WirelessHART Network Manager

JUAN HÉCTOR SÁNCHEZ

KTH Electrical Engineering

Master's Degree Project
Stockholm, Sweden

XR-EE-LCN 2011:003

KTH
The Royal Institute of Technology

WirelessHART Network Manager
Software Design and Architecture

MASTER OF SCIENCE THESIS

Juan Héctor Sánchez

Supervisors
Shahid Raza, SICS; Jonas Neander, ABB

Examiner
Viktoria Fodor, KTH

Stockholm
March, 2011

Abstract

WirelessHART standard is becoming a reference as a wireless solution in industrial process automation and control. The *WirelessHART* network performance is mainly determined by its main component: the Network Manager, responsible for creating and configuring the WHART network, as well as managing routing and scheduling communications between devices.

Due to the novelty of the *WirelessHART* standard (2010), there is not an open-source design or implementation of the *WirelessHART* Network Manager available. Only Dust Networks has a commercial Network Manager in the market. This fact makes the *WirelessHART* Network Manager an interesting area of research.

In this thesis, we present a layered interface-oriented component-based architecture and the software design for the *WirelessHART Network Manager*. Furthermore, we give solution to some of the question marks left by the WHART specification regarding the Network Manager operation. Due to the modularity of the proposed design and architecture, the software components can be reused in other *WirelessHART* devices such Gateway or Field Devices.

Acknowledgements

I would like to thank my advisors Shahid Raza from SICS and Jonas Neander from ABB for their support and kind advices during my work. I would also like to thank all the Networked Embedded Systems group and my examiner Viktoria Fodor from the Royal Institute of Technology.

I would like to thank as well to my colleagues and friends Jordi Solsona from Mobile Life and Igor Konovalov from SICS.

Finally, I really appreciate the support and help of Raúl Gracia from Universitat Rovira i Virgili (Spain), Victor Núñez, Marc Santiago, Bianca Chanel Portón, Claire Giron, Audrey Cauvin and, of course, my family.

This thesis has been performed within the SICS Center for Networked Systems funded by ABB.

Contents

List of Figures

List of Tables

List of Abbreviations

AP	Access Point
DLL	Data-Link Layer
DLPDU	Data Link Protocol Data Unit
FD	Field Device
GW	Gateway
HART	Highway Addressable Remote Transducer
ISM	Industry, Scientific and Medical
NDDB	Network Device Database
NL	Network Layer
NM	Network Manager
NMAL	Network Manager Application Layer
NMCC	Network Manager Core Component
NMNL	Network Manager Network Layer
NMPU	Network Manager Process Unit
NPDU	Network Protocol Data Unit
SM	Security Manager
UTC	Coordinated Universal Time
WSN	Wireless Sensor Networks

Chapter 1

Introduction

In the latest years, emerging technologies have been introduced to meet the requirements of industrial process automation. Particularly, wireless technologies such Wireless Sensor Networks (WSN) have notably grown in importance. In this sense, WSNs represent a paradigm shift in today's industrial processes. A reduced cost and easier installation and deployment of the wireless components are the main reasons for considering a wireless solution in industrial applications.

Despite the significant advantages provided by wireless technologies, the available wireless standards were not conceived to cope with industrial requirements. For this reason, HART Communication Foundation developed *WirelessHART*. *WirelessHART* is considered to be the first open standard for Wireless Sensor Networks specifically designed for industrial process and control automation.

WirelessHART is based on the proven and widely-used wired HART Communication Protocol [13]. On March 2010 *WirelessHART* was approved as the First International Standard for Industry Process Automation by the IEC (International Electrotechnical Commission). Furthermore, on June 2010 it was approved as the First European National Standard for Wireless Communication in Process Automation by the CENELEC, the European Committee for Electrotechnical Standardization [2]. *WirelessHART* establishes a secure and reliable wireless mesh network communication protocol. It adds wireless capabilities to the HART Protocol while maintaining backwards compatibility with widely-used HART devices.

However, *WirelessHART* is a recently developed standard. Consequently, a fraction of the components defined in its architecture remain without a detailed-requirement analysis and design. In this regard, the most important component, which lacks of an exhaustive analysis, is the Network Manager. The Network Manager is the central component in *WirelessHART* and is responsible for forming and configuring the network, managing routing and scheduling, and monitoring the overall communication performance.

In this thesis, we extend the specification for the *WirelessHART* Network Manager and provide a software architecture and design.

1.1 Motivation and Problem Statement

Configuration and formation of the *WirelessHART* mesh network is centrally managed by the Network Manager (NM). Thus, the Network Manager plays a key role within the *WirelessHART* network. It is responsible for configuring the network, scheduling communications, managing route tables and monitoring the health[1] of the network. Its design determines the overall performance of the network. A proper design of the NM means to optimally fulfill the requirements of the industrial communications.

Nevertheless, there is not a standardized open-source design of the Network Manager available. In our view, the absence of a clear and detailed description of the Network Manager can discourage the establishment of the *WirelessHART* standard: (1) Other existing standards may provide a complete specification of their architecture, making them easier to adopt by the industry. (2) A misinterpretation of the NM definition may lead to inefficient and heterogeneous implementations. (3) Finally, delegating the requirement analysis and design to developers may introduce a supplementary effort, in addition to the implementation cost itself.

The purpose of this thesis is to provide a better understanding and our own *WirelessHART Network Manager* specification and architecture design. It can be used as a reference for implementing or perform specific research in subfunctions of the *WirelessHART Network Manager*.

1.2 Goals

The main goals of the present thesis are depicted as follows:

- Perform an exhaustive analysis of the *WirelessHART* specification and its components, giving special emphasis to the kernel of the present thesis: the Network Manager.
- Provide the specification of the requirements of the Network Manager. The *WirelessHART* standard does not dedicate a single document for the Network Manager. All the information for understanding the Network Manager procedure is spread over several specifications
- Give solutions to the specification gaps regarding the Network Manager, since the *WirelessHART* standard leaves open a plenty of question marks regarding the important operation of the NM.
- Propose a software architecture and design of the Network Manager, including detailed description of its internal subcomponents.

1.3 Research Method

In the present thesis, we deal with a qualitative research methodology. We proceeded initially identifying the general problem by an exhaustive analysis of the *WirelessHART* specification and related work. Later on, we decomposed the identified general problems into several defined and standalone tasks. Then, we specified the software requirements of the WirelessHART Network Manager and subsequent design. Finally,

[1] The *WirelessHART* Network Manager gathers health reports from the network, which provide information regarding communications performance.

we performed iterative enhancements of the design after exhaustive revision and verification.

1.4 Scientific Contributions

The scientific contributions of the present thesis are mainly three:

- The requirements of the Network Manager are spread over several documents within the *WirelessHART* standard. This work contributes in a compact specification of the software requirements of the *WirelessHART* Network Manager and its functionality.
- A significant part of the Network Manager specification remains uncompleted, leaving up to the designer important decisions. We propose solutions to some of these unspecified features regarding the Network Manager. For instance, time synchronization and communication between the NM and the Gateway[2].
- The most relevant task of the present thesis is to provide a complete and fully detailed design and architecture of the *WirelessHART Network Manager*, which has not been provided yet to the best of our knowledge. The final goal of the mentioned design is to go one step further in research regarding *WirelessHART* instead of providing a commercial solution of the Network Manager.

1.5 Thesis structure

The remaining part of this thesis has been structured as follows. In Chapter 2, we give a general overview of the requirements for sensor networks in Industrial Applications. Later on, we introduce alternative protocols for Wireless Sensor Networks: Bluetooth, ZicBee and ISA100.11a, concluding why *WirelessHART* is chosen as object of study. At the end of the same chapter, the related work is depicted. In chapter 3, we present a detailed description of *WirelessHART* standard. Afterwards, we provide the software requirements of the *Wireless HART Network Manager*, that is, what the software is intended to do. Moreover, we introduce the proposed solutions to the specification gaps. In chapter 5, we introduce the necessary background, such as software design tools —UML and software design patterns—, to properly understand the rest of the chapter. Then, we elucidate the architecture and detailed design of the Network Manager. We conclude the thesis proposing future research lines and drawing some conclusions.

[2]The Gateway is an important component of the WirelessHART, link between the host applications, the Network Manager and the Field Devices (refer to section 3.2)

Chapter 2

Background and Related Work

This work deals with an emerging open standard, *WirelessHART*, specifically designed to fulfill the requirements for Wireless Sensor Networks applied to the process industry. In this manner, a prerequisite in the process of understanding the *WirelessHART* standard resides in giving a general overview of the requirements for industrial wireless communications. This knowledge will aid the reader to recognize the motivations behind the distinct features of *WirelessHART* and to understand the need of a standardized solution in the process industry. Furthermore, in this chapter we will describe briefly the generic characteristics of other existing communication protocols for Wireless Sensor Networks (WSN) such as Bluetooth, Zigbee and ISA100.11a, that may be applicable in the industry. Later on, we conclude motivating the reasons why *WirelessHART* is the chosen object of study. Finally, the last part of this chapter will refer to the related work.

2.1 Requirements for industrial wireless sensor network applications

Industrial process and automation networks have been traditionally wired networks which imply additional costs of deployment and maintenance. With the growing development of Wireless Sensor Networks (WSN) many efforts have been done for introducing the wireless technology in the process industry. A reduced cost and easier installation and deployment of the wireless components are the main reasons why for considering a wireless solution in industrial applications.

Nevertheless, once it has been proven that the wireless technology can be applicable in the process industry, the next step is to guarantee the fulfillment of the strict industrial-communications requirements. The latter, combined with the need of interoperability between vendors, motivate the necessity of a standard for wireless industrial communications.

The requirements for industrial communications do not change substantially with the introduction of the wireless technology in the process industry. The key requirements for communications in industrial automation and control processes are security and reliability. However, in the WSN's domain there are other parameters to take into account when designing a communications protocol for the industry. These parameters are power consumption, scalability and backwards compatibility.

Security in industrial communications is of great importance, specially when considering wireless communications, where the medium is shared and accessible. In the industry, the exchange of information must be irrevocably confidential in order to avoid possible industrial espionage, that may lead the competence to a strategic vantage. A standard specifically design for industrial communications must include powerful methods of encryption and authentication for the sake of guaranteeing secure communications.

Reliability is another relevant industrial requirement since it determines the ability of the system to perform its required functions under determined conditions. For instance, guaranteeing transmissions with max delay is of great importance in industrial communications. Additionally, industrial environments are typically harsher for wireless communications in terms of interferences and physical obstacles which can spoil the proper operation of the system. Failures in the industrial equipment due to communications are translated into economical loss for the industrial user. Thus, a industrial communication standard must consider techniques such network robustness, frequency and path diversity and reliable message delivery[1] in favor of increasing the reliability of the system.

Concerning Wireless Sensor Networks, other factor to take into consideration is the power consumption. Lower power consumption implies higher battery lifetime, that is to say lower cost. Notwithstanding, power consumption may not be as critical as other requirements such security and reliability.

Wireless technology is considered to be a paradigm shifter in the process industry. The industrial standard should consider backwards compatibility with the existing wired networks in the process industry. The reason why is to save massive cost in equipment renovation, allowing a more gradual investment by the industrial companies.

There are a few standardized solutions for Wireless Sensor Networks available that could be used for industrial process automation. Those will be discussed in the next section.

2.2 Wireless protocols for industrial automation

The present thesis refers to WirelessHART as the main candidate as a wireless solution in the process industry. However, there are other standardized communication protocols that may be used for industrial automation but with some drawbacks. In the following subsection we will describe briefly the features of Bluetooth, Zigbee, ISA100.11a and conclude why *WirelessHART* is the chosen wireless standard in this thesis.

Bluetooth and ZigBee are wireless standards designed for general-purpose Wireless Sensor Networks (WSN) applications, whereas *WirelessHART* and ISA100.11a are specifically targeted at wireless industrial communications. Industrial applications usually have stricter timing requirement and higher reliability and security concern, in comparison with office applications.

Finally, all the communication protocols to be depicted bellow have in common the use of the unrestricted 2,4 G Hz ISM radio band.

[1] Assure end-to-end communications.

2.2.1 Bluetooth

Bluetooth [25] [1] is an open wireless communication protocol targeted at Personal Area Networks (PAN). It is widely-used in short-range[2] office applications.

Bluetooth provides useful features such channel hopping and time slots which increase the reliability of the system. However, the standard does not make any effort to guarantee an end-to-end wireless communication delay. In addition, Bluetooth assumes quasi-static reduced star topology[3], making it inviable in large process control applications where scalability plays an important role. Furthermore, Bluetooth's security[6] is optional and only provides a Eo stream weak cipher algorithm with no security at application layer. Finally, Bluetooth wireless devices have a limited battery lifetime compared to other energy efficient wireless standards. In summary, Bluetooth is not suitable for industrial applications as it does not meet the strict industrial requirements regarding security, scalability and power consumption.

2.2.2 ZigBee

The same way as Bluetooth, ZigBee protocol [5] [28] is targeted at Personal Area Networks (PAN). However, unlike Bluetooth, ZigBee provides means for an energy efficient operation, thus increasing the battery lifetime and saving costs.

ZigBee is built upon the IEEE 802.15.4 phyisical and MAC layer (see Figure 2.1). ZigBee is secured by a 128-bit AES cipher algorithm and user defined security at application layer. In this sense, ZigBee is secure enough to be used in an industrial set-up. In addition, the stated communication protocol allows mesh topology and provides relatively fast communications. Nevertheless, as Bluetooth, ZigBee does not make any effort to provide a guarantee on end-to-end wireless communication delay. Furthermore, ZigBee does not provide means of frequency diversity, path diversity or reliable message delivery. Interferences and persistent obstacles, usually inherent to industrial environments, are a great problem for a communication protocol such as ZigBee.

In conclusion, ZigBee is not suitable for industrial process applications since it does not meet the requirements of reliability and industrial-grade network robustness. In the other hand, ZigBee satisfies the industrial requirements of security and power consumption.

[2] Range up to 10 meters.

[3] Star topology with 1 master and up to 7 slaves.

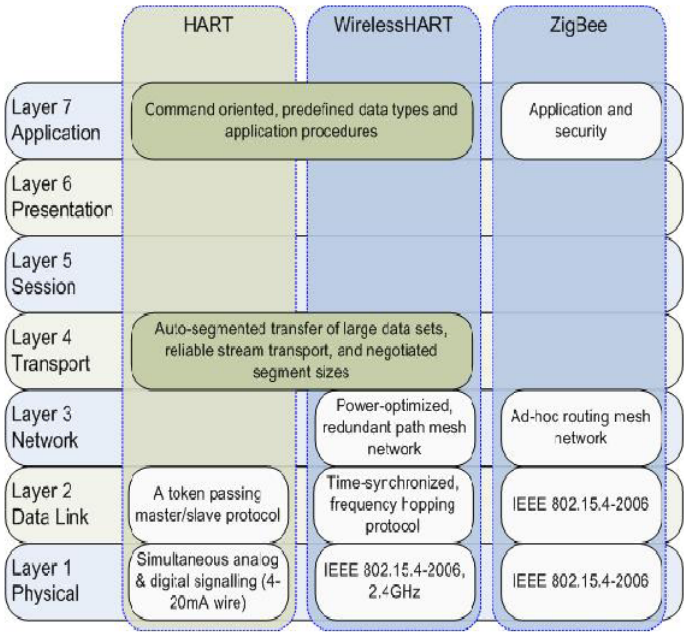

Figure 2.1: *Comparison of Wireless HART and Zigbee protocol stack in the OSI model.*

2.2.3 ISA100.11a

As stated earlier, *WirelessHART* is considered to be the first open standard for Wireless Sensor Networks specifically targeted for Industrial process automation and control systems. However, the International Society of Automation (ISA)[3] developed parallely the ISA100.11a standard: "Wireless Systems for Industrial Automation: Process Control and Related Applications".

ISA 100.11a was recently approved as a wireless communication standard. It provides great methods of security such asymmetric cryptography and object-based application layer security. Additionally, ISA 100.11.a establishes means of reliability though frequency diversity, network robustness and reliable message delivery.

ISA 100.11a is regarded as the great competitor of *WirelessHART*. It is designed to satisfy optimally the industrial requirements of security, reliability, scalability and power consumption. However, it does not include a backwards compatibility with the existing industrial technology, making more costly its establishment in the process industry.

2.2.4 WirelessHART

WirelessHART was created to fulfill the existing gap in the industrial wireless standardization. It was born as an extension of the widely-used HART communication protocol. It is designed to be simple-to-use, self-organizing and self-healing, flexible, reliable, secure and support the widely-used HART technology.

WirelessHART is a centrally managed mesh network. It is built upon the IEEE 802.15.4 physical layer and it adds its own Data-Link, Network and Application Layer.

Industrial security and authentication is reached through 128-bit AES (Advanced Encryption Standard)[4] algorithms that cover end-to-end and hop-to-hop communications. Medium Access Control (MAC) is based on TDMA schedule with frequency hopping. Reliability is achieved using methods of frequency diversity, path diversity and message delivery retrials. Power Consumption can be efficiently optimized by a proper management of the communications schedule.

As stated earlier, *WirelessHART* is designed to fulfill optimally the wireless industrial requirements. In that way, security, reliability, scalability, power consumption and backwards compatibility are fundamental in *WirelessHART*.

2.2.5 Summary

The table 2.1 depicts briefly the distinct features of the communication protocols described in the previous sections:

	Bluetooth	**Zigbee**	**ISA100.11a**	**WirelessHART**
Security	Optional	High	Very High	Very High
Reliability	Low	Very Low	Very High	High
Power Consumption	High	Medium	Low	Low
Scalability	Limited (8 device)	Medium	High	High

Table 2.1: Comparative table including the main features of Bluetooth, Zigbee, WirelessHART and ISA100.10a.

In conclusion, *WirelessHART* is the chosen wireless solution as it is specifically designed to fulfill the industrial requirements of wireless communications and, moreover, provides a backwards compatibility with the widely-used and proven-to-be-effective HART technology in the process industry, in contra position with ISA 100.11.a.

[4]Symmetric-Key Encryption

2.3 Related Work

WirelessHART has been thoroughly studied by Kim et al. [23]. This paper describes, among other things, implementation and design issues when implementing *WirelessHART* architecture.

Many of the related security issues in *WirelessHART* was studied in Raza et al. [26]. The authors introduce pros and cons of *WirelessHART* security scheme alongside with a prototype implementation of a security manager.

Song et al. [27] and Gustafson [11] propose prototype architectures for *WirelessHART*. In the first case the authors implement a fairly complete *WirelessHART* architecture targeting such critical parts such as synchronization, time keeping, routing and scheduling. In the later case, the author mainly focuses on higher level issues such as the Network Layer and the Transport Layer services including reliable data delivery through the mesh network and handling of commands. None of the mentioned papers, however, specifically targets specification and implementation issues in *WirelessHART* Network Manager as even specification remains independent from it. In this thesis, we look closely at the Network Manager design issues and the corresponding timing problems both in real devices and in augmented reality.

A hybrid simulation framework for *WirelessHART* has been discussed in Konovalov et al. [24] based on cross platform implementation in COOJA using real *WirelessHART* Network Manager plus Gateway and Field devices.

Chapter 3

WirelessHART

WirelessHART Network Manager is the main component of the WirelessHART network since it determines the overall performance. On the grounds that *WirelessHART* has a central management architecture[1], it is really important to understand the *WirelessHART* communication protocol for getting a better conception of the requirements fulfilled by the standard and the requirements of the Network Manager itself.

For that reason we present in this chapter a general overview of the *WirelessHART* standard. We will describe briefly the main features of the *WirelessHART* protocol and, later on, we will depict the general architecture of a *WirelessHART* network, presenting the components and main functions related to them. Communications protocols are better understood when described from OSI-model point of view. Thus we will describe succinctly the different layers composing the *WirelessHART* protocol stack.

