timeframe and heart beat as presented in equation 5.23. For the sake of simplicity, timeframes should be chosen as a multiple of the heartbeat.

$$n = \frac{timeframe}{heartbeat} \tag{5.23}$$

With small values of n in a network with p close to one, the variance tends to zero. The consequence of such small variances is that a reading with one message difference from its expected average can generate a very high normalized euclidean distance for the pattern reading, while the majority of its readings will generate zero distance. This occurs since the distance contribution of each node is calculated as $(\frac{(x_i - \overline{x_i})^2}{\sigma_i^2})$.

The consequence of this poor approximation is that when a part of the network has such quality on the links, outlier readings will occur in the DEN distribution when n is not high enough for the network setup. Therefore, we can predict that:

Prediction 2: *"In networks where the path quality between nodes and sinks is close to one, outlier readings are more likely to occur as the number of message sending attempts for each pattern reading is reduced."*

5.4.2 Transient Analyzer

The *Transient Analyzer* relies on the expectation that nodes suffering link failures manage to send messages sporadically. This is heavily dependent on the link failure model, and according to the strength of the interference, it may result in incorrect diagnosis. Therefore we can make the following predictions:

Prediction 3: *"The Transient Analyzer has a higher rate of misdiagnosis between link failures and crash failure as the interference increases."*

Prediction 4: *"The Transient Analyzer has a lower fault isolation effectiveness as the interference becomes increasingly small."*

5.4.3 Failures Match Maker

The *Failures Match Maker* is built on top of the *Membership Evaluator* and *Transient Analyzer*. This component analyzes each node individually, "recovering" the message throughput value of nodes that did not generate enough messages. This approach enables the *Failures Match Maker* to have a higher fault isolation effectiveness than the membership function. However, this performance increase comes at the cost of node crash and link failure false positive identification.

Additionally, this component relies on a maximum and minimum message throughput vector to determine the unexpected failures and unexpected healthy states. These vectors define a threshold on the normal distribution of the message throughput of each node. This results in some of the readings being considered as faulty or healthier than expected, even though they are originated from the same state.

Having these considerations in mind, it is possible to perform the following predictions:

Prediction 5: *"For the same confidence interval, the Failures Match Maker component presents a fault isolation effectiveness higher than the Membership Evaluator, if the threshold of the Transient Analyzer is kept constant."*

5.4.4 Neural Classifier

Due to the nature of neural networks and the difficulties involved in analyzing their behaviour, neural networks are usually treated as a "black box" rather than a mathematical model. This makes it difficult to predict the efficiency of the Neural Classifier. Nevertheless, it is possible to assume that:

Prediction 6: *"The Neural Classifier should have a high fault isolation effectiveness performance if the diagnostic signals are close to the ones used in the training set."*

5.5 Summary

This work presented an approach for fault isolation in WSNs based on pattern recognition. This proposed solution focuses on WSNs applied to business processes and therefore assumes a very restricted set of information available to the back-end. The challenge of isolating failures under this restriction is addressed by using several pattern recognition methods, such as neural networks, statistic analysis and rule based approaches.

The proposed fault isolation method for WSNs achieves the goal of identifying failures occurring in heterogeneous WSNs applied in business processes by providing

the necessary concepts and algorithms to perform this task in an efficient manner. As the fault isolation proposed is platform independent, together with the proposed framework from Chapter 4, the goal of enabling the integration of heterogeneous WSNs with enterprise applications is also achieved.

In Chapter 7, we analyze the efficiency of the fault isolation approach by evaluating it in a real world application trial and through simulations. The evaluation serves as tool to confirm the hypotheses made in the prediction model defined in Section 5.4.

6. Applications

In order evaluate the efficiency of the fault isolation method proposed in Chapter 5, and confirm the predictions made on Section 5.4, we investigated the limits and efficiency of the proposed fault isolation approach.

The investigation consisted of isolating faulty components originated from two different sources: application trial and simulation. The first source was an application trial conducted for a period of three months. It involved a WSN containing 36 nodes, 5 sinks, and one computer to collect the generated network events. The second source consisted of a simulator specifically developed for generating WSN traffic considering node, sink and link failures. During both experimentations, the *FT-WiseNets Monitoring Application* evaluated the WSN and provided the isolated failures as output.

The simulation procedures enabled us to investigate the limitations of the fault isolation method, since it was possible to use large scale networks, and a simulation time that would be equivalent to months in a real world application trial. Simulation also allowed the observation of the effects that changes in the fault isolation method parameters cause on the effectiveness of the proposed solution.

To confirm that the method not only works in theory but also in practice, we analyzed the data acquired during a real world application trial with the same fault isolation method.

In this Chapter we explain the procedures applied to gather the data sets in the application trial and the implementation details of the FT-WiseNets simulator and monitoring application.

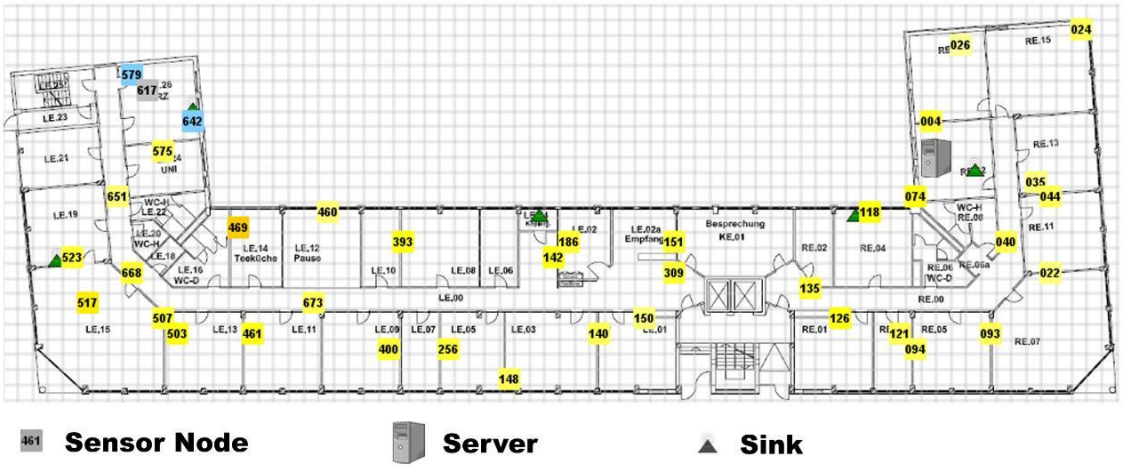


Figure 6.1: Nodes deployed in the research facility.

Figure 6.2: Placement of nodes during the application trial.

6.1 Application Trial: Research Lab Monitoring

To analyze the performance of our approach in a real world setting, we performed an application trial in our research facility. There, μParts [169] were deployed in the offices to monitor light, temperature and motion of the deployed sensor nodes. The choice for this hardware platform was made based since it can be easily modeled as described in Section 3.1.1.

In this trial 36 nodes and 5 sinks were distributed as depicted in the floor plan of SAP Research facility in Figure 6.1. A computer representing the back-end was responsible for storing the events generated by the sensor nodes in a database. The positions of the sinks were chosen in order to provide full coverage of the nodes during deployment time.

The enclosure of the sensor nodes used in this trial had an opening to allow the measurement of light. Figure 6.2 presents a sensor node used during the trial and the placement of six sensor nodes. A double-sided tape was used in order to attach nodes to cupboards, white boards, walls and doors.

Under normal operational conditions, the heart beat of the sensors was configured to 45s. A brief overview of the type of data that is transmitted from each of the three sensors is presented below:

- Light sensor: The light sensor is the cost-optimized, highly integrated light-to-voltage optical sensor TSL13T from Taos Inc [1].
- Temperature sensor: The temperature sensor is the TC1047A temperature to voltage converter from Microchip [2]
- Movement sensor: The motion sensor is a (digital) micro-machined ball-switch sensor.

Initially, sinks were intended to create a mash-up to forward the packages generated by the sensor nodes to a sink directly connected to the computer. During a first evaluation of the deployment setup, up to thirty wireless networks were found operating on the the same frequency range as the sinks (IEEE 802.11 2.4GHz band).

After this initial setup and evaluation of the wireless media usage, it became clear that the mash-up approach would not be feasible. Hence, each sink was connected to the Local Area Network (LAN) and forwarded packages received from sensor nodes from the LAN.

The dense wireless utilization of the same frequency band used by the sinks did not affect the communication of the wireless sensor nodes. This is due to the frequency used by these device (868 MHz band).

6.1.1 Data Collection

The FT-WiseNets Monitoring Application was installed on the computer depicted in Figure 6.1. This application is responsible for storing the data generated during the entire period of the trial in the database managed by the *System State*.

The data collected during the trial is intended for evaluating the efficiency of the fault isolation method when sink and node failures occur. The evaluation performed with the data collected during this trial is discussed in Chapter 7.

[1] http://www.taosinc.com
[2] http://www.microchip.com

To collect the data required for this evaluation, patterns of the system during normal operation and during individual sink failures were acquired. First a pattern of the system during normal operation was acquired, which enabled calculating the minimum expected message throughput of the sensor nodes. Before the acquisition of this pattern, a node timeout verification was performed to ensure that all nodes were sending messages during this period. Additionally, all nodes were verified and had at least 30% of their maximum throughput.

Individual sink failure patterns were generated by physically introducing power failures to the sinks for a period of 20 minutes. To ensure that the system only contained one sink failure on each pattern acquired, the message throughput of each sensor node was verified before introducing the failure. If the message throughput of a sensor node was lower than its expected minimum, the node would be maintained to correct the problem.

After the acquisition of each system pattern, the nodes' throughputs were once again verified to define if failures occurred after the pattern acquisition started.

A second set of data was collected applying the same methodology as the one used for the sink failure pattern acquisition. This data set was collected to enable the evaluation of the fault isolation method through cross-validation (Chapter 7). Both data sets were acquired during the same time period, during 2 days.

6.1.2 Preliminary Data Analysis

In the performed trial, nodes were deployed for a period of three months and generated a total of 3 million events. Figure 6.3 presents the distribution of events generated by each individual node. As depicted in this graph, the number of events per node is not even, which indicates that parts of the system did not work properly during the trial period.

During the trial period, the WSN suffered from failures caused by different sources. This was a valuable input for this thesis, since it showed the practical problems that occur in a WSN. In one of the cases, after one week of normal operation, five nodes suddenly became unreachable. After investigating the cause, we found that a door (with built in metal) was separating a sink from all nodes. This door was installed in our research facility after the deployment of the sensor nodes.

To have a better understanding of partial and global failures that occurred in the system, we analyzed the number of nodes appearing as failed (F_o) by applying a timeout fault detector with a crash timeout equals to 20 minutes. The result is presented in Figure 6.4.

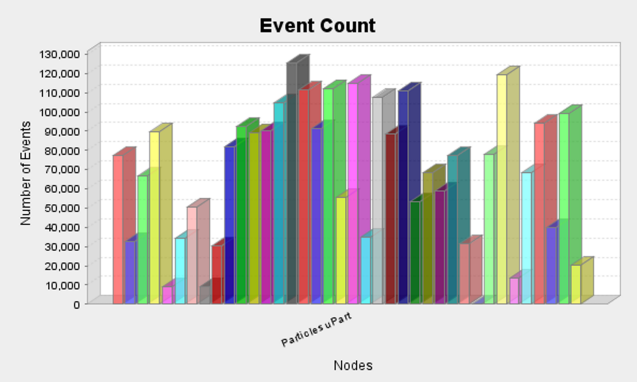


Figure 6.3: Number of events per node.

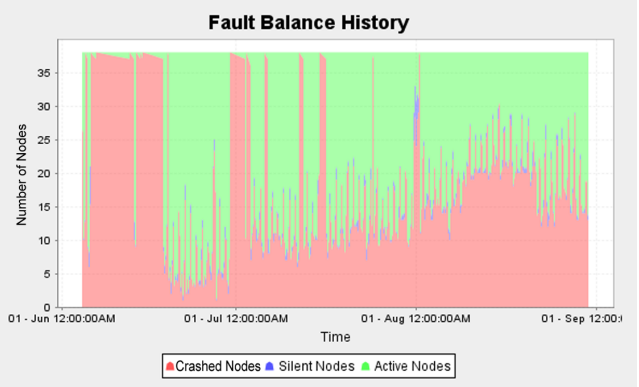


Figure 6.4: Fault balance for the period of 2 months.

This graph demonstrates that during the period of the trial the system suffered several global failures and was constantly indicating partial failures. This information however does not precisely indicate the malfunctioning parts, but rather presents an overview of the system state history.

As demonstrated by this graph, during the initial setup of the trial, a massive failure of the WSN was identified by the timeout fault detector. This occurred since the backend system presented problems that prevented storing the events generated by sensor nodes in the database.

It is also interesting to note that a partial failure occurred periodically and on a daily basis at 12:00AM. Given the time of the occurrence, it was difficult problem to investigate. Nevertheless, the fault isolation approach proposed in this thesis, correctly identified this failure. With further investigation we discovered that one of the sinks was rebooting periodically.

6.2 Application II: Simulation

Simulation enables scientifically exploring the efficiency and limitations of algorithms through the use of network setups that would otherwise impose a significant efforts to be realized in a real world application trial. Therefore, simulation was applied in this thesis in conjunction with a real world trial.

To evaluate the proposed approach by simulating WSNs, we researched existing simulation environments. Given the difficulties to adopt the investigated existing simulation environments, we developed a simulator which is presented in Section 6.2.2.

6.2.1 Existing Simulation Environments

In order to decide the most suitable simulation environment, we analyzed existing solutions [65][149][170]. The followinf list describes some of the existing simulation environments.

NS-2

Broadly used in academic fields, NS-2 [65] provides a network visualization and animation tool called Nam, where packages can be visualized and traced during the simulation. Several extensions and tools exist for this solution. Nevertheless, the adoption of this approach presents a set of drawbacks:

- Network topology, event scheduling and data output should be written in the programing language Tcl and OTcl.
- New protocols, packets and agents (the entity that receives and sends packets) is done in C++, and NS-2 must be recompiled at each change.
- The Tcl/C++ variable binding scheme is not evident and supports only a few basic data types.
- Although the documentation is extensive, structures needed to implement new packages and protocols in C++ are not well documented.

QualNet

QualNet [149] is designed for fixed topology and wireless networks, offering a simulation environment with an intuitive interface. The program is capable of simulating thousand of nodes.

The major drawback of adopting this tool is that it is a commercial product freely available to companies for thirty days, only upon request. The short time frame that this tool is available for and the delay caused by approving the usage of the tool, make its adoption unpractical for this thesis.

OMNeT++ with Mobility Framework

OMNeT++ [170] provides a component-based, modular and open-architecture simulation environment with GUI support and an embeddable simulation kernel. This tool is freely available for educational purposes but requires special license for commercial use. Additionally, it offers only one radio propagation model and uses its own language for network description, NED. Therefore the development of more complex simulations would impose the requirement to learn NED.

6.2.2 FT-WiseNets Network Traffic Simulator

The analysis performed and previous experience [164] with the evaluated frameworks, showed that the adoption of the afore mentioned simulators causes initial overhead due to cumbersome programming languages, licenses and difficulties of propagating the messages being generated during simulation time to the local network.

In comparison to the existing simulation environments, the initial overhead of developing a WSN traffic generator with specific functions for automatic failure generation is smaller. The reason for this is that evaluating the fault isolation method described in Chapter 5 is simple due to restricted information set it uses as basis for the isolation process. Therefore we developed and adopted a simulator called FT-WiseNets Network Traffic Simulator to address the needs of this thesis.

The simulator was developed using the Java [114] programming language because of several advantages: platform independence, object-oriented programming paradigm, robustness, security and portability.

The *FT-WiseNets Network Traffic Simulator* is composed of three modules that generate events, which are propagated to the *Notification Manager* component of the FT-WiseNets framework. The three modules:

- Topology Generator: Creates random network topologies based on input parameters.
- Failures Generator: Automatically inserts failures in the network at specific time intervals.
- Traffic Generator: Calculates the messages that reach the back-end and propagates the events.

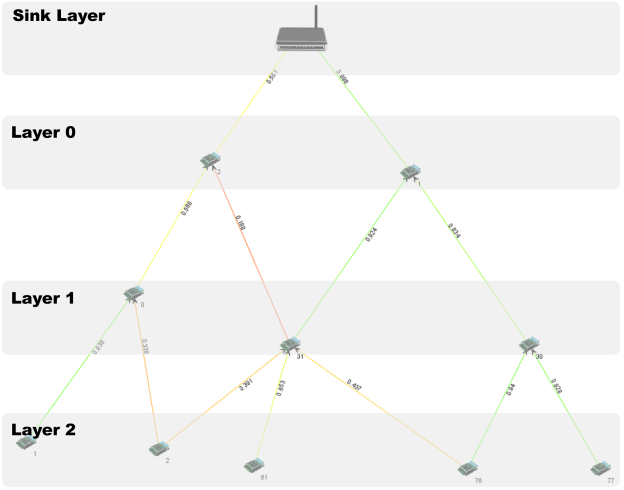


Figure 6.5: Topology scheme in the FT-WiseNets Network Traffic Simulator.

6.2.2.1 Topology Generator

In the *FT-WiseNets Network Traffic Simulator*, the topology of the randomly generated networks is based on a layered structure as presented in Figure 6.5. This structure was selected to prevent infinite loops in the communication path during simulation.

In this approach, each node is assigned to a layer according to the number of hops required to reach the sinks. Nodes in direct connection with a sink (single hop) are placed in "Layer 0". "Layer 1 " consists of nodes that have a two hop distance from the sink. The remaining layers in the hierarchy follow the same logic.

The random generation of the network topology requires a set of input information:

- Number of sinks
- Number of node layers
- Number of nodes in each layer
- Link quality range
- Network overlap

The network overlap indicates the percentage of connections each node has with a higher layer in the network. As an example, let us consider a network containing 20 sinks and 100 nodes in each of its two node layers. With 10% overlap, each node in "Layer 1" connects to 10 randomly selected nodes in "Layer 0". Each node in "Layer 0" connects to two randomly selected sinks.

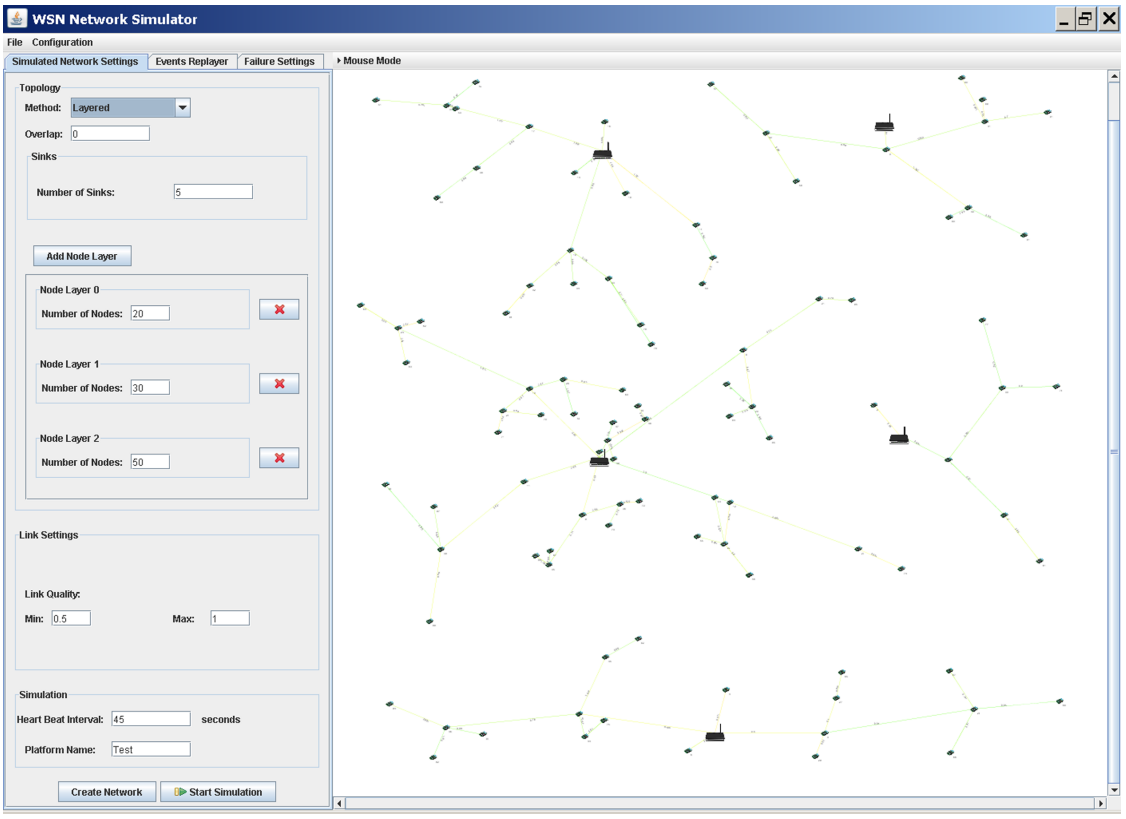


Figure 6.6: Topology generation view in the FT-WiseNets Network Traffic Simulator.

Figure 6.6 presents a screen shot of the the *FT-WiseNets Network Traffic Simulator* topology generation view. The quality of the link indicates the probability of a message reaching its destination on each attempt and is illustrated by the colors of the links' connections.

The algorithm for the random topology generation is presented in pseudo-code in Algorithm 1. The basic idea of this algorithm is to randomly distribute nodes in each layer among nodes in a higher layer.

To enable the same probability of child node assignment among parent nodes, each parent node is assigned to a randomly generated number. These numbers are summed and divided by the number of nodes that are to be distributed. This procedure defines a cost for each node that needs to be assigned to the upper layer and defines how many child nodes belong to each parent node.

Once the parent node is defined a link connection is created and additional connections (due to overlap) are added.

Input: Number of nodes per layer, Number of Sinks, Link Quality Range, Overlap Percentage
Output: Network Topology

```
1  Create node layers;
2  Create and add sinks to sink layer;
3  currentLayer ← sink layer;
4  for next layer← Layer 0 to Last Node Layer do
5      sum ← 0;
6      z ← number of nodes to be added in next layer;
7      y ← number of nodes from current layer;
8      c ← ceil(overlap ∗ y);
9      foreach node n of the current Layer do
10         Calculate r = Random(0,1) (value between 0 and 1);
11         StoreKeyValuePair(n, r);
12         sum = sum + r;
13     end
14     nodeCost ← sum/z ;
15     rest ← 0;
16     foreach node n of the current Layer do
17         r ← GetValue(n);
18         v ← (r + rest)/cost;
19         x ← round(v);
20         rest ← (x − v) ∗ cost;
21         for i ← 1 to x do
22             Create node n_i and add it to next layer;
23             Add Link(n_i,n,Random(minQuality, maxQuality));
24             while c > addedlinks do
25                 Select random node n_j from current layer;
26                 if n_i not connected to n_j then
27                     AddLink(n_i,n_j,Random(minQuality, maxQuality));
28                 end
29             end
30         end
31     end
32     current Layer← next layer;
33 end
```

Algorithm 1: Random network topology creation

For the graphical representation of the network topology, the Java Universal Network/Graph Framework (JUNG) [75] was applied. This framework supports the easy manipulation and presentation of graphs, including several display layouts, graph search functions and graphical editing tools.

6.2.2.2 Failures Generator

Failures can be inserted in the simulated network manually or through the failures generator module. This module offers the possibility to periodically insert failures in the network according to different strategies defined by the user.

To facilitate evaluating the simulation results, the network status can be saved to a log file together with the equivalent timestamp. If failures are periodically inserted, the status of the network will be recorded in the log file on each new state.

This module allows the insertion of random failures based on the following strategies:

- Percentage of sinks and nodes
- Fixed number of sinks and nodes
- Interference area
- Random number of sinks, nodes, and interference

Percentage

In this approach, a percentage defined by the user determines the number of sinks and nodes that should be marked as failed. The module uses the defined values to calculate the specific numbers, randomly select nodes and sinks, and mark the selected components as failed. Figure 6.7 presents a network where the module randomly inserted failures in 30% of the nodes and 30% of the sinks.

Fixed number

The strategy based on a fixed number of nodes and sinks is similar to the percentage approach. However, this approach prevents the rounding of numbers of failed sinks and nodes, therefore, the user must specify integer values. The output of this strategy is equivalent to the one presented in Figure 6.7.

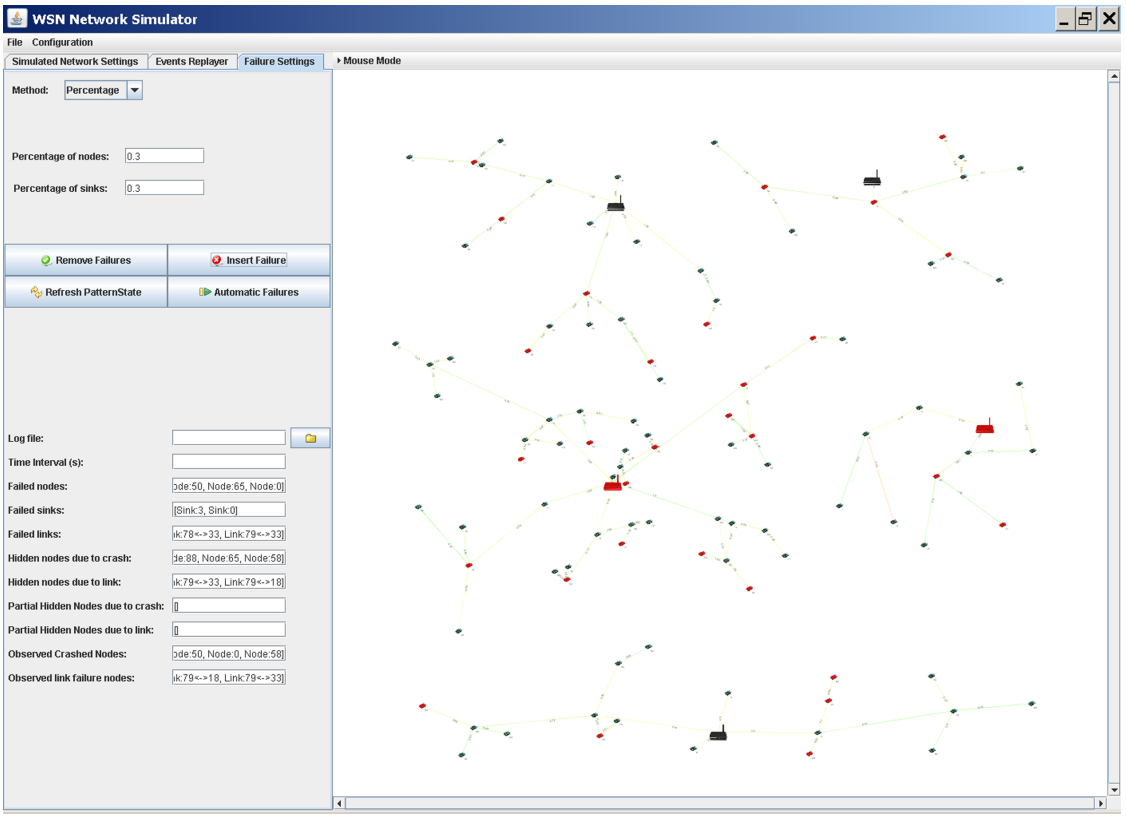


Figure 6.7: Failure percentage view in the FT-WiseNets Network Traffic Simulator.

Interference Area

The framework allows the insertion of areas of interference, as illustrated in Figure 6.8. The area of interference reduces the link quality of the connections that intercept the area. The quality reduction value is randomly selected, and resides between the values defined in the interference range.

The interference area is defined by a radius and can be dragged and dropped by the user on the network graph. According to the area where the interference area is placed, it is possible that no changes in the link connections occur (no connections intercept the area). Also, as the link interference is randomly generated, it can generate different reductions in the quality of the links, even if it is placed in the same position.

