Technical Report

Fault-Tolerance Mechanisms for ZigBee Cluster-Tree Wireless Sensor Networks

Skender Ben Attia

HURRAY-TR-070710

Version: 1

Date: 10-07-2007

Fault-Tolerance Mechanisms for ZigBee Cluster-Tree Wireless Sensor Networks

Skender Ben Attia

IPP-HURRAY!

Polytechnic Institute of Porto (ISEP-IPP)

Rua Dr. António Bernardino de Almeida, 431

4200-072 Porto

Portugal

Tel.: +351.22.8340509, Fax: +351.22.8340509

E-mail: sbat@isep.ipp.pt

http://www.hurray.isep.ipp.pt

Abstract

المدرسة العليا للمواصلات بتونس
ECOLE SUPERIEURE DES COMMUNICATIONS DE TUNIS

CYCLE DE FORMATION DES INGENIEURS EN TELECOMMUNICATIONS

OPTION :
INGENIERIE DES RESEAUX

RAPPORT DE PROJET DE FIN D'ETUDES

Thème :

Fault-Tolerance Mechanisms for ZigBee Cluster-Tree Wireless Sensor Networks

Réalisé par :
SKANDER BEN ATTIA

ENCADRANTS :

Dr. Anis KOUBAA	IPP-Hurray !
Pr. Mario ALVES	IPP-Hurray !
Dr. Mounir FRIKHA	Sup'Com

Travail proposé et réalisé en collaboration avec :

ANNÉE UNIVERSITAIRE : 2006/2007

Cité technologique des communications – Route de Raoued Km 3.5 – 2083 El Ghazala Ariana
E-mail : courriel@supcom.rnu.tn – Web : www.supcom.mincom.tn
Tél : (+216) 71 857 000 – Fax : (+216) 71 856 829

This work is dedicated to all those I love…

Acknowledgments

*My deepest gratitude goes to both my supervisors **Dr Anis Koubâa** and **Prof. Mário Alves** for having given me the opportunity to make this project and whose invaluable help, guidance and support greatly improved the quality of this work.*

*I also thank all the members of the "IPP-Hurray!" lab, especially **André Cunha** for the great atmosphere in the lab that made the daily work on this project a real pleasure.*

*A special thank also goes to **Dr. Mounir Frikha** for his supervision and the efforts he provides for our school as a teacher and as a coordinator helping improving international cooperation in Sup'Com.*

A special thought also goes to all the teachers I had during the three years I spent in Sup'Com, they have done a great job proving me (and my friends) with a great amount of knowledge, ensuring at the same time a particularly cheerful atmosphere that ensured that learning with them a nice experience.

Foreword

The following work has been performed in the context of preparing the national engineering diploma in Telecommunications (specialty: network engineering) at the “Higher School for Communications of Tunis” (Sup’Com). This project was carried out in the IPP-Hurray! laboratory at the “Instituto Superior de Engenharia do Porto” part of the “Instituto Polytechnico do Porto”, Portugal. The aim of the project is to improve standard’s native behaviour in the presence of a failure, by proposing two fault-tolerance mechanisms that reduce network inaccessibility times and that are backward compatible with the IEEE 802.15.4/ZigBee standards.

Contents

List of Figures

List of Tables

List of Annexes

List of Abbreviations

APL	Application
BI	Beacon Interval
BO	Beacon Order
CID	Cluster Identifier
CLH	Cluster Head
Dp	Parent Depth
ED	Energy Detection
Ei	Energy indicator
FFD	Full Function Device
ISM	Industrial Scientific and Medical
IT	Inaccessibility Time
LCS	Logical Channel Set
LQ	Link Quality
LQI	Link Quality Indicator
LR-WPAN	Low-Rate Wireless Personal Area Network
NWK	Network
NWL	Network Layer
OSI	Open Systems Interconnection
PAI	Parent Assessment Indicator
PAN	Personal Area Network
PANC	PAN Coordinator
POS	Personal Operation Space
RF	Radio Frequency
RFD	Reduced Function Device
SD	Superframe Duration
SO	Superframe Order
Tf	Transmit failure
TRX	Transceiver
TX	Transmit or Transmitter
WSN	Wireless Sensor Networks
ZR	ZigBee Router

Chapter 1

Introduction

In this first chapter, we present the general motivation of this project and the main problem that has been addressed in this work. In addition, we present the research context where this project has been carried out, and finally, we conclude with an outline of the report structure.

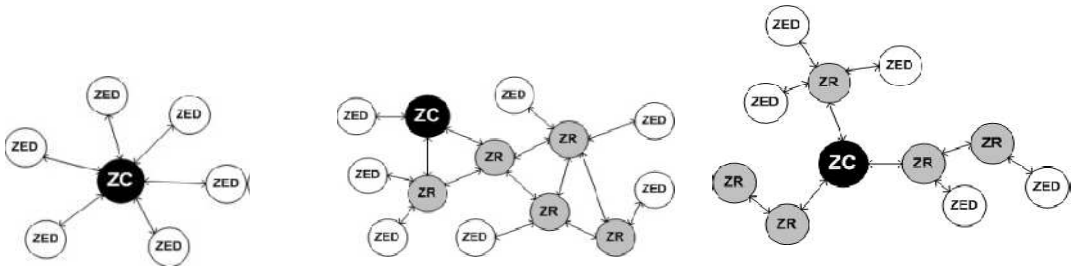
1.1. General Problem and Motivation

Recent advances in wireless communications and electronics has enabled the development of low-cost, low data rate Wireless Sensor Networks (WSNs). Such networks are enabling a giant leap over current distributed control systems paradigms, paving the way for large-scale ubiquitous computing applications, ranging from environmental monitoring to building automation.

Several research initiatives, aiming at providing different design solutions for WSNs protocols, have recently emerged ([1], [2]; [3]; [4]). However, we believe that the use of standard technologies pushed forward by commercial manufacturers can speed-up a wider utilization of WSNs. In this context, the IEEE 802.15.4 protocol [5], recently adopted as a communication standard for Low-Rate Wireless Local Area Networks (LR-WPANs), shows up itself as a potential candidate for such a deployment. This protocol provides enough flexibility for fitting different requirements of WSN applications by adequately tuning its parameters, even though it was not specifically designed for WSNs.

The IEEE 802.15.4 standard specifies the two lower layers of the protocol stack, i.e. the Physical Layer (PHY) and the Medium Access Control Layer (MAC). The ZigBee [6] protocol focuses on the higher layer: both the Network (NWK) and Application (APL) layers. This standard also enables three network topologies – star, mesh and cluster-tree. In the star topology (Fig. 1a) the communication paradigm is centralized i.e. communications are always relayed through a central node - the ZigBee

Coordinator (ZC). ZigBee End Devices (ZEDs) act as sensor nodes and ZigBee Routers (ZRs) additionally feature message routing capabilities.



a) Star b) Mesh c) Cluster-tree

Figure 1.1: Examples of the different ZigBee network topologies

In the mesh topology (Fig. 1b) the communication paradigm is decentralized i.e. each node can communicate with any other node (directly, within its radio range, or indirectly through multi-hop). The cluster-tree network topology (Fig. 1c) is a special case of a mesh network where there is a single routing path between any pair of nodes and a distributed synchronization mechanism (beacon-enabled mode).

One of the most promising network architectures for WSNs is the Cluster-Tree architecture (fig. 1.c) since it offers several advantages as compared to the star or peer-to-peer architectures, such as: low energy consumption by adjusting the duty cycles (the duration of the activity and inactivity periods), lower and predictable end-to-end delays by the use of deterministic MAC protocol (e.g. Time Division Multiple Access). However, it may be prone to a significant risk of failure because of the single point of failure problem. In fact, if a parent node fails, all the child nodes that have a parental relationship with him will be isolated from the network, and will not be able to rejoin the network again, which prevents them from communicating normally with the other part of the network. The main objective of this work is to first to evaluate the default fault-tolerance mechanisms defined by the ZigBee standard and show their limitations for supporting real-time applications. Then, we propose an amendment to the ZigBee standard specification with the proposal of efficient fault-tolerance mechanisms for wireless sensor network operating under the cluster-tree topology, as defined by the ZigBee protocol. Using analysis and simulation, we show that our fault-tolerance proposals outperform the standard approach in terms of real-time properties since the proposed mechanisms are designed to minimize the recovery time from failures.

1.2. Research Contributions

The central objective of this work was to *improve ZigBee's native behaviour in the presence of a ZigBee router failure, by proposing fault-tolerance mechanisms that reduce network inaccessibility times and that are backward compatible with the IEEE 802.15.4/ZigBee standards.*

In order to achieve that goal, our work has been organized in four steps:

- Survey of the state of the art including an overview of Wireless Sensor Networks, the structure of the IEEE 802.15.4/ZigBee protocol stack, and related works on fault-tolerance mechanisms for WSNs.
- The identification and analysis of the impact of ZigBee Router faults/failures in ZigBee Cluster-Tree Networks.
- Proposal of two fault-tolerance approaches: a reactive approach and a proactive approach for ZigBee cluster-tree networks in order to reduce the time needed to recover from a failure. The proposed mechanisms are to be implemented within the child node in order for them to associate to a new parent in case of a failure or the deterioration of their current parent.
- Evaluation of our proposed mechanisms using simulations, and their validation by presenting the guidelines to be considered in the implementation.

Importantly, the proposed mechanisms ensure backward compatibility with the standard protocols, allowing nodes (ZEDs, ZRs) that implement the new fault-tolerance mechanisms to cohabit with other nodes (ZEDs, ZRs) that do not implement them. Moreover, the mechanisms should reduce inaccessibility times between a failure and its correction. Finally, the different mechanism proposals should be light-weight in order to reduce overall energy consumption and not to deplete the little resources of the nodes.

1.3. Specific Research Context

This project was carried out in the IPP-HURRAY! Research Group [Hurray], at the Engineering School (ISEP – Instituto Superior de Engenharia do Porto) part of the Polytechnic Institute of Porto (IPP), Portugal, under a research scholarship supported by the Portuguese Science and Technology Foundation (FCT). HURRAY stands for *HUgging Real-time and Reliable Architectures for computing sYstems*, which means that the group focuses its activity in the analysis, design and implementation of real-time and dependable computing systems. The IPP-HURRAY Research Group was created in mid 1997. Since then, it has grown to become one of the most prominent research groups in the area of Real-Time Systems and Real-Time Communications. Currently, it is the only Portuguese Research

Unit (as CISTER) rated as "EXCELLENT" by FCT, among a group of more than twenty units fitting the area of "Electrical and Computer Engineering".

This work has been developed within the context of the ART-WiSE (Architecture for Real-Time communications in Wireless Sensor networks) research framework [REF], aiming at the specification of a scalable two-tiered communication architecture for improving the timing and reliability behavior of WSNs. One of its major goals is to rely as much as possible on existing standard communication protocols and commercial-off-the-shelf (COTS) technologies – IEEE 802.15.4/ZigBee for Tier 1 and IEEE 802.11 for Tier 2 (Fig. 1.2).

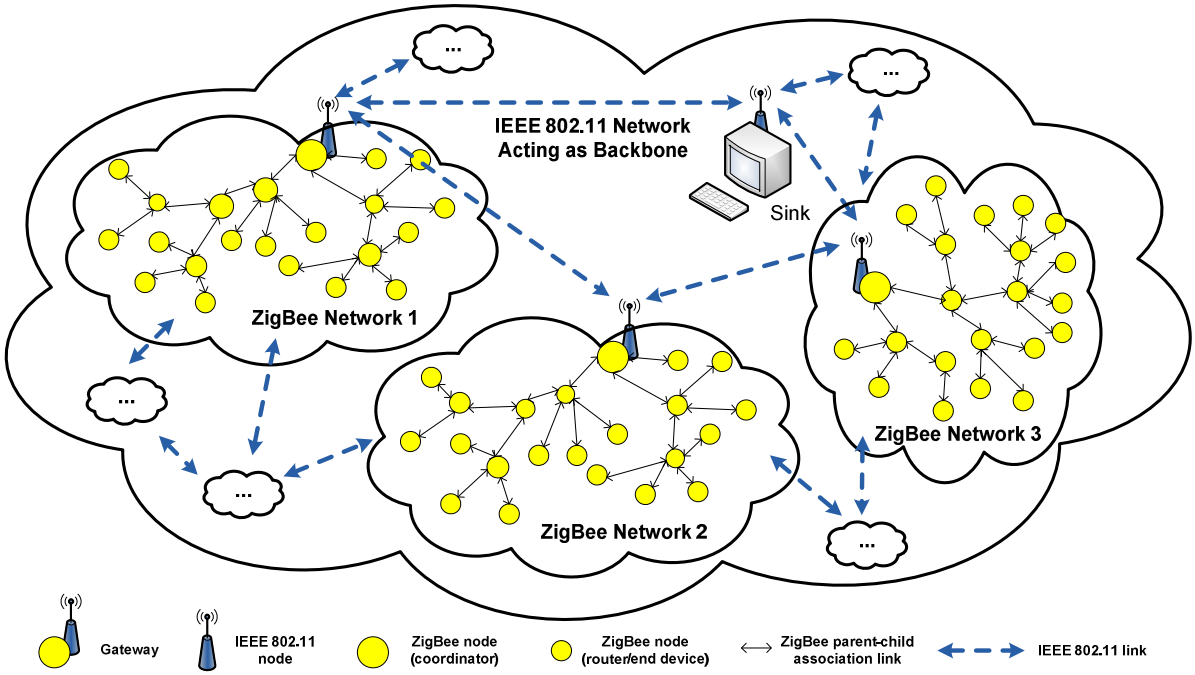


Figure 1.2: Example of the ART-WiSe two-tiered network architecture

In order to use ZigBee cluster-tree topologies at Tier 1, there is the need to achieve the required levels of Quality of Service (QoS), namely energy-efficient, reliable and real-time communications. In this context, it is of paramount importance to overcome the "single-point-of-failure" problems in ZigBee cluster-tree networks, in a way that critical applications will not suffer from unacceptably high network inaccessibility times. This may be achieved by providing the necessary hardware redundancy (i.e. in a way that each nodes "sees" two or more ZigBee Routers) and by using the proposed reactive/proactive fault-tolerance mechanism.

1.4. Report Organization

This report is organized as follows. Chapter 2 offers a state of the art of WSN and related fault-tolerance mechanisms that have been developed for these networks. Chapter 3 presents the most

relevant characteristics of the IEEE 802.15.4 / ZigBee protocol stack for WSN. In Chapter 4, we propose our reactive fault-tolerance approach for improving the recovery mechanism of the IEEE 802.15.4/ZigBee standard and motivate the proposal by presenting a comparative analysis of inaccessibility times in both cases. In chapter 5, we propose our proactive re-association mechanism that is useful for preventing link failures between associated nodes. Chapter 6 presents the performance evaluation of our proactive proposal using simulations. Chapter 7 presents the implementation guidelines to integrate our proposals in the IEEE 802.15.4/ZigBee protocol. Finally, Chapter 8 concludes the reports and presents discussions on lessons from this work, open issues, and future works.

Chapter 2

Fault-Tolerance Aspects in Wireless Sensor Networks

This chapter addresses some relevant fault-tolerance issues in Wireless Sensor Networks (WSNs). We start by outlining aspects on WSNs. Then, we define concepts and terminology related to fault-tolerance in these networks. Lastly, we present an overview on relevant contributions to the state-of-the art on improving reliability in WSNs.

2.1. General Aspects on Wireless Sensor Networks

2.1.1. The WSN paradigm

Wireless Sensor Networks are large-scale networks of embedded devices, where each device features sensing/actuating, data processing, storage, communication capabilities. In this context, large scale means a large number of network nodes that may be deployed in wide geographical regions, usually, within or in proximity of the monitored/controlled phenomena. Due to its large scale nature, individual WSN nodes must be cheap and maintenance-free, therefore having hard computational (e.g. processing, memory), energy and communication (radio coverage, bit rate) limitations.

The previously referred characteristics turns WSNs quite different from traditional wireless local area networks (WLANs), leading to a completely new communication paradigm. New protocols, architectures and mechanisms that tackle problems such as scalability and security must be devised.

Reliability and timeliness are major concerns within the context of this work. These attributes are particularly important when addressing applications with critical requirements, such as process

control, industrial automation, utilities transportation systems (e.g. electrical, gas, oil) or health care. The large-scale factor, the limited capabilities of the nodes and the multi-hop nature of communications are important impairments to fill those requirements.

2.1.2. On the resource constraints impact

As already stated; the large scale factor implies a very low cost per node (tending to less than 2$ a node), no maintenance (at least concerning most of the nodes) and long network lifetime (several years). Therefore, WSN nodes will have limited memory, processing, communication and energy capabilities.

Table 2.1 outlines some relevant performance indicators of currently available technologies.

	MICAz	**TelosB**	**FireFly**	**ScatterWeb**
Memory (total)	644K Bytes	1089K Bytes	---	512K Bytes
Communications	2.4 GHz	2.4 GHz	2.4. GHz	868 MHz
Bit Rate	250 Kbps	250 Kbps	250Kbps	19.2 Kbps
Processing	ATmega128L	16-bit RISC	8-bits microcontroller	---

Table 2.1: Main characteristics of some typical WSN nodes

All these constraints impact the design of WSNs, since the nodes can only store small programs and data. Thus, operating systems, communication protocols and applications must consider resource and energy consumption.

Additionally, all these factors play against the reliability of a WSN, motivating the need for robust, thus time and energy-efficient communication protocols.

2.2. Fault-Tolerance in WSNs

2.2.1. Faults Classification in WSNs

2.2.1.1. General fault classification

Wireless Sensor Networks are highly prone to faults and errors due to their restrictive capabilities and limited resources, i.e. energy, memory and processing. Once a sensor node detects a given error it will be considered as being in a faulty situation that prevents it from having a normal behaviour. The fault types in WSN vary with the type of error that may occur in the node.

In what follows, we present different fault classifications.

1. **Time-based classification:** faults can be differentiated based on their timing behaviour. Three fault types can be considered:

- ***Permanent***: a fault is said to be *permanent* if it occurs and it can not be recovered unless there is a human correction measure.
- ***Transient***: a fault is said to be *transient* if it occurs in the system and may be repeated several times.
- ***Temporary***: a fault is said to be temporary if occurs and then recovers without being periodically repeated in the future.

2. **Origin-based classification:** Errors can occur due to:

- ***Software errors***: programming errors contained in the embedded software of the node can lead to problems during the use of the nodes, potentially creating a complete failure of the system.
- ***Erroneous estimation***: the node measures a wrong value of the data of interest. This can lead to major problems in applications where the data consistency is very important. Software mechanisms can be implemented to detect this type of errors.
- ***Hardware errors***: any part of the node's hardware can fail preventing it from performing normal operation such as battery depletion, memory/CPU errors or transceiver failure. Our work will particularly deal with such kind of errors.

2.2.1.2. Classification in case of cluster-tree networks

For the particular case of cluster-tree networks (such in IEEE 802.15.4/ZigBee), faults can be differentiated based on the types of the nodes where the error occurs. Three cases can be differentiated:

- Root Failure (in ZigBee, PAN Coordinator Failure): This is the most critical fault that prevents the network form working normally, which may also prevent the whole network from communicating.
- Router Failure (In ZigBee, ZigBee Router/Coordinator Failure): this error prevents the cluster, which the ZR is head of, and all its descendant clusters to communicate with the rest of the network. This failure will block only a part of the network and thus may be less critical than the Root failure. The criticality of this failure type increases when the broken router is closer to the root, since the number of affected descendant clusters increases.

- Node Failure (in ZigBee, End-Device Failure): in this case, the network will not be able of sensing the region where the end-device is located in. It is, usually, not a critical failure unless it occurs in applications where the sensed data is crucial such as medical applications. This work will not address this failure type, but will focus more on the two previous failures that are more critical for WSN applications.

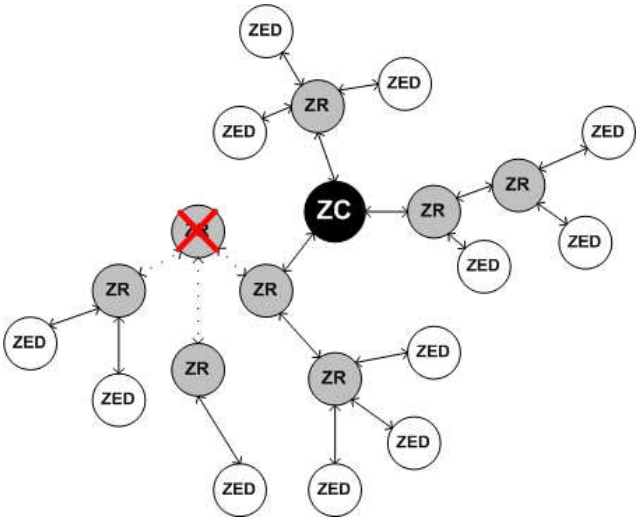


Figure 2.1: Example of a ZigBee Router failure

In a cluster-tree network, an error occurs when a child device (router or node) looses connectivity with its parent, which results in a total link failure. The main reasons are:

1. Problems on the wireless link between the parent and the child device that mainly result from high interference rates within the operating frequency band. This may occur at different moments:
    - *During a data transfer*: In this situation, the data being transferred is lost and the data frame needs to be re-sent.
    - *During the synchronization*: In this case, the device does not receive beacon frames and cannot properly synchronize with its parent.
2. The parent may be experiencing hardware, battery or software problems that prevent him from performing normal operation.

2.2.2. Related Work on Fault Tolerance in WSNs

Several research works have focused on the proposal of fault-tolerance mechanisms for wireless sensor networks to improve their reliability in terms of data transfer. In this section, we provide a survey on some existing works and discuss their limitations.

2.2.2.1. Improving reliability in Wireless Sensor Networks (WSNs)

Reliability is an important and challenging issue in wireless sensor network due to the inherent severe constraints of sensor nodes coupled with the sensitivity of many applications to the loss of data or to the inaccessibility of the network during a time period. Several research works have been conducted for proposing fault-tolerance mechanisms to improve WSN reliability.

In Reference [7], the authors have presented the limitations of WSN communications and their need for a certain level of reliability. They have also overviewed the different characteristics and design decisions to be considered when implementing a fault-tolerance system for WSNs, finally they depict the main characteristics of an ideal fault-tolerance system. In Reference [8] the authors have presented an overview of the main problems of reliability, have reviewed some existing fault-tolerance mechanisms and have discussed open issues related to the reliability in WSNs. In [9], the authors have proposed a new approach that consists in dedicating a reliability layer for WSNs, this additional layer would supervise data transfers and use multicast-like routing protocol based on Multipath-On Demand Routing [10]. In case of the next hop failure and in order to send the information, the node will broadcast the information to all devices in vicinity in order to reach the destination.

On the other hand, some other works such as [11] and [12] have proposed mechanisms for detecting the occurrence of a fault in the network by using a *network management system* to supervise the behaviour of the nodes, and thus allowing the network manager to take the necessary corrective measures, in case of errors.

However, the aforementioned proposals do not offer any mechanism to tackle the total failure of a node but simply provide solutions to improve the network overall reliability or to supervise its operation.

The evaluation of the real impact of a node failure in a WSN has been studied in [13]. The authors have designed and developed a simulation tool based on a Bayesian model that presents the impact of a node failure on a mesh network. In addition, they assessed the reliability improvement when using the hardware redundancy technique as a simple fault-tolerance system noticing that hardware redundancy will increase the overall reliability of the network but not sufficiently enough to justify the cost of a global hardware redundancy. This latest work points out the need to fault-tolerance systems in the situation of vulnerable WSNs.

2.2.2.2 Existing Fault-tolerance Mechanisms for WSNs

Other research works have dealt with the fault problems in WSN, particularly the problem of a hardware failure of a node in the network. Most of these proposals are topology-based fault

tolerance mechanisms that use specific topology adaptations in order to indirectly implement fault-tolerance mechanisms.

In [14], the authors have focused on the design of specific grid-like (k-vertex connected) network topologies for ad-hoc wireless networks (including WSNs) that have a fault-tolerant characteristic. This objective was achieved by implementing the *Fault-tolerant Local Spanning Subgraph* (FLSS) algorithm establishing the most energy-efficient routes and providing the best performances by using both a centralized and a decentralized algorithm that will calculate the relative cost of each route in terms of energy and performance and elect the best one to be used.

Other research works (e.g. [15]) have proposed fault-tolerance mechanisms specifically for Cluster-Tree WSNs (different from the ZigBee cluster-tree model), , that is based a consensus approach. The protocol proposed in [REF GUPTA] acts in two phases: a *fault detection* phase followed by a *fault recovery* phase.

