Application of wacnet to ambient intelligent systems

Antoine Desmet
University of Wollongong

APPLICATION OF WACNET TO AMBIENT INTELLIGENT SYSTEMS

A thesis submitted in fulfillment of the
requirement for the award of the degree

MASTERS OF ENGINEERING - RESEARCH

from

UNIVERSITY OF WOLLONGONG

By

ANTOINE DESMET

BACHELOR OF ELECTRONICS

SCHOOL OF ELECTRICAL, COMPUTER AND
TELECOMMUNICATION ENGINEERING

2008

Certification

I, Antoine DESMET, declare that this thesis, submitted in fulfillment of the requirement for the award of Masters of Engineering by Research, in the School of Electrical, Computer and Telecommunication Engineering, university of Wollongong, is wholly my own work unless otherwise referenced or acknowledged. The document has not been submitted for qualification at any other academic institution.

.

.....

Antoine DESMET

October 2008

Contents

List of Figures

List of Tables

List of Abbreviations

HTTP: HyperText Transfer protocol

CH: Cluster Head

CN: Cluster Node

Tx: Transmission

Rx: Reception

OSI: Open System Interconnection

WACnet: Wireless Ad-hoc Control NETwork

ZB: ZigBee

BT: Bluetooth

DCS: Distributed/Decentralized Control System

WiFi: Wireless Fidelity, IEEE 802.11x wireless standards

PnP: Plug and Play

HMI: Human-Machine Interface

I/O: Input/Output

TCP/IP: Transfer Control Protocol / Internet Protocol

CSMA/CD: Carrier Sense Multiple Access / Collision Detection

XML: eXtended Markup Language

JVM: java virtual machine

PCB: Printed Circuit Board

NCAP: Network capable node

XDCR Transducer

FTP: File Transfer protocol

HEX file : hexadecimal file uploaded to the microcontroller

LCD: Liquid Crystal Display

CPU: Central processing Unit

Abstract

Control Systems have gone through several stages of development over the past decades. While Fieldbus is now the established global standard for factories and plants, new areas of development are opening, requiring radically different control network architectures.

There is currently a strong demand for wireless control networks capable of meeting the application-specific requirements of military, agricultural and biological, building control, land surveying, monitoring and control networks. These new applications have characteristics which are much different from factory applications, including high autonomy and low maintenance, flexibility and adaptation in dynamic environments, extremely small size, rapid and sometimes random deployment, automatic handling of the failed nodes, etc.

With the advent of widespread, standardized and reliable wireless standards, a new generation of wireless control networks appears feasible. The concept of Wireless Ad-hoc Control Networks (WACNets) is an ongoing research project which began in 2004, at the University of Wollongong. A WACNet consists of a large number of geographically distributed intelligent and heterogeneous nodes with sensing and/or actuation, local intelligence and control, data processing and wireless communication components.

This research project pursues the technological development of the WACNet architecture and hardware, using the state of the art technologies.

The other main contribution of this work is the validation of the WACNet platform for real-life applications. A Home Ambient Intelligent system for resource consumption reduction is designed to run on a WACNet architecture.

The capacity of a WACNet to support a learning algorithm and the resulting network load is studied.

The results of the validation process demonstrate the potential of WACNets to support real-life applications, and also highlight the technical challenges which will have to be tackled before the stage of a commercially-viable product can be reached.

Acknowledgements

First of all, I would like to thank my two supervisors: Fazel Naghdy and Montserrat Ros, for their advice, guidance and support.

I would also like to thank the staff from the School of Electronics, Computer and Telecommunication Engineering, at the University of Wollongong, especially all the workshop technicians, the SECTE receptionists, Brian Biehl and Sacha Nicolic.

I wish to express my gratitude to Bezhad Fatahi who introduced me to the masters of engineering by research degree, and gave me guidance and motivation to apply for this degree.

Finally I would like to dedicate this work to three persons:

- My parents, without whom nothing would have been possible, and who also provided me with their advice and unconditional support throughout the entire duration of studies.
- My partner Clare, who everyday wholeheartedly supported and helped me throughout my defeats and successes.

Chapter 1 Introduction

1.1 Introduction

This thesis represents a study on the development of an ad-hoc distributed control concept, capable of supporting applications such as a Home Ambient Intelligent system. Such a system represents the next stage in the evolution of Building Automation Systems (BAS). The distributed architecture is called Wireless Ad-hoc Control Networks.

Intelligent algorithms are deployed in Home Ambient Intelligent systems to control various aspects of a buildings and achieve various objectives including optimizing the level of comfort for the user, enhancing security, and saving resources.

In this study, a hardware platform and an effective design for a Home Ambient Intelligent system towards optimizing resource consumption is designed and developed.

1.2 Context and Overview

Building Automation and Control systems have relentlessly evolved due to continuous market demand. Building control systems aim at optimizing comfort and security within a building while reducing maintenance cost.

In the past, the control of lights, heating, and ventilation was primarily achieved through simple mechanical switches, usually located in the same room as the appliance. Electronic sensing and control devices first appeared in plants and factories. They soon became a fundamental element of most factories, resulting in the

emergence of a new market and research area. This encouraged the development of automation systems, which became progressively more sophisticated as they continued to reduce the installation and maintenance cost. Eventually, low-cost electronic control systems were developed for domestic and commercial buildings. At

present, every modern house is equipped with at least one of these systems, either built in or retrofitted. The most common examples of current Building Automation Systems (BAS) are dedicated to the sensing and control of temperature, ventilation, detection of presence and sometimes to the control of access or the verification of identity. Access cards replaced mechanical keys. Electronic alarm systems and Closed-Circuit Television (CCTV) assisted night watchmen. In larger buildings, centralized control consoles appeared, allowing remote control and surveillance of an entire building from a single room.

1.2.1 First Automated Systems

Continuing beyond remote control, automation systems also have begun to automate most of the tasks which were previously carried out by employees or users. The control functions have become increasingly complex, integrating finer models for output calculations. The term “smart (or intelligent) buildings” [45,46,47] appeared around the year 1980, as the control devices could predict the preference of the users. In reality, however, those controllers did not have any form of intelligence. They were the product of an exhaustive study of the subject, resulting in all-encompassing and specific control functions. These systems eventually became obsolete, as they could neither adapt nor learn. Indeed, these hard-coded systems were incapable of extending a basic configuration or learn in order to adapt to unknown situations.

Many definitions exist to describe an intelligent building system, as this concept can be analyzed from various points of view. The two main visions are based on either the observation of the behavior, or by an analysis of the core of the system. This results in totally different definitions.

The building industry defines an intelligent building as follows: “An Intelligent Building is one that provides a cost-effective environment through the optimization of four basic elements: systems, structures, services, management and the inter-relationship between them” [48]. This definition focuses exclusively on the behavior of the system.

A definition from a computer-science point of view is provided by Sharples et al. [49] “An Intelligent Building is one that utilizes computer technology to

autonomously govern the building environment so as to optimize user comfort, energy-consumption, safety and monitoring functions". This definition encompasses not only the behavior of the system but also introduces a method of reasoning and decision-making by computers, similar to human intelligence.

1.2.2 Home Ambient Intelligent Systems

Home Ambient Intelligent systems represent the next stage of evolution for Building Automation Systems. This concept aims at integrating modern artificial intelligence algorithms and agent-based technologies with the electronic sensing devices prevalent in modern BAS.

Compared with programmed building control, Ambient Intelligent systems offer many advantages, such as flexibility and complex problem solving abilities. It has been proven that human behavior is highly unpredictable and constantly influenced by subtle factors [46]. Artificial intelligece systems are designed to deal with levels of complexity which surpass the capacity of human minds. Indeed, they are often used to solve problems for which a precise model does not exist. They are used when the phenomenon is difficult to define, or requires a unique solution or sometimes because the response must vary with time. Unlike traditional control loops which are purpose built, artificial intelligence systems take a very global and methodical approach towards a solution. Artificial intelligence systems consider methodically every variable of a problem and evaluate all possible correlations. The result is a generic system which self-configures to provide a specific solution to any problem. A Home Ambient Intelligent system offers the potential to learn its task by observing the behavior of the inhabitants. This approach is far more realistic than attempting to state a set of rules which dictates the expected response of the system.

Over the past ten years, many attempts have been made to create Ambient Intelligent systems capable of anticipating the behavioral patterns of the monitored users [46]. These projects are aimed at automatically tuning the environment to the user's expectations. As an example, the brightness of a light would be automatically adjusted to the optimum setting, in relation to the user's current activity (reading, eating, watching television, etc.). Most of these attempts have shown satisfactory

results, but their success has been limited due to their inability to sense character-specific parameters such as mood. The infrastructure used for the implementation of concepts has also made the solution impractical and cost-ineffective as a large number of PC servers and expensive industrial control networks were required [45,46].

Around 2005, a new direction of development has emerged for Home Ambient Intelligent systems towards resource consumption reduction. The price of energy has been increasing rapidly and resources such as water are becoming a precious commodity in many parts of the world. This has spawned a strong demand for simple-to-use systems capable of lowering the energy and water consumption. As a sign of interest in this growing market, the leading manufacturing firms around the world have started to invest in the research and development of consumption reduction solutions. Such solutions include miniaturized and wireless gas, water, and electrical energy consumption meters, which are becoming available as off-the-shelf components.

Indeed, resource consumption monitoring and reduction is an ideal target application for Home Ambient Intelligent systems. Intelligent systems can automatically create a tailored consumption reduction program for each single building and user. As they are very generic, they can be used for any kind of resource requiring conservation. They can work with minimal configuration, by simply learning the habits of the users and the consumption characteristics of the appliances.

A Home Ambient Intelligent system for resource consumption reduction is a potential solution full of promises, though there are still many uncertainties which need to be addressed. Research must determine whether the current learning algorithms are capable of appropriately define the consumption patterns of a building. One of the major remaining challenges is the creation of affordable solutions which integrate seamlessly into the user's environment.

1.2.3 WACNets

A novel concept called Wireless Ad-hoc Control Network (WACNets) has been under development at the centre for Intelligent Mechatronics Research at the

University of Wollongong, for the purpose of providing a framework for highly distributed, intelligent wireless control networks [5,6]. The nodes in WACNets have sufficient resources to locally execute many tasks such as sensor/actuator control and data conditioning and processing while constantly communicating with each other. The overall network structure is ad-hoc for added flexibility. The WACNet platform has been developed to create a low-cost platform for virtually any control task. The last research project developed the specifications of the WACNet platform and created one network node. The project concluded theoretically with potential for application in large systems.

The WACNet architecture matches the specifications required for such an application. The use of WACNets can potentially solve the previous problems that Ambient Intelligent systems has encountered. In this research project, a Home Ambient Intelligent system is designed to run on a WACNet platform.

1.3 Aim and Objectives of the Thesis

Previous research projects involving WACNets [5,6] have demonstrated the potential of WACNets for real-life applications to some extent. In this thesis, the concept and structure of WACNet is further developed with a specific focus on an Ambient Intelligent system real-life application.

In pursuit of this aim, the study will achieve the following objectives:

- Provide a comprehensive and critical review of the technological evolutions and the outcomes of research projects which are relevant to the thesis topic.
- Pursue the technological development of the WACNet architecture and hardware, using the most current and most adapted technologies and hardware available.
- Benchmarking and simulating the upgraded platform and identifying its optimal characteristics.
- Design a Home Ambient Intelligent system for resource consumption reduction based on WACNet architecture.

- Design, testing and validating a learning algorithm, operating within the Ambient Intelligent system.
- Implement and validating elements of the designed system on a WACNet platform.
- Validate and analyzing the performance of the WACNet-based ambient intelligent system

1.4 Structure of thesis

The thesis will continue with a review of the previous literature on the work related to this project. It will report on the historical development of the concept from its origins to the state of the art. The major challenges faced by control networks will be also be highlighted.

In Chapter 3, the concept of WACNet will be introduced and an overview of the work carried out in this study will be given. The aims and objectives of the study, along with the methodology deployed, and a real-life example of the concept will also be provided.

Having introduced control networks and the WACNet, an in-depth technical description of the platform will be given in Chapter 4. This will include a detailed analysis of the basic hardware elements followed by introducing the software components of the WACNet, and general structure and organization of the proposed platform.

Chapter 5 describes the learning algorithm developed for the WACNet to fulfill the requirements of the application. The related concepts and the algorithm will be presented, along with some preliminary tests.

Chapter 6 reports of experimental work carried out to validate the findings of the research. The first experiment is the benchmark test of the upgraded WACNet platform. The second experiment is a large-scale network load simulation. The last experiment is a real-life test of the learning algorithm presented in Chapter 5, using five WACNet nodes. For each experiment, the methodology, aims, setup and results

will be presented. Each section will also be concluded by a thorough analysis and justification of the results obtained.

The final chapter will conclude this thesis. The major achievements and outcomes will be summarized. The feasibility of the WACNet concept for large-scale and real-life applications will be discussed. The conclusion will be followed by some suggestions for future development.

Chapter 2 Literature Review

2.1 Introduction

In this chapter, a review of the literature associated with the main theme of the study is carried out. The literature review aims at introducing the main subject of this thesis and provides the reader with a fundamental knowledge of the related topics.

The study begins with a definition of process control systems. This will be followed by a historical presentation of the evolution of control systems, from the early stages of process control to modern distributed embedded control systems currently used. The improvements and challenges addressed at each major step during this evolution will be addressed. The study will continue with reviewing the current industrial solutions for control networks. The communication standards will be presented first, followed by the current control network standards.

The chapter will then continue by reviewing the most relevant work in the literature to this study. In the final stage, the significant characteristics of the WACNet concept will be introduced.

2.2 Definition, Origins and Evolution of Industrial Control Networks

A control system ensures that a variable or a set of variables in a system conforms to a desired value. The control system either holds the values of the controlled quantities constant or causes them to vary in a prescribed way. A control system may be driven electrically, mechanically, hydraulically, pneumatically or a combination of them. It is defined by its input(s), the function(s) which processes input(s) into output(s), and finally the output(s)" [1].

Process control originates from the desire to automate processes. The motivation behind automation is to remove or reduce the need for human intervention or

supervision of a process. Automated process control has been an area of interest for many centuries. Indeed, an early example dates back to the Roman civilization, were an archaic mechanical control system was used to regulate the water supply [29]. The modern process control science flourished during the industrialization period. With the invention of the very first steam machines around the 18th century, the need for automation of large industrial processes led to an extensive development of this field.

During this period, advanced mechanical controllers emerged including the famous James Watt's Centrifugal Governor [12]. The scientists at the time were aware of the need for precise modeling of control systems through mathematical and physical descriptions. They consequently developed various criteria for stability and response time, in search of stable real-time systems.