Random number of sinks, nodes, and interference

The fourth "failure insertion strategy", offered by the failures generator module, uses as input a maximum number of failed sinks, nodes and links. It also uses as input the maximum value of interference that can affect links. These values are used to randomly select the number of components that are marked as failed.

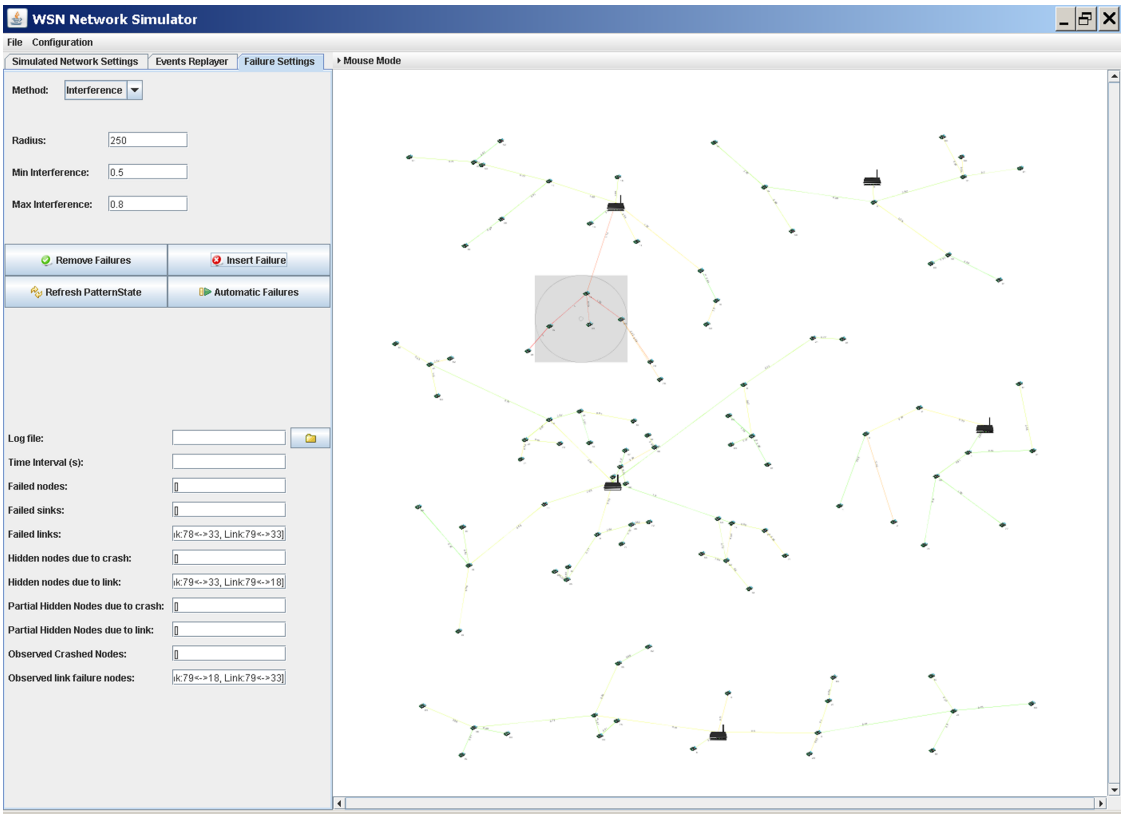


Figure 6.8: Area of interference failure view in the FT-WiseNets Network Traffic Simulator.

6.2.2.3 Traffic Generator

Based on the network topology graph, the traffic generator module verifies at every heart beat interval, which messages reached the back-end. The traffic generator then propagates the messages to the *Notification Manager*.

Algorithm 2 presents the logic executed by this module in pseudo-code. The algorithm browses the entire network starting at the last node layer until it reaches the sink.

On each iteration, a node attempts to propagate the messages in its queue (including its own message) to its parent node(s). The probability that each attempt will succeed is defined by the quality of the link connecting the node and the parent node destination. Every time a message is successfully propagated, it is added to the node destination queue until the message reaches the sink.

Finally, once the entire network has been scanned, the events that reached the non faulty sinks are propagated to the FT-WiseNets middleware.

```
   Input: Network Topology
   Output: Notification of messages that reach the backend
 1 for layer ← Last Node Layer to Layer 0 do
 2     foreach node n of the layer do
 3         if n is NOT faulty then
 4             foreach message m in node n do
 5                 foreach outgoing link l from node n do
 6                     Calculate Random(0,1) (value between 0 and 1);
 7                     if Quality from l ≥ Random(0,1) then
 8                         Get node nDest from l destination;
 9                         Add message to nDest;
10                     end
11                 end
12             end
13         end
14     end
15 end
16 Notify events from non faulty sinks ;
```

Algorithm 2: Message generation in the network

6.3 FT-WiseNets Monitoring Application

The *FT-WiseNets Monitoring Application* is a partial implementation of the framework defined in Chapter 4. This implementation includes a user interface to facilitate the display of the network status and the analysis of fault isolation results.

Following the same approach as the *FT-WiseNets Traffic Simulator*, the *FT-WiseNets Monitoring Application* was fully developed in Java [114]. MySQL Server 5.1 [115] was used as a database system to store all the events generated by the network, the patterns acquired, and the information about the nodes and sinks available in the system.

The *FT-WiseNets Monitoring Application* is responsible for processing events generated by either simulation or real devices. These events are processed by the *FT-WiseNets Monitoring Application* in the same manner, as defined by the "Platform Abstraction Layer" of the FT-WiseNets framework (Chapter 4).

The implementation effort of this system was separated into five fault isolator components and five Graphical User Interface (GUI) views.

6.3.1 Fault Isolator Implementation

The implementation of the fault isolator approach proposed in this thesis consists of five core classes. Each implemented class is equivalent to one component in the fault isolator architecture:

- Pattern Base
- Membership Evaluator
- Failures Match Maker
- Transient Analyzer
- Neural Classifier

Additional auxiliary classes were also implemented. Nevertheless, for the sake of simplicity, we will only discuss the implementation details of these five core classes in this thesis.

Pattern Base

The *PatternBase* class is responsible for managing system patterns. Its main functions include saving and deleting patterns, appending additional pattern readings to already stored system patterns and retrieving information about the stored patterns.

This class manages the pattern data stored in the database. Figure 6.9 presents the database structure used in this thesis to store system patterns and events. To normalize the database tables, the variance and mean/minimum message throughput of each sensor node is stored as a special pattern reading with different identification types.

The combination of system patterns is performed by the *PatternBase* class. The implementation of this process follows the description of the pattern combination method in Section 5.3.3.

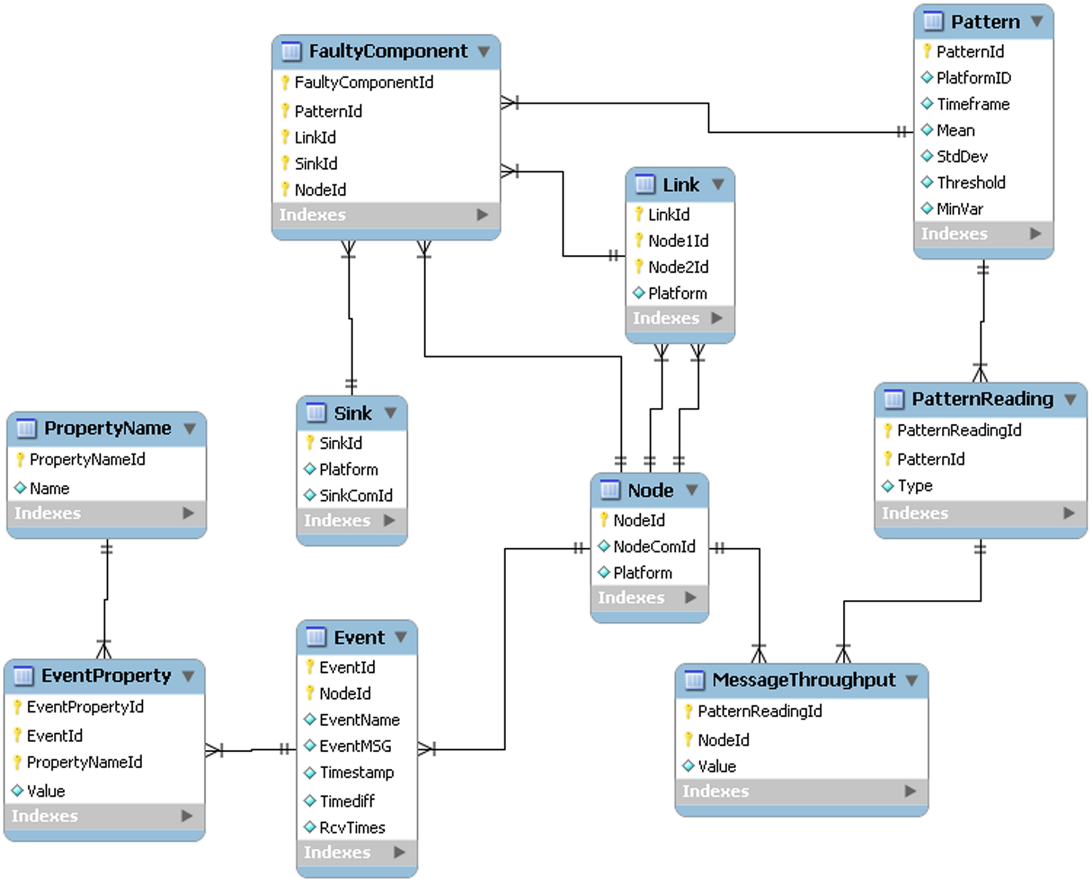


Figure 6.9: Database scheme.

Membership Evaluator

The *MembershipEvaluator* class defines a method called "isMember" where a diagnostic signal is analyzed with the goal to determine if it belongs to a given system pattern. This analysis uses the "calculateNormalizedEuclideanDistance" method to determine the distance of the current diagnostic signal from the mean of the current pattern.

According to the current threshold and the calculated distance, the membership method will return a boolean value indicating if the diagnostic signal belongs to the given pattern or not. This class also implements a method to calculate the distance threshold. This method is only called when the threshold is modified. The calculated value is stored in the *PatternBase* together with additional statistical properties.

Failures Match Maker

The *FailuresMatchMaker* class follows the specification defined in Section 5.3.6. This class offers one method to determine the possible system states of a given diagnostic signal. During the system state analysis, the *FailuresMatchMaker* class uses the method "isCrashFailure" from the *Transient Analyzer* class to return the type of failure affecting the node, i.e interference failure or crash failure.

Neural Classifier

The implementation of the *NeuralClassifier* class integrates the facilities of the Java programming language with the high performance of the Joone framework [72].

Joone offers a neural network framework with built in functionalities to train and execute multi-layer neural networks. Joone is fully implemented in Java, which facilitates its adoption for this thesis.

6.3.2 FT-WiseNets User Interface

The functionalities of the *Pattern Base* defined in Section 5.3.3 were implemented together with a GUI to facilitate the acquisition and combination of system patterns. The GUI designed for this component is composed of a *Pattern Acquisition View* and a *Pattern Combination View*.

A GUI was also developed to depict the online results of the fault isolation process from each of the remaining components of the fault isolation process: *Membership Evaluator*, *Failures Match Maker*, and *Neural Classifier*. This Section presents the implementation details of each of these components.

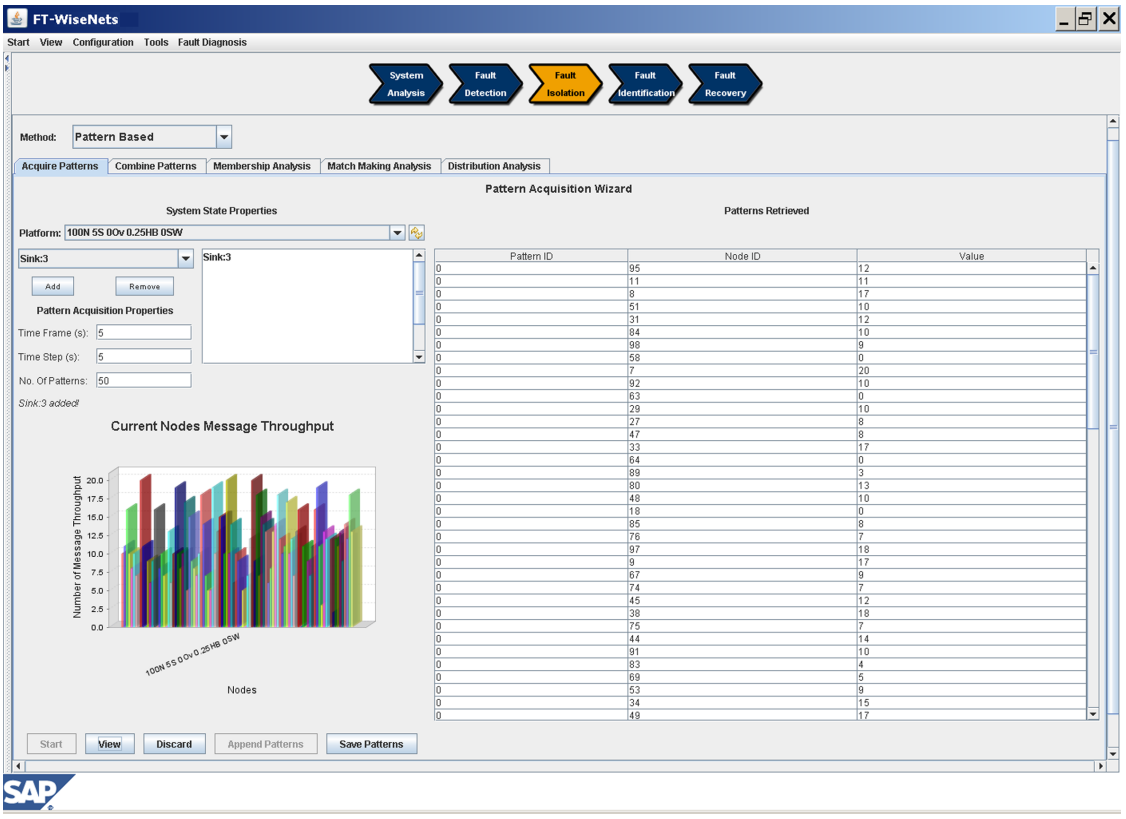


Figure 6.10: Pattern acquisition view.

6.3.3 Pattern Acquisition View

The main functionality of the *Pattern Acquisition View* is to provide a GUI that assists maintenance worker in acquiring system patterns, guide the acquisition of patterns during normal operation and sink failures, and store the patterns in the pattern base.

This view uses the information from the *Device Manager* to display the platforms registered in the framework. Once the platform is selected, additional parameters must be provided by the user to start the pattern acquisition process. Such parameters include: timeframe, time step, number of desired readings, and the current state of the system.

Figure 6.10 depicts the *Pattern Acquisition Wizard* view. As visual feedback, a graph is displayed containing the message throughput of each node during the specified timeframe. At the end of the acquisition phase the values acquired are displayed in a table.

The pattern acquisition task is executed in a separate *Thread* to detach the system load (caused by the user interaction with the GUI) from the periodic time sensitive operation of acquiring a pattern reading.

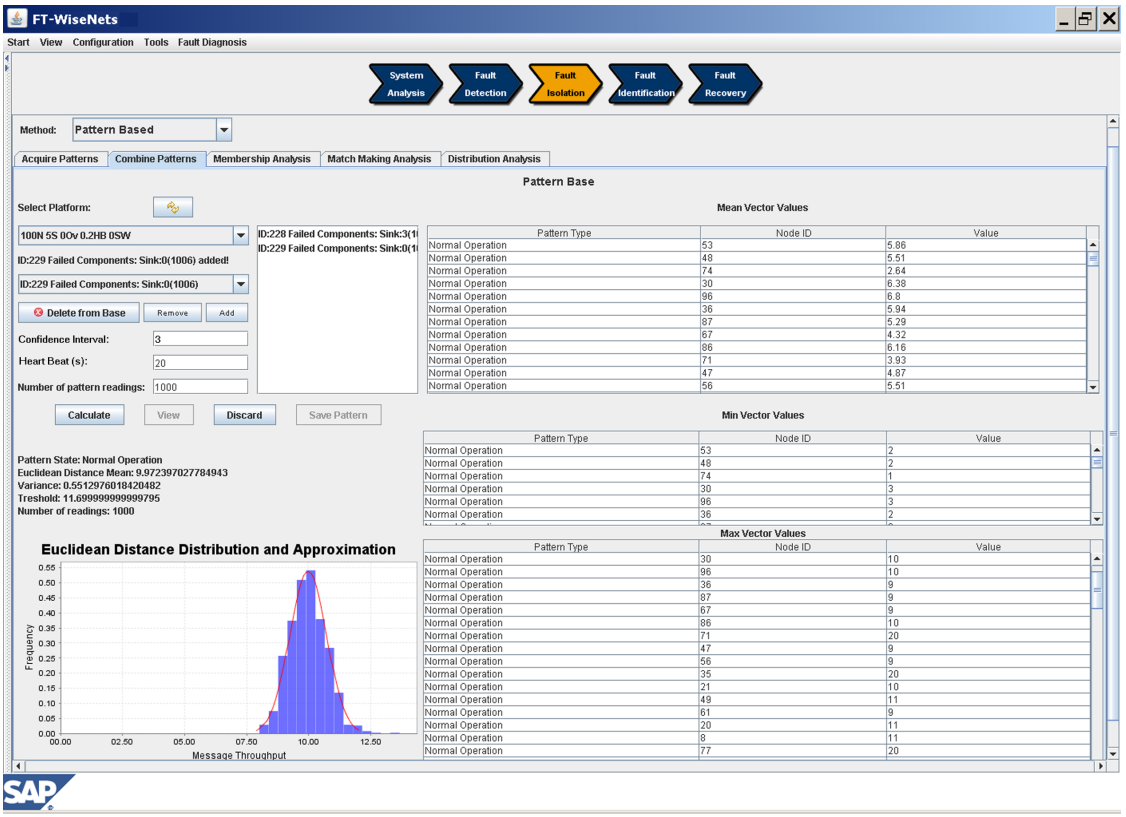


Figure 6.11: Pattern combination view.

6.3.4 Pattern Combination View

The fault isolation method proposed in this thesis is based on probabilistic information of each pattern stored in the *Pattern Base*. The *Pattern Combination View* is developed in order to provide a simple user interface to calculate the probabilistic values, associated with each pattern, and combine already known patterns.

Figure 6.11 depicts the *Pattern Combination View*. This view allows the selection of platforms based on information provided by the *Device Manager*. The patterns stored in the *Pattern Base* are available for the pattern combination process.

This view is used to calculate the metrics of single or combined patterns. In both cases the user must provide additional information: confidence interval, and heart beat. After the calculation is performed, a histogram displays the distribution of the *Normalized Euclidean Distance* for the calculated combined/single pattern and the Gaussian approximation curve.

6.3.5 Online Isolation Views

During runtime it is possible to visualize the results from each step of the fault isolation process through the *Online Isolation Views*. These views (depicted in

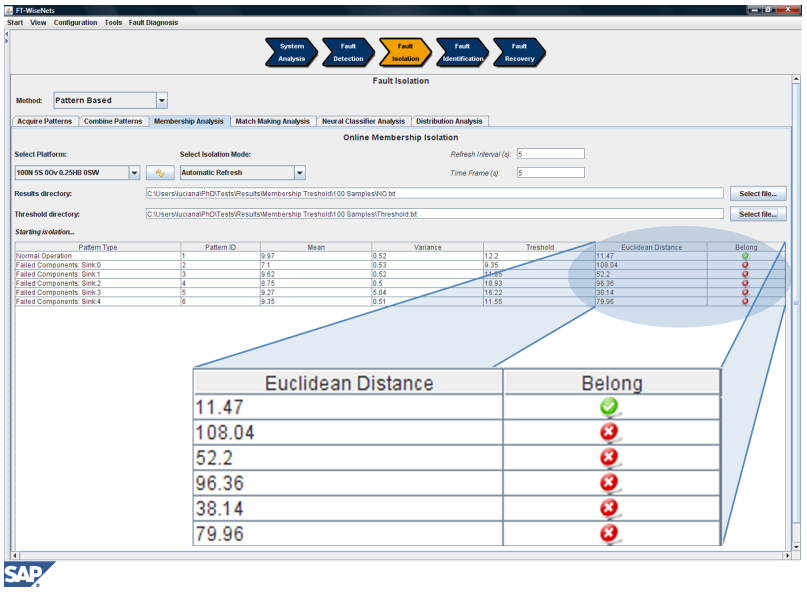


Figure 6.12: Online isolation view of the Membership Evaluator.

Figures 6.12, 6.13, and 6.14) provide a graphical representation of the results from the following components:

- Membership Evaluator
- Failures Match Maker
- Neural Classifier

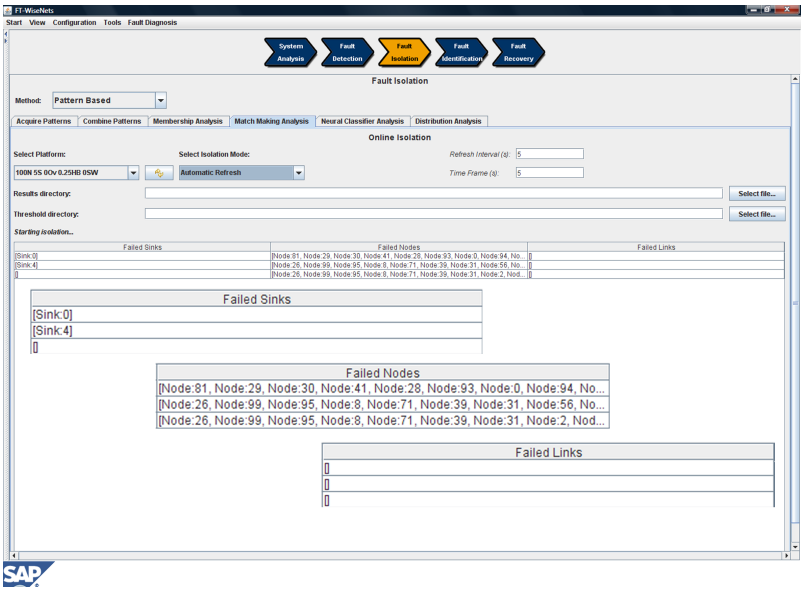


Figure 6.13: Online isolation view of the Failures Match Maker.

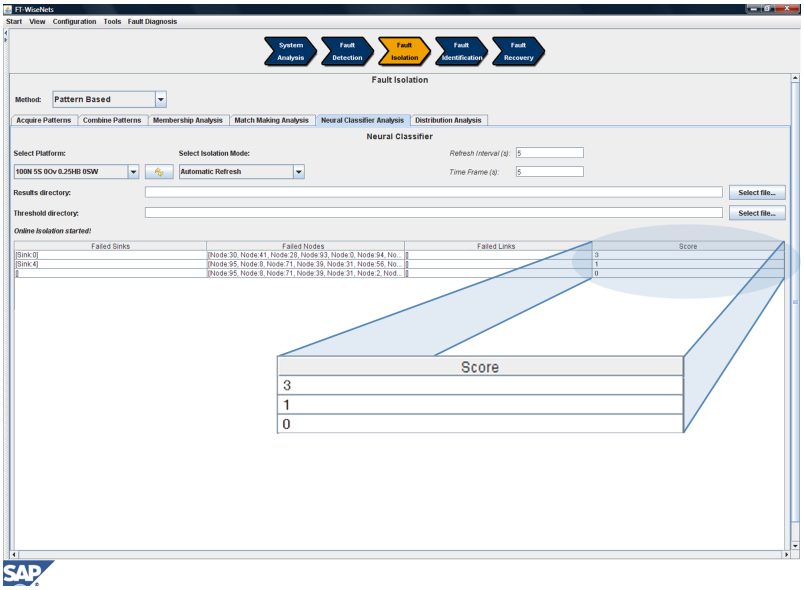


Figure 6.14: Online isolation view of the Neural Classifier.

7. Pattern Based Fault Isolator Evaluation

In this Chapter, we evaluate the fault isolation approach proposed in this thesis. The evaluation was performed in order to verify the prediction model defined in Section 5.4 and in order to investigate the overall efficiency of the proposed algorithm.

The data used for the evaluation tests were generated from an application trial and from simulation, as described in Chapter 6. This enabled investigating the limitations of the fault isolation method through simulation, and confirmed that the approach works in theory and in practice.

We analyzed the performance of our approach in terms of False Positive (FP) and Fault Isolation Effectiveness (FIE). FP indicates the number of failures identified while no failures were present in the component. FIE indicates the percentage of failures present and correctly identified by the system.

7.1 Simulation Procedures

The simulations were performed on a computer with an Intel Core 2 Duo 1.83 GHz, 4GB of RAM, running on Windows Vista. The Simulator and Monitoring Application were started from the Eclipse IDE using Java version 1.6.11. The database was on the same machine to prevent delays in the delivery of the generated events due to network latency.

7.1.1 Membership Evaluator

The membership function is the core element of the fault isolation method defined in this thesis, as its efficiency directly impacts the performance of the other compo-

nents. Therefore, we have evaluated the *Normalized Euclidean Distance* membership function to determine its limitations and the best choices for the threshold and time-frame parameters.

7.1.1.1 Threshold Definition

The definition of the threshold in the *Membership Function* determines the boundaries within which a diagnostic signal will be considered as part of a pattern. Ideally the definition of the threshold should allow 100% of FIE and not generate FP. As the diagnostic signals from WSNs are generated based on a probabilistic model, this efficiency cannot be achieved.

Although improbable, it is possible that at a given time, the WSN will generate a diagnostic signal that is distant from the average value but still belongs to it. To include these possible distant diagnostic signals, the threshold would need to be set to a value that would generate a high number of FP. This high number is due to overlaps with other patterns known to the system.

Therefore, defining the threshold should aim at maximizing the FIE and minimizing the number of FP. To achieve this task, in this thesis we define a threshold function based on a *Gaussian distribution approximation*. We made the following prediction for this threshold function defined in Section 5.4:

Prediction 1: *"In networks where a Gaussian distribution approximation is suitable and no near overlaps with other patterns exist, the threshold from the membership function will determine its fault isolation effectiveness as the number of acquired pattern readings and analyzed diagnostic signals tend to infinity."*

Simulation Setup

To confirm the prediction made, we performed two tests. The first one evaluated the FIE of a network without overlaps for different numbers of acquired pattern readings. The second test evaluated the FIE of networks with 10, 25, and 50 percent overlap with a fixed number of acquired pattern readings.

Test 1

The first test used a simulated WSN containing 100 nodes and 5 sinks in a three-hop network. The topology of this network was randomly generated with a link quality varying from 0.5 to 1. For generating this topology, we set the network overlap to zero.

The heart beat of the simulated sensor nodes was set to 250ms. This value was selected in order to execute a fast simulation, and considering the time required for

the propagation of all events generated by the sensor nodes, and provided a margin to accommodate system fluctuations. The timeframe of 5 seconds was selected, which included 20 attempts from the sensor nodes to send a message.

The test evaluated the FIE of the membership function considering thresholds of μ_D, $\mu_D + \sigma_D$, $\mu_D + 2\sigma_D$, $\mu_D + 3\sigma_D$, and $\mu_D + 4\sigma_D$.

To investigate the influence of the number of pattern readings on the efficiency of the system, the test was performed for pattern configurations containing 50, 250 and 1000 pattern readings.

The system states evaluated were normal operation and each isolated sink failure. A total of 18.000 diagnostic signals were evaluated. These diagnostic signals were divided in 6.000 per pattern configuration, i.e 1.000 diagnostic signals per system pattern.