The main objective of the detection phase is to determine if there is really a fault in the network. For that reason, a consensus mechanism is used between the different gateways, if they have coherent information about a node failure. That node is considered faulty and the recovery process begins. The authors claim that this consensus method maintains the synchronization of the network, however no proof of that has been presented. Gateways maintain the current state of other gateways by the means of "*state updates*" corresponding to the other gateways and a table summering all that information. When a failure is detected, the failure recovery procedure is started. To do so, the *backup gateway* that centralizes the information about the sensor nodes in vicinity, will take the decision about which sensors are to be attached to it.

This proposal presents severe limitations that contradict the inherently restricted resources in WSNs since the signalling process imposes an important overhead, consumes a big amount of memory in order to maintain the tables and makes an extensive use of the CPU resources to perform the appropriate computations.

Another protocol supporting a fault-tolerance mechanism called LEACH has been proposed in [16]. This protocol indirectly supports fault-tolerance mechanisms by controlling the cluster-head section in a given cluster-tree network. In fact, LEACH is a schedule-based MAC protocol based on a dynamic clustering technique that aims to evenly distribute energy between nodes. It 'elects' a cluster head on a random base in each round. This ensures that there is no permanent cluster head, and by that means it minimizes the possibility for a cluster-head to be faulty since it is not a static one. Although originally designed to improve energy issues, LEACH seem to be a suitable approach to support fault-tolerance schemes. LEACH can be adapted for WSN fault management by changing the cyclic update

to an “on event” update; that would trigger a new cluster-head update round. LEACH achieves two main objectives: it reduces energy consumption and implements an elementary fault-tolerance technique.

In this current work, we deal with a different reliability problem as compared to the previous works. In fact, we consider the particular case of fault-tolerance in ZigBee cluster-tree networks, which was not addressed in the literature, yet to our best knowledge.

2.2.2.3. Erroneous Estimation Detection

It may happen that nodes perform an erroneous estimation of the sensed data. This leads to incorrect data sent by the nodes and may be associated to a fault situation. It is important to detect the situations where such errors occur and potentially find out the appropriate methods to correct them. Several research works have assessed this type of problems. In [17] the authors have proposed a correlation function in order to find out the suitable relation between the different sensed data sent by the nodes and find out the erratic one that will not fulfil the correlation condition. This centralized solution is implemented in the data sink and allows the network manager to detect the faulty nodes. This proposal does not offer a mechanism to correct this type of errors.

2.2.2.4. Software Faults Detection

Embedded software for WSN nodes is complex and difficult to debug, it may often happen that software problems occur during the runtime of the network, possibly preventing it from performing normal operations. The sensed data can then lead to software errors such as: deadlocks, live locks, stack overflow or application errors.

Specific solutions have been developed in order to tackle this specific type of faults. *NodeMD* [18] provides both reactive and proactive approaches that support both a fault notification and diagnosis system and can offer, in some specific situations, a correction mechanism.

2.3. Conclusion

In this project, we addressed the specific case of ZigBee cluster-tree WSNs, which are prone the single point of failure problem in the ZigBee routers (cluster-heads). Our aim is to improve on the ZigBee’s native orphan realignment mechanism by reducing inaccessibility times; as it will be outlined in Chapters 4 to 7.

Chapter 3

The IEEE 802.15.4/ZigBee Standards

The most common Wireless Sensor Networks rely on the IEEE 802.15.4 / ZigBee standards. In this chapter we present an overview of the most relevant aspects of these standards since that understanding the standards mechanisms and their limitations is helpful when implementing fault tolerance mechanisms.

3.1. The IEEE 802.15.4 Standard

3.1.1. General Overview

The lower layers of the protocol stack for WSN has been standardized by the IEEE part 15.4 (Wireless Medium Access Control (MAC) and Physical Layer (PHY) Specifications for Low-Rate Wireless Personal Area Networks (LR-WPANs)) of the 802 work-group on the 12th of May 2003. The standard defines the protocol and interconnection of devices via radio communication in a PAN. The standard uses carrier sense multiple access with collision avoidance (CSMA/CA) medium access mechanism and supports star as well as peer-to-peer topologies. It defines frequency uses, modulation, spreading techniques and other functions that are performed by the PHY layer. It also defines the procedures of medium access that are located in the MAC sub-layer.

3.1.2. Physical layer

3.1.2.1. Frequencies and Modulations

The physical layer is the lowest layer of the protocol stack and is responsible for frequency selection, carrier frequency generation, signal detection, modulation, and data encryption. WSNs

operate in the 433 MHz and 868 MHz frequencies bands for Europe and Japan and the 915 MHz frequency band in the United States of America. In addition, the 2.4 GHz band is proposed for higher data rates in short range. To each frequency band are associated specific modulation/spreading schemes and data rates as shown in table 2.2.

Physical (Mhz)	Frequency Band (Mhz)	Modulation	Bit Rate (Kbps)
868 / 915	868 – 868,6	BPSK	20
	902 – 928	BPSK	40
2450	2400 – 2483,5	O - QPSK	250

Table 3.1: Frequency specifications in the PHY layer of the IEEE 802.15.4 standard

27 different channels within these three frequency bands can be used: 16 in the 2450 MHz band, 10 in the 915 MHz and 1 in the 868 MHz band. Two different spreading techniques can be used during transmissions: DSSS (Direct Sequence Spread Spectrum) and FHSS (Frequency Hopping Spread Spectrum) that are the two spreading techniques typically used in wireless networks including WSNs.

3.1.2.2. Physical Layer tasks and procedures

The PHY layer is responsible of the following tasks and procedures:

- *Activation and deactivation of the radio transceiver:* This feature enables the transmitter to have 3 different operation modes: ON, OFF and SLEEP and switching periodically from one state to another on request form the MAC layer. The aim of this function is to reduce power consumption.
- *Energy Detection (ED) within the current channel:* Consist in listing to the channel for a given period of time (8 symbol periods as specified in the standard) and detecting if there is any transmission within in it without any identification or decoding. If important energy is found in the channel that means that it used or highly interfered. The result of this measurement is sent to the network layer or used in the CCA procedure that is explained here under.
- *Link Quality Indicator (LQI) for the received packet*: A metric of the strength or quality of a received packet. This measurement can be done using receiver ED, a signal to noise ratio estimation or a suitable combination of the two methods. The result of this indicator is sent to the higher layer where it will be used and stored (in taking routing decisions for example).
- *Channel Clear Assessment (CCA) for CSMA-CA:* this procedure lasts 8 symbol periods and tests the radio channel to know if it is busy or idle. It can be performed in 3 different modes: *Energy*

detection mode: It can be considered as the basic mode since that if the ED is above a given threshold, then the channel is reported as busy. *Carrier sense mode:* If any signal corresponding to the IEEE 802.15.4 modulation and spreading characteristics is detected on the channel it is considered as busy. There is no use of any threshold. *Carrier Sense with ED threshold*: Is similar to the previous mode but introduces the fact that the energy is compared to a threshold in order to know if the energy is still inferior than a given limit that enables good transmission conditions.

- *Channel frequency selection:* Upon receipt of a specific request form the higher layer, the PHY must be able to tune the transceiver to the appropriate channel.
- *Data transmission and reception*: On request from the higher layers, the PHY activates its TRX and enables transmission and reception of data using the adequate RF parameters (Frequency, modulation, spreading…), on receipt of information it sends it to the higher layers for processing.

3.1.3. Data Link layer

The data link layer as it is described in the OSI standard is divided in two layers, the data link sub-layer (DLL) and the MAC sub-layer. In the following we will be focusing on the MAC sub-layer since that in the case of WSNs it has a bigger impact than the DLL on energy consumption and real time issues.

In order to access to the wireless medium, the nodes can use different medium access methods, such as:

- *Scheduling based protocols:* This class of protocols is based on the presence of a central node that schedules the transmissions of the nodes that are connected to it using for example TDMA access method. To each child node is associated a time slot duration in which it can communicate with the central node. Nevertheless, this method has strong limitations regarding the number of nodes that can be connected at the same time to a node, moreover this method is very error prone since that in case of the failure of the central node (that schedules the communications between the nodes), no more communications will be possible between the nodes but also it presents limitations in terms of nodes mobility and requires strong time synchronization between the child nodes and the central station.
- *Collision free protocols:* This includes both the FDMA and CDMA medium access methods. Since that in FDMA, distinct transmissions use different frequencies and that in CDMA, distinct transmissions use different spreading codes. This class of protocols in usually used when transmitting information between, different clusters of the network.

- *Contention based protocols:* this class of protocol uses the CSMA/CA channel access method that is inspired form the 802.11 channel access method supporting the use of the channel access protocol and the support of contention free and contention based periods but eliminating the RTS/CTS mechanism in order to adapt to the low data rate characteristic of WSN where collisions are more luckily to occur.
- Other medium access protocols for WSN have been proposed, mainly LEACH [16], S-MAC [19], DMAC [20], DB-MAC [04].

In the following we will be considering the particular use of the CSMA/CA mechanism as it is proposed by the IEEE 802.15.4 mechanism as the only channel access mechanism.

3.1.3.1. CSMA/CA operational Modes:

CSMA/CA functions in two different operational modes:

- *The beacon enabled mode:* the central node (called coordinator) issues a beacon frame on a regular basis to its child devices in order to synchronize their transmissions. The interval of time between two beacon frames is called a superframe, transmissions occur during that period of time.
- *The non-beacon enabled mode:* there is no transmission of beacon frames and devices can send their frames using the classic unslotted CSMA/CA transmission scheme and there is no use of the superframe structure.

3.1.3.2. The Superframe Structure:

The duration between two beacon frames is called the Beacon interval (BI), it can be divided into two periods, one active period and one inactive period. The active period is called the superframe duration, which can also be divided in two parts, a contention access period (CAP) and a contention free period (CFP). The durations of both the beacon interval (BI) and the superframe duration (SD) are determined respectively by the beacon order (BO) and the superframe order (SO) and the following relationships:

$$BI = aBaseSuperframeDuration \cdot 2^{BO}$$

$$SD = aBaseSuperframeDuration \cdot 2^{SO}$$

$$Where\ 0 \leq SO \leq BO \leq 14$$

The general aspect of the superframe is given in figure 3.1.

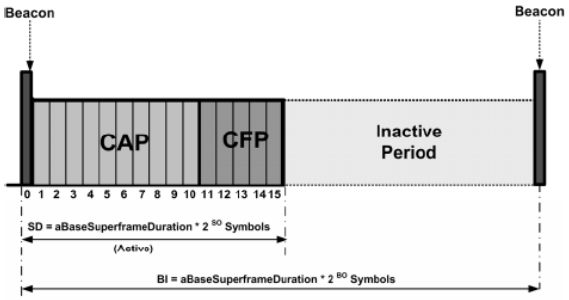


Figure 3.1: The Superframe Structure

The CFP can be subdivided into different Guaranteed Time Slots (GTS) and can occupy more than one time slot.

3.1.3.3 CSMA/CA mechanisms:

The CSMA/CA mechanism functions in two different modes:

- *Slotted CSMA/CA:* used in the beacon enabled mode
- *Unslotted CSMA/CA:* used in the non beacon enabled mode

A summary of the different operational modes is given in figure 3.2:

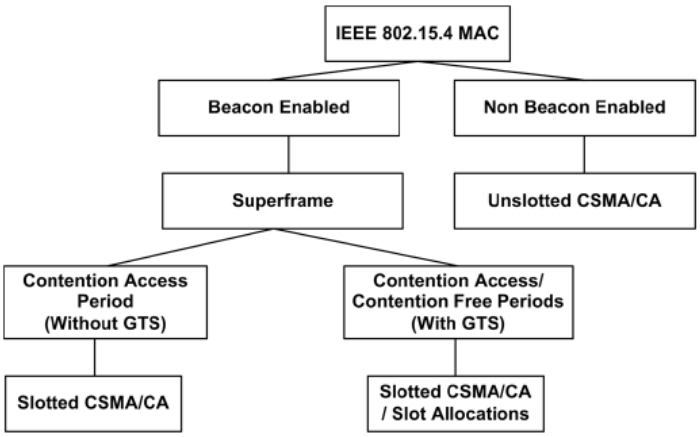


Figure 3.2: The different operating modes of the MACsub- layer

3.2. The ZigBee protocol stack

3.2.1. General Overview

The ZigBee protocol stack has been developed by the ZigBee alliance that is a grouping of industrials aiming to create standard WSN solutions. Its main interest is the NWK and APL layers that closely interact with the lower MAC layer. More specifically it defines the application support sub-layer

(APS), the ZigBee device objects (ZDO), ZigBee device profile (ZDP), the application framework, the network layer (NWK) and ZigBee security services.

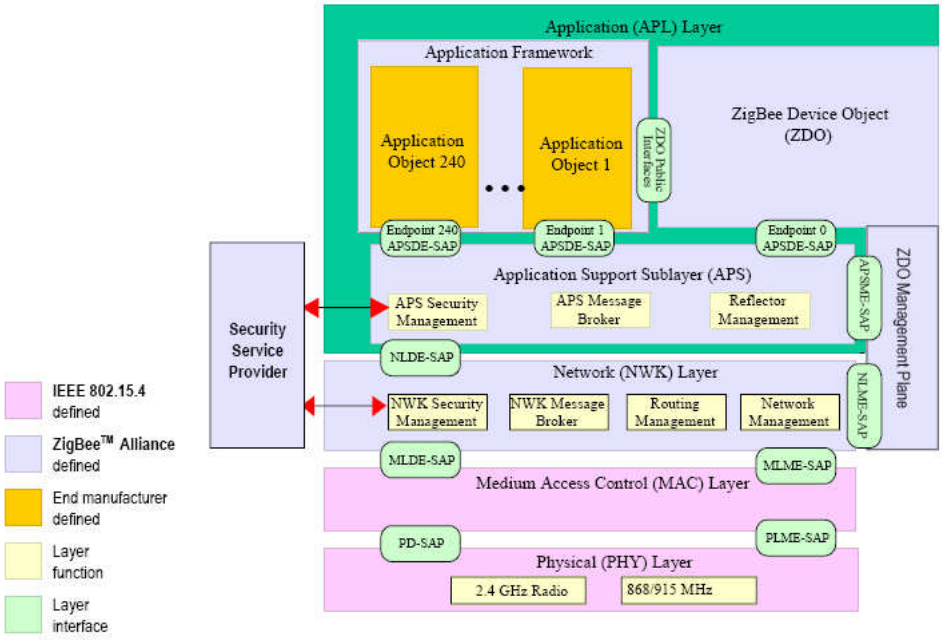


Figure 3.3: A detailed overview of the IEEE 802.15.4/ ZigBee protocol stack

Figure 3.3 presents the IEEE 802.15.4/ZigBee protocol stack with further details about the components of each ZigBee layer and the relationships between them. It also summarizes the names of different service access points used in the protocol stack.

3.2.2. Network Layer

The network layer is required to provide functionality to ensure correct operation of the IEEE 802.15.4 MAC sub-layer and to provide a suitable service interface to the application layer. The main role of the NWK layer is to implement define the NWK layer frame formats, ensure correct data transfers between the networked nodes through adequate routing protocols, manage the association and disassociation processes through predefined mechanisms. The NWK layer also maintains a database of managed objects known as the Network Information Base (NIB), this data base contains operational information about the network and the neighbour table that contains information about the neighbouring devices in the network.

Moreover, the NWK layer saves some persistent data during the operation of the node that will be retrieved automatically when the device restarts such as: its current network address and the used stack profile. This mechanism can be useful when implementing management or fault tolerance mechanisms that need to have specific information rapidly after a node failure.

The following sub-sections present the different network architectures and the way to establish such networks in the ZigBee context.

3.2.2.1. Network topologies

The Star Topology

Star networks are centred on a PAN coordinator that communicates with the devices (Fig 3.4). Communication between child nodes is not possible. PAN coordinators in this topology are usually mains powered since they have to be continuously on to receive the information sent by the child nodes. That's why it appears that this topology is more suitable for applications such as: personal health care, peripherals communicating with a computer or in toys and games.

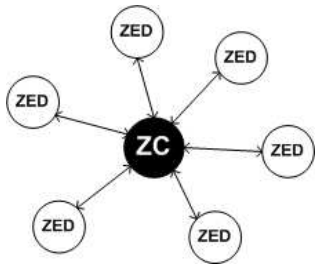


Figure 3.4: The Star Topology

The Mesh Topology

This network topology is formed by a PAN coordinator that ensures the control of the network and a set of routers and end devices that communicate together on a peer-to-peer fashion (Fig. 3.5). RFDs can only communicate with a unique FFD to which they are associated, that FFD is called their parent. On the other hand, FFDs can communicate between each other without any problem. In this topology, information is relayed from node to node throughout the network until it reaches the data sink.

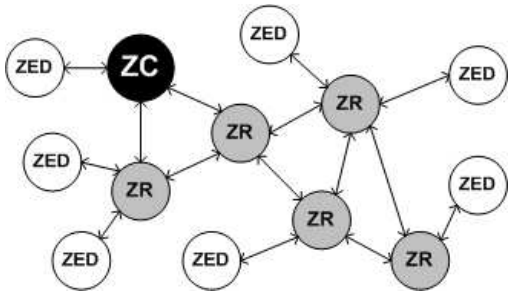


Figure 3.5: The Mesh topology

The Cluster Tree Topology

This network topology is a specific case of the mesh topology but has some particular features. It is also composed of a PAN coordinator to which connect routers and end devices (Fig. 3.6). Each router can create a cluster and become the cluster head. Other routers that are connected to him can also become cluster heads by electing themselves and creating a cluster to which will be connected other routers and devices. Clusters are identified by an Id. The final structure will be a tree where the root will be the PAN coordinator and the clusters will constitute the branches and the end devices will form the leaves.

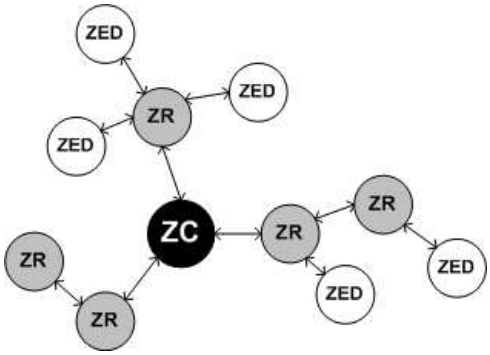


Figure 3.6: The Cluster Tree topology

3.2.2.2. Device maintenance

All the ZigBee devices must be able to join or leave a network. In addition, ZigBee routers or coordinators must permit devices to join and leave the network by association, participate in logical network address assignment and the maintaining a table of neighbouring devices.

Some of the functionalities are presented in the following:

- ***Establishing a new network:*** This can only be initiated by coordinators that are not currently member of a network. It consists in creating a new network
- ***Joining a network:*** can be done using different association procedures:
  - *Using the MAC layer association procedure*

    This procedure must take place in both the parent and the child device. From the child device point of view the following tasks are executed:

    1. The network layer issues a primitive asking the MAC to perform an active or passive scan.
    2. Each beacon frame received during the listening period with a payload of zero length will generate an MLME-BEACON-NOTIFY.indication to the network layer containing addressing information and association possibilities.

3. Only the devices that are not associated to a network can attempt to associate. The network layer searches on the neighbour table if there exists a suitable parent device.
4. If there is no suitable parent in the table an error message is send to the management layer, if there is more than one suitable parent, the device chooses the one with the minimum depth from the ZigBee coordinator, if there are more than one, the device is free to choose.
5. When a suitable parent is found, an association command is sent to the MAC layer containing addressing information of the parent found in the “neighbour table”.
6. If an association failed, it can be because the parent we are attempting to associate to, has already reached the maximum associated routers number it can handle. In that case, the NWK layer can try to connect as an end device. If that association failed, the device can connect to another router in his “neighbour table” (if it exists). If the association is successful, a 2 bytes short address is acquired to transmit over the network.

From the parent device point of view:

1. The parent verifies that the device is already registered to his network by comparing 64-bits logical address existing in his “neighbour table” with the device’s address. If a match is found, the parent assigns a 16 bits address to the new device if an address is available in the address space.
2. When the request is granted, a new entry for the child device is created in the “neighbour table” of the parent.
3. Shall there be any error, it will be reported to the higher or lower layer using the appropriate primitives.

- *Through the NWK join procedure*

  This procedure gives the opportunity for a node that has lost all connectivity with the network. It allows a device to rejoin a network that does not allow new devices to join. There is no use of the MAC association procedure.

  The Child procedure:

  1. Is initiated by the NLME_JOIN.request primitive (a parameter can specify of the device wants to join as router).
  2. MAC layer makes a passive or active scan of the channels.
  3. Test the received beacon and registration of information in the “Neighbour Table”.

4. On scan complete, choose the best parent device and rejoin.
5. If rejoin was unsuccessful, the NLME shall find a new suitable parent to join to in the table.

The Parent procedure is similar to the one described earlier.

- *Through orphaning*

  In this case, a device is directly added to the network by a previously designated parent device. Only ZigBee coordinator/router may initiate this procedure, otherwise the higher layer would be notified by an invalid request notification.

  1. Parent verifies if the specified device already exists in the network by searching in the neighbour table.
  2. If not match is found: allocate a 16 bits network address to the new device if possible.
  3. Now that the parent has created a link with the child device, the child should complete the establishment of the parent-child relationship by initiating the orphaning procedure.

- ***Leaving a network:*** A device may leave the network. This can be done either when the device initiates its own removal; in that case it sends a leave command frame to its parent and waits for its response**.** The parent device can also remove a child device from the network; in this case, on request from the higher layers the parent's NWK layer sends a leave request to the child device to disassociate and waits for the confirmation from the target device.

More information about these specific mechanisms and other mechanisms implemented in the NWK layer are found in Annex A.

3.2.3. Application Layer

The APL's role is to implement the effective functions of the sensor node. It integrates the roles defined by the manufacturer in order to achieve the goals it was designed. For example, in the case of a node used to control a light system, the APL will implement the different sensing and control functions needed for the execution of these different tasks in addition to others.

The APL consists of the APS (Application Support Sub-layer), the ZDO (ZigBee Device Object) sub-layer and its associated management plane and the manufacturer's defined application parts and are located in the ZigBee application Framework.

The APS's major tasks are:

- Maintaining tables for binding, that is, the ability to match two devices together based on their services and their needs
- Forwarding messages between bound devices
- Group address definition, removal and filtering of group addressed messages
- Address mapping from 64 bit IEEE addresses to and from 16 bit NWK addresses
- Fragmentation, reassembly and reliable data transport

The ZDO is responsible for:

- Defining the role of the device within the network (e.g., ZigBee coordinator or end device)
- Discovering devices on the network and determining which application services they provide
- Initiating and/or responding to binding requests
- Establishing a secure relationship between network devices

The Application Framework hosts the applications of the ZigBee device; it controls and manages the protocol layers of the device and initiates standard functions.

3.2.4. The management layers

As pointed out in section 2.1.4 the protocol stack also contains three management layers that are implemented perpendicularly to the previously presented functional layers. Theses layers' role is to help in the sensing task and to improve the energy efficiency of the overall network by coordinating collectively the individual behaviour of the network's nodes.

- *The power management plane* manages how the sensor node uses its power, by for example turning off the receiver when no incoming messages are expected; in fact, turning off the receiver can lower the overall energy consumption. The node can also, when it is on low energy level, broadcast to its neighbours that it is not more capable of routing messages so that the remaining energy is devoted for sensing and other primary operations. It is interesting to denote that this last feature may be considered as a preventive fault management mechanism.
- *The mobility management plane* is responsible of taking track of the node's localization changes. It enables, for example, the creation of a new path to the data sink and detects the presence of the node's new neighbours.
- *The task management plane* coordinates the tasks between the different nodes of the network in function of their available power levels i.e. the nodes with the more power

available are asked to do more tasks, as a consequence, the network can live longer since power consumption is evenly distributed between the nodes; this may be seen as an energy balancing system.

3.3. Conclusions:

In this chapter we provided a general overview of the IEEE 802.15.4/ZigBee protocol stack. We reviewed the different properties of each layer and especially the network layer that provides useful mechanisms and procedures for implementing fault tolerance mechanisms. In the next sections we present both a reactive and a proactive re-association mechanism that offer fault tolerance features in order to tackle the limitations of the standard on this point.