The earlier closed-loop control mechanisms invented had many drawbacks. For example, they were custom designed for one particular system and usually processed a single input into a single output. Therefore all the control systems worked in isolation from each other. Their design required expert knowledge in various fields and required high levels of maintenance and tuning.

2.2.1 Development of Electrical and Electronic Systems

The science of process control underwent its first noticeable leap with the advent of electricity, and from then on developed at a fast pace. Electrical motors came into use in industrial plants by the late 19th century, and replaced most of the steam engines. Electrical machines could be finely controlled by switches or regulators, which could be wired remotely to a command console.

The latest major evolution was brought about around the 1950's with the development of modern electronics and semiconductors. With analogue electronics, it became possible to implement relatively advanced closed-circuit functions such as integral, proportional and differential control loops. These systems had the advantages of being small, efficient, low-maintenance and real-time. Vacuum tubes, followed by transistors offered the possibility of regulating a large supply of energy with a small control voltage.

The evolution of control systems facilitated the development of automated processes. These processes were eventually able to provide results well beyond the capacity of any human operator in terms of speed and accuracy. The complexity of the processes increased as the plants spread over wider areas, comprised of more and more machines, and ran inter-dependently at higher speed. The high level coordination and synchronization required by such systems could not be met with contemporary technology. This necessitated the development of a global, more advanced system capable of coordinating and synchronizing the plant.

2.2.2 Computer Controlled Systems

The introduction of micro-computers in the 1970's provided a flexible solution for implementing control systems. A central computer was deployed to control every step of the entire process. An individual current or voltage loop linked each sensor and actuator to the central computer through I/O racks. This allowed the central computer to read sensory data from many sensors simultaneously, and control and synchronize several actuators. The main computer was in the centre of what is called a "star topology" network of actuators and sensors.

Although being the best solution at the time, such systems had major shortcomings such as requiring a pair of wires to connect to each peripheral and appropriate hardware to convert the data for transmission. This complex wiring system also severely limited the bandwidth of data transmission [32].

2.2.3 The Emergence of Distributed Control Systems

Given sufficient computing power, signal interfacing to all parts of the process and an efficient program, a single central computer was theoretically sufficient to control any process in spite of major flaws including lack of robustness, lack of scalability and difficulty of upgrades [25, 26]. The major problem faced by the engineers was the increasing complexity of designing one program for a single central computer to control all aspects of a large process.

In the late 1980's, the semiconductor industry released microcontrollers which were low-cost and downsized computers integrated in one chip [30]. Microcontrollers

were soon deployed alongside sensors and actuators. The use of microcontrollers increased the performances of the sensor, by providing a higher accuracy and resolution, a remote, automatic pre-calibration, pre-processing, diagnostics, automated configuration, and most importantly, data communication management [31]. The microcontroller was capable to take charge of some pre-defined routines, or implement simple, local control loops, with less central supervision.

The main contribution of microcontrollers was the potential of relieving the central computer, by taking under its control some simple parts of the global task, and leaving the tasks of synchronization and high-level decision-making to the central computer [32].

For example, instead of letting the central computer determine and regulate the speed of a motor, it could simply transmit a target speed to the motor's microcontroller, and this microcontroller would regulate the speed of the motor locally, according to the command received from the central computer.

Microcontrollers were used as the core of the Programmable Logic Controller (PLC), which became widespread in factories, bringing simplified techniques to design control systems (ISaGRAF, 1995) [33].

The increasing development and availability of computers and microcontrollers led to the concept of Decentralized (or Distributed) Control Systems (DCS). DCS offers the advantage of "deploying the system in accordance with the physical layout of the plant, the minimum wiring criterion, and specific environmental requirements" [31]. From the very beginning, the expected gains of simplicity and low cost brought about by a minimalistic wiring scheme were the major forces pushing the development of this concept.

2.2.4 The First Standard of Control Networks: Fieldbus

In a DCS, the implementation of intelligence and control can be achieved only if a serial network such as a Field Bus can interconnect all the instruments to both themselves and the higher level [31]. Clearly, the direct hard-wiring method employed to interconnect the elements of a process was a major hindrance. Indeed, it had two major downsides including cost and non-scalability. The initial deployment

and maintenance costs of the wires were high, and hardwiring was a non-scalable option as it did not offer any flexibility, severely limiting the capacity to upgrade or expand. The further development of DCS continued to address this constraint by sharing the control devices through a network [32].

With the introduction of digital communication techniques, the potential to use a "single wire" to interconnect several units (serial bus) became feasible. Using digital communication also removed the necessity for analog-to-digital (and *vice versa*) conversions.

The proposed solution from the industry was the development of a network standard referred to as *Field Bus*, (recently changed into Fieldbus) defined in the IEEE standard as P1046-1991. The Field Bus standard is described as: "A digital data communication that makes possible the exchange of information among periphery or separate elements of a Distributed Control System or between such elements and other sub-systems" [35]. The aim of the Field Bus[1] network is to distribute the "intelligence" to all levels of the process.

At the early stages of the design of Fieldbus, several efficient standards for computer networks emerged. However, serious incompatibilities were identified between the micro-computer networks and industrial control networks. The most obvious difference was the way the bit rate and delays were treated. While bit rate is one of the main characteristics of a computer network, it is of little concern in the design of a control network. A control network should be reliable, support real-time and have a small delay. Most of the computer networks are only "best effort" and do not guarantee small delays. These considerations led to the decision to create specific network standards for industrial control.

The research community, anticipating the issues faced by computer networks (competing standards and absolute lack of inter-operability) decided to create a standardized and open interface for Fieldbus. As a result, the International Fieldbus Consortium (IFC) was founded in 1990. All applications of the Fieldbus standard

[1] Here, 'a Field Bus' refers to a new concept for field networks, whereas 'Fieldbus' is the name which is given to a group of industrial network standards.

were to be compatible with the 7 Open System Interconnection (OSI) layers [34]. Likewise, most of the standards chosen to implement the lower layers were IEEE 802 standards (Ethernet, Token Ring, Token Bus, etc).

From one Fieldbus standard, several Fieldbus solutions from different industrial corporations in various countries emerged. An overview of the current Fieldbus standards can be found in Section 2.3.3.1.

2.2.5 Current and Future Trends: The Emergence of New Applications and Constraints

Most of the research and development on control networks over the past decades has focused on industrial process control applications. The numerous technological advances and standardization efforts resulted in the creation of optimal control networks. These network standards, now widely adopted, have played an active role in supporting the development of industrial processes. The wired control networks will remain the most adequate solution for plants and factories. Indeed, wired control networks match the requirements of harsh and hazardous industrial environments and the demand for maximum reliability. This is why Fieldbus control networks have successfully maintained their position as the industry standard for the past twenty years, and are likely to continue to do so.

With the new millennium, a new area of development and application has emerged for control networks. Such applications comprise of military battlefield management, animal habitat monitoring [23], ground and air surveillance, various agricultural applications, home automation, etc. These new fields of application which are very different from industrial environments have brought about new perspectives and constraints to control networks. The new applications have characteristics and requirements different from factory applications including high autonomy and low maintenance, flexibility and adaptation in dynamic environments, extremely small size, rapid and sometimes random deployment, soft real-time constraints, automatic handling of the failed nodes, etc. Some applications do not require real-time operation: the data can be stored during a certain period of time, and be processed later, when more energy is available.

Clearly, a control network node meeting such constraints will be wireless, battery powered, and possibly integrated on a single chip. The software will include protocols and algorithms providing a support for intelligent and autonomous operation, with flexible and dynamic adaptation to the changes in the environment or network configuration. A review of the major research projects on modern wireless control networks will be provided in the Section 2.4.

2.3 Modern Control Networks

In this section an overview of modern control networks, their application, and specific characteristics compared with other industrial networks will be provided.

2.3.1 Control Networks: General Approach

A control network is a network which interconnects sensing, actuating units and processing units in a control system. It is a complex and autonomous system which transports a flow of information through all levels of a process, from the central computer to the sensors and actuators, and also the command console. The processes supported by a control network can be of many types and serve various purposes within diverse fields. It can range from industrial manufacturing processes to a brake system in a car.

A control network is the framework which supports decentralized systems. It is much more advanced than the basic point-to-point communication medium used to wire the centralized architectures. Modern control networks aggregate the communication medium, hence minimizing the wiring complexity and costs. They embed advanced routing algorithms; manage congestion and priority to guarantee timely delivery, integrity and coherence of messages.

The predecessors of control networks were sensor networks. The earliest initiatives were led by military research (SOSUS, AWACS and DSN), during the Cold War [28]. The research explored the feasibility of monitoring of a large environment such as the ocean or a field to detect submarines and boats, or planes and motor vehicles.

A sensor network is a limited version of a control network which only monitors the environment whereas a control network reacts to signals received from the environment and attempts to make changes through actuators. The control network manages the entire process by controlling the actuators, based on the execution of a algorithm, and the reading of the relevant sensors.

As mentioned in the previous section, the major development area for control networks has been industrial process control. Control network standards emerged in the early 1990's, with the introduction of the Distributed Control System (DCS) concept. This concept is very broad and embraces different research areas including sensing, communication and computing [13].

- Control, sensing and actuating encompasses the sensor and actuator types, control techniques and data acquisition or control orders.
- Communication deals with access and distribution of information over the network. The information is made accessible via a control network. Control networks are studied at different levels, called layers. The lowest layer is usually the physical (communication medium) layer, and the highest is the top-level application. The Open System Interconnect (OSI, IEEE RFC 1983 [34]) standard stack is the most common 7-layer stack model.
- Computing covers many aspects including the physical hardware platform which carries out the computation, the global structure of the different computing units, the deployed algorithms, the human-machine interface, etc.

This literature review will only address the last two subjects which are relevant to this research project; communication and computation will be discussed separately in the following sections.

2.3.2 Wireless Communication Standards

Although successful wireless communication systems have been achieved since the early 1900's, the development of reliable wireless communication standards only dates back to the 1990's. The advent of wireless standards such as the ones used for mobile phones (GSM), and more recently computer (WiFi) and device networks

(Bluetooth) have proven that wireless communication can now be effectively used to support large networks.

The main benefit of wireless solutions is cost reduction. Indeed, the cost associated with wires linearly increases with the distance between two nodes. As the cost of hardware decreases, wiring cost progressively represents a greater part in the total cost of a unit. Studies have shown cases in which the cost of wiring for a given sensor unit represents up to 80% of the total price [8]. Besides, wired solutions not only involve acquisition costs, but also significant installation and maintenance costs. Retrofitting of wires is generally a difficult task, sometimes impossible. Similarly, the maintenance of communication wires (although not frequent) is a tedious task, and remote fault-area detection on wires requires specialized equipment.

Even though wireless hardware is generally more complex and more expensive to purchase, the cost is not significantly influenced by the distance between the two nodes, nor the number of nodes within the network. In addition, modern wireless solutions are made increasingly simple to install and set up while also requiring less maintenance.

Since the mid-1990's an increasing number of wireless standards have been developed for specific functions. The most common applications are: mobile phone networks, computer networks, and peripheral-to-workstation communications. Some manufacturers have developed proprietary solutions (wireless keyboard and mouse are some examples), while others have formed various alliances to create open standards (Bluetooth, GSM, ZigBee). The open standards aim at increasing interoperability, and reducing the research and development costs, by using an identical common platform.

The following part of this section will focus on wireless communication standards. Similar to Fieldbus systems, several wireless communication standards can potentially be applied to control systems, and there are many more proprietary and non-standard wireless systems. The list of standards considered will be narrowed down to those which have the potential to meet the requirements of a Wireless Ad-hoc Control Network (battery-powered, adequate range and network size). The review will also focus on certified and widely adopted standards. The applicable

standards will be presented and analyzed with respect to target application. Table 2.1 presents the main characteristics of the wireless communication standards considered in this literature review.

Table 2.1: Comparative Table of the reviewed Wireless Technologies [7]

Standard	Typ. range (m)	Throughput	Avg. batt. life	Nw. size	standard	application
Wi-Fi	< 100	54 000 kbps	1day to 1 week	30	open	Multimedia internet
Bluetooth	< 10 (low-power)	720 kbps	A week	7	Open	Wire replacement
IEEE802.15.4	<30 indoor, <100	250 kbps	years	64,000	Open	Control networks

A. Wi-Fi (Wireless - Fidelity)

Wi-Fi is the most successful standard for computer wireless networks. It provides high-speed wireless connectivity to laptops and computers in the neighbourhood of an Access Point (AP). Due to the mass production and efficient standardization, this technology is very affordable, well developed and interoperable.

Colandairaj et al. [1] proved, using simulations, that the Wi-Fi standard is suitable for control networks. They demonstrated that Wi-Fi can cope with the high-noise industrial environments, as long as the Wi-Fi bit rate is well above the control network's target. This is not a problem for Wi-Fi networks, as this standard is designed to provide high throughput. However, the main drawback of the Wi-Fi standard for WACNets is that high throughputs are achieved at the cost of high energy-consumption. Hence, the average lifetime of a battery-powered Wi-Fi transmitter does not go beyond a week. A wireless standard for WACNet requiring every node to be wired to a power outlet is obviously not a viable solution. Indeed, the objective of WACNet is to create a wireless network with no communication or power wires. The Wi-Fi standard also imposes a limit on the network size, both geographically and in terms of the total number of nodes.

The characteristics of Wi-Fi make it unsuitable for WACNets, as WACNet nodes are expected to be numerous and to run on batteries with a high autonomy.

B. Bluetooth

Bluetooth (BT) provides a short-range, point-to-point connection between devices within a Personal Area Network (PAN). The Bluetooth standard has been ratified as the IEEE 802.15 standard [17]. Bluetooth was originally created as a wire replacement solution for electronic devices working synergistically. Some examples include a printer and a computer, a telephone and a headset, and generally portable devices with larger appliances. Compared to Wi-Fi, Bluetooth has a smaller range (10m only) and throughput (720 kbps max.), but offers a lifetime for battery-powered nodes of up to one week. Bluetooth is also much simpler from a hardware or software point of view, and more economical. The Bluetooth network topology is similar to Wi-Fi, with several nodes clustered around one coordinator.

The Bluetooth standard defines ranges of up to 100m and the possibility of deploying multi-hop mesh networks with routing algorithm. Although some Bluetooth hardware has been developed to meet these specifications, hi-power and multi-hop Bluetooth networks are yet to be adopted by industry. The Bluetooth standard has been developed primarily for applications in Personal Area Networks, which do not require long-range communication or multi-hop networking. Consequently, the current Bluetooth solutions do not allow more than 7 active nodes per network, and a standard to implement the multi-hop network specification is yet to be adopted. The Bluetooth nodes also induce high latency, especially for node wake-up delay [18].