3.1 Main features

WirelessHART was created as an extension to the legacy HART technology, which is widely used in industrial process automation applications, providing wireless capabilities to the HART5 4-20mA protocol. *WirelessHART* establishes a secure communication protocol standard that utilizes a time synchronized, self-organizing, and self-healing mesh network architecture. Also *WirelessHART* provides capabilities of monitoring network diagnostics and optimizing performance of the industrial process.

The main objectives of the wireless HART standard are:

- Backwards compatibility, i.e. support existing HART technology.
- More flexibility for installing and operating process automation equipment.
- Reliable, easy-to-use, simple communications.
- Interoperability, i.e. allow HART-enabled devices from different manufacturers to work together.

WirelessHART uses Time Division Multiple Access (TDMA) with frequency hopping for Medium Access Control (MAC) and a centrally managed[2] configuration of the routes and schedule of the network. *WirelessHART* works on the widely used and

[1] The Network Manager is the responsible for managing the overall WHART Network.
[2] By the Network Manager

shared 2.4 GHz ISM band. For avoiding interferences with other systems working in the same band (such Bluetooth or Wifi), *WirelessHART* standard provides mechanisms such Frequency Hopping Spread Spectrum (FHSS) and channel *Blacklisting*, i.e. disallowing the use of determined channels. Reliability is also provided in *WirelessHART* by using redundant paths an redundant communication opportunities, that is time slots in the WHART schedule. These mechanisms, along with the frequency hopping and channel blacklisting, allow to achieve a high network robustness and a *high reliability*, required in industrial applications.

Furthermore, the use of TDMA for the medium access control reduces collisions, by decreasing shared transmissions, and lowers the *power consumption* of the devices, since their radio transceivers are only awake in the slots pre-scheduled by the Network Manager.

Security in *WirelessHART* is of great importance and it is achieved through centrally managed secure sessions at the Network Layer and encryption at Data-Link layer. Due to the significance of security in industrial applications, we will describe thoroughly the security in *WirelessHART* in section 4.2.1.

WirelessHART is a hybrid network consisting of wireless and wired devices. Its architecture and functionality will be described in the following section.

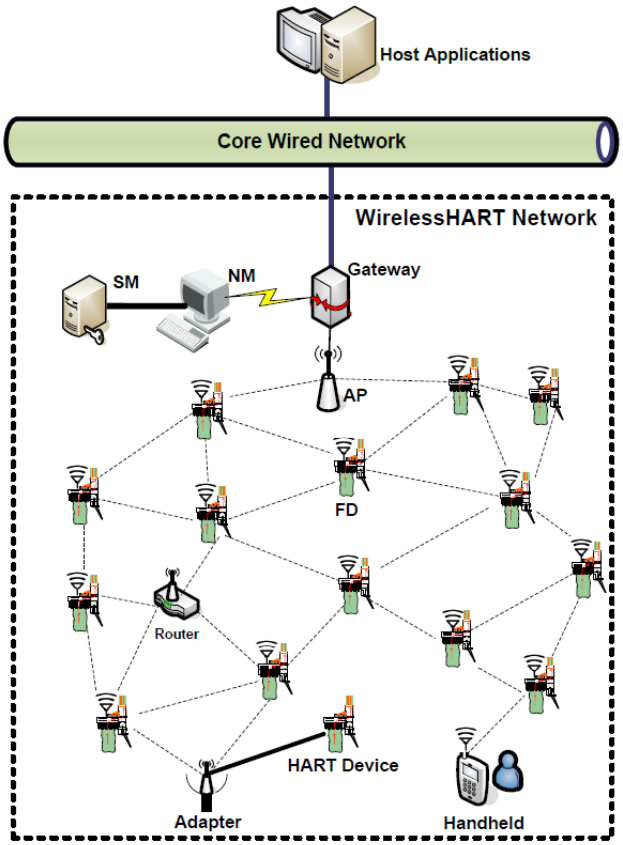

Figure 3.1: *Wireless HART network components.*

3.2 Components

Figure 3.1 depicts the architecture of a WirelessHART network indicating all components included in the WirelessHART standard. The components of the WirelessHART network (refer to [18]) are described briefly bellow:

- **Network Manager (NM)** It is the object of study of the present thesis. It is the application responsible for forming and configuring the network, scheduling communication between devices, managing the routing in the network and monitoring and reporting the health of the network. There is only one active Network Manager per WirelessHART network.

- **Security Manager (SM)** Application responsible of managing the security resources, that is the security keys, and monitoring the status of the network security.

- **Gateway (GW)** Divided into virtual Gateway and the Access Points (AP) (1 or more). It is the link between the host applications, Network manager and the wireless HART network. Responsible of buffering, protocol conversions and clock source.

- **Field Devices (FD)** The actual sensors distributed in the industry process capable of routing and forwarding packets.

- **Host applications** User applications connected to the backbone network of the industry that communicate with Field Devices in behalf of fetching process and control data. The Gateway is the connection point between host applications and the WirelessHART Network

- **Adapters** They are the devices providing backwards compatibility by adding wireless HART capabilities to wired HART devices. It can provide wireless access to one or more devices.

- **Handheld** Host application residing on a portable device. It's aim is the configuration, monitoring, calibrating and maintenance of devices. It can be connected to the WHART network or the plant automation network.

- **Routers** Devices capable of routing and forwarding packets in the network. However, they are not connected to the industrial process (sensors or actuators). They are required when wireless connectivity needs to be improved.

3.2.1 Network Device

In the rest of the thesis we will refer to the concept of Network Device. The term Network Device refers to any device of the WHART network capable of participating in forwarding packets within the *WirelessHART* network. Gateways, Field Devices, Adapters, Handhelds and Routers are considered network devices. The most common type of Network Device is the Field Device. It is important to notice that the Network Manager is NOT considered a Network Device since it does not participate in the *WirelessHART* network. All the packets which source is the Network Manager are injected into the *WirelessHART* network through the Gateway (see [18]).

3.3 Protocol Stack

The most intuitive way to understand a communications protocol is to describe it from the packet or layer point of view. Figure 3.2 shows the distribution of the different layers of the *WirelessHART* standard on the OSI layer design model.

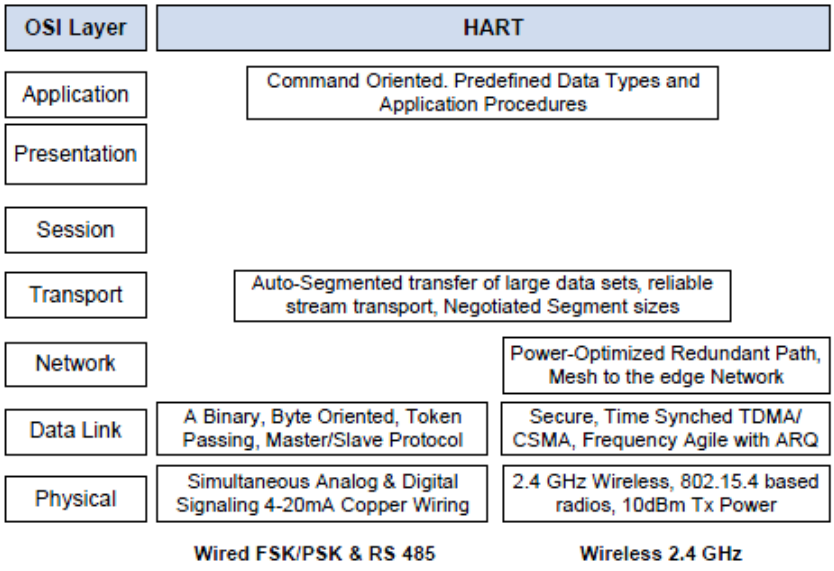

Figure 3.2: *HART (left) and* WirelessHART *(right) protocol stack in the OSI model.*

WirelessHART is built upon the physical layer IEEE 802.15.4 defining new Data-Link, Network, Transport and Application layers. In the next sections we will give a general idea of each layer forming the *WirelessHART* protocol stack.

3.3.1 The WirelessHART Physical Layer

The *WirelessHART* physical layer is based mostly on the IEEE STD 802.15.4-2006 2.4GHz DSSS physical layer [22]. This layer defines radio characteristics, such as the signaling method, signal strength, and device sensitivity. *WirelessHART* operates in the 2400-2483.5MHz license-free ISM band with a data rate of up to 250 Kbits/s. Its channels are numbered from 11 to 26, with a 5MHz gap between two adjacent channels.

3.3.2 The WirelessHART Data-Link Layer

The WirelessHART Data Link Layer Protocol Data Unit (DLPDU) establishes the mechanisms for a reliable and secure communication at the Data-Link Layer (DLL). The DLL resides on top of the IEEE 802.4.15 Physical Layer. The WirelessHART DLL differs from the IEEE 802.15.4-2006 DLL introducing frequency hopping and channel blacklisting. The interference in a industrial environment can be reduced with good management of channel blacklisting, competence of the network manager. In that way *WirelessHART* increases the reliability of the system. The DLPDU payload is ciphered

using a 128-bit AES cipher algorithm thus reaching secure communication at these level. The Data-Link layer is also responsible for keeping correct synchronization of the network among the WHART devices.

The *WirelessHART* DLPDU structure is illustrated in the Figure 3.3

The TDMA Data Link Layer Specification[15] defines five *WirelessHART* frame types: Acknowledgment DLPDU, Advertise DLPDU, Keep-Alive DLPDU , Disconnect DLPDU, Data DLPDU.Only data DLLPDU contain higher layer's information in their payload, the rest are exclusively used by Data Link information. Advertisement is particularly important in WHART since it contains information about the joining superframes and links for the joining devices. This information is provided by the object of study in this thesis: the Network Manager.

The configuration of the schedule tables that determine when and to whom to send the DLPDUs is delegated to the Network Manager. The Network Manager configures them in the process of formation of the *WirelessHART* Network and, lately, when adapting the network in behalf of a better performance.

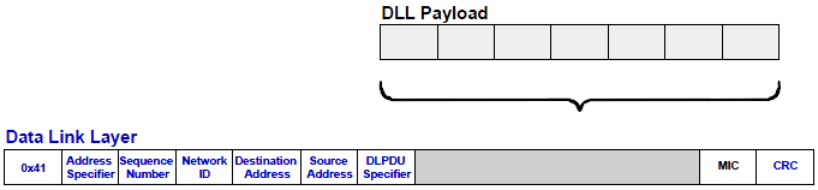

Figure 3.3: *Wireless HART Data-Link Layer PDU.*

3.3.3 The WirelessHART Network Layer

The Network Layer responsibilities consist of several functions including packet routing, ensuring secure end-to-end communications and encapsulating the Transport Layer information exchanged across a network.

The Network Manager is responsible for configuring the routing tables of every Network Device in the joining process. The routing tables are updated by the Network Manager when adapting the *WirelessHART* Network accordingly to the necessities of performance and communication requirements. In *WirelessHART* there are three ways of routing: (1) Graph routing, used when the devices have joined the network and, therefore have been configured by the NM. It establishes a group of paths, identified by Graph ID, from source to destination. (2) Source Route, used for diagnostics, establishes a fixed path from source to destination, and (3) Proxy Route, used when the device has not yet joined the network.

As stated previously, security is a MUST in industrial communications. For that reason, *WirelessHART* provides at this level secure end-to-sessions which are configured by the Network Manager when the devices join the *WirelessHART* network. The NPDU is ciphered and authenticated with a 128-bit AES Session Key. In section 4.2.1, we describe thoroughly security in *WirelessHART*.

The NPDU header (Figure[3] 3.4) starts from a Control byte that specifies an addressing scheme employed and indicates if special routes are used in the reminder of the header. A Time-To-Live (TTL) field is a counter which is decremented at the each

[3]Figure from [21] sect. 9.1.

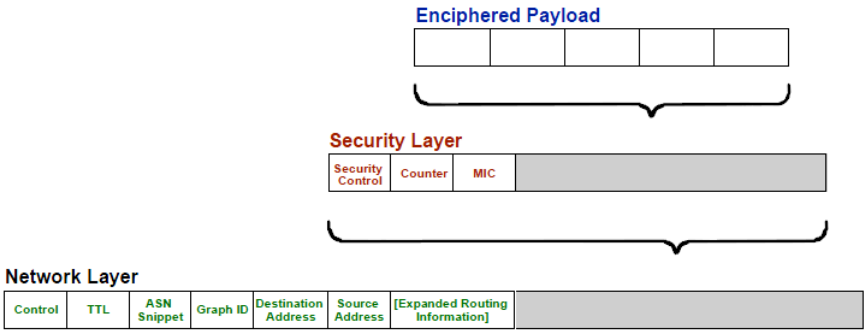

Figure 3.4: *Wireless HART Network Layer PDU.*

next hop, hence determining an amount of hops a packet can travel before it is dropped. An Absolute Slot Number (ASN) Snippet field provides performance metrics and a diagnostic information of a network operation. This field specifies the time passed since a packet was created. A Graph ID field is used to route a packet across a network, identifying nodes which can be used along the way. Remaining fields specifies addresses and additional routing options such Proxy Route and Source Route.

Security sublayer is a part of the NPDU header, it is used for data encryption and the NPDU authentication. Security sublayer Control specifies a type of a security employed: Join Key, Unicast Session Key or Broadcast Session Key. A Message Integrity Code (MIC) is responsible for checking data integrity. An overall length of the NPDU header may vary depending on the length of the source and the destination addresses, special routes and the counter length. The minimum length of the NPDU header is 21 bytes.

The payload of the NPDU corresponds to the enciphered Transport Layer PDU which contains the actual HART commands used for communicating in the WHART Network.

3.3.4 The WirelessHART Transport Layer

WirelessHART Transport Layer ensures an end-to-end packet delivery for all the communications that require acknowledgment such REQUEST/RESPONSE traffic. Figure 3.5 shows the structure of the *WirelessHART* Transport Layer PDU (TPDU) structure.

The TPDU is enciphered using one of the session keys or the join key. Devices that wish to communicate must be provisioned with the identical join keys. The Transport Layer encapsulates *WirelessHART* Application Layer data, that is an array of aggregated commands.

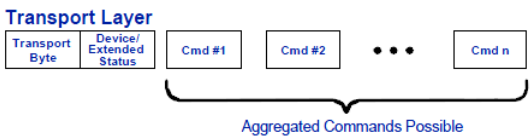

Figure 3.5: *Wireless HART Transport Layer PDU.*

3.3.5 The WirelessHART Application Layer

The application layer is the topmost layer in *WirelessHART*. It defines various device commands, responses, data types and status reporting. In *WirelessHART*, the communication between the devices and gateway is based on commands and responses. The application layer is responsible for parsing the message content, extracting the command number, executing the specified command, and generating responses.

The Network Manager utilizes the Application Layer commands for configuring and managing the entire *WirelessHART* Network.

Figure 3.6: *Wireless HART Application Layer HART command format.*

3.4 Security in WirelessHART

Security in industrial applications, thus in *WirelessHART*, is mandatory. *WirelessHART* provides a reasonable strong security [26] by end-to-end and hop-to-hop secure mechanisms through authentication and payload encryption on the Network Layer and authentication on the Data-Link layers.

WirelessHART uses a 4-byte Message Integrity Code (MIC) for authentication and deciphering. The MIC is generated and confirmed using CCM* mode (Counter with CBC-MAC (corrected)) with AES-128 as the underlying block cypher. CCM* needs 4 byte-strings as parameters (a, m, N, K), where K is 128-bit AES key, a the additional data to be authenticated but not enciphered, m the message to be enciphered and N the 13-byte nonce. In the *Data-Link Layer*, DLPDU are not encrypted but authenticated, thus the second parameter m is empty, while a includes the DLPDU header and payload. In the case of the *Network Layer*, the NPDU is authenticated but only the NPDU payload is encrypted, thus m is the NPDU payload and a is the NPDU header with the NPDU TTL, Counter and MIC fields set to zero.

The security keys that *WirelessHART* uses for secure communications are provided by the Security Manager and distributed by the Network Manager to all devices when they integrate the network. There are six different security keys for the secure sessions provided by the Network Layer:

1. *Join Key* : used by devices for enciphering the join request to the Network Manager. It is provided to the devices manually using the maintenance port.

2. Session Keys: used in pairwise communications between devices and Network Manager, Gateway and Handheld [4]. These keys along the Network Key are provided in the joining process by the Network Manager using the Join Key to encipher them (refer to 4.1.1). This Keys will be provided by the Network Manager only if the credentials presented in the Join request are valid.

   (a) *Network Manager Unicast Key*.

   (b) *Network Manager Broadcast Key*.

[4] For maintenance purposes

(c) *Gateway Unicast Key*.

(d) *Gateway Broadcast Key*.

3. *Handheld Key*: used for peer-to-peer communications with the handheld for maintenance and configuration. This key is provided by the Network Manager when requested by the Handheld.

In the Data-Link layer devices use two type of keys for hop-to-hop authentication: the *Well Known key*, when they wish to integrate the WHART network or in advertisement packets, and the *Network Key*, provided in the process of joining by the Network Manager[5] and used in normal communications. The Well-Known Key is public to all devices.

All Security Keys are renewed and rotated according to the security requirements of the plant automation.

In summary, *WirelessHART* provides a strong security, transparent to the application layer, that fulfills the industrial requirements. The only vulnerability resides in the fact that the Join Key has to be distributed to all devices prior to initialize the network. However, if this process is done in a secure way, breaking *WirelessHART* security is highly improbable.

[5]The Network Manager uses the Join Key to encipher the NPDU that contains the Network Key.

Chapter 4

WHART Network Manager. Software Requirements Specification

From the previous chapter, it can be deducted that the Network Manager plays a really significant role in the configuration and management of the *WirelessHART* network. Each protocol-stack layer of each device forming the *WirelessHART* network is configured by the NM.

The goal of this thesis is to provide a Software Design and Architecture of the *WirelessHART Network Manager*. In this sense, it is important to specially understand the particular functions of the Network Manager.

The Network Manager is responsible for initializing itself and initialize the WHART Network by configuring the Gateway and Access Points in order to begin advertising. Advertising will permit other Network Devices to join the network by requesting it to the Network Manager. The Network manager configures the joining devices with nickname, security keys, routes and schedule. Once this step is performed, these devices are integrated into the network and can consequently advertise in order to allow even more devices to join the network. Integrated devices can request for communications resources to the Network Manager. The Network Manager is in control of readjusting the route map and schedule in order to satisfy the needs. In addition, the Network Manager monitories the performance of the network by keeping track of the Health reports and alarms sent. Using this information the Network Manager can update the routes and schedule in favor of a better communication performance.

This chapter refers to the main software requirements of the *WirelessHART Network Manager* specified in the standard and it also provides solutions to the question marks left by the specification. This requirements are divided into functional and non-functional requirements.

4.1 System Functional Requirements

Functional requirements represent the functions that our software system, in this case the *WirelessHART Network Manager*, is intented to perform. The reasons why we include this section in the thesis, instead of referring to the standard, are: to assure a better understanding of the functions performed by the NM and to provide a compact version of the requirements due to the fact that there is not a single specification dedicated to the *WHART Network Manager*. All the information needed to recognize the functionalities and qualities of the WHART Network Manager is spread into several specifications [15, 21, 14, 12, 16, 17, 18].

4.1.1 Network Formation and Configuration

The Network Manager is responsible for initializing itself and configuring the Gateway(s) for advertising, allowing the *WirelessHART* network to start. It is also responsible for configuring the joining devices by means of Nicknames[1], Security keys, Routing and Scheduling.

There is extensive information about network formation and joining process in the WHART standard (refer to [21], [18]). However, the initialization process of the WHART network, that is the initial configuration of the Gateway and Access Points. remains open in the standard since the interface between the Network Manager and the Gateway are not specified (see section 4.3.2).