Chapter 4

A Reactive Fault-tolerance Mechanism for ZigBee Cluster-Tree WSNs

In this chapter we present a reactive re-association mechanism that enables faster rejoin of orphan devices to the network. We start by presenting the considered system model and expose the standard orphan realignment procedure. Then we present details about the proposed mechanism and perform timing analysis that points out the performance gained from this novel mechanism.

4.1. System Model

In our analysis, we consider the ZigBee cluster-tree topology as presented in Fig. 4.1.

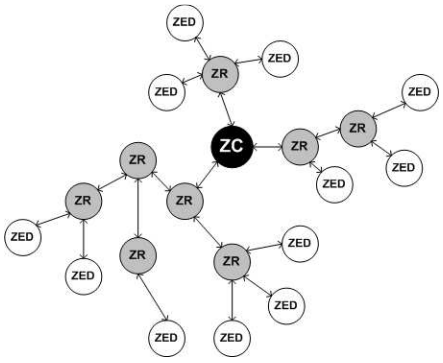


Figure 4.1: Generic Cluster-Tree network model

Like in any tree network, the cluster-tree topology contains a special node called root or *PAN Coordinator*, which identifies the entire network. In addition, in a tree network, some special devices may have the ability to allow the association for other nodes. These nodes are called *ZigBee routers*

(or *ZigBee Coordinators*). End devices with no routing ability are called *child nodes*. Both *child nodes* and *routers* are assumed to have sensing capabilities and are referred to as *sensor nodes*.

The nodes have a parent-child relationship between each other. Each node (except the *PAN coordinator*) has to associate to another node that will become its parent. When the association process is successful, the child device is said to have joined the network.

A cluster-tree network is composed of a number of clusters that contain routers and end devices. Each cluster is managed by a unique ZigBee router, which act as a cluster-head that coordinates the communication within its cluster and with its parent cluster. The different clusters of the network are identified using a unique cluster identifier. Inside a cluster, communications are possible between the parent and its child devices; however, direct communication between two children is not possible even if they are in physically in range of each other.

Both the PAN Coordinator and ZigBee Routers are assumed to send beacon frames to synchronize the nodes of their cluster. Collision-free beacon frame scheduling approaches that maintain the network synchronization have already been proposed in [21].

4.2. Fault Model and Solution Approaches

In the current study, we only consider the case of the single-point-of-failure problem. This means that we only consider the case of ZR failures, but not those of ZED neither of the ZC, as illustrated in Fig. 4.2. The reason for not considering the two latter cases is that the failure of a ZED does not have any impact of the network normal operation, and the ZC failure requires the re-configuration of the whole network by electing a new root for the tree, which is out of the scope of this work. The latter case may be considered in a future work.

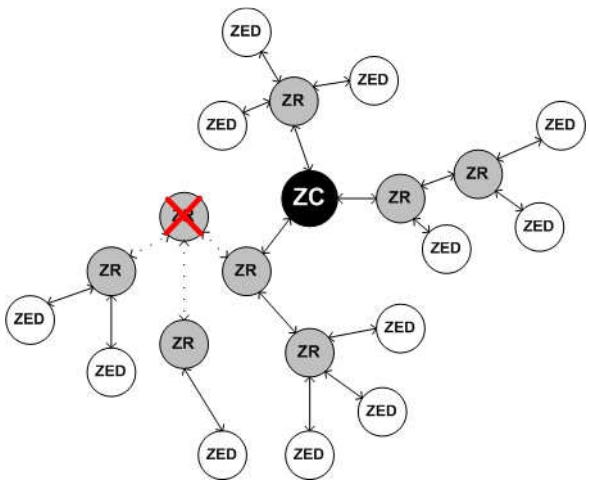


Figure 4.2: Single point of failure in Cluster Tree WSN

The single-point-of-failure can induce either a total link failure between parent-child devices or a link quality degradation between two associated devices.

In case of total link failures, the child node will neither be able to exchange any data nor receive beacon frames from its parent. In this case, the child node would decide that its parent is dead.

In order to recover from the failure of its direct parent, a node should run an appropriate recovery mechanism. In this Chapter, we propose the **reactive approach** that extends and improves the existing orphan realignment procedure of the IEEE 802.15.4/ZigBee standard protocol stack in terms of inaccessibility times and energy-efficiency.

In addition to the link failure situation, two devices may present frequent (i.e. temporary or transient) loss of synchronization; this happens when the beacon frames are received on an erratic way. The most common reason of this behaviour is the bad quality of the wireless parent-to-child link that may be due to electromagnetic interference (EMI), the presence of obstacles between the nodes or to the node's mobility. In addition, it can be due to faults on the parent side (e.g. the parent can be losing power).

In this situation, it is better to switch to a new parent presenting better performances for the following reasons:

- Saving energy
- Improving the reliability
- Reducing end-to-end delays

It can also be envisaged that a node anticipates the link failure by associating to a new parent before the occurrence of the link break under certain conditions. Such a mechanism is called a **proactive fault-tolerance approach**, which will be presented in Chapter 5.

As stated earlier, for each class of faults, a different fault-tolerance mechanism is proposed. Each method corrects a particular fault situation since there is no unique solution for different problems: "*To different diseases, different cures*"

4.3. IEEE 802.15.4/ZigBee Standard Mechanisms

In this section, we present the most relevant standard mechanisms of the IEEE 802.15.4/ZigBee protocol stack for the design of fault-tolerance mechanism specifically for the standard protocol. These mechanisms must be taken into account in the design process of the fault-tolerance mechanisms to ensure backward compatibility with the standard specification.

The following standard parameters are useful in the design process of the reactive and proactive re-association mechanisms:

a. **The Network Information Base (NIB):** it contains the information about the network status. It particularly includes a field called *nwkNeighborTable* that contains information about the neighbouring devices of the current node. This Neighbour table (as defined in Table 3.43 of [2]), is implemented in all the devices of the network and possess an entry for each neighbour device including the following mandatory fields: *extended address*, *network address*, the *Device type* of the neighbour, the *Relationship* between the current device and the neighbours, a *transmit failure* indicator (indicating if the previous transmissions of the device were successful or not, higher values indicate more failure rates) and a *Link Quality Indicator*, *RxOnWhenIdle* (indicates if the neighbour receiver is enabled during idle portions of the CAP). Refer to Annex B for more information about the entries of the *nwkNeighborTable*.

   In our approaches, this NIB will be used, with some minor add-ons, in order to store the information about potential new parents if the current parent of the device fails.

b. **The rejoin through the NWK rejoin procedure:** This procedure can be launched by all the devices of the network (except the PAN coordinator) and gives the opportunity for a node that has lost all connectivity with the network to rejoin it. In this case, there is no use of the MAC association procedure. This mechanism can be useful to initiate a correction process after the detection of a fault between the parent and the child device in order to connect to a new parent.
c. **The rejoin through orphaning procedure:** is a basic fault-tolerance mechanism that is specified in the standard and that will be explained in the next section.
d. **The logical addressing scheme:** it can be very useful in determining the pool in which the devices are located and by that means a device can prefer to associate to a new device rather than another taking into consideration the address class of the target device. Section 5.2 of annex A presents further information about the logical addressing scheme.
e. **Broadcast and multicast communications:** Broadcast and multicast communications can be exploited to propagate information related to the nodes state to a management entity that will be responsible of monitoring the network general status and eventually provide information to the network operator about the general status of his equipment and communications.
f. **The persistent data in the network:** This information can be used to recover from failures since that information can always be gathered. By choosing and using the correct persistent information, the recovery phase duration can be decreased. Section 7 of annex A presents the different persistent data in the network.

The *reactive approach* will provide the child node with an appropriate behaviour to react to a brutal link failure with its parent in order to re-associate to a new parent. On the other hand, the proactive mechanism will prevent the link failures and will make the child node associate to a better parent node before the failure of the link. The latter approach will prevent the network being inaccessible. Nevertheless, fault-tolerance mechanisms always introduce costs in terms of network load, battery life and memory and CPU usage. One of the design objectives of our proposed mechanisms is to minimize the overall consumption of these parameters and to ensure backwards compatibility with other nodes that do not support the proposed mechanisms.

4.4. The Standard Orphan Scan Procedure

4.4.1. Overview

The IEEE 802.15.4/ZigBee standards define an elementary recovery procedure in case of loss of synchronization or connectivity between a parent and its child node referred to as the "*orphan scan procedure*" or "*re-joining through orphaning*".

The child device detects the loss of connectivity/sychronization with its current parent when it does not receive *aMaxBeaconLoss* consecutive beacon frames from its parent. Then, it will initiate the orphan scan procedure, which behaves as follows:

On the child node side

- The NWK layer requests the MAC sub-layer to perform an orphan scan over the rest of the channels given by the *ScanChannels* parameter.
- The device defines a logical set of channels and sends an orphan notification on each one of them then waits for *aResponseWaitTime* duration for a coordinator realignment command from a coordinator in the network.
- During the orphan scan, the device discards all the received frames that are not *coordinator realignment* MAC command frames.
- If the orphan scan was successful, (the child has found its parent) the NWK layer informs the higher layer of this success to take specific actions.
- If the orphan scan is unsuccessful, the NWK layer informs the higher layers about this result to take specific actions.

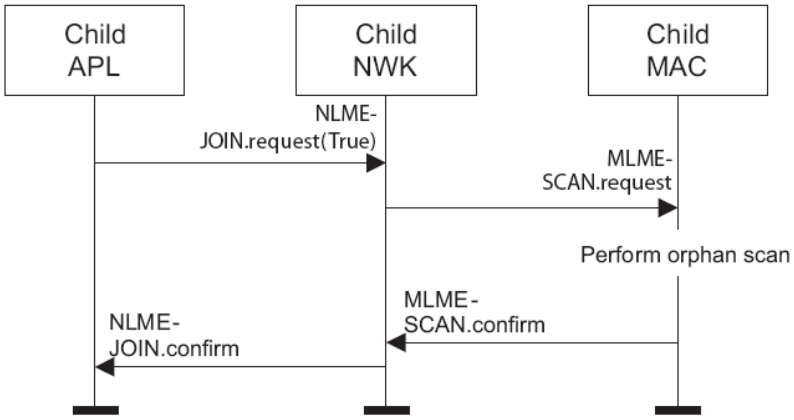


Figure 4.3: Child procedure for the join or re-join a network through orphaning

The standard does not specify the appropriate procedure when it is unable to determine its depth in the tree, which can prevent it from performing normal operations.

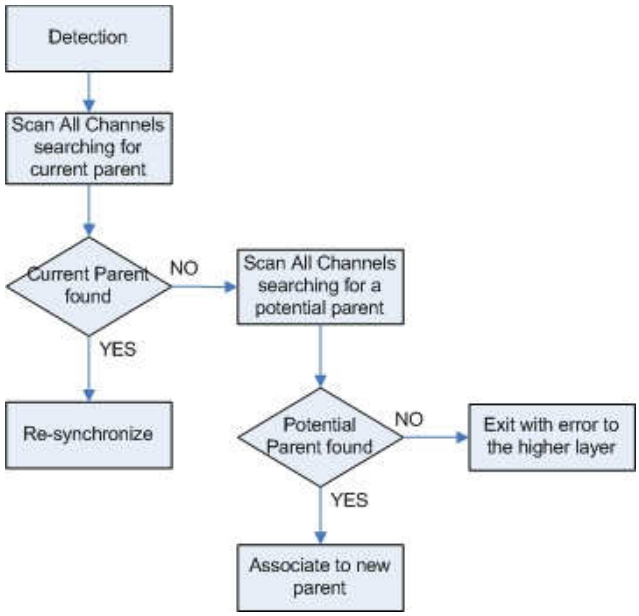


Figure 4.4: Flow chart of the standard orphan scan procedure from the orphan's point of view

On the Parent node side (only applies for devices with parental capabilities)

- The NWK layer of the parent device is notified by the MAC layer of the presence of an orphan device through the *orphan indication primitive* when it receives orphan notification message.
- The parent determines whether the orphaned device is its child: this is done by comparing the IEEE 64 bits address of the orphan with the addresses of its children that are recorded in the neighbour table.

- If a match is found in the neighbour table, then the NWK layer includes the short address of the device, which will be sent by the MAC sub-layer. An acknowledgment is sent to the NWK layer when the procedure is successful using the appropriate primitive.
- If no match is found, the procedure is stopped and no indication is sent to the higher layer.

4.4.2. Timing analysis

In this section, we propose to evaluate the performance of the standard orphan scan procedure in terms of inaccessibility time. The inaccessibility time is defined as the duration starting from the instant where the fault occurs until it is completely recovered and the node resumes its normal operation.

In our timing analysis throughout this report, the inaccessibility time is considered as the key indicator to compare the performance of the proposed mechanisms. In fact, the main objective of the reactive mechanism is to reduce the latency of re-association when the currently used parent becomes unreachable by proposing a faster parent re-selection procedure.

4.4.2.1. General Assumptions

In order to estimate the duration of the latency period, we will express it in terms of Beacon Intervals and Superframe Durations, expressed respectively in terms of Beacon Orders (BO) and Superframe Orders (SO). This results in a general estimation when changing these parameters in the network settings. We consider that two nodes are associated when the association procedure succeeds on each part at the network layer level.

In our analysis, we do not consider the latency that may induced by collision when the CSMA/CA is used in the MAC layer. Our main objective is to evaluate the pure latency induced by the different fault-tolerance mechanisms independently from the MAC layer protocols.

The inaccessibility duration starts from the instant when there is a total link failure between the parent and the child device. The computation of the inaccessibility duration will be done in the MAC layer of the child device. Although the proposed mechanisms are localized in the NWK layer, we can consider that the inaccessibility duration calculated on the MAC layer is the nearly the same in the NWK layer since the delay between the two layers is so small that it can be neglected.

4.4.2.2. General Formula

The total inaccessibility time (IT) starts from the instance when the link failure occurs and comprises the duration needed to detect the fault occurrence (defined by the network designer), and the time durations of all sub-routines that have to be executed by the child node in order to achieve the complete procedure. Formally, this can be expressed by the following equation:

$$IT = T_{dect} + T_{scan} + T_{cal} + T_{sync} + T_{asso} \quad \textbf{4.1.}$$

Where:

- **T_{dect} =** Detection duration: is the time duration between the occurrence of the fault and the moment the node detects it.
- **T_{scan}=** Channel Scan duration: is the time duration that the orphan node spends listening to the channel(s) in order to detect potential parent(s).
- **T_{cal} =** PAI Calculation duration: is the time duration spent by the node in order to process the calculation of the PAI indicator.
- **T_{sync} =** Synchronization duration: is the needed time duration in order to synchronize the orphan node with its newly elected parent.
- **T_{asso} =** Association duration: is the time duration needed in order to perform and complete the association procedure between the two devices.

4.4.2.3. Calculations

Taking into consideration the different phases presented in sub-section 4.4.1, if the child device finds its parent after the first scan of all logical channels of the network, the inaccessibility duration is expressed as:

$$\text{ITstd'} = \text{aMaxBeaconLoss} \cdot \text{BI} + \text{LCS} \cdot aWaitResponseTime \quad \textbf{4.2}$$

where:

- *BI*: Beacon Interval = aBaseSuperframeDuration * 2BO
- *BO*: Beacon Order: integer value between 0 and 14
- *LCS* (Logical Channel Set): Number of logical channels defined
- *aResponseWaitTime* = 32 * aBaseSuperframeDuration
- *aBaseSuperframeDuration* = aBaseSlotDuration * aNumSuperframeSlot*s*
- *aBaseSlotDuration* = 60 symbols
- *One symbol* = variable depending on the operating frequency and throughput of the sensor device.

In order not to lose generality the following IT will be calculated in symbols.

$$\mathbf{IT_{std}}' = \mathbf{aMaxBeaconLoss} \cdot \mathbf{aBaseSuperframeDuration} \cdot 2^{\mathbf{BO}} + \mathbf{LCS} \cdot 32 \cdot \mathbf{aBaseSlotDuration} \cdot \mathbf{aNumSuperframeSlots} \quad \textbf{4.3}$$

Illustrative example.

Let us consider the case of a ZigBee cluster-tree operating with *BO* = 3, *aMaxBeaconLoss* parameter set to 3 and a symbol duration = 6 μs. This results in:

$$\boldsymbol{IT'_{std} = 3,24\ sec}$$

As it can be noticed, such a value is extremely high, especially in time critical applications. This points out the need to lower the value of this IT by proposing improvements to the standard orphan re-association mechanism

End of the example

If, after the channel scan, the orphan device finds it parent, it will synchronize to it and restart normal operation again. This means that the inaccessibility time presented above is the minimal possible value, with the orphan scan procedure.

If the orphan device does not find its current parent after the channel scan, it will start another channel scan in order to find out in its vicinity possible parents to which it may re-associate in order join the network again and resume its normal operation. This will introduce a second contribution to IT_{std}, expressed as follows:

$$IT_{std}" = LCS \cdot aWaitResponseTime + BI \quad \textbf{4.4}$$

The total inaccessibility time in the case the orphan does not find its current parent will be:

$$IT_{std}total = aMaxBeaconLoss \cdot BI + 2 \cdot LCS \cdot aWaitResponseTime + BI \quad \textbf{4.5}$$

Illustrative example.

Using the same network parameters of the previous example we would obtain:

$$\boldsymbol{IT_{std}total = 6,08\ sec}$$

It can be observed that the inaccessibility time value with the standard orphan scan procedure is quite high and may be unacceptable by WSN applications with timing constraints.

End of the example

Fig. 5.3 presents the total inaccessibility time of the standard orphan scan procedure as a function of the Beacon Interval (BI) considering *aMaxBeaconLoss = 3*. The figure shows that the time spent when the parent is not found is approximately twice higher than when the parent is found. In addition, the inaccessibility times increase exponentially in function of BO.

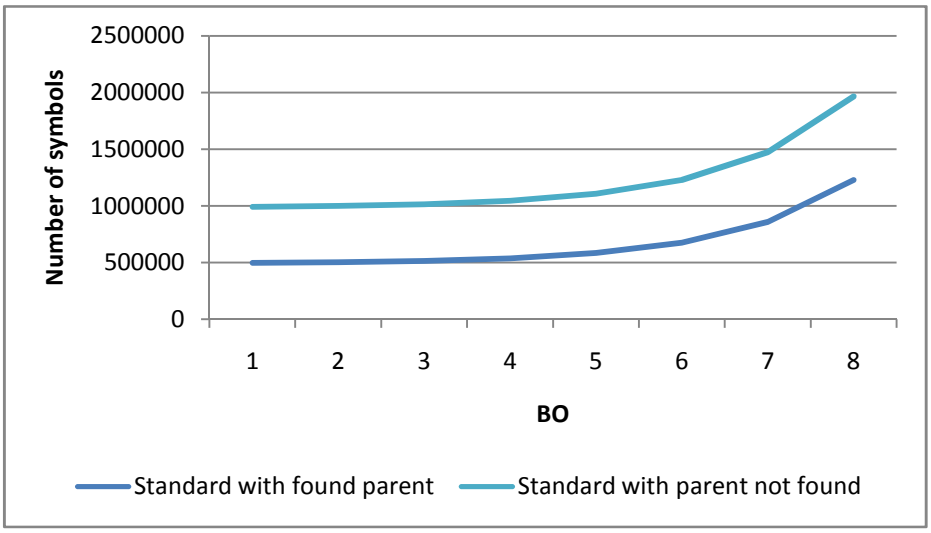


Figure 4.5: The inaccessibility times using the orphan scan procedure

4.5. The Reactive Re-association Mechanism

The reactive re-association mechanism is designed to provide the nodes with the ability to perform a faster **election** and then **association** to a new parent in case of a link failure with their current parent. It is triggered by a child node as a reaction of failure of the link.

4.5.1. The PAI indicator

In the standard specification, the selection criteria of a new parent only takes into consideration the *Link Quality Indicator* (LQI) measured at a given moment as a quality indicator for the potential new parent. However, this indicator cannot provide the child node with an accurate idea on the real performance of the potential parent. In order to provide the node with a more accurate decision knowledge, we propose to introduce a *hybrid indicator* that does not only rely on the *LQI* measure

but also on other important parameters that may affect the selection of the new parent in the cluster-tree such as: the depth of the possible parent (*Dp*), the transmit failure rate (*Tf*), the energy indicator (*Ei*) and the capabilities of the possible parent node. Thus, we define the new metric called the Parent Assessment Indicator (PAI) as:

$$PAI = LQI \cdot (a \cdot Ei) \cdot \left(\frac{b}{Dp}\right) \cdot \left(\frac{c}{Tf}\right) \quad \textbf{4.6}$$

Where *a*, *b* and *c* are integer weighting indicators values. Where:

$$0 < b < 6$$

Since greater values of the *Dp* and *Tf* parameters indicate less performance of the potential parent node, we make them contribute in the appropriate way to the *PAI* calculation. The greater the values of these parameters, the more they will decrease the value of the PAI.

As previously mentioned, the higher the PAI, the better the parent is. For that reason, when taking a decision to switch to a new parent, the child node must choose the potential parent offering the highest PAI value.

Since most of the variables and parameters used in the IEEE 802.15.4/ ZigBee standard are encoded in an 8-bit format, the *PAI indicator* is chosen to have the same format to be easily integrated in the neighbor table. This means that the *PAI indicator* can vary between 0 and 255. For that reason the entry parameters and weighting variables considered when processing the value of the PAI must be chosen so that the resulting value of the *PAI* varies in the allowed interval.

Table 5.1 summarizes the different parameters that affect the *PAI* with their variation range and the resulting values that are considered for calculations.

Id	Name	Bit range	Variation	Considered values
LQI	Link Quality Indicator	0x00 – 0xff	0 – 255	0 – 255
Ei	Energy Indicator	0000 – 1100	Critical, 33%, 66%, 100%	0,2; 0,6; 0,8; 1
Dp	Parent Depth	0x00 – *nwkcMaxDepth*	0 - *nwkcMaxDepth*	1 – 6 (Max)
Tf	Transmit Failure	0x00 – 0xff	0 – 255	0 – 255
PAI	Parent Assessment Indicator	0x00 – 0xff	0 – 255	0 – 255

Table 4.1: PAI input parameters

In order to respect the variation range of the PAI, it is important that the network operator chooses adequate values of the weighting parameters *a*, *b* and *c*. If fact, even though no specific conditions were imposed for the choice of the weighting variables, these parameters should carefully configured; if not the PAI runs the risk of taking values that are higher than the allowed maximum (255) which will cause overflows and incorrect behaviour of the mechanism.

Since the *PAI* reflects the quality of link of potential parent in a given node, it is interesting to define the intervals that define the quality level of the potential parents. This enables to determine the range of PAI values under which a parent device considered as good or bad. Figure 5.4 presents the range of PAI values and their interpretation.

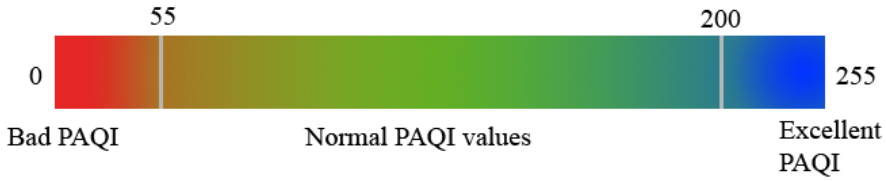


Figure 4.6: The PAI variation interval

As shown in Figure 5.4, the PAI variation interval is subdivided into 3 parts.

- [0 – 54]: This interval defines the bad PAI values. If the PAI relative to the parent fall in this interval, the node decides that this parent is a bad alternative to join the network.
- [55 – 199]: This interval defines the normal PAI values. This should be the variation zone for the majority of received packets during normal operation.
- [200 – 255]: This interval defines the excellent PAI values. The packets received with these values of PAI are very rare, since that in order to obtain such results, the communicating nodes have be in near proximity and should communicate in excellent radio conditions. Moreover, the other parameters of the PAI have also be as best as possible: i.e. the parent shall have excellent power level, the child node should have a minimal depth and the transmit failure should be very low.

4.5.2. Mechanism description:

Figure 5.5 depicts the reactive re-association mechanism.