Test 2

The second test also used a simulated WSN containing 100 nodes, 5 sinks, and a link quality varying from 0.5 to 1. The three additional networks were randomly generated using three layers, but configured to 10, 25, and 50 percent overlap. The heart beat and timeframe used the same values from *Test 1* (250ms and 5 seconds).

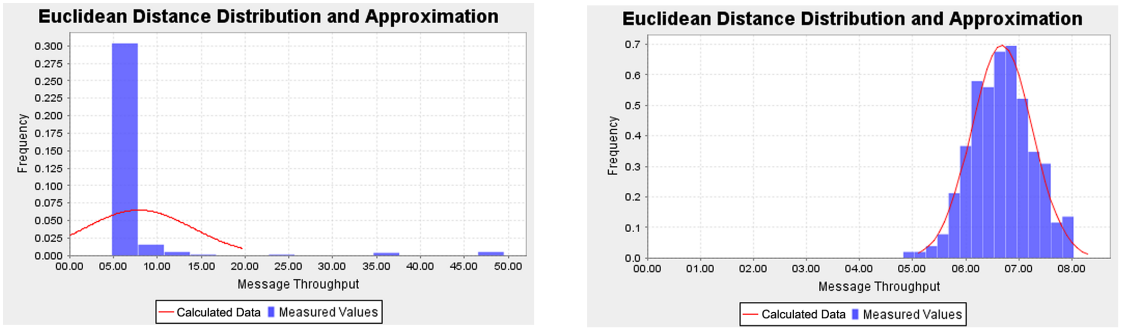
Each system pattern was acquired with 250 pattern readings. As in *Test 1*, 1.000 diagnostic signals were evaluated per system pattern. In total, 18.000 diagnostic signals were evaluated for the second test.

Pre-processing procedure

As predicted in Section 5.4, when the variance of sensor nodes throughput approaches zero, the distribution of DEN may present outlier readings.

Figure 7.1(a) presents an example of this phenomenon, where 5 readings out of 250 became far from the average due to three nodes reporting message throughputs with a difference of one message from the expected average. This situation is crucial during the pattern acquisition phase, since these outlier readings cause a strong impact on calculating the statistical properties of a pattern, as depicted by the normal approximation in this Figure.

Hence, we approached the problem of poor Gaussian distribution approximation due to outlier readings, by eliminating readings that are further away than $5\sigma_D$ from the averaged normalized euclidean distance (μ_D). The threshold of $5\sigma_D$ was chosen in order to contain the majority of the readings but still be able to remove these



(a) Raw acquired pattern distribution

(b) Post- processed distribution

Figure 7.1: Pre-processing procedure improves the normal distribution approximation.

outlier readings. This procedure is iteratively executed until no more readings are eliminated from the pattern.

Figure 7.1(b) presents the distribution of the pattern depicted in Figure 7.1(a) after applying the pre-processing procedures defined here. As depicted in this graph, the procedure improves the normal distribution approximation of the majority of acquired samples.

Simulation Results

Figure 7.2 depicts the histogram and calculated normal distribution of each system pattern acquired in *Test 1* for 1000 readings. As expected, the observed distributions approximate a Gaussian curve. Similar curves were obtained for 50 and 250 readings. These curves were then used as a basis for evaluating diagnostic signals in a cross-validation procedure.

Figure 7.3 depicts a graph containing the results from *Test 1*, and presents the FIE for a network without overlap, different settings of pattern readings, and thresholds. This graph confirms the following prediction made: with an increase in the amount of pattern readings the fault isolation effectiveness will tend to migrate to the cumulative density function value, represented by the "Ideal" line. The values for the "Ideal" line are taken from Table 5.1 in Section 5.4.

The reduced performance of the system when 50 pattern readings are used is a direct consequence of errors in the calculated average and standard deviation. This is due to the reduced number of samples. Although the fault isolation effectiveness migrates to the "Ideal" line as the number of readings increases, it is possible to note that there is error associated with the calculated average and standard deviation even with 1.000 readings.

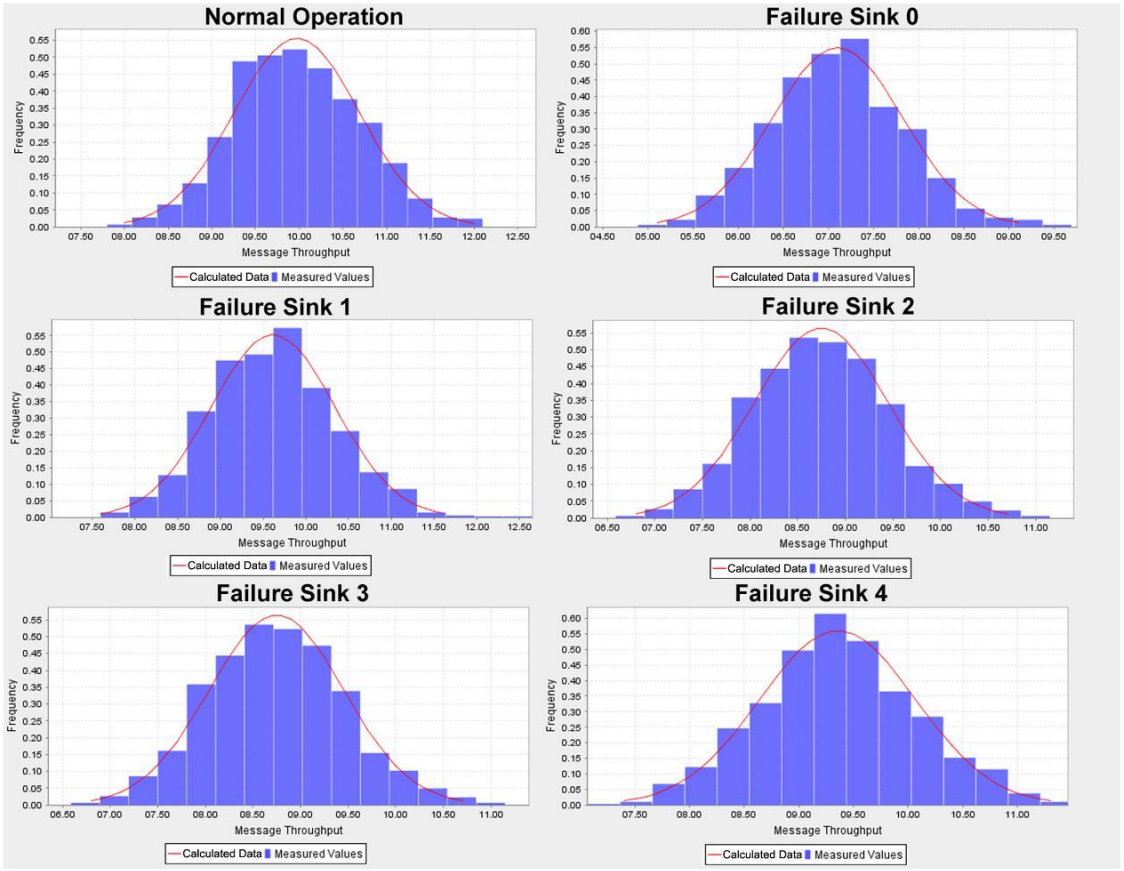


Figure 7.2: Normalized euclidean distance distribution for each pattern analyzed.

Figure 7.3: Influence of threshold and number of pattern readings on fault isolation effectiveness. With more pattern readings the FIE approaches the “Ideal” value.

Figure 7.4: Influence of threshold and number of pattern readings on fault isolation effectiveness. Membership function behaves in the same manner for networks with and without overlap.

The second test performed, confirms the prediction made also for the case of networks with overlaps. Figure 7.4 presents the results of the simulation performed on *Test 2*. Each point in the graph was calculated based on six thousand diagnostic signals, being one thousand diagnostic signals per system pattern. The value plotted on the graph is the average efficiency for the six system states analyzed.

All networks had a similar FIE and resulted in zero FP among the acquired patterns with 4σ threshold. Nevertheless, we noticed an increase in the amount of outlier readings as the network overlap raised.

To investigate the reason for not accepting diagnostic signals as part of the pattern, for each network setup we analyzed the number of diagnostic signal occurrences between: 4σ and 5σ, 5σ and 6σ, and beyond 6σ. The result of this evaluation is presented in Figure 7.5.

In a normal distribution it is expected that the number of occurrences decreases as we observe intervals further away from the average. Nevertheless, as depicted in the graphs from Figure 7.5, there is a considerable increase in the number of readings beyond a distance of 6σ as the overlap raises.

This effect is a direct consequence of the timeframe used in the tests and confirms the assumption made, that heavily connected networks are more likely to generate outlier readings. Outlier readings are originated from the error introduced by

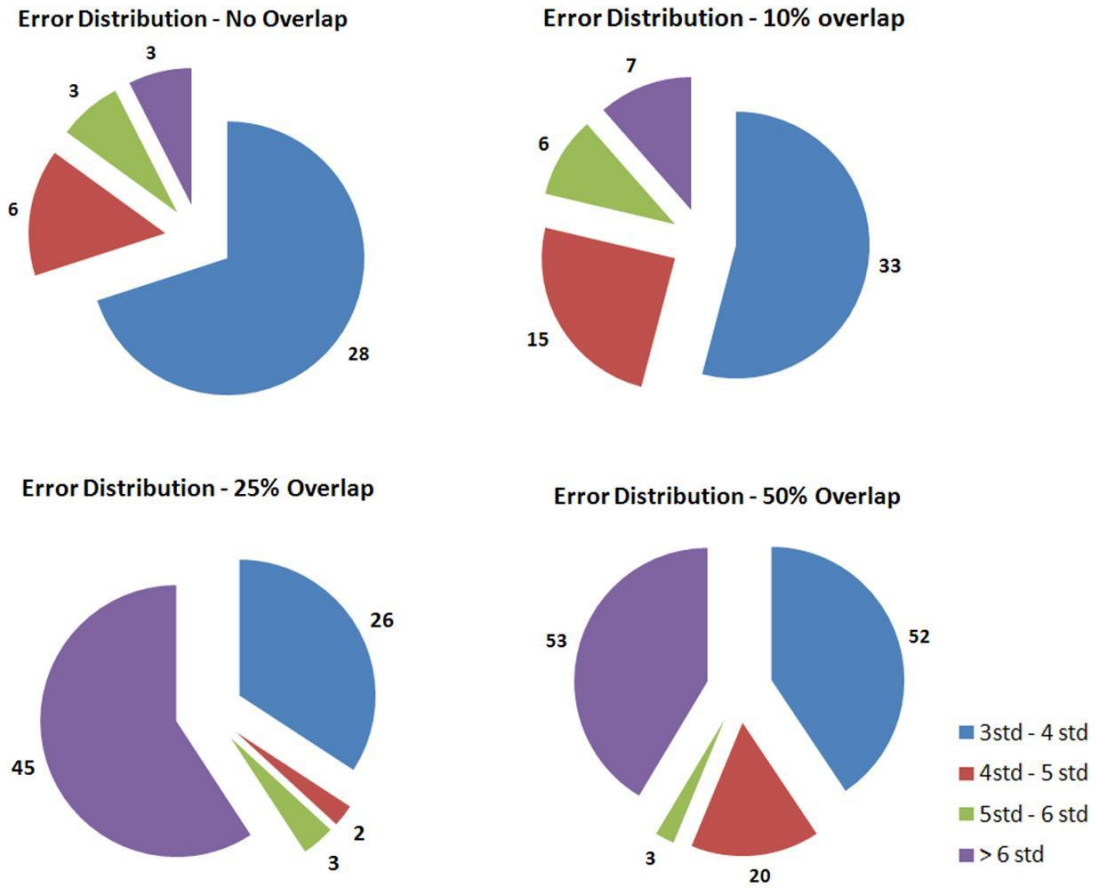


Figure 7.5: Diagnostic signal distribution for different topologies setups. There is an increase in the number of outlier readings as the overlap raises.

the normal approximation of the sensor nodes message throughput when the variance approximates zero. A further investigation of outlier readings proceed in the following Section.

7.1.1.2 Timeframe

The timeframe parameter determines the number of message sending attempts each node performs during a pattern reading period. Modifying this parameter affects the distribution approximation of the sensor nodes' message throughput, as described in Section 5.4.

The use of longer observation periods provides a better approximation of the sensor nodes' message throughput, and therefore, it is a technique that can improve the efficiency of the membership function. Although this benefits the performance of the membership function, the use of prolonged timeframes also impacts the time required to diagnose a system. In order to diagnose the system, and provide precise results, the system must remain in the same state for the timeframe period.

Therefore, short timeframes and high efficiency are desirable characteristics in a fault diagnosis system. Additionally, the time required to train the system should also be minimized to reduce the costs associated with the system setup.

Hence, we investigate the limitations of reducing the timeframe value and impact on the efficiency of the membership function. Additionally we perform tests to confirm the *Prediction 2* made in Section 5.4.

Prediction 2: *"In networks where the path quality between nodes and sinks is close to one, outlier readings are more likely to occur as the number of message sending attempts for each pattern reading is reduced."*

Simulation Setup

Test 3

For this evaluation test we used a simulated WSN network containing 105 nodes and 5 sinks in a one-hop topology. The link quality was defined in a range from 0.6 to 0.8 for 100 nodes, while 5 nodes had a link quality of 0.99. The values of the links were selected in this manner in order to generate outlier readings.

The normal operation system pattern was acquired containing 1000 readings. This pattern was acquired with three timeframe setups to analyze the influence from this variable on the fault isolation effectiveness of the membership function.

The timeframe values used for this evaluation were 5, 20 and 50 attempts. These values were selected to provide a range from a low value (5), to a medium value (20) that provides an efficiency close to the ideal with 1000 readings, as presented in Section 7.1.1.1, and up to a high value (50). In a real setup, the time required for 50 messages sending attempts is high and therefore not practical. In here we use this value only to illustrate the impact of longer timeframes on the system results.

The goal of this test is to verify the number of readings beyond $3\sigma_D$ for each timeframe configuration.

Simulation Results

The results obtained from *Test 2*, presented in Figure 7.5, indicate that *Prediction 2* is correct. As networks with overlaps have more links connecting sensor nodes, the probability of messages reaching the backend increases significantly for some nodes, while other nodes may remain with a lower path quality. This has the same effect as networks that contain some nodes with a path quality to the sinks close to one.

As a consequence, in *Test 2* the variance of the message throughput of several sensor nodes became close to zero as the overlap increased. This resulted in an inaccurate distribution approximation and increased the amount of outlier readings for the network with 50% overlap.

Figure 7.6 depicts the histograms of the message throughput of a sensor node with link quality equal to 0.99 for the three timeframe configurations. These histograms were generated from the data acquired for *Test 3*. As depicted, with 5 attempts, the distribution is poorly approximated by a normal distribution. The discrete distribution mainly consists of a "peak" close to five indicating that the majority of readings receives all messages from the node. As the timeframe increases, the distribution approximation becomes more accurate, although a cut on the right side of the curve still exists.

The variance calculated for this sensor node is approximately equal to 0.04 with 5 message sending attempts per pattern reading. With this variance, the node seldom generates readings with a difference f one message from the average, which represents a distance contribution equal to $(\frac{1}{0.04})$. This distance contribution adds a value close to 25 to the calculation of DEN. When two or three nodes generate values with a difference of one message from the average, the diagnostic signal additionally adds 50 or 75 to the calculated value of the normalized distance.

The outcome of such an event is outlier readings. These readings can be expected to appear with a value close to $\sqrt{DC + 25}$, $\sqrt{DC + 50}$, and $\sqrt{DC + 75}$, where DC is

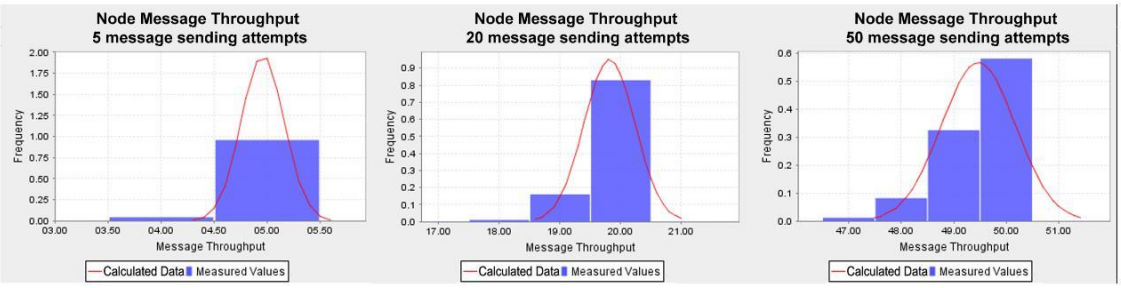


Figure 7.6: Node message throughput distribution with different timeframe setups. The distribution is better approximated by a Gaussian curve as the timeframe increases.

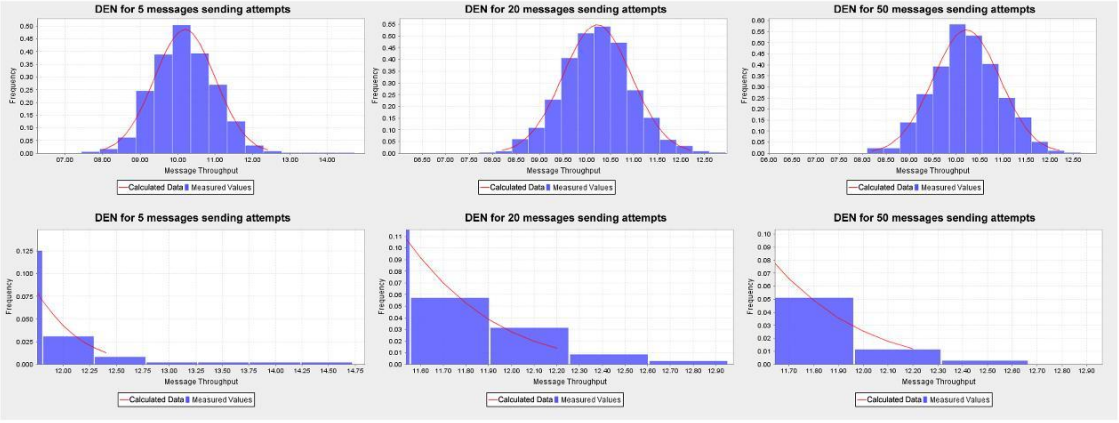


Figure 7.7: Diagnostic signal distribution for different topology setups. The number of outlier readings reduces as the timeframe increases.

the distance contribution from all nodes but the nodes with low variance (link quality equal to 0.99). As the nodes with low variance generate a distance contribution to DEN close to zero in most of the readings, the values expected from DC can be approximated by DEN^2.

Figure 7.7 depicts the histograms of DEN for the three timeframe configurations of *Test 3*. This test confirms *Prediction 2*, demonstrating that lower timeframes are more likely to generate outlier readings if the network contains nodes with low message throughput variance.

In this graph DEN varies from zero to approximately 12,75 (not considering the outlier readings) for the configuration with 5 message sending attempts. Therefore readings are likely to occurr from $\sqrt{0^2 + 25} = 5$ to $\sqrt{12.75^2 + 75} = 15.41$. The results from *Test 3* show values occurring up to 14.7, this value is very close from the expected maximum being only a bit lower than the calculated value. This can be a result of the approximation of DC by DEN, or due to the probabilistic nature of the simulated test.

In total, 5 readings were identified lying beyond 3σ for the configuration with 5 message sending attempts, 3 readings for 20 message sending attempts and one reading for 50 message sending attempts. The ideal value is one reading, since 3σ is equivalent to 0,998 of the population when 1000 diagnostic signals were evaluated (approximately 1,34 readings).

As the quality of the network links increases, it is more likely that outlier readings occur. This impacts the FIE of the membership function, as these readings are likely to lie beyond the defined threshold. Defining very long timeframes is also not desirable, therefore a balance between efficiency and fault diagnosis time has to be determined for each application setup.

7.1.2 Pattern Combination

In Chapter 5, we described a method to reduce the time necessary to acquire various system patterns by combining already known sink failure patterns. The proposed method calculates the influence of sink failures on the message throughput of sensor nodes and uses this information to determine the statistical properties of combined failures.

In this Section we verify this method by applying it to networks with different overlap configurations.

Simulation Setup

Test 4

The network topologies applied in this evaluation are the same ones used in *Test 1* and *Test 2*. These networks consist of 100 nodes, 5 sinks and are configured to have 0%, 10%, and 50% overlap. These topologies were randomly generated with the *Topology Generator*, discussed in Chapter 6, and the link quality varied from 0.5 to 1.

The test consisted of the following steps:

- Calculate the statistical properties of a combined failure using acquired isolated failure patterns as the basis
- Acquire a pattern with the same set of failed components as the combined failure
- Calculate the statistical properties using the acquired pattern

- Compare the statistical properties extracted from the calculated combined pattern and the acquired one

For each isolated sink and normal operation pattern, 1000 pattern readings were acquired. This helps reduce the error caused by sampling. As analyzed in Section 7.1.1.1 this amount of readings already provides statistical properties that approximate the "real" value of the system.

Simulation Results

Figure 7.8 presents a comparison between the mean message throughput value calculated for each node, and the values acquired through simulation. The combined failure causes several nodes to completely interrupt their communication with the back-end. The result is a message throughput equal to zero. This message throughput is represented by the "empty" slots in this graph.

In Figure 7.9, a similar comparison is made. In this graph, however, the calculated variance is compared to the acquired one. Figure 7.10 presents the errors associated with the *Normalized Euclidean Distance* average (μ_D) and the standard deviation (σ_D). This error is calculated by subtracting the respective computed values from the acquired value. This graph proves that the approach is correct since only minor errors exist and is a probable consequence of sampling.

The results for networks with overlaps showed similar behaviour. The graph results for these networks can be found in Appendix B. It is interesting to note that the pattern combination approach has the same limitations of the membership function. As the normal distribution approximation of the message throughput of sensor nodes becomes inaccurate, the results from the calculated combined failures also present more errors.

7.1.3 Transient Analyzer

The *Transient Analyzer* component is responsible for determining if a specific node is healthy, or if it suffered a crash or link failure. As described in Section 5.4, this component executes its task by analyzing the degradation of the message throughput signal over time. In this Section we perform tests to confirm the following predictions made:

Prediction 3: *"The Transient Analyzer has a higher rate of misdiagnosis between link failures and crash failures as the interference increases."*

Prediction 4: *"The Transient Analyzer has a lower fault isolation effectiveness as the interference becomes increasingly small."*

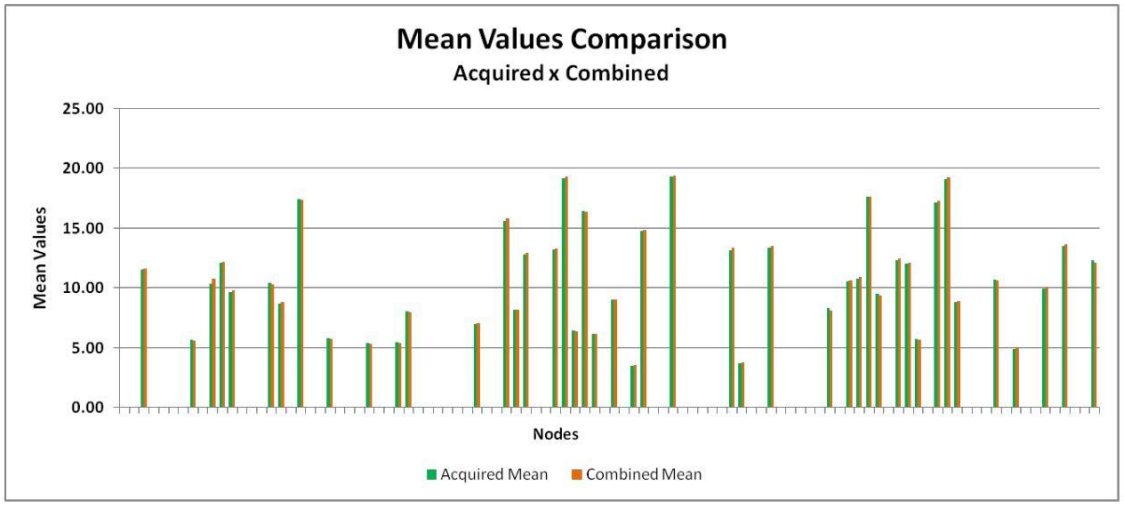


Figure 7.8: Comparison between calculated and acquired mean values.

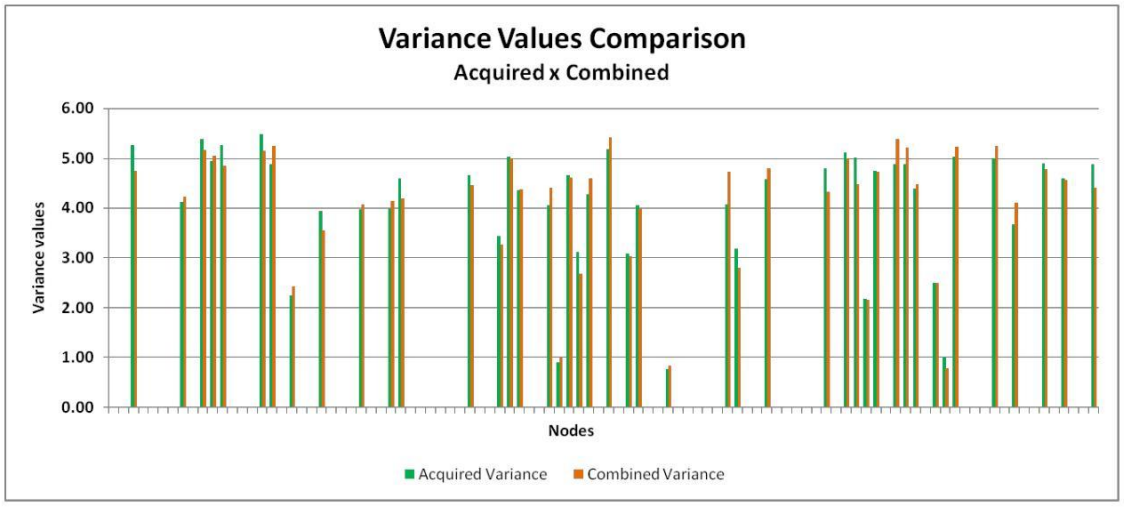


Figure 7.9: Comparison between calculated and acquired variance values.

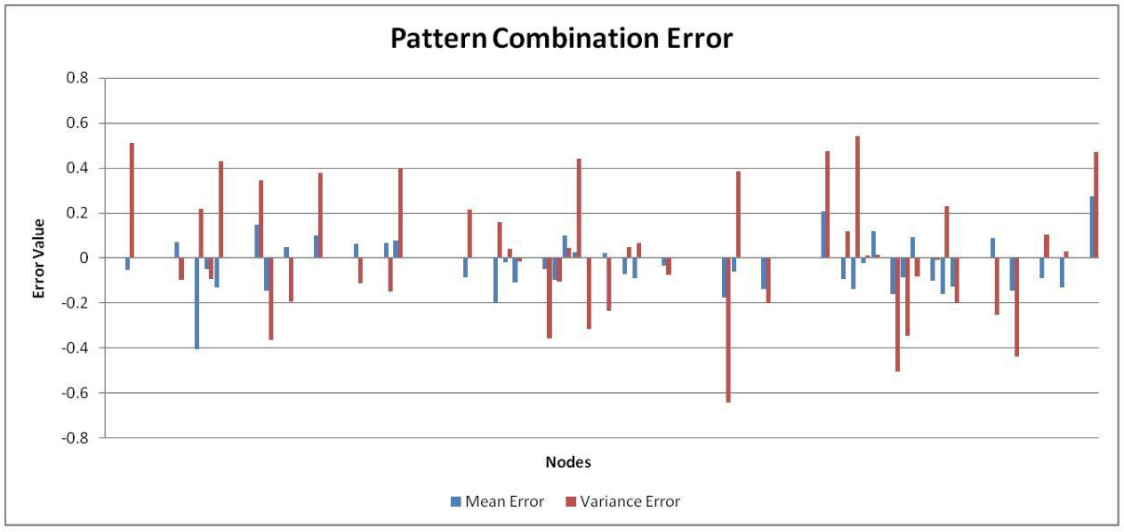


Figure 7.10: Error between calculated and acquired mean and variance values.