4.1.1.1 HART Commands related to Network Formation

The HART commands used by the process of joining to the WHART Network (see 4.3.2 and [17]):

- Command 977 Gateway Join Request (refer to A.1)
- DL Command 961 Write Network Key
- DL/NL Command 962 Write Device Nickname Address
- NL Command 963 Write Session
- NL Command 964 Delete Session
- Commands used for configuring the Schedule (refer to 4.1.3.2)
- Commands used for configuring the Route (refer to 4.1.3.2)

The HART commands used by Field Devices in the process of joining to the WHART Network (see [16] [17]):

- Command 0 Read unique Identifier
- Command 20 Read Long Tag
- DL Command 787 Report Neighbor Signal Levels
- DL Command 961 Write Network Key

[1] Short Addresses (2 bytes) used in the Network and Data-Link Layer to make the packet length shorter. The Long Address (4 bytes) of the devices is used before joining.

- DL/NL Command 962 Write Device Nickname Address
- NL Command 963 Write Session
- NL Command 964 Delete Session
- DL Command 971 Write Neighbor Property Flag
- Commands used for configuring the Schedule (refer to 4.1.3.2)
- Commands used for configuring the Route (refer to 4.1.3.2)

4.1.2 Routing

Managing Routing in the WHART Network is one of the most important functions that the Network Manager performs. Routing determines the *WirelessHART* performance and reliability. The Network Manager is responsible for creating and managing the network route, i.e. the complete map of the WHART network. This is accomplished by managing the route tables of all the devices participating in the WHART network.

There are three types of Routing in WHART:

- **Graph:** Graphs represent a collection of paths that connect two nodes in the WHART network. It is the common way of representing the network route for both upstream and downstream communications. The devices have to be preconfigured by the Network Manager before using graph routing. Several graphs can be defined in the network and the graph to be used is indicated in the header of the Network Layer packet by the Graph ID.
- **Source Route:** Source routing is a supplement to the graph routing aiming at network diagnostics. A source route indicates a direct path between source and destination devices (including intermediary devices). The routing information is included in the header of the network layer packet (See section 3.3.3). The intermediary devices do not need to be preconfigured previously with routing tables. However, there has to be a communication opportunity (link, refer to section 4.1.3) configured for every hop of the path. In our design we will consider Source Routes derived from graphs, i.e. a source route as a particular path of a determined graph.
- **Proxy Route:** The device that has received the join request from a joining device serves as the Proxy for initial communications with the NM. The Proxy address is indicated in routing options in the Network Layer packet.

It is important to mention that the Network Manager is responsible for configuring the routing scheme and manage the communication tables of the devices but does not participate actively in routing packets in the WHART network. Network Manager's network address is fixed in all the networks and is 0xf390. All the communications between the Network Manager are made through the Gateway (and Access Point(s)).

All the information about routing in WHART can be found in [18] section 9.1.8, 9.3.3 and 9.4, in [21] section 6.2.3 and 9.1. The Routing tables that the Network Manager has to configure in every device of the WHART network are the Route Table, the Graph table and the Source Route Table. For more information about the communication tables refer to [21] section 9.3.2 and [15] section 9.2.

4.1.2.1 Routing Algorithm

The WHART specification does not define a specific routing algorithm. It is up to the implementor to decide which routing strategy and algorithm is used for deciding the best route. A brief routing strategy is summarized in [18] section 9.4.2.

The election of the best suitable routing algorithm for WHART is far beyond of the scope of the present thesis. In the design of the Network Manager we will not consider any specific algorithm. We will provide a design as flexible as possible for allowing implementation of several algorithms without changing the architecture and structure of the software.

4.1.2.2 HART Commands related to Routing

The HART commands used by the Network Manager for configuring the Network Route and manage the route tables for all the devices (including the Gateways, refer to section 4.3.2 and [17]) are:

- NL Command 969 Write Graph Connection
- NL Command 966 Delete Graph Connection
- NL Command 974 Write Route
- NL Command 975 Delete Route
- NL Command 976 Write Source-Route

4.1.3 Network Schedule

The most determinant and challenging function performed by the *WirelessHART Network Manager* is creating and managing the WHART Network Schedule.

As explained in chapter 3, *WirelessHART* is based in Time Division Multiple Access(TDMA) with frequency hopping. All communications between devices must be allocated within the WHART schedule, that is to say communications must be configured by the NM, in each device participating, in terms of Time Slot (TS) and Frequency Offset (FO). Every communication opportunity, determined by time and frequency is called *link*. Links are organized within *superframes*, which is a collection of links assigned to time slots repeating in time. Links are part part of the Data-Link Layer data tables of every Network Device.

Links are configured in every device depending on the features of each communication. For instance, regarding an unidirectional communication from device A and B, the Network Manager must assign: (1) Link in A where, in TS_0 and FO_0, A is configured to transmit and (2) Link in B where, in TS_0 and FO_0, B is configured to Receive. This example can be extend to other cases: broadcast, bidirectional or multicast transmissions and dedicated or shared transmissions Furthermore, this must be extended to multi-hop communications.

Superframes are organized by type within the Network Schedule. There are four types of superframes:

- **Management Superframe:** Used to schedule all communications related to the network management such as: join links, Request/response traffic, keep alives...

- **Data Superframe:** Configured to include the communication links related to publish data.
- **Gateway Superframe:** Superframe that includes links to the Access Points. Must be shared and active permanently since they are regarded as the portal to the Gateway and Network Manager.
- **Maintenance Superframe:** Used for schedule communications between handhelds and devices for maintenance reasons.

The entire WHART network schedule can be observed as a collection of superframes and links, which is only configured by the Network Manager. Each device is configured with the superframes and links, i.e. communication opportunities, where it participates.

Information about scheduling communications in the *WirelessHART* network can be found in sections 9.3.4, 9.5, 9.6 in Wireless Device Specification [18] and the specifications indicated in the sections of the present chapter where scheduling takes part.

4.1.3.1 Scheduling Algorithm

As deducted from the previous section, there exist plenty of communication needs in *WirelessHART*. Finding an optimal scheduling algorithm in *WirelessHART* is regarded as a challenging and complex task. The *WirelessHART* specification provides some guidelines but leaves up to the designer the scheduling algorithm, which is far beyond the scope of this thesis. However, We will provide a design of the Network Manager as independent as possible from the scheduling algorithm.

4.1.3.2 HART Commands related to Scheduling

The HART commands used by the Network Manager for configuring the Network Schedule for all the devices (including the Gateways, refer to section 4.3.2 and [17]) are:

- DL Command 965 Write Superframe
- DL Command 966 Delete Superframe
- DL Command 967 Write Link
- DL Command 968 Delete Link

4.1.4 Network Diagnostics and Adapting

4.1.4.1 Network Diagnostics

The Network Manager readjusts continuously to changes of the WHART network. For being aware of how well the network is performing, the Network Manager keeps track of the health information of each Network device. This is performed by maintaining record of the Health reports that the network devices send periodically to the Network Manager. Health reports contain information regarding communication performance between the wireless devices and their neighbors. Performance is measured using parameters such: packet loss, number of transmissions, number of retries, etc.

The Network Manager also maintains record of possible communication failures, which are indicated to the Network Manager through alarm HART commands. More information regarding Path failure procedure can be found in section 9.4.5 in Network Management specification [21].

4.1.4.2 Adapting

Using the information provided by the Network Diagnostics, health reports and alarms, The Network Manager updates the network schedule and route conveniently. This operation can be done periodically or triggered by performance conditions of the WHART network.

Furthermore, devices can request to the Network Manager for communication resources (bandwidth), i.e. slots in the TDMA schedule. This bandwidth is used for services between host applications and Field devices.

There are two type of services:

- **Block Data Transfer.** Data pipes between the Host Applications and Field Devices.
- **Publish Data.** Publishing process data to the Gateway. Host Applications can subsequently fetch the Process data from the Gateway's cache.

The Network Manager is responsible for allocating dynamically the service requests from the WHART network. This procedure may need to readjust the scheduling and routing tables of the devices involved in the determined communication.

The process of Service Request is explained in the standard in Block Transfer Specification [12], in Wireless Device specification, section 6.3.2, 6.3.3 and 9.6.2 [18],

4.1.4.3 HART commands related to Diagnostics and Adapting

The HART commands used by the Devices for reporting the Health to the Network Manager are shown bellow (refer to [17]) are:

- Command 779 Report Device Health
- Command 780 Report Neighbor Health List
- DL Command 787 Report Neighbor Signal Levels

In the other hand, the HART commands used by the Devices for reporting alarms to the Network Manager are (refer to [17]):

- DL Command 788 Alarm Path Down

- NL Command 789 Alarm Source Route Failed
- NL Command 790 Alarm Graph Route Failed
- TL Command 791 Alarm Transport Layer Failed

Finally, the HART command used by Field Devices for requesting services to the Network Manager is (refer to [17]):

- Command 799 Request Service

HART commands related to Routing and Scheduling may be issued by the Network Manager if there is a need of readjustment of the WHART network.

4.2 System Non-functional Requirements

Non-functional requirements, often called qualities of a system, describe how a system is supposed to be. In contrast with the functional requirements, they do not specify behavioral aspects of the software system.

4.2.1 Security

Security in *WirelessHART* is of great importance. The Network Manager is responsible for distributing all the keys of the WHART Network during the joining process . The NM must have a secure connection with the Security Manager, which is responsible for storing and providing the security keys.

All communications with the Network Manager are secured pipes using Session keys at the Network Layer level.

Regarding the security to be used in the communications between the Network Manager and the Gateway, the standard leaves an open question. It is up to the designer to decide which type of secure connection will be used. We propose a simple solution for solving this problem in section 4.3.2.

4.2.1.1 HART commands related to Security

The HART Command that are used for configuring the Security in the WHART network are:

- NL Command 768 Join Key
- DL Command 961 Write Network key
- NL Command 963 Write Session
- NL Command 964 Delete Session
- Command 823 Request Session

4.2.2 Reliability

Reliability is a key factor to take into account when managing industrial process applications. The Network Manager needs to be reliable for assuring the correct and controlled operation of the WHART network and, therefore the industrial applications. Reliability is provided by the Network Manager by the following mechanisms:

- **Redundancy in Routing** The Network Manager is responsible for configuring and managing routing of the WHART network. Creating graphs with redundant paths makes the system more reliable.
- **Redundancy in Scheduling** The Network Manager is responsible for managing the network schedule. By allocating links for different paths of the graph and allocating links for retries makes, the Network Manager can improve the reliability of the overall system.
- **Redundancy of the Network Manager itself** Network Manager redundancy is outside of the scope of the WirelessHART specification. Redundancy of the Network Manager itself can be ensured by having two instances, one active and

one in-standby Network Manager. Data of both processes must be synchronized correctly (refer to [18] section 8.8.2).

- **Reliability by Transport Layer** The WHART transport layer (see 3.3.4) provides mechanisms for ensuring end-to-end reliable communications between devices in the WHART network (included the Network Manager).

4.2.3 Performance

The performance of the Wireless HART Network Manager can be measured by:

- Maximum number of devices
- Time to Initialize Network
- Time to Join a Device
- Overall throughput

The WHART specification does not define any specific performance constraints for the creation and management of the WHART network. The only constraints are related to timing and they refer to the TIMEOUTs for requests to be served. These requests refer to operations from WHART devices to the Network Manager or vice versa. However, the TIMEOUTs are configurable and thus the constraints can be adapted to the particular needs.

The performance of the *WirelessHART* Network Manager also can be measured by the resulting performance of the network being managed. In this case, the main qualities that determine the performance of the Network Manager are:

- Latency
- Power utilization
- Number of hops to the Gateway
- Overall throughput

Performance will not be evaluated in this thesis since no implementation is provided.

4.2.4 Interoperability

This requirement is inherent to the *WirelessHART* specification. The Network Manager must be compatible with all HART-enabled devices from different manufacturers, by meeting all the requirements from the WHART specification. Since we propose some solutions to open questions of the standard (refer to 5.2), our system will not be compatible with the WHART gateways or WHART Security Managers unless they share common interfaces and functionalities[2].

[2]The standard does not specify,among other things, the interfaces between the NM and the GW and the NM and the SM

4.3 Considerations

Once the main requirements of the *WHART Network Manager* have been introduced, we will define the operation of our system regarding unspecified or unclear features in the *WirelessHART* standard.

4.3.1 Connection between Network Manager and Gateway

The question about the connection between the *WirelessHART Network Manager* remains open in the specification, i.e. *WirelessHART* does not define this connection. It is up to the designer whether to use Ethernet, Wifi, Serial or other type of connection between the Network Manager and the Gateway. It is possible also to merge the Network Manager and the gateway in the same host.

Since the physical connection is a matter of deployment, we are not going to consider any specific connection between the Network Manager and the Gateway. The design we proposed will be as independent as possible of the physical connection used.

4.3.2 Interface between Network Manager and Gateway

Another *WirelessHART* feature, which is not specified in the standard, is the interface between the Network Manager and the Gateway(s).

In the process of initialization of the WHART Network, the Network Manager must configure the Gateway(s) and Access Points in terms of initial superframes and links. This allows the gateway to begin advertisement, which provides joining information to the Field Devices. This configuration process must be secure and authenticated. The standard does not specify:

- Security and authentication to be used.
- Interface or API for configuring the Gateway.

That's the reason why we have to define a secure communication and interface. The solution we propose is to use:

- HART commands for the configuration of the Gateway
- The security and authentication provided by the WHART Network Layer (Secure end-to-end sessions).

Thus, the Gateway initialization will be a special case of the joining process, similar to the joining process of a Field Device.

The reasons why this decision was taken are:

- It is a particular case of configuring a Network Device with HART commands, not a case itself. Thus all the procedures will be the same.
- More security is provided since, apart from the private secure connection, it provides a secure pipe between both entities using WHART secure sessions, which keys are managed centrally by the Security Manager. This WHART secure session could even be considered the secure private connection required by the standard.

- No further interfaces have to be defined besides initialize and send/receive NPDU.

Nevertheless, there are some drawbacks that have to be taken into account such as:

- It will take more time to configure the Gateway since the Network Layer Packets have to be processed (encryption/Decryption) in both ends.
- More overhead.

Field devices send a Join Request (composed by 3 HART commands in response) to the network Manager when joining the network presenting the following credentials:

- Device's Identity (command 0)
- Long Tag (command 20)
- List of detected Neighbors (command 787)
- Join key (used for encrypting the NPDU)

In the other hand, we decided that the Gateway will present the following credentials when connecting to the Network Manager:

- Gateway's Identity
- Gateway's Long Tag
- Access Points' Identity
- Access Points' Long Tag
- Join key (used for encrypting the NPDU)

Since no HART command is provided by the WHART standard for providing this information to the Network Manager we define command 977 Gateway Join request (refer to appendix A.1).

The process of Joining of the Gateway will be as follows:

1. **Initial Provisioning**: Before attempting to join the Network the Gateway must be provided with the required Join Key and the Network ID.
2. **Presenting Credentials**: The gateway generates a command 977 Gateway Join request to the Network Manager, which will check if the presented credentials are trusted.
3. **Writing Keys and nicknames**: The Network Manager generates a Gateway Join request response with a *DR_initiated* status [3] and starts to configure the Virtual Gateway and Access Points by writing the Session keys, the network key and nickname. The Gateway acknowledges using the new session key.
4. **Configuring initial schedule and graph**: In this stage the Network Manager starts to configure the initial Superframes and links as well as the network graph necessary for the formation of the WHART network. At the end of this process the Network Manager sends the response of command 977 Gateway Join Request with a success state.

[3]Delayed Response. It indicates that the request it's being processed. After finishing the process of initialization the Network Manager will response with Success status.

5. **WHART Network epoch**: The WHART Network starts when the Network Manager activates the first superframe (ASN 0). The access points will start advertising allowing Network Devices join the the network.

4.3.2.1 Security and Authentication between the Network Manager and Gateway

HART commands encapsulated in WHART NPDUs are used for communicating between the Network Manager and the Gateway. Thus, the security and authentication is provided by the secure end-to-end Network Layer sessions. The gateway will use the Join Key for encrypting the Gateway Join Request command and later will be provided by session Keys with the Network Manager. Depending on the physical connection between the Network Manager and the Gateway, another security layer could be added on top. SSL is one of the possible solutions although it may be regarded as security over security.

4.3.2.2 Alternative interface between the Network Manager and Gateway

Another solution regarding the interface between the Network Manager and the Gateway is to define special service primitives for configuring the Gateway and Access Points. This service primitives could be defined in a similar fashion as the LOCAL_MANAGEMENT SP from the standard (refer to [21]). This service primitives must be sent in a secure way, for instance, using SSL for securing the communication between the NM and the GW.

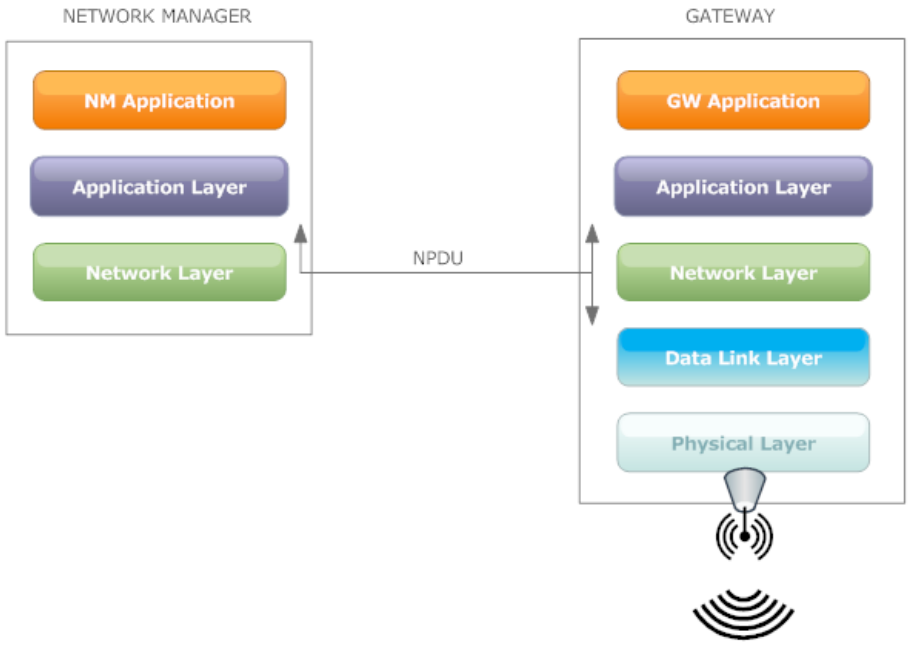

Figure 4.1: *Network Manager and Gateway interface.*

4.3.3 Time Synchronization in the WHART Network Manager

Time synchronization is critical in *WirelessHART* considering that the media access control is based in TDMA. All the network devices have to keep track of the time, in terms of ASN (Absolute Slot number)[4], in order to communicate with each other using the correct scheduled time slots. The entity responsible for providing the time clock to the overall WHART network is the Gateway.

When trying to design the *WHART Network Manager*, which is actually the creator of the WHART Network Schedule, a question comes up: does the WHART Network Manager have to keep strict Time Synchronization with the WHART Network? and, in this case, how will it be performed?

After the analysis of the WHART specifications and many discussions we decided that our *WHART Network Manager* will not keep a strict time synchronization with the WHART Network, since the gateway is responsible for the distribution of the time clock.

However, current time of the WHART Network is needed when constructing the Network Layer Packets in order to determine when the packet was created, i.e. the time of birth. This is used for discarding old packets in the other end. The attribute of the Network Layer that determines how old a packet should be for being discarded is the MaxPacketAge which is set by default to 300 sec.