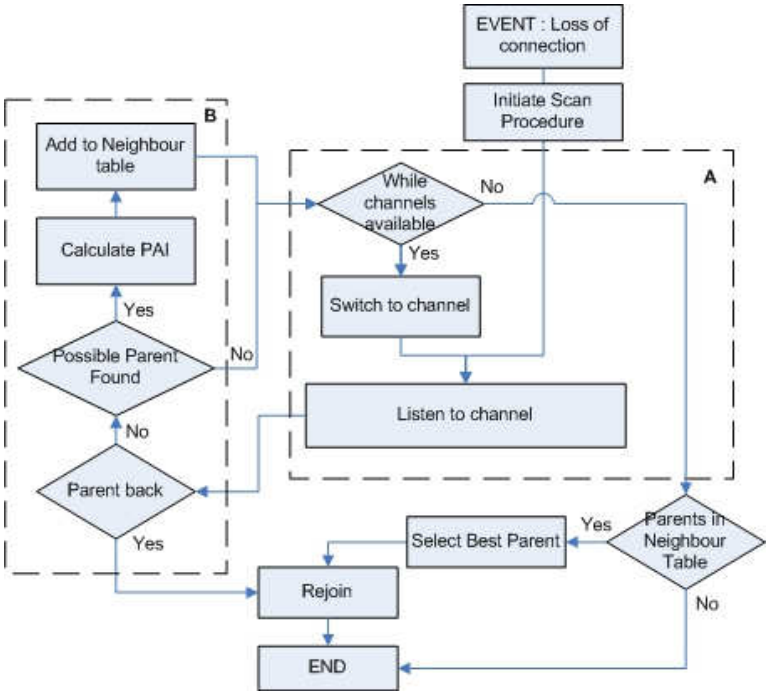


Figure 4.7: The Reactive Re-association Mechanism

As shown in Figure 5.5, the reactive mechanism is sub-divided into two main phases (i.e. A and B).

The reactive mechanism process begins with the occurrence of a fault in the communication process between the child node and its parent. This event initiates the reactive orphan scan procedure mechanism that will enter *phase A* of the mechanism:

- **Phase A: Scanning & Parent Classification Phase.** The device performs the channel scan starting with the channel in which it is actually operating in order to search for its current parent (i.e. the one to which it was associated before the link failure). In the meanwhile, it also searches for other devices in its vicinity that can be potential parents. After scanning the current channel, the scan procedure goes through all the defined logical channels if the initial parent is not found. On each channel, an *orphan notification* message is sent, and the device listens to the channel for *aWaitResponseTime* in order to allow sufficient time for potential parents to send a *realignment command* or to listen to their beacon frames. If the device detects beacon frames from possible parents, it will compute *the PAI indic*ator associated to each one of them and store that information in the neighbour table.

- **Phase B: Re-association Phase.** This phase starts with the end of the listening period. If the device received a beacon frame from the parent to which it was initially associated, the mechanism stops and the device realigns with its current parent and resumes normal operation. If the device did not realign with its current parent, but received one or more

beacon frames from other potential parents, it will select the best potential parent based on the PAI information. Recall that the best devices are those with the highest PAI indicator. The child device will associate to the best found device even its quality is not very high. This is a design choice since that we prefer that an orphan rejoins the network and resumes, even elementary data transmission, than that it remains disconnected from the network.

Another condition must be considered when choosing the new parent. In fact, the child device can request the potential parent to fulfil the following condition:

$$PAI < S \quad \textbf{4.6}$$

Where S is a minimum threshold assuring a sufficient performance level for the new parent.

This latter condition assures that the newly elected parent must have a sufficient quality to act as a parent coordinator with child device, which ensures a good performance of the communications with the child device. In fact, if the performance of the best selected parent from the neighbour table is bad (lower than a given threshold) the orphan device should avoid associating to it since the link failure would be very likely to happen, which induces to restart the reactive mechanism. Such a behavior will consume more energy from child device due to repeating the scan procedure after each fault detection.

For that reason Condition 5.2 can be considered as a stopping condition that will avoid cyclic calls to the reactive mechanism, but the orphan node may not be able to join the network anymore if the *S* threshold is too high.

If no potential parent was found, the device finishes the reactive re-association mechanism by sending an error to the higher layer indicating that it is not in range with any possible parent.

Although the device could not connect to any parent, it can potentially perform point-to-point data transfers with another child device, which may be a member of another cluster, in order to establish a communication flow. Nevertheless, our proposal does not consider this situation.

4.5.3. Timing analysis

As explained in Section 4.5.2., the reactive re-association mechanisms may present two situations:

- The orphan device finds its current parent while executing the reactive re-association mechanism. This will make it re-associate again to its current parent and the reactive mechanism stops.

- The orphan device does not find its current parent in any logical channel. Thus, it will re-associate itself to the best parent device found during the scanning phase.

These two situations induce different inaccessibility times, which will be analyzed in the following paragraphs.

4.5.3.1. When the current parent is found

The first step is the detection of the current parent failure. The time duration of this phase is similar to the detection phase of the orphan scan procedure described in the IEEE 802.15.4/ZigBee standards (i.e. *aMaxBeaconLoss*BI*).

The minimal time duration of the reactive mechanism occurs when the device would find its current parent in the current channel. This results in an Inaccessibility Time (IT) equal to:

$$IT_{reac}min = aMaxBeaconLoss \cdot BI + aWaitResponseTime \quad \textbf{4.7}$$

Illustrative Example

We consider the same network configuration as in the previous example where the BO = 3 and the *aMaxBeaconLoss* = 3 and a symbol duration = 6μs. It results that:

$$IT'_{reac} = 0{,}368\ sec$$

However, the orphan device may not find its current parent in the first channel that it scans. It can have to go through all the channels of the network and find its current parent in the last channel. This would be the worst case of the IT when using the reactive mechanism to associate to the previously used parent. In this case, the IT is expressed as:

$$IT_{reac}\, max = aMaxBeaconLoss \cdot BI + LSC \cdot aWaitResponseTime \quad \textbf{4.8}$$

Illustrative Example

In this case, using the same network settings as in the previous example, we would obtain:

$$IT_{react}\ \max = 3,29\ sec$$

It is interesting to notice that if the orphan device has to perform the scan through all the logical channels to find its current parent it will spend as much time as the orphan scan procedure defined in the standard to re-associate. However, in general, the reactive re-association mechanism reduces the IT as compared to the standard behaviour. In fact, the current parent of the orphan device is usually found within the current operating channel and not in another channel of the network.

4.5.3.2. When current parent is not found

If the current parent was not found after channel scan, the mechanism will select the best potential parent device in its vicinity based on their PAI values and associate to it. It will not have to perform a scan through all the channel set a second time since that it already saved the information concerning the potential parents.

For that reason, the IT in this situation is expressed as:

$$IT_{reac}2 = aMaxBeaconLoss \cdot BI + LCS \cdot aWaitResponseTime + BI + Tasso \quad \textbf{4.9}$$

Numerical Example.

Using the same network settings considered in the previous examples, we obtain:

$$IT_{reac}2 = 3{,}133\ sec$$

4.5.3.3. Reactive Mechanism vs. Standard Mechanism

The PAI calculation duration can be considered as very small since it is processed within the node itself. For that reason, it is considered in the following calculation. Figure 5.6 shows a comparative plot for the reactive and the standard mechanisms, which present the IT as a function of the Beacon Order (*BO*) when *aMaxBeaconLoss = 3*.

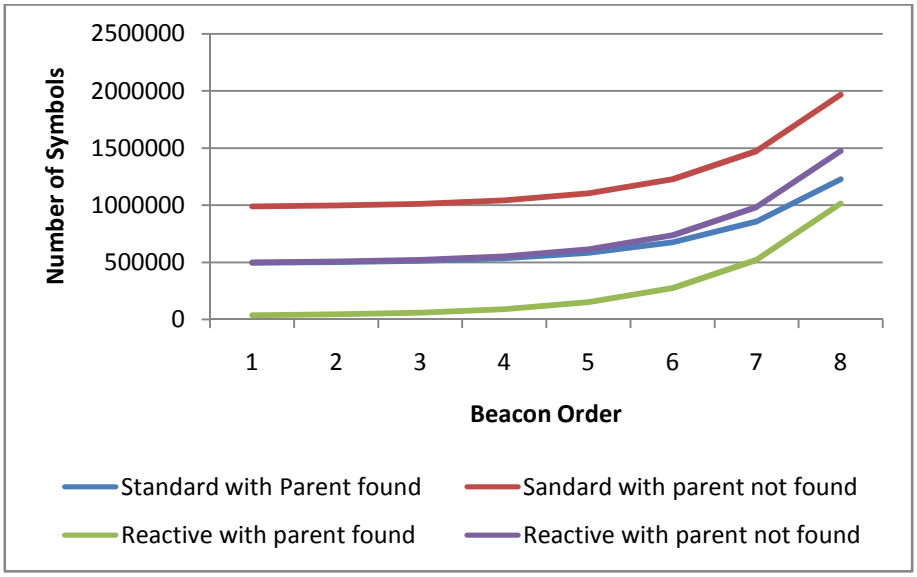


Figure 4.8: Comparing the ITs in the cases of the standard mechanism and the reactive re-association mechanism

The reactive re-association procedure significantly reduces the ITs as compared to the standard orphan scan procedure only in the cases where the current parent device is found in the first channels during the channel scan.

Considering the case where *BO = 5* and *aMaxBeaconLoss = 3*

The decrease in IT between the two mechanisms is equal to:

- When the current parent is found

$$IT_{improvement}1 = \left({(IT_{std}1 - IT_{reac}2)}/{IT_{std}1} \right) \cdot 100 = 73{,}68\ \%$$

- When associating to a new parent

$$IT_{improvement}2 = \left({(IT_{std}2 - IT_{reac}2)}/{IT_{std}2} \right) \cdot 100 = 44{,}4\ \%$$

4.5.4. Advantages of the reactive approach

4.5.4.1. Overcoming the limitation of the standard orphan scan procedure

As described in the previous sections, the standard specification mentions that the "orphan scan procedure" can only connect an orphan device to its parent. For that reason, it is not a fault-tolerance mechanism since in case of a permanent parent failure, the child devices will not be able to associate to any new parent and would be totally isolated from the network unless they perform again a channel scan looking for potential parents. It can noticed that the standard orphan scan procedure is very time consuming and will highly increase the delays in the network in failures occur frequently. Our proposal offers the opportunity for the child device to perform two operations in one: it can search for its parent and, at the same time, look for other parents in case the former one was not found. This improves timing constraints and the efficiency of the parent search procedure.

4.5.4.2. Optimal selection of the new parent based on a hybrid indicator

When choosing a new parent, the standard only takes into account the Link Quality Indicator (LQI) as it is measured at a given moment. This indicator cannot provide sufficiently precise information about the potential parent device since it reflects the quality of a received message a given moment. The PAI takes into consideration other information about the potential parent device that and summarizes them in a unique indicator that is more accurate than the LQI.

4.5.4.3. Reducing the inaccessibility times

As mentioned in section 5.5.3.3., the proposed reactive re-association mechanism significantly reduces the inaccessibility times in comparison to the standard orphan scan procedure. This is very important for time-critical WSNs applications.

4.6. Conclusion

In this Chapter we first introduced a new hybrid quality indicator used by the child node to assess the quality of the parent nodes. Then, we presented details about the reactive re-association mechanism that corrects the sudden link failures between a child device and its parent and preformed timing analysis in order to compare the performance of this proposal to the standard orphan scan procedure. The proposed mechanism offers valuable improvements compared to the standard, mainly the reduction of inaccessibility times.

Chapter 5 will present a proactive fault-tolerance mechanism aiming to prevent the quality degradation of the parent device.

Chapter 5

A Proactive Fault tolerance Mechanism for Cluster-Tree WSNs

In this chapter, we present a proactive re-association mechanism that will anticipate the parent's failure by analyzing the parent-child link quality and select a new parent In advance in order to avoid the link failure. We also overview the influence of each parameter on the general behaviour and propose timing analysis of the performance of this mechanism

5.1 Introduction

To improve the overall reliability of the network, it is better for a child node to take preventive countermeasures by changing to a more reliable parent before the link between them totally fails or presents a too high degradation. The newly chosen parent will be the device in vicinity offering parental capabilities that presents the best performance. This is the main idea of the proactive approach that is developed in details in this section.

5.2. Standard Specifications

There is no preventive change of parent specified in the standard. This is explained by the fact that re-association, when made on a bad trigging condition, is very resource consuming and implies changing the addressing scheme which introduces high delays and has a bad influence on time sensitive applications.

As a consequence of this, a proactive approach shall introduce adequate conditions before deciding to switch to a new parent. The forecast that the parent is bad, decreasing quality or is failing must be as accurate as possible. If the conditions are wrong chosen or are weak, we run le risk that the network nodes will change too often from parent without having the need to do that.

5.3. Mechanism Description

The proactive mechanism, unlike the reactive mechanism, doesn't need a triggering event; it is continuously executed within the child node. As shown on figure 5.7, the mechanism can be subdivided into 6 main phases.

The initialization of the model parameter is run at the first execution of the mechanism, usually when turning on the node or during the deployment phase. It consists in the initialization of the different parameters that determine the behaviour of the mechanism. These parameters are explained in detail in the next steps of the mechanism when they are introduced.

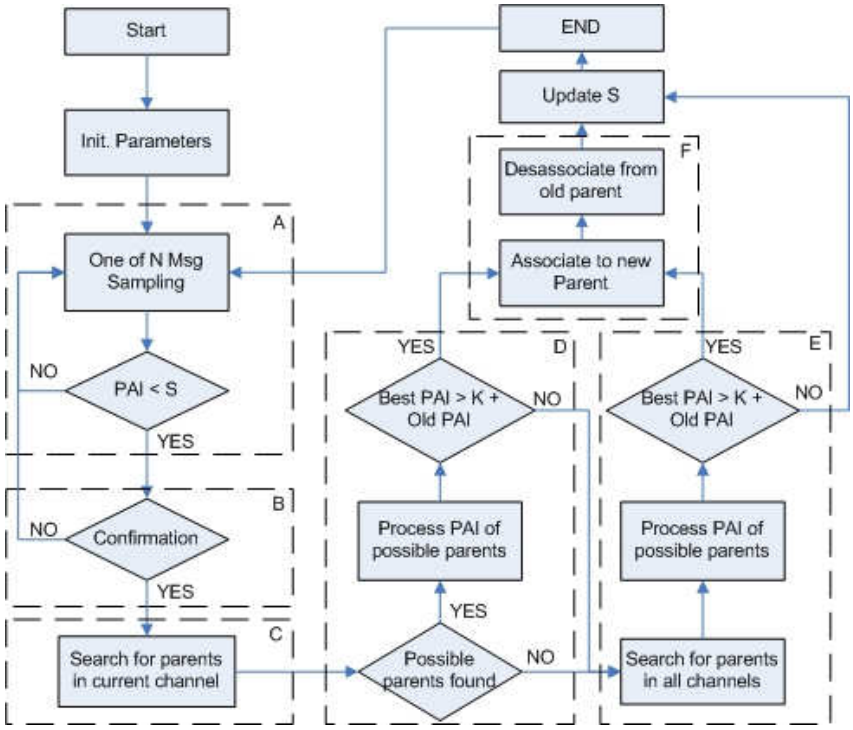


Figure 5.1: The proactive re-association mechanism architecture

Step A (Fig 5.1.A) starts after the parameters initialization. Comparatively to all other steps of the mechanism, step A is the longest one since it is the continuously executed during the run time of the proactive mechanism. Step A consist in a sampling phase where a defined number of packet is analysed in order to detect the current parent quality. The sampling consists in sampling a received frame periodically or each every fixed number of received packets, defined as *R*. Once every *R* received packets, the node will process the PAI relative to its parent and compare it to the *S* threshold value thas been set during the initialization period. The child node will then test the following condition:

$$PAI < S \qquad 5.1$$

If condition 5.1 is triggered, the mechanism switches to the confirmation phase (step B), otherwise it remains in step A and restarts the same sampling operations explained earlier.

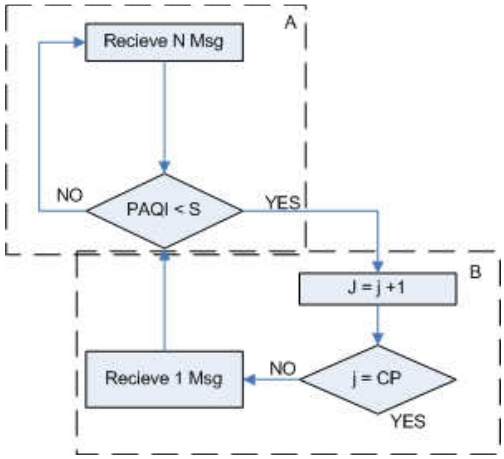


Figure 5.2: Details of phases A and B

Step B is activated by the fulfilment of condition 5.1. Its main aim is to confirm that the parent is effectively bad. Each received packet must verify that condition. If the number of received packets fulfilling that condition reaches a defined number of confirmation packets (*CP*), the child node acquires the certainty that the parent is degrading.

It is important to introduce such a confirmation phase. In fact, if the decision to activate the rest of the mechanism is made only based on the first bad received packet it would lead to excessive triggers, since the packets on the wireless medium can experience sudden but timely limited perturbations. That is why the confirmation phase is introduced to have only relevant (i.e. that reflect the deterioration of the Parent-child link) triggering of the overall mechanism.

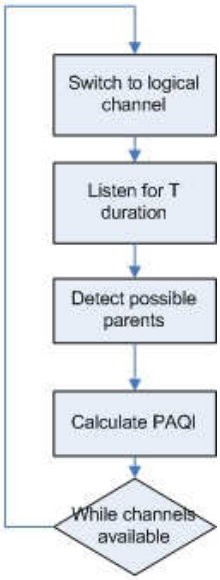


Figure 5.3: Details on the Channel Listening Phase

When the child device has the certainty that the parent device is failing it initiates the following steps of the proactive re-association mechanism by initiating a scan in its current channel searching for other potential parents during its inactive period. This scan procedure does not interfere with the normal operational behaviour of the node with its current parent.

If possible parent devices were found in the current channel, the child device will go through the different steps of phase D (Fig 5.D) of the mechanism: it will process the PAI indicator for the different potential parents and test if the potential parent presenting the best PAI fulfils the following condition:

$$Best\ PAI\ > K + old\ PAI \qquad \textbf{5.2}$$

Condition 5.2 is introduced to insure a given improvement when performing a parent switch. In fact, since that the parent switching operation is resource consuming and introduces a non negligible delay in downstream data communications (refer to section 5.6.6.), if the child device decides to switch parent, that switch should compensate the temporary performance degradation.

If a potential parent fulfilling the previous condition is found, the mechanism will pass to step F (Fig 5.F). In that phase, the child initiates the association to the newly elected parent device and then disassociate from its current degrading parent. Finally, if the dynamic update of S is activated, the node updates the value of the S threshold with regard to its new parent quality.

If no potential parents were found during the current channel scan or if there were no potential parents within the current channel fulfilling condition 5.2., the child will initiate a channel scan

through all the channels looking for potential parents. It will calculate the PAI of the different devices that were found and test if the best device (with regard to the PAI) fulfils condition 5.2. If such a potential parent is found, the mechanism jumps to step F (similar to the previous case), otherwise it will update the *S* triggering threshold.

The dynamic update of the S threshold is optional, meaning that the mechanism can be active or not. This is defined as a pre-processing constant. If activated, the dynamic update of the *S* threshold is the last step of the mechanism that can be executed in two different situations:

1. If a parent change occurred
   In this case, the S threshold should be dynamically updated in order to take into consideration the effective performance of the new parent. The threshold is set to:

$$S_{new} = PAI_{new-parent} - TU\ (\%) \qquad \textbf{5.3}$$

   This update makes sure that the S threshold is always adapted to the parent device.

   To give a practical intuition of this S update, consider the following example: If the device was previously associated to a good device (i.e. that presents high values of PAI in normal operations, thus imposing a relatively high value of S). When the current parent degrades, the child associates to a new parent; such a parent can present less quality than the current one used to offer, still it is better at the moment of the mechanism trigger. If the S threshold remains at the same previous value, it would be too high with regard to the quality offered by the new parent. Choosing a lower S value assures that the proactive mechanism will only be triggered when really needed, i.e. when the relative quality of the current parent is lower than a given value.

2. If the no parent change occurred
   This means there are currently no potential parent devices in vicinity of the child. If the parent quality effectively follows a decreasing trend, the following samples will also be bad; if the child keeps the previous value of S, the mechanism would trigger again and again. That is why we decide to lower the value of S in order to prevent the mechanism of triggering too many times.

$$S_{new} = S_{old} - Tu\ (\%) \qquad 5.4$$

   If fact, lowering the S threshold means that the child device accepts less quality from its parent so that the proactive mechanism triggers less often.

If the node's current parent continues degrading and no potential parent never appears, the S threshold will decrease until a value where the mechanism would not trigger anymore, meanwhile allowing the child node to continue functioning even with the degraded quality.

5.4. The Switching Conditions

In this section we analyze the role and influence of each parameter of the proactive mechanism defined in 5.3. In fact, varying the model parameters has a significant influence on the overall behaviour of the mechanism; that is why the values of the different attributes should be chosen in a manner that optimizes the performance of the mechanism.

5.4.1. The Sampling Rate R:

The sampling rate defines the frequency of the channel assessment through the calculation of the PAI based on the information retrieved from the reception of all messages during the active period of the child device (i.e. data frames and beacons) and the information included in the neighbour table of the child device. In fact, *R* expresses the frequency by which the received messages will be tested by processing the PAI.

The lower the value of R, the better is the child's knowledge of the parent-child link. The higher that value, the less frequently the parent-child link will be assessed, thus it could be more prone to deterioration without the child initiating the proactive procedure in order to avoid parent loss.

The trade off of a higher sampling rate is the resource consumption it introduces, namely energy consumption. This disadvantage has to be taken into consideration when searching for an optimal value of *R*. Since it consumes more energy, the implementation of the proactive mechanism should be well studied in order to avoid the rapid depletion of the network's energy resources. Moreover, the processing cost of the PAI calculation and decision taking, even if it could be negligible, should be taken into account knowing the low processing abilities of the nodes.

5.4.2. The minimum parent quality threshold S:

The *S* quality threshold is the minimal value of the PAI indicator that ensures sufficient transmission conditions between the parent and child device, thus it also defines the link quality level under which the child device decides to initiate the rest of the proactive mechanism. If the quality of the parent-child link is sufficiently good (higher than *S*) there is no need to switch to a new parent even if there are potential parent devices that present better quality indicators. The *S* threshold avoids frequent and useless parent changes.

The *S* threshold can be set either on a static basis or be dynamically updated during the execution of the proactive mechanism (in step F) using the user-defined *Tu* factor. The advantage of a dynamically updated S value is explained in the general description of the mechanism (section 5.3).

5.4.3. The adaptation factor Tu:

The *Tu* factor is used in the dynamic adaptation of the S value (step F). It is a user defined percentage of deterioration that is acceptable by the child device when it is connected to a new parent, as in formulas 5.3 and 5.4. The higher the *Tu* value, the lower will be the new *S* value, so less frequent will be the triggering of the mechanism. If the Tu value is low, the *S* value will be very close to the PAI of the actual parent or to the previous value of *S*. An optimal value of *Tu* should be found in order to avoid any bad influence of it on both the number of triggers and the resource consumption of the mechanism.

5.4.4. The confirmation phase

In wireless environments, the received signal strength by the child device can present an erratic behaviour. If fact, the signal quality may present a temporary degradation (figure 5.4) and go back to normal quite rapidly.

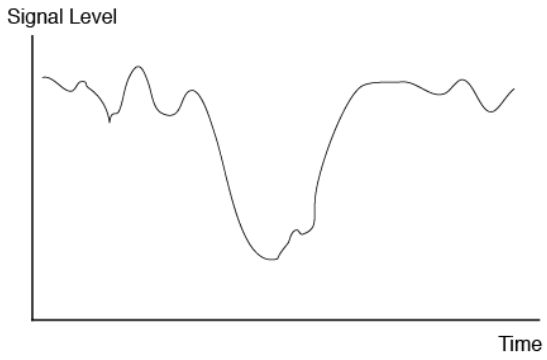


Figure 5.4: Temporary perturbation of received signal level

However, it may also be experiencing real problems that will translate into a gradual degradation trend (figure 5.5).

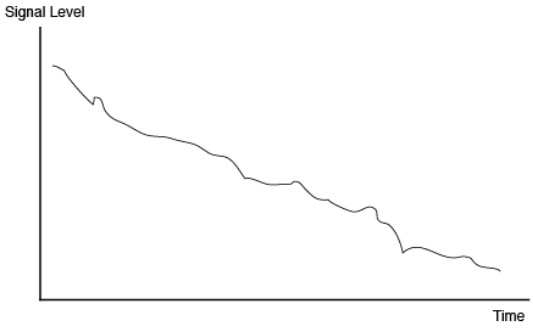


Figure 5.5: Decreasing received signal level

It is important for the child device to make the difference between both situations. That is why that when detecting a bad transmission quality state, the child node initiates a confirmation mechanism to verify if the parent-child link is really degrading.