Simulation Setup

Test 5

This test evaluates the ability of the *Transient Analyzer* to identify link failures in the system. For this evaluation test we used the same simulated WSN from *Test 1*, which contains 100 nodes and 5 sinks in a three-hop topology. The link quality of this network varies from 0.5 to 1.

During this test, the simulator randomly inserted link failures in the system every 50 seconds. Between each randomly generated set of failures, the system was put to a period of normal operation. Each failure set inserted interference in 10% of the links in the network. The interference inserted in the links had values of: 0.1, 0.25, 0.5 and 0.75. The interference inserted for this test reduces the value of the link quality ζ. If the interference is greater than the link quality, ζ is set to zero.

We also performed separate tests with the system in normal operation to evaluate the number of FP indicated under this condition.

As the *Transient Analyzer* component used a message throughput threshold for each sensor node, we evaluated its performance for threshold values varying from 1σ to 5σ.

Test 6

In this test, we evaluated the performance of the *Transient Analyzer* when it is requested to evaluate nodes that are not sending messages. The same network from *Tests 1 and 5* was used. For this test, every 50 seconds a random set of node crash failures were inserted into the network by the simulator, followed by a period of normal operation. Each set of failures included 10 node crash failures. The threshold for the *Transient Analyzer* was set to values varying from 1σ to 5σ.

For Tests 5 and 6, the evaluation was performed using 1000 pattern readings for the system in normal operation. For each combination of threshold and interference, 100 diagnostic signals were evaluated. Sink failures were not inserted in the system in order to ensure that observed results were not affected by additional failures.

Simulation Results

As the link interference and crash failures of one node may reduce the message throughput of several other nodes, the evaluation tests verified if the diagnosed link and node failures were within the generated failures set that can be observed by the back-end (F_o).

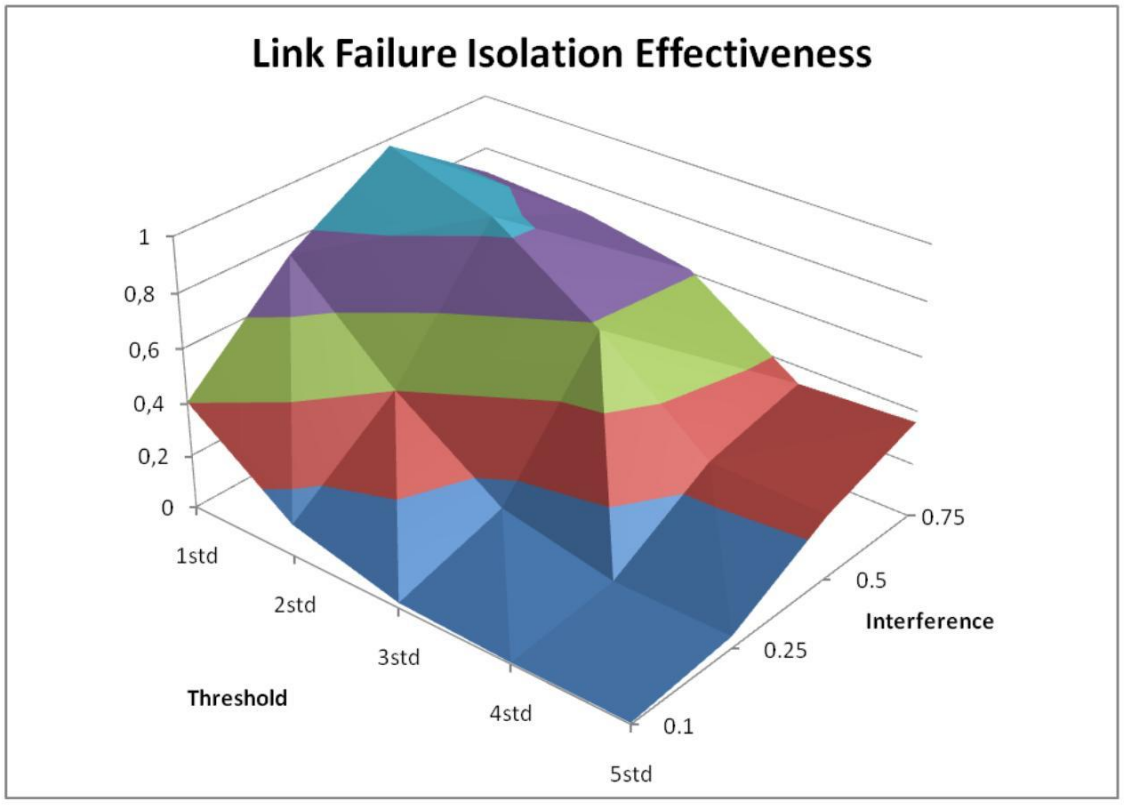


Figure 7.11: Fault isolation effectiveness for link failures. The FIE increases as the interference raises, until the point where misdiagnosis start to occur.

Figure 7.11 depicts the fault isolation effectiveness for *Test 5*. With an interference of 0.5 and 1σ as the threshold, the *Transient Analyzer* provides a fault isolation effectiveness of failed links of 97,5%. With lower interferences, the fault isolation effectiveness reduces as it was expected according to *Prediction 4*.

In this graph it is interesting to note that the fault isolation effectiveness reduces for interferences of 0.75 when compared to interferences of 0.5. This occurs since strong interferences have a similar effect as node crash failures and are therefore misdiagnosed. Figure 7.12 presents the rate of link misdiagnosis for *Test 5*.

As expected from *Prediction 3*, the results show that as the interference increases, there is also a raise in the number of misdiagnosis, indicating crash failures while the real failure is due to link interference.

It is also interesting to note the number of FP generated for each test setup. The number of FP originate from the message throughput distribution of the nodes. Given a large enough number of attempts, nodes will generate values that deviate from the average. Table 5.1 presents the percentage of generated values that can be expected to stay within a given threshold.

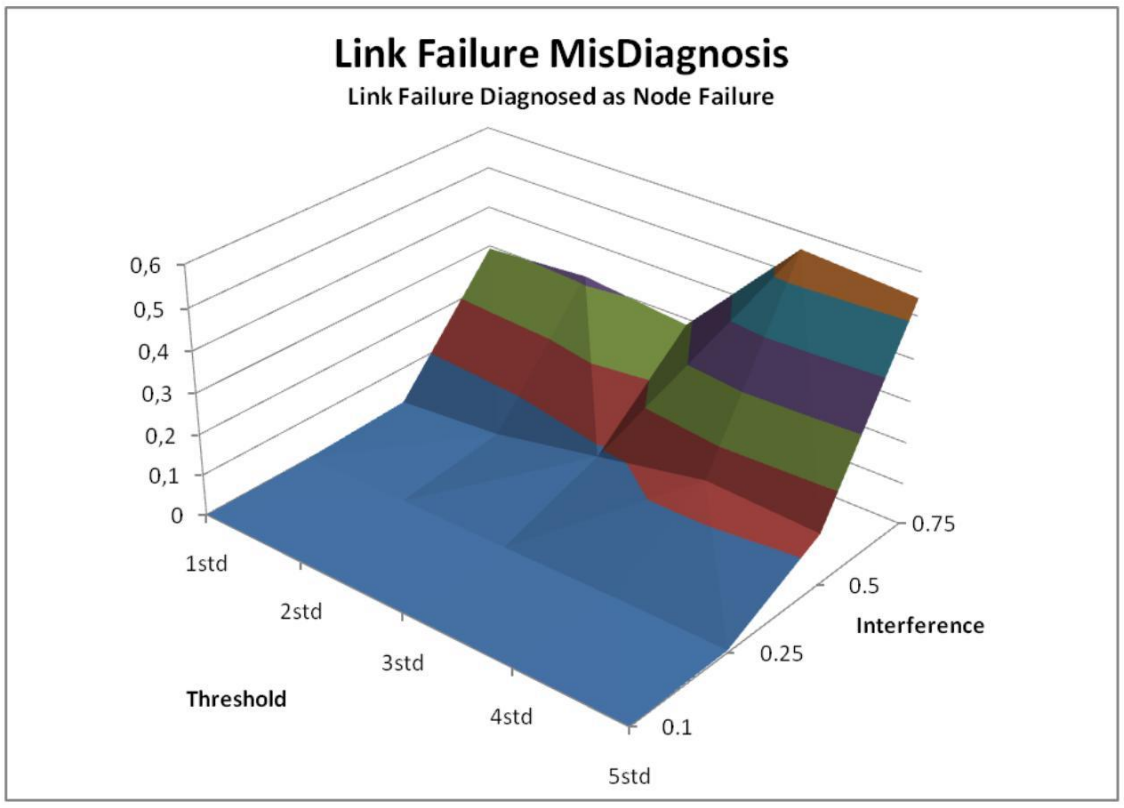


Figure 7.12: Misdiagnosis: link failure diagnosed as node failure. A higher number of misdiagnosis occur as the interference increases.

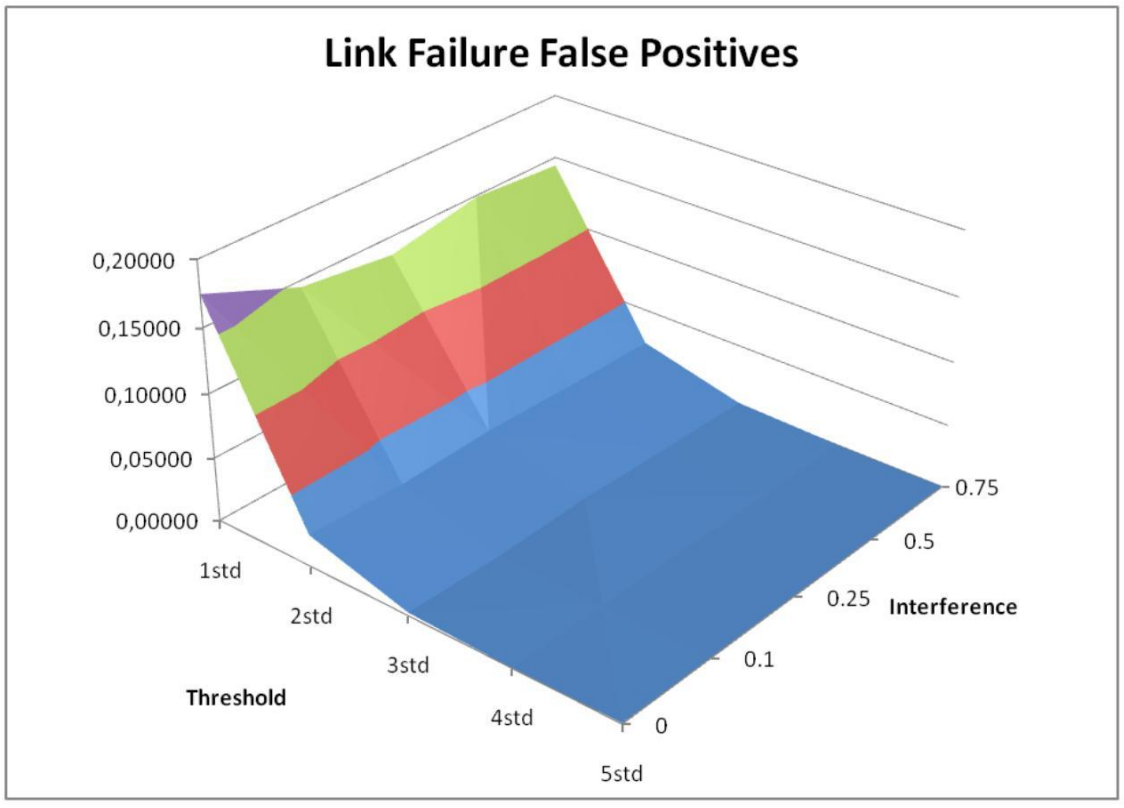


Figure 7.13: Link failure false positives. There is a high number of false positives for a threshold of 1std.

The number of expected FP can be calculated based on the defined threshold and the values from this table. In the case of 1σ the expected number of FP is 15,9%(1 - 0.841). For 2σ this value is 2,3%. Figure 7.13 depicts the values generated through simulation.

With 1σ as the threshold, the number of FP generated is very high and varies from 17,47% to 13,6% of all link evaluations for the system, in normal operation and with link interference. This high number of FP makes the use of this threshold value disadvasible for the isolation of failures. With 2σ, the number of FP drops to 2,4% of all link evaluations during normal operation. This shows that the simulated values are very close from the calculated theorethical values.

Another interesting aspect of the system is that the number of FP decreases as we insert link failures in the system. This occurs since the number of attempts that are not a link failure become lower than the total number of attempts.

The results from *Test 6* show the efficiency of the *Transient Analyzer* for the evaluation of crash failures, and can be visualized in Figures 7.14, 7.15, 7.16, and 7.17. By analyzing these graphs we conclude that with 1σ, the system has a high fault isolation effectiveness, and is able to identify 100% of the failures. This high efficiency, however, comes at the cost of FP and node crash misdiagnosis.

As the threshold increases, the effectiveness of the *Transient Analyzer* reduces. This is expected if the node does not generate any messages and is identified either as a link or crash failure. Nevertheless, with a greater value of the threshold, the system is less likely to misdiagnose a crash failure for a link failure, and performs better in evaluating healthy nodes.

The number of FP for this evaluation was low and is a result of the probabilistic nature of the system. With enough observations, a node will fail at some point on all its attempts to send messages to the back-end in a given timeframe. This, however, occurs with a low probability.

With the results from *Test 5* and *Test 6* we can analyze the best operation point for the *Transient Analyzer*. First, it is important to determine the the sensibility of the system for the identification of link failure. If minor alterations on the link quality must be observed, the selected threshold should be very small, such as 1σ. This will identify most link failures, but will also generate a high amount of FP.

If the goal is to detect link failures that cause a quality degradation of around 0.5, a value of 2σ is more recommended. With this threshold, the expected number of FP drastically reduces to 2,3%, the link isolation effectiveness of the system for this

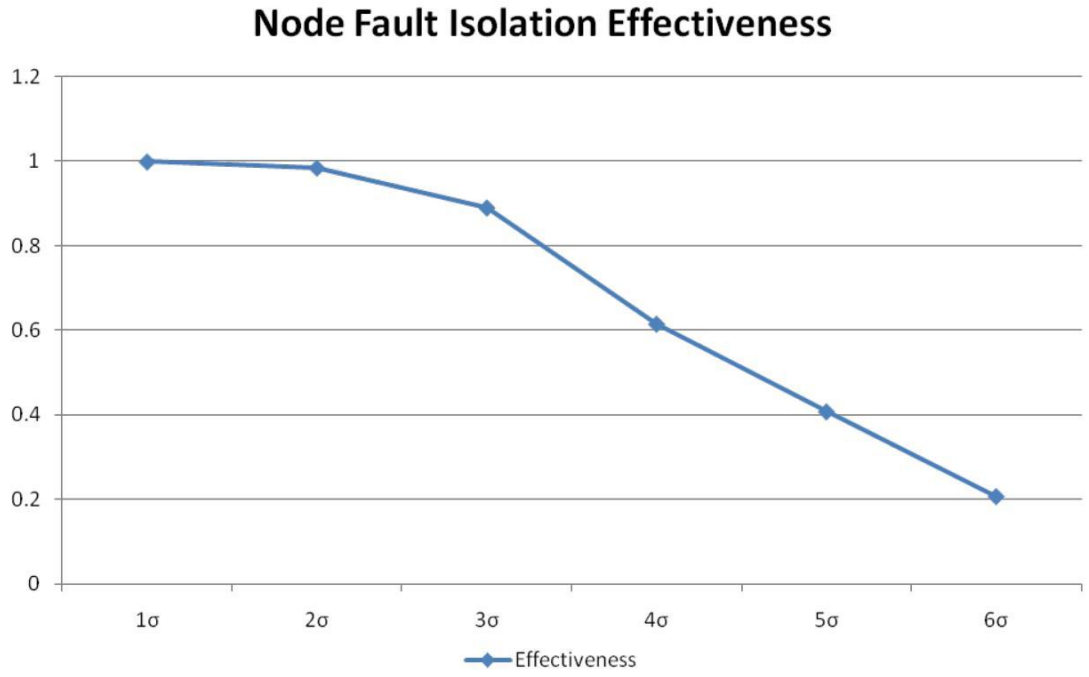


Figure 7.14: Fault isolation effectiveness for node failure.

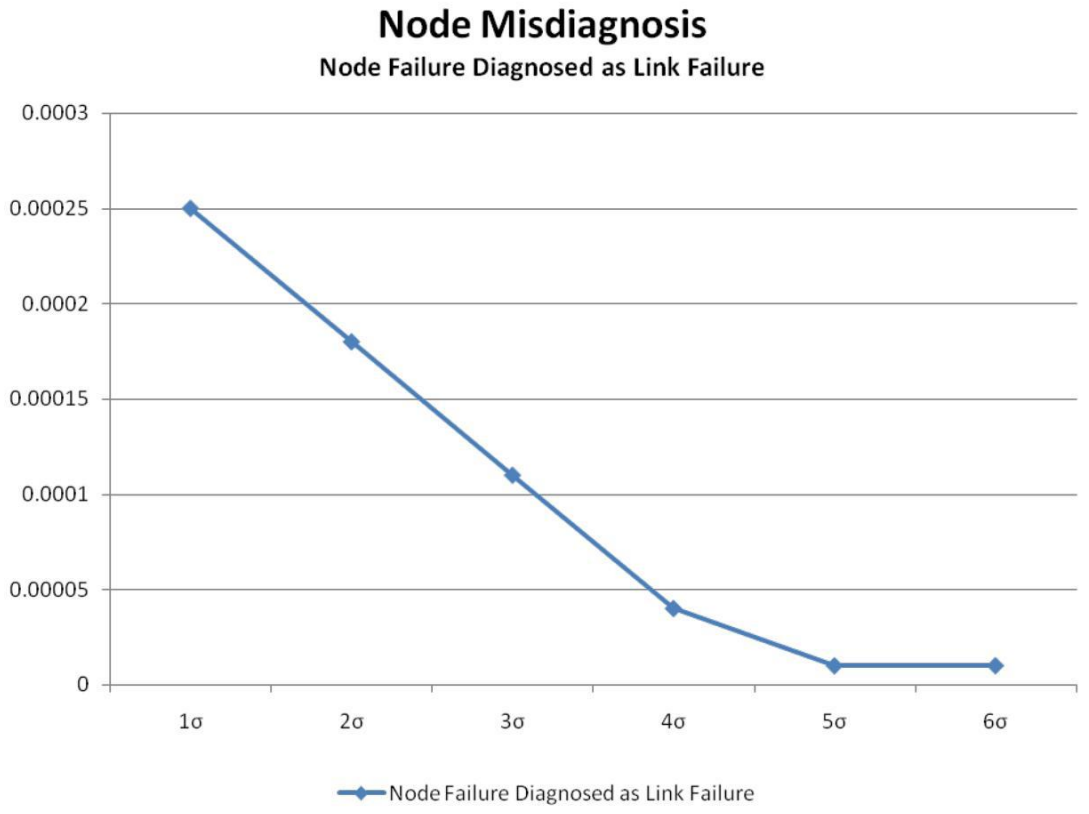


Figure 7.15: Misdiagnosis: node failure diagnosed as link failure.

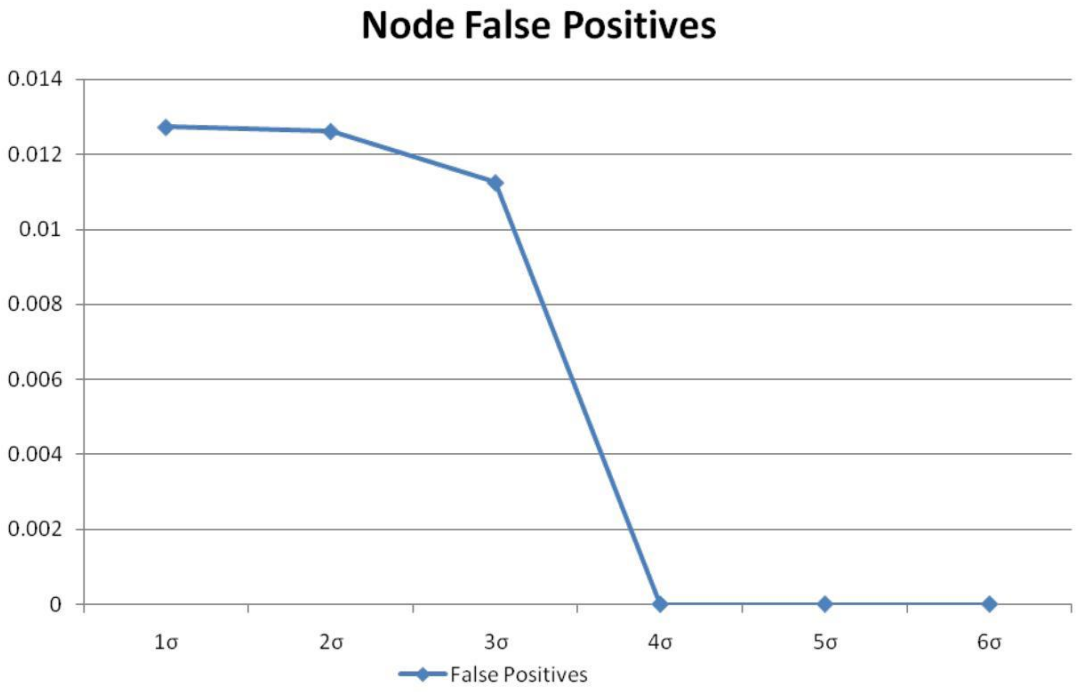


Figure 7.16: Node failure false positives.

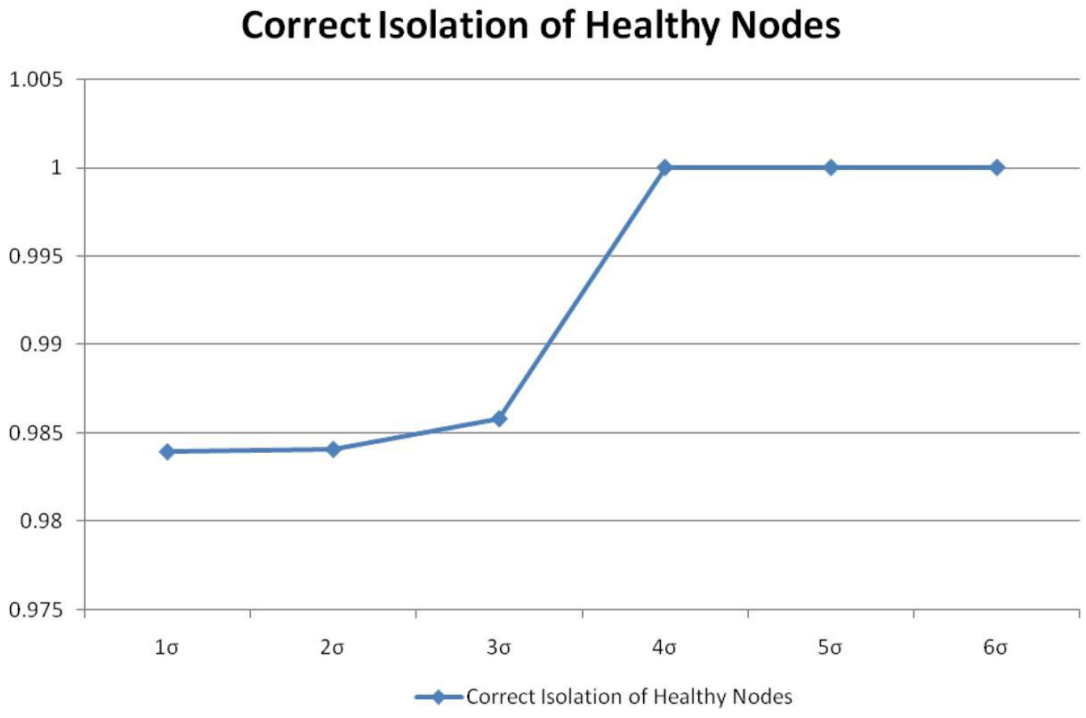


Figure 7.17: Correct isolation of healthy nodes.

network was approximately 85%, and the effectiveness of node failures was also high, reaching 98,4%. The use of 3σ also provides good results, although the efficiency on the isolation of link and node failures is reduced.

7.1.4 Failures Match Maker

The *Failures Match Maker* component combines known system patterns with additional node and link failures, in order to find a match for the evaluated diagnostic signal.

In this Section, we evaluated the performance of this component and verified the predictions made in Section 5.4.

Prediction 5: *"For the same confidence interval, the Failures Match Maker component presents a fault isolation effectiveness higher than the Membership Evaluator, if the threshold of the Transient Analyzer is kept constant."*

Simulation Setup

Test 7

The goal of this test is to evaluate the performance of the *Failures Match Maker* and to confirm the predictions made for this component.

In this evaluation test, we check the FIE and FP for combined failures. The network used for this simulation is the same network applied in *Test 1*. This network contains 100 nodes and 5 sinks, and has a link quality that varies from 0.5 to 1.

During this test, the simulator inserted a randomly generated combined failure every 50 seconds into the network. The combined failures were inserted after a period of normal operation. These failures consisted of 1 sink failure, 5% link failures with 0.5 interference, and 5 node crashes.

The test was performed using a range from 1σ to 6σ for the thresholds of the *Transient Analyzer* and *Membership Evaluator*. For this test, 250 pattern readings were used for each system state. For each threshold, 1000 readings were evaluated. In total 6 thousand readings were evaluated by the *Failures Match Maker* component.

The same events and failure sets were applied on all standard deviation configurations. These sets were also further used in the evaluation of the *Neural Classifier*.

Simulation Results

The analysis of the simulation results performed for the *Failures Match Maker* follows the same guidelines defined for the *Transient Analyzer*. As the link interference

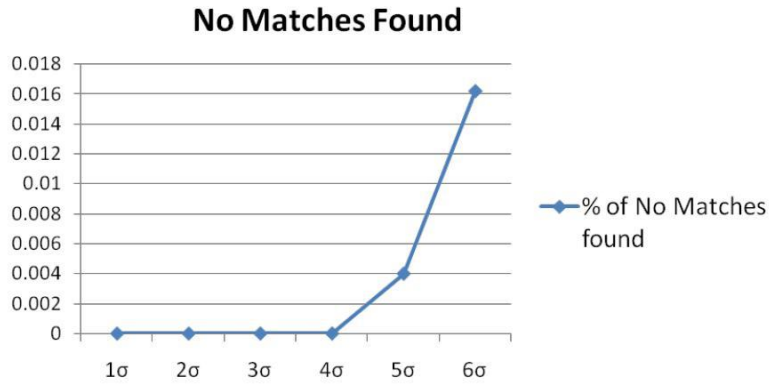


Figure 7.18: Percentage of total attempts where no matches were found by the Failures Match Maker.

and crash failures of one node may reduce the message throughput of several other nodes, the evaluation tests consisted of verifying if the diagnosed link and node failures were within the generated failures set that can be observed by the back-end (F_o).

Figure 7.18 depicts the number of times that the *Failures Match Maker* did not find any match for the diagnostic signal. A situation where no matches are found can occur if the *Membership Evaluator* rejects all patterns.

It is interesting to note from this graph that the *Membership Evaluator* rejects diagnostic signals as part of the known patterns, for thresholds higher than 4σ. This phenomenon occurs since the state analyzed also contains node and link failures. With large thresholds, the *Failures Match Maker* does not identify these failures anymore. Without the identification of the node and link failures, the diagnostic signal is not "recovered". The consequence is that the *Membership Evaluator* rejects the signals as part of the pattern.