The birth time of an NPDU is determined by the field ASN Snippet (refer to background, section network layer) and this field is set by the Network Manager when creating the NPDU. This field cannot be set by the Gateway for security reasons, the ASN snippet it is used when authenticating the NPDU by the NM. Thus NM has to know the approximate time of birth of the packet in terms of ASN.

As stated earlier, on the grounds that the *WHART Network Manager* does not need to keep strictly time synchronization with the WHART Network, an estimation of the current time can be performed. The solution we propose is to used command 794 Read UTC Time Mapping [17] which provides the time when the WHART Network was created with a precision of 1/32 ms. Once having this time, the only thing left for calculating the current time is knowing the current time in UTC with the same precision and divide the lapse between both by 10 ms for obtaining the current time in terms of ASN.

For making it work we need to accomplish some requirements:

- The UTC time of the *WHART Network Manager* and the Gateway has to be synchronized. This is required if the NM and the GW are deployed in two different physical boxes.
- The precision of the current UTC time has to be 10 ms order.
- The MaxPacketAge has to be adjusted appropriately.

[4] ASN 0 indicates the time when the WHART network was created and each ASN count means 10ms.

4.3.4 Regarding the HART Commands implemented

HART commands are the main tool of the *WHART Network Manager* for configuring the overall WHART network. There is a big set of commands that the Network Manager could implement for giving more functionalities to the system.

However, for narrowing the scope of the present Software we will consider only the Mandatory and Recommended commands specified in [17]. We designed the system in such a way that more functionalities related to new HART commands can be added easily without modifying the architecture and general design of the *WHART Network Manager*.

Chapter 5

WHART Network Manager. Design and Architecture

In this chapter the Software Architecture of the *WirelessHART Network Manager* will be depicted. All the requirements specified in the previous chapter are mapped and distributed into the different software components and classes forming the *WirelessHART Network Manager*. This chapter contains the most significant part of our work in the present thesis.

First, we will provide a brief background regarding the software design tools used in the thesis. Secondly, we introduce the general design considerations and, later, we will introduce the use cases, which depict graphically the main global functions and its synthesized description. Finally we will depict the general architecture and design of the Network Manager, concluding with the detailed description of its subcomponents.

5.1 Background. Software Design Tools

Since our work is based in Software Design and Architecture, it is important to introduce the reader to the Software Design Tools used: Unified Modeling Language (UML) and Software Design Patterns.

5.1.1 UML Unified Modeling Language

Unified Modeling Language (UML)[4] is the most-used standardized modeling language in the field of software engineering. The standard was created by the Object Management Group (OMG) and includes graphics for providing a visual description of a object-oriented software system[7]. UML models are used because they help to create software designs, they permit analysis and review of those designs and they can be used as the core documentation describing the software system. It is important to know that the objective of UML is to assist in software development, it is not a software development methodology itself.

UML describes a software system from two different points of view:

- **Static or Structural:** It depicts the structural elements composing a system or function, reflecting the static relationships of a structure. This view includes

class diagrams, component diagrams, composite structure diagrams and package diagrams.

- **Dynamic or Behavioral:** It focuses in the interaction of the system internally (among objects) and externally (with other systems or users) as well as the description of dynamic behavior such internal states. This point of view includes Activity, Use Case, Interaction and State Machine Diagrams.

5.1.1.1 Structural Diagrams

These diagrams reflect the static relationships of a structure, such as Class or Package diagrams, or run-time architectures such as Object or Composite Structure diagrams. They are really important when trying to document the architecture of a software system [8].
In the present thesis, Class diagrams and Component diagrams are used for describing the structure and architecture of the *WirelessHART Network Manager*. Thus, we will present a brief description of the mentioned diagrams.

Class Diagrams The Class diagram captures the logical structure of the system: the Classes and the relation between them. Class diagrams are most useful to illustrate relationships between Classes and Interfaces. Generalizations, Aggregations and Associations are all valuable in reflecting inheritance, composition or usage, and connections, respectively. Figure 5.1 shows an example of class diagram depicting several classes, including their attributes, operations and relations among them.

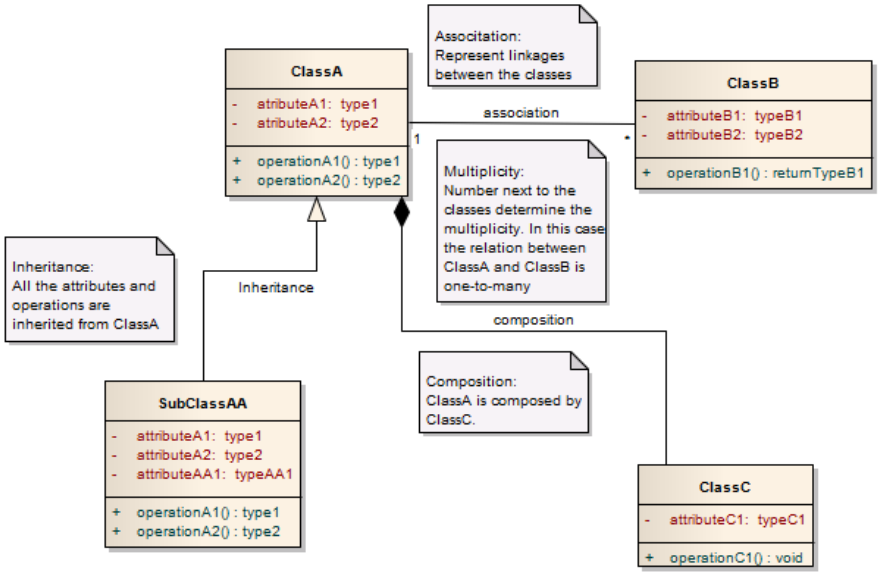

Figure 5.1: *Class diagram example showing classes with attributes and operations and relation among them.*

Component Diagrams Component diagrams illustrate the pieces of software, embedded controllers and such that make up a system, and their organization and depen-

dencies. They represent a higher level than the class diagram. Normally, the software components can be described internally by class diagrams. Figure 5.2 shows an example of component diagram depicting two components and their interfaces.

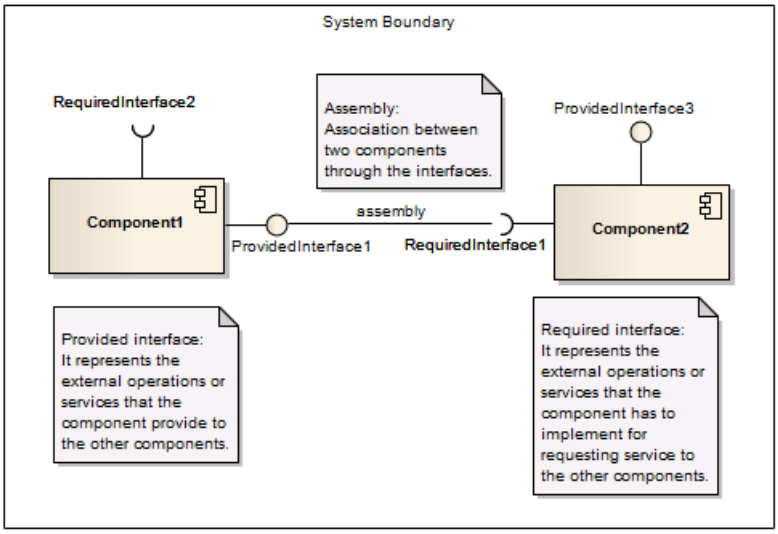

Figure 5.2: *Component diagram example showing the components and interfaces relating them.*

5.1.1.2 Behavioral Diagrams

Behavioral diagrams depict the behavioral features of a system or business process. In the present thesis, Use Case diagrams and Sequence diagrams are used for describing the dynamic behavior and interaction of the *WirelessHART Network Manager*. Thus, we will present a brief description of the mentioned diagrams.

Use Case Diagrams They capture Use Cases and relationships among Actors and the system. They describes the functional requirements of the system, the manner in which external operators interact at the system boundary, and the response of the system. Figure 5.3 shows an example of a use case diagram. A use case diagram can be documented with written use case description [9] in which the interaction of the system with the actors is specified.

Sequence Diagrams A sequence diagram is a type of interaction diagram. It represents series of sequential steps over time. It is used to depict work flow, message passing and how elements, normally classes or objects, cooperate in general over time to achieve a result. Figure 5.4 shows an example of sequence diagram.

5.1.2 Software Design Patterns

Software design patterns[10] are widely used in software engineering when designing a system. They refer to design solutions, already successfully used by others, suitable for a wide range of design problems. In the process of design of the present thesis,

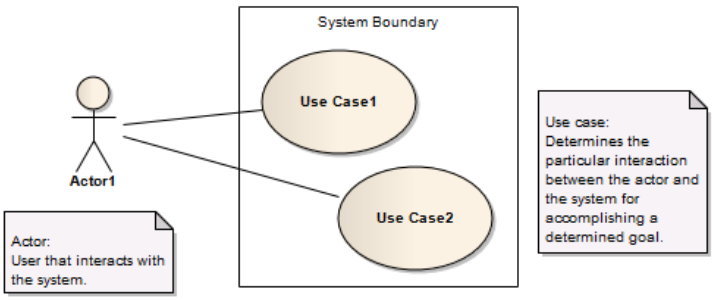

Figure 5.3: *Use case diagram example.*

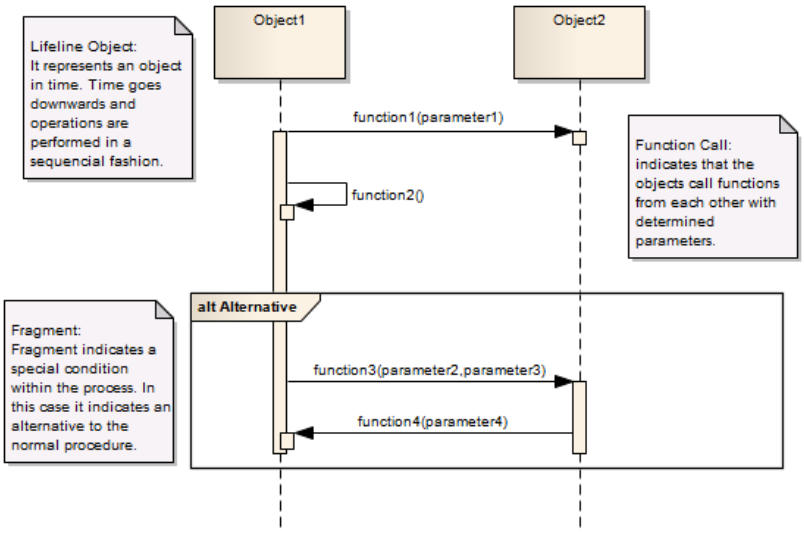

Figure 5.4: *Sequence diagram example showing objects interacting over time.*

some software design patterns has been used such *Singleton pattern*, *Factory pattern* and *Delegator pattern*.

5.2 Design considerations

Before going into the design details, it is important to mention the design assumptions, constraints and goals.

5.2.1 Assumptions and Dependencies

The *WirelessHART Network Manager* design is dependent on the Security Manager and Gateway since it communicates with them. The communication interfaces between the gateway and the Network Manager and between the Security Manager and the Network Manager have to be common.

5.2.2 General Constraints

The constraints are the ones specified in the previous chapter. The *WirelessHART Network Manager* has to be interoperable and has to take into account all the constraints determined by the *WirelessHART* specification.

5.2.3 Goals and Guidelines

The main goal for the WHART Network Manager design is to provide a flexible, extensible and modular simple architecture. Components should be replaceable and reusable.

5.3 Use Case Analysis

In this section we map the requirements specified previously into uses cases specifying the operations of the *WirelessHART Network Manager* from an external user point of view.
This is the first step for creating a software design of the *WirelessHART Network Manager*. In the next section we will define the users that interact with the Network Manager, i.e. the actors. Finally, we specify with details the most important use cases.

5.3.1 Actors

An actor indicates the particular role played by an entity, person or thing when interacting with the system. The actors can be classified into *primary* and *secondary actors* depending on how direct is the interaction.

5.3.1.1 Primary Actors

The primary actors interact directly with our system. They are listed bellow:

- **Security Manager** The Security Manager will be responsible for providing securely the necessary WHART keys to the *WHART Network Manager*.
- **Gateway** The Gateway will forward requests from the WHART network (From itself and the Network Devices) to the *WHART Network Manager* using their private connection.

- **Administrator** The Administrator of the system is the person able to monitor and configure the overall operations of the system.

5.3.1.2 Secondary actors

The secondary actors interact with the system trough another primary actor, in this case the Gateway. Our secondary actors are listed bellow:

- **Field Device** The Field Device will use the gateway as a bridge for communicating and requesting services to the Network Manager.
- **Handheld** The maintenance tool or Handheld will interact with field devices, locally or through the WHART network, and will request services to the Network Manager using the Gateway.

5.3.2 Primary Use Cases

Use cases determine the interactions between the actors and the studied system for accomplishing a determined goal. The next figure indicates the primary use cases of the *WHART Network Manager*. For a better understanding of the services provided by the studied system, we will provide a detailed description of the main use cases including goals, actors, preconditions, postconditions and sequence of operations.

5.3.2.1 Use case 1 - Initialize *WirelessHART* network (GW Join Request)

- **Actors:** -*Primary*: Gateway, Security Manager -*Secondary*: Field Device, Access Points
- **Goal:** Configure *WirelessHART* network for allowing devices to join.
- **Preconditions:** The Network Manager has a secure connection with Security Manager and the Gateway.
- **Trigger:** The gateway sends a Gateway Join Request to the Network Manager.
- **Related use cases:** includes: configure initial superframes, configure initial graph.
- **Success guarantee:** The gateway is configured correctly and the access points begin to advertise. New devices can listen to the advertisement and start joining the network by join requests.

    1. Gateway sends an GW JOIN request (Command 977) to the Network Manager with information about the identity of itself and Access Points.
    2. NM authenticates the GW and APs.
    3. NM responds with a DR_initiated status and processes the initialization.
    4. NM requests for Gateway's Session keys to the SM.
    5. NM sends the Network Key Session keys of the Gateway and Access Points by HART commands.

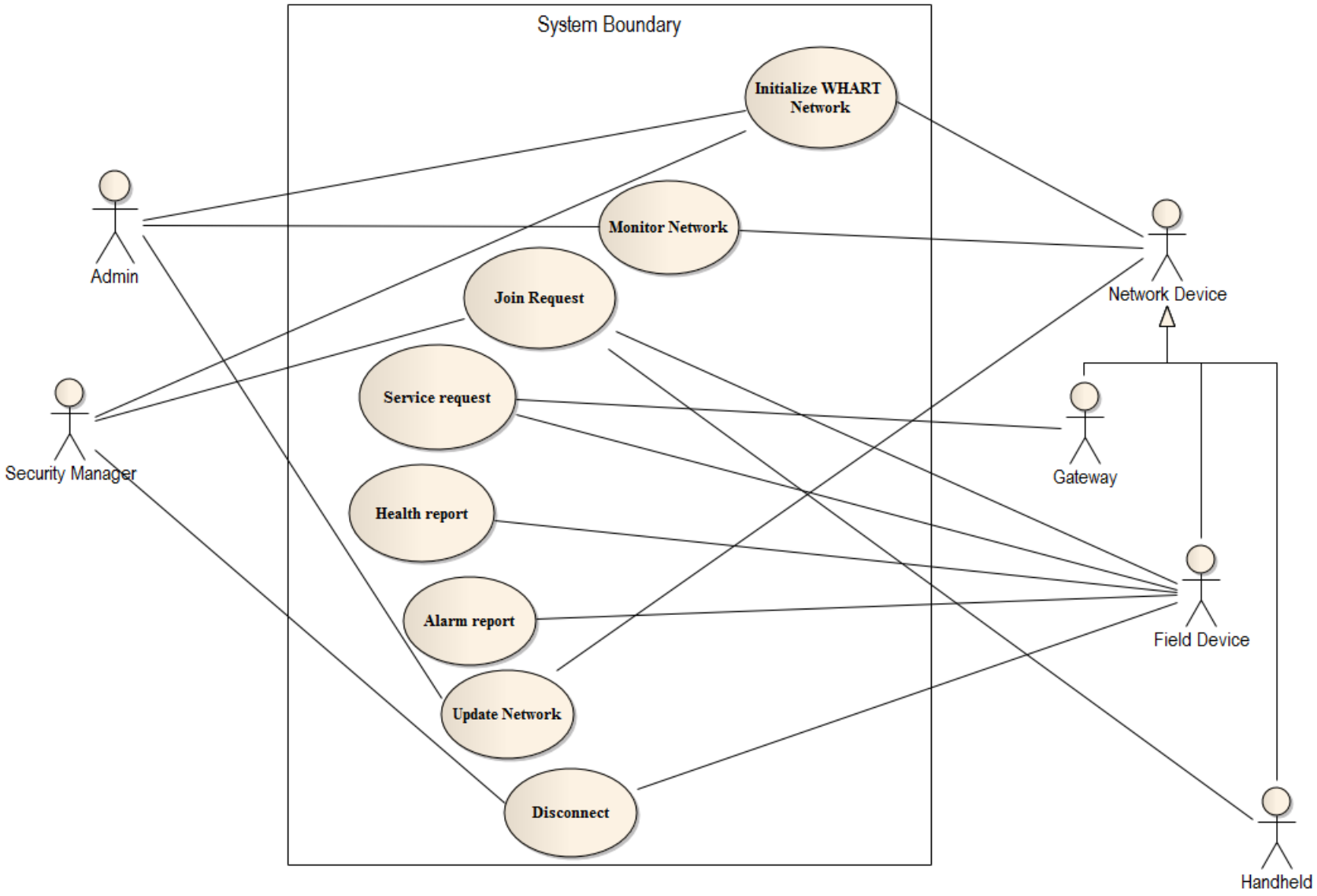

Figure 5.5: *Primary use cases of the* WirelessHART Network Manager.

6. NM assigns nickname to Access Points and writes them by HART commands.
7. GW sends a response to the HART commands using the configured new session key.
8. NM creates the initial Route Map and the initial Schedule.
9. NM sends commands for configuring the route tables, superframes and links.
10. GW sends a response to the HART commands.
11. NM sends the GW Join request response to indicate that the configuration is performed.
12. NM sends write superframe command (command 965) to activate the first superframe and thus the network is ready to advertise.
13. GW responds

- **Extensions:**

2a Gateway not trusted

2b Terminate with message *Gateway not trusted.*

4a Security Manager does not respond.

4b Terminate with message *Security Manager not responding*.

7a If t > TIMEOUT then retry number 6 and 7.

7b If number of retries is more that Max number of Retries then Remote Device not responding.

7c Terminate with message *Gateway not responding*.

10a If t > TIMEOUT then retry number 6 and 7.

10b If number of retries is more that Max number of Retries then Remote Device not responding.

10c Terminate with message *Gateway not responding*.

5.3.2.2 Use case 2 - Device joining the WHART network

- **Actors:** -*Primary*: Gateway, Security Manager -*Secondary*: Field Device, Handheld
- **Goal:** Integrate a Device to be able to fully participate in the *WirelessHART* network
- **Preconditions:** The Network Manager has a secure connection with Security Manager and the Gateway. The wireless HART network is already initialized and the access points and joined devices are already advertising.
- **Trigger:** The Field Device sends a Join request to the Network Manager through the Gateway (Network Manager will receive the join request through the wired connection to the Gateway).
- **Related use cases:** includes: Add Device to Route map and add Device to schedule

- **Success guarantee:** The joining Device is configured correctly and can participate actively in the network.