CP is a fixed, user-defined value that is introduced when initiating the proactive mechanism. The confirmation mechanism consists in the reception of a number equal to *CP* consecutive packets from the parent and processing their relative PAI. If all of the *CP* received packets fulfil condition 5.1, the device knows that the parent-child link is degrading. If one of the *CP* packets does not fulfil condition 5.1 the confirmation phase stops and returns to the sampling phase. The value of *CP* determines the duration (in terms of timing or number of received packets) of the confirmation phase and thus the needed certainty about the link state.

5.4.5. The hysteresis factor K:

An hysteresis factor is introduced in the inequality illustrating the difference in quality between the old parent and the new parent (condition 5.2). This is useful to make sure that the new parent presents a sufficiently better quality than the old one so there will be no ping-pong effect causing the child device to rapidly alternate between parents. This hysteresis factor will also ensure that the new elected parent presents a given higher quality than the old one in order not to initiate the association process (resource consuming and introduces delays) too frequently for parents that are not much better that the actual one.

5.5. Advantages of the Proactive Approach

The presented proactive mechanism presents several advantages in comparison with the standard behaviour.

5.5.1. Improved energy balancing

When taking into consideration the energy indicator to choose a new parent, the mechanism improves the overall energy balancing. In fact, devices offering better energy performances (that have more battery power available or are mains powered) have a grater probability of selection. This makes sure the network efficiently balances energy between the parent nodes.

Switching to a new parent that offers a better link quality also helps in saving energy of both the parent and the child nodes since there will be less retransmissions.

5.5.2. Improved traffic balancing

The nodes with less traffic will have a grater probability of selection compared to those with high traffic or with a big number of nodes associated to them. This way the mechanism offers a means of traffic balancing preventing nodes that are already on heavy load to be more solicited and eventually causing them to have problems and becoming possible faulty nodes.

5.5.3. Better transmission conditions

When associating to a new parent based on both the LQI and the TF indicator (both parameters are part of the PAI formula) the mechanism will tend to elect the device that presents the best transmission conditions and so offering the best Parent-Child link quality. Ultimately, the proactive mechanism will lead to establishing a stable connection between all the nodes that offer the best transmission conditions.

5.5.4. Improved reliability

Since the proactive re-association mechanism prevents the occurrence of faults in the network by anticipating a parent's failure, the overall reliability of the network will improve.

5.5.5. Lower end-to-end delays

By reducing the retransmission rate, improving reliability and traffic balancing and preventing the parent-child link break, the proactive mechanism will sensitively reduce end-to-end delays during normal operation of the network.

It is interesting to note that when defining the weighting values of the PAI indicator, the network designer can define the importance of each performance parameter thus influencing the previously stated advantages.

5.6. Timing Analysis

This situation is very different than the Inaccessibility Time (IT) calculation of the standard mechanism and the reactive re-association mechanism. In fact, the triggering of the mechanism is made by the reception of samples presenting bad PAI quality. In fact, there is no detection IT since that even if bad samples are received during the sampling phase and the confirmation phase, these packets are still useful and communication between the parent and its is maintained.

As long as the child device is connected to its parent it is able to perform normal operations. It is important to avoid the break of communication; that is why the scan for potential parents is done during the inactive period of the superframe of the child device. By that means, the child can gain information about other devices without being disconnected from its parent.

From the IT point of view, the fact that the communication is never broken to perform channel scan will make that:

$$IT_{proactive} = 0$$

This value of IT is valid in both cases that the child device searches for possible parents within it current channel or in other channels. In fact, the device has the ability to switch to another channel when entering its inactive period in order to look for potential parents in other channels.

As pointed out earlier, this inexistent IT is only valid in upstream communications. Downstream communications still can experience delays, since that, after that the child device associated to a new parent, the network will still have to update the routing tables, addressing information and other parameters relative to the new situation of the node.

The IT in the downstream is very difficult to evaluate, since it highly depends on the routing mechanism characteristics and the size of the network.

5.7. Conclusion

In this chapter we introduced the proactive re-association mechanism that offers a new re-association approach that prevents the degradation or expected failures of parent devices. Even if relatively complex, the proactive approach has the important feature of offering inexistent upstream inaccessibility times which is a crucial issue for time critical applications.
Chapter 6 presents a simulation model developed in order to test the behaviour of the proactive mechanism and test the effects of its different parameters.

Chapter 6

Simulation and Performance Evaluation of the Proactive Re-association Mechanism

In this chapter, we present a simulation study of the proactive re-association mechanism aiming to evaluate its performance before being implemented. We show that using the PAI enables fewer but more accurate parent switches than when using the LQI. Finally, we gain a precise idea about the performance modifications when varying the different model parameters, this will help the network designer have a clear idea about the appropriate way to choose the initialization values of the proactive mechanism.

6.1. The Simulation Model

6.1.1. Objectives and general characteristics

The main objective of the developed simulation tool is to demonstrate the feasibility of the proactive re-association mechanism and to assess its behaviour when tuning its different parameters. The obtained results will be helpful to determine the optimal values that will help the network designer to choose adequate settings of the different parameters based on some predefined requirements.

The simulation tool that we have developed is a code line standard C program generating random LQI values that represent different received packets and follow a realistic behaviour.

The simulation does not take into consideration timing issues. It is not developed to compute the inaccessibility times or delays. The only evolution metric of interest is the number of (generated/received) packets.

The simulation results are seen from the child device point of view.

Figure 6.1 presents a general view of the simulation's inputs and outputs.

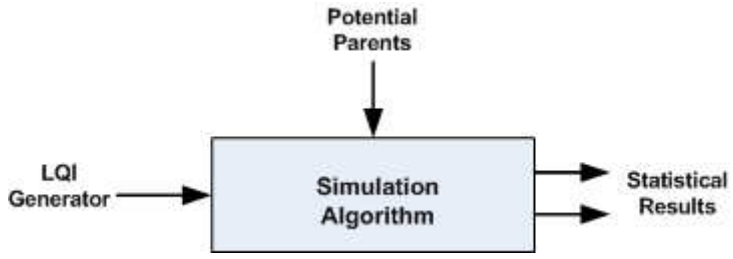


Figure 6.1: Simulation algorithm inputs and outputs.

The inputs of the simulation are:

- A parent structure containing information about the different potential parents in vicinity of the child device. This structure is further explained in section 6.1.3.
- LQI values generated by a random number generator and following the Gaussian distribution in order to mimic the behaviour of the LQI of real packets.

The objective of the simulation is to obtain a clear picture of the number of times the different phases of the mechanism were activated. This is helpful to know which parameters of the model influence which parts of the mechanism. That is why the simulation tool will compute the following outputs:

- The number of triggers of the confirmation phase: represents the number of times the mechanism detected a bad received packet (presenting a PAI under the S threshold) and passed to the confirmation phase.
- The number of parent changes requests: is the number of successes of the confirmation phase, which is also the number of times the child device assessed the network searching for a potential parent.
- The number of effective parent changes: is the number of times the child device found a potential parent fulfilling condition 5.2 and successfully associated to the new parent.

The algorithm of the simulation is the following:

6.1.2. Implementation details

The proactive mechanism is triggered using the PAI indicator. In order to simulate the progress of the overall mechanism, a large number of different PAI indicators have to be generated. The PAI in the simulation will be generated with respect to the general PAI formula (Eq. 4.6) described in section 5.1 of Chapter 4.

Without loss of generalities and for sake of simplifying the simulation, the Tf parameter is not taken into consideration; nevertheless, the general behaviour of the mechanism is not affected and the conclusions remain valid.

This simulation only uses the Ei and Dp parameters as fixed characteristics describing a possible parent device. This information is stored in a C structure as follows:

```
struct Parent
{
int id;
long LQI;
int Dp;
long Ei;
double PAI;
};
```

This structure stores the following information:

- **Id:** The identifier of potential parent device. It is used to differentiate between different devices.
- **LQI:** The Link Quality indicator reflects the link quality between the child and the potential parent device. It is generated as explained is section 6.1.3.1.
- **Dp:** The parent depth. In practice, this information is usually determined from the device's short address as explained in section 4.1 of Chapter 7.However, for the sake of simplicity of representation in the simulation tool we have chosen to make that information directly available to the child device.
- **Ei:** the energy indicator. This information is sent by the potential parent devices to the end devices. It can be considered as a characteristic of the parent device. When the node is battery fed, the Ei of the node would progressively decrease. However, in this simulation study we consider that Ei is fixed. Each parent device will possess a given fixed Ei value. This will not have any impact on the results since that we consider the fact that the simulation is time limited thus will not allow the Ei to show noticeable decrease.
- **PAI:** it stores the result of the PAI equation (Eq. 4.6 in section 5.1 of Chapter 4). This field will be tested in order to take the decisions regarding the parent device switch.

As stated in section 6.1.1. an LQI attribute relative to each parent is generated every time it is needed. The generation of the LQI follows a Gaussian distribution generated using the *gsl_rng.h* and *gsl_randist.h* libraries part of the GNU Scientific Library [22] – GSL 1.9, an open source numerical C/C++ library that provides mathematical routines and random number generators widely used by the open source community in scientifically calculation. During normal operation, when the parent

presents a good performance, we assume that the child device receives packets presenting an LQI value close to 126 which is the mean LQI value. Much scarce are packets possessing high LQIs (i.e. with values higher than 200) or low LQIs (i.e. with values lower that 50) as observed in Figure 6.2.

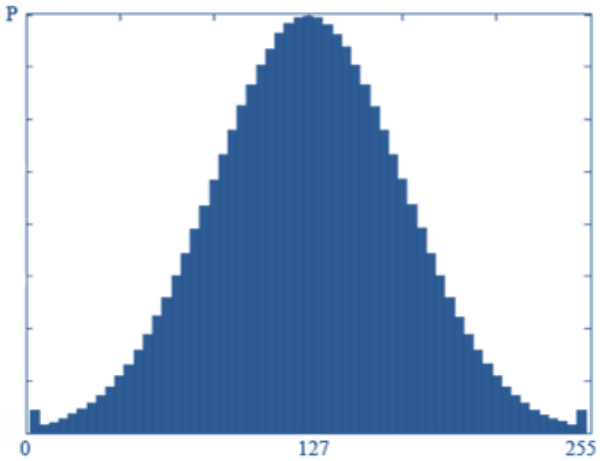


Figure 6.2: Probability distribution of the PAI values

For that reason, during a connection between a child and a parent, it is very likely to receive packets within the [51-200] range. LQIs outside that range are much less probable. That's why, during the simulation, when a child device receives a high number of consecutive bad packets it will conclude that the parent is failing.

This modelling seems to be suitable to simulate basic fault cases without having to have a very realistic model of the parent behaviour.

The values returned by this couple of functions are random numbers (RN) within the [-3, 3] range following a Gaussian distribution. We divide the obtained values by 3 in order to get them within the [0-1] range.

In order to obtain adequate LQI values, we apply the following transformation:

Where RN are the random numbers generated by the random number generator.

This equation results in random numbers within the [0-255] range with respect to the Gaussian distribution and cantered on the value of 126.

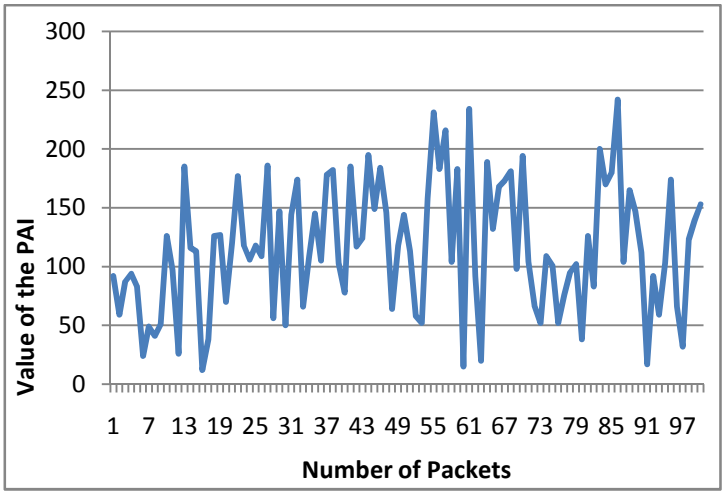


Figure 6.3: PAI variation in function of the number of received packets

Figure 6.3 presents the evolution of the PAI indicator in time. It shows the LQI values varying in the previously defined range with a means around 125.

In the following is presented a simplified example of the used algorithm in the simulation:

```
While (TM < 10 000)
        {
        If (i = R)
                {
                generate_PAI ;
                TM++;
                If PAI < S
                        {
                        CPT++;
                        j =0;
                        while (j <= CP)
                                {
                                generate_PAI;
                                TM++;
                                If (PAI < S) {j++}
                                Else {j = CP +1};
                                }
                        If j = CP
                        {
                        PCR++;
                        Generate_Parents_PAI;
                        If ( PAI_Best_Parent > PAI_old_Parent + K)
                                {
                                Current_Parent = Parent_Best_PAI.id
                                EPC++;
                                }
                        }
                        }
                }
        TM++;
```

```
i++;
}
```

Where: TM = Total number of received messages

I, j = generic counters

CP = Number of needed messages for the confirmation phase

CPT = Number of triggers of the confirmation phase

PCR = Number of parent change requests

EPC = Number of effective parent switches

Generate_PAI = A function returning a PAI value given a random LQI and parent characteristics

6.2. Simulation results

The simulation scenarios were run on a UNIX like system (Mac OS X – Darwin Kernel) with a 2 GHz Dual Core CPU and 1Gbyte of RAM. The results were obtained for different parameter settings to evaluate their impact on the proactive mechanism. The same simulation has been performed in two different cases:

- **Case 1**. The LQI has been considered as the quality indicator,
- **Case 2**. The PAI indicator has been considered as the quality indicator

The objective is to compare the performance of the two quality indicators. In what follows, we present the simulation results.

6.2.1. Effects of the Sampling Rate R

The value of the sampling rate in step A of mechanism is a key factor in the proactive re-association approach. It deeply influences the accuracy of the information the child device has about its parent. Figure 6.4 shows that the number of triggers of the confirmation decreases very fast when the sampling rate decreases (i.e. packets are tested less frequently). In fact, when sampling slowly, probability of testing a bad packet is lower.

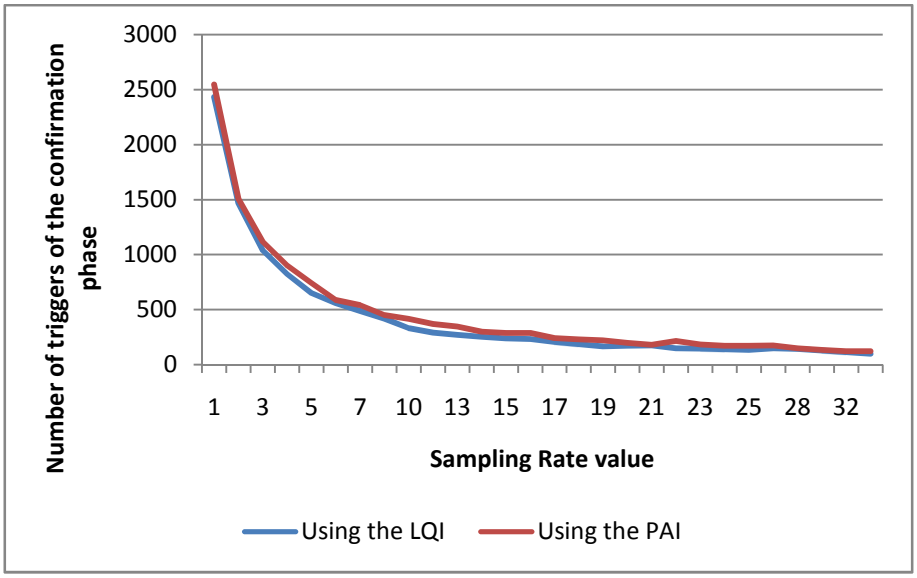


Figure 6.4: Number of the confirmation phase triggering in function of R

The network designer can decide, when planning his network, which sampling rate is the most adapted to his needs with regard to the quality he expects from the mechanism and the confirmation phase triggers rate per number of received packets. In fact, the confirmation phase, even if not very resource-consuming, remains a processing mechanism that, when called too frequently, can be a major source of energy consumption.

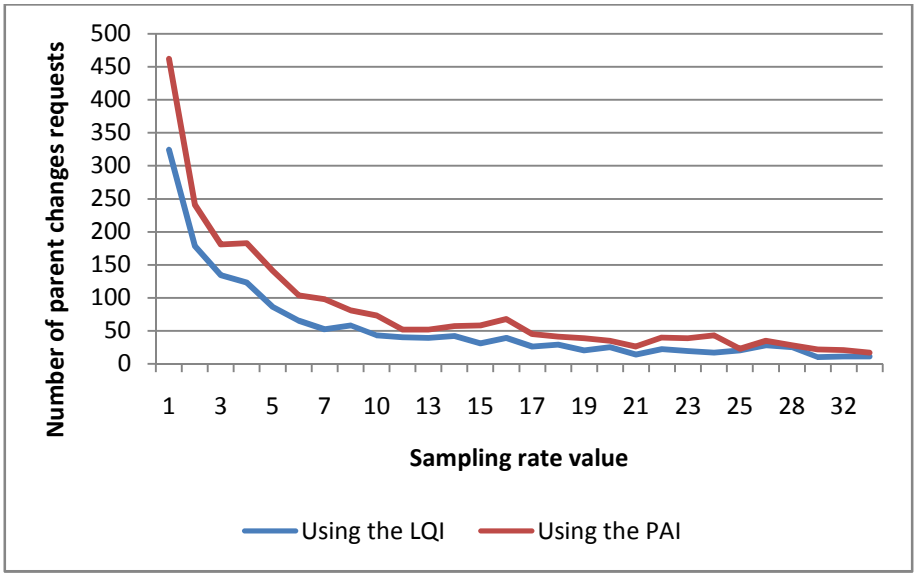


Figure 6.5: Number of parent changes requests in function of R

Figures 6.4 and 6.5 both reveal that the number of triggers of the confirmation phase and the number of parent changes requests are higher when using the PAI than the LQI for the same threshold S. In fact, since that the PAI comprises the Ei and Dp that are degrading parameters, the

value of the PAI will generally be lower than the value of the LQI. For that reason, when using the PAI, the threshold S should be lower than the one when the LQI is used.

The number of parent changes requests tends to be very similar in both cases for high values of the sampling rate. This reflects the difference between the nodes that experience total failure (these are lower in number and lately discovered because of a low sampling rate) and the nodes that experience temporary problems (occurring more frequently but only discovered when using a high sampling rate).

The network designer should find a suitable value of the sampling rate R that represents an appropriate trade-off between the number of triggers of the mechanism and the reliability level expects from his network.

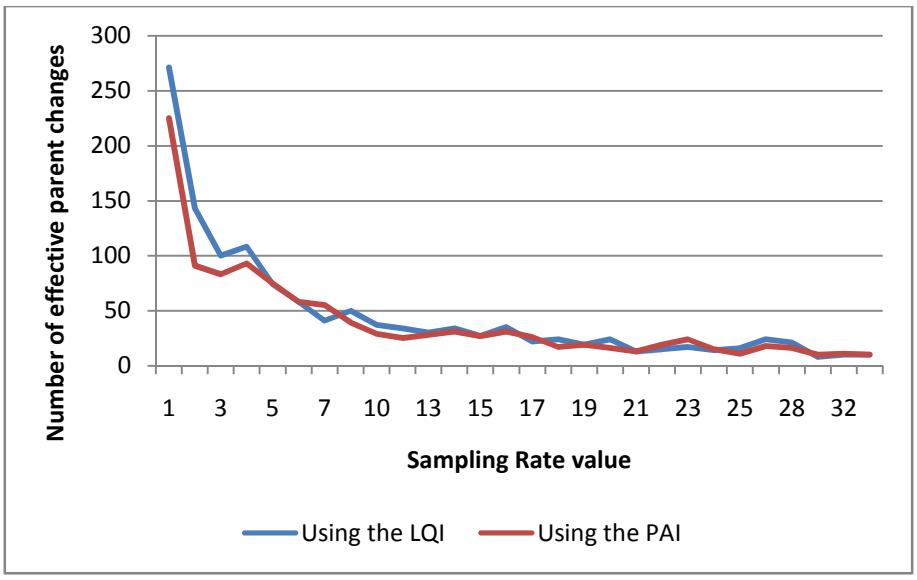


Figure 6.6: Number of effective parent changes in function of R

Figure 6.6 shows that, even if the number of parent changes requests is higher when using the PAI, the number of effective parent changes is generally lower than when using the LQI. In fact, the PAI makes sure that the node associates to the best suitable parent in terms of link quality, energy capabilities and depth in the networks, thus avoiding the child of switching to parents that are not good enough.

6.2.3. Effects of the S threshold

The threshold S is the minimum PAI (or LQI) acceptable level above which a parent switch is possible in the proactive approach. Recall that a trigger is an event of receiving a packet with a PAI or LQI value below the threshold value. It can be observed in Figure 6.7 that the number of activations of

the overall mechanism increases with the threshold value. We notice that the PAI and LQI parameters follow approximately the same variation trend.

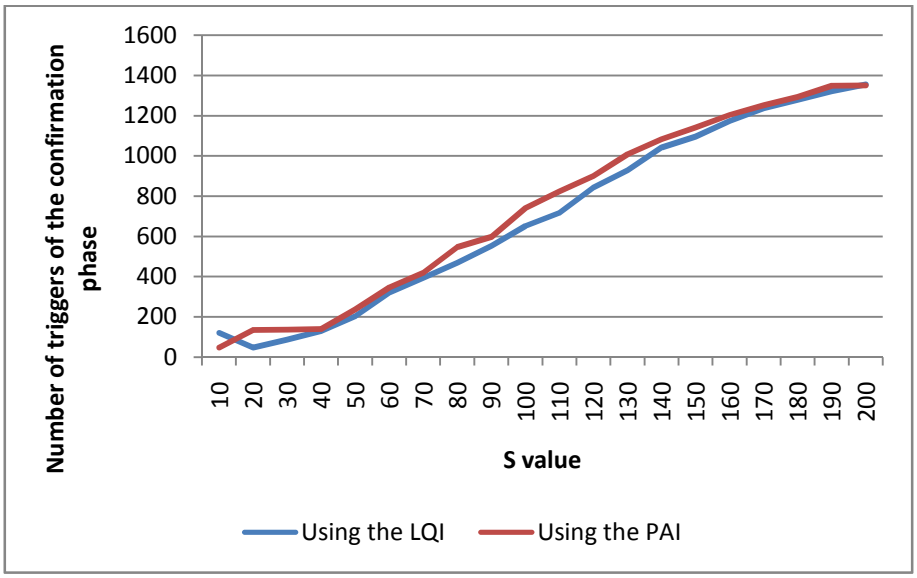


Figure 6.7: Number of triggers of the confirmation phase when varying the S value

Figure 6.8 shows that the number of parent changes requests in function of the S value. It can understood from the figure that the number of parent changes requests when using the PAI as a quality indicator is greater than that the one when LQI is used as a quality indicator. This is due to the same reasons explained earlier in the case of sampling rate in Section 6.2.1.

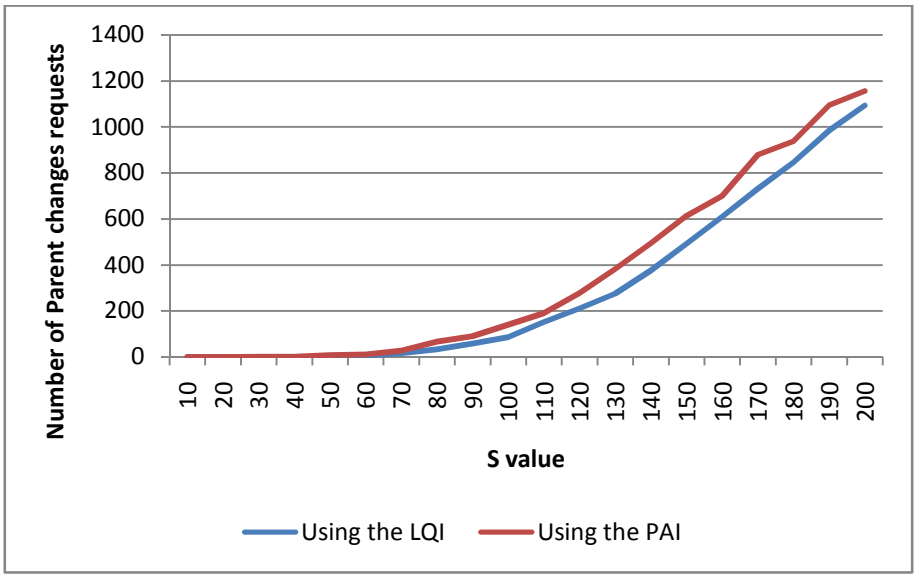


Figure 6.8: Number of parent changes requests in function of the S value

Taking only the LQI into consideration to evaluate the quality of a parent device can be sufficient in some situations but does not provide complete information about the parent quality. The PAI, on the other hand provides more accurate information about the parent quality, and thus when the PAI

value is low it means with more confidence that the parent quality in not acceptable. Figure 6.9 shows the number of effective parent changes as a function of the threshold value.