Figure 7.19 depicts the sink isolation performance for both operating modes evaluated. The *Failures Match Maker* was capable of identifying the sink failures in 100% of the evaluated failures for the thresholds of 1σ to 4σ. This result confirms *Prediction 5*, as the sink failure FIE of the *Failures Match Maker* is higher than the one from the Membership Evaluator for the confidence interval between 1σ to 6σ.

Figure 7.20 presents the performance of the *Failures Match Maker* in the isolation of node crashes. The *Failures Match Maker* showed a high performance, reaching 99,4% of correctly identified node states. The number of node crashes misdiagnosed as a link failure is very low, and only occurred in 0.02% of the evaluated cases, having a threshold of 2σ.

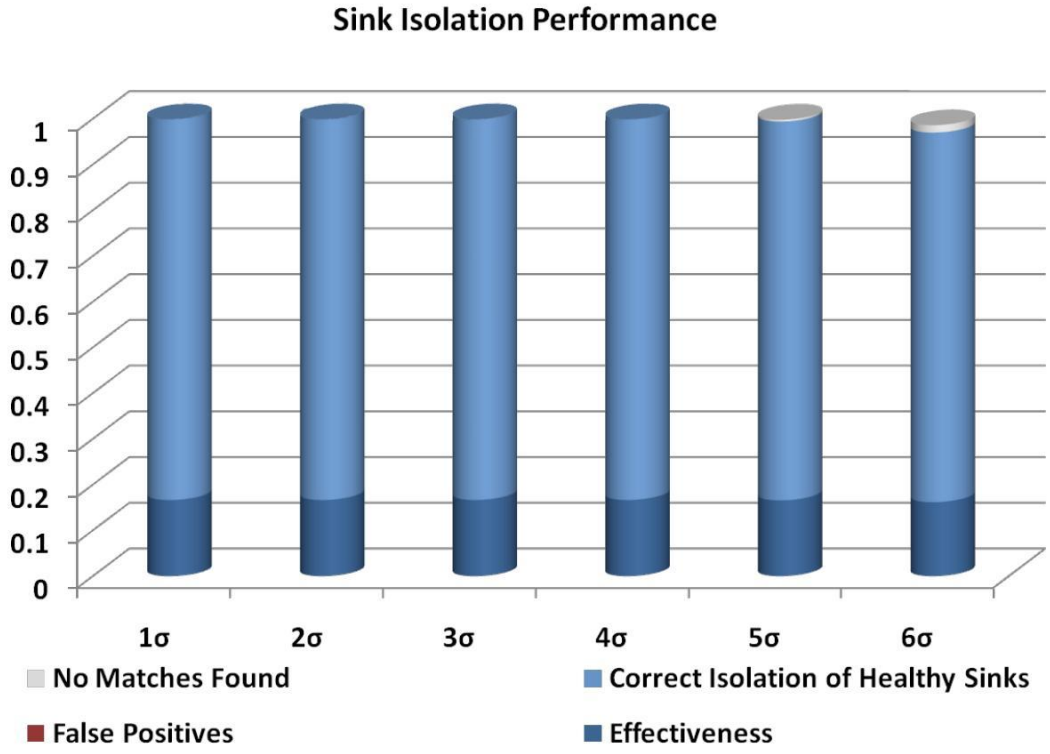


Figure 7.19: Failures Match Maker results for sink failure isolation performance. Results confirm Prediction 5.

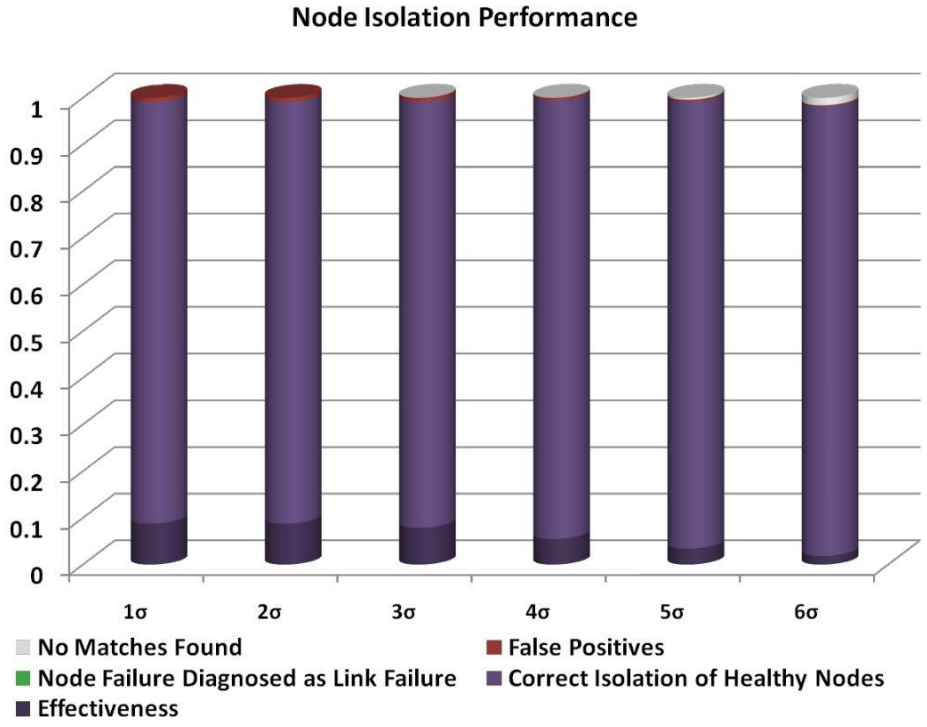


Figure 7.20: Failures Match Maker results for node failure isolation performance.

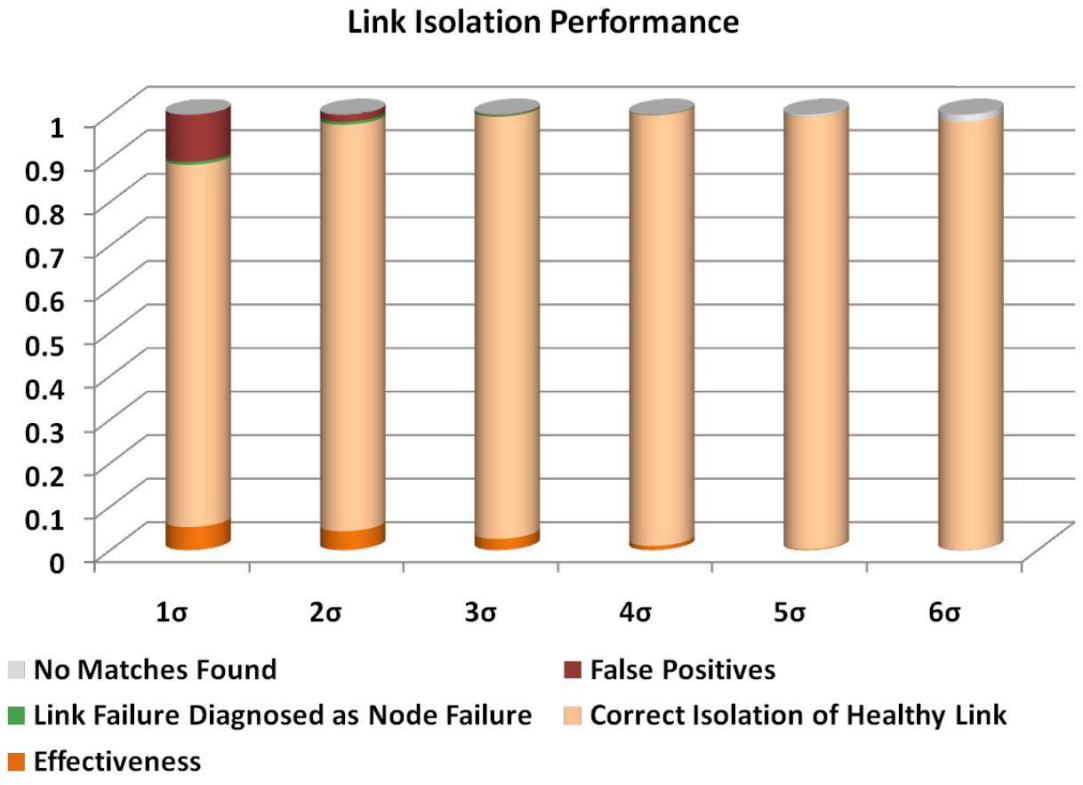


Figure 7.21: Failures Match Maker results for link failure isolation performance. There is a high number of false positives with 1std as the threshold.

Figure 7.21 depicts the link isolation performance for the *Failures Match Maker* component. As these results are related to the *Transient Analyzer* efficiency, similar behaviour is expected. The isolation of link failures resulted in 10,8% FP for 1σ threshold. This value drastically reduces to 1,6% and 0,16% for thresholds of 2σ and 3σ respectively.

In total, the *Failures Match Maker* correctly identified the states of the links in 97,6% of the cases with 2σ as the threshold, and 99,5% of the cases with a threshold of 3σ. Although 3σ has the highest overall performance, the FIE reduced considerably from 2σ to 3σ.

The results of this evaluation enables us to define guidelines for the selection of the parameter set for the *Failures Match Maker* component. Overall, thresholds of 2σ and 3σ, for the *Membership Evaluator* and *Transient Analyzer*, provide high efficiency on the isolation of failures. With 3σ, the effectiveness of isolating link failures considerably reduces. However, the number of link failure FP also reduces. Therefore, the selection between these thresholds depends on the requirements of the application scenario.

7.1.5 Neural classifier

The *Neural Classifier* is responsible for getting the multiple results generated by the *Failures Match Maker*, and is responsible for scoring each result. The result that receives the highest score is the one selected at the end of the fault isolation process.

There are two situations that can lead to multiple results generated by the *Failures Match Maker*. The first one occurs when patterns have overlaps, causing the *Membership Evaluator* to consider the diagnostic signal as part of more than one known system pattern. The second case occurs when failures happen in a way that the "recovered" signals are accepted as part of more than one pattern.

One example of the second case is the overlap with the normal operation pattern that results from sink failures. The "recovered" signal is accepted as part of the normal operation pattern, while the original signal is accepted as part of the sink failure pattern.

In this Section we use the second case for investigating behaviour of the *Neural Classifier*. This case is applied in *Test 8* performed to confirm *Prediction 6*:

Prediction 6: *"The Neural Classifier should have a high fault isolation effectiveness performance if the diagnostic signals are close to the ones used in the training set."*

Simulation Setup

Test 8

In order to verify *Prediction 6*, we performed a test to force the use of the *Neural Classifier* in the fault isolation process. In this test, we inserted sink failures and additional node and link failures in the simulated network. This procedure created an overlap between the patterns of sink failures and the system in normal operation. This overlapping result required the use of the *Neural Classifier* to select one of the results generated by the *Failures Match Maker*. Since the normal operation pattern is further away from the diagnostic signal, the neural classification should provide a higher score to the sink failure.

In this test we used the event and failure set generated in *Test 7*. This enables us to compare the results with previous evaluations and provide a full set of tests for one network that was randomly generated. For this test, 250 pattern readings were used for each system state. For each threshold, 1000 readings were evaluated. In total 6.000 readings were evaluated by the *Failures Match Maker* component and forwarded to the Neural Classifier if more than one result was generated.

Simulation Results

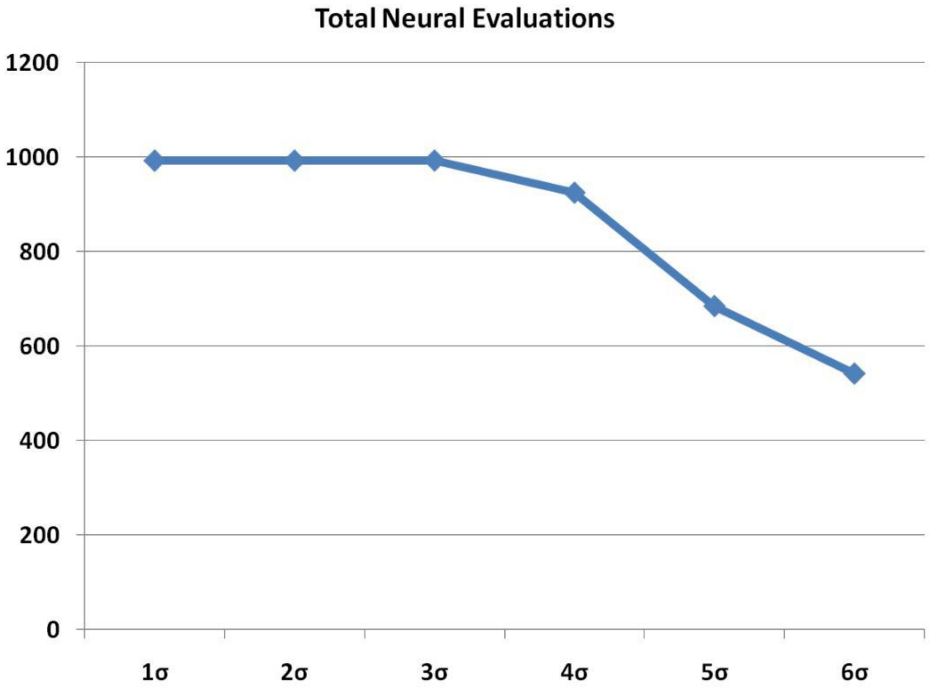


Figure 7.22: Total number of neural classifications.

Figure 7.22 depicts the number of neural classifications performed for each threshold configuration. As demonstrated in this graph, the number of evaluations reduced as the threshold increased, although the sink isolation effectiveness did not suffer major modifications.

This effect occurred since link and node failures stopped being recognized as the threshold increased. As a consequence, the diagnostic signal is not "recovered" and is refused by the *Membership Evaluator*. As the pattern of the system in normal operation requires the "recovery" of more nodes and links, this pattern stopped being accepted by the *Membership Evaluator* as the threshold increased. Without the pattern of the system in normal operation, the *Failures Match Maker* returned only one result (the sink failure), therefore not requiring a neural classification.

Figure 7.23 presents the results of the neural classification. The percentages presented represent the number of occurrences of the evaluated variable (e.g correct classification) divided by the number of neural classifications.

Overall, no major differences occurred to the classification when the threshold was modified. The results show that a slightly better result was generated when the threshold reached 5σ, when the system reached 100% of correct classification. The worse result produced 97.3% of correct classification. This improvement is a consequence of the elimination of the normal operation pattern from the results of the *Failure Match Maker*.

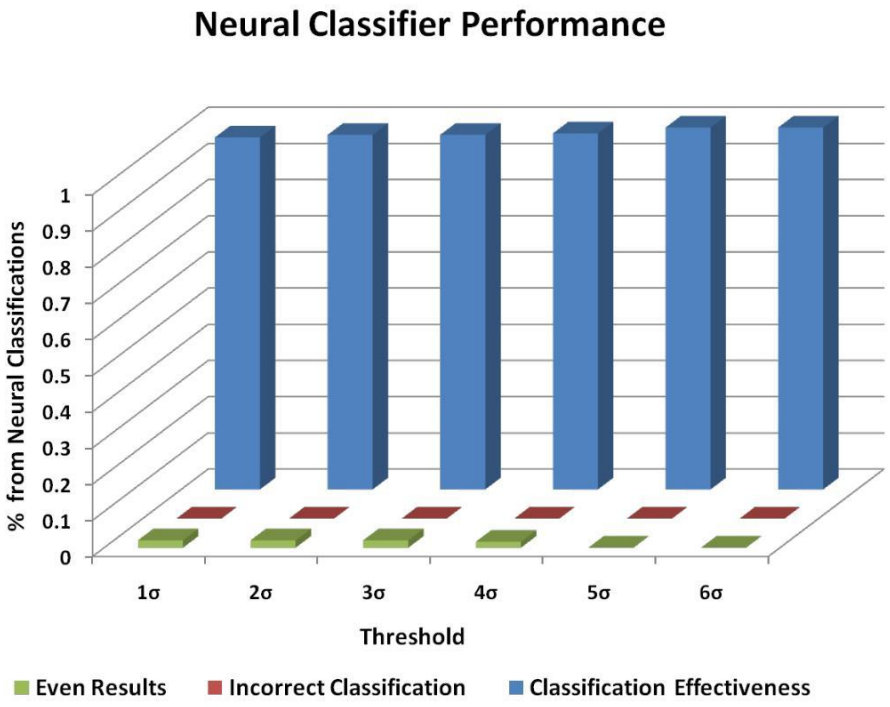


Figure 7.23: Neural Classification results.

These results confirm *Prediction 6* since the majority of the readings are correctly classified. This shows that the performance of the *Neural Classifier* is slightly affected by the performance of the *Failures Match Maker*. If the *Failures Match Maker* is operating with parameters that enable it to diagnose most failures and not generate too many FP, the results from the neural classification will also show good performance.

7.2 Application trial results

In this Section, we present the results of applying the proposed approach on data generated during the application trial described in Section 6.1. The expected outcome of this experiment is to prove that the proposed approach works in theory and in practice.

As described in Section 6.1, we acquired two sets of data: one for learning the patterns and one for cross validation. The states of the acquired patterns consisted of normal operation and each isolated sink failure.

With these sets of data, we evaluated the following components: *Membership Evaluator*, *Failures Match Maker*, and *Neural Classifier*.

7.2.1 Membership Evaluator

The first test consisted of verifying the fault isolation effectiveness of the membership function with different configurations of thresholds, timeframes (TF), and timesteps (TS). The timestep is a mechanism used to raise the number of pattern readings that can be acquired at a given time interval. It is a sliding window that determines how long the system waits to acquire each pattern reading.

For the evaluations performed, we selected timeframes equivalent to 3, 4, and 5 heart beats (135, 180, and 225 seconds). The timestep configurations chosen were the same as the timeframe (3, 4, and 5 heart beats), and also 1 heart beat.

The results of this test can be visualized in Figure 7.24. As depicted in this graph, the effectiveness of the membership function is similar to the results observed in the simulations. As the number of samples is lower than the ones used in the simulation, errors originated from sampling appear (also observed in simulation for 50 pattern readings). This results in an efficiency lower than the threshold defined.

Although the efficiency is lower than the ideal due to the low number of samples, the fault isolation effectiveness of the membership function reached a value of 95,1% with a threshold=5σ, TF=180s, and TS=180s. With 6σ, the same configuration reached 100% of fault isolation effectiveness.

The drawback of using large thresholds is that the membership function starts to accept the diagnostic signal as part of patterns that it does not belong to. In some situations, the diagnostic signal is accepted as part of a pattern that it does not belong to and it is not accepted as part of its own pattern. We consider this case as a FP and the results for such case is presented in Figure 7.25.

This graph shows that in the majority of cases, the number of FP increases as the threshold increases. In some situations, however, the number of FP decreases after reaching a "peak". This occurs since the diagnostic signal starts to be accepted by its own pattern.

For this test, the highest percentage of FP observed was 0,5%. It is also interesting to note that no FP occurred while the threshold was less than 2σ.

Stretching the threshold to accept most diagnostic signals as part of the pattern can also result in overlaps in the membership evaluation. In such case, the membership function accepts the diagnostic signal as part of a pattern that it does not belong to, but accepts the signals as part of the pattern that it in fact belongs. Figure 7.26 presents the results for this case.

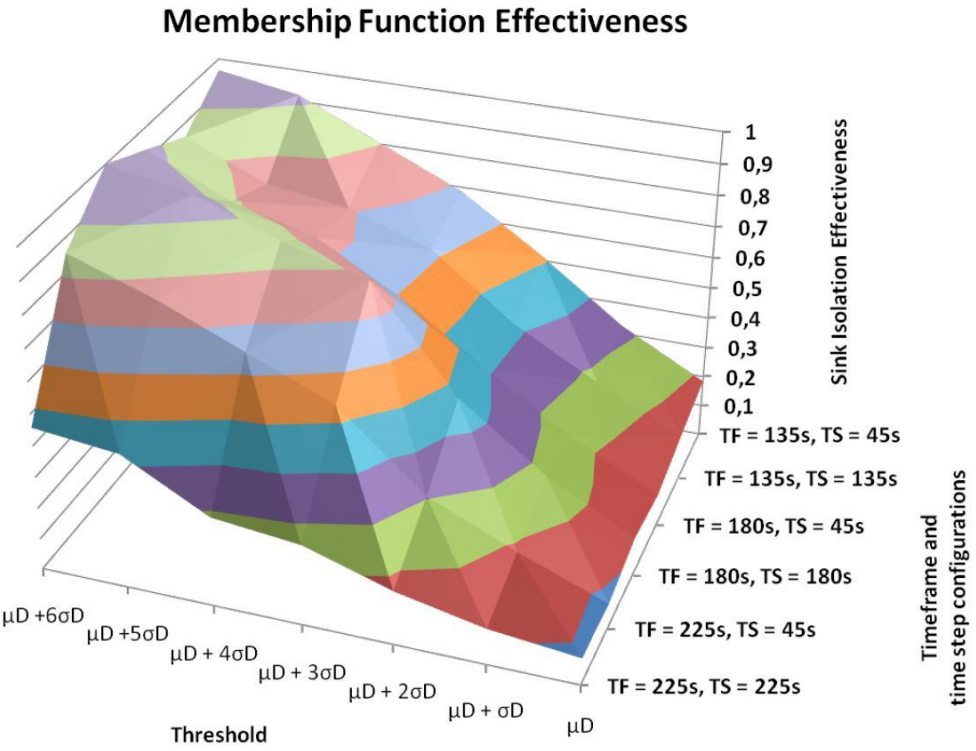


Figure 7.24: Percentage of sink failures correctly identified by the Membership Evaluator.

Through the analysis of the overlapping results, we observed that the majority of occurrences happen between sink failures and the system in normal operation.

Once again, overlaps did not occur for a threshold up to 2σ. The highest percentage of overlaps observed was 2,9%. This occurred for the configuration where the threshold was set to 6σ, TF=135s, and TS=45s.

The cases where overlaps exist are addressed by the *Neural Classifier* component, which is further evaluated in this Chapter.

7.2.2 Failures Match Maker

The test performed to evaluate the *Failures Match Maker* consisted of identifying isolated sink failures for different configurations of thresholds, timeframes, and timesteps.

The fault isolation effectiveness results, depicted in Figure 7.27, confirm *Prediction 5*. The minimum observed isolation effectiveness was 88%, which is higher than the one observed with the membership function. We can also notice a reduction in the isolation effectiveness for the configuration with 5 heart beats as the timeframe.

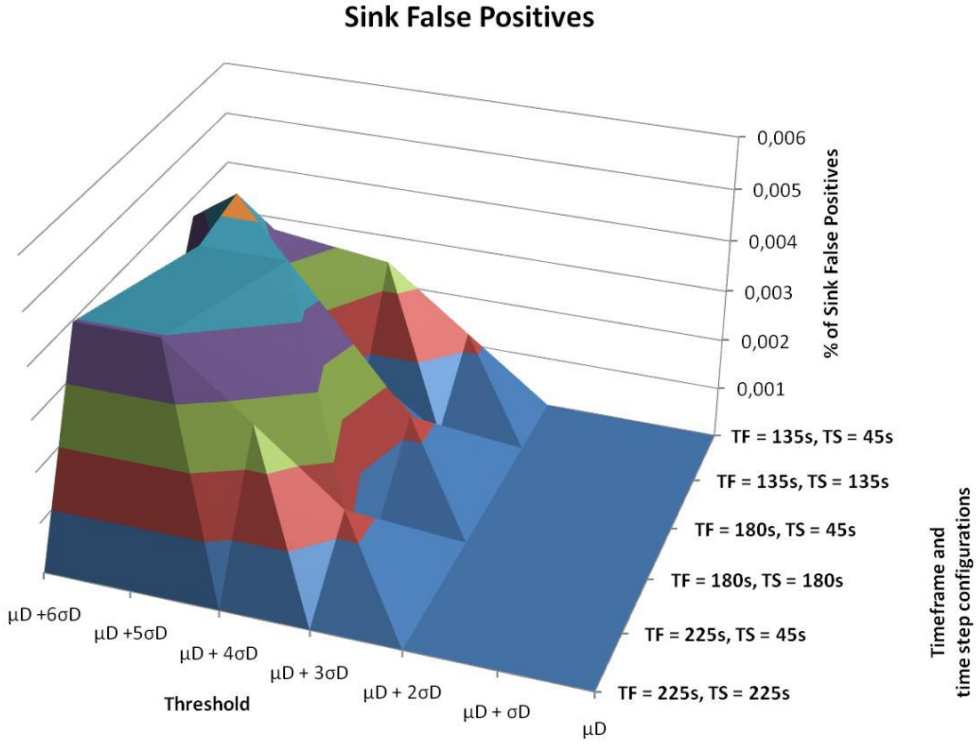


Figure 7.25: Percentage of sink failures false positives accused by the Membership Evaluator. The number of FP decreases after reaching a "peak", because the diagnostic signal starts to be accepted by its own pattern.

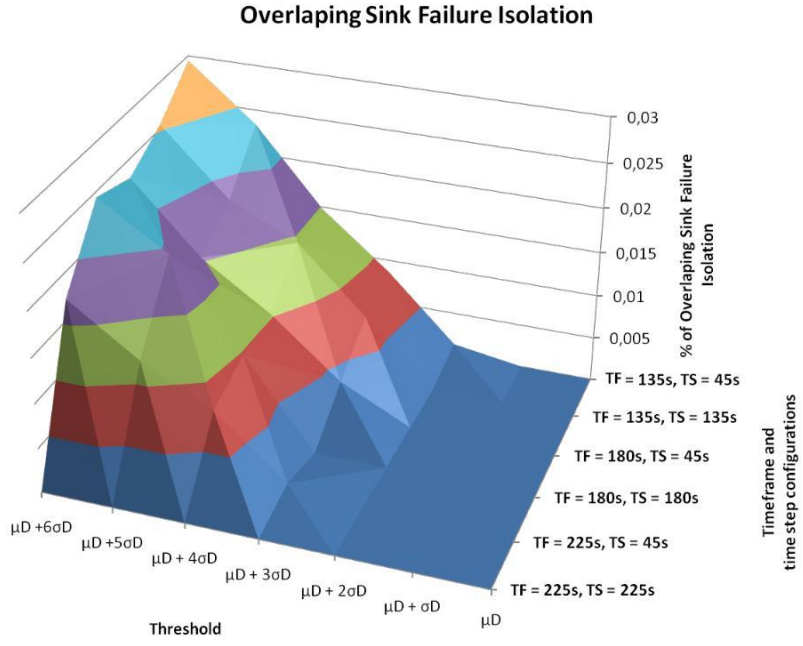


Figure 7.26: Membership Evaluator results for overlapping sink failures.