    1. FD sends an Join request (Command 0,20 and 787 in response mode) to the Network Manager with the credentials.
    2. NM authenticates the joining device.
    3. NM requests for Device's Session keys to the SM.
    4. NM sends the Network Key and Session keys of the Device by HART commands.
    5. FD sends a response to the HART commands using the configured new session key.
    6. NM assigns nickname and send HART commands to configure device.
    7. NM provides clock parent to the device by HART command.
    8. NM adds the FD to the Route Map and the network Schedule.
    9. NM sends commands for configuring the route tables, superframes and links to all devices affected by the new configuration.
    10. FDs send a response to the HART commands.
    11. NM request for Gateway session key with the FD.
    12. NM sends HART command writing the Gateway Session key to the FD.
    13. FD responds the HART command and becomes OPERATIONAL.

- **Extensions:**

    2a Device not trusted

    2b Terminate with message *Device not trusted.*

    3a Security Manager does not respond.

    3b Terminate with message *Security Manager not responding*.

    5a If t > TIMEOUT then retry number 6 and 7.

    5b If number of retries is more that Max number of Retries then Remote Device not responding.

    5c Terminate with message *Device not responding*.

    10a If t > TIMEOUT then retry number 6 and 7.

    10b If number of retries is more that Max number of Retries then Remote Device not responding.

    10c Terminate with message *Device not responding*.

    11a Security Manager does not respond.

    11b Terminate with message *Security Manager not responding*.

    13a If t > TIMEOUT then retry number 6 and 7.

    13b If number of retries is more that Max number of Retries then Remote Device not responding.

    13c Terminate with message *Device not responding*.

5.3.2.3 Use case 3 - Device's Service Request

- **Actors:** -*Primary*: Gateway -*Secondary*: Field Device
- **Goal:** Allocate bandwidth needed by a Field Device to perform certain task
- **Preconditions:** The Network Manager has a secure connection with Security Manager and the Gateway. The wireless HART network is already initialized and the Field Device requesting bandwidth to the Network Manager has already joined the network.
- **Trigger:** The Field Device sends a Service Request to the Network Manager through the Gateway (Network Manager will receive the join request through the wired connection to the Gateway).
- **Related use cases:** includes: Schedule service
- **Success guarantee:** Bandwidth is allocated to the Field Device to perform the service

  1. FD sends a Service request (Command 799) to the NM.
  2. NM responds with a DR_initiated status and processes the request.
  3. NM processes by allocating the requested bandwidth in the schedule. If necessary updates teh configuration of the network.
  4. NM sends HART commands to the FD to configure the superframes (commands 965) and links (Command 967) allocated and the route tables (command 969-976) if necessary.
  5. FD responds to acknowledge the HART commands.
  6. NM sends the Service Request response to indicate that the configuration is performed.
  7. FD can perform the service.

- **Extensions:**

  2a If no bandwidth available then NM responds the Service Request indicating so.

  5a If t > TIMEOUT then retry number 6 and 7.

  5b If number of retries is more that Max number of Retries then Remote Device not responding.

  5c Terminate with message *Device not responding*.

5.3.2.4 Use case 4 - Health report update

- **Actors:** -*Primary*: Gateway -*Secondary*: Field Device
- **Goal:** Update the health of the Wireless HART network in terms of quality and performance of the different links.
- **Preconditions:** The Network Manager has a secure connection with Security Manager and the Gateway. The wireless HART network is already initialized and the Field Device reporting the health to the Network Manager has already joined the network.

- **Trigger:** The Field Device sends a Health report () to the Network Manager through the gateway. This operation can be performed periodically.
- **Related use cases:** It may trigger use case 6 - *Update Network*.
- **Success guarantee:** The Network Manager stores the updated information about the Health of the Network.

  1. FD sends a Health report to the NM through the GW.
  2. NM verifies the Health report.
  3. The NM stores the information within the correspondent Health report in the database.
  4. If the health of the WHART network is under the performance threshold then Update Network.

- **Extensions:**

  11 Device Health Report (HART Command 779).

  12 Neighbor Health List Report (HART Command 780).

  13 Neighbor Signal Levels Report (HART Command 787).

  2a Health report arrives before Health report update timeout.

  2b Discard Health report.

5.3.2.5 Use case 5 - Alarm report

- **Actors:** -*Primary*: Gateway -*Secondary*: Field Device
- **Goal:** Monitor the communications failures indicated by the Alarm reports.
- **Preconditions:** The Network Manager has a secure connection with Security Manager and the Gateway. The wireless HART network is already initialized and the Field Device reporting the Alarm report to the Network Manager has already joined the network.
- **Trigger:** The Field Device send an Alarm report to the Network Manager through the Gateway (Network Manager will receive the join request through the wired connection to the Gateway).
- **Related use cases:** It may trigger use case 6 - *Update Network*.
- **Success guarantee:** The Network Manager stores the updated information about the communications failures of the Network.

  1. FD sends a Health report to the NM through the GW.
  2. NM verifies the Alarm report.
  3. The NM stores the communication failure within in the database.
  4. If the number alarm reports for the same failure is over a determined threshold then Update Network.

- **Extensions:**

11 Path Down Alarm Report (HART Command 788).

12 Source Route Alarm Report (HART Command 789).

13 Graph Route Alarm Report (HART Command 790).

14 Transport Layer Alarm Report (HART Command 791).

2a Alarm report not valid or redundant.

2b Discard Alarm report.

5.3.2.6 Use case 6 - Update network

- **Actors:** -*Primary*: Gateway, Administrator -*Secondary*: Field Devices, Routers, Adapters
- **Goal:** Update the configuration of the network regarding routing and scheduling.
- **Preconditions:** The Network Manager has a secure connection with Security Manager and the Gateway. The wireless HART network is already initialized and there are Field Devices joined to the network.
- **Trigger:** Updating the network can be done periodically, triggered by the Administrator or triggered by a succession of Health reports or Alarms.
- **Related use cases:** includes: update Route Map and update Schedule
- **Success guarantee:** The Route map and the Schedule are updated so that a better performance is achieved.

  1. The NM evaluates the available health information about the network (Health reports and alarms).
  2. The NM executes a Route Map optimization using the mentioned information.
  3. The NM sends HART commands that configures the new routing tables for each Field Device affected.
  4. The NM waits for the responses
  5. The NM executes a Scheduling optimization using the health information.
  6. The NM sends HART commands that configures the new routing tables for each Field Device affected.
  7. The NM waits for the responses.
  8. Updating the network is performed.

- **Extensions:**

  1a If the performance is acceptable, terminate routine.

  4a If t > TIMEOUT then retry number 3.

  4b If number of retries is more that Max number of Retries then Remote Device not responding.

  4c Disconnect not-responding device then update network again.

  7a If t > TIMEOUT then retry number 6.

  7b If number of retries is more that Max number of Retries then Remote Device not responding.

  7c Disconnect not-responding device then update network again.

5.4 System Software Architecture

In this section we will present the features of the architecture design for the *WHART Network Manager*.

5.4.1 General System Architecture

Figure 5.6 shows the general system architecture of the *WHART Network Manager*. We define a *component-based architecture* with interface oriented communication between components. This is a new approach to the *Network Manager* since no software design architecture appears in the standard. The benefits of the defined architecture are:

- **Modularity** The advantage of using modular architecture is that a component (module) can be replaced easily without affecting the rest of the system.The division of *WirelessHART Network Manager* into different components reduces the complexity of the system. Responsibilities are distributed and smaller problems will be handled by each component. For example, the Security Manager Client component can be replaced if different Security Manager is used and the system will remain unaffected as long as the interfaces are implemented correctly.
- **Reusability** The system has been divided in such a way that components con be reused in other WHART software systems such as the *Gateway* and *Field device* with really small changes. In this case, the Application Layer and the Network Layer components are reusable.
- **Decoupling** The components are coupled to the interfaces defined between components but not to the other components. That makes the division of tasks in a software development team much more easier.

We decided to divide the system into five software components:

- **Network Manager Core Component** Responsible for the high-level operations such as formation and configuration, routing, scheduling and monitoring of the WHART Network.
- **Security Manager Client** Responsible for ensuring a private secure connection with the Security Manager and providing an interface for requesting keys from it.
- **Network Manager Server** Responsible for ensuring a private secure connection and providing an interface for initializing the Gateway and sending/receiving Network Layer packets to/from the Gateway.
- **Network Manager Application Layer**[1] Responsible for handling and formatting HART commands, from specific HART commands for configuring the different layers to Network Management Commands.
- **Network Manager Network Layer**[2] Responsible for ensuring secured end-to-end communications with the network devices and (Transport Layer and security sub-layer) constructing the Network Layer packet.

[1] The Application Layer of the *WHART Network Manager* will add functionalities to the WHART Application Layer of the Field Device, i.e. it will be a superset of it.

[2] The network Layer of the *WHART Network Manager* will have less functionalities than the WHART Network Layer of a Field Device, i.e. it will be a subset of it.

The *WirelessHART Network Manager* interacts with the Security Manager and Gateway, defined as primary actors in the previous chapter. The main component in the design is the Network Manager Core Component.

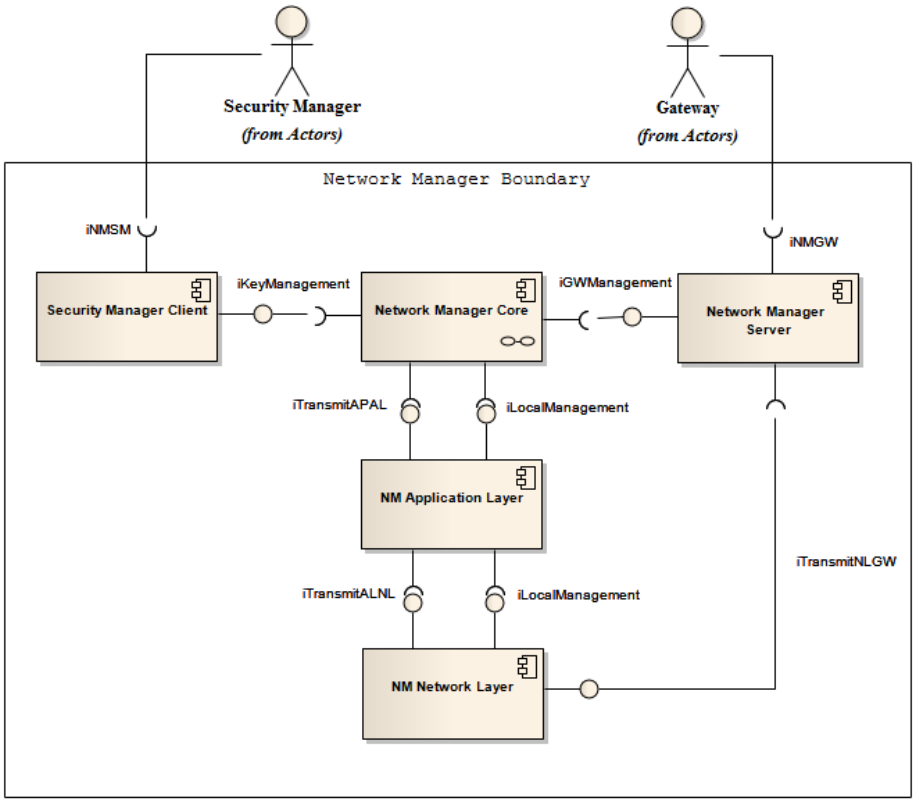

Figure 5.6: *Wireless HART network manager architecture.*

The *WirelessHART* Standard [19] specifies that the Network Manager is an application on top of the *WirelessHART* protocol stack (refer to network manager scope in Network Device Specification). However, this could lead to misinterpretations since the Network Manager must handle secure sessions in the WHART network and end-to-end communications with WHART Network Devices. The WHART network layer, which includes transport and security sub-layer, is the responsible for keeping track of the sessions, encryption-decryption and assuring end-to-end communications. That is why we decided to include the WHART network layer component in the design. This software component can be reused in other devices such the Gateway or the Field Devices.

Furthermore, the *WHART Network Manager* has to handle HART commands since all the network transactions are performed using them. This specific responsibilities are delegated to the *WirelessHART* Application layer. We include a *NM Application Layer* component in the design, that is responsible for handling commands formatting and interpretation. In this way, the *Network Manager Core Component* is transparent to the HART commands construction. The Network Manager Core Component is responsible for the core tasks regarding network formation, routing, scheduling and monitoring the

network. For that reason, we provide an subarchitecture design for this component in the next section.

Since the Network Manager has to handle secure communications with the Security Manager and Gateway, we have provided the Security Manager Client and the Network Manager Server to perform this task. In this manner, changing the Security Manager or Gateway implies changing the mentioned components without affecting the rest of the design.

5.4.2 Subsystem Architecture. Network Manager Core Component

On the grounds that the main responsibilities of the *WHART Network Manager* are delegated to the Network Manager Core Component (NMCC), we present a special section where the internal architecture of this subsystem is explained. The NMCC is the center of the whole design. It represents the source and the sink[3] of any request towards / from the WHART network. The critical responsibilities are Routing and Scheduling, as stated in section 4.1.2 and 4.1.3.

The main implication of this design choice is that the functions of Routing and Scheduling are performed separately, i.e. routing is performed independently from scheduling and scheduling is performed after having a routing map already calculated. Therefore, a combined algorithm that will consider cross correlation between routing and scheduling will not be suitable with our design. The reason why we chose this architecture was to simplify the complexity of the process of routing and scheduling.

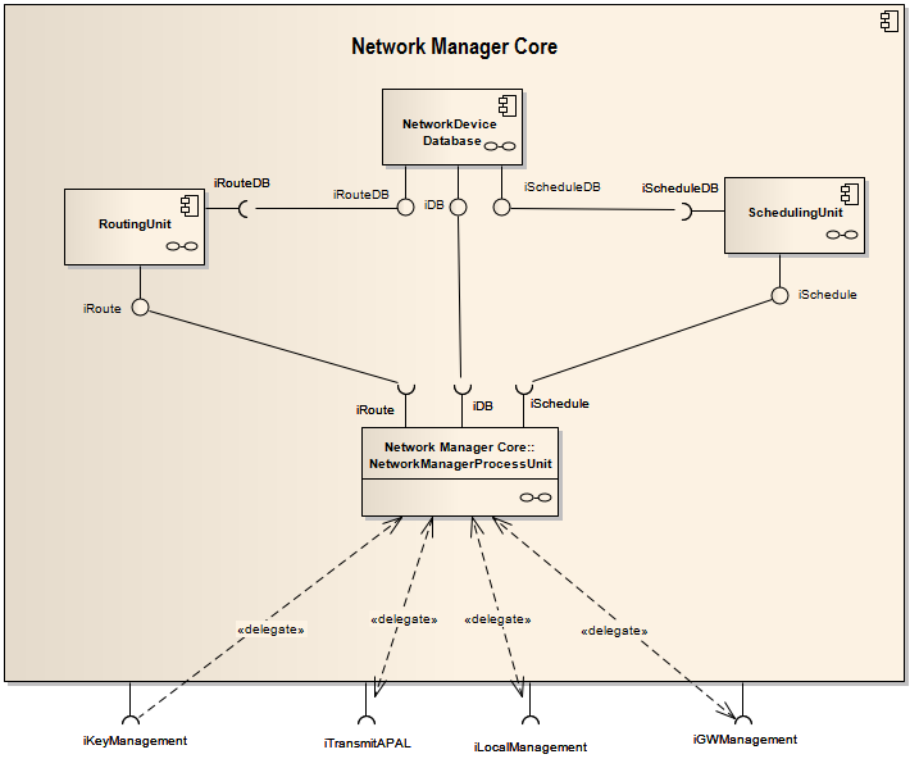

Figure 5.7: *Wireless HART Network Manager Core Component architecture*

[3]Within the overall architecture of the *WHART Network Manager.*

5.5 Detailed System Software Design

In this section the architecture of the Network Manager Core component introduced in the previous section will be described thoroughly including subcomponents and classes. The rest of the components will be described briefly without introducing details about the internal architecture. Furthermore, it is very important to understand how the different components from the general architecture model are going to communicate with each other and how the information will flow. That is why we will first present and define the interfaces introduced in the general architecture model (refer to figure 5.6).

5.5.1 Interfaces in the General Architecture

The detailed description of the service primitives and interfaces introduced in the general model architecture will be presented in the following subsections. Figure 5.8 shows the specific interfaces that we defined for communicating between the different software components of the general model.

First, the description of every service primitive or function will be depicted and later on some examples of use will be shown indicating the information flow between components.

5.5.1.1 Network Manager Application Layer Services. iTransmitAPAL

In order to separate the design of the Application Layer of WHART Network Manager from the actual Network Manager application we need to define the Service Primitives for communicating. It is important to notice that, although they look similar, this service primitives are not in the standard, we used similar semantics for simplifying the understanding. In the standard, Data Link and Network Layer Service Primitives are defined, no Application Layer Service Primitives appear. The Service Primitives are used for calling to services that the application layer is going to provide to the Network Manager Core Component or vice versa.

TRANSMIT.request (handle, dest, service, [data]) This Service Primitive will be called by the Network Manager Core Component when trying to send a HART command over the WHART network.

- **handle** The packetHandle is supported for the convenience of the Application. The Application Layer returns this handle in the correspondent TRANSMIT.confirm, specified later on.
- **dest** Indicates the packet's destination. There are three possibilities.
  - **uniqueID** The long destination address.
  - **Nickname** The short destination address previously assign by the *WHART Network Manager*.
  - **Broadcast** Short broadcast address.
- **service** The service corresponds to the specific HART command that want to be send. The Application Layer can assemble different *TRANSMIT.request* in the same packet if the destination address it is the same and the rules for assembling commands are respected(refer to Network Management specification, p.56). The

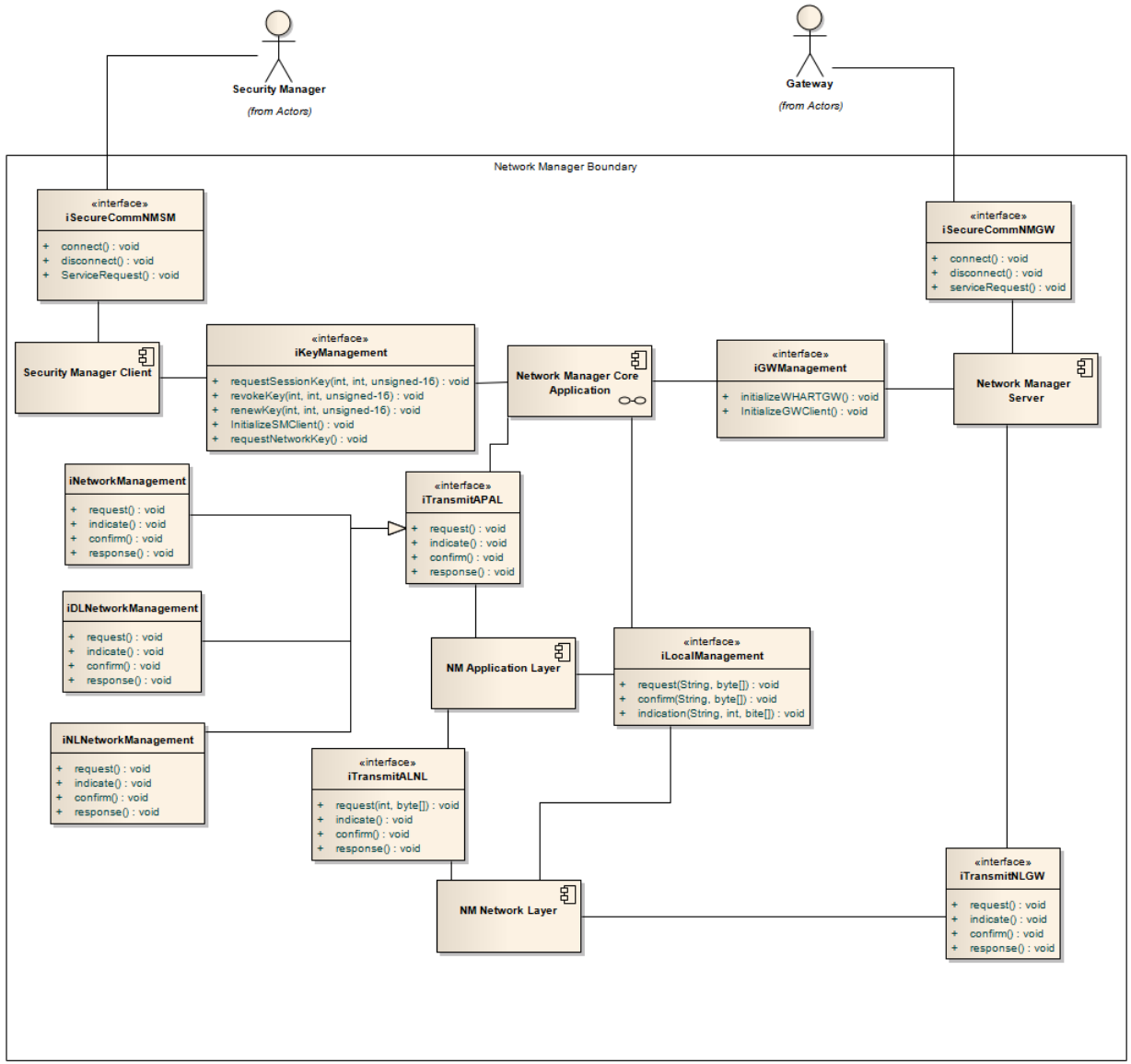

Figure 5.8: *Wireless HART network manager architecture including interfaces*

services names correspond to the mandatory command names from the Data Link Layer, Network Layer and Network Manager commands in Wireless Command Specification. In table **??** the service related to Network Management are indicated.