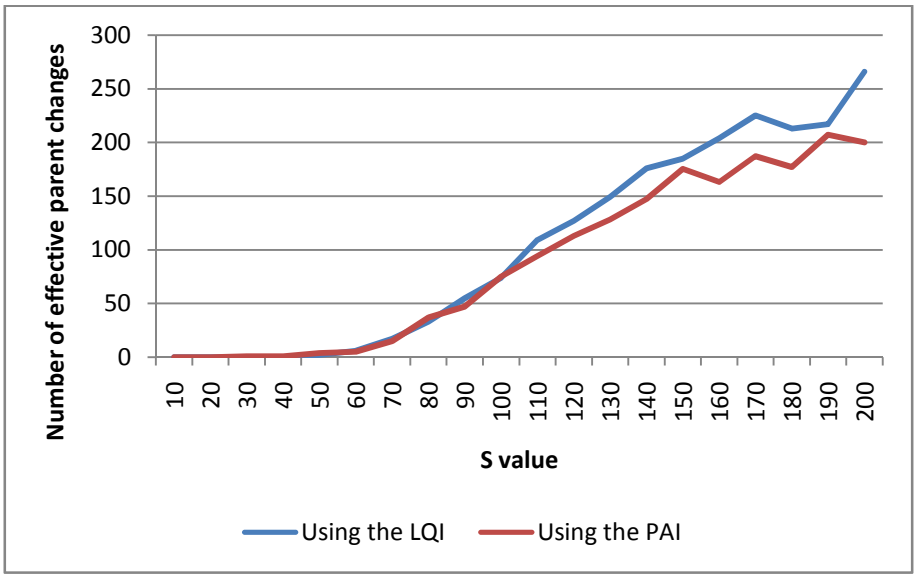


Figure 6.9: Number of effective parent changes in function of the S value

Based on this figure, it can be observed that the number of effective parent changes is generally lower when using the PAI than when using the LQI. This is a noticeable advantage of using the PAI as a quality indicator since it reduces the number of effective parent changes, which maintains the stability of the network with less frequent unnecessary changes.

In fact, since the PAI comprises the Ei and the Dp variations that are included as weighting variables under 1 to the LQI, it will not be possible for the PAI to reach high values as the LQI do. In fact, a given parent will usually have a PAI value lower than its LQI value. For that reason, the number of effective parent changes stabilizes at a certain moment when using the PAI, however in the case of the LQI it keeps on increasing.

6.2.4. Effects of the Hysteresis Factor K

The hysteresis factor K represents the expected improvement that the new parent of the child device should have in order for the switching process to happen. This parameter will not affect the number of triggers of the confirmation phase or of the parent changes requests since it is independent from those factors. It only affects the number of effective parent changes. It is quite obvious that the number of parent changes will be much lower when using the PAI than when using the LQI, as demonstrated in the previous section. Figure 6.10 presents the number of effective parent changes as a function of the hysteresis Factor K and argues the previous intuition. Figure 6.10 depicts the variation of parent changes as a function of K.

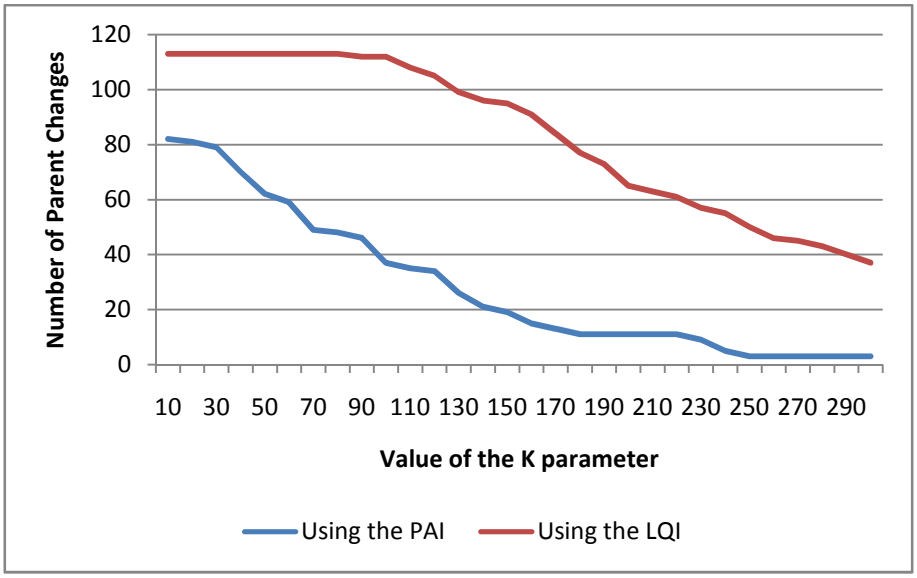


Figure 6.10: Number of parent changes in function of the value of K

As observed from figure 6.10, the number of effective parent changes for different values of K when LQI is used as a quality indicator is greater than the one with PAI.

6.3. Conclusions

The simulation results demonstrated the feasibility of the proactive re-association approach and evaluated it behaviours for different parameters settings. It also presented the advantage of using the PAI as a quality indicator instead of the LQI in order to assess the real quality of a parent. In fact, it has been shown the use of the PAI results in better and less frequent effective parent changes. Moreover, the simulation confirmed the expectations concerning the influence of the different parameters used in the proactive re-association mechanism and offered a means to choose the appropriate value of a given parameter as a function of a prefixed behaviour of the mechanism.

Chapter 7:

Implementation Guidelines

In order to assess the real behaviour of the proposed mechanisms and confirm the theoretically processed inaccessibility times, we opt to implement the two proposals within open-ZB, an open source ZigBee compliant protocol stack developed at IPP-Hurray! In this chapter, we start by describing TinyOS, NesC and the open-ZB architecture and then present some implementation guidelines that should be followed when implementing the proposed mechanisms. We conclude this Chapter by proposing relevant deployment strategies to be considered when deploying both proposals in real environments.

7.1. Introduction

When implementing the proposed re-association mechanisms, we take into consideration the specific constrains and behaviours of the IEEE 802.15.4/ZigBee standards. In this Chapter we present the adaptations that have to be applied to the NWK layer and MAC sub-layer of the IEEE 802.15.4/ZigBee standard in order to implement both proposed mechanisms. The implementation integrates into open-ZB [23] which is an open source implementation of the IEEE 802.15.4/ZigBee protocol stack developed within the IPP-HURRAY Research Group in the context of the ART-WiSe ([24], [25]) framework. The implementation will use the existing open-ZB implementation and add the newly introduced mechanisms within it.

7.2 TinyOS and NesC

In order to function, WSN nodes need specific operating systems and programming languages that enable flexible software development for the multiple applications of WSNs. One of the widely used operating systems for WSNs is TinyOS [26] which is an event-driven operating system. TinyOS is developed in nesC [27], a language for programming structured component-based applications. NesC has a C-like syntax and is designed to express the structuring concepts of TinyOS. This includes the concurrency model, mechanisms for structuring, naming and linking together software components into embedded system applications. The component-based application structure provides flexibility to the application design and development. NesC applications are built out of components and interfaces.

The component defines two target areas:

- *The specification*, a code block that declares the functions it provides (implements) and the functions that it uses (calls);
- *The implementation* of the functions provided.

The interfaces are a bidirectional collection of functions provided or used by a component. The interfaces *commands* are implemented by the providing component and the interface *events* are implemented by the component using it. The components are "wired" together by the means of interfaces forming an application.

TinyOS defines a concurrency model based on tasks and hardware event handlers/interrupts. The TinyOS tasks are synchronous functions that run without pre-emption until completion and their execution is postponed until they can be executed. Hardware events are asynchronous events that are executed in response to a hardware interrupt and also run to completion. As all the asynchronous tasks and events may pre-empt running code, the nesC software components are predisposed to race conditions. Due to the nesC compiler report of data races during compile time, these can be avoided either by accessing shared data exclusively within tasks, or by having all accesses within atomic statements.

7.3 Overview of the open-ZB implementation architecture

The open-ZB development efforts include the implementation of the IEEE 802.15.4 protocol and the ZigBee network layer. Nevertheless, the future objectives of it are to implement the full IEEE 802.15.4 protocol stack and the full functionalities of the ZigBee NWK layer. The first implementation of the IEEE 802.15.4 only supported the Crossbow MICAz [28] motes and it was conditioned to that hardware architecture. The latest version also supports the Crossbow TELOSB.

The open-ZB implementation has three main areas, the development of the hardware abstraction layer, including the IEEE 802.15.4 physical layer and the timer module supporting both MICAz and TELOSB mote platforms; the IEEE 802.15.4 MAC layer; and the ZigBee Network Layer.

Figures 7.1 and 7.2 present the layered view of the different TinyOS components and interfaces of the IEEE 802.15.4/Zigbee protocol stack implementation. The organization in modules is to enable fast and easy extensions to the implementation by adding or updating new functionalities. Each of these modules makes use of auxiliary files used to implement some generic functions (e.g. functions for bit aggregation into variable blocks), constants declaration (e.g. layer constants), enumerations (e.g. data types, frame types, response status) and data structure definitions (e.g. frame construction data structures).

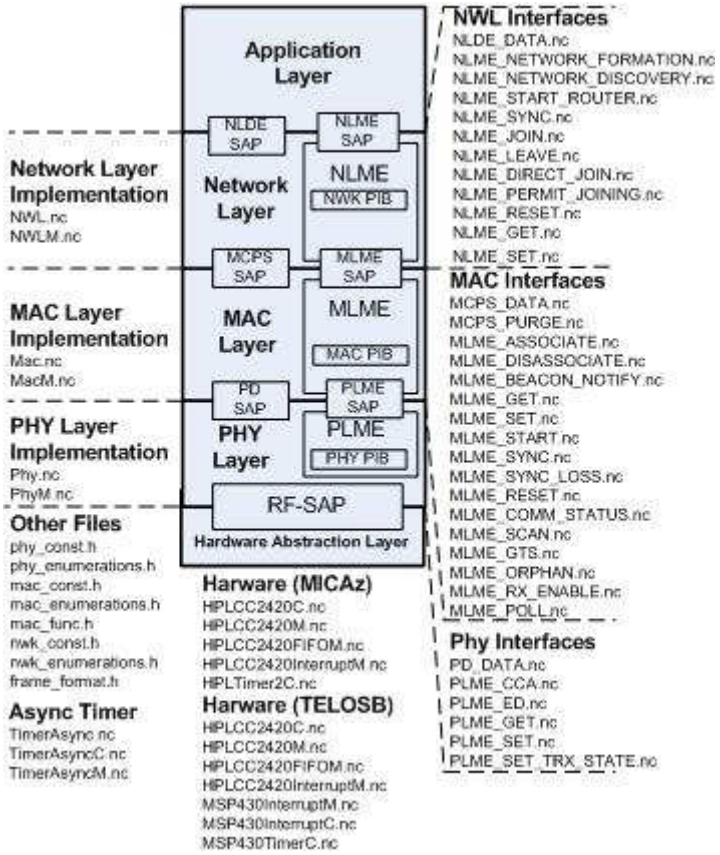


Figure 7.1: Protocol stack software architecture

The interface files (Fig 7.1, right side) are used to bind the components and represent one Service Access Point (SAP). Each of these interfaces provides functions that are called from the higher layer module and are executed/implemented in the lower layer module. The interfaces also provide functions used by the lower layer modules to signal functions that are executed/implemented in the higher layer modules (e.g. the PD_DATA.nc interface is used by the MacM module to transfer data to the PhyM module, that is going to be transmitted, and also enables the signalling by the PhyM in the MacM of received data).

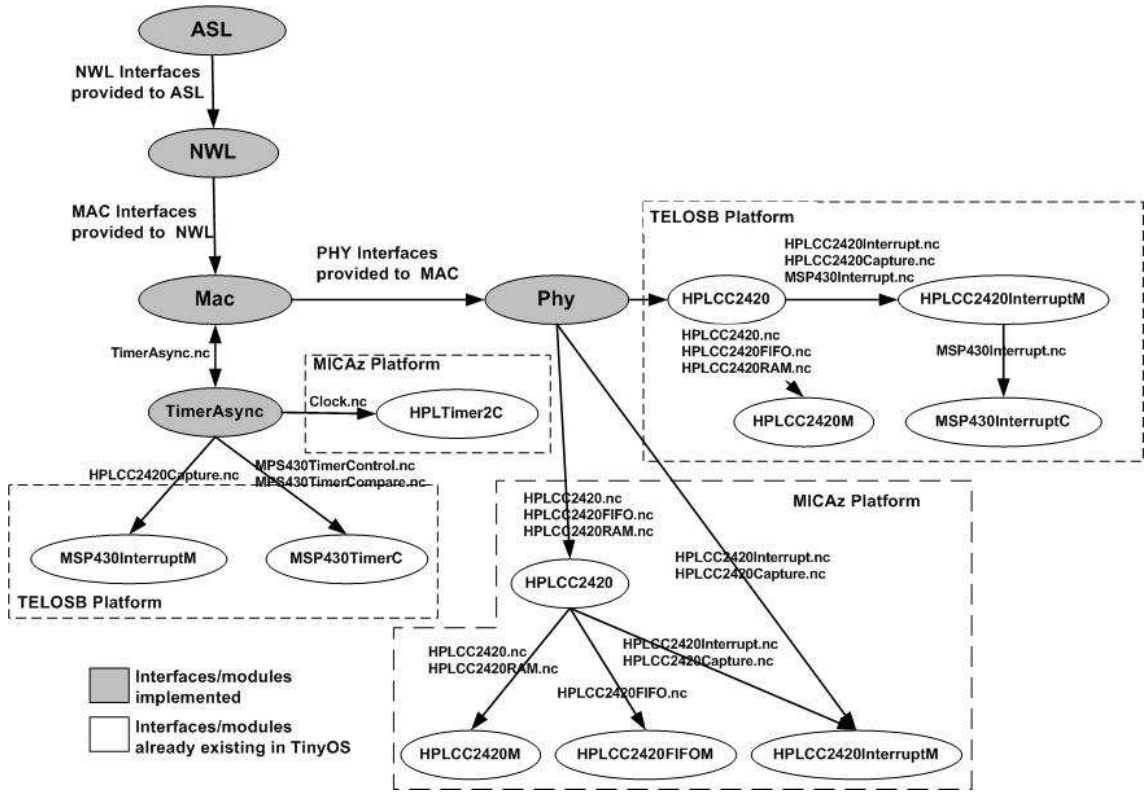


Figure 7.2: TinyOS implementation diagram

Figure 7.2 depicts the relations between different components of the IEEE 802.15.4/Zigbee protocol stack implementation. Note that some components used in this implementation are already part of the TinyOS operating system, namely the hardware components.

In this implementation, there is no direct interaction with the hardware, in fact, TinyOS already provides hardware drivers forging a hardware abstraction layer used by the Phy component. In Figure 7.3., observe that the components highlighted in white are hardware components already provided by the TinyOS operating system.

Refer to [29] for a detailed description of the implementation functions, variables and protocol mechanisms.

7.4. Integration of the fault-tolerance mechanisms within open-ZB

As explained earlier, both the proactive and reactive mechanism are located in the NWL and can be centralized in a unique fault-tolerance module that interacts with the higher and lower layers using the standard defined procedures. This is very interesting since that it minimizes the changes to the standard and simplifies the programming and the debugging processes. Figure 7.3 depicts the localization of the fault-tolerance module.

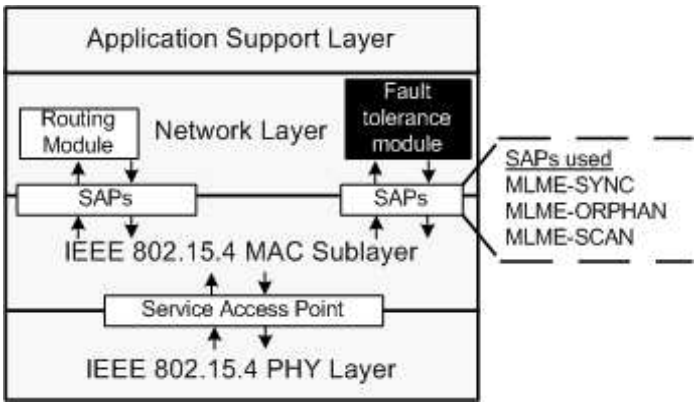


Figure 7.3: Fault tolerance module localization within the protocol stack

The fault-tolerance module uses the following Service Access Points to interact with the MAC sub-layer:

- MLME_SYNC [page 104 of ref. 6] – Used to enable the MAC layer to start synchronizing the coordinator by always keeping track of the beacon. In uses the *LogicalChannel* parameter to select a channel from the available logical channels supported by the PHY, and the Boolean *TrackBeacon* parameter, set to TRUE if the device wants to track all next beacons or to FALSE if device wishes to synchronize only with the next beacon.
- MLME_SCAN [page 92 of ref. 6] - Implements the channel scan mechanism in order to inform the upper layer about the energy detection on each channel. It uses the *ScanType* parameter in order to specify the performed scan type (i.e.. active, passive or orphan), the *ScanChannels* parameter that defines the channels to be scanned and the integer *ScanDuration* parameter that specifies the time spent scanning each channel.
- MLME_ORPHAN [page 84 of ref. 6] - Used by the coordinator to inform the higher layer of the presence of an orphan device. It uses the *OrphanAddress* parameter containing the extended address of the orphan device, the Boolean *SecurityUse* parameter to decide of the extended address uses security and the integer *ACLEntry* parameter that defines additional security features.

It is important to notice that those SAPs are standard defined and are already implemented within open-ZB.

7.4.1 PAI calculation

After performing the channel scan, the device processes the PAI indicator for all potential parents. A potential parent node is defined as a full function device (FFD) that fulfils the following conditions:

1. Is a ZigBee router or a ZigBee coordinator

2. Permits joining for other devices
3. Not a current child or sibling of the orphan node.

This preliminary information about the parental capabilities of the neighbour devices is found in the following fields of the neighbour table:

- *Device type* indicator: varies from 0x00 - 0x02 and specifies the type of neighbour device (PAN coordinator, router or end device).
- *Permit joining* indicator: specifies if the neighbour device is accepting associations from other devices that want to join the network through it.
- *Relationship* indicator: specifies the relationship between the current node and the neighbour device. It takes values from 0x00 to 0X04 to indicate if the distant device is either: a parent, a child, a sibling, a previous child or none of the above. In order to prevent relationship conflicts potential parents of the child device cannot be one of its children or siblings

It is important to notice that the *Relationship indicator* may not be used since it does not give sufficient information about the parent (e.g. the depth of the potential parent, its current cluster…). That is why it is possible to use the Cskip function that gives extended information about the position of the neighbour device in the network topology but also determines the address pool, the current cluster and the depth of the potential parent device.

The standard specifies that a parent is chosen with regard to its LQI. The higher is this parameter, the best is the parent. However, as pointed out in Chapter 4, this is not the best way to choose a parent since the LQI does not include key quality factors such as energy, traffic and load indicators. That is why implementing a new parameter to choose the potential parent is appealing but requires adequate changes to the standard.

The results of the PAI calculations shall be stored in the *nwkNeighborTable* that will be adapted in order to accept this new indicator. The PAI is an integer variable that characterizes all potential parents.

The PAI will retrieve its different parameters from the *nwkNeighborTable* (for more information on the different fields of this table, refer to Annex B) and the PAN descriptor table (table 7.2). In this case we are going to use the "*extended neighbour table*" that includes more information than the standard one and that will be useful in the calculation of the PAI we propose. This information is found in the following fields of the *nwkNeighborTable*:

- *Transmit Failure:* is included in the *neighbour table* and is relative to the performance of the link between the nodes during a given period of time since it keeps track of the number of

failures that occurred during the transmission of previous frames. The higher this value, the more errors occurred and so the worse the link is.

- *Depth of the parent:* It can be deduced form the standard neighbour table by using the network address and the Cskip function (refer to Annex A for more information about the Cskip fuction). It can also be simply retrieved by using the additional fields of the extended neighbour table.
- *Energy Indicator:* The power status of a node is described in the ALP layer within a "Node Power" descriptor that is mandatory and unique on each node of the network. It gives an up to date indication of the power status of the node. It contains the following fields: "*Current power mode*", "*Available power sources*", "*Current power source*" and "*Current power source level*". The last indicator is very interesting since it provides information about the remaining power in the node (100%, 66%, 33% or critical). This information is used by the PAI indicator when calculating its value. Obviously this information is only interesting when the nodes are battery powered, which is generally the case of ZigBee WSN.

  Another way to include the power indicator in the PAI is the use a power information that is located in the NWK layer (Page 333 of ref 6). In fact, when a frame experiences transmission problems, a route error command frame is sent including an error code. One of those error codes is "Low battery level". So when the node detects that the last frame failed to be send due to that specific error, it deduces there is a power problem and reports it in the PAI equation by decreasing its value.

Moreover, the *PAI* needs the *LQI* indicator which is a field included in the beacon or data frames received from the neighbouring devices during normal operation or channel scan.

The node will process the *PAI* of the potential parent device and store that information in the neighbour table within the appropriate field as described in table 7.1.

Field Name	Field Type	Valid Range	Description
PAI	Integer	0x00 – 0xFF	The Parent Assessment Indicator that will be used by the child device in order to choose the best new parent. The higher this value, the best the potential parent

Table 7.1: Added variable to the neighbour table

Moreover, the different weighting variables that will be used during the calculation of the *PAI* have to be stored in the node's memory as explained in table 7.2.

Field Name	Field Type	Valid Range	Description
a, b or c	Integer	0x00 – 0xFF	The weighting variables used in the calculation of the PAI. Refer to Chapter 4 for more information about them.

Table 7.2: The weighting variables specifications

7.4.2. The Reactive Mechanism

The reactive mechanism defines a new scan procedure that can be considered as an enhanced version of the orphan scan already described in the IEEE 802.15.4. It induces minor changes to the MAC sub-layer specification since it defines a new scan procedure called the *Reactive Orphan Scan*. On activation of the mechanism (link break with the current parent) the NWK layer sends an *MLME-SCAN.request* request to the MAC sub-layer to launch the *reactive orphan scan*. This scan will send beacon request frames on each channel and wait for a response from the devices within the Personal Operating Space (POS) of the child device. This scan provides two results:

(1) The PAN descriptor of the parent

(2) The PAN descriptors of the possible parents

The different fields of the PAN descriptor are exposed in Annex C.

On reception of these *PANDescriptor* elements, the device will perform the decision phase exposed in Chapter 4 within the NWK layer and send:

- An association request to the current/new parent if it found respectively, its current parent or a new parent device.
- An error indication to the higher layer to inform it that the device is orphan and out of range of potential parent device in case it found neither its current parent nor other possible parent device.

7.4.2. The Proactive Mechanism

As in the case of the reactive approach, the pro-active re-association mechanism is implemented in the NWK layer and uses the standard defined interfaces to communicate with the adjacent layers. In addition, it does not affect the functioning of the MAC sub-layer.

The proactive mechanism effectively starts after the confirmation phase is validated. The child device will call a passive scan procedure during its superframe inactive period using the standard SAP *MLME-SCAN.request*. The child device sets the *ScanChannels* parameter to the number of channels supported by the PHY layer in order to go through all the defined channels and ensures that the scan duration lasts long enough to detect all potential parents. During the channel scan, the device may find potential parent devices. As soon as it gets all the information regarding them, it processes the PAI associated to them and store it in the appropriate field of the newly modified *nwkNeighborTable*. If a suitable new parent was found, the child device sends a *MCPS-DATA.Request* command with a *NWK_Leave-indicator* to the MAC sub-layer in order to disassociate from its current parent and a *MLME-ASSOCIATE.Request* to the MAC sub-layer to associate to a new elected parent. After confirmation of the previous request, by a *MLME-START.Request* is sent to start normal operations.