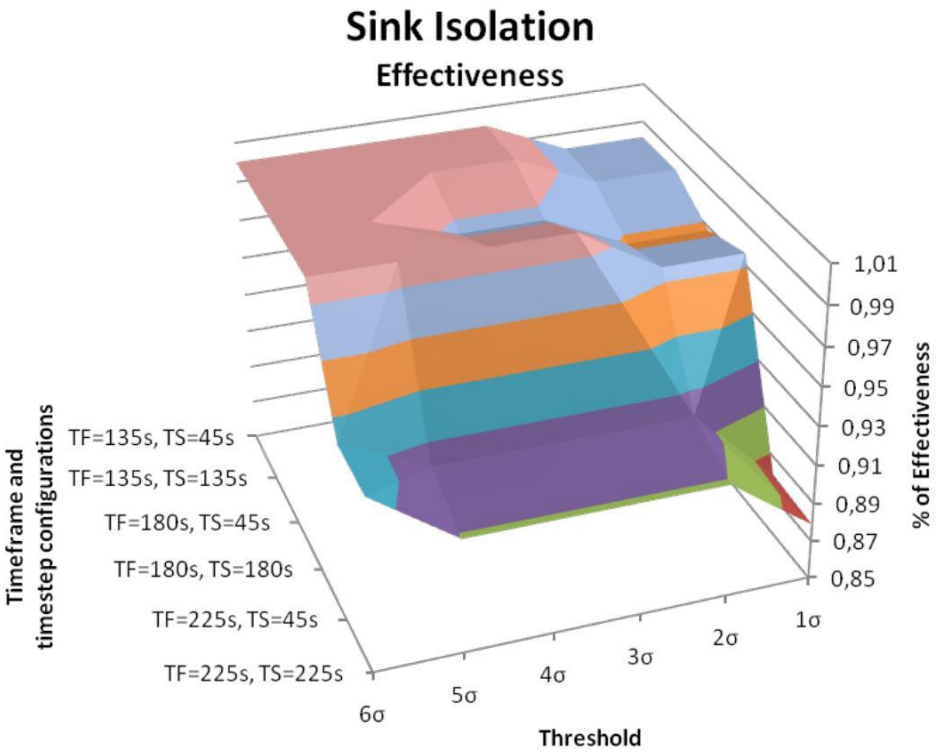


Figure 7.27: Failures Match Maker results for sink isolation effectiveness. The results of this test confirm Prediction 5.

This reduction is a probable consequence of the reduced number of pattern readings that these configurations have when compared to 3 or 4 heart beats.

As no additional link or node failures were present, it is important to analyze the number of FP indicated by the *Failures Match Maker* for these components.

Figure 7.28 depicts the number of link failure FP indicated by the *Failures Match Maker*. The result from this evaluation confirms the results found through simulation. With low thresholds, the number of link failure FP considerably increased. With 2σ the number of FP resulting from this test ranges from 3-4%. These values are close from the theorethical values calculated for these configurations (2.3%) and shows that the proposed approach works in theory and practice.

Figure 7.29 presents the node crash FP results generated by the *Failures Match Maker*. It is interesting to note that the number of FP decreases as the threshold increases until it reaches 5σ, which follows the same behaviour as the simulation results. After 5σ, this number increases again. This phenomenon occurred since at this point, the *Failures Match Maker* also starts to accept the diagnostic signal as part of incorrect sink failure patterns. The consequence is that the *Failures Match Maker* identifies node crash FP with the incorrect sink failure.

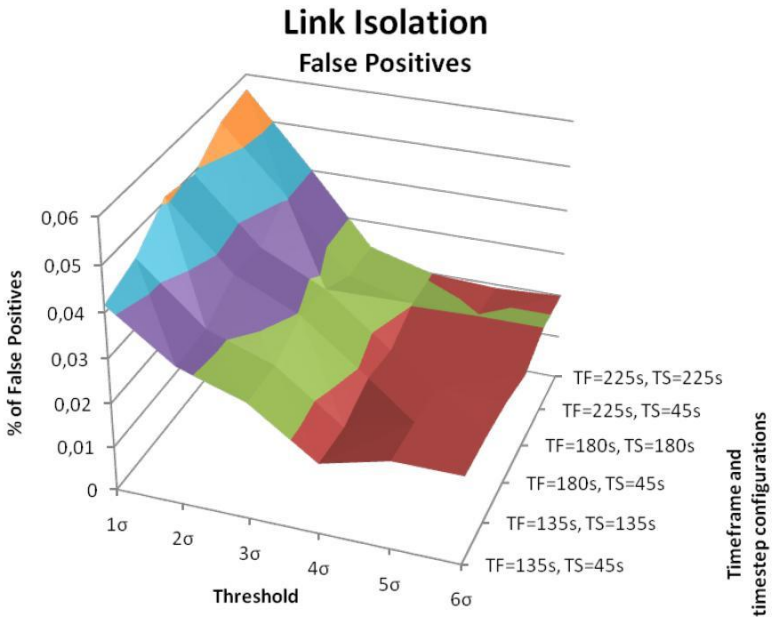


Figure 7.28: Failures Match Maker results for link failure false positives. There is a reduction on the number of false positives as the threshold value increases.

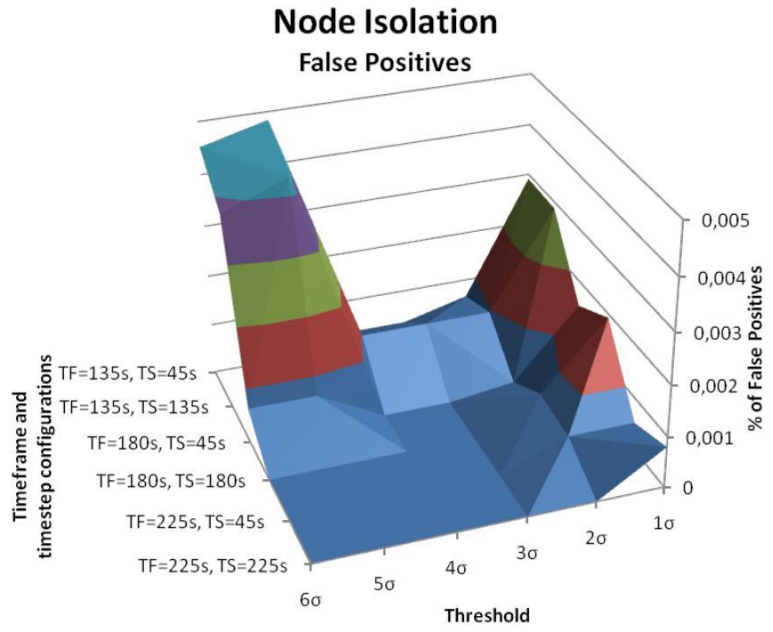


Figure 7.29: Failures Match Maker results for node failure false positives. After 5std the number of FP increases because the Failures Match Maker also starts to accept the diagnostic signal as part of incorrect sink failure patterns.

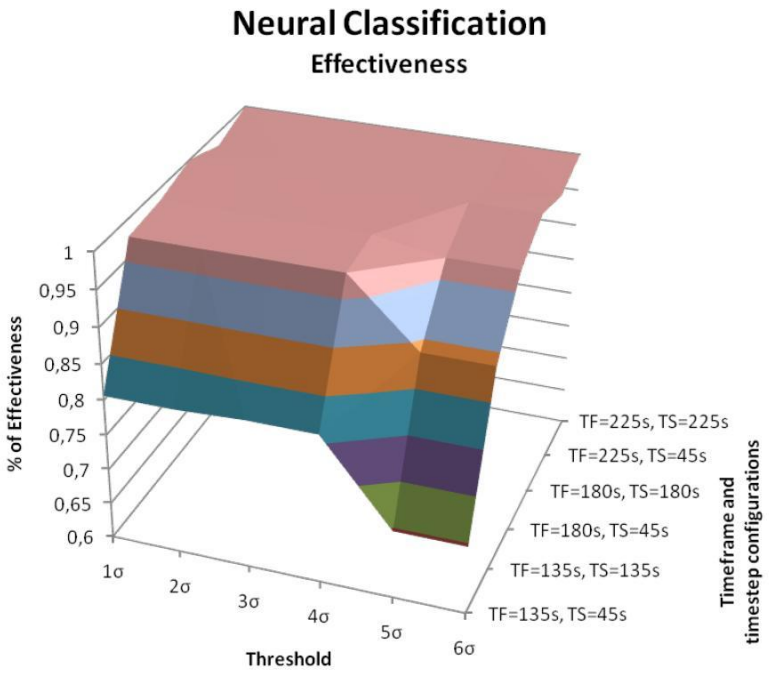


Figure 7.30: Neural Classifier effectiveness. Lower performance of the configuration TF=135s and TS=45s is due to stronger a overlap of a sink failure and the system in normal operation.

7.2.3 Neural Classifier

The *Neural Classifier* evaluation consisted of differentiating overlapping results generated by the *Failures Match Maker*. The majority of the overlapping states were caused by the normal operation state and the sink failure states, although sink failure overlaps also occurred.

In Figure 7.30, the classification effectiveness of the *Neural Classifier* is presented. Overall the performance of the *Neural Classifier* was very high, reaching 100% of correct classification. Nevertheless, these results show a clear reduction in the performance of the *Neural Classifier* as the threshold reaches 4σ. This occurs since changes in the expected message throughput of each node are not recognized, leading to the evaluation of diagnostic signals that may deviate from the patterns used to train the neural network. This graph also shows reduced performance for the configuration TF=135s and TS=45s. This indicates that the sliding window approach to increase the number of readings does not necessarily improves the performance of the system.

Figure 7.31 depicts the number of incorrect classifications performed by the *Neural Classifier*. This graph partially complements the graph in Figure 7.30, as it indicates the source of lower performance of the TF=135s and TS=45s configuration.

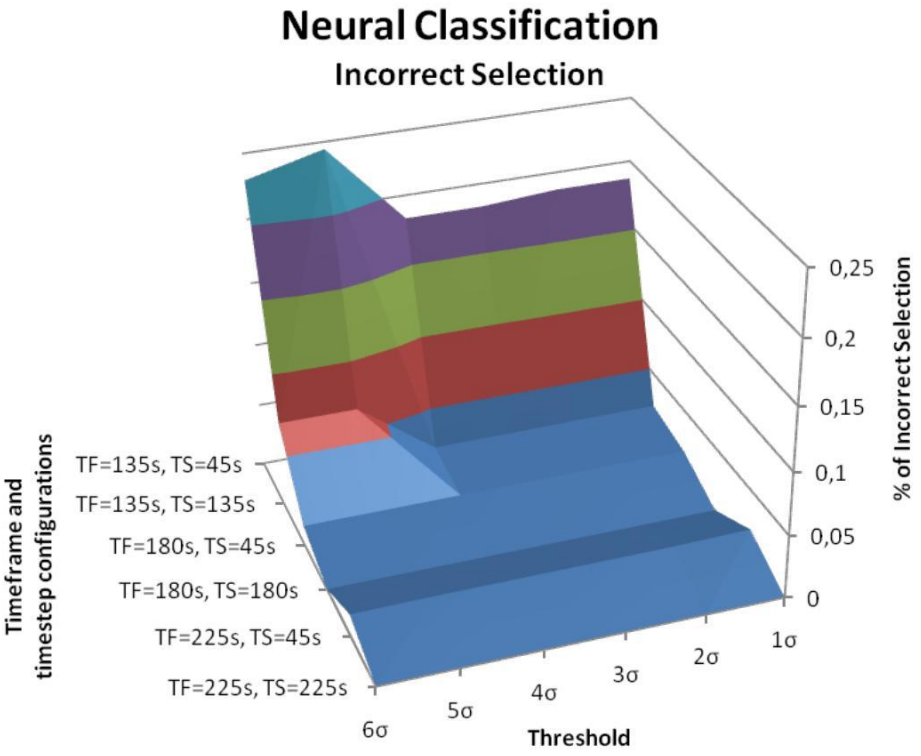


Figure 7.31: Incorrect classification of sink failures by the Neural Classifier.

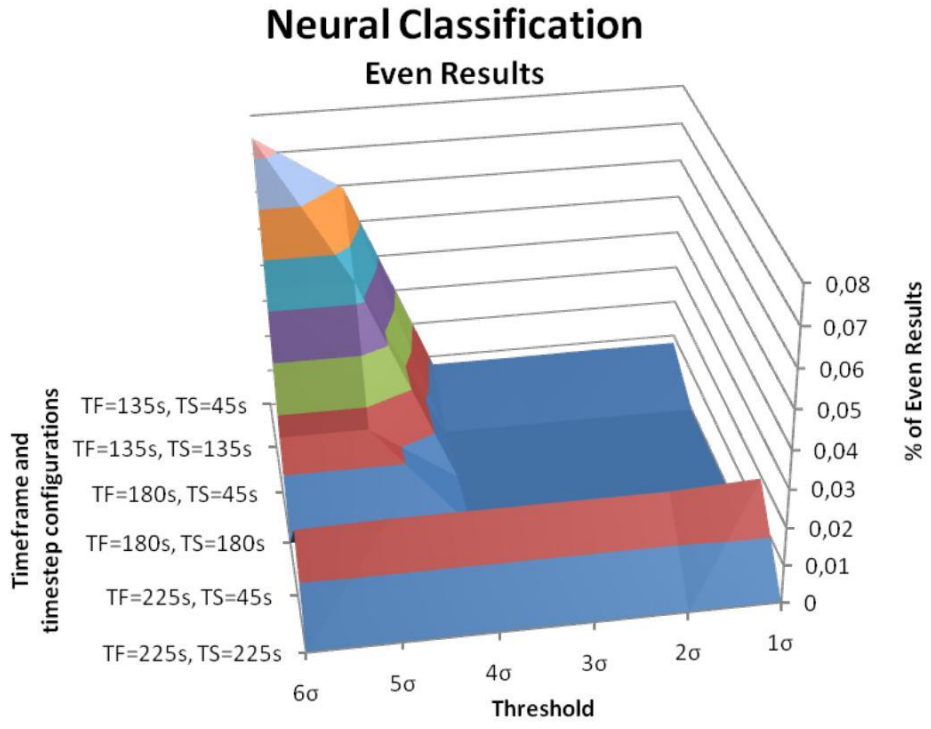


Figure 7.32: Even classification of sink failures by the Neural Classifier. The diagnostic signals are not "recovered" for large thresholds, and get further away from the values used to train the network.

By investigating, we discovered that for one specific sink failure, this configuration resulted in statistical properties for one sink failure that was very close to the normal operation pattern. Longer timeframe and timestep configurations help avoid such situations and improve the efficiency of the system.

Figure 7.32 shows that as the threshold increases, the number of cases where the neural network cannot differentiate between the patterns also increases. This also confirms *Prediction 6*, as the diagnostic signals are not "recovered" with large thresholds, and get further away from the values used to train the network. In this graph, the number of even results reached 7,4% for the configuration threshold = 6σ, TF=135s, and TS=45s.

7.3 Evaluation Results Discussion

The results presented in this Chapter confirmed all predictions made in Section 5.4. The results indicated a high efficiency of the proposed approach in terms of fault isolation effectiveness for the data generated through simulation and the data acquired during the application trial.

The evaluation also shows the limitations of the approach when the configurations adopted reach the borders of the assumption made about the system. For example, outlier readings appear when the timeframe selected is not long enough to allow for an appropriate approximation of the message throughput of the sensor nodes by a normal distribution (discussed in Section 7.1.1.2).

Through analyzing results, we can derive recommendations for the selection of parameters used in this framework. The timeframe selected should be the longest period that the application can wait for the result. The longer the timeframe, the better the distribution approximation of the sensor node message throughput. This results in a better performance of the membership function and a lower number of FP. Nevertheless, some applications require a system that is highly responsive and cannot afford long timeframes. One approach is to reduce the heart beat interval of the sensor nodes, which also improves the distribution approximation.

The threshold selection also depends on the requirements of the application. If minor alterations in the link quality need to be identified, a threshold of 1σ for the *Transient Analyzer* can be selected. This however, also results in a high number of FP. If only alterations that cause a reduction in the link quality around 0.5 need to be identified, then a threshold of 2σ or 3σ provide a better isolation effectiveness versus FP trade off.

Finally, the threshold defined for the *Membership Evaluator* should be lower than 3σ. Since the *Failures Match Maker* "recovers" the message throughput of sensor nodes that considerably deviate from the average, small thresholds for this component result in high overall performance.

8. Conclusions

The deployment of WSNs in enterprise scenarios will enable in the near future a new paradigm of business process management. Nevertheless, the transparent integration of heterogeneous WSNs with business applications represents the first challenge faced for the adoption of this technology.

Due to the multitude of low level protocols used by heterogeneous WSNs, their integration with back-end applications becomes cumbersome. Additionally, the effects that failures occurring on these networks can cause, and the harsh environment in which these networks are deployed, will become a major concern for the adoption of WSNs.

In this scope, this work presented the efforts performed to propose a framework for a transparent integration of WSNs with business applications focusing on fault diagnosis and recovery for heterogeneous networks.

The proposed framework embraces standards, such as SOA based on Web Services, which are the current trend for enterprise applications. Through its layered architecture, this framework enables a seamless integration of heterogeneous WSN platforms.

The diagnosis of failures in a heterogeneous environment presents additional challenges, since it cannot be assumed that networks provide support to specific functionalities and may only offer limited information about the nodes. In this thesis we approached this problem by using only a restricted information set to perform diagnosis: sensor node identification number, timestamp information, and the heart beat interval.

Since pattern recognition methods can learn the dynamics of the system based on example, these methods are the first class choice for the diagnosis of failures in WSN using only limited information from nodes. Some of the adopted techniques are widely used in fault diagnosis for industrial applications, and have not been applied in the WSN domain due to the scale of such networks and the combinatorial explosion of system patterns.

As major contribution to science, this work brings together the research on fault diagnosis for industrial applications and the research domain of WSNs. Through careful selection and combination of techniques, this thesis proposes a fault diagnosis approach for WSNs that uses techniques developed for the industrial domain, avoiding the drawbacks that their adoption may cause. This approach is based on pattern recognition techniques, such as statistical analysis and neural networks, and allows the isolation of failures represented in the crash failure analysis by the boxes 1 to 13 in Figure 5.1.

The work performed in this thesis also opens a new branch of fault diagnosis research for WSNs by considering a hardware platform abstraction based on a restricted information set. Although the restrictions imposed could lead to low performance of the algorithm, our tests showed that the use of pattern recognition techniques can yield good results. For instance, the identification of sink failures applying the membership function of the proposed fault diagnosis approach resulted in 99,3% of fault isolation effectiveness and zero false positives in a network with 50% overlap.

During the course of this thesis we also made interesting discoveries. For instance, with careful investigation, we proved that the *Normalized Euclidean Distance* method and the calculation of the probability of a pattern occurring defines the exact same node message throughput subspace. This discovery enabled us to avoid calculations with exponential values, which is costly and can insert more errors in the system due to floating point manipulations.

Another contribution made was the analysis of crash failures based on observed timeout failures, indicating their source and conditions. This analysis can be applied for rule based fault isolation in networks were the topology information is available.

The contributions made by this work focused on achieving the goals set in Chapter 1 and proving the thesis defined in the same Chapter. The first goal of this thesis was: *"To enable the integration of heterogeneous WSNs with enterprise applications"*. The proposed framework together with the fault isolation approach, enable the integration of a variety of hardware platforms with back-end applications. Together these proposed solutions provide transparent access to heterogeneous hardware platforms

and support the diagnosis of failures and the integration of fault recovery techniques. This achieves the first goal of this thesis.

The second goal of this thesis was: *"To identify failures occurring in heterogeneous WSNs applied in business processes"*. The method proposed for fault isolation was designed in order to use a restricted information set that can be provided by most hardware platforms. The developed approach proved to be efficient in the isolation of failures and therefore fulfills the second goal of this thesis.

Finally, the high effectiveness of the proposed fault isolation algorithm proved the thesis defined in this work: *"The isolation of failed components in heterogeneous WSNs platforms applied in business processes can be achieved based on a restricted information set: sensor node identification number, timestamp information, and heart beat interval"*.

As SAP is a leading company on innovative solutions for business processes management, one of their goals is to integrate WSNs with business applications. Hence, the results from this thesis provide the grounds for this integration and for the diagnosis of WSN failures in a platform independent manner.

Currently the approach we proposed in this thesis is being analyzed with the goal of adopting its pattern recognition techniques in two projects. The first project is financed by the European Union and focuses on the industrial automation domain. It faces a challenge in the recognition of machine states due to the restricted information available and complex system states. The second project is a proposal that aims at using the results from a fault diagnosis system as input to a decision support system. This system then re-maps the business processes to cope with system failures.

8.1 Future Work

This work has presented significant results in the isolation of failures in static wireless sensor networks. Nevertheless, improvements can still be made in order to increase the efficiency of the proposed method and tackle business process demands.

One approach is to insert an input filter in the system. This filter can average the message throughput from sensor nodes acquired in a number of diagnostic signals. This procedure would help reduce the number of readings that lie beyond the defined threshold and that in fact belong to the system pattern. The use of input filtering also has the potential of reducing the number of pattern readings that need to be acquired in order to calculate the statistic properties of the system. As the filtered

values are shifted towards the average value, the membership function will have a better resilience to imprecise calculated statistic properties.

As next step towards enabling the adoption of WSNs in business processes, a solution for fault diagnosis that tackles dynamic networks should be investigated. One approach is to create maps of the link quality in the regions where sensor nodes are deployed. This would be a solution similar to RADAR [7], where this concept is used to track sensor nodes. In such approach, the fault isolation method would gather maps of the system in different states with the expected link quality for different locations. During operation time, image recognition methods can be applied to find a match between a map containing the current link qualities of the locations and the known states of the system.

The evaluation of the threshold definition through the Gaussian approximation method showed that outlier readings may occur in networks where the message throughput of sensor nodes can only be poorly approximated by a normal distribution. One approach to overcome this phenomenon is to use a different threshold definition method. For instance, the threshold can be defined based on the maximum observed distance during pattern acquisition time. This method can also improve the effectiveness of the membership function by allowing values that are further away from the mean distance to be considered as part of the pattern. The major drawback caused by this threshold definition method is the high risk of false positives.

Another approach to handle outlier readings is the use of output filters. These filters can verify the results of a number of fault isolation executions and average them. This method can also help reduce the number of false positives of link failure.

Although the processing of the fault diagnosis is performed on the back-end and therefore the strong computational and power restrictions do not apply, an approach for handling large scale networks should be investigated. As these networks are used in business processes, one approach is to separate them into sectors. This breaks the problem down and enables the localized diagnosis of the system, which can be processed with little resources.

Finally, an insight of the business potential of the approach proposed in this thesis would facilitate its adoption by the industry. This insight can be achieved through a business oriented analysis on the cost reduction associated with the adoption of WSNs and the proposed solution in a real enterprise application.

A. Message Exchange

One of the major tasks of the message exchange functionality is to help overcome the technical heterogeneity of different sensor networks. Sensor nodes have severely restricted computational resources, such as CPU, RAM, and ROM. Hence, networking technologies most appropriate for these types of devices are optimized for device-to-device communication. These technologies do not offer the bandwidth required for common networking (such as via TCP/IP), and application level communication (such as standard Remote Procedure Calls (RPC)). Due to these limitations, the seamless connection of sensor nodes functionality to business applications becomes a hard task for application developers who usually are not familiar with the proprietary protocols used by the platforms.

The message exchange functionality adopted for this thesis were developed during the CoBIs project [26] and were further integrated with the FT-WiseNets framework. The message exchange functionality provided by the CoBIs components, bypasses both resource and connectivity issues by building the 'technology bridge' [147].

This bridge provides technologies for business applications access to sensor node data in a commonly known and thus acceptable manner. Accordingly, it provides service endpoints and corresponding interfaces, based for instance on RPC. This bridge can be used by application developers for either direct or indirect connection.

The message exchange mechanisms also ensure that a proper platform-specific wrapping is taken place behind the service endpoints. This enables the translation between messages directed to the back-end application and the message generated by the physical nodes.

This wrapping approach allows sensor nodes to expose their functionality via standardized RPC-based mechanisms, such as web services. In addition to the provision of proper interfaces and the corresponding service endpoints, the message exchange functionality has to ensure that these endpoints are properly linked to the WSN.

The first step towards defining the elements required to support the message exchange mechanism provided by the framework, is the analysis of the message exchange patterns that are used in an industrial environment. We have identified three main communication patterns which reflect the requirements of the industry for applying WSNs in business processes [146]:

- **One-way:** The WSN/Node receives a message. This communication paradigm is applied when no response is expected from the endpoint side. One example of such communication is presented in section 4.1. During re-configuration of storage regulations, the back-end propagates the new information and expects the nodes to start behaving accordingly.
- **Notification:** The WSN/Node sends a message. WSNs are commonly applied in scenarios where their main functionality is to report sensor readings either continuously or due to some change in the environment [1, 109]. In the scenario presented in section 4.1, this communication paradigm is applied when alert events are propagated to the back-end system.
- **Request-response:** The WSN/Node receives a message, and sends a correlated response message. The input and output elements of a message specify the abstract format for the request and response, respectively. This is a paradigm frequently used in enterprise applications where data from specific sensor nodes are required. For example, in the scenario presented in section 4.1, once an alert has been triggered it is important to get information on the specific drums involved in the situation.

To enable the exchange of messages between the backend and the WSN the FT-WiseNets framework proposes the use of three components: *Message Handler*, *Notification Broker* and *Request Processor*.

A.1 Message Handler

Message handlers are platform-specific components that mediate between protocols used by different hardware platforms and the ones used in the back-end (where the

platform-independent part of the framework is being executed). As the protocols used by various hardware platforms are usually different, a distinct implementation of this component is required for each platform. Therefore multiple message handlers can be installed in parallel to support different hardware platforms at once.

Message handlers can be realized in two different ways. A message handler can run integrated into a Platform Gateway (which is usually a separate piece of hardware, often including a fully-fledged computing platform able to run arbitrary programs), or it can be implemented as part of the framework. In both cases, the nodes only exchange messages with their platform gateway in a proprietary format.

If the message handler is running on the gateway, it can immediately pick the messages and convert them into a format compatible with the back-end. In the second case, the gateway transforms the messages in a format that can be understood by the message handler (the gateway is then effectively acting as a bridge, being agnostic to the content of the messages). In a second step, the message handler converts the messages into the back-end format.

It is preferred to completely separate the implementation of the message handlers from the gateway. The advantage is that the message handler software can be easily changed, since it is located in the back-end and access to a separate hardware component (the gateway) is not required.

A.2 Notification Manager

The *Notification Manager* is responsible for handling the distribution of all events generated in WSNs. Interested consumer applications can subscribe with this component to receive all relevant events.

As the number of events generated by WSNs is usually very high, this component provides filtering mechanisms where message types and the source of the event can be specified to reduce the amount of data that the applications have to process.

The *Notification Manager* provides a web service interface for the distribution of event messages. Notifications are distributed based on the WS-Brokered Notification [126]. This specification was chosen since it is based on web services, which is widely accepted by the industry, and it provides additional mechanisms (such as the *Notification Broker*) when compared with WS-BaseNotification [125] and WS-Eventing [15].

In FT-WiseNets, the *Notification Manager* supports two event distribution paradigms: *push* and *pull*. To enable event consumers to receive events through *pull* and *push*

the *Notification Manager* implements the *Notification Broker* and *Pull Point* roles from the WS-Brokered Notification.

A.3 Request Processor

The *Request Processor* component is responsible for the execution of invocations targeted at node and network services. Service requests must be acknowledged, either by a specific wireless sensor node, or by a set of nodes within the WSN, where collaboration between nodes occur. For instance, a request for the information about a chemical drum, as described in the scenario in 4.1, or the request for the average temperature readings of several nodes to determine the temperature of a room.

Usually, an invocation of a web service is carried out by wrapping the service request and response in a Simple Object Access Protocol (SOAP) message, performing a SOAP message exchange. This procedure is not entirely sufficient for handling service invocations whose target is located on a node, since its connectivity cannot be ensured. A node may become unavailable due to various reasons:

- A node may not be immediately available for dealing with a request, for example due to temporary disconnectivity from the network during transportation, or due to other pending requests.
- A node can be temporarily unable to respond to requests due to wireless interference.
- A node has failed and is unable to recover (e.g., due to battery exhaustion or physical destruction).
- A service is not available on, or has been removed from the addressed node.

This volatile nature of WSNs imposes the need for not only synchronous invocations but also for asynchronous request support. With asynchronous invocations applications can post requests that will be buffered and further executed. This is achieved either when a specific sensor node becomes available, or when a sensor node receives the assignment for provisioning the requested service.