Ramamurthy et al [3] studied a multi-RF platform for distributed control network. The platform was tested with both Bluetooth and Wi-Fi systems, and the comparative analysis showed that Bluetooth is better adapted to industrial control, whereas Wi-Fi should only be used for applications requiring high data transfer rates, such as images, sound, etc.

Leopold et al. [2] have demonstrated the feasibility of self-configuring multi-hop sensor networks, using Bluetooth as communication media. The research group has

reached significant throughputs (up to 35Kbps, 668bits payload), but showed a battery life expectancy (with two AA batteries) of 8 days, which is unacceptable for applications such as home automation or industrial control.

C. IEEE 802.15.4 [9, 8]

The IEEE 802.15.4 standard was developed around the year 2003 to provide "a standard with ultra-low complexity, cost and power for low-data-rate wireless connectivity among inexpensive fixed, portable and moving devices".

This standard achieves very low complexity: the total number of primitives is only 1/3 of Bluetooth's, and is very suitable for minimalistic devices, such as simple sensors.

The IEEE 802.15.4 standard defines two of the 7 OSI layers: the Data Link layer, OSI layer 2 (MAC) and the Physical layer: PHY, OSI layer 1. The physical layer transmits using DSSS (Direct-Sequence Spread Spectrum) over two different bands: 2.4 GHz band, where 16 channels are available, delivering a speed of up to 250 kb/s and 915-868 MHz, where a total of 11 channels are available at 10kb/s. The stipulated range is up to 30m for indoor and 100m for outdoor environments. The standard enables various topologies including star/clustered, as well as mesh (peer to peer), and uses two types of addresses; physical (8-bit) and network (64-bit). The Data Link Layer supports association and disassociation which is a fundamental requirement for a dynamic network aiming at self-configuration.

The Medium Access Control allows two different modes; beacon-enabled or disabled mode. The use of Beacon-enabled mode allows the assignment of Guaranteed Time Slots (GTS), which form a contention-free period. The frequency of the beacons can be modified from a wide range of 15ms to 245s, depending on the latency required.

The IEEE 802.15.4 standard appears an ideal communication medium for WACNet applications, mostly because it can be used by numerous low-power and simple nodes. It matches the requirements set by a control network which includes low latencies, and reliability, for low data rates. Finally, it is a very flexible standard

which can be used in multiple ways by higher layers to implement peer-to-peer or centralized networks, and flexible enough to allow dynamic reconfiguration.

2.3.3 Overview of Control Network Standards

In the following section the characteristics of control networks will be reviewed. The different groups and associated standards will be presented and analyzed. The wired standards will be presented first, followed by the wireless standards.

2.3.3.1 Wired Standards for Control networks

A. The First Control Networks: Fieldbus Standards

From the end of the 80's and throughout the 90's, many control network standards emerged complying with the Fieldbus standard. These Fieldbus systems were the first ones to match the description of a control network. Indeed, they were engineered to meet the specific requirements set by industrial process applications including real-time operation, very high reliability and robustness, and lower throughput compared with computer networks. Reaching well beyond the capacity of point-to-point communication lines, these networks exhibited many innovations such as providing an autonomous network of hundreds of nodes, offering various topologies, and allowing several nodes to communicate via a single bus. Most importantly, they were at this time the first systems with the capacity to perform efficient data packet routing over an area as wide as a factory.

The "Fieldbus" term usually refers to a dozen mainstream and multi-purpose standards, although several hundreds of application-specific or manufacturer-specific systems similar to Fieldbus exist. Over the last two decades, Fieldbus networks have continuously evolved and are still the most commonly used systems in plants and factories. Protocols such as CAN (Controller Area Network), AS-Interface, LONWorks and HART are some of the most popular Fieldbus systems.

The Fieldbus transmits data using fully digital communication lines. Recently, the nodes have become more and more powerful and multifunctional. For example, sensor or actuator nodes are now capable of routing, hence reducing the need for dedicated infrastructure nodes (such as dedicated repeater and bridge nodes). The

nodes also provide a support for network self-diagnosis. Every Fieldbus standard tends to be specialized for one specific application [14]. In most large factories, different Fieldbus standards co-exist. For example, the simple and cost-efficient AS-i Fieldbus is often found on assembly lines for simple sensors, while the LONWorks or PROFIBUS standards are most commonly used for control network backbones.

Fieldbus standards can support quite complex networks. They are robust and reliable in harsh conditions but are also expensive. The cost factor restricts their use to high-budget industrial applications.

B. Ethernet-Based Control Systems

Ethernet networks are local area networks based on the TCP/IP protocol and Manchester-code transmission. Ethernet is currently the most popular standard (becoming the only solution) for wired local-area computer networks. This standard has now reached a very advanced stage of development and the mass-production of compliant hardware has resulted in the availability of low-cost infrastructure elements. As a result of the low-cost, advanced technology and high availability of this standard, many attempts have been made to convert Ethernet into an industrial control network system.

The application of an Ethernet network is fundamentally different from of a control network. Ethernet networks focus on delivering the highest throughput, in a “best effort” mode. Ethernet’s layer 2 protocol, the Carrier Sense Multiple Access with Collision Detection (CSMA CD) involves the probability of message collision, which can seriously delay the transmission of the information. In case of collision, only limited retries will occur, which means that the delivery of the message is not guaranteed. Unreliability and unpredictability of the network are the main obstacles to the deployment of a TCP/IP network in industrial control. A transfer of Ethernet standard to a control network can only be successful if latency and reliability are prioritized over throughput.

A Fieldbus protocol called EtherCAT has been designed to address the shortcomings of an Ethernet network for control applications. EtherCAT retains the advanced and proven routing algorithms, efficiency and throughput of an Ethernet

network, while meeting the fundamental needs of a control network [15]. The result is a protocol compatible with the same mass-produced Ethernet hardware and cables. CSMA/CD is replaced by a deterministic media access protocol. The packets are downsized, to fit the needs of real-time control networks by providing frequent, low-latency, and short messages. The major problem associated with EtherCAT is that the nodes cannot be powered through the Ethernet wire as offered by many competing control networks. This will significantly simplify the deployment of the network and its autonomy.

C. BACNet^{tm}

Building Automation Control Network (BACnet) is an example of a control network designed for a specific application. BACnet defines a set of standardized high layers to implement building control networks. The BACNet initiative was announced in January 1987 by the American Society of Heating, Refrigerating and Air-conditioning Engineers (ASHRAE). The objective was to provide an industrial-grade dedicated protocol to realize an interoperable building control network with equipment from different vendors. BACNet is designed to control appliances, make sensory signals available at a central point of network, transmit configuration and settings point, and carry out schedule requests. It defines two layers of application and network layers. The data link and physical layers (OSI layers 1 and 2) can be "borrowed" from various Fieldbus systems, LONTalk being the most common one. A complete review of BACNet can is available in [30].

2.3.3.2 Wireless Standards for Control Networks

A. ZigBee

ZigBee is an open wireless standard designed for control networks and operates on unlicensed bands. It has been developed by the ZigBee Alliance, which is backed by major industry leaders, including Freescale, Phillips, Honeywell, Siemens and ST microelectronics. The ZigBee standard is based on the IEEE 802.15.4 which defines a physical layer (PHY) (OSI layer 1) and MAC (Medium Access Control) layer (OSI 2). The PHY layer operates in the same ISM band as Bluetooth (975-868MHz & 2.4

GHz bands). ZigBee features very low power consumption, low latency (around 15ms). Its network layer (OSI 3) implements an Ad-hoc On-Demand Vector (AODV) routing protocol, capable of routing packets over a network of several thousand nodes (64,000 = 2^{16}) [18, 8]. The range of a single node is up to 30m indoor and 100m outdoor, and the stipulated throughput reaches to 250kbps. While the maximum throughput is only 25% of the Bluetooth's capacity, it is still sufficient for a control network.

The extremely small size of the ZigBee transmitters makes them easy to retrofit. Currently, a ZigBee node costs approximately around USD $15 to $20, but the prices are expected to decrease like those of Bluetooth hardware when mass production is reached.

The ZigBee standard can be integrated in a control network to form wireless sensing or actuating nodes in three different ways:

a) The ZigBee transmission function can be incorporated into the sensor or actuator's microcontroller by including a ZigBee stack.
b) It can be used as an independent, off-the-shelf module which is linked to the sensor/actuator's controller.
c) Some off-the-shelf sensors currently have ZigBee network connectivity as a built-in function [21].

A ZigBee network is organized around a ZigBee Coordinator. The coordinator is the node which starts the network and determines its initial parameters (Personal Area Network number, transmission channel, etc.). Once a coordinator is started, ZigBee Routers or End Device nodes can join the network.

The ZigBee End Device behavior is especially designed for low-power nodes. End Device nodes can only communicate with the Coordinator or a Router, and do not store any routing information. A ZigBee router can act as a bridge, thus extending the network coverage beyond the range of the coordinator's transmitter, and therefore provide multi-hop data transfers to End Devices and other Routers.

The ZigBee Alliance is currently developing higher layers for control networks which include common platforms for increased interoperability (called Application Framework). Such platforms are intended for home automation systems, energy

management, etc. The full ZigBee specification can be obtained from the ZigBee Alliance website [22].

B. Z-Wave

Z-Wave is a proprietary wireless network solution exclusively designed for home control. It has been created by Zensys Inc. and is endorsed by the Z-Wave alliance (over 70 members, including Intel and Panasonic) [11, 10]. Z-Wave is a standard for wireless, ad-hoc and self-configuring control networks. The specifications of Z-Wave are comparable to those of ZigBee. That is why it is currently recognized as the main competitor of ZigBee in the field of home automation applications [19]. Indeed, Z-Wave and ZigBee share the following characteristics: standardized platforms for interoperability (in the domain of home automation only), mesh networking, low-latency, range of 30m indoor and 100m outdoor, very low power consumption (battery operated nodes can last for up to 5 years), and a relatively low data rate of 9600bps for Z-Wave.

The major difference between Z-Wave and ZigBee is their scope. While ZigBee is generic in regard to the application of the control network, Z-Wave is focused on low-budget home automation. Since the Z-wave modules are built for a specific purpose, they offer less possibilities and are less advanced than the ZigBee chips, which results in a lower retail price (presently USD $4 [20]). Currently the development of the Z-Wave-based products far exceeds that of ZigBee's. As a result, several comprehensive Z-Wave home automation systems are now available on the market, while ZigBee modules are still mainly sold as evaluation products and prototypes or tailored, high-budget applications fitted by specialists (in large hotels or corporation buildings).

On the other hand, ZigBee has a broader application, is backed by a growing number of strong industry leaders, and is extensively referred to in the wireless control networks literature.

2.4 Next Generation: Wireless Control Networks

In this section, the latest projects on new generation wireless control networks will be reviewed. The first part will deal with wireless control and sensor networks by introducing each project and underlying its specific characteristics. This will be followed by innovative solutions developed by various research groups to tackle challenges faced in this area.

2.4.1 Research projects

This section will introduce and analyze the predominant wireless control networks research projects. Each paragraph will provide an overview of the project, and highlight the special characteristics of each project.

2.4.1.1 "Smart Dust" Project at Berkeley University [28]

The smart dust project aims at realizing the development of miniature wireless nodes. The project uses MEMS technology to achieve node sizes below a cubic millimeter with extremely low power consumption for extended lifetime.

The most unique characteristic of this project is the wireless transmission medium. Most of the research projects investigating wireless networks choose to communicate using high-frequency radio transmissions, whereas the smart dust nodes use light emission. A combination of laser and CCD (Charge Coupled Device) is used to transmit and receive data, respectively. The capacity of the smart dust nodes to transmit weather-related information at a distance of over 20km (experimented over the San Francisco Bay), and a full-duty lifetime of a day has been demonstrated. The conclusions are that wireless transmission via light signal is worth investigating, although the line-of-sight transmission constraint for light signal is an impediment to random deployment. The applications considered in this research project only explore the Smart Dust's capacities as a sensor network.

2.4.1.2 Wireless Integrated Network Sensors (WINS) Project at UCLA [3, 24]

The WINS project comprises of a series of research projects which explore the capacity of dense sensors, controls and processor networks. The nodes presented are

only acquiring data and perform localized signal processing (seismic activity in [24]). The study provides a thorough analysis of various processing structures, with a focus on consumption minimization. The authors conclude that local processing and multi-hop communications are key factors when aiming towards reducing the energy consumption.

The ReWINS project [3] explores the possibilities offered by a flexible network supporting various radio transmission standards. The research outcomes present a hardware node design, software architecture, and protocol for network set up. The set-up protocol is simulated and yields successful results.

2.4.1.3 Low-Power Wireless Sensor Networks at MIT [36]

This project introduces low-power wireless sensor networks for generic applications. The authors justify their work by the fact that collaborative and dense networks of microsensors can achieve better results than macrosensors.

The node presented is the MIT μAMPS communicates through a custom-built 2.4GHz radio module (same as Bluetooth). The research team studies and validates the use of microprocessor voltage and frequency scaling to lower the electrical consumption. While developing the radio module, the group highlights the importance of message size. They demonstrate that larger messages lead to less individual transmissions, which results in a lower consumption. The research outcomes present an interesting concept of beamforming, allowing the most appropriate sensor to use more power, while disabling the less relevant sensors. This method yields overall higher sensitivity and resolution, while reducing the overall power consumption. Finally, the group has introduced an Application Program Interface (API), allowing any non-expert programmer to develop an application on this platform, without understanding the complex transmission and consumption-reduction mechanisms.

2.4.1.4 Unattended Ground Sensor System (UGS) at Sensoria Corporation [25]

This study is part of the Defence Advanced Research Projects Agency (DARPA) “Self Healing Minefield” (SHM) project, investigating the potential of ground sensor

networks. The first application considered is for a Self-Healing Minefield in which nodes are embedded in anti-tank mines, with the objective of ensuring a complete and consistent deployment of an area. This project is quite unique as it introduces the concept of geographically relocatable nodes (the mines are equipped with rocket thrusters). The thrusters are used to move the nodes, in order to modify the geographical distribution of the network. This ensures a consistent coverage of the area, by getting a neighboring node to "fill in a gap" or replace a mine deactivated by an enemy.

The nodes are equipped with various sensors (including accelerometers for node localization), Li-Ion batteries, and a microprocessor running a Linux kernel. A complete study of energy efficiency and battery lifetime is also provided. The research group positively verified the capacity of the network to localize and set up nodes.

In the second application, Sensoria Corporation's WINS (refer to B.) NG nodes are developed for the DARPA SensIT program. Seventy-five nodes with video cameras, microphones, passive infra-red sensors and GPS were deployed on a military training field, to monitor ground activities. The successful application was developed in three weeks and was robust enough to operate continuously for one week.