- **[data]** The data associated to the service. It is specified in the data structure of each command in Wireless Command Specification for simplicity.

Service	Data	Description
DISCONNECT_DEVICE		Removes a device from the network
	Unsigned-8 reason	The Reason of disconnection (See common table 50. Disconnect Cause Codes).
WRITE_NETWORK_KEY		Writes Network key on a Network Device
	Unsigned-128 key	Key value
	Unsigned-40 execTime	Execution time for command (ASN). 0 immediately
WRITE_DEVICE_NICKNAME		Reads the device nickname, i.e. short address
	Data	Same data order as specified in command 963 in [17]. Request data in the TRANSMIT.request and response data in TRANSMIT.indicate
WRITE_SESSION		Write session in Network Device.
	Data	Same data order as specified in command 963 in [17]. Request data in the TRANSMIT.request and response data in TRANSMIT.indicate
DELETE_SESSION		Deletes a session from a Network Device.
	Data	Same data order as specified in command 964 in [17]. Request data in the TRANSMIT.request and response data in TRANSMIT.indicate
WRITE_SUPERFRAME		Write a new superframe entry to a Network Device.
	Data	Same data order as specified in command 965 in [17]. Request data in the TRANSMIT.request and response data in TRANSMIT.indicate
DELETE_SUPERFRAME		Deletes a superframe entry from a Network Device.
	Data	Same data order as specified in command 966 in [17]. Request data in the TRANSMIT.request and response data in TRANSMIT.indicate
WRITE_LINK		Writes a link entry to a Network Device.
	Data	Same data order as specified in command 967 in[17]. Request data in the TRANSMIT.request and response data in TRANSMIT.indicate
DELETE_LINK		Deletes a link entry to a Network Device.
	Data	Same data order as specified in command 968 in [17]. Request data in the TRANSMIT.request and response data in TRANSMIT.indicate
WRITE_GRAPH_CONNECTION		Adds a new graph connection entry to a Network Device.
	Data	Same data order as specified in command 969 in [17]. Request data in the TRANSMIT.request and response data in TRANSMIT.indicate
DELETE_GRAPH_CONNECTION		Deletes a previously defined graph connection entry to a Network Device.

Continued on next page

Service	Data	Description
	Data	Same data order as specified in command 970 in [17]. Request data in the TRANSMIT.request and response data in TRANSMIT.indicate
WRITE_NEIGHBOR_PROPIERTIES		Writes the Neighbor table to a Network Device and sets which ones are the clock source.
	Data	Same data order as specified in command 971 in [17]. Request data in the TRANSMIT.request and response data in TRANSMIT.indicate
WRITE_SUSPEND_DEVICES		Suspends operation of one or more devices.
	Data	Same data order as specified in command 972 in [17]. Request data in the TRANSMIT.request and response data in TRANSMIT.indicate
WRITE_SERVICE		Notifies the network device of a new service allocated to it.
	Data	Same data order as specified in command 973 in [17]. Request data in the TRANSMIT.request and response data in TRANSMIT.indicate
WRITE_ROUTE		Writes a new route to the network device.
	Data	Same data order as specified in command 974 in [17]. Request data in the TRANSMIT.request and response data in TRANSMIT.indicate
DELETE_ROUTE		Deletes a route of the network device.
	Data	Same data order as specified in command 975 in [17]. Request data in the TRANSMIT.request and response data in TRANSMIT.indicate
WRITE_SOURCE_ROUTE		Adds a specific source route to the network device.
	Data	Same data order as specified in command 976 in [17]. Request data in the TRANSMIT.request and response data in TRANSMIT.indicate

TRANSMIT.indicate (handle, sourceAddr, service, [data]) This service primitive will be called by the Application Layer for indicating that there has been a request, Health report or Alarm from the WHART network.

- **handle** The packetHandle is supported for the convenience of the Application. The Application Layer returns this handle in the correspondent TRANSMIT.response, specified later on.
- **sourceAddr** Indicates the packet's source Address. There are three possibilities.
  - **uniqueID** The long destination address.
  - **Nickname** The short destination address previously assigned by the *WHART Network Manager*.
- **service** The services supported are specified in table 5.2. Notice that this services differ from the ones from TRANSMIT.request.
- **[data]** The data associated to the service is specified in table 5.2.

Table 5.2: Services provided by the Network Manager.

Service	Data	Description
JOIN_REQUEST		Join request from a Network Device which includes commands 0, 20 and 787 in response [17].
	Unsigned-40 uniqueID	Unique Identifier of the field device. It is composed by the expanded type (Unsigned-16) and deviceID (Unsigned-24) [16]
	Latin-1 longTag	Long Tag of the network device
	Neighbor Signal Levels	Same data structure as specified in response command 987 [17].
REPORT_DEVICE_HEALTH		Provides information about the device's communication statistics.
	Data	Same data structure as specified in response command 779 [17].
REPORT_NEIGHBOR_HEALTH		Provides information about the neighbors' communication statistics.
	Data	Same data structure as specified in response command 780 [17].
REPORT_NEIGHBOR_SIGNAL_LEVELS		Provides information about the neighbors' communication statistics.
	Data	Same data structure as specified in response command 787 [17].
ALARM_PATH_DOWN		Network device notifies the network manager that the path to a neighbor failed.
	Data	Same data structure as specified in response command 788 [17].
ALARM_SOURCE_ROUTE_FAILED		Network device notifies the network manager that a source route failed.
	Data	Same data structure as specified in response command 789 [17].
ALARM_GRAPH_ROUTE_FAILED		Network device notifies the network manager that a graph route failed.
	Data	Same data structure as specified in response command 790 [17].
ALARM_TRANSPORT_LAYER_FAILED		Network device notifies the network manager that a transport layer failed.
	Data	Same data structure as specified in response command 791 [17].
SERVICE_REQUEST		Network device requests additional bandwidth for providing a service to a particular application.
	Data	Same data structure as specified in request command 799 [17].
ACTIVE_ADVERTISE_REQUEST		Network device requests active and fast advertising.
	Data	Same data structure as specified in response command 770 [17].
REQUEST_HANDHELD_SESSION		Network device notifies the network manager that a graph route have failed.
	Data	Same data structure as specified in response command 790 [17].

TRANSMIT.response (handle, [data]) This service primitive is used by the Network Manager to respond to the requests from the network which will be notified previously by the TRANSMIT.indicate SP. There will be response only for service request, active advertise request and hanheld session request.

- **handle** The handle is supported for the convenience of the Application. The Application Layer gives this handle in the correspondent TRANSMIT.indicate, specified above.

- **[data]** The data associated to the service. It is specified in the table 5.2.

TRANSMIT.confirm (handle, ResponseMessage, [data]) This primitive will notify the Network Manager that a request has been responded by the remote network device.

- **handle** The packetHandle is supported for the convenience of the Application Layer. The Application gives this handle in the correspondent TRANSMIT.request, specified above.

- **ResponseMessage** Message from the Command-Specific Response Code. It is compose by the result of the transaction (Success or Error) and the description in [17].

- **[data]** The data associated to the service. The service can be determined by the handle and the related data is indicated in **??**.

5.5.1.2 Local Management service primitives. iLocalManagement

We will use the Local Management Service Primitives defined in the Network Management specification [21]. They are used by the Network Manager Core Component to configure the local Network layer tables and attributes. For example, when creating new session for the joining device the NMCC will call to TRANSMIT.request from the iTransmitAPAL for sending a HART command that will write a session in the Field Device. At the same time the NMCC will call to a LOCAL_MANAGEMENT.request for configuring the session locally in the NM Network Layer. Notice that, for respecting the layered architecture, the NMCC manages the NM Network Layer using the NM Application Layer. At this stage the NMAL just forwards the service primitive to the NMNL without being affected. In the future Local Management SP could be added for configuring the NMAL as well.

5.5.1.3 NM Network Layer Service Primitives. iTransmitALNL

We will use the same service primitives specified in the standard for WHART Network Layer [21]. They are used by the NM Application Layer in order to send the aggregated commands to the WHART network. The NMNL also will indicate the NMAL that incoming packet has arrived from the WHART network and will provide the aggregated commands to the NMAL. All the information needed for understanding the functions of the Network Layer Service Primitives can be found in [21].

5.5.1.4 NM Network Layer to NM Server Service Primitives. iTransmitNLGW

This service primitives will provide an interface between the NM Network Layer and NM Server components. We define this service primitives in order to indicate the Network Manager Server to send a new packet to the Gateway or vice versa. It is important to notice that, although they look similar, this service primitives are not defined in the standard, we used similar semantics for simplifying the understanding. In the standard, Data Link and Network Layer Service Primitives are defined.

TRANSMIT.request (handle, NPDU, priority) This service will be called by the Network Layer when requesting to send a Network packet to the Gateway using the Network Manager Server.

- **handle** The handle is supported for the convenience of the Network Layer. The Network Manager Server gives this handle in the correspondent TRANSMIT.confirm, specified bellow.
- **NPDU** The actual Network Layer packet that has to be injected into the WHART network.
- **priority** Priority of the packet determined by the payload. It can be Command, Process Data, normal or alarm (refer to [21]).

TRANSMIT.confirm (handle, GWstatus) This service primitive will be used by the NM Server to indicate that the Network Layer packet has been sent into the WHART network.

- **handle** The handle is supported for the convenience of the Network Layer. The Network Layer gives the same handle in the correspondent previous TRANSMIT.request, specified above.
- **GWstatus** Informs about possible errors in injecting the packet into the WHART network by the Gateway.

TRANSMIT.indicate (handle, NPDU, priority) This service primitive will be used by the NM Server to indicate the NMNL that a Network Layer packet has been received from the WHART network

- **handle** The handle is supported for the convenience of the Network Layer.
- **NPDU** The actual Network Layer packet from the WHART network.
- **priority** Priority of the packet determined by the payload. It can be Command, Process Data, normal or alarm (refer to [21]).

5.5.1.5 Key Management interface. iKeyManagement

In this section, we will describe the interface between the Network Manager Core Component and the Security Manager Client. The definition of an interface will make the design of both mentioned components decoupled. Also, as stated in section 5.4.1 the SM Client component will be easily replaceable to suit the Security Manager used for providing the correspondent keys.

void initializeSMClient() This function will initialize the Security Manager Client for creating a secure connection with the Security Manager. After this step, the SM Client will be ready to serve all the requests from the Network Manager Core component.

SecKey requestSessionKey (networkID, keyType, deviceID) This function is going to be used by the Network Manager Core Component when creating a new secure session with a WHART network device, i.e. in the Joining process or when initializing (secure session with gateway). The Session Key for the device will be returned.

void revokeSessionKey(networkID, keyType, deviceID) This function will be used by the NMCC when eliminating a device from the WHART network, i.e. after the disconnect command [17].

SecKey renewSessionKey(networkID, keyType, deviceID) Function used by the NMCC to renew a Session key to a particular network device.

SecKey requestNetworkKey(networkID) This function will be called by the NMCC for requesting the security key used for the Data Link Layer encryption. It will be used when initializing the WHART network in order to configure the WHART network devices. The new Session Key for the device will be returned.

SecKey renewNetworkKey(networkID) Used by the NMCC when changing the Network Key of the WHART network. The new Network Key will be returned.

5.5.1.6 Gateway Management interface. iGWManagement

This interface defines the interaction between the NMCC and the NM Server. Since all the configurations will be made by HART commands (refer to previous chapter, section considerations) the only interface provided is the initialization request.

boolean InitializeGW (networkID, GWuniqueID, APUniqueID) This function will be called by the NM Server when receiving a Initialization request from the Gateway. This will trigger the process of initialization of the WHART Network.

5.5.1.7 Information Flow

We will present how the information will flow between components in the case of normal requesting from the WHART Network and from the WHART Network Manager. This visual aid will help to understand the relationship between components.

In the case of a request from the WHART network (Fig. 5.9), for example a Join request [4] , the Gateway will forward the NPDU to the *WirelessHART Network Manager* using the secure private connection (using Network Manager Client). The NM Server will receive the encrypted [5] NPDU, then decrypt it and indicate it to the NM

[4] We define it as a request although the actual HART commands are 0, 20 and 787 in response mode [21, 16, 17].

[5] Meaning wired encryption used for secure communicating between NM and GW, NOT WHART Session encryption.

Network Layer using the correspondent TRANSMIT.indicate service primitive from iTransmitNLGW. After that, the NMNL will decrypt the NPDU using the correspondent Session Key and will check from the Transport table whether the packet is the response to a previously sent packet. In the particular case of a request from the network the NMNL will indicate the NMAL that a new packet has arrived from the network using the correspondent TRANSMIT.indicate service from iTransmitALNL. The NMAL will receive the aggregated commands from the payload of the NPDU and will analyze them. In the case of Join Request this aggregated commands will be HART command 0, 20 and 787 (refer to [17]). The NMAL will extract the data from them and will indicate the NMCC of a new request from the WHART using the correspondent TRANSMIT.indicate (service: JOIN_REQUEST) service primitive from iTransmitAPAL. The NMCC, after receiving the data of the correspondent service, will handle the request. In the case of Join Request, the NMCC will proceed generating requests for configuring the particular device by calling TRANSMIT.request's from iTransmitAPAL (Fig. 5.10). In case of a Service Request, Active Advertise Request or Handheld Session Request, the NMCC will have to respond after processing the request using the TRANSMIT.response from iTransmitAPAL. The response data will be indicated to the NMAL which will format the command(s) and will call to TRANSMIT.response from iTransmitALNL. The NMNL will encrypt and construct the NPDU and will forward it to the NM Server using TRANSMIT.request from iTransmitNLGW. The NM Server will encrypt the data and send it to the Gateway's NM client. After the packet being sent into the WHART network the Gateway will send a confirmation to the NM server which will indicate the NMNL that the packet has been sent by TRANSMIT.confirm from iTransmitNLGW. In case of not receiving the correspondent TRANMSIT.confirm from the NM Server (Gateway) the NMNL will initiate a retry after a RETRY TIMEOUT.

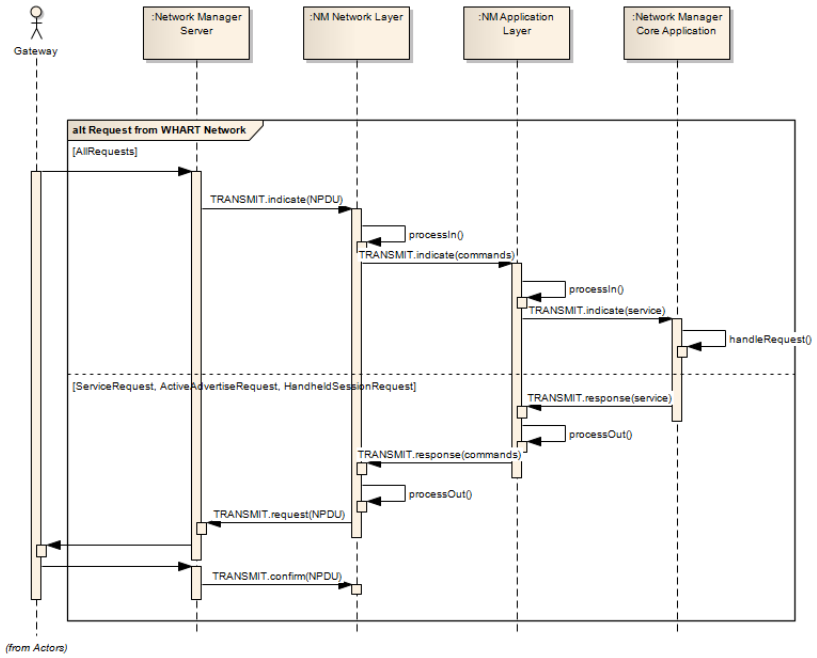

Figure 5.9: *Information Flow when a request is received from the Network*

When the Network Manager needs to send HART commands requests to the WHART network (Fig. 5.10), the NMCC will generate the TRANSMIT.request(s) from iTransmitAPAL indicating the service and the data to the NMAL. The NMAL will format (and aggregate if possible) the HART commands and will forward them (TRANSMIT.request from iTransmitALNL) to the NMNL which will encrypt them and construct the NPDU. After this, the NPDU will tell the NM Server to send the packet to the Gateway by TRANSMIT.request from iTransmitNLGW. The NM Server will encrypt the data and send it to the Gateway's NM client. After the packet being sent into the WHART network the Gateway will send a confirmation to the NM server which will indicate the NMNL that the packet has been sent by TRANSMIT.confirm from iTransmitNLGW. In case of not receiving the correspondent TRANMSIT.confirm from the NM Server (Gateway) the NMNL will initiate a retry after a RETRY TIMEOUT. When the correspondent response NPDU comes from the Gateway, the NM Server will indicate it to the NMNL by TRANSMIT.indicate from iTransmitGWNL. The NMNL will decrypt the packet and will check in the Transport table if the packet is the response of a previous request. In the case of a response packet, the NMNL will forward the aggregated response commands to the NMAL using TRANSMIT.confirm from iTransmitNLAL. The NMAL will extract the response data from the commands and will forward it to the NMCC by a TRANSMIT.confirm from iTransmitAPAL. The NMCC will process this data and the transaction is finished.