In contrast with the reactive re-association mechanism that imposes minor changes in the MAC sub-layer, the proactive mechanism only uses the MAC available mechanisms.

7.5. Deployment Strategies

After implementing the fault-tolerance module within all or some of the nodes of the network, and in order for the mechanisms to behave correctly, the network designer has to take into consideration a given number of requirements, mainly the density of the network and the way to correct ZigBee coordinator failures.

7.5.1. Density of the sensor network

It is important to always keep in mind the fact that, in order to function, the network should be densely populated. This means that the number of nodes in the networks should be sufficiently high so that every node implementing the fault-tolerance mechanism processes at least one other potential parent device in his POS. If this condition is not respected, the proposed mechanisms will be inefficient. Finding an optimized fault-tolerant topology can be considered as a potential future work.

7.5.2. ZigBee Coordinator Failures Repairing

The proposed fault-tolerance techniques are useless if the ZigBee coordinator that ensures communication between the sensor network and the data sink fails. In fact, ZigBee coordinator failures are critic since they will prevent totally the network from functioning. That is why it is very important to find out a suitable way to introduce fault-tolerance mechanisms that enable the network to recover from its coordinator failure. In the following we draw some possible proposals.

7.5.2.1. Hardware redundancy

A simple and reliable approach to tackle the PAN coordinator (PANC) failure would be to install hardware redundancy within the gateways that interface the ZigBee network with the higher tier network or any other data sink device.

A first approach is to install multiple FFDs within the same gateway (the device interfacing the sensor network with the data sink system) where one will be functioning and the others in waiting mode. When the first device experiences problems, a second device performs its role. In order to minimize the time the gateway is unavailable, a cloning mechanism is implanted between the different FFDs of the gateway so that it creates continuously exact images of the active PANC on the spare PANC. All the FFDs share the same interface to the higher tier or data sink.

A second approach is to spare gateways composed of an FFD and an interface to the higher tier or data sink. The same imaging mechanism has to be implemented between the active and the spare gateways in order to clone the data, thus minimizing the down time of the gateway system.

7.5.2.2. Electing a new PAN from the 1st level router set

In this proposal, the current PAN coordinator supervises the signal strength of its neighbouring routers with a depth equal to 1 and finds out the best one. It informs the designated router that it can be the PAN coordinator in case the first one fails by activating a given indicator that is created in that aim.

Once the PAN coordinator fails; the elected router begins functioning as the new coordinator and informs its neighbour routers that it is the new coordinator.

It is important to know when to decide that the PAN coordinator is failing; in fact, the router can take bad decisions, thinking that the PAN coordinator failed but it did not.

Notice that if the newly elected PAN coordinator is not in range with all the level 1 routers, there will be creation of multiple PANs depending on the number of "not in range" routers.

The designation of the potential new coordinator can be done with regard to the signal level or another hybrid indicator taking into account the performance of first the level routers.

In order for the proposal to function correctly, we need to introduce novel sensing and decision procedures on all routers of the network (since that every router can become a new coordinator, that will need to perform the above described operations). That's why a major issue will be to evaluate to what extent the new procedures and parameters we are going to introduce will consume power, memory and CPU resources.

7.5.2.3. Contenting 1st level routers to become PAN Coordinator

When a router notices that the PAN coordinator has disappeared for a given period it initiates the "*orphan scan procedure*". If that procedure fails, it performs a channel scan to see if there is another coordinator to which it can associate and by that way rejoin the PAN. If no such a coordinator is found, it will wait a random duration before listening again to the channel. If there are still no coordinators found, the router elects itself as a PAN coordinator and reinitializes its communication with its child devices based on the new information.

In order not have a PAN Id conflict, we can use the "*PAN identity conflict resolution*" mechanism (page 147 of [5]) after the assignment of the new PAN Id to the newly elected PAN coordinator. The coordinator and the devices that are attached to it perform this procedure.

The drawback of such an approach is the generation of conflicts when electing a new coordinator this will result in sending more messages through the network thus increasing delays and energy consumption of the nodes.

7.6. Conclusion

In this section we presented general implementation guidelines for the proposed mechanisms. The advantage of this implementation is that it preserves backwards compatibility with other nodes that do not implement the previously defined mechanisms; this enables the coexistence of different device types within a same network. This property allows a high flexibility for the network architecture and design. Furthermore, these guidelines will enable easy and rapid implementation within the IEEE 802.15.4 / ZigBee standard. Real experimentation of the proposals based on these implement guidelines will provide us with valuable information about the real behaviour and performance of the mechanisms.

Chapter 8

General Conclusions and Future Works

In this work, we have analyzed existing strategies for improving the reliability of Wireless Sensor Networks (WSNs). We have particularly addressed the case of IEEE 802.15.4/ZigBee cluster-tree WSNs, figuring out how these protocols react to a degraded link quality between a child and its parent (ZigBee Router) and proposing two different fault-tolerance mechanisms that enhance the native orphan realignment mechanism by reducing or even eliminating network inaccessibility times.

First, we outlined some important concepts, terminology and existing work related to fault-tolerance in WSNs, highlighting the case of cluster-tree topologies. Then, we proposed and specified two re-association mechanisms: one that reacts to the loss of connection with a parent – the reactive mechanism, and another one that heuristically predicts a potential link break, enabling a node to associate to another parent before getting orphan – the proactive mechanism. We proved that network inaccessibility times can be reduced or eliminated, improving on the default ZigBee mechanism. Finally we presented simulation results for the proactive approach using a simulation tool that we have developed and also outlined some implementation guidelines for both proposals.

These mechanisms offer important add-ons to the standard specifications. In fact, they reduce inaccessibility times by offering faster re-association procedures and decrease communication errors by electing the novel parents based on a multi-dimensional function – the Parent Adoption Indicator (PAI) – that has been shown to be more efficient than the default Link Quality Indicator (LQI) provided by the IEEE 802.15.4 Physical Layer. Nevertheless, these mechanisms remain backward compatible, thus allowing the coexistence of devices implementing these mechanisms with others that do not.

Mobility support shows up to be an important requirement in emerging and future ubiquitous computing applications. Thus, we want to tackle that issue in the ART-WiSe research framework. In

this context, we believe that the proactive re-association mechanism can offer an interesting potentiality for supporting mobility in WSN.

In the short term, we intend to carry out the implementation of the proposed mechanisms in the open-ZB protocol stack and to develop experimental tests for assessing their behaviour in real situations, measuring their effective timing performance. Moreover, as most fault-tolerance techniques, the proposed mechanisms may increase the overall resource consumption of network, namely energy consumption; nevertheless, they may also decrease the energy needed for data transmission, since the links offer better quality (thus avoiding or leading to less retransmissions). Therefore, it is worthwhile to analyze the overall energy efficiency/consumption resulting from the proposed mechanisms, finding suitable tradeoffs between energy consumption and reliability. Between many other interesting and promising ideas for future work, we may still outline the interest of analyzing the adequateness and feasibility of the proposed mechanisms (or finding alternative ones) for ZigBee mesh WSNs.

Annex A: The ZigBee Network Layer

A.1. Introduction

The NWK Layer is of great importance in the context of this work since that it is within it that the proposed fault tolerance mechanisms are implemented. In this annex we present the different aspects of the NWK layer such as the network topology, the frame formats, the standard defined mechanisms and procedures and other relevant information about routing and beaconing.

A.2. Network Topologies

A.2.1. The Star Topology

Star networks are centred on a PAN coordinator that communicates with the devices. Communication between child nodes is not possible. PAN coordinators in this topology are usually mains powered since they have to be continuously on to receive the information sent by the child nodes. That's why it appears that this topology is more suitable for applications such as: personal health care, peripherals communicating with a computer or in toys and games.

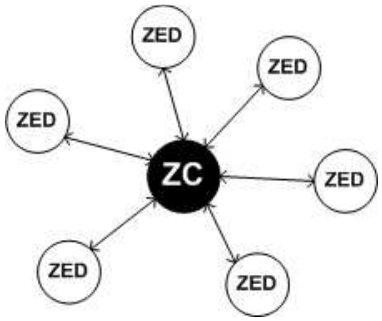


Figure A.1: The Star Topology

A.2.2. The Mesh Topology

This network is formed by a PAN coordinator that ensures the control of the network, and a set of routers and end devices that communicate together on a peer-to-peer fashion. RFDs can only communicate with a unique FFD to which they are associated, that FFD is called their parent. On the other hand, FFDs can communicate between each other without any problem.

In this topology, information is relayed from node to node throughout the network until it reaches the data sink.

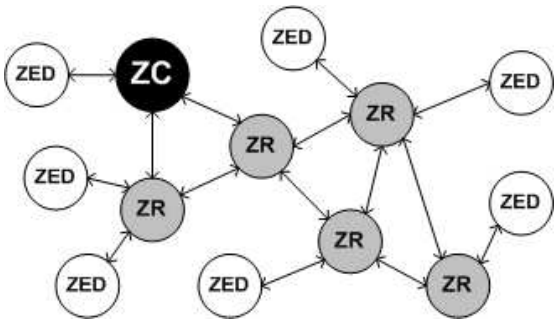


Figure A.2: The Mesh topology

A.2.3. The Cluster Tree Topology

This network is a specific case of the mesh topology but has some particular features. It is also composed of a PAN coordinator to which connect routers and end devices. Each router can create a cluster and become the cluster head. Other routers that are connected to him can also become cluster heads by electing themselves and creating a cluster to which will be connected other routers and devices. Clusters are identified by an Id. The final structure will be a tree where the root will be the PAN coordinator and the clusters will constitute the branches and the end devices will form the leaves.

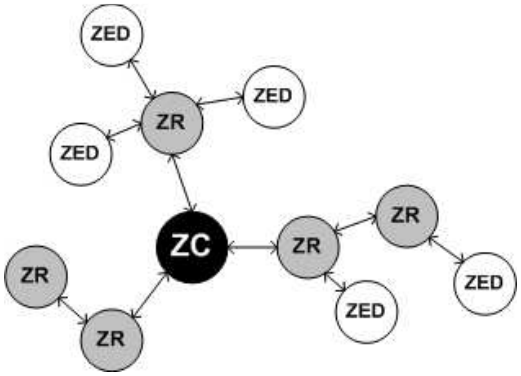


Figure A.3: The Cluster Tree topology

A.3. Network Layer Frame Format

A.3.1. Frame Format

A header and a payload of variable lengths form all of the Network Layer frames (NPDU). The header contains specific information that is used by the network layer to trigger or control layer specific procedures and functions and the payload contains the useful information for the higher layers and is of variable length as is shown in table A.1.

Octets: 2	2	2	1	1	0/8	0/8	0/1	Variable	Variable
Frame control	Destination address	Source address	Radius	Sequence number	Destination IEEE Address	Source IEEE Address	Multicast control	Source route subframe	Frame payload
NWK Header									Payload

Table A.1: Network layer general frame format

The NWK header contains the following fields:

- Frame Control: Is the first field of the frame and is 2 bytes long. It contains information about the frame type and the following fields of the header such as: the protocol version, a multicast flag (1 if activated), Security enabling (1 if activated), Source route enabling (set to 1 if present in NWK header), IEEE long addresses usage. The different subfields of the field are found in table 2.

Bits: 0-1	2-5	6-7	8	9	10	11	12	13-15
Frame type	Protocol version	Discover route	Multicast flag	Security	Source Route	Destination IEEE Address	Source IEEE Address	Reserved

Table A.2: The frame control format

- Destination address: is the always present 2 bytes short address of the destination device that the device receives after it associates to a given PAN.
- Source Address: is the always present 2 bytes short address of the Source device once it is associated to the PAN.
- Radius: Is one octet long and is analog to the Time to Live field in TCP. It specifies the range of a radius transmission and is decremented by one at each hop.
- Sequence Number: Takes track of the number of transmitted frames. It is incremented by 1 at each new frame transmission.
- Destination IEEE address: the 64 bits long unique IEEE address of the destination device. Is used only when the short destination address is not known.
- Source IEEE address: the 64 bits long unique IEEE address of the source device. It is only used when the 2 bytes short address is not known, i.e. when the device is not yet associated to the network or lost his short address for any reason.

- Multicast Control: Is only present when the multicast subfield is set to 1. It is divided into 3 subfields specifying the multicast mode (member mode or non member mode), the range of the member group multicast (NonmemberRadius) and the maximum value of the range of the NonmemberRadius field.
- Source Route Subframe: is only present when the source route indicator in the frame control in set to 1.

A.3.2. Frame Types

There exist 2 different frame types:

- *Data Frames:* They conform to the general NWK frame we described in 3.a. The header contains the frame control field with the value indicating it is a data frame, an appropriate combination of addresses and broadcast fields. The payload is a sequence of bytes that will be used by the higher layers.
- *Command Frames:* They conform to the general NWK frame format described in 3.a. The header contains the frame control field, an appropriate combination of addresses and broadcast fields. The payload starts with a NWK Command Identifier that is one byte long and indicates the NWK command type being used and the command payload that contains the command itself. There exist 7 types of Frame commands: Route request, Route reply, Route error Command, Leave command, Route Record Command, Rejoin Request and Rejoin Response.

A.4. Constants and Attributes

The network layer maintains a number of constants and attributes that are useful to monitor the state of the layer and in taking decisions concerning it.

- *Constants:* A given number of constants define the characteristics of the network layer, such as: *nwkcCoordinatorCapable* that indicates if the device is capable of being a coordinator, nwkcMaxDepth that indicates the minimum number of logical hops form the coordinator the device can have, *nwkcProtocolVersion* that gives the version of the ZigBee NWK protocol in the Device, *nwkcWaitBeforeValidation, nwkcRepairTheshold, nwkcRouteDiscoveryTime, nwkcMaxBroadcastJitter….* More details about these constants can be found in amendment 3.6.1 of the ZigBee Standard.
- *Attributes:* The Network Information Base (NIB) contains attributes required to manage the NWK layer of the device. Each of the attributes can be written and read. These attributes are listed in amendment 3.6.2 of the ZigBee standard as for example: *nwkSequenceNumber*,

nwkMaxDepth, *nwkMaxChildren*, *nwkMaxRouters, nwkNeighborTable, nwkRouteTable, nwkCapabilityInformation, nwkShortAddress, nwkProtocolVersion…*

A.5. Functions and Procedures

Since the NWK layer is responsible of data transmission, network formation and maintenance, addressing routing, the standard defines the different procedures that can be performed by this layer. In the following sub-chapter we will review the different functions and procedures of the NWK layer.

A.5.1. Transmission and Reception

Only the devices that are currently associated to the network can send data frames from the NWK layer. If a device receives a request to transmit while it is not associated, it notifies the higher layer by sending an INVALID_REQUEST error.

The transmission uses the MAC sub-layer data service. Before transmission, the NWK layer creates the data frame as specified in 1.3. If security services are needed, the frame will be transmitted to the security service provider for processing. When a frame is constructed and ready for transmission it is passed to the MAC data service. The MAC responds by issuing MCPS-DATA transmission to return the result of the transmission.

In order to receive data, a device must enable its receiver. The higher layer initiates receiving by the NLME-SYNC.resquest primitive. In beacon enabled networks, receipt of this primitive will cause a device to synchronize with his parent's next beacon and, optionally, track future beacons; in a non beacon enabled network this will cause the NWK layer to poll the device's parent, in those networks, the receiver should be enabled whenever the device is not transmitting. On a beacon-enabled network, the NWK layer should ensure that the receiver is enabled when the device is not transmitting during the active period of its own superframe and of its parent's superframe. The NWK layer may use the macRxOnWhenIdle attribute of the MAC PIB for this purpose. Once the receiver is enabled, the NWK layer will begin to receive frames via the MAC data service. On receipt of each frame, the radius field of the NWK header shall be decremented by 1. If, as a result of being decremented, this value falls to 0, the frame shall not, under any circumstances, be retransmitted. It may, however, be passed to the next higher layer or otherwise processed by the NWK layer. The device can either use the information contained in the frame by passing it to the next higher layer, or retransmit it again to a given destination if it has router abilities.

A.5.2. Network and Device Maintenance

All the ZigBee devices must be able to join or leave a network. In addition, ZigBee routers or coordinators must permit the devices to join and leave the network by association, participate in logical network address assignment and the maintaining a table of neighbouring devices.

Some of the functionalities are presented in the following:

- ***Establishing a new network***

This can only be initiated by coordinators that are not currently member of a network:

1. Energy detection scan by MAC on one, a set, or the complete set of available channels.
2. Order the channels according to increasing energy levels discarding those who are above than a given level (implementation specific).
3. Perform active scan on the selected list of channels to search for other ZigBee devices.
4. In order to determine the best channel on which to establish the network, an overview of the PAN descriptors is made; we choose the channel with the lowest number of PANs.
5. If no channel was found then notice the higher layer.
6. If suitable channel found
    a. Choose a PAN Id < 0x3fff and not yet in use in the selected channel.
    b. If no PAN Id could be chosen, issue an error message to the higher layer.
7. If the above is successful, the PAN operation is launched.

- ***Joining a network***
    - *Using the MAC layer association procedure*

This procedure must take place in both the parent and the child device. From the child device point of view the following tasks are executed:

1. The MAC association procedure is initiated by the NLME-NETWORK-DISCOVERY.request primitive that specifies the channels to be scanned and the scan duration.
2. The network layer issues a primitive asking the MAC to perform an active or passive scan.
3. Each beacon frame received during the listening period with a payload of zero length will generate an MLME-BEACON-NOTIFY.indication to the network layer containing addressing information and association possibilities.

4. The NLME checks the protocol ID to verify that it matches the one of ZigBee, if not, the beacon is ignored; if it corresponds, the information contained in the beacon will be copied to the “neighbour table”.
5. Only the devices that are not associated to a network can attempt to associate. The network layer searches on the neighbour table if there exists a suitable parent device, such a device should have a link cost less than 3.
6. If there is no suitable parent in the table an error message is send the NMLE, if there is more than one suitable parent, the device chooses the one with the minimum depth from the ZigBee coordinator, if there are more than one, the device is free to choose.
7. When a suitable parent is found, an association command is sent to the MAC layer containing addressing information of the parent found in the “neighbour table”.
8. If an association failed, it can be because the parent we are attempting to associate to, has already reached the maximum associated routers number it can handle. In that case, the NWK layer can try to connect as an end device. If that association failed, the device can connect to another router in his “neighbour table” (if it exists). If the association is successful, a 2 bytes short address is acquired to transmit over the network.

From the parent device point of view:

1. The parent verifies that the device is already registered to his network by comparing 64-bits logical address existing in his “neighbour table” with the device’s address. If a match is found, the parent assigns a 16 bits address to the new device if an address is available in the address space.
2. When the request is granted, a new entry for the child device is created in the “neighbour table” of the parent.
3. Shall there be any error, it will be reported to the higher or lower layer using the appropriate primitives

    - *Through the NWK join procedure*

    This procedure gives the opportunity for a node that has lost all connectivity with the network. It allows a device to rejoin a network that does not allow new devices to join. There is no use of the MAC association procedure.

    The Child procedure:

6. Is initiated by the NLME_JOIN.request primitive (a parameter can specify of the device wants to join as router).
7. MAC layer makes a passive or active scan of the channels.
8. Test of the received beacon and registration of information in the “neighbour table”.
9. On scan complete, choose the best parent device and rejoin.
10. If rejoin was unsuccessful, the NLME shall find a new suitable parent to join to in the table.

The Parent procedure is similar to the one described earlier.

- *Through orphaning*

In this case, a device is directly added to the network by a previously designated parent device. Only ZigBee coordinator/router may initiate this procedure, otherwise the higher layer would be notified by an invalid request notification.

1. Parent verifies if the specified device already exists in the network by searching in the neighbour table.
2. If not match is found: allocate a 16 bits network address to the new device if possible.
3. Now that the parent has created a link with the child device, the child should complete the establishment of the parent-child relationship by initiating the orphaning procedure.

- ***Leaving a network***

A device can initiate it’s own removal: A device can remove itself from the network by his own:

On receipt of the demand from the higher layer, the network layer asks the MAC to issue a leave command frame. Once the MAC answers the NWK layer returns a confirmation to the higher layer with a status parameter indicating the success or the failure of the procedure.

This procedure is of special interest in the proactive fault tolerant mechanism, since when a node wants to associate to a new better parents, it should use the self-disassociation procedure to re-associate to the new parent. It is important to know that disassociating form a parent is similar to leaving the network.

A parent can remove its child from the network: When the NWK layer receives a command from the higher layer with the 64-bits IEEE address of the child device and the silent parameter set to FALSE, the NWK layer sends a leave command frame with the 16 bits short address of the child device to

disassociate. If the transmission of the frame was successful, the NWK layer issues a confirmation command to the higher level, otherwise it indicates the type of error that occurred. Once the device removed, the NWK layer of the parent device has to make sure that the entry of the left device in the "neighbor table" and all other data structure that refer to the left child.

- ***Neighbor tables***

They exist on every device and contain information about other devices in RF range as: Link state information, device type, candidate parents and transmit failure rates. It is useful in two contexts, first, it is used during network discovery or rejoining to store information about routers within RF reception range that may be candidate parents, second, after joining the network it stores information about the relationship and link state information about in range devices.

It is updated each time the device receives a frame from the corresponding device.

These tables are very useful since they keep track of possible new parents that can be chosen from in case of failure of the first elected parent. Ordering the nodes in a decreasing way of reception performance will permit the node to easily select the best new parent node. Annex B presents the different entries of the neighbour table.

- ***Permitting a device to join the network***

Only ZigBee coordinators and router can attempt to permit devices to join the network. This procedure permits the application layer to fix a duration to the NWK and MAC layers during which other devices can join the network.

- ***Short Address assignment***

The 16 bits short addresses are assigned by a distributed address assignment scheme. The PAN Coordinator is always assigned 0x0000 and has a network depth of 0. The assignment scheme needs the following parameters to work: The maximum number of children (Cm), the maximum number of child routers (Rm) and the depth of the network (Lm). Each parent device uses the previous parameters to compute the Cskip parameter:

$$Cskip(d) = \begin{cases} 1 + Cm \cdot (Lm - d - 1), & \text{if } Rm = 1 \quad \cdots\cdots\cdots\cdots (a) \\ \dfrac{1 + Cm - Rm - Cm \cdot Rm^{Lm-d-1}}{1 - Rm}, & \text{Otherwise} \quad \cdots\cdots\cdots\cdots (b) \end{cases}$$

Cskip is used to know the address pool of the child devices.

The Cskip parameter could be useful to determine relationships between the nodes since each pool of addresses indirectly identifies sibling devices. Identifying the pool of device is very interesting

since it permits to prevent conflicts between nodes that try to reach a new parent (this problem will be analyzed in an upcoming document)

A.5.5. Routing

ZigBee coordinators and routers must provide the following functionality:

- Relay data on behalf of higher layers or other ZigBee routers
- Participate in route discovery in order to establish routes for subsequent data frames or on behalf of end devices
- Participate in end-to-end and local route repair using the ZigBee path cost metric.

Coordinators and routers can also maintain routing tables, initiate route discovery on behalf of higher layers or other ZigBee routers and initiate end-to-end and local route repair.

Routing Cost:

The ZigBee routing algorithm uses a path cost metric for route comparison during route discovery and maintenance. In order to compute this metric, a cost, known as the link cost, is associated with each link in the path and link cost values are summed to produce the cost for the path as a whole. Details of the route cost calculus are implementation specific.

Routing tables:

A routing table is constructed and maintained as in table A.3:

Field Name	Size	Description
Destination address	2 bytes	The 16-bit network address or Group ID of this route; If the destination device is a ZigBee router or ZigBee coordinator, this field shall contain the actual 16-bit address of that device; If the destination device is an end device, this field shall contain the 16-bit network address of that device's parent
Status	3 bits	The status of the route. See Table 3.47 for values
Many-to-one	1 bit	A flag indicating that the destination is a concentrator that issued a many-to-one route request
Route record required	1 bit	A flag indicating that a route record command frame should be sent to the destination prior to the next data packet
GroupID flag	1 bit	A flag indicating that the destination address is a Group ID
Next-hop address	2 bytes	The 16-bit network address of the next hop on the way to the destination

Table A.3: The Routing table format

Routing decisions are taken based on the routing table entries or the route discovery results. Frames are relayed from a device to another until arriving at the destinations.

Routing types:

Neighbor routing: is based on the neighbour tables. If the target device is in RF range, the message is sent directly.