2.4.1.5 ZigBee-Based Initiatives

In October 2006, the Ember Corporation, manufacturer of ZigBee chips, announced that a business partner had developed and installed several energy monitoring and building control systems, using Ember ZigBee chips. The company boasted a 40% reduction of the energy cost of a hotel in Toronto. The platform used is Riga Development Inc. WiSuite™ [38], a ZigBee-based building control network for hotels. The system is used in conjunction with a scheduling application, and an interface at the hotel reception. The system wirelessly controls the lights and HVAC system of the hotel's suites. At the time, Ember and Riga Development announced that the system was available at USD 350 per hotel room [37].

The August 2007 ZigBee newsletter focuses on the Siemens' APOGEE system which is a ZigBee-Based home or building automation system. The Siemens modules control HVAC systems via a ZigBee-based network, using Ember chips [39, 40].

In October 2007, Develco announced the release of SmartAMM devices including ZigBee-enabled integrated meters for gas, electricity and water. The aim was to extract resource-consumption data from each individual appliance within a building, via a ZigBee-based monitoring network. This solution had the aim of making the energy-consumption data available to the users. The SmartAMM system does not take any action to reduce the consumption. Hence, it cannot be classified as a control network [41].

2.4.2 Wireless Control Networks: Challenges and Solutions in the Literature

2.4.2.1 Remote Control Interface

The early wired control systems were administered from a monitoring panel or screen, located in a pre-determined and fixed location. The newer wireless control systems benefit from the inherent mobility and flexibility of wireless solutions. This means that the control interface can be displayed on various hardware systems, and in diverse locations. Platforms for control terminal include portable computers, purpose-built consoles or even handheld devices.

The ReWINS wireless control network project [3] presents a node-centric approach for monitoring. A node referred to as "aggregator" collects and dispatches the messages issued by the monitoring interface. Four-byte IP addresses are used to enable Internet remote monitoring.

A platform-independent and portable language is necessary for a device-independent control terminal. The concept of a monitoring station using a system similar to a dynamic Internet page or a Java Applet is the most convenient solution [5, 22]. Indeed, Java is a platform-independent programming language, and Internet is the most common network standard, offering many possibilities for connection (wireless (Wi-Fi), or cable). The use of XML (eXtensible Markup Language) also

allows a consistent platform-independent reading and presentation of the data. Provided with a common interpreter, a uniform presentation of the interface can be achieved, similar to how Internet browsers do for XHTML pages.

2.4.2.2 Node Interoperability

Node interoperability is one of the crucial points for the future success of Wireless Control Networks. History has proven that adequate standardization is a common point between all the successful networking technologies (GSM being the best example).

The importance of interoperability can be illustrated by considering a simple light switch. It is unreasonable to question whether a light switch acquired from a hardware shop would successfully turn a light on and off. Any domestic switch is compatible with any domestic light. Yet, when it comes to a wireless light switch, the problem of interoperability becomes important. This requires that the appliances are compatible across both the RF standards and communication protocols. For consumer applications, "Plug and Play" (PnP) functionality is the main requirement. Ideally, any wireless node should operate with the rest of the nodes on the network with minimal time devoted to configuration.

Otsuka et al. [42] have developed a concept of interoperability based on the existing interoperability-enabling standards (UPnP, ECHONET). They use translating bridges to communicate with the monitoring station. In this concept, each device is able to provide a definition of its actions, a standard GUI (Graphical User Interface) for the control screen to display, and a method to map the GUI to actions.

Pottie and Kaiser [24] recommend an application compatible with conventional networks and databases, so that the data can be accessed from any platform. They designed gateways to interconnect Wireless Integrated Network Sensors (WINS) with TCP networks.

2.4.2.3 Network Type and Topology

Wireless control networks are designed for specific applications where wired networks are not viable or cost efficient. In some cases, they can offer a better

alternative in dense networks than wired solutions. In dense networks, the short distance between nodes allows multi-hop communications, thus forming an infrastructure-less network. A multi-hop complying topology combined with sleep-enabled radios can meet the low-energy consumption requirement of the wireless control networks [24].

Generally, clustered architecture is the most recommended architecture. It allows an efficient coordination of local objectives and common goals in decentralized architectures. A centralized structure is unsuitable for wireless control networks as it provides a central point of failure, lacks scalability and is not energy efficient [26]. Clustering also eases the compilation of data for the nodes working together. The WINS project proposes a clustering technique which does not require any address. The cluster formation is based on the strength of the relationship with other nodes. Messages are broadcasted and forwarded through a limited number of hops which depends on the strength of the relationship.

Several projects (including [26, 25]) also highlight the importance of dynamic reconfiguration, to adapt to failing or moving nodes [26], or even nodes disabled by the enemy in military applications [25]. Wireless control networks must be able to adapt to modifications in the structure of the network. Therefore it is important to create a “loose” network topology which can adapt to the changes in setup or environment.

2.4.2.4 Distribution of Control Tasks

Since wireless transmission consumes more energy than wired transmission, the amount of data sent over the network must be reduced as far as possible. This can be done by defining an appropriate scheme for task distribution.

A substantial amount of research has been carried out in the area of task distribution in the WINS research projects [24, 26, 27]. The WINS architecture promotes local processing of the data, implying that the unit acquiring the data will process it as far as possible before transmission. Pottie and Kaiser present an analysis of the energy cost of transmission over computation. They conclude that with the energy used to transmit 1kBytes of data over 100meters, a general-purpose processor

could execute 3 million instructions [24]. Kahn et al. found that while computing cost could go as low as 1nJ per 32-bit instruction, a Bluetooth-band radio still burns about 100nJ for each bit transmitted [28]. In both cases, the findings point towards maximal local computation, in order to send the shortest possible message. Both of these research projects underline the necessity to give priority to low-energy consumption, over time and latency.

Estrin *et al*. confirm this concept of task distribution. They advocate localized algorithms, only interacting with sensors in the vicinity, where groups of nodes achieve a collective objective [26]. The difficulty associated with the design of such algorithm is also acknowledged. These difficulties exist due to the presence of complex relationships and dependencies involving local and global behaviors [28].

A fundamental difference between sensor networks and computer networks is also highlighted [26]. The control networks are data-centric and specific, while computer networks are node-centric and general-purpose. In such cases the localization of a sensor is more important than its precise identification. An example found in the literature is a temperature-monitoring application over a large geographic area. This application is more likely to require the location of the nodes sensing a temperature above a certain threshold, than it is to query individual nodes for their temperature reading. Thus a unique address is not always required, as long as a node knows its location [26]. The same research group also shows that sensor networks are meant to be tailor-made to the requirements of a specific application, unlike computer networks which are generic.

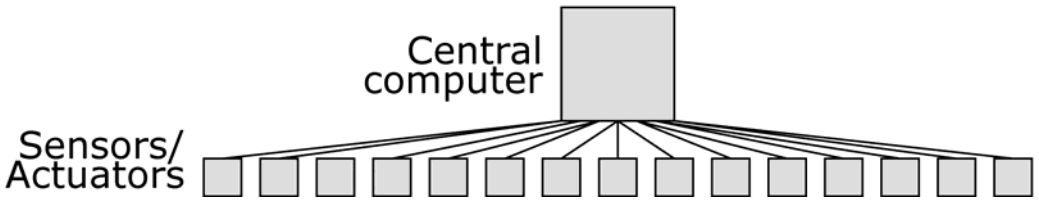


Figure 2.1 : Centralized architecture hierarchy tree.

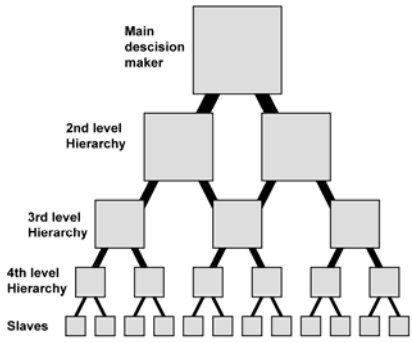


Figure 2.2 : Decentralized architecture hierarchy tree.

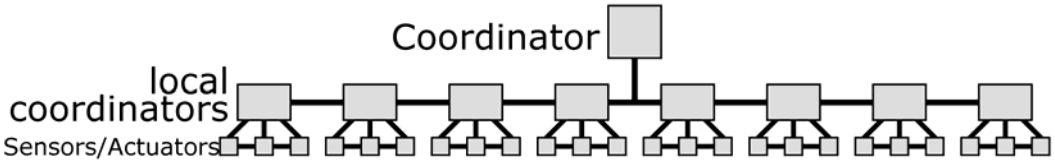


Figure 2.3 : Collaborative architecture hierarchy pyramid.

The hierarchy of tasks can be represented by a tree diagram (Figures 2-1, 2-2 and 2-3). The root of the tree diagram is generally a coordinating node. In the case of a distributed control system, this hierarchy should comprise of multiple levels. The top-level node is solely used as a coordinator, by shifting the authority to lower nodes (Figure 2.2) [5]. For a collaborative architecture, horizontal communication should form the main stream of information, while vertical communication should take place only on-demand or based on events, such as logging applications or high-level control by the end user (Figure 2.3).

Recent military research projects, especially the SensIT project, are developing new networking techniques with military field-specific objectives. The aim is to create wireless networks, designed “for rapid deployment in an *ad-hoc* fashion, and in highly dynamic environments” [28]. The objective is to efficiently share the information acquired on a battlefield and obtain a “common operating picture” [28].

2.4.3 WACNets [5, 6]

The concept of Wireless Ad-hoc Control Networks (WACNets) represents a new stage in the evolution of distributed control and monitoring. It explores a framework for organic, evolutionary and scalable integration of a large number of nodes with sensing and/or actuation, local intelligence and control, data processing and communication capabilities. WACNet control networks are focused on autonomy and self-reliance. The network is capable of dynamic self-configuration and healing, while the nodes should be capable of reaching battery autonomies of up to several years. The WACNet framework promotes decentralization and task distribution, and is based on a hierarchy structure with very few levels. Intensive horizontal communication through inter-node cooperation allows efficient task distribution with limited supervision or central coordination.

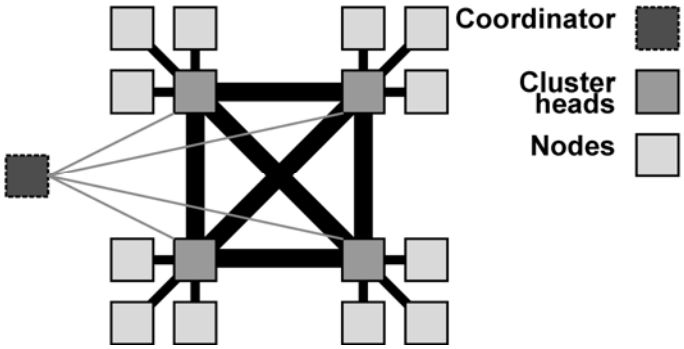


Figure 2.4 : WACNet Architecture.

A WACNet consists of a set of geographically distributed, intelligent and heterogeneous nodes. Each node consists of a wireless communication module controlled by a processing unit. The nodes provide multiple ports which can connect to one or more sensors and/or actuators. Besides communication management, the processing unit can perform various tasks such as signal and data processing or control, depending on the services required.

The WACNet framework also offers a flexible solution for human control and supervision. A dedicated node acts as a WACNet-to-Ethernet bridge, allowing

bidirectional communication between a workstation using a browser and any node in the network.

The network Topology of a WACNet is based on clusters of nodes which control closely related areas of the process. In a successfully organized WACNet, most of the communications stay within the cluster. In order to allow inter-cluster communications, each cluster has a node elected as the Cluster Head, which acts as a gateway to the rest of the network, as shown in Figure 2.4. A Cluster Head not only serves as a gateway, it also intelligently optimizes the flow of data at a network scale and maximizes the availability of information at a cluster scale.

An architecture based on IEEE 1451 [43] smart sensor standards is deployed for realization of WACNet. The IEEE 1451 compliant smart sensor is modified to include a medium-range wireless communication link. Such a system will have the ability to provide self-identification, self-testing and adaptive calibration.

The previous research projects on the WACNet [5, 6] have led to the development of a basic hardware node, which was benchmarked using a dedicated test-bed. The analysis of the results highlighted the inadequacy of the wireless standard chosen (Bluetooth). The benchmark tests results led to the conclusion that a large-scale WACNet control network running a real-life task is potentially feasible, using adequate wireless technology.

Chapter 3 WACNet Design

3.1 Introduction

The concept of WACNet will be described in this chapter and its evolution since its inception will be highlighted. The contribution of this study to the further development of the concept will be also explained.

An overview of WACNet will be provided first. This will be followed by defining the aim and objectives of this study and outcomes it intends to achieve. The approach and the application designed to validate the work carried out in this thesis will be presented next.

3.2 WACNet

3.2.1 Overview

The concept of Wireless Ad-Hoc Control Network (WACNet) has been under development at the Centre for Intelligent Mechatronics Research, University of Wollongong [44]. It has been designed for distributed control and remote monitoring of a process. In WACNets, control, actuation, sensing and communication are integrated directly into intelligent wireless network nodes.

A WACNet consists of a large number of geographically distributed intelligent and heterogeneous nodes with sensing and/or actuation, local intelligence and control, data processing and communication components. The processing unit can perform various tasks such as signal processing and control as well as communicating with other nodes. As part of a distributed architecture, each node is expected to make local decisions. The size, number, density, capabilities and location-dependency of the nodes are determined by the specific application running on the platform. The WACNet platform is structurally flexible and adaptable. The nodes are expected to be low-cost, low-power, multi-functional and small in size.

3.2.2 Components and Implementation

A WACNet is formed by a set of nodes used to control and monitor the activity of a process through actuators and sensors. A WACNet also comprises of a bridging node, to allow inter-network compatibility to other networks such as Ethernet. Finally, a monitoring and control unit is incorporated in a WACNet to provide remote access to nodes via the bridging node.

The WACNet nodes are the atomic elements of this control network. In the early stage of the project, the nodes were fitted with a ST microelectronics PIC16F877 microcontroller. This microcontroller was the central computing device of the node. The communication standard used was Bluetooth. The Ericsson ROK007 Bluetooth modules were used by the nodes to communicate. The modules were connected to the nodes via a Universal Asynchronous Receiver and Transmitter (UART). Finally, the nodes were fitted with an LCD display and various digital sensors for testing purposes.

The bridging node interconnects the WACNet to an Ethernet LAN network. This node was based on a DALLAS SC TINI board. This board offered various sockets including connectors suitable for Bluetooth modules and Ethernet cables. The TINI board also provided an operating system, which embeds a Java Virtual Machine (JVM). The TINI board operating system and JVM were used to create a web server, and various applications coded in JAVA language.