A.4 Component Interaction

Wireless sensor nodes continuously generate messages, either as responses to service requests, as regular sensor data and status reports, or due to randomly occurring

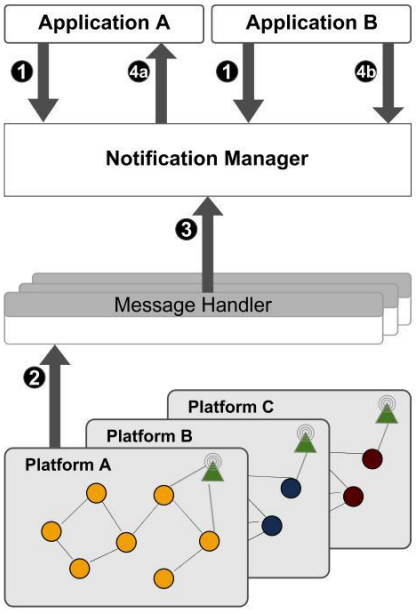


Figure A.1: Components interaction: notification.

events. The receivers of these messages are dynamically established depending on the requirements of the application, feedback from the user, or contextual information. This requires a loose coupling between message creators (i.e., the nodes) and their consumers (services and applications).

In the framework proposed, this loose connection is achieved through the interactions of the components responsible for enabling the supported message exchange patterns. These interactions can be divided in three main groups: Notification, Request-response and One-way.

Notification

Figure A.1 presents the components involved in the process of delivering the events generated by the wireless sensor nodes to the interested parties.

At first, client applications subscribe to receive events (1). As a client may not be interested in all messages from nodes within the WSN, a client is able to specify filters. An application might for example, only be interested in messages that originate from a specific node.

When a new event is generated by the WSN (2), the corresponding *Message Handler* captures the event, performs the required protocol conversions and posts it to the *Notification Manager* (3). At this point, the *Notification Manager* applies the filters and identifies which applications should be notified. According to the publishing method selected by the application at the subscription type, the *Notification Man-*

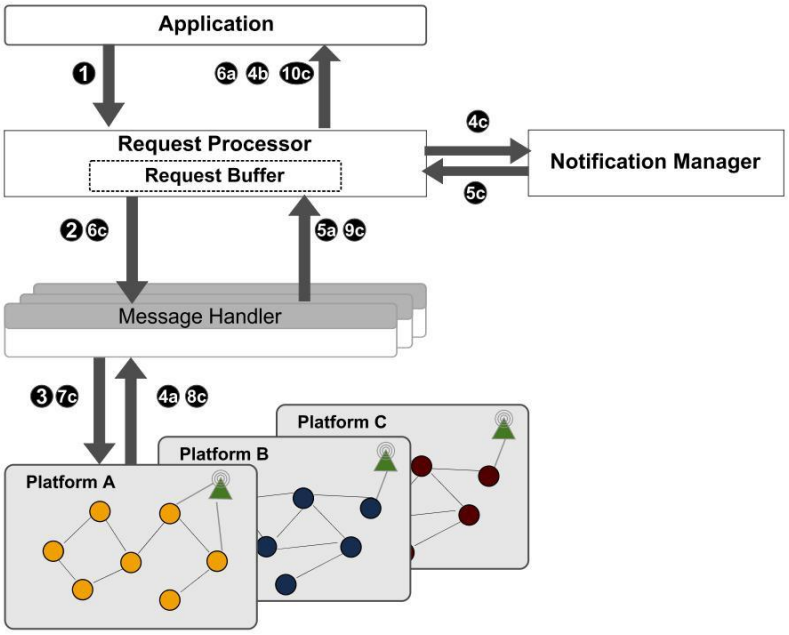


Figure A.2: Components interaction: invocation.

ager will either push the event to the applications (4a), or store it for further pulling from the application side (4b).

Request-response and One-way

Request-response and one-way message exchange patterns are similarly handled within the framework. Figure A.2 depicts the interaction between components for handling these two communication patterns.

At first, the application posts a request to the *Request Processor* (1). This component analyses the request, selects the corresponding *Message Handler* and forwards the request (2). At this point, the *Message Handler* performs the necessary protocol conversions and executes the invocation in the WSN (3).

For one-way communication the invocation is finalized at this point. For request-response, however, additional steps are required to handle the response from the WSN.

Steps (4a-6a) represent the message flow for synchronous and asynchronous requests when the sensor node (or set of nodes) is (are) connected. The node (or set of nodes) executes the request and send the response (4a), the *Message Handler* then converts the protocol and sends the result to the *Request Processor* (5a). Finally this component provides the result to the application (6a).

If no response is generated, after a timeout period, the service request is cancelled. This is a precaution against an overloading of the *Request Buffer*. For synchronous

requests, this represents the end of the invocation. In this case the requesting client application is notified about the failure (4b). For asynchronous invocations, the request will be buffered and executed once the node (or set of nodes) has recovered. To achieve this, the *Request Processor* subscribes itself to receive recovery events for the nodes involved in the request, or for the service reassignment events according to the request that has to be executed (4c).

As soon as the system recovers, the *Notification Manager* informs the *Request Processor* (5c). The request is once again forwarded to the *Message Handler* (6c) and sent to the WSN (7c). Once executed the result is delivered back to the application (8c-10c).

B. Pattern Combination

In Section 7.1.2 we presented the evaluation results for the pattern combination approach developed in this thesis for a network setup without overlap. Appendix B presents additional results for a network containing 10, 25 and 50 percent overlap.

Figures B.1, B.2 and B.3 depict the results for a network containing 10% overlap.

Figures B.4, B.5 and B.6 depict the results for a network containing 50% overlap.

It is interesting to note that the pattern combination approach has the same limitations of the membership function. With heavily connected networks, the variance of the message throughput of sensor nodes becomes very close to zero. If the timeframe is not large enough, the normal distribution approximation of the message throughput of sensor nodes becomes inaccurate. The consequence is that the calculated combined failures of such networks present more errors if the timeframe is not large enough.

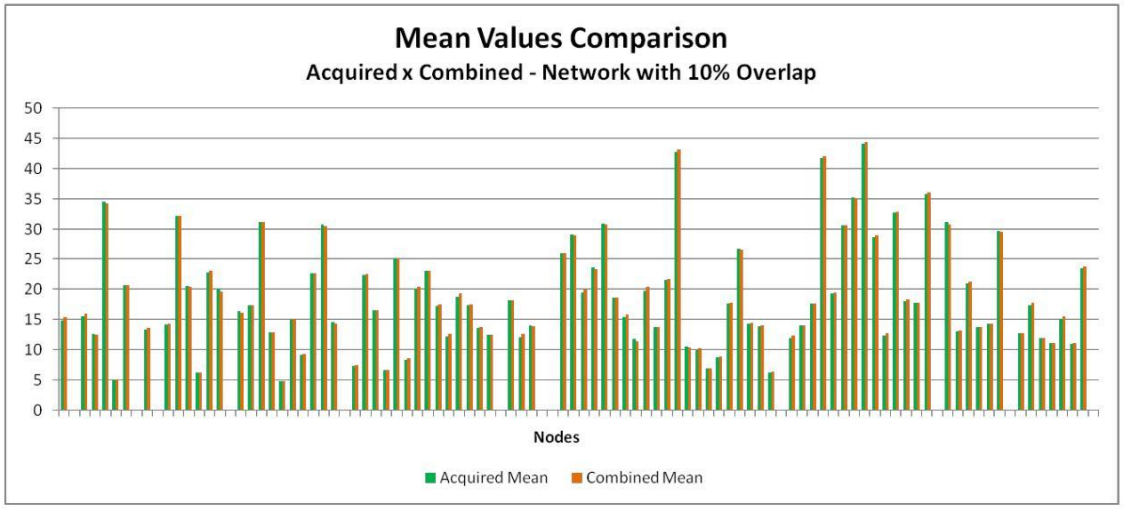


Figure B.1: Comparison between calculated and acquired mean values.

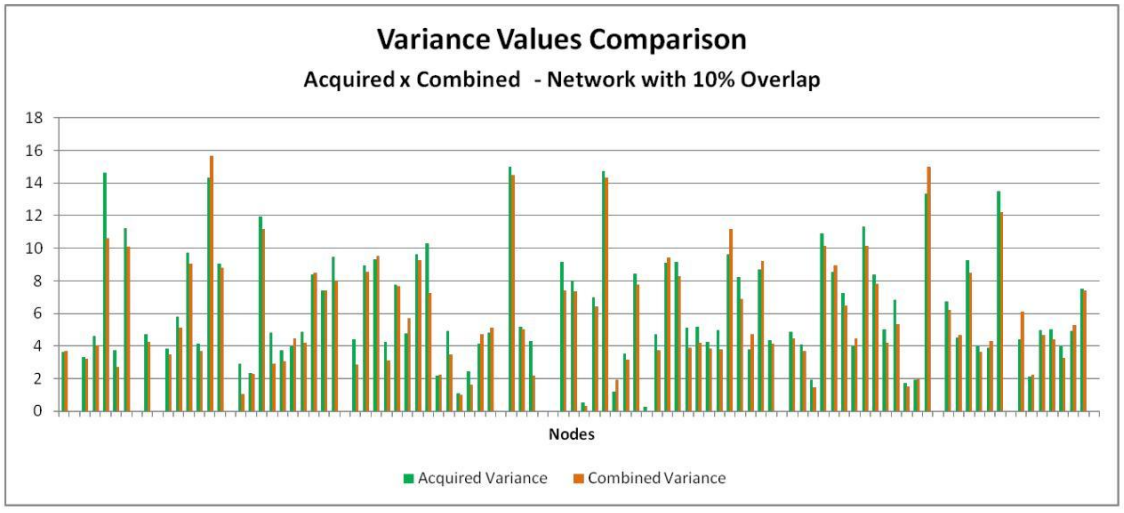


Figure B.2: Comparison between calculated and acquired variance values.

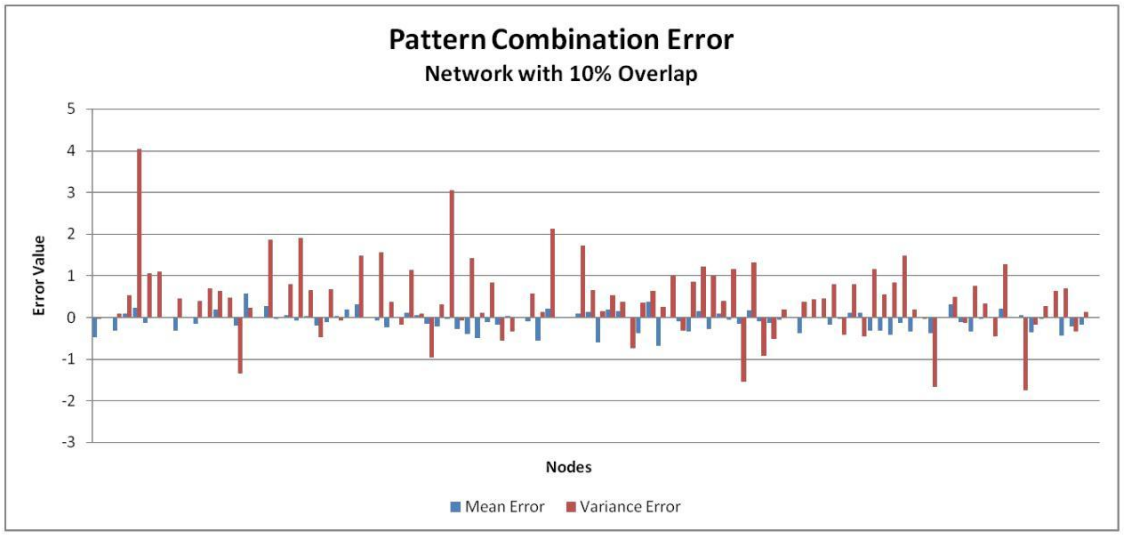


Figure B.3: Error between calculated and acquired mean and variance values.

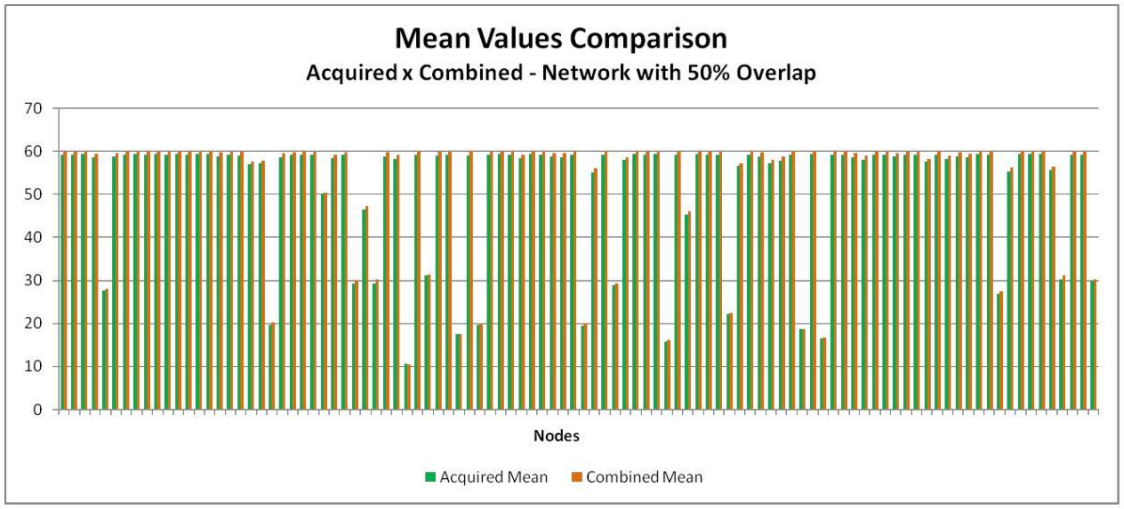


Figure B.4: Comparison between calculated and acquired mean values.

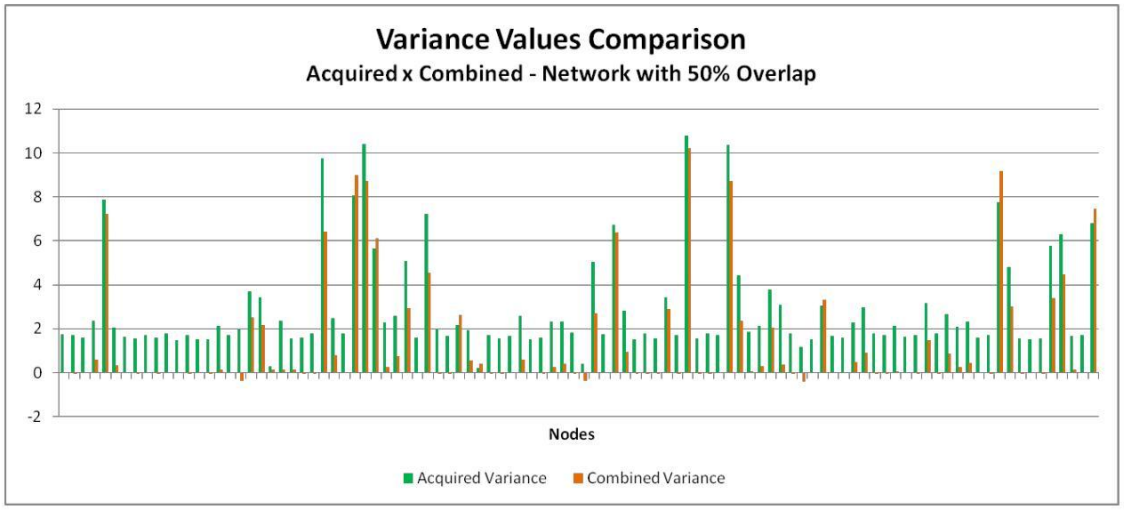


Figure B.5: Comparison between calculated and acquired variance values.

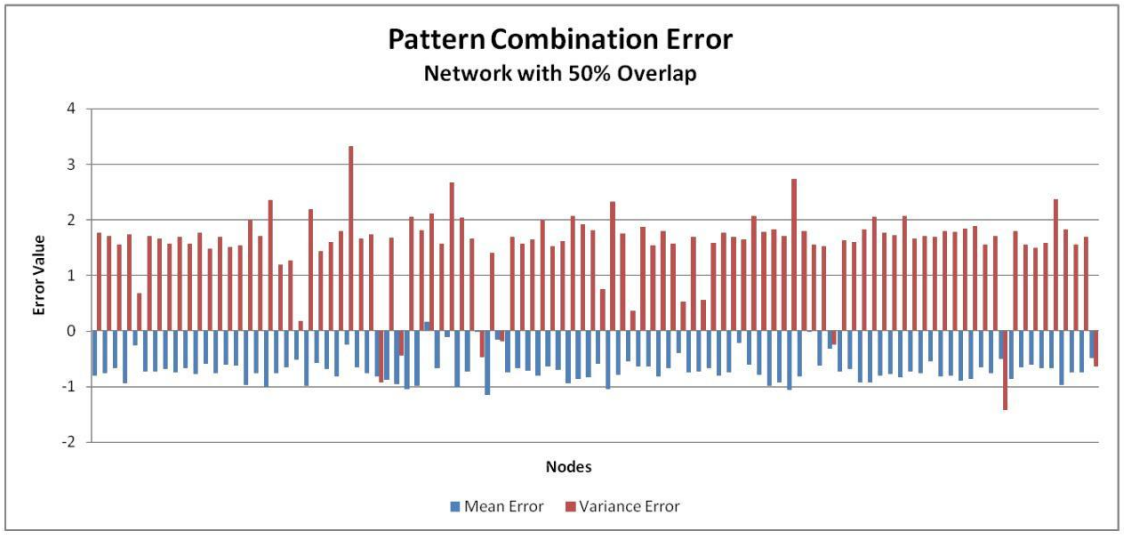


Figure B.6: Error between calculated and acquired mean and variance values.

Bibliography

[1] I. F. Akyildiz, W. Su, Y. Sankarasubramaniam, and E. Cayirci. A Survey on Sensor Networks. *IEEE Communications Magazine*, 40:102–114, August 2002.

[2] Z. Alliance. Zigbee specifications, version 1.0, April 2005.

[3] T. W. Anderson and J. D. Finn. *The New Statistical Analysis of Data*. Springer, 1997.

[4] J. Anke, J. Müller, P. Spieß, and L. W. F. Chaves. A Service-Oriented Middleware for Integration and Management of Heterogeneous Smart Items Environments. In *Proceedings of the 4th MiNEMA workshop in Sintra*, pages 7–11, July 2006.

[5] J. Anke, J. Müller, P. Spieß, and L. Weiss Ferreira Chaves. A service-oriented middleware for integration and management of heterogeneous smart items environments. In *4th MiNEMA workshop*, Sintra, Portugal, July 2006.

[6] O. Ardaiz, F. Freitag, and L. Navarro. On service deployment in ubiquitous computing. In *2nd International Workshop on Ubiquitous Computing and Communications*, 2001.

[7] P. Bahl and V. N. Padmanabhan. Radar: An in-building rf-based user location and tracking system. In I. C. S. Press, editor, *19th Annual Joint Conference of the IEEE Computer and Communications Societies*, volume 2, pages 775–784, Tel-Aviv, Israel, March 2000.

[8] S. Banerjee. *Industrial Hazards and Plant Safety*. Taylor & Francis Inc, 2003.

[9] T. Bass. Intrusion detection systems and multisensor data fusion. *Commun. ACM*, 43(4):99–105, 2000.

[10] M. A. Batalin, M. Rahimi, Y. Yu, D. Liu, A. Kansal, G. S. Sukhatme, W. J. Kaiser, M. Hansen, G. J. Pottie, M. Srivastava, and D. Estrin. Call and response: experiments in sampling the environment. In *SenSys '04: Proceedings*

of the 2nd international conference on Embedded networked sensor systems, pages 25–38, New York, NY, USA, 2004. ACM.

[11] L. Benini, G. Castelli, A. Macii, E. Macii, M. Poncino, and R. Scarsi. A Discrete-Time Battery Model for High-Level Power Estimation. In *Proceeding of the Design, Automation and Test in Europe Conference and Exhibition 2000*, pages 35–39, 2000.

[12] A. Birolini. *Quality and Reliability of Technical Systems:theory, practice, management*. Springer, 1997.

[13] A. Birolini. *Reliability Engineering: Theory and Practice*. Springer Berlin Heidelberg, 2007. ISBN 978-3-540-49390-7.

[14] Bluetooth.org. Bluetooth specification, version 2.1. http://www.bluetooth.com/Bluetooth/Learn/Technology/Specifications/, July 2007.

[15] D. Box, L. F. Cabrera, C. Critchley, and F. Curbera. Web Services Eventing (WS-Eventing). W3C, March 2006.

[16] T. Bray, J. Paoli, C. M. Sperberg-McQueen, E. Maler, F. Yergeau, and J. Cowan. Extensible markup language (XML) 1.1 (second edition). W3C, August 2006.

[17] J. L. Bredin, E. D. Demaine, M. Hajiaghayi, and D. Rus. Deploying Sensor Networks with Guaranteed Capacity and Fault Tolerance. In *Proceedings of the 6th ACM international symposium on Mobile ad hoc networking and computing*, pages 309–319, 2005.

[18] R. E. Bryant. Binary decision diagrams and beyond: Enabling technologies for formal verification. *iccad*, 00:0236, 1995.

[19] J. Burch, E. Clarke, D. Long, K. McMillan, and D. Dill. Symbolic model checking for sequential circuit verification. *Computer-Aided Design of Integrated Circuits and Systems, IEEE Transactions on*, 13(4):401–424, Apr 1994.

[20] T. A. Byrd, K. L. Cossick, and R. W. Zmud. A synthesis of research on requirements analysis and knowledge acquisition techniques. *MIS Q.*, 16(1):117–138, 1992.

[21] R. Cardell-Oliver, K. Smettem, M. Kranz, and K. Mayer. Field testing a wireless sensor network for reactive environmental monitoring [soil moisture

measurement]. In *Intelligent Sensors, Sensor Networks and Information Processing Conference, 2004. Proceedings of the 2004*, pages 7–12, Dec. 2004.

[22] L. W. F. Chaves, L. M. S. de Souza, J. Müller, and J. Anke. Service lifecycle management infrastructure for smart items. In *MidSens '06: Proceedings of the international workshop on Middleware for sensor networks*, pages 25–30, New York, NY, USA, 2006. ACM.

[23] B. Chen, K. Jamieson, H. Balakrishnan, and R. Morris. Span: An energy-efficient coordination algorithm for topology maintenance in ad hoc wireless networks. In *Mobile Computing and Networking*, pages 85–96, 2001.

[24] J. Chen, S. Kher, and A. Somani. Distributed fault detection of wireless sensor networks. In *DIWANS '06: Proceedings of the 2006 workshop on Dependability issues in wireless ad hoc networks and sensor networks*, pages 65–72, New York, NY, USA, 2006. ACM Press.

[25] P. W. H. Chung and M. Jefferson. The integration of accident databases with computer tools in the chemical industry. *Computers & Chemical Engineering*, 22(11):S729–S732, March 1998.

[26] CoBIs – Collaborative Business Objects. http://www.cobis-online.de/.

[27] Drools. http://www.jboss.org/drools/.

[28] D. S. Consortium. Scilab. http://www.scilab.org/.

[29] F. Cristian. Understanding fault-tolerant distributed systems. *Commun. ACM*, 34(2):56–78, 1991.

[30] L. M. S. de Souza. FT-CoWiseNets: A Fault Tolerance Framework forWireless Sensor Networks. In *International Conference on Sensor Technologies and Applications (SENSORCOMM)*, 2007.

[31] L. M. S. de Souza, R. Padilha, and C. Decker. Neural fault isolator. In *International Conference on Networked Sensing Systems*, 2008.

[32] C. Decker, T. Riedel, M. Beigl, L. M. S. de Souza, P. Spiess, S. Haller, and J. Müller. Collaborative business items. In *3rd IET International Conference on Intelligent Environments*, 2007.

[33] C. Decker, P. Spiess, L. Moreira Sá de Souza, M. Beigl, and Z. Nochta. Coupling enterprise systems with wireless sensor nodes: Analysis, implementation,

experiences and guidelines. In *Pervasive Technology Applied @ PERVASIVE*, pages 393–400, May 2006.

[34] F. C. Delicato, P. F. Pires, L. Rust, L. Pirmez, and J. F. de Rezende. Reflective Middleware for Wireless Sensor Networks. In *Proceedings of the 2005 ACM symposium on Applied computing*, pages 1155–1159, 2005.

[35] D. Desovski, Y. Liu, and B. Cukic. Linear randomized voting algorithm for fault tolerant sensor fusion and the corresponding reliability model. In *IEEE International Symposium on Systems Engineering*, pages 153–162, October 2005.

[36] W. E. Dietz, E. L. Kiech, and M. Ali. Pattern-based fault diagnosis using neural networks. In *IEA/AIE '88: Proceedings of the 1st international conference on Industrial and engineering applications of artificial intelligence and expert systems*, pages 13–23, New York, NY, USA, 1988. ACM Press.

[37] M. Ding, D. Chen, K. Xing, and X. Cheng. Localized fault-tolerant event boundary detection in sensor networks. In *INFOCOM*, 2005.

[38] V. Ebrahimipour, K. Suzuki, and A. Azadeh. An integrated off-on line approach for increasing stability and effectiveness of automated controlled systems based on pump dependability–case study: Offshore industry. *Journal of Loss Prevention in the Process Industries*, 19(6):542–552, November 2006.

[39] D. Estrin, D. Culler, K. Pister, and G. Sukhatme. Connecting the physical world with pervasive networks. *IEEE Pervasive Computing*, 1(1):59–69, 2002.

[40] D. Estrin, L. Girod, G. Pottie, and M. Srivastava. Instrumenting the world with wireless sensor networks. In *Acoustics, Speech, and Signal Processing, 2001. Proceedings. (ICASSP '01). 2001 IEEE International Conference on*, volume 4, pages 2033–2036, Salt Lake City, UT, USA, 2001.

[41] D. Estrin, R. Govindan, J. Heidemann, and S. Kumar. Next century challenges: scalable coordination in sensor networks. In *MobiCom '99: Proceedings of the 5th annual ACM/IEEE international conference on Mobile computing and networking*, pages 263–270, New York, NY, USA, 1999. ACM Press.

[42] L. Fausett, editor. *Fundamentals of neural networks: architectures, algorithms, and applications*. Prentice-Hall, Inc., Upper Saddle River, NJ, USA, 1994.

[43] R. Fielding, J. Gettys, J. Mogul, H. Frystyk, L. Masinter, P. Leach, and T. Berners-Lee. Hypertext transfer protocol – http/1.1. RFC 2616, June 1999.

[44] C.-L. Fok, G.-C. Roman, and C. Lu. Mobile Agent Middleware for Sensor Networks: an Application Case Study. In *Proceedings of the 4th International Symposium on Information Processing in Sensor Networks (IPSN)*, 2005.

[45] C. L. Forgy. Rete: a fast algorithm for the many pattern/many object pattern match problem. *Expert systems: a software methodology for modern applications*, pages 324–341, 1990.

[46] C. Frank and K. Römer. Algorithms for Generic Role Assignment in Wireless Sensor Networks. In *Proceedings of the 3rd international conference on Embedded networked sensor systems*, pages 230–242, 2005.

[47] P. Frank. Application of fuzzy logic to process supervision and fault diagnosis. In *Proceedings of the international workshop fuzzy technologies in automation and intelligent systems*, 1994.

[48] L. Fu. *Neural Networks in Computer Intelligence*. McGraw-Hill, Inc., New York, NY, USA, 1994.

[49] D. Ganesan, R. Govindan, S. Shenker, and D. Estrin. Highly-Resilient, Energy-Efficient Multipath Routing in Wireless Sensor Networks. *Mobile Computing and Communications Review*, 1(2), 1997.

[50] D. Giraud, C. Aubrun, D. Theilliol, and L. Vela Valdes. A fuzzy fault diagnosis method applied to a steam circuit. *Fuzzy Systems, 1996., Proceedings of the Fifth IEEE International Conference on*, 3:1944–1950 vol.3, Sep 1996.