Table Routing: is an implementation of the AODV Ad-hoc On Demand Distance Vector routing protocol, and is based on routing tables and path cost metrics calculations.

Tree Routing: is based on the address assignment schemes and hierarchical routing along the tree. The routing protocol acts as following:

- Each device, upon the reception of a data frame, reads the routing information fields and checks the destination address.
- If the destination address is equal to its own address, the device will signal the upper layer with the NLDE_DATA.indication primitive along with the frame payload as argument. If the destination is a child of the device (neighbour table check), the device relays the packet to the appropriate child address. If the destination address is not a child, the device must check if the address is a descendent using the following condition, being *A* the device network address, *D* the destination address and *d* the device depth in the network.

A < D < A + Cskip(d-1)

- The device address of the next hop when route down is given by:

$$N = A + 1 + \left\lfloor \frac{D - (A+1)}{Cskip(d)} \right\rfloor \times Cskip(d)$$

- If the destination address is not a descendant, the device will relay the packet to its parent.

A.5.6. Beaconing

Beacon scheduling is necessary in a multi-hop topology to prevent the beacon frames of one device from colliding with either the beacon frames or data transmissions of its neighboring devices. Beacon scheduling is necessary when implementing a tree topology but not a mesh topology, as beaconing is not permitted in ZigBee mesh networks. An appropriate scheduling method should be used in order to organize the transmission of beacons and by that means allow nodes to sleep when not waiting for a beacon. Therefore a device receiving a beacon frame shall know the beacon transmission time of both the neighbouring device and the parent of the neighbouring device, since the transmission time of the parent may be calculated by subtracting the time offset from the timestamp of the beacon frame. The receiving device shall store both the local timestamp of the beacon frame and the offset included in the beacon payload in its neighbour table. The purpose of having a device know when the parent of its neighbour is active is to maintain the integrity of the parent-child communication link by alleviating the hidden node problem. In other words, a device

will never transmit at the same time as the parent of its neighbour. In the process of joining the network, the new device shall build its neighbour table based on the information collected during the MAC scan procedure. Using this information, the new device shall choose an appropriate time for its beacon transmission and CAP (the active portion of its superframe structure) such that the active portion of its superframe structure does not overlap with that of any neighbour or of the parent of any neighbour. If there is no available non-overlapping time slot in the neighbourhood, the device shall not transmit beacons and shall operate on the network as an end device.

A.6. Communication Paradigms

Communication between nodes can follow different communication patterns:

Unicast communications

A Unicast communication is a point-to-point network connection between a source node and a destination node. The source node defines in the data frame the address of the destination. The only device that will process the information on the receiver side is the device possessing the destination address. This is the most common data transfer method when transferring useful data form a source node to the destination (or sink) node.

In cluster tree networks, all the data frames sent from the leaves to the sink are transferred on a unicast fashion.

Broadcast communications

In the actual (2006) ZigBee specification, broadcast across multiple networks is not supported. When a broadcast communication takes place, there is no use of the MAC sub-layer acknowledgement in non beacon-enabled networks, instead a passive acknowledgment method is used. The ZigBee coordinator and each ZigBee router shall keep a record of any new broadcast transaction that is either initiated locally or received from a neighbouring device. This record is called the broadcast transaction record (BTR) and shall contain at least the sequence number and the source address of the broadcast frame. The broadcast transaction records are stored in the broadcast transaction table (BTT).

Multicast communications

Multicasts provide many-to-many routing. Multicast addressing is done using 16-bit multicast group Ids. Each device has a multicast table indicating which multicast groups that device is a member of. A multicast message is sent to a particular destination group and is received by all devices that list that group's ID in their multicast table. Only data frames are multicast; there are no multicast command frames. Multicast frames have a mode flag that indicates whether they are in non-member mode or

member mode. Member mode is used to propagate multicasts between the devices that are members of the destination group.

A.7. Persistent data in the network layer

If for any reason, a device is reset, due to maintenance or accidentally, the following information have to be preserved in the network layer:

- The device's PAN Id and the Extender PAN Id,
- The device's 16-bits network address
- The 16-bits network address of each associated child
- The stack profile in use
- The values of nwkNextAddress
- The nwkAvailbleAddresses NIB attributes if used and the device's tree depth if distributed addressing is used

Although the ZigBee standard does not specify the use that can be done of these parameters, we can imagine that upon reset, these attributes can be used in order to get the device back to normal use as soon as possible without having to do all the initialization operations. For example, if the device goes down due to battery lack; once the battery is changed, the device restarts working seamlessly.

Annex B: Network Neighbor Table

Field Name	Field Type	Valid Range	Description
Extended address	Integer	An extended 64-bit, IEEE address	64-bit IEEE address that is unique to every device; This field shall be present if the neighbor is a parent or child of the device
Network address	Network address	0x0000 – 0xffff	The 16-bit network address of the neighboring device. This field shall be present in every neighbor table entry.
Device type	Integer	0x00 – 0x02	The type of the neighbor device: 0x00 = ZigBee coordinator 0x01 = ZigBee router; 0x02 = ZigBee end device This field shall be present in every neighbor table entry
RxOnWhenIdle	Boolean	TRUE or FALSE	Indicates if neighbor's receiver is enabled during idle portions of the CAP: TRUE = Receiver is on FALSE = Receiver is off This field should be present for entries that record the parent or children of a ZigBee router or ZigBee coordinator
Relationship	Integer	0x00 – 0x03	The relationship between the neighbor and the current device: 0x00=neighbor is the parent 0x01=neighbor is a child 0x02=neighbor is a sibling 0x03=none of the above 0x04=previous child This field shall be present in every neighbor table entry
Transmit Failure	Integer	0x00 – 0xff	A value indicating if previous transmissions to the device were successful or not; Higher values indicate more failures/ This field shall be present in every neighbor table entry.

LQI	Integer	0x00 – 0xff	The estimated link quality for RF transmissions from this device. See sub-clause 3.7.3.1 for discussion of how this is calculated. This field shall be present in every neighbor table entry.
Incoming beacon timestamp	Integer	0x000000-0xffffff	The time, in symbols, at which the last beacon frame was received from the neighbor. This value is equal to the timestamp taken when the beacon frame was received, as described in IEEE 802.15.4-2003 [B1]. This field is optional.
Beacon transmission time offset	Integer	0x000000-0xffffff	The transmission time difference, in symbols, between the neighbor's beacon and its parent's beacon. This difference may be subtracted from the corresponding incoming beacon timestamp to calculate the beacon transmission time of the neighbor's parent. This field is optional.

Annex C: PAN Descriptor Table

Field Name	Field Type	Valid Range	Description
CoordAddrMode	Integer	0x02 – 0x03	The coordinator addressing mode corresponding to the received beacon frame. This value can take one of the following values: 2=16 bit short address. 3=64 bit extended address.
CoordPANId	Integer	0x0000 – 0xffff	The PAN identifier of the coordinator as specified in the received beacon frame.
CoordAddress	Device address	As specified by the CoordAddrMode parameter	The address of the coordinator as specified in the received beacon frame.
LogicalChannel	Integer	Selected from the available logical channels supported by the PHY	The current logical channel occupied by the network.
SuperframeSpec	Bitmap	variable	The superframe specification as specified in the received beacon frame.
GTSPermit	Boolean	TRUE or FALSE	TRUE if the beacon is from a PAN coordinator that is accepting GTS requests.
LinkQuality	Integer	0x00–0xff	The LQ at which the network beacon was received. Lower values represent lower LQ.
TimeStamp	Integer	0x000000 – 0xffffff	The time at which the beacon frame was received, in symbols. This value is equal to the timestamp taken when the beacon frame was received.
SecurityUse	Boolean	TRUE or FALSE	An indication of whether the received beacon frame is using security. This value is set to TRUE if the security enable subfield was set to 1 or FALSE if the security enabled subfield was set to 0.
ACLEntry	Integer	0x00 – 0x08	The *macSecurityMode* parameter value from the ACL entry associated with the sender of the

			data frame. This value is set to 0 x 08 if the sender of the data frame was not found in the ACL.
SecurityFailure	Boolean	TRUE or FALSE	TRUE if there was an error in the security processing of the frame or FALSE otherwise.

Annex D: Source Code of the Proactive mechanism Simulation

```
#include <stdio.h>
#include <stdlib.h>
#include <time.h>
#include <gsl/gsl_rng.h>
#include <gsl/gsl_randist.h>
// The two last included files are relative to the inclusion of the GNU Scientific
Library used for the generation of packets LQIs

typedef struct Parent Parent;

struct Parent
{
int id;
long LQI;
int Dp;
long Ei;
double PAI;
};


//Begenning of the main program

void waits ( int seconds )     //function needed in order to calculate the seeds of
the random numbers
{
  clock_t endwait;
  endwait = clock () + seconds * CLOCKS_PER_SEC ;
  while (clock() < endwait) {}
}

int main (int argc, const char * argv[]) {

    // Variables definitions for the main program

long k, ASP, SSP;
int i, j, NBTC, NBSR, cp, PPV[10], l, EP, VEP, AP, NS, res, NBPC, rt, Tu, temp, ac,
S, TM;
double NPQ, PQ, w, PQP, oPAI, SPQP, PAI1;

FILE* fichier=NULL;                          //Initializing file pointers
FILE* fichier2=NULL;


// Initializing the random number generator
const gsl_rng_type * T;
       gsl_rng * r;

       //int i, n = 400; Optional - Deactivated
       double sigma = 1.0;

       /* create a generator chosen by the
          environment variable GSL_RNG_TYPE */
```

```
       gsl_rng_env_setup();

       T = gsl_rng_default;
       r = gsl_rng_alloc (T);
// End of random number generator initialization


// Entering the different parameters of the model

//do
//{
//printf("Enter the percentage of the expected improvement of the new parent
(K):\n");    //Entry of vriable k
//scanf("%d", &k);
//} while (k < 0);

k = 60;

//do
//{
//printf("Enter the number of packets needed for the confirmation phase:\n");
// Entry of variable S
//scanf("%i", &cp);
//} while  (cp < 0);

cp = 2;

do {
printf("Enter the value of the S parameter:\n");     // Entry of variable S
scanf("%i", &S);
} while ( S < 0);

//S = 100;

//do
//{
//printf("Enter the value of the sampling rate in terms of beacon orders, R:\n");
// Entry of variable R
//scanf("%i", &rt);
//} while (rt < 0 && rt > 20);

rt = 5;

//do
//{
//printf("Do you wich to activate the dynamic adaptation of the S parameter ? 1 for
YES, O for NO:\n");       // Activate or not dynamic S
//scanf("%i", &res);                                          //    res    is    the
response to question : Activate or not the dynamic S mechanism
//} while (res != 0 && res != 1);

res = 0;
/*
if (res == 1)
{
     do
     {
     printf("Enter the value of the Tu parameter:\n");        //  Entry  of  variable
S
     scanf("%i", &Tu);
     } while  (Tu < 0);
}
*/

Tu = 35;
```

```
printf("List of initial values: \n");
printf("Improvement: %i\n",k);

Parent Pparent[6];  // Defining of a variable Pparent of type Parent.

//First parent to which the device is associated has id 1.

//Initializing the different Pparent values
Pparent[0].id= 1;
Pparent[0].Dp= 8;
Pparent[0].Ei= 6;


Pparent[1].id=2;
Pparent[1].Dp= 10;
Pparent[1].Ei= 8;

Pparent[2].id=3;
Pparent[2].Dp= 3;
Pparent[2].Ei= 8;

Pparent[3].id=4;
Pparent[3].Dp= 8;
Pparent[3].Ei= 8;

Pparent[4].id=5;
Pparent[4].Dp= 10;
Pparent[4].Ei= 10;

Pparent[5].id=6;
Pparent[5].Dp= 3;
Pparent[5].Ei= 3;


// Step B in the proactive mechanism: Recieving messages

ac = 0;                                          // Initiate the first parent to id
0
TM = 0;
i = 0;
j = 1;
NBSR = 0;
ASP = 0;                        // Average of quality indicator of Sample Packets
NBTC = 0;                       // Number of Triggred Confirmation Mechanism
SSP = 0;          // Sum of the quality indicator of the primary Sampled Packets
NS = 0;                                          // Number of sample packets
SPQP = 0;
NBPC = 0;                                  // Number of effective parent changes


fichier = fopen("data.txt","a+");           // Opening data file 1, contains the
PAIs of all the packets generated during the simulation
fichier2 = fopen("data2.txt","a+");         // Opening data file 2, contains the
PAIs of the sampled packets


if (fichier2 != NULL)
{
if (fichier != NULL)                        // If file opens we can work on it,
else (see down) print error message
    {

// Start of simulation loop


while (TM < 10000)
{
// Generating a normal LQI, i.e. not to be used for sampling
```

```
w = gsl_ran_gaussian_ratio_method (r, sigma)/2;
            if (w < 1 && w > -1)
            {
            Pparent[ac].LQI = 127 + w*127;
            }
            else if (w < -1)
            {
            Pparent[ac].LQI = 127 + w/2 * 127;
            }
            else if (w>1)
            {
            Pparent[ac].LQI = 127 - w/2 * 127;
            }

// Generating the PAI value using the previously generated LQI and the parent's
attributes

Pparent[ac].PAI = (Pparent[ac].Ei) * (Pparent[ac].LQI) * (Pparent[ac].Dp) / 100;


fprintf(fichier,"%i;%g\n", TM, Pparent[ac].PAI);     // Writing the value of normal
packet quality in data file 1


// Low sampling rate starts

if (i == rt)                                        // Low sampling rate during normal
operation mode
    {
        i=0;
        w = gsl_ran_gaussian_ratio_method (r, sigma)/2;
           if (w < 1 && w > -1)
           {
           Pparent[ac].LQI = 127 +  w*127;
           }
           else if (w < -1)
           {
           Pparent[ac].LQI = 127 + w/2*127;
           }
           else if (w>1)
           {
           Pparent[ac].LQI = 127 - w/2 * 127;
           }

    Pparent[ac].PAI  =  (Pparent[ac].Ei)  *  (Pparent[ac].LQI)  *  (Pparent[ac].Dp)  /
100;


    printf("The quality of the %i packet is: %g\n", TM, Pparent[ac].PAI);

    fprintf(fichier,"%i;%g\n", TM, Pparent[ac].PAI);     //  Writing  the  value  of
the sampled packet in data file 1

    fprintf(fichier2, "%i;%g\n", NS, Pparent[ac].PAI);        // Writing the value
of the sampled packet in data file 2

    SSP = SSP + Pparent[ac].PAI;                                  // SSP is Sum of the
PAIs of the primary Sampled Packets, average is processes later

    NS++;                                                          //   incrementing
the number of sampled packets

    temp = Pparent[ac].PAI;

    if (temp < S)
        {
        NBTC++;                                        // Total number of times the
```

```
triggering mechanism was activated
         SPQP = 0;
         printf ("Parent may be bad, preforming extended tests\n");
         do{

             // Simulate again the LQI indicator and generate new PAI

            w = gsl_ran_gaussian_ratio_method (r, sigma)/2;
            if (w < 1 && w > -1)
            {
            Pparent[ac].LQI = 127 +  w*127;
            }
            else if (w < -1)
            {
            Pparent[ac].LQI = 127 + w/2*127;
            }
            else if (w>1)
            {
            Pparent[ac].LQI = 127 - w/2 * 127;
            }


              Pparent[ac].PAI    =    (Pparent[ac].Ei)    *    (Pparent[ac].LQI)    *
(Pparent[ac].Dp) / 100;

              fprintf(fichier,"%i;%g\n", TM, Pparent[ac].PAI); // Writing the value
of PQP in data file 1
              SPQP = Pparent[ac].PAI + SPQP;              // Caculating the sum of
the quality indicators of the higer sampling rate samples
              printf ("PQP value is: %g\n", Pparent[ac].PAI);

              if (Pparent[ac].PAI < S)
              {
                  j++;
              }
              else
              {
                  j = cp + 1;
              }

         // Activation of the
              if (j == cp)
              {
                  oPAI = SPQP/cp;               // oPAI is defined as the average
of the quality indicator of the packets recieved during the high sampling rate
period
                  printf("WARNING! Parent failure detected! Switch to new parent
!!\n");
                  printf ("Old PAI is %g\n", oPAI);

                  NBSR++;                              // Increment the number of
switching requests
                  printf ("Creating parent list...\n");

                  l = 0;
                  while ( l < 6 )               //   Initializing   the   possible
parents vector with different values of PAI

                      {
                          printf ("PAIs of potential parents are:\n");
// Afficher les PAI générés à des fins de vérification

                         w = gsl_ran_gaussian_ratio_method (r, sigma)/2;
                         if (w < 1 && w > -1)
                         {
                         Pparent[l].LQI = 127 +  w*127;
                         }
                         else if (w < -1)
```

```
                         {
                         Pparent[l].LQI = 127 + w/2*127;
                         }
                         else if (w>1)
                         {
                         Pparent[l].LQI = 127 - w/2 * 127;
                         }

                    Pparent[l].PAI   =   (Pparent[l].Ei)   *   (Pparent[l].LQI)   *
(Pparent[l].Dp) / 100;
                    printf ("%i \t %g \n", l, Pparent[l].PAI);
                    l++;
                    }


               // Finding out the best PAI
               EP = 0;
               VEP = 0;          // Value of the Elected parent's value set to
the first one
               l=0;              // Re initialization of the l counter in order
to increment in following loop

               while (l < 6 )
                    {
                    if ( Pparent[l].PAI > VEP )
                         {
                         EP = l;                              // The Id of the best
parent is given by EP
                         VEP = Pparent[l].PAI;          //  The  best  PAI  of  the
list is given by VEP
                         }
                    l++;
                    }

               printf ("The Id of the best parent of the list is: %i\n", EP);
               printf ("The associated PAI is %i \n", VEP);



               // Decision and reassociation phase

               if (VEP > (oPAI + ((oPAI * k) / 100)) && ac != EP)
//oPAI is the average of the PAIs recieved during the intense sampling phase

                    {
                    ac = EP;                  // The elected paren (EP) becomes
the Actual Parent (ac)
                    printf ("oPAI is equal to :%g\n", oPAI);
                    printf  ("A  new  parent  has  been  elected......  it  is  parent
number: %i \n", ac);
                    NBPC++;
                    }
          }
          //i++;
          TM++;
          } while (j < cp);

     }
     }

TM++;
i++;
j = 0;
}
}

fclose(fichier);      // Closing files before exiting
fclose(fichier2);
```

```
}

if (NBTC != 0)
     {
     ASP = SSP/NS;
     }

// Summary

// Average quality of the sample packets: ASP
// Count the number of triggers of the confirmation mech: NBTC
// Count the number parent changes requests: NBSR

printf ("The average quality of the Sample packets is: %i \n", ASP);
printf ("The number of triggers of the confirmation mechanism is: %i \n", NBTC);
printf ("The number of parent changes requests is: %i \n", NBSR);
printf ("The number of effective parent changes is: %i \n", NBPC);

    return 0;
}
```

Annex E: ECRTS-WiP (Euromicro Conference on Real-Time Systems, Work in Progress) paper submission: “Fault-Tolerance Mechanisms for ZigBee Wireless Sensor Networks”

Bibliography

[1] S. Bandyopadhyay and E.J. Coyle. “An Energy Efficient Hierarchical Clustering Algorithm for Wireless Sensor Networks”, *INFOCOM 2003. Twenty-Second Annual Joint Conference of the IEEE Computer and Communications Societies. Volume: 3* , 30 March - 3 April 2003

[2] T. He, J. Stankovic, C. Lu, and T. Abdelzaher, “SPEED: A Real-Time Routing Protocol for Sensor Networks,” in Proc. of ICDCS’03, May 2003, pp. 46–55

[3] W. Ye, J. Heidemann, *Medium Access Control in Wireless Sensor Networks*, Wireless Sensor Networks, pp. 73-91, ISBN:1-4020-7883- 8, Kluwer Academic Publishers, 2004

[4] G. D. Bacco et al., “A MAC Protocol for Deal-Bounded Applications in Wireless Sensor Networks”, In Proc. of Third Annual Mediterranean Ad Hoc Networking Workshop, June 27-30, 2004

[5] LAN-MAN Standards Committee of the IEEE Computer Society, Wireless Medium Access Control (MAC) and Physical Layer (PHY) Specifications for Low-Rate Wireless Personal Area Networks (LRWPANs), IEEE, 2003

[6] ZigBee Alliance (2007), “ZigBee specification”, December 2006. Available at http://www.zigbee.org

[7] S-J. Park, R. Sivakumar, “Sink-to-Sensors Reliability in Sensor Networks”, Georgia Tech, MobiHoc’03, 2003

[8] A. Willig & H. Karl, “Data Transport Reliability in Wireless Sensor Networks — A Survey of Issues and Solutions”

[9] E. Biagioni & S.H. Chen “A Reliability Layer for Ad-Hoc Wireless Sensor Network Routing”, Proceedings of the 37th Hawaii International Conference on System Sciences – 2004

[10] Shu Hui Chen. Multipath on-demand routing in sensor network topologies. Master’s thesis, Department of Information and Computer Sciences, University of Hawaii at M¯anoa, May 2003

[11] A. Sheth, C. Hartung & R. Han, “A Decentralized Fault Diagnosis System for Wireless Sensor Networks”, University of Colorado, Boulder, 2005

[12] W. L. Lee, A. Datta & R. Cardell-Olivier, “Network Management in Wireless Sensor Networks”, University of Western Australia, 2006

[13] D. Bein, V. Jolly, B. Kumar & S. Latifi, “Reliability Modelling in Wireless Sensor Networks”, International Journal of Information Technology, Vol. 11 No. 2, 2005

[14] N.Li & J. C. Hou, “FLSS: A Fault-Tolerant Topology Control Algorithm for Wireless Networks”, University of Illinois at Urbana-Champaign, 2004

[15] G. Gupka & M. Younis “Fault-Tolerant Clustering of Wireless Sensor Networks”, University of Maryland Baltimore County, IEEE, 2003]

[16] W. R. Heinzelman, A. Chandrakasan, and H. Balakrishnan, “Energy-efficient communication protocols for wireless microsensor networks,” in Proc. of the Hawaii International Conference on Systems Sciences, Jan. 2000

[17] B. Krishnamachari & S. Iyengar “Distributed Bayesian Algorithms for Fault-Tolerant Event Region Detection in Wireless Sensor Networks”, 2005

[18] V. Krunic, E. Trumper & R. Han “NodeMD: Diagnosing Node-Level Faults in Remote Wireless Systems”, University Of Colorado, Dec 2006

[19] Wei Ye, John Heidemann, and Deborah Estrin, “Medium access control with coordinated, adaptive sleeping for wireless sensor networks,” IEEE/ACM Transactions on Networking, April, 2004

[20] G. Lu, B. Krishnamachari, C.S. Raghavendra, “An adaptive energy efficient and low-latency MAC for data gathering in wireless sensor networks”, *Proceedings of 18th International Parallel and Distributed Processing Symposium*, Pages: 224, 26-30 April 2004

[21] Anis KOUBAA, Andre CUNHA, Mário ALVES, “A Time Division Beacon Scheduling Mechanism for IEEE 802.15.4/ZigBee Cluster-Tree Wireless Sensor Networks”. to be presented in Euromicro Conference on Real-Time Systems (ECRTS 2007), Pisa(Italy), July 2007

[22] GNU Scientific Library (2007), http://www.gnu.org/software/gsl/

[23] A. Cunha, A Koubâa, R. Sevrino, M. Alves, “Open-ZB: an open-source implementation of the IEEE 802.15.4/ZigBee protocol stack on TinyOS”, TR IPP-Hurray!, 04-2007

[24] A. Koubâa, M. Alves, and E. Tovar, "A Two-Tiered Architecture for Real-Time Communications in Large-Scale Wireless Sensor Networks: Research Challenges," in the WiP Session of ECRTS'05, Palma de Mallorca, Spain, 2005

[25] IPP Hurray! Art-WiSE (2007), http://www.hurray.isep.ipp.pt/art-wise

[26] TinyOS, (2007), http://www.tinyos.net

[27] Gay D., Levis P., Von Behren R., Welsh M., Brewer E., Culler D.,(2003), The nesC language: A Holistic Approach to Networked Embedded Systems, in Proceedings of the Programming Language Design and Implementation, 2003

[28] Crossbow Technologies INC. http://www.xbow.com

[29] An IEEE 802.15.4 protocol implementation (in nesC/TinyOS): Reference Guide v1.1, TR IPP Hurray!, 02-2007