The monitoring station was used to access nodes in order to control or monitor the system. The station used was a PC computer. The nodes of the WACNet were accessed through the Bridging node, using the PC's Ethernet connection. A JAVA applet was downloaded from the Bridging node's server, through an Internet browser. The bridging node was used as a relay to send messages to the WACNet nodes.

3.2.3 Topology

The Wireless Ad-hoc Control Network is organized in a clustered topology. The nodes dedicated to a common task are gathered in a group called *cluster*. Each cluster has a node known as Cluster Head (CH). The Cluster Head manages the cluster and coordinates the nodes which form part of it. The overall WACNet structure consists

of a group of clusters. Inter-cluster communication is achieved through the Cluster Heads, which provide a gateway service to the rest of the nodes within their cluster.

The networking environment of a WACNet is likely to change during a node's lifetime. The nodes are expected to be mobile and to eventually stop operation when the energy supply runs out. This is why WACNets support dynamic cluster reconfiguration. Cluster rearrangement can also be carried out to improve the task-wise cohesion of the clusters.

The past trend in industry standard has been towards highly centralized structure with many levels of hierarchy, as illustrated in Figure 3.1. Furthermore, the networking hardware and control modules were usually distinct, even different networks for sensing and actuating.

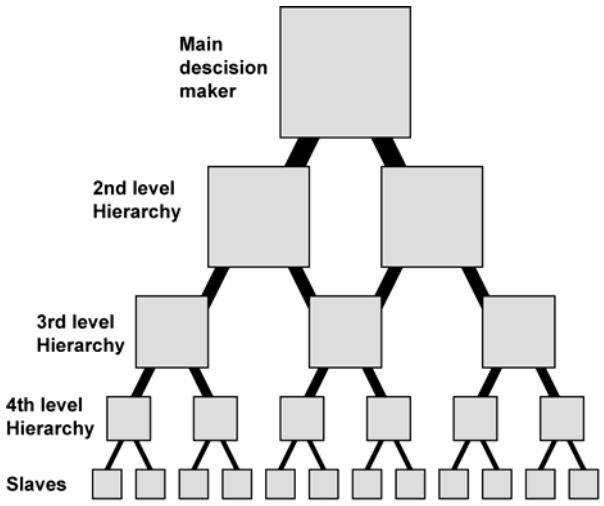


Figure 3.1: Traditional industrial control network architectures.

The WACNet approach is very innovative in this regard. Indeed, unlike architectures in which a central node dictates the behavior of all the remaining nodes, the WACNet relies on the intensive cooperation and collaboration of hierarchically-equal nodes to achieve a global task. A WACNet is classified as *ad-hoc* as the network is capable to adapt to different situations. A WACNet does not have a rigid and predefined architecture and configuration. Instead, the nodes form clusters and elect a Cluster Head in strict autonomy, without prior definition. This self-organization process is carried out at every network boot-up, or when network

condition is changed. Self organization is achieved through the analysis of the current configuration, node characteristics, environment, etc. A typical network topology for a WACNet is illustrated in Figure 3.2.

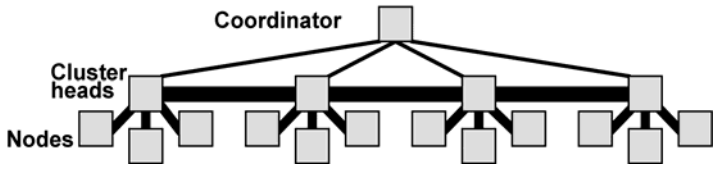


Figure 3.2: WACNet decentralized architecture.

3.2.4 IEEE 1451 Compliant Architecture

The IEEE 1451 [43] architecture is employed to realize the WACNet concept. IEEE 1451 is a family of standards aiming at creating a uniform interface between different types of sensors and networks. It defines standardized elements for smart devices as well as common data-exchange and parameters architectures. This standard provides a uniform way of creating totally self-describing measurement and control devices. The IEEE 1451 standard is deployed in WACNet to take advantage of its characteristics. Bluetooth is used to provide a short-range link between the IEEE 1451-compliant nodes.

3.2.5 Evolution of a WACNet

This section presents the different stages of evolution which lead to a fully-capable WACNet.

(a) Installation

The first step consists of laying out the WACNet hardware. The geographical arrangement must be done in regards to various characteristics of the nodes, such as: range, distance to the monitored or controlled equipment and energy requirement. The nodes are then connected to various sensors and actuators.

(b) Configuration

Appropriate drivers must be downloaded onto the node's microcontroller. These drivers control read and write procedures to the peripherals, and

also the specific data/signal processing or control routines, when relevant. Finally the task-managing application is downloaded to the node's central computational device.

(c) Boot Up

Once the nodes are correctly programmed and powered, a coherent Ad-hoc network must be formed. The first step is to identify the nodes within range.

(d) Node Task Discovery

Once a list of nodes within range is determined, each node queries every neighboring node for their specifications. These specifications include tasks, areas of the process in which the node is involved, sensory and control capacity, computational power, energy resources, etc.

(e) Cluster Formation

Provided with the characteristics of the neighboring nodes, each node is capable of defining a "collaboration metric" in regards to other nodes. The collaboration metric reflects the extent to which two nodes are likely to share information. An optimal cluster configuration is formed with nodes sharing the highest collaboration metric among each other.

(f) Cluster Head Election

As soon as a cluster is created, a Cluster Head node is selected. Usually, the node which present the best combination of stable supply of energy, high degree of collaboration with the rest of the cluster and superior computational capacity is chosen as the Cluster Head.

(g) Network Operation

Once the network is formed, all the clusters begin to operate concurrently. Each cluster is dedicated to one area of the global task. The Cluster Head regularly exchanges messages to guarantee large-scale synchronization. The information is transported across the clusters in a typical Ad-Hoc fashion and the most efficient route is determined by the network. Packets are transported across the network using the Cluster Heads as relays.

(h) Network Evolution

During operation, the network environment and configuration can change for various reasons (new node, node failure, adjustment of the task, etc.) All the modifications of the network environment will result in adequate network re-organization to maintain an optimal ad-hoc configuration, given new characteristics of the network.

3.2.6 Application of WACNet

Wireless Ad-hoc Control Networks are suitable for a variety of applications. Generally, the most appropriate implementations are those facing one of the following challenges: applications where the installation of a wired system is impractical or too expensive, instances where nodes must be deployed quickly, where the system must be autonomous, and generally cases requiring a large number of low cost nodes.

Ideal applications for WACNets are building and home automation, because the nodes are small and autonomous, and the wireless communication eases the deployment. The WACNet can be used to maximize the user's comfort, support security, identification services, access control and restriction, energy management, etc.

Other applications for wireless control networks include agricultural control and biological habitat or geographic monitoring. A WACNet can be deployed in a field to control the growth of crops, by monitoring the characteristics of the soil and taking appropriate measures to maintain ideal conditions for optimal growth. Geographical applications include rainwater channelling and natural flow survey, etc.

Finally, the WACNet concept can be used for military or rescue operations where wireless communication infrastructures are inexistent or have been destroyed. These situations take full advantage of the ad-hoc network structure of the WACNet. In such cases, each node of the network extends the range of the network. This creates large field networks without infrastructure or forward planning, on the condition that the network of nodes remains dense enough.

The WACNet can be used for industrial control network applications, but Fieldbus remains a serious competitor to the installation of wireless systems in plants. Indeed,

wired standards offer the essential levels of reliability and robustness that their wireless counterparts fail to reach [2,3].

3.3 Project-specific aims

A Home Ambient Intelligent System will be developed based on the WACNet platform as a case study in this research project.

3.3.1.1 Testing Application

The system is expected to meet the following requirements:

- The primary goal of the Home Ambient Intelligent system is to reduce the consumption of various resources. This must be done while providing an optimal level of comfort for the users. This task is carried out through issuing of alerts in conjunction with taking preventive measures to reduce the consumption. The system can also take direct action, when required.
- While achieving the first goal, the proposed solution must be cost-efficient. This means that given sufficient time, the accumulated savings become higher than the acquisition and maintenance costs.
- The Home Ambient System is quite generic and can be applied to efficiently manage the consumption of any resource (water, electricity, gas, etc.).
- The program behaves in an intelligent and autonomous way. It learns about the monitored equipment and the habits of its users. Ideally, it takes into account each separate individual, the time of the day, and period of the year.
- The system integrates seamlessly into the user's environment and is non-intrusive. The nodes are highly miniaturized and the network is wireless and infrastructure-less. Nodes can be powered by batteries in cases where a power outlet can not be reached conveniently. The only maintenance task is battery replacement.
- The network and the application do not require any configuration. The network of nodes is self-configuring and self-healing. The system is robust and designed for a fast and simple owner-installation. This means the installation of a WACNet does not require any specific technical knowledge.

- The user can become aware of every detail about the system, via a standard browser. The browser can be hosted on a computer or handheld device connected to a bridging node.

3.3.1.2 Data Collection

To determine the maximal capacities and performances of a WACNet under real working conditions, a series of experiments will be performed. The experiments will provide data which will quantify the following performances:

- The efficiency of the chosen network topology.
- The efficiency of the communication media based on throughput, delays, reliability, data loss.
- Robustness, self-healing, and resistance to node failure and data loss.
- Efficiency of the self-organisation algorithm.
- Efficiency and outcome of the learning algorithm.

3.4 Methodology

The research work leading to the goals listed above will be carried through different stages as described in the following sections.

3.4.1 Upgrade of the WACNet Hardware

In order to reach the targets previously listed, the first step of this project is to upgrade the hardware platform. This will be done in compliance with the WACNet specifications defined in previous research works. A critical analysis of the latest available technologies and material has been made in the literature review chapter, with a special focus on the wireless standard. This analysis justifies the choices made for every element of the upgraded WACNet nodes.

Once the platform's hardware is upgraded to the latest technologies, drivers will be developed to build a hardware-abstraction layer. The aim is to create an Application Programming Interface (API) to ease the development of applications. The API functions will provide access to the hardware functionalities through simple

high-level functions. These functions will control sensing, actuating, communication, display, etc.

The upgraded nodes will be validated through a series of benchmark tests. The WACNet platform will then be simulated at a larger scale, based on the node benchmark tests results. This will provide numerical data on metrics such as the optimal throughput, latencies, memory availability, processing speed, errors and data loss.

3.4.2 Design of the WACNet Nodes' Software

Once provided with a stable and robust hardware platform and API, the top-level software structure will be designed to realize a Home Ambient Intelligent System.

The first step is a comparative analysis of different solutions. The most adapted technique will be used to create the foundations of learning agents and the self-organisation algorithm. When possible, these elements will be simulated using a model based on the first step's simulations. The outcomes of the simulations will be analysed and conclusions will be drawn. This stage will ensure that an adequate solution is implemented as part of the following steps.

3.4.3 Implementation, Measurements and Validation

The application designed at the previous step will be implemented on the target. Once the application is successfully running, relevant measures will be made directly on the system. The outcomes will reflect the performances, speed and accuracy of the system supported by the WACNet.

The analysis of the results will lead to two different outcomes:

- A critical evaluation of the techniques used to fulfil the task
- A statement on the performances and suitability of the WACNet-based control network studied.

At this stage a conclusion regarding the validation of the WACNet framework for real-life applications will be made.

3.4.4 Project's Limitations.

The aims and objectives described above outline a complete and ready-to-use Home Ambient Intelligent system. Due to limited time, this project will only explore the foundations of this concept and discuss its feasibility.

Solutions to all the described hardware requirements will be provided, with the exception of the node miniaturization and integration.

Regarding the software development, algorithmic concepts will be explored for machine learning. In relevant cases, further improvements will be suggested to cope with any problem which may arise. The fine tuning and real-life application of these algorithms will be left for further research.

A simple Human to Machine Interface (HMI) using a PC computer will also be developed to send and receive messages to nodes on the network.

3.5 Project Design

This section will provide an overview of the elements and architecture designed to reach the goals. Each element introduced in this section will be thoroughly studied in the remainder of this thesis.

3.5.1 Hardware Drivers

The fundamental component of the WACNet is a node. The nodes used in this project will be composed of a processing unit (microcontroller) and an ad-hoc wireless communication system. The node can support multiple sensing and actuating peripherals. The microcontroller of a WACNet node will carry out various tasks, such as sensory data acquisition, data conditioning, control, local decision making, while constantly communicating.

Hardware drivers will be designed to easily access the functionalities provided by each hardware module. These drivers will facilitate the implementation of an application on the platform while minimizing the computation load and memory usage.

A bridging node will be designed to serve as interface between the WACNet and various types of networks. The processing unit will embed a TCP/IP stack, and provide sockets for connection to an Ethernet network.

3.5.2 Software Architecture

The high-level software can be broken down in two categories: data gathering management and ambient-intelligence modules.

A. Data Gathering Management

A data acquisition protocol will be designed to aggregate data, and algorithms will be developed to optimize data availability. The protocol is implemented using the API, and allows the transmission of recognizable data packets as well as interpretation of the messages received. This protocol is designed to query and gather sensory data in a way which supports node co-operation and maximizes data accessibility. A specific part of the data-acquisition software is designed to optimize the query/reply process. It reduces the amount of data flowing in the network by sharing the data already available.

B. Ambient Intelligent Modules

The ambient intelligent software will comprise of several types of agents. Each software agent will work towards a specific goal. Different agents which form part of the Home Ambient Intelligent System are described below.

C. Learning agents

The learning agents will be the fundamental component of the intelligent system. The learning agents are specialized in learning and prediction of different states of the system. One learning agent will be present in each node, to obtain high-granularity data. They use the API functions to acquire learning sets, which are processed in order to extract a set of rules. The rules created by the learning algorithm are used to produce forecasts. The data extracted by the learning agents is a critical input to the rest of the agents.

D. Self-organisation agents

The self-organization agents are in place to continuously enhance the network structure. The agents focus on cluster rearrangement and cohesion reinforcement. This is done by identifying and gathering nodes working towards a common goal.

The self-organisation agents determine the likelihood that two given nodes will have to collaborate. This is done through processing of the learning agents' outcomes. Based on a collaboration metric, the self-organisation agents consider various structures and modify the nodes' association. This groups functionally close nodes, thus providing them with strong communication links and limiting the average number of hops.

E. Policy agents

The policy agents apply the resource-consumption reduction rules to the network. Policy agents combine the output of the learning agents and a set of abstract policies, to take actions which reduce the consumption of resources. The policy agents are modular and adaptable. They interpret and implement the policies. The output is a strategy for the network to optimize a specific situation which matches a policy.

Policy agents are among the few non-distributed agents. Nevertheless, once a saving scheme has been determined, the optimization task is distributed from the agent to the relevant local nodes and carried out autonomously.