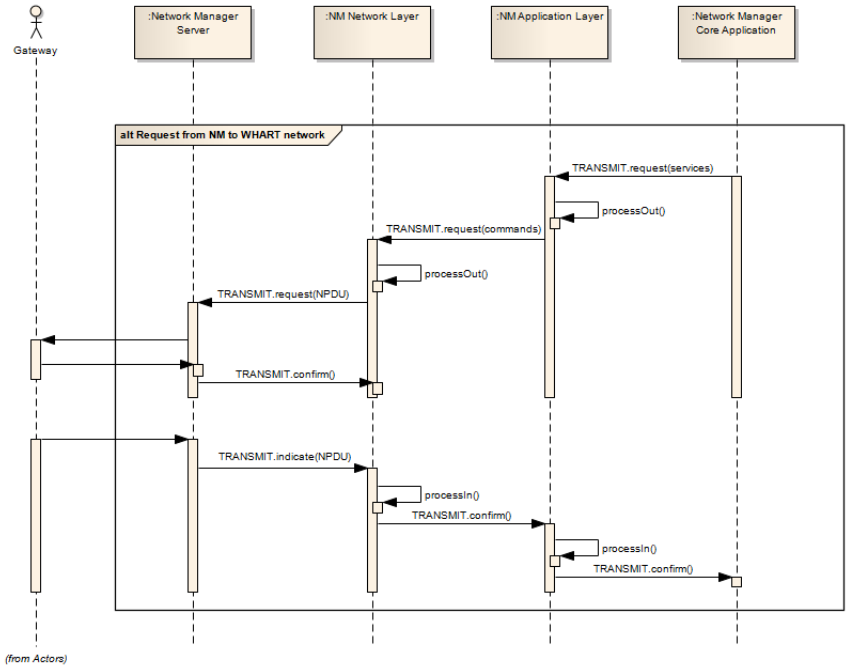

Figure 5.10: *Information Flow when a request is generated at the Network Manager.*

It is important to notice that the described transactions are not sequential, i.e. different service primitives can be called by the components at any time from different transactions. Thus buffering it is of great importance for handling this service primitives requests.

5.5.2 Description for Network Manager Core Component

This component is the core component of the *WirelessHART Network Manager*, as stated earlier. It represents the Sink and the Source of any request to/from the WHART network, Security Manager and Gateway. In this section we will describe the main classes which compose this software component.

5.5.2.1 Description for NetworkManagerProcessUnit

This class is central within the Network Manager Core Component. It is responsible for handling requests from the WHART network such as initializing the WHART network, joining, allocation of services, health reports and alarms. It is important to mention that there can be only one instance of the *NetwrorkManagerProcessUnit*, i.e. we use a singleton design pattern.

We will proceed describing the main attributes and methods that define this class.

Attributes Only the main attributes will be summarized for a better understanding:

- **NetworkID : Unsigned-16** Determines the ID of the network controlled by the *WHART Network Manager*. It Has to be the same than the *Gateway* and *Field Devices*.
- **JoinKey : Unsigned-128** Key provided by the Security Manager that devices use for join sessions. The *Network Manager* uses it for configuring the Network Layer component, which is responsible for handling secure sessions. This Join Key must be the same as the one used in the WHART network.
- **NetworkKey : Unsigned-128** Key provided by the Security Manager that devices use for the encryption of the DLPDU. This is used for configuring the *Gateway* and all the WHART *network devices*.

Methods The public methods will be describe briefly:

InitializeWHART
(networkID : Unsigned-16, GWuniqueID : Unsigned-16, APuniqueID : Unsigned-16) : boolean This function is responsible for the initialization of the *WirelessHART* network, which includes configuring the Gateway and Access Points with the initial Superframes and Links and the network graph (refer to 4.1.1, 4.1.2, 4.1.3 and 4.3.2) along with the Network and Session Key and nicknames. Figure 5.11 shows the sequence diagram describing the interaction between the different classes in the initialization of the WHART network process. More information about the sequential procedure can be found in section 5.3.2.1

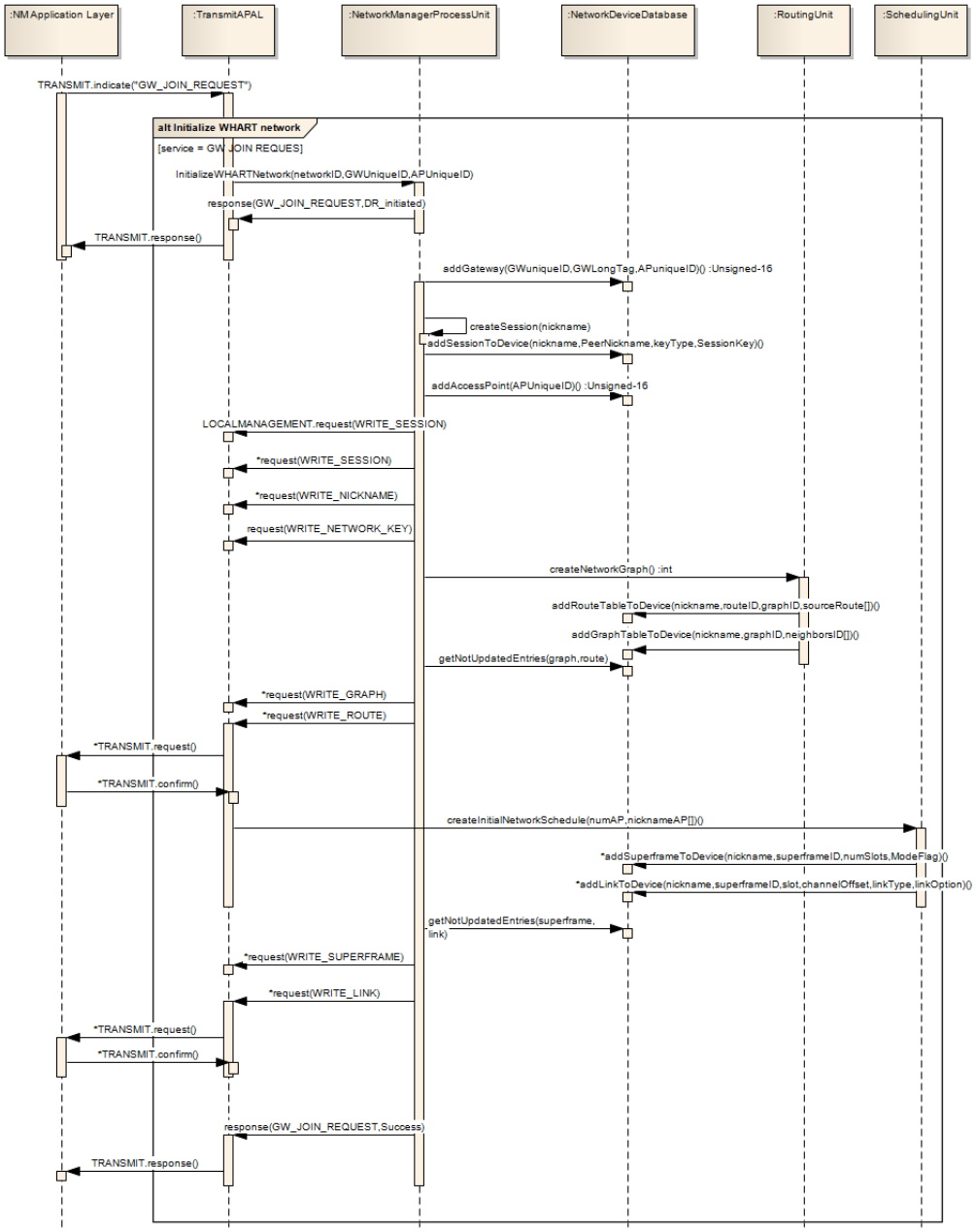

Figure 5.11: *Sequence diagram for initialization of the WHART network: initializeWHART.*

handleJoin (UniqueID:Unsigned-16, LongTag:Latin-1, NeighborData:byte[]):boolean Figure 5.13 shows the sequence diagram describing the interactions between the different classes in the joining process. This process is perform in a similar way to the process of initialization. The joining device presents the credentials consisting of UniqueID,

which indicates the 5-bye address of the device, LongTag, which describes the device in words, and finally the Neighbor information necessary to perform routing and scheduling. If the device is trusted, the NMPU responds to the request by configuring it, i.e. allocating session keys, nickname and managing the route and schedule. More information about the sequential procedure can be found in section 5.3.2.2

handleHealthReport (UniqueID:Unsigned-16, RSL[]:Signed-8, healthData[]:byte, neighborHealthData[]:byte) :boolean Figure 5.12 shows the sequence diagram for the case of storing health reports from the *WirelessHART* network. More information about the sequential procedure can be found in section 5.3.2.4

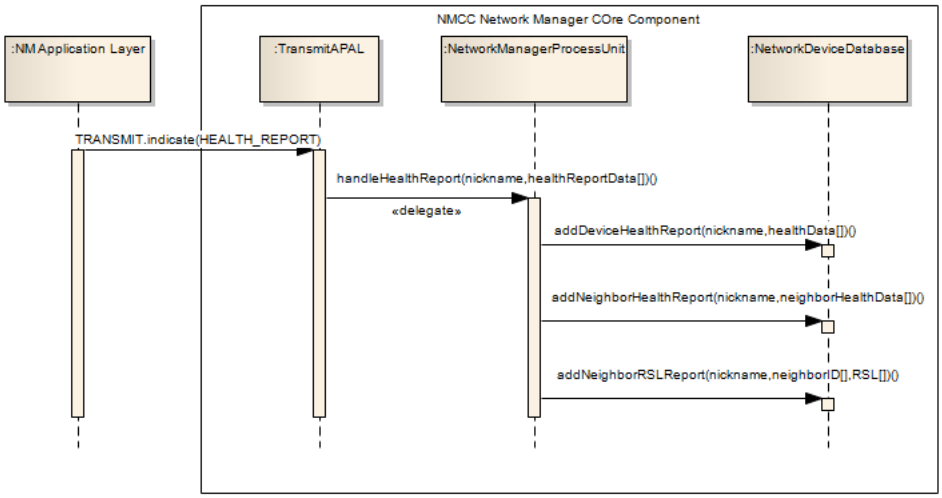

Figure 5.12: *Sequence diagram for Health report storage.*

boolean handleAlarm (Unsigned-16 UniqueID, alarmType:Enum-8) :boolean The NMPU stores information of alarm data in the database in order to maintain information about the health of the network. More information about the sequential procedure can be found in section 5.3.2.5

handleServiceRequest ((nickname:Unsigned-16, serviceID:Unsigned-8, serviceDomain:Enum-8, period:Time) : boolean The NPDU processes the service request by allocating bandwidth, i.e. links in superframes in order to make possible the communication. More information about the sequential procedure can be found in section 5.3.2.3

handleUpdate():boolean The NMPU checks the health information stored in the database and, later, if necessary, updates routes and schedules of the *WirelessHART* network. More information about the sequential procedure can be found in section 5.3.2.6

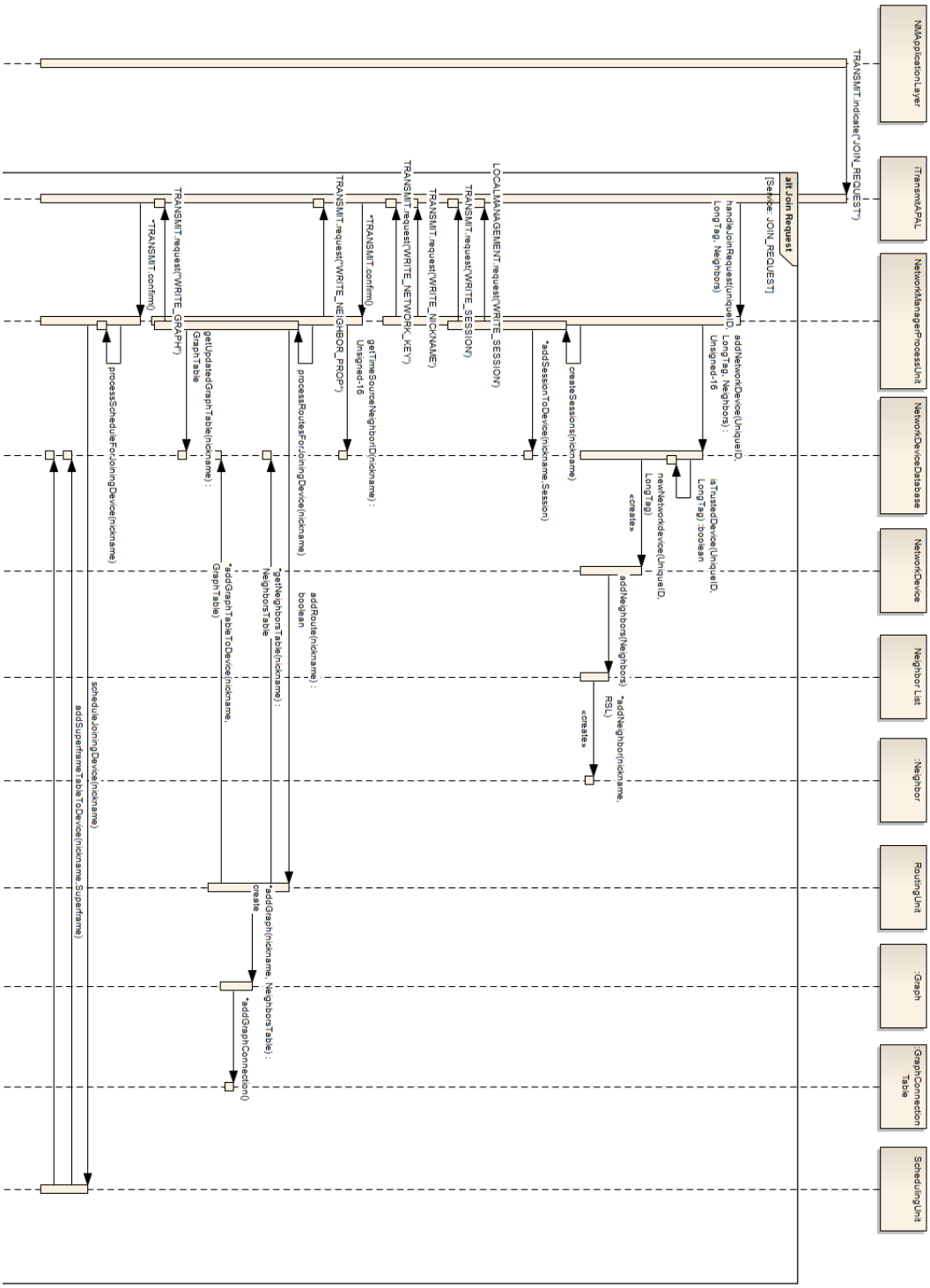

Figure 5.13: *Sequence Diagram for the Join process: handleJoin.*

5.5.2.2 Description for Routing Unit

Figure 5.14 shows a simple view of the internal structure for this subcomponent.

Routing Unit Interface to NMPU. iRoute

setRoutingAlgorithm(algorithm:Enum-8):boolean It will be used by the NetworkManagerProcessUnit for configuring the Routing algorithm that is going to be used.

numAP:int, nicknameAP:Unsigned-16 []):boolean Used in initialization of the WHART Network for creating the initial graph or Network Graph that defines an upstream graph in which the Gateway is the root.*numAP* indicates the number of access points and *nicknameAP* the array of nicknames for the access points. It will return true when correctly created.

routeDevice(nickname:Unsigned-16):boolean The Routing Unit will be responsible for creating a downstream Graph and add the device into the upstream Graph (Network Graph). Source Routes can be extracted directly from particular paths of the graphs. It will fetch information about the Neighbors and it will also save all the routing tables (Route, Graph and source route tables) for each device affected to the Network Device Database (refer to 5.5.2.4). It returns true when correctly routed.

unRouteDevice(nickname:Unsigned-16):boolean Used when deleting a device from the WHART network. It will delete all the tables of all devices affected from the Network Device Database.(refer to 5.5.2.4). It returns true when correctly routed.

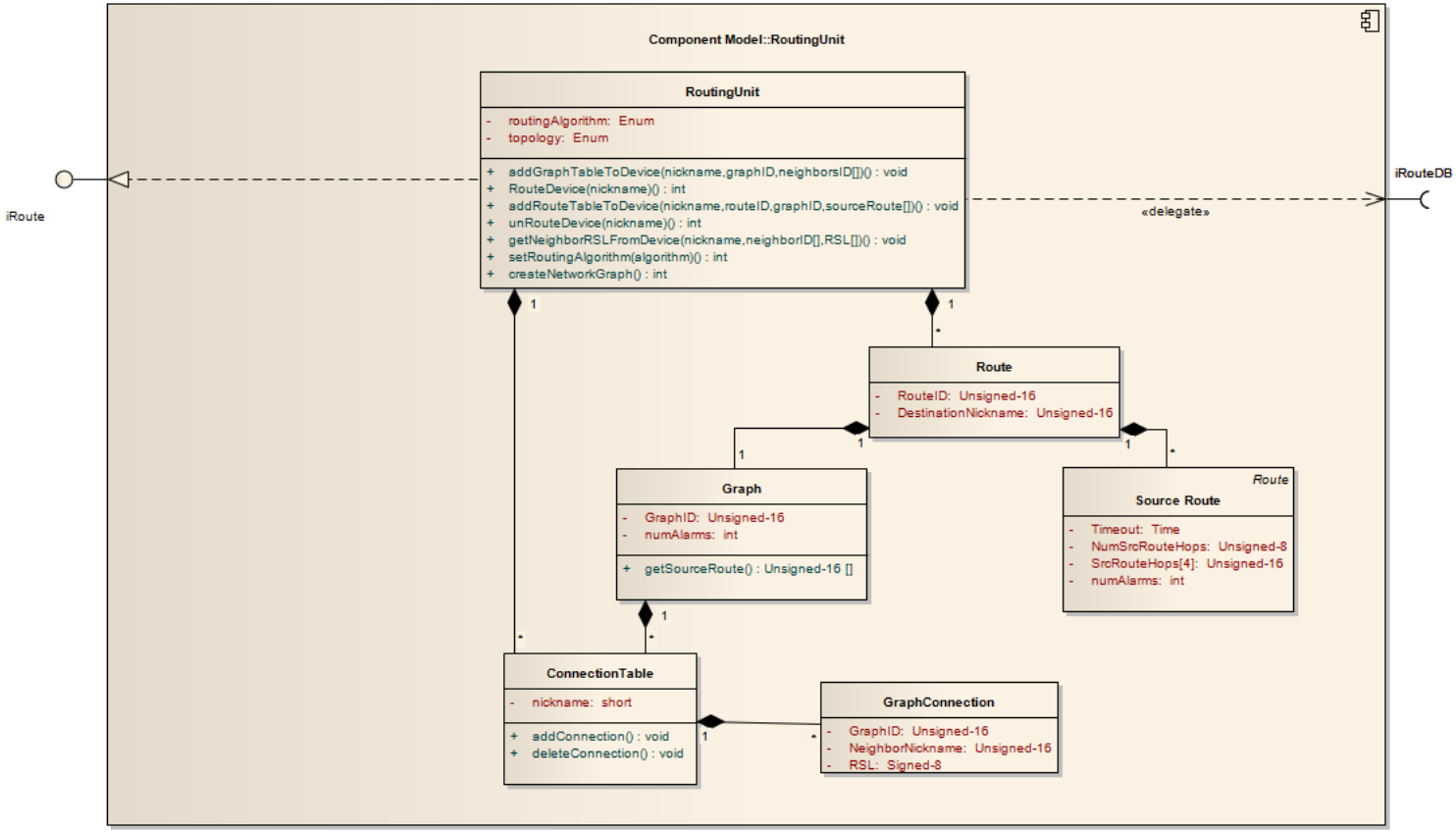

Figure 5.14: *Internal architecture for the Routing Unit subcomponent.*

5.5.2.3 Description for Scheduling Unit

Figure 5.15 shows a simple view of the internal structure for this subcomponent.