[51] T. Gou and J. Nurre. Sensor failure detection and recovery by neural networks. In *IJCNN-91-Seattle International Joint Conference on Neural Networks*, 1991.

[52] R. Guerraoui and A. Schiper. Fault-Tolerance by Replication in Distributed Systems. In *Proceedings of the 1996 Ada-Europe International Conference on Reliable Software Technologies*, pages 38–57, 1996.

[53] G. Gupta and M. Younis. Fault-Tolerant Clustering of Wireless Sensor Networks. *Wireless Communications and Networking*, 3:1579–1584, 2003.

[54] H. Gupta, V. Navda, S. R. Das, and V. Chowdhary. Efficient gathering of correlated data in sensor networks. In *MobiHoc '05: Proceedings of the 6th ACM international symposium on Mobile ad hoc networking and computing*, pages 402–413, New York, NY, USA, 2005. ACM.

[55] I. Gupta, D. Riordan, and S. Sampalli. Cluster-Head Election Using Fuzzy Logic for Wireless Sensor Networks. In *Proceedings of the 3rd Annual Communication Networks and Services Research Conference*, pages 255–260, 2005.

[56] A. Hamzeh and K. Zaidan. Development of an expert system for off- and on-line faults diagnosis in electric power systems. *Information and Communication Technologies: From Theory to Applications, 2004. Proceedings. 2004 International Conference on*, pages 135–136, 19-23 April 2004.

[57] S. Harte and A. Rahman. Fault Tolerance in Sensor Networks Using Self-Diagnosing Sensor Nodes. In *The IEE International Workshop on Intelligent Enviroment*, pages 7–12, June 2005.

[58] W. R. Heinzelman, A. Chandrakasan, and H. Balakrishnan. Energy-Efficient Communication Protocol for Wireless Microsensor Networks. In *Proceedings of the 33rd Hawaii International Conference on System Sciences*, volume 8, page 8020, 2000.

[59] A. T. Hoang and M. Motani. Collaborative broadcasting and compression in cluster-based wireless sensor networks. *ACM Trans. Sen. Netw.*, 3(3):17, 2007.

[60] K. Holtzblatt and H. R. Beyer. Requirements gathering: the human factor. *Commun. ACM*, 38(5):31–32, 1995.

[61] J. C. Hoskins and D. M. Himmelblau. Artificial neural network models of knowledge representation in chemical engineering. *Computers & Chemical Engineering*, 12(9-10):881–890, September-October 1988.

[62] Y.-C. Huang, H.-T. Yang, and C.-L. Huang. An evolutionary computation based fuzzy fault diagnosis system for a power transformer. *Fuzzy Systems Symposium, 1996. 'Soft Computing in Intelligent Systems and Information Processing'., Proceedings of the 1996 Asian*, pages 218–223, Dec 1996.

[63] M. N. Huhns and M. P. Singh. Service-Oriented Computing: Key Concepts and Principles. *IEEE Internet Computing*, 9(1), 2005.

[64] E. Hull, K. Jackson, and J. Dick. *Requirements Engineering*. Springer, 2005.

[65] I. S. Institute. Network Simulator 2 (NS2). http://www.isi.edu/nsnam/ns/.

[66] C. Intanagonwiwat, R. Govindan, D. Estrin, J. Heidemann, and F. Silva. Directed diffusion for wireless sensor networking. *IEEE/ACM Trans. Netw.*, 11(1):2–16, 2003.

[67] R. Isermann. On fuzzy logic applications for automatic control, supervision, and fault diagnosis. *Systems, Man and Cybernetics, Part A, IEEE Transactions on*, 28(2):221–235, Mar 1998.

[68] Y. Ishida. An application of qualitative reasoning to process diagnosis: automatic rule generation by qualitative simulation. *Artificial Intelligence Applications, 1988., Proceedings of the Fourth Conference on*, pages 124–129, 14-18 Mar 1988.

[69] H. Jain, P. Vitharana, and F. M. Zahedi. An assessment model for requirements identification in component-based software development. *SIGMIS Database*, 34(4):48–63, 2003.

[70] D. B. Johnson and D. A. Maltz. Dynamic source routing in ad hoc wireless networks. In Imielinski and Korth, editors, *Mobile Computing*, volume 353. Kluwer Academic Publishers, 1996.

[71] D. B. Johnson and D. A. Maltz. Dynamic source routing in ad hoc wireless networks. In Imielinski and Korth, editors, *Mobile Computing*, volume 353. Kluwer Academic Publishers, 1996.

[72] Joone - Java Object Oriented Neural Engine. http://www.jooneworld.com/.

[73] C.-C. Jou. Comparing learning performance of neural networks and fuzzy systems. *Neural Networks, 1993., IEEE International Conference on*, pages 1028–1033 vol.2, 1993.

[74] P. Juang, H. Oki, Y. Wang, M. Martonosi, L. S. Peh, and D. Rubenstein. Energy-efficient computing for wildlife tracking: design tradeoffs and early experiences with zebranet. *SIGARCH Comput. Archit. News*, 30(5):96–107, 2002.

[75] JUNG. Java universal network/graph framework. http://jung.sourceforge.net/.

[76] C. Karlof and D. Wagner. Secure Routing in Wireless Sensor Networks: Attacks and Coutermeasures. In *Proceedings of the First IEEE Sensor Network Protocols and Applications*, pages 113–127, May 2003.

[77] B. Karp and H. T. Kung. GPSR: greedy perimeter stateless routing for wireless networks. In *Mobile Computing and Networking*, pages 243–254, 2000.

[78] T. Kimura, S. Nishimatsu, Y. Ueki, and Y. Fukuyama. Development of an expert system for estimating fault section in control center based on protective system simulation. *Power Delivery, IEEE Transactions on*, 7(1):167–172, Jan 1992.

[79] K. Kishida, M. Maeda, H. Miyajima, and S. Murashima. A self-tuning method of fuzzy modeling with learning vector quantization. *Fuzzy Systems, 1997., Proceedings of the Sixth IEEE International Conference on*, 1:397–402 vol.1, Jul 1997.

[80] M. Kochhal, L. Schwiebert, and S. Gupta. Role-based hierarchical self organization for wireless ad hoc sensor networks. In *Proceedings of the 2nd ACM international conference on Wireless sensor networks and applications*, pages 98–107, 2003.

[81] M. Kochhal, L. Schwiebert, and S. Gupta. Role-based Middleware for Sensor Networks. Technical report, Wayne State University, May 2004.

[82] N. Komoda. Service oriented architecture (SOA) in industrial systems. In *Industrial Informatics, 2006 IEEE International Conference on*, pages 1–5, Singapore, Aug. 2006.

[83] C.-Y. Koo. Broadcast in Radio Networks Tolerating Byzantine Adversarial Behavior. In *Proceedings of the twenty-third annual ACM symposium on Principles of distributed computing*, pages 275–282, 2004.

[84] J. Korbicz, J. M. Koscielny, Z. Kowalczuk, and W. Cholewa. *Fault Diagnosis: Models, Artificial Intelligence, Applications*. Springer, 2004.

[85] A. I. Kostrikin and Y. I. Manin. *Linear Algebra and Geometry (Algebra, Logic and Applications)*. Gordon and Breach Science Publishers, 1989.

[86] F. Koushanfar, M. Potkonjak, and A. Sangiovanni-Vincentell. Fault Tolerance Techniques for Wireless Ad hoc Sensor Networks. In *Proceedings of IEEE Sensors*, volume 2, pages 1491–1496, 2002.

[87] M. A. Kramer and J. A. Leonard. Diagnosis using backpropagation neural networks–analysis and criticism. *Computers & Chemical Engineering*, 14(12):1323–1338, December 1990.

[88] B. Krishnamachari and S. Iyengar. Distributed Bayesian Algorithms for Fault-Tolerant Event Region Detection in Wireless Sensor Networks. *IEEE Transactions on Computers*, 53:241–250, March 2004.

[89] K. Kumar, V. Dakshinamoorthy, and M. S. Krishnan. Does SOA improve the supply chain? an empirical analysis of the impact of SOA adoption on electronic supply chain performance. In *System Sciences, 2007. HICSS 2007. 40th Annual Hawaii International Conference on*, Waikoloa, HI, Jan. 2007.

[90] U.S. Department of Labor. http://www.dol.gov.

[91] S. N. Laboratories. Jess. http://herzberg.ca.sandia.gov/.

[92] L. Lamport, R. Shostak, and M. Pease. The Byzantine Generals Problem. *ACM Transactions on Programming Languages and Systems*, 4:382–401, 1982.

[93] K. Langendoen, A. Baggio, and O. Visser. Murphy loves potatoes: experiences from a pilot sensor network deployment in precision agriculture. In *IPDPS 20th International Parallel and Distributed Processing Symposium*, 2006.

[94] C. Legner and F. Thiesse. RFID-based maintenance at Frankfurt airport. *IEEE Pervasive Computing*, 5:34–39, March 2006.

[95] P. Levis and D. Culler. Maté: A Tiny Virtual Machine for Sensor Networks. In *ASPLOS-X: Proceedings of the 10th international conference on Architectural support for programming languages and operating systems*, pages 85–95, New York, NY, USA, 2002. ACM Press.

[96] H. Li, M. C. Price, J. Stott, and I. W. Marshall. The development of a wireless sensor network sensing node utilising adaptive self-diagnostics. In *International Workshop on Self-Organizing Systems*, 2007.

[97] N. Li and J. C. Hou. FLSS: A Fault-Tolerant Topology Control Algorithm for Wireless Networks. In *Proceedings of the 10th Annual International Conference on Mobile Computing and Networking*, pages 275–286, 2004.

[98] Q. Liang. Clusterhead Election for Mobile Ad hoc Wireless Network. In *Proceedings on Personal, Indoor and Mobile Radio Communications*, volume 2, pages 1623–1628, 2003.

[99] T. Liu and M. Martonosi. Impala: A Middleware System for Managing Autonomic, Parallel Sensor Systems. In *PPoPP '03: Proceedings of the ninth ACM SIGPLAN symposium on Principles and practice of parallel programming*, pages 107–118, New York, NY, USA, 2003. ACM Press.

[100] B. Lloyd, G. Stone, and J. Stein. Development of an expert system to assess machine insulation condition. *Electrical Electronics Insulation Conference, 1991. Boston '91 EEIC/ICWA Exposition., Proceedings of the 20th*, pages 33–37, 7-10 Oct 1991.

[101] Y. Lu, T. Q. Chen, and B. Hamilton. A fuzzy system for automotive fault diagnosis: fast rule generation and self-tuning. *Vehicular Technology, IEEE Transactions on*, 49(2):651–660, Mar 2000.

[102] C. Ma and Y. Yang. A Prioritized Battery-aware Routing Protocol for Wireless Ad hoc Networks. In *Proceedings of the 8th ACM international symposium on Modeling, analysis and simulation of wireless and mobile systems*, pages 45–52, 2005.

[103] P. C. Mahalanobis. On the generalized distance in statistics. In *National Institute of Science of India*, volume II, 1936.

[104] A. Mainwaring, D. Culler, J. Polastre, R. Szewczyk, and J. Anderson. Wireless sensor networks for habitat monitoring. In *WSNA '02: Proceedings of the 1st ACM international workshop on Wireless sensor networks and applications*, pages 88–97, New York, NY, USA, 2002. ACM Press.

[105] G. M. Marakas and J. J. Elam. Semantic structuring in analyst acquisition and representation of facts in requirements analysis. *Info. Sys. Research*, 9(1):37–63, 1998.

[106] D. Marculescu, N. H. Zamora, P. Stanley-Marbell, and R. Marculescu. Fault-Tolerant Techniques for Ambient Intelligent Distributed Systems. In *Proceedings of the 2003 IEEE/ACM international conference on Computer-aided design*, page 348, 2003.

[107] M. Marin-Perianu, N. Meratnia, P. Havinga, L. Sa De Souza, J. Muller, P. Spiess, S. Haller, R. T., C. Decker, and G. Stromberg. Decentralized enterprise systems: a multiplatform wireless sensor network approach. *Wireless Communications, IEEE [see also IEEE Personal Communications]*, 14(6):57–66, December 2007.

[108] S. Marti, T. J. Giuli, K. Lai, and M. Baker. Mitigating Routing Misbehavior in Mobile Ad hoc Networks. In *Proceedings of the 6th annual international conference on Mobile computing and networking*, pages 255–265, 2000.

[109] K. Martinez, J. K. Hart, and R. Ong. Environmental sensor networks. *Computer*, 37(8):50–56, Aug. 2004.

[110] K. Martinez, P. Padhy, A. Riddoch, H. Ong, and J. Hart. Glacial environment monitoring using sensor networks. In *REALWSN' 05*, 2005.

[111] K. Marzullo. Tolerating failures of continuous-valued sensors. *ACM Transactions on Computer Systems*, 8(4):284–304, 1990.

[112] D. L. Mattern, L. C. Jaw, T.-H. Guo, R. Graham, and W. McCoy. Using neural networks for sensor validation. Technical report, National Aeronautics and Space Administration, 1998.

[113] A. Meier, T. Rein, J. Beutel, and L. Thiele. Coping with unreliable channels: Efficient link estimation for low-power wireless sensor networks. In *5th International Conference on Networked Sensing Systems (INSS2008)*, June 2008.

[114] S. Microsystems. Java. http://java.sun.com/.

[115] S. Microsystems. Mysql server. http://www.mysql.com/.

[116] A. Misra, G. Provan, G. Karsai, G. Bloor, and E. Scarl. A generic and symbolic model-based diagnostic reasoner with highly scalable properties. *Systems, Man, and Cybernetics, 1998. 1998 IEEE International Conference on*, 4:3154–3160 vol.4, 11-14 Oct 1998.

[117] R. K. Mobley. *An Introduction to Predictive Maintenance*. Butterworth-Heinemann, second edition, 2002.

[118] R. Moore. *Making Common Sense Common Practice: Models for Manufacturing Excellence*. Butterworth-Heinemann, 2004.

[119] E. F. Nakamura, A. A. F. Loureiro, and A. C. Frery. Information fusion for wireless sensor networks: Methods, models, and classifications. *ACM Comput. Surv.*, 39(3):9, 2007.

[120] F. Niederman, R. G. Mathieu, R. Morley, and I.-W. Kwon. Examining rfid applications in supply chain management. *Commun. ACM*, 50(7):92–101, 2007.

[121] Z. Nivolianitou, M. Konstandinidou, C. Kiranoudis, and N. Markatos. Development of a database for accidents and incidents in the greek petrochemical industry. *Journal of Loss Prevention in the Process Industries*, 19(5):630–638, November 2006.

[122] Z. Nochta and L. M. S. de Souza. Deliverable d102: Architecture and service description. Technical report, EU Project CoBIs IST-004270, 2005.

[123] G. P. Norstrom. Fire/explosion losses in the cpi. *Chemical Engineering Progress*, 8:80, 1982.

[124] T. Novak, J. Meigs, and R. Sanford. Development of an expert system for diagnosing component-level failures in a shuttle car. *Industry Applications, IEEE Transactions on*, 25(4):691–698, Jul/Aug 1989.

[125] OASIS. Web Services Base Notification 1.2. http://docs.oasis-open.org/wsn/2004/06/wsn-WS-BaseNotification-1.2-draft-03.pdf, June 2004.

[126] OASIS. Web Services Notification. http://docs.oasis-open.org/wsn/wsn-ws_brokered_notification-1.3-spec-os.pdf, October 2006.

[127] P. Patrick. Impact of soa on enterprise information architectures. In *SIGMOD '05: Proceedings of the 2005 ACM SIGMOD international conference on Management of data*, pages 844–848, New York, NY, USA, 2005. ACM.

[128] P. Perera. Fault diagnosis system for wireless sensor networks. Master's thesis, Universität Karlsruhe (TH), 2007.

[129] R. Perla, S. Mukhopadhyay, and A. Samanta. Sensor fault detection and isolation using artificial neural networks. In *TENCON 2004. 2004 IEEE Region 10 Conference*, 2004.

[130] W. R. Pestman. *Mathematical Statistics*. Walter de Gruyter Gmbh & Co., 1998.

[131] W. H. Press, S. A. Teukolsky, W. T. Vetterling, and B. P. Flannery, editors. *Numerical Recipes 3rd Edition: The Art of Scientific Computing*. Cambridge University Press, 2007.

[132] D. Rakhmatov, S. Vrudhula, and D. A. Wallach. Battery Lifetime Prediction for Energy-aware Computing. In *Proceedings of the 2002 international symposium on Low power electronics and design*, pages 154–159, 2002.

[133] D. Rakhmatov and S. B. Vrudhula. Time-to-Failure Estimation for Batteries in Portable Electronic Systems. In *Proceedings of the 2001 international symposium on Low power electronics and design*, pages 88–91, 2001.

[134] N. Ramanathan, L. Balzano, M. Burt, D. Estrin, E. Kohler, T. Harmon, C. Harvey, J. Jay, S. Rothenberg, and M. Srivastava. Rapid deployment with confidence: Calibration and fault detection in environmental sensor networks. Technical Report 62, Center for Embedded Networked Sensing, UCLA and Department of Civil and Environmental Engineering, MIT, April 2006.

[135] N. Ramanathan, K. Chang, R. Kapur, L. Girod, E. Kohler, and D. Estrin. Sympathy: a debugging system for sensor networks. In *IEEE International Conference on Local Computer Networks*, 2004.

[136] A. Renfrew and J. Tian. The use of a knowledge-based system in power electronic circuit fault diagnosis. *Power Electronics and Applications, 1993., Fifth European Conference on*, pages 57–62 vol.7, 13-16 Sep 1993.

[137] F. C.-H. Rhee and R. Krishnapuram. Fuzzy rule generation methods for high-level computer vision. *Fuzzy Sets Syst.*, 60(3):245–258, 1993.

[138] M. Rieback, B. Crispo, and A. Tanenbaum. The evolution of RFID security. *IEEE Pervasive Computing*, 5:62–69, March 2006.

[139] M. Ringwald, K. Römer, and A. Vitaletti. Snif: Sensor network inspection framework. Technical Report 535, ETH Zürich, Institute for Pervasive Computing, October 2006.

[140] P. Rong and M. Pedram. Extending the lifetime of a network of battery-powered mobile devices by remote processing: A markovian decision-based approach. In *DAC '03: Proceedings of the 40th conference on Design automation*, pages 906–911, New York, NY, USA, 2003. ACM Press.

[141] S. Rost and H. Balakrishnan. Memento: A health monitoring system for wireless sensor networks. In *SECON*, 2006.

[142] A. Rudenko, P. Reiher, G. J. Popek, and G. H. Kuenning. The Remote Processing Framework for Portable Computer Power Saving. In *SAC '99: Proceedings of the 1999 ACM symposium on Applied computing*, pages 365–372, New York, NY, USA, 1999. ACM Press.

[143] L. B. Ruiz, I. G. Siqueira, L. B. e Oliveira, H. C. Wong, J. M. S. Nogueira, and A. A. F. Loureiro. Fault Management in Event-driven Wireless Sensor Networks. In *Proceedings of the 7th ACM international symposium on Modeling, analysis and simulation of wireless and mobile systems*, pages 149–156, June 2004.

[144] S. J. Russell and P. Norvig. *Artificial Intelligence: A Modern Approach*. Pearson Education, 2003.

[145] SAP. Environment, health, and safety compliance management applications. http://www.sap.com/solutions/grc/ehscompliance/index.epx.

[146] SAP. CoBIs, Deliverable D102, Architecture and Service Description. http://www.cobis-online.de/, March 2007.

[147] SAP. CoBIs, Deliverable D104, Final Project Report. http://www.cobis-online.de/, March 2007.

[148] A. Scaglione and S. D. Servetto. On the interdependence of routing and data compression in multi-hop sensor networks. In *MobiCom '02: Proceedings of the 8th annual international conference on Mobile computing and networking*, pages 140–147, New York, NY, USA, 2002. ACM.

[149] Scalable Network Technologies. QualNet Simulator. http://www.scalablenetworks.com.

[150] T. Schmid, H. Dubois-Ferrière, and M. Vetterli. Sensorscope: Experiences with a wireless building monitoring sensor network. In *REALWSN' 05*, 2005.

[151] E. Shih, S.-H. Cho, N. Ickes, R. Min, A. Sinha, A. Wang, and A. Chandrakasan. Physical layer driven protocol and algorithm design for energy-efficient wireless sensor networks. In *MobiCom '01: Proceedings of the 7th annual international conference on Mobile computing and networking*, pages 272–287, New York, NY, USA, 2001. ACM.

[152] P. Smyth. Hidden markov models and neural networks for fault detection in dynamic systems. In *Proceedings of Neural Networks for Signal Processing*, 1993.

[153] P. Spieß, H. Vogt, and H. Jütting. Integrating sensor networks with business processes. In *Real-World Sensor Networks Workshop at ACM MobiSys*, Uppsala, Sweeden, June 2006.

[154] M. R. Spiegel, J. J. Schiller, and R. A. Srinivasan, editors. *Schaum's Outline of Probability and Statistics*. McGraw-Hill, Inc., 2000.

[155] J. Staddon, D. Balfanz, and G. Durfee. Efficient Tracing of Failed nodes in Sensor Networks. In *Proceedings of the 1st ACM international workshop on Wireless sensor networks and applications*, pages 122–130, 2002.

[156] M. Strohbach, H.-W. Gellersen, G. Kortuem, and C. Kray. Cooperative artefacts: Assessing real world situations with embedded technology. In *Ubicomp*, pages 250–267, 2004.

[157] H. S. Su and Q. Z. Li. Transformer insulation fault diagnosis method based on fuzzy expert systems. *Properties and applications of Dielectric Materials, 2006. 8th International Conference on*, pages 343–346, June 2006.

[158] R. Szewczyk, A. Mainwaring, J. Polastre, J. Anderson, and D. Culler. An analysis of a large scale habitat monitoring application. In *SenSys '04: Proceedings of the 2nd international conference on Embedded networked sensor systems*, pages 214–226, New York, NY, USA, 2004. ACM Press.

[159] R. Szewczyk, J. Polastre, A. M. Mainwaring, and D. E. Culler. Lessons from a sensor network expedition. In *EWSN*, pages 307–322, 2004.

[160] S. T. and K. H. N. Application of artificial neural networks in process fault diagnosis : Fault detection, supervision and safety for technical processes. *Automatica*, 29(4):843–849, 1993.

[161] V. Talwar, D. Milojicic, Q. Wu, C. Pu, W. Yan, and G. Jung. Approaches for service deployment. *IEEE Internet Computing*, 9(2):70–80, March-April 2005.

[162] A. S. Tanenbaum and M. van Steen. *Distributed Systems: Principles and Paradigms*. Prentice Hall, 2002.

[163] J. Tateson, C. Roadknight, A. Gonzalez, T. Khan, S. Fitz, I. Henning, N. Boyd, and C. Vincent. Real world issues in deploying a wireless sensor network for oceanography. In *RealWSN' 05*, 2005.

[164] R. C. Teske. Fault detection using consensus in wireless sensor networks. Master's thesis, Federal University of Santa Catarina, August 2007.

[165] A. Thede, A. Schmidt, and C. Merz. Integration of goods delivery supervision into e-commerce supply chain. In *WELCOM '01: Proceedings of the Second International Workshop on Electronic Commerce*, pages 206–218, London, UK, 2001. Springer-Verlag.

[166] J. Thelen, D. Goense, and K. Langendoen. Radio wave propagation in potato fields. In *First Workshop on Wireless Network Measurement*, 2005.

[167] G. Tolle and D. Culler. Design of an application-cooperative management system for wireless sensor networks. In *EWSN*, 2005.

[168] G. Tolle, J. Polastre, R. Szewczyk, D. Culler, N. Turner, K. Tu, S. Burgess, T. Dawson, P. Buonadonna, D. Gay, and W. Hong. A macroscope in the redwoods. In *SenSys '05: Proceedings of the 3rd international conference on*

Embedded networked sensor systems, pages 51–63, New York, NY, USA, 2005. ACM Press.

[169] Particle Computers. http://www.teco.edu.

[170] A. Varga. The omnet++ discrete event simulation system. In *European Simulation Multiconference (ESM'2001)*, Prague, Czech Republic, June 2001.

[171] I. Vessey and S. Conger. Learning to specify information requirements: the relationship between application and methodology. *J. Manage. Inf. Syst.*, 10(2):177–201, 1993.

[172] L. Wang and M. Mendel. Generating fuzzy rules by learning from examples. In *IEEE Transactions on Systems, Man and Cybernetics*, 1992.

[173] X. Wang and W. Liu. A fuzzy fault diagnosis scheme with application. *IFSA World Congress and 20th NAFIPS International Conference, 2001. Joint 9th*, 3:1489–1493 vol.3, July 2001.

[174] G. Werner-Allen, K. Lorincz, M. Ruiz, O. Marcillo, J. Johnson, J. Lees, and M. Welsh. Deploying a wireless sensor network on an active volcano. *IEEE Internet Computing*, 10:18–25, 2006.

[175] A. Woo, T. Tong, and D. Culler. Taming the Underlying Challenges of Reliable Multihop Routing in Sensor Networks. In *SenSys*. ACM, 2003.

[176] A. D. Wood and J. A. Stankovic. Denial of Service in Sensor Networks. *IEEE Computer*, 35:54–62, 2002.

[177] D.-M. Xie, X. Song, H.-L. Zhou, and M.-W. Guo. Fuzzy vibration fault diagnosis system of steam turbo-generator rotor. *Machine Learning and Cybernetics, 2002. Proceedings. 2002 International Conference on*, 1:411–415 vol.1, 2002.

[178] W. Xu, W. Trappe, and Y. Zhang. Channel surfing: defending wireless sensor networks from interference. In *IPSN '07: Proceedings of the 6th international conference on Information processing in sensor networks*, pages 499–508, New York, NY, USA, 2007. ACM.

[179] X. Xu, J. Hines, and R. Uhrig. Sensor validation and fault detection using neural networks. In *Proceedings of the Maintenance and Reliability Conference (MARCON), Gatlinburg, TN, May 10-12*, 1999.

[180] Y. Xu, J. Heidemann, and D. Estrin. Geography-informed energy conservation for ad hoc routing. In *MobiCom '01: Proceedings of the 7th annual international conference on Mobile computing and networking*, pages 70–84, New York, NY, USA, 2001. ACM.

[181] R. R. Yager, S. Ovchinnikov, R. M. Tong, and H. T. Nguyen, editors. *Fuzzy sets and applications*. Wiley-Interscience, New York, NY, USA, 1987.

[182] L. A. Zadeh. Towards a theory of fuzzy systems. *Fuzzy sets, fuzzy logic, and fuzzy systems: selected papers by Lotfi A. Zadeh*, pages 83–104, 1996.

[183] E. Zeeb, A. Bobek, H. Bohn, and F. Golatowski. Service-oriented architectures for embedded systems using devices profile for web services. In *Advanced Information Networking and Applications Workshops, 2007, AINAW '07. 21st International Conference on*, volume 1, pages 956–963, Niagara Falls, ON, Canada, May 2007.

[184] B. Y. Zhao, J. Kubiatowicz, and A. D. Joseph. Tapestry: An Infrastructure for Fault-tolerant Wide-area Location and Routing. Technical Report UCB/CSD-01-1141, Computer Science Division, University of California, Berkeley, 2001.

[185] J. Zhao and R. Govindan. Understanding packet delivery performance in dense wireless sensor networks. In *SenSys '03: Proceedings of the 1st international conference on Embedded networked sensor systems*, pages 1–13, New York, NY, USA, 2003. ACM.

[186] H.-J. Zimmermann. *Fuzzy set theory—and its applications (3rd ed.)*. Kluwer Academic Publishers, Norwell, MA, USA, 1996.