F. Human-Machine Interface (HMI)

The human-machine interface will let the user(s) remotely access the WACNet nodes. The network can be accessed and monitored from any computer connected to the WACNet via the bridging node. The bridging node serves many purposes, such as download server for the JAVA control Applet, and Ethernet/WACNet frame converter.

The control station's browser downloads a JAVA applet from the bridging node. This applet features an interface and embeds a protocol for remote control and monitoring of the entire network.

3.6 WACNets-Based Ambient Intelligent System

This section explores the characteristics of a Home Ambient Intelligent system supported by a WACNet based on an example. The system considered monitors and reduces the consumption of electrical energy and water.

A. Installation

The installation of the system consists of two stages. The first step consists of the installation of meters, sensors and actuators or controllers. Meters will monitor the consumption of the resources. They must be installed at the points where a significant amount of electricity or water is consumed (e.g. shower, refrigerator, television, lawn watering system, etc.). In addition, various sensors are installed in the building. They are used to determine what motivates the consumption, through constant monitoring of the users and the parameters of their environment. The sensors installed in the building could include presence detectors, thermostats, ambient light brightness sensors, etc. The house must also be fitted with controllers. These can be electronic switches, valves, or mechanical actuators for windows, blinds, etc.

The second stage of the installation consists of connecting all sensors, meters and actuators to the nearest WACNet Node. A WACNet node can accommodate up to 4 inputs.

B. Learning

Once the system is installed and powered, it automatically performs self-configuration. The nodes begin to share their local data. After a few days of processing, the learning module produces a set of rules which describes the system's consumption, triggers and parameters.

Some examples are described below

- "The electrical consumption of lights is heavily correlated with the state of a switch. It is also loosely related to the presence of individuals in the concerned room and the ambient light brightness"
- "While the TV set is never used between 1:00AM and 5:30 AM, it still consumes p kWh^{-1}".

C. Operation

The rules learnt are employed mostly for self-organization of the network and to minimize the consumption of resources. The self-organization module will improve the network configuration based on the knowledge of the related nodes. This minimizes the cost of communication and latencies between the nodes which are closely related.

To save resources, the application is programmed with generic policies and validated by the user. The policies describe very obvious and simple ways to save resources. A policy must identify a situation and how it can be improved.

Examples of such policies are:

- "Issue a warning if there is a sudden, unjustified or unusual rise in consumption",
- "Disable appliances at hours at which they are not used",
- "Delay the use of resource-consuming appliances to a moment when the resource will be available at a lower price"

These policies are interpreted by the application to match concrete cases. The application constantly analyzes the current situation, and attempts to recognize events or situations which are detailed in one of the policies as illustrated in Figure 3.3. Once a situation is recognized, a relevant set of commands is created to improve the situation. To execute the policy, the application makes queries to the relevant learning module to obtain the normal consumption pattern of the appliance. Through comparison with the current value, the application can immediately realize if the consumption is normal (i.e. that it matches the usual consumption behavior) or abnormal. If it is abnormal, the program takes action.

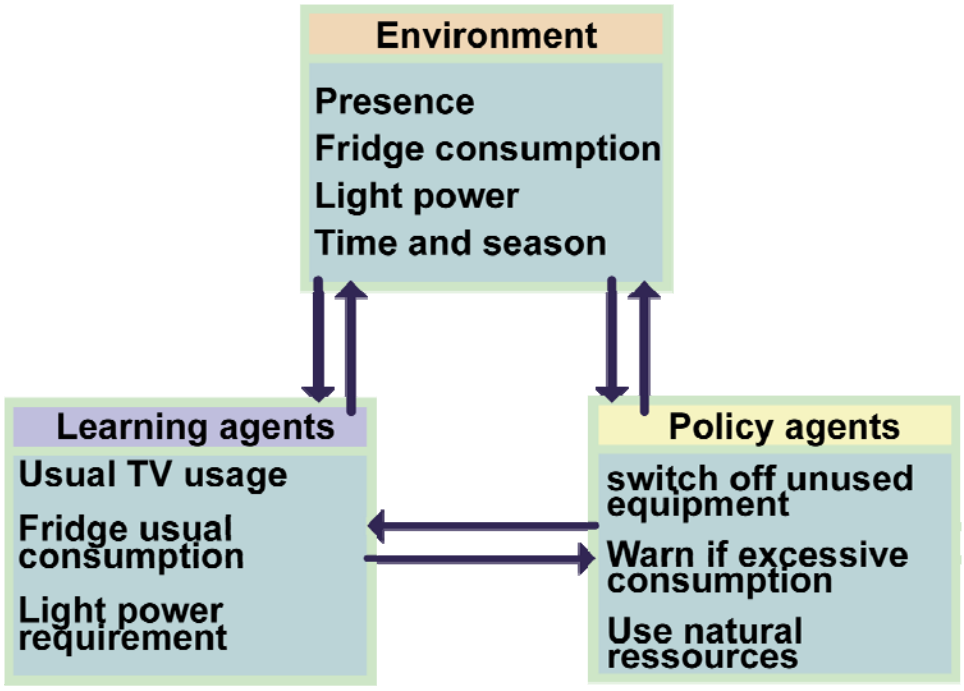


Figure 3.3: The cooperation of learning agents and policy agents.

D. Concrete Scenarios

To illustrate the first policy example in section C, let us consider that the refrigerator door is accidentally left open. This usually results in an excessive consumption of electrical energy by the appliance. This is because the refrigerator constantly tries to maintain a low temperature despite the "thermal leak" caused by the open door. This consumption will not match the usual consumption behavior known by the learning module. This will result in a procedure to improve the situation, by applying the solution described in the policy. The application predominantly attempts to reduce the consumption without interacting with the users. In this particular case, if the system has control over the appliance's door, it will close it. If it does not, it will simply alert the user. A similar process would take place if a water valve or tap is leaking, or a gas pipe ruptured.

Beyond simple observations, the system can decide to reschedule resource-consuming tasks to a time when they are available at a lower cost. It can also shut down appliances which consume energy while not being used.

E. Summary

The system can take into account large number of parameters to achieve the largest savings. For example, the system can use intelligent algorithms to optimally plan the use of resources acquired at no cost (solar panels, water tank, etc.). It can also take as parameters the varying cost of the resources depending on the time of the day or the season. The system can even be programmed to retrieve weather forecasts from the Internet to determine the likelihood of rain or limited lighting.

In conclusion, a Home Ambient Intelligent system dedicated to resource consumption reduction and running on a WACNet is a simple and low-cost solution to optimize the energy and water efficiency of a building. The Home Ambient Intelligent system evaluates many possible plans to improve the house efficiency and automatically carries out the best plans with minimal interaction with the users.

Chapter 4 Experimental Setup

4.1 Introduction

This chapter offers a comprehensive view of the experimental setup for this research project. A detailed analysis of the individual components of the system as well as their integration will be provided as well. The focus of the chapter will be initially on the hardware aspects of the experimental rig followed by the software modules. This will be organically evolving to a higher level including the global structure of the system.

4.2 Hardware Elements

In this section, the *nodes* which are the main constituent of the WACNet will be studied. Initially, the hardware components present in the node and their corresponding drivers will be described. The choices of standards, specifications, capacities and other necessary details of each individual element found in a node will be explored and justified. Once all the constituents of the node have been presented, the node itself will be described. In this section, the focus will be on the assembly of the individual elements including node layout, interconnection, power supply, etc. The description of the hardware will be followed by a presentation and critical analysis of the drivers developed for each element.

4.2.1 WACNet Nodes Hardware

A. Microcontrollers

The central component of a node is a microcontroller, which coordinates all the tasks for which the node is responsible. The nodes developed in this project use ATMEL ATMEGA 32 microprocessors, on an ERE development board [52]. The ATMEGA 32 has been chosen because it is a widely available, low cost and low

power. It runs at 16MHz, and holds 32Mb of RAM, 1024 bytes of EEPROM and 2048bytes of SRAM. The ATMEGA 32 operates with an supply voltage between 4.5 and 5.5V. This 8-bit microcontroller provides four bidirectional ports, which allows the connection of multiple sensing and actuating devices to a single node. The ATMEGA 32 also offers functionalities found on most microcontrollers, such as a built-in UART, timers and interrupt.

The ERE development board is mainly used to facilitate the project development, as the access to the microcontroller's pins is more convenient. The input/output port pins are wired to boxed header connectors, while the UART and power supply are made accessible through PCB connector blocks.

The ERE development boards provide all the equipment required to program the microcontroller. Software is supplied by the manufacturer to download the compiled file into the program memory.

B. Communication Modules

In order to communicate with the remaining nodes, each node requires a wireless communication module. A complete analysis of the different standards and technologies is provided in the literature review.

The ZigBee standard is a compelling choice since it is the only widely available solution dedicated to control networks. The ZigBee specifications match the application-specific demands, featuring high autonomy, and long-range and multi-hop transparent communication.

Z-Wave has also been considered, since it is dedicated to building control systems. However ZigBee appears to be a more open standard, which does not restrict the use of the WACNet platform in the future. The Z-Wave products are currently at a more advanced stage of development than the ZigBee-compatible products, but the analysis presented in the literature review demonstrates that the ZigBee standard aims for higher and more general objectives than Z-Wave.

The ZigBee standard can be applied to communicate either directly from the central microcontroller, or through an integrated peripheral communication module.

The first solution is the closest to the specifications of the WACNet. It involves the sole use of the node's microcontroller to manage the communications and ZigBee networks. This requires the microcontroller to perform routing, frame assembly, transmission, network organization, etc. This solution is realized by the integration of a ZigBee stack onto the microcontroller and by interfacing a transmitter to the microcontroller.

In this study, a ZigBee stack is not integrated to the node's microcontroller. Instead, a ZigBee integrated module solution is deployed to reduce the development time. ZigBee modules provide all the components required to handle signal transmission and reception, as well as network management. The modules are fitted with their own microcontroller, a transceiver and a communicating interface. The module's microcontroller is responsible for ZigBee frames creation, transceiver control, packet routing and network organization. This reduces the load on the node's microcontroller, as it is only required to read and write data to the module. The node's microcontroller communicates with the module through an RS-232 Link. The two microcontrollers use UART modules to exchange data. The RS-232 link's flow control cannot be used as the development boards do not implement this feature.

The ZigBee module solution provides the WACNet with a simple and transparent multi-hop means of communication. The major drawback is that there are two different microcontrollers in the node, which significantly increase the electrical power consumption. From a communication point of view, this solution creates inter-microcontroller communication latencies and also severely limits the data transfer rate between the microcontroller and the module. The ZigBee modules are an ideal solution to explore the feasibility of this concept, but are clearly not an optimal solution. The final version of the WACNet node should embed a ZigBee stack in the main microcontroller and be interfaced to a transceiver. This would require a more powerful and faster microcontroller, but would reduce the size and energy consumption of the node. To a certain extent, the chosen ZigBee module could have been used as a very basic WACNet node, as it can be programmed to perform some very simple tasks such as reading and automatically transmitting sensory values.

The ZigBee modules selected are the MaxStream XBee and XBee PRO chips. The characteristics obtained from the manufacturer's website [50] can be found in Table 4.1. The XBee modules are among the smallest chips and most advanced solutions available on the market at the time this research project began. They are widely used and regularly updated with new functions. One of the major drawbacks is that it is not currently possible to dynamically switch the behavior of a node from ZigBee router to the end device, and the ZigBee End Device nodes are not yet implemented in the firmware.

Table 4.1: XBee and XBee pro technical specifications.

Model	XBee	XBee PRO
Price (each)	18 USD	32 USD
Size	27x25x8mm	32x25x8mm
Voltage	2.8-3.4V	2.8-3.4V
Average Current	45-50mA	50 -200mA
Range indoor	30m	100m
Range outdoors	100m	1500m

C. Other Peripherals and Accessories

Each node is fitted with a two-line, 16 characters per line (16×2) Liquid Crystal Display (LCD). This peripheral is used to debug or monitor the node by displaying status messages.

Four nodes are fitted with an external half-wave articulated antenna.

D. Power Supply

The nodes are powered by a 9V DC regulated supply. This supply directly powers the XBee development board. A 3V DC supply is generated from the 9V supply by an integrated regulator, mounted on a heat dissipater. The microcontroller is powered by the output of the 3V voltage regulator.

E. Node Layout

The components of the WACNet node are mounted in a PVC box. The box protects the boards from accidental damage due to poor handling or falls, prevents the occurrence of faulty connection issues and also isolates the boards. The size and shape of the box was chosen to provide convenient access to all the components during the development stage. The dimensions of the PVC box are 113×198×60mm. On completion of the development stage, the two development boards could be integrated into one custom-made board to reduce the overall size. Figure 4.1 represents a simplified view of a WACNet node (the wires and some components of least importance have been omitted).

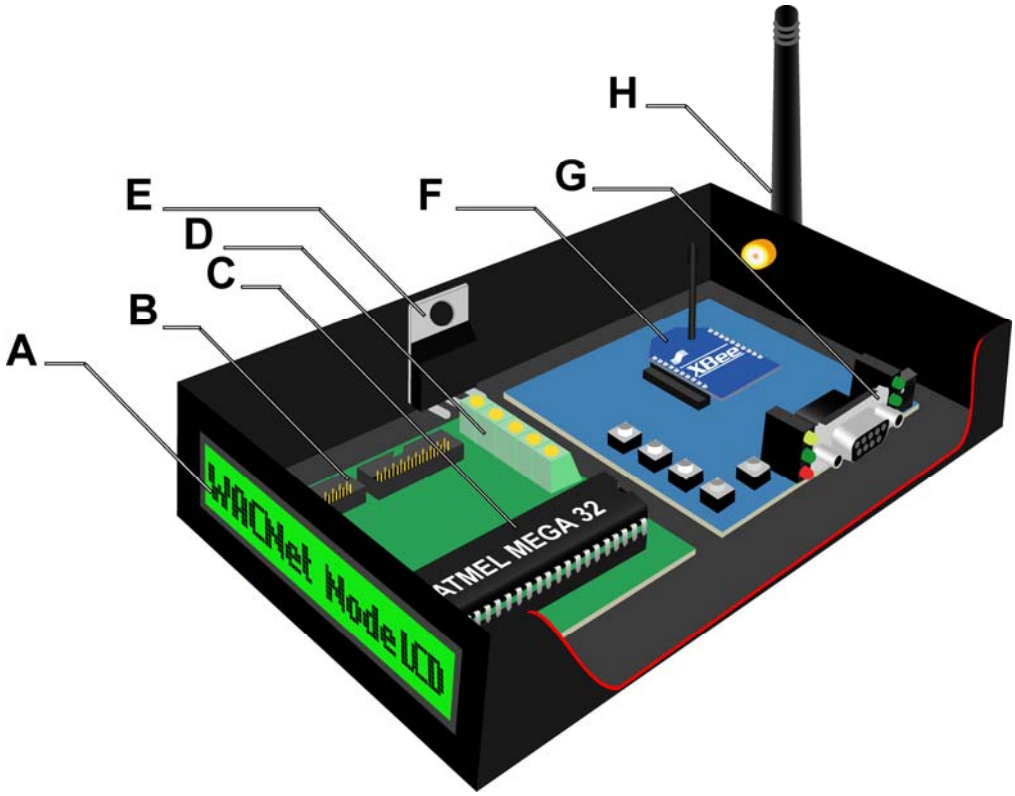


Figure 4.1: Layout of a WACNet node.