Scheduling Unit interface to NMPU. iSchedule

setWhiteListChannels
(whiteList:Unsigned-16) :void Function for setting the channels available in the Network Schedule. Every bit of *whiteList* represents the correspondent channel.

createInitialNetworkSchedule
(numAP:int, nicknameAP[]:Unsigned-16) :boolean The present function will create the initial superframes: management and Gateway superframes as well as the initial links. The links will support joining, advertisement, request/response traffic, keep alive s and health reports. *numAP* indicates the number of access points and *nicknameAP* the array of nicknames for the access points. This function will fetch information about routing from NDDB and store the Scheduling tables correspondent to the Virtual Gateway and Access Points in the NDDB (refer to 5.5.2.4). It returns true when the schedule is created correctly.

scheduleDevice
(nickname:Unsigned-16) :boolean Function used by the NetworkManagerProcessUnit during the Joining Process of a network device. It will basically allocate superframes and links to the device regarding joining, advertisement, request/response traffic, keep alive s and health reports. *nickname* represents the short address of the joining device. This function will fetch information about routing from NDDBstore the Scheduling tables (Superframe and Link table) correspondent to Network Devices affected in the communication in the NDDB (refer to 5.5.2.4).It returns true when the schedule is created correctly.

unScheduleDevice
(nickname:Unsigned-16) :boolean Function used by the NetworkManagerProcessUnit when disconnecting a network device from the WHART Network. *nickname* represents the short address of the disconnecting device. It returns true when correctly deallocated from the schedule.

scheduleServiceToDevice
(nickname:Unsigned-16, serviceID:Unsigned-8, serviceDomain:Enum-8, period:Time) : boolean Function used by the NetworkManagerProcessUnit when allocating services to a Network Device. It returns true when correctly allocated.

unScheduleService
(nickname:Unsigned-16, serviceID:Unsigned-8) :boolean Function used by the NetworkManagerProcessUnit when deallocating services to a Network Device. It returns true when correctly deallocated.

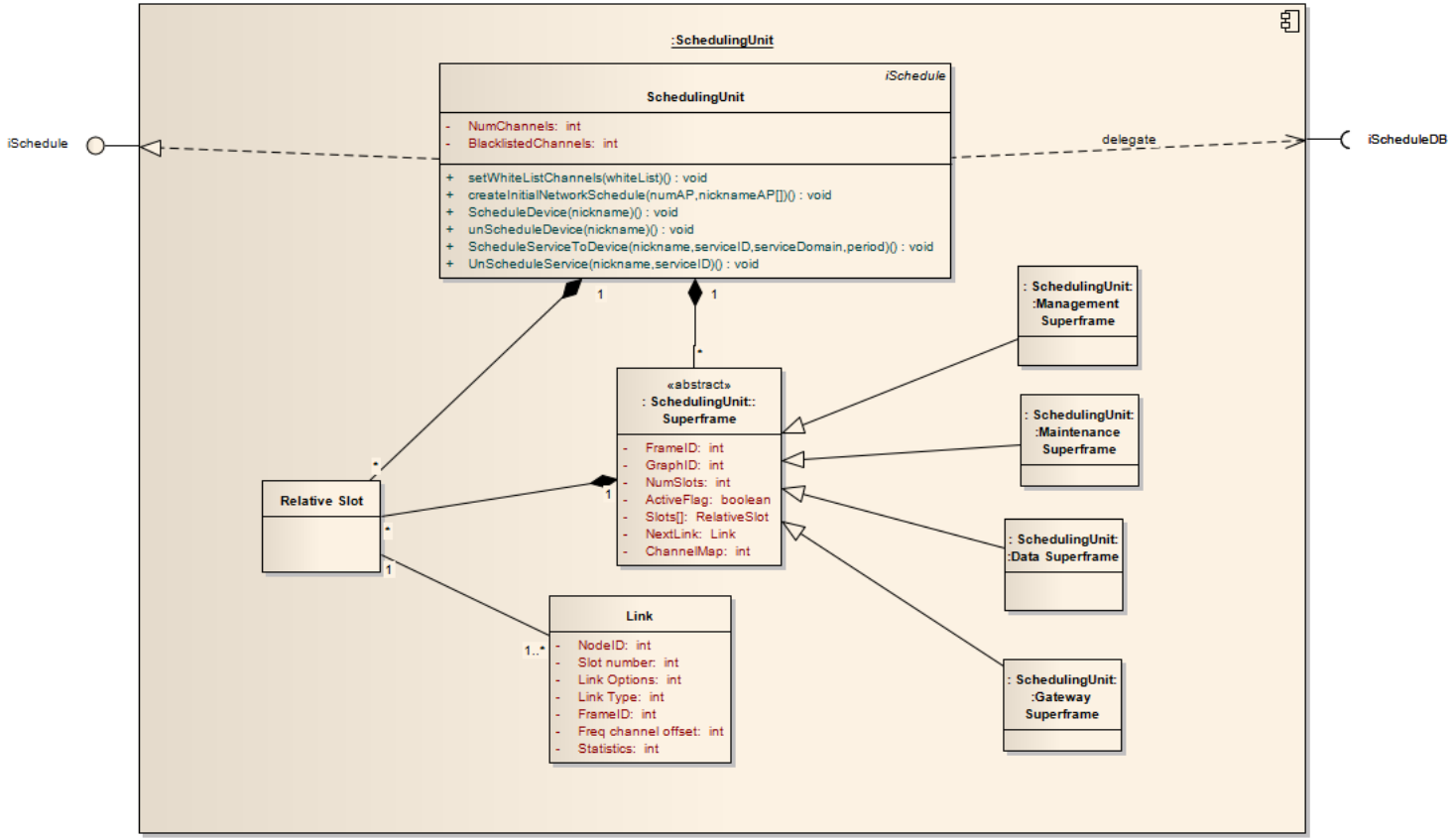

Figure 5.15: *Internal architecture for the Scheduling Unit subcomponent.*

5.5.2.4 Description for Network Device Database

Figure 5.16 represents the internal design of the Network Device Database.

Network Device Database Interface to NMPU. iDB

addGateway (GWuniqueID:Unsigned-16,GWLongTag:Latin-1) :Unsigned-16 It registers the Gateway in the database.

addAccessPoint (APUniqueID:Unsigned-40,GWUniqueID:Unsigned-40) :Unsigned-16 It returns the nickname of the Access Point.

addNetworkDevice (uniqueID:Unsigned-16,LongTag:Latin-1) :Unsigned-16 It adds the Network Device to the WHART Network Device Database. If the process has been successful it will return the nickname of the Network Device added to the Network.

addNeighborRSLReport (nickname:Unsigned-16,neighborID[]:Unsigned-16,RSL[]) :Unsigned-16 It stores the information regarding the neighbors of the device.

addSessionToDevice (nickname:Unsigned-16,PeerNickname:Unsigned-16,keyType:Enum-8,SessionKey:Unsigned-128) :Unsigned-16 It adds session to the device and it is stored in the database.

addDeviceHealthReport (nickname:Unsigned-16,healthData[]:byte) :Unsigned-16 It adds the Health information of device to the database.

addNeighborHealthReport (nickname:Unsigned-16,neighborHealthData[]:byte) :Unsigned-16 It adds the Neighbor's Health information of the device to the database.

getTimeSourceNeighborID (nickname:Unsigned-16):Unsigned-16 It returns the neighbor ID of the time source for the specific network device

getNotUpdatedEntries (type:Enum-8,nickname[]:Unsigned-16,data[][]:byte) :TableEntry[] It returns the Table Entries that has not been updated in the WirelessHART Network.

Network Device Database Interface to Routing Unit. iRouteDB

addRouteTableToDevice
(nickname:Unsigned-16, routeID:Unsigned-8, graphID-:Unsigned-16, sourceRoute[]:Unsigned-16) :boolean Adds a Route table to a Network Device in the database. This info will be fetch by the NetworkManagerProcessUnit from the Network Device Database for configuring the correspondent network device through HART commands. It returns true when correctly added.

addGraphTableToDevice
(nickname:Unsigned-16, graphID:Unsigned-16, neighborsID[]:Unsigned-16) :boolean
Adds a Graph table to a Network Device in the database. This info will be fetch by the NetworkManagerProcessUnit from the Network Device Database for configuring the correspondent network device through HART commands.

getNeighborRSLFromDevice
(nickname:Unsigned-16, neighborID[]:Unsigned-16, RSL[]:Signed-8) :Void Gets Neighbor Received Signal Levels from a determined Network Device. This information will be used in deciding the graph paths.

Network Device Database Interface to Scheduling Unit. iScheduleDB

addSuperframeToDevice
(nickname:Unsigned-16, superframeID:Unsigned-8, numSlots:Unsigned-16, ModeFlag:Enum-8) :boolean Adds a Superframe to a device in the database. If the device does not exist it returns -1.

addLinkToDevice
(nickname:Unsigned-16, superframeID:Unsigned-8, slot, channelOffset, linkType, linkOption) :boolean Adds a link to a device in the database. If the device or the superframe does not exist it returns -1.

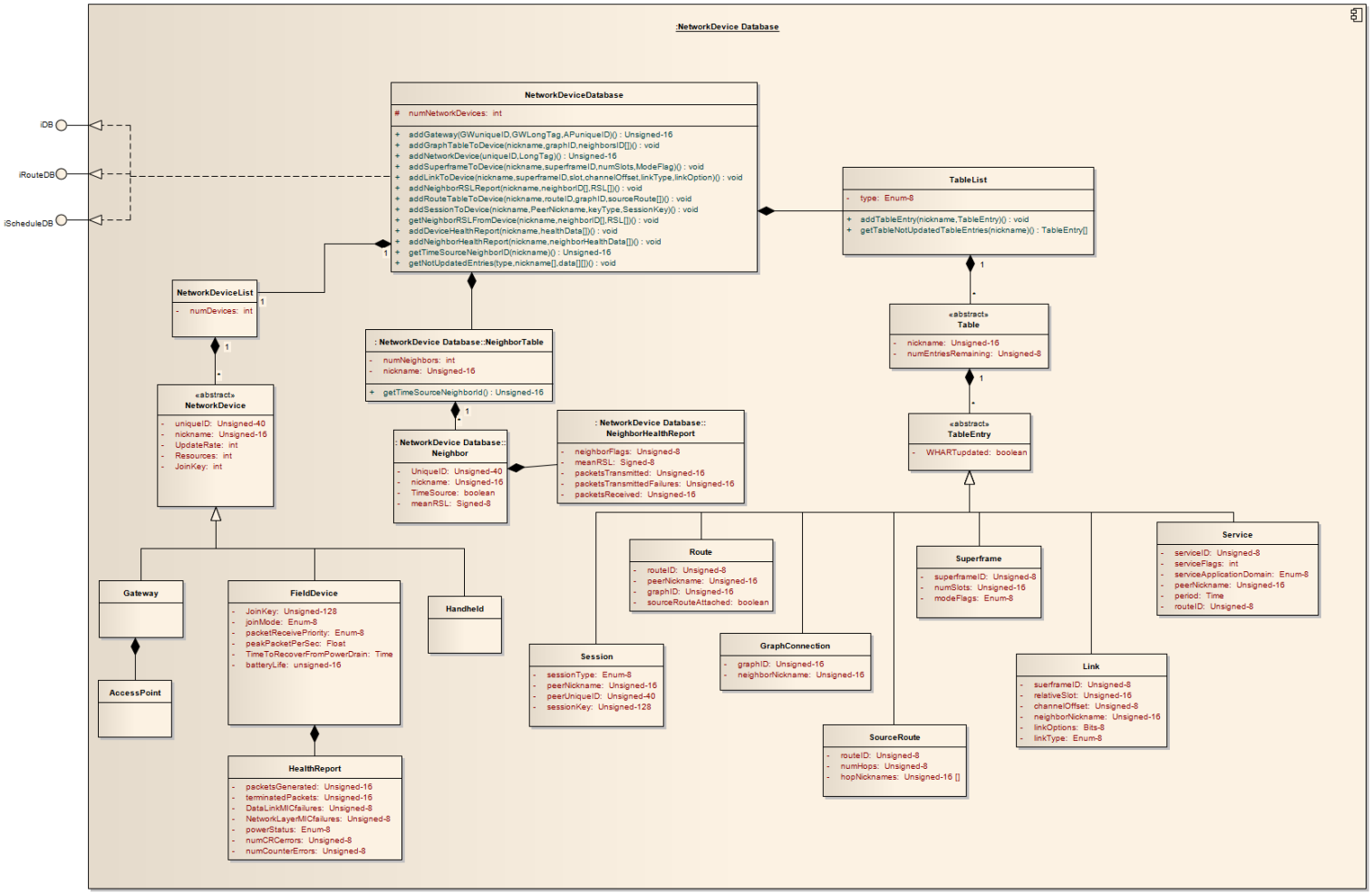

Figure 5.16: *Internal architecture for the Network Device Database subcomponent.*

Chapter 6

Conclusions and Future work

6.1 Conclusions

The introduction of the wireless technology in the industrial realm revealed an imperious need of a standardized wireless communication protocol, capable of guaranteeing the fulfillment of the strict industrial requirements. In this sense, the HART Communication Foundation(HCF) created the first open standard addressed to the stated matter: *WirelessHART*.

Due to the novelty of the *WirelessHART*, plenty of research is in progress in order to analyse the real capabilities of *WirelessHART*. In this manner, we have performed an exhaustive scrutiny of the most decisive component of *WirelessHART*: The Network Manager.

The Network Manager plays a key role as the central point of management within the *WirelessHART* network. It is responsible for configuring the network, scheduling communications, managing route tables and monitoring the health of the network. However, a lack of a standardized software design and architecture of the Network Manager, together with the absence of a clear and detailed description of the Network Manager can discourage the establishment of the *WirelessHART* standard.

On that account, we have addressed these key concerns with the present thesis. We have performed (1) an exhaustive analysis of the *WirelessHART* specification, and later, we have focused in every matter regarding the operation and functionalities of the Network Manager, isolating and presenting (2) a compact specification of the software requirements of the Network Manager. In addition, we have proposed (3) solutions to some question marks unclear in the standard. Finally, we have provided a (4) software design and architecture of the Network Manager, describing its internal subcomponents and functionalities.

In conclusion, the process of implantation of wireless technology in industry communications, together with the need for secure, reliable and interoperable standards, challenges the academic and industrial research to achieve a collective and optimal solution, which implies reducing costs and saving energy. With the present work, we have provided a module for *WirelessHART* which contributes in establishing *WirelessHART* as the wireless solution in the industrial process domain.

6.2 Future work

This thesis can be regarded as a big step towards the implementation of the *WirelessHART Network Manager*. We have provided a modular and extensible design that can be used in future research. Future work can be focused on particular functionalities of the *WirelessHART Network Manager*.

The most critical responsibilities of the *WirelessHART Network Manager* are routing and scheduling, which determine the performance of the overall WHART network in terms of throughput and reliability. Future efforts will be focused on determining the most appropriate routing and scheduling algorithms for specific applications in Automation Industry. Scheduling in *WirelessHART* is, to the best of our knowledge, the most challenging and still unsolved area of research.

Building prototypes of the Network Manager will also aid to adapt the complex task of management to the concrete necessities of the distinct industrial applications.

Bibliography

[1] Bluetooth. http://www.bluetooth.com.

[2] Hart communication foundation. http://www.hartcomm.org.

[3] Isa100: Wireless systems for automation. http://www.zigbee.org.

[4] Unified modeling group. http://www.uml.org.

[5] Zigbee alliance. http://www.zigbee.org.

[6] J. Persson C. Gehrmann and B. Smeets. Bluetooth security. In *Artech House Boston, London, 2004, ISBN: 1-58053-504-6.*

[7] John Cheesman and John Daniels. *UML Distilled: A Brief Guide to the Standard Object Modeling Language*. Addison-Wesley Professional; 2nd Edition, 1999.

[8] John Cheesman and John Daniels. *UML Components: A Simple Process for Specifying Component-Based Software*. Addison-Wesley Professional, 2000.

[9] Alistair Cockburn. *Writing Effective Use Cases*. Addison-Wesley Professional, 2000.

[10] Katy Sierra Elisabeth Freeman, Eric Freeman and Bert Bates. *Head First Design Patterns*. O'REILLY Media, 2004.

[11] David Gustafsson. Wirelesshart - implementation and evaluation on wireless sensors. Master's thesis, Royal Institute of Technology, 2009.

[12] HART Communication Foundation. *Block Data Transfer Specification*, April 2001.

[13] HART Communication Foundation. *The HART Protocol - A Solution Enabling Technology*, February 2004.

[14] HART Communication Foundation. *Command Summary Specification*, July 2007.

[15] HART Communication Foundation. *TDMA Data Link Layer Specification*, May 2008.

[16] HART Communication Foundation. *Universal Command Specification*, May 2008.

[17] HART Communication Foundation. *Wireless Command Specification*, May 2008.

[18] HART Communication Foundation. *WirelessHART Device Specification*, May 2008.

[19] HART Communication Foundation. *WirelessHART Device Specification, p87*, May 2008.

[20] HART Communication Foundation. *Common Tables Specification*, March 2009.

[21] HART Communication Foundation. *Network Management Specification*, March 2009.

[22] IEEE Computer Society. *Wireless Medium Access Control (MAC) and Physical Layer (PHY) Specifications for Low-Rate Wireless Personal Area Networks (WPANs)*, September 2006.

[23] Anna N. Kim, Fredrik Hekland, Stig Petersen, and Paula Doyle. When hart goes wireless: Understanding and implementing the wirelesshart standard. In *ETFA*, pages 899–907, 2008.

[24] Igor Konovalov. A framework for wirelesshart simulations. Master's thesis, Royal Institute of Technology, 2010.

[25] N.Baker. Zigbee and bluetooth strengths and weaknesses for industrial applications. In *Computing and Control Engineering Journal, vol 16(2): 20-25*.

[26] Shahid Raza, Adriaan Slabbert, Thiemo Voigt, and Krister landernås. Security considerations for the wirelesshart protocol. In *Proceedings of the 14th IEEE international conference on Emerging technologies and factory automation*, 2009.

[27] Jianping Song, Song Han, Al Mok, Deji Chen, Mike Lucas, Mark Nixon, , and Wally Pratt. Wirelesshart: Applying wireless technology in real-time industrial process control. *Real-Time and Embedded Technology and Applications Symposium, IEEE*, page 386, 2008.

[28] Stefan Svensson Tomas Lennvall and Fredrik Hekland. A comparison of wirelesshart and zigbee for industrial applications. 2009.

Appendix A

HART Commands

A.1 Command 977. Gateway Join Request

This HART command is used by Gateways for Requesting the Network Manager initialization of the WHART network. It

Request Data Bytes

Byte	Format	Description
0-4	Unsigned-40	UniqueID of the Gateway
4-35	Latin-1	Long Tag of the Gateway
36	Unsigned-8	Number of Access Points
37-41	Unsigned-40	UniqueID of Access Point 1
...		UniqueID of AccessPoint n

Response Data Bytes

Byte	Format	Description
0-4	Unsigned-40	UniqueID of the Gateway
4-35	Latin-1	Long Tag of the Gateway
36	Unsigned-8	Number of Access Points
37-41	Unsigned-40	UniqueID of Access Point 1
...		UniqueID of AccessPoint n

Command-Specific Response Codes

Byte	Format	Description
0	Success	No Command-specific Errors
1-4		Undefined
5	Error	Too Few Data Bytes Received
37-41	Unsigned-40	UniqueID of Access Point 1
...		UniqueID of AccessPoint n

Appendix B

Data types

This appendix will clarify wich data types we use in the design of the *WirelessHART Network Manager*. They refer to data types included in the standard. It's is up to the implementor to decide wether to use a dedcated class for each data type. For interested readers, more information about data types on WHART specification can be found in [14] sect. 5.

- **Bits** Byte in which each bit has a specific meaning. bit 0 is the least significant
- **Date** The date consists in 8-bit binary unsigned integers that represents the day, month and year minus 1900, respectively. Day is first.
- **Enum-n** An integer enumeration in which each number has a specific meaning. n represents the Only the values specified in [20] will be valid.
- **Latin-1** It is a 32 bytes string composed by 8-bit ISO Latin-1 characters. Lating-1 is padded with zeroes (0x00)
- **Signed-n** a signed integer of size n bits.
- **Time** An unsigned 32-bit binary integer representing the time in 1/32 of a millisecond, i.e. the least significant bit is 0.03125 ms.
- **Unsigned-n** An usingned integer of size nbits.