The microcontroller (**C**) and Xbee (**F**) boards are secured at the bottom of the box using spacers, screws and nuts. A window is provided for mounting the LCD display (**A**). The external antenna (**H**) is mounted on the side of the node. The sensors and

actuators are connected to the data port connectors of the microcontroller development board (**B**)

The microcontroller and ZigBee chips communicate via an RS-232 link. The microcontroller has three connectors for the Tx (Transmit), Rx (Receive) and Gnd (Ground) pins (**D**), which are soldered to a DB-9 male connector (not represented) inserted in the module's development board RS-232 female connector (**G**).

The two other connectors on (**D**) are connected to the 3V DC regulator's output (**E**). The communication module's 9V DC supply is directly connected to the socket on the right of (**G**).

F. Bridging Node

The bridging node requires specific functions such as LAN connectivity and a TCP/IP stack. A specific hardware platform has been chosen for this particular node. The DALLAS SC TINI 390 board forms the core of the bridging node.

The TINI 390 is an "off the shelf" solution featuring an all-in one and multi purpose SIMM-sized board. This board is small, low-cost, and comes with a complete software suite which greatly reduces development time. The TINI 390 is based on DALLAS SC DS80C390 microprocessor, interfaced with various chips and components. The board is mounted with external FLASH and SRAM, a TCP/IP stack chip, and various chips serving specific tasks (1-wire, Real Time Clock (RTC), etc.). The SIMM board is fitted on a DALLAS SC TINI development board, which provides all the connectors required (RJ45 for Ethernet, DB9 for serial communications).

Similar to other nodes, the bridging node uses an XBee module. This module is connected to the TINI board through RS-232 link, using DB-9 sockets.

The two development boards are mounted on a similar PVC box to any other node. The bridging node has no LCD display but is supplied with an external antenna for extended range. The XBee module is fastened to the bottom of the PVC box, while the TINI development board sits above the XBee board. The boards are attached to the PVC box using spacers and screws.

The TINI and ZigBee development boards require a 9V DC supply. Therefore the main supply to the box (9V) is distributed among development boards. The bridging node is intended to remain active at all times. The 9V power supply should be obtained from a stable and durable source of energy (i.e. a power outlet).

4.2.2 Sensors and Actuators

The WACNet is designed based on the IEEE 1451 smart sensor standard. IEEE 1451 compliant sensors provide a common communication interface which is independent of the type of network [43]. This type of sensor contains metadata such as the transducer name, manufacturer, status and parameters. This metadata can be read by the node to configure and calibrate the sensor.

The IEEE 1451 standard also defines a Smart senor unit, which consists of a Transducer (XDcr), which is connected to a Smart XDcr Interface Module (STIM), interfaced with a Network-CAPable application and node (NCAP). The WACNet is a combination of the IEEE 1451 smart sensor interface and the ZigBee communication standard. The WACNet nodes and the program they run are considered here as the NCAP.

An in depth-description of the IEEE 1451 application in WACNet was provided in the previous work [44].

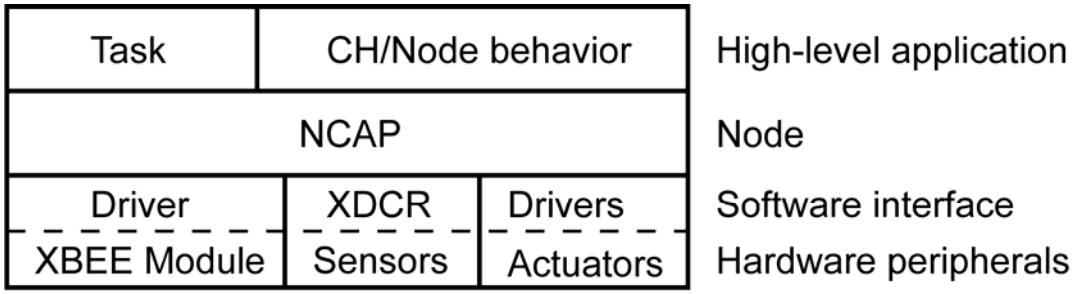


Figure 4.2: Layers of a WACNet Node.

Figure 4.2 illustrates different layers of a WACNet node. The Sensors and actuators are part of the hardware layer, and are controlled by an XDcr or a driver.

The sensors used for a Home Ambient Intelligent system can be grouped into three categories:

a) The sensors which measure the consumption of different resources, such as watt-meters, water and gas flow meters.
b) The sensors which measure various events related to the users, such as presence detectors, switches, user identification devices, etc.
c) The sensors which measure environmental variables, such as a light, temperature, rainfall sensor, etc.

The WACNet will mostly control typical applications for home automation systems. The control of resource consumption will be carried out by operating relays, automated switches and proportional controllers, regulating the power in appliances such as lights, heaters or electrical motors. Some possible applications are Heating, Ventilation, Air Conditioning (HVAC) systems control, mechanical actuators operating blinds, windows and doors, lights regulators, or garden watering systems. The system is also open to innovative solutions such as water and gas valves to prevent leaks, solar panels, etc.

The sensors and actuators are connected to one of the ports of the node. The ports offer a variety of configurations including serial or parallel (up to 8 pins) data transmission.

4.3 Hardware Peripherals' Software Drivers

A set of drivers are developed to form a layer interfacing the main application with the hardware modules. A distinct hardware driver is in place to abstract each hardware component. This allows the top-level programmer to access the whole range of functionalities with a set of high-level functions. The application programmer can use the peripherals without knowing the interfacing details. The driver handles all technical aspects such as initialization process, sequencing, waiting periods, escape characters and conversions, buffering, compatibility, etc.

This section will deal with the technical aspects of the drivers, and the high-level functions they implement on the interface.

4.3.1 Code Generation and Download

The ATMEL microcontroller code is created from a C-embedded source. The source code is compiled using the WIN-AVR compiler, which outputs a HEX file. The HEX file is downloaded with a manufacturer-specific cable and application. The cable provided uses the SPI link, which (unlike JTAG) does not allow *in situ* debugging.

The TINI board's operating system must be downloaded and configured prior to the usage of the board. This step is carried out by a specific software, using a serial connection. Once the TINI OS is set up, the microcontroller can run JAVA programs. The JAVA code is compiled using the conventional JAVAC compiler, and post-processed by a TINI-specific converter. The TINI code can be downloaded and launched through the serial link, but it is preferable to use the FTP server for code downloading and the TELNET console for remote operation.

4.3.2 XBee Module Driver

Communication is the fundamental purpose of a node. The XBee module driver is developed to manage reliable, fast and efficient communications. This driver operates on a layer set between the main program and the UART. It carries out all the transmission and reception routine tasks and manages the module. The driver is stable and robust, completely transparent, and is designed to make an efficient use of the microcontroller's time through interrupts.

The diagram in Figure 4.3 shows the general organization of the layers involved in communication. It shows that the XBee module driver layer bridges the main program and the microcontroller communication hardware (UART).

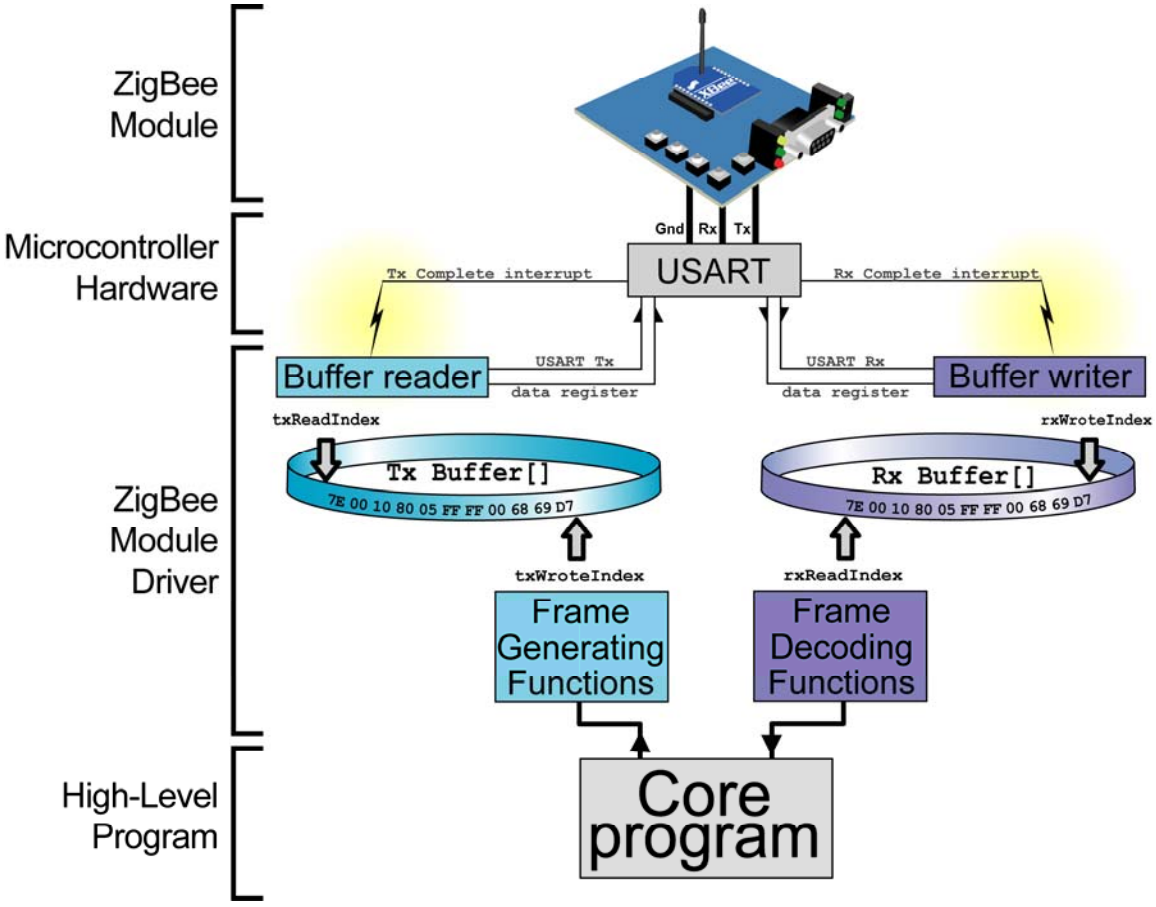


Figure 4.3: General architecture of the communication process, presented by layers.

A. XBee transmission driver

The transmission process is interruptible and burst resistant. Resistance to bursts of data is achieved through the use of a First-In First-Out (FIFO) circular buffer. One C function is implemented to execute each of the following XBee module commands:

- SendMessage is the routine called to send a message to another node. The MAC and Network address must be specified. The data is sent by passing a reference to an array of characters, with a specified length. Some implementation-specific details must also be specified when calling the routine (broadcast radius, disable the transmission of acknowledgement, disable the network address discovery process). The C function header is as follows:

```
sendMessage(uint8_t addressHigh[4], uint8_t
            addressLow[4],  uint8_t netAddress[2],
            uint8_t bcastRadius, uint8_t disableAck,
            uint8_t diasable_NetAddrDiscovery, uint8_t
            data[], int length );
```

- SendATCommand is a routine called to modify a variety of parameters on the ZigBee module. An AT command is defined by a command name (which consists of two characters) and optional parameters, passed as a numerical value. The complete list of AT commands can be found in the product datasheet available on the manufacturer's website. Some examples are soft reset, network reboot (used to restart the network formation process), set transmission power, etc. The C function header is as follows:

```
void sendATCmd(char ATCmd[2], uint8_t parameters[], int
               parametersLength);
```

The data encapsulation and transmission control is handled by the driver. When one of the transmission functions is called from the main program, the underlying driver proceeds as follows. A space in memory is dynamically allocated to hold a frame corresponding to the message. This frame is generated by the driver in conformity with the XBee modules' manual. The Frame encapsulates the message's data, along with a checksum and a header. The frame's header consists of a start delimiter (character 0x7E), a length field, a command identifier (character 0x10 for message transmission or character 0x08 for AT command) and a sequence number. If the frame is a request to send, the MAC and NET addresses are inserted between the header and the data. If the network address is unknown, the value 0xFFFF can be used to enable the network address discovery.

As soon as the frame is assembled, it is transferred to the transmission (Tx) buffer. As it is relocated to the Tx buffer, all special characters are escaped as prescribed in the XBee module manual. When the frame is written in the buffer, the temporary array is made free and the transmission begins immediately. The transmission is initiated by calling the UART transmit function on the first byte. The transmission driver then relies on interrupts to send the rest of the frame. UART transmit interrupts

allow the microcontroller to pursue the main application while a byte is being transferred through the RS-232 link. Each time a byte has been transferred, the main program is interrupted and the next byte is sent. This process is repeated while the buffer is not empty, i.e. the *wrote* index is not equal to the *read* index.

Receive interrupts occur at a higher level of priority than transmission interrupts. This means that the transmission process is systematically interrupted when data is received. The transmission process is resumed normally as soon as any higher-priority interrupt returns.

One of the major drawbacks of the ERE development board is that it does not implement the RS-232 standard data flow control. In the event of inability to send (e.g. high traffic on the communication media), the XBee module starts buffering the data. The CTS signal (Clear To Send) is invalidated 17 bytes before the XBee's Tx buffer overruns. As the module's CTS pin is not wired to the microprocessor, this signal will be disregarded, and data will continue to arrive. This situation can overflow the XBee module's Tx buffer, which results in the loss of data.

Figure 4.4 illustrate the sequence diagram of one of the two transmission functions.

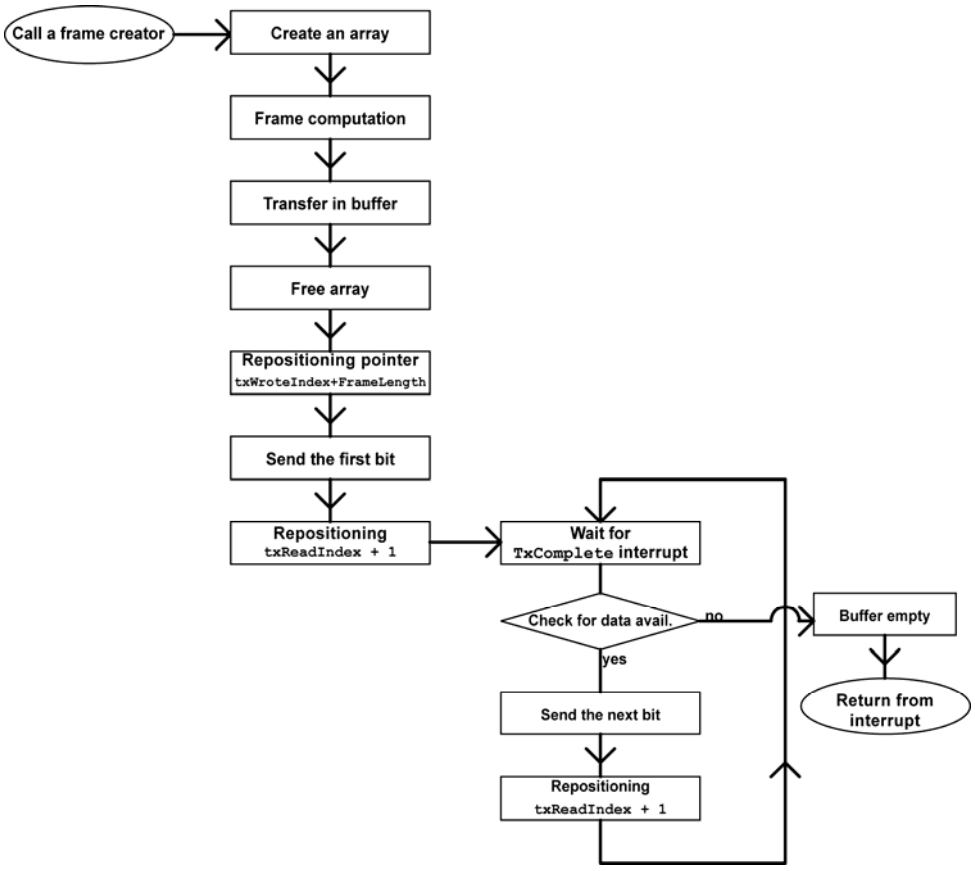


Figure 4.4: Sequence diagram of a transmission function.

B. XBee receive driver

The receive driver divides the receiving process in two parts: incoming data acquisition, and received data processing.

The acquisition of an incoming frame is totally automated and independent of the main program. The receive driver relies on interrupts to trigger immediate action whilst consuming the least amount of Central Processing Unit (CPU) time. When data is sent through the RS-232 link by the XBee module, the microcontroller automatically reads the signal (bit) as a background task. The received byte is written to the UART receive register. An interrupt is set to occur each time a byte of data is available in the UART register. This interrupt triggers a routine which copies the byte from the UART register to the driver's receive buffer. If necessary, the escape sequence is processed prior to the transfer in the receive buffer. Figure 4.5 shows a

sequence diagram of the data acquisition routine triggered by the UART byte receive interrupt.

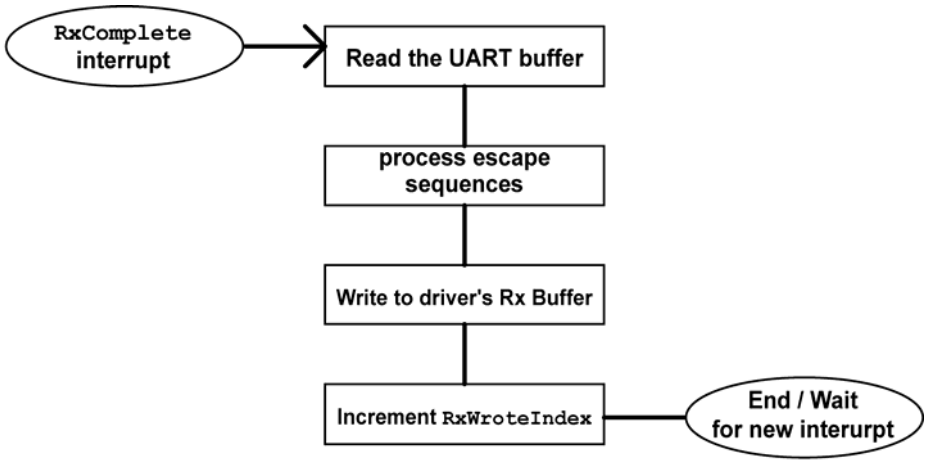


Figure 4.5: Sequence diagram of the UART Rx interrupt routine.

The receive interrupt has the highest priority level in this application. Although the receiving can be interrupted by user-defined interrupts, blocking this routine is not recommended. Indeed, a higher-level interrupt occurring while data is being received will prevent the microprocessor from reading the incoming data. If the UART register reading routine is interrupted for too long, data will be lost.

The incoming data written to the driver's Rx buffer is processed by calling the driver's `frameDetector()` function. This function is called from the main program at regular time intervals or when the microcontroller is idle. The frameDetector function searches the receive buffer for unprocessed frames. If a frame is fully acquired and positively verified, the frameDetector function will call the `handleFrame()` function. The handleFrame function reads the *API identifier* field of the first frame found in the buffer. The API identifier field contains a code which indicates the type of frame, for example: modem status message, receiving of an acknowledgement, or incoming message. The handleFrame function's task is to hand over the frame to the right purpose-built function, based on the API identifier. Once the frame is processed by its appropriate function, the program returns to the frameDetector function. Before exit, frameDetector verifies that there is no remaining

data in the buffer to process. The function repeats while there is data and only returns when the receive buffer is empty. Figure 4.6 shows a sequence diagram of the frameDetector and handleFrame functions.

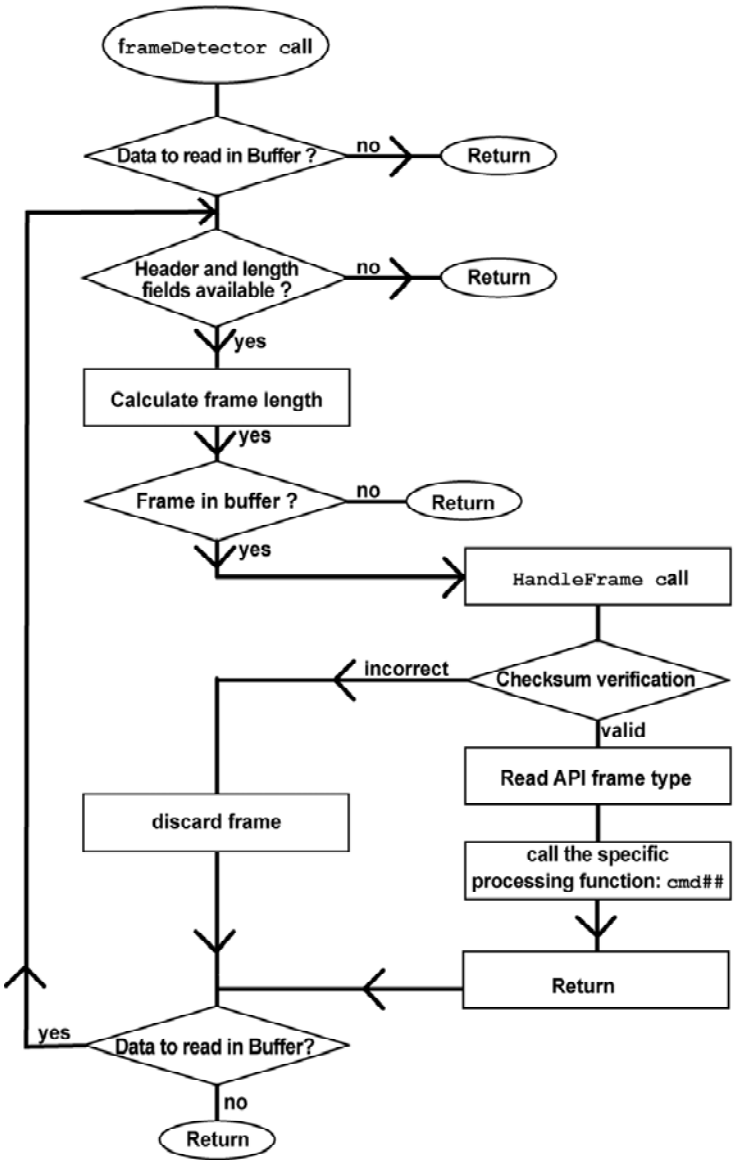


Figure 4.6: Sequence diagram of the FrameDetector and HandleFrame function.

The handleFrame function calls one of the following functions, depending on the API identifier read:

- `void cmd90(uint16_t frameStart, uint16_t);`

This function is called when a message is received from a node on the network.

- `void cmd88(uint16_t frameStart, uint16_t length);`
  This function is called when a reply from module commands is received (usually acknowledgments or failure reports).
- `void cmd8B(uint16_t frameStart, uint16_t length);`
  This function is called when a transmission status frame is received. Transmission status frames acknowledge a successful transmission, and contain information about the transmission, such as the number of retries. These frames also report transmission failure or abandonment and the causes.
- `void cmd8A(uint16_t frameStart,uint16_t length);`
  This function processes the node status messages. These frames are spontaneously sent when the ZigBee node changes its status on the network (e.g. association, loss of association, restart, watchdog timer reset, etc.)

Every specific processing function assigns pointers to each relevant field of the received frame. These pointers have an explicit name, corresponding to the field they point to. They refer directly to a location in the buffer. This provides a simple way for the top-level programmer to access the data encapsulated in the frame. The programmer decides to store the data which will be used in the future into a specific area of the memory. This must be done to prevent new incoming frames from overwriting the buffer area where the current frame is stored.

The body of each processing function contains no instructions and is to be completed by the top-level programmer. The desired behaviors corresponding to the specific messages received are coded in the unit.

Data processing is the only part of the receive process that must be called from the main program. As this task must be carried out with a low priority, it cannot be called from any interrupt routine. If the processing function is called from the receive routine, data processing becomes part of the interrupt routine. This would result in the entire data processing task being performed at a high priority level, thus blocking the main program and all the transmissions. For identical reasons, the processing of received data can not be called by a *timer overflow* interrupt. This is because the

timer overflow interrupts are set with the highest level of priorities. The only solution is to call the frameDetector routine from the main program, as often as possible.

4.3.3 Other Peripherals' Drivers

Drivers have been developed to abstract various tasks, such as sensors, actuators (Input/Output) and display. This section describes the main aspects of these drivers

A. Liquid Crystal Display (LCD)

Each node is provided with a two-line, 16 characters per line (16×2) LCD which displays the steps of the boot up process, signals node errors or failure, and more generally serves any application-specific task. The LCD driver has been developed in compliance with the microprocessor's development board specifications. The wiring of the development board makes it impossible to query the state of the LCD controller and to asses its readiness. Therefore the only way to communicate successfully with the LCD controller is to wait for a specified time after each write operation. As the delays are multiple and small, they are executed using loop delays. The application programmer should bear in mind that LCD display is a CPU time-consuming activity.

B. Sensors and actuators (inputs and outputs)

The node's microcontroller has three available 8-bit ports. An additional port can be made available if the LCD display is not used. These bidirectional ports can be used to read digital sensory data or to control an actuator. An interface is developed to initialize these ports and perform reads and writes on demand. Drivers for specific sensors and actuators can easily be added by the application programmer. For the Home Ambient Intelligent system, a module has been developed to carry out the acquisition and conditioning of sensory data required for the experiments described in the following chapters.

4.3.4 Bridging Node Drivers

The TINI board which forms the core of the bridging node is provided with an operating system software, and a JAVA Virtual Machine (JVM). These two pieces of

software provide drivers for all functionalities of the TINI board. This includes direct access to the time and date through the Real-Time-Clock, and a complete interface for the TCP-IP stack and the UART.

Figure 4.7 shows different layers forming a Bridging Node.

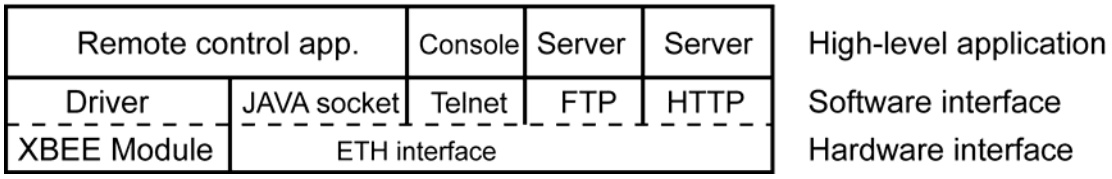


Figure 4.7: Layers of a Bridging Node.

The bridging node has a driver for each communication interface including the ZigBee module and the Ethernet link. These drivers are implemented to ease the transmission of data from the high-level programs.

A. ZigBee driver

The ZigBee driver performs the same tasks as the one on common nodes. The main difference is that there is no need to create buffers. Indeed, the standard JAVA classes `InputStream` and `OutputStream` which interface the serial port are implemented with a buffer. The buffer size can be increased by the OS to prevent overflow.

The `XBeeTransmission` class, which is responsible for message transmission to the module, has two methods. One is used to send configuration messages to the module (AT commands) and the other, to send messages to other nodes on the network. These functions instantiate and process frame objects, which are passed to the driver. The driver uses the `InputStream` JAVA class. This class, once instantiated and initialized, handles all aspects of the serial communication. Therefore the driver simply writes the data from the Frame object into the `InputStream`.

The `XBeeReceive` class implements a set of functions dedicated to processing the information obtained from the ZigBee module. The general structure is similar to

the one used in common nodes. When a frame is available from the JAVA `OutputStream`, a specific function calls the appropriate method to verify and deal with the received frame. A variable with an explicit name is assigned to each data field of the received frame. The body of all the frame-processing methods is left empty, for the application programmer to configure the desired behaviors.

B. Ethernet Driver

The driver which manages the Ethernet communications consists of very few elements, since most of the transmission control is carried out by standard JAVA classes. The `MonitoringServer` class opens a server socket and waits for connections from monitoring station. Once a connection is opened, two concurrent threads are instantiated for receive and transmission of data. `ServerReceiveThread` deals with data coming from the monitoring station while `ServerTransmitThread` sends messages to the monitoring stations. The threads can be programmed by the application programmer, to render the desired behaviors.

4.4 Software Modules

The Home Ambient Intelligent system is an organic application. The system is composed of multiple software modules called agents. Different types of agents are designed, each of them carry out a very specific and distinct task. Through intensive collaboration, the network of agents achieves the global aim of the application. This section describes all the agents which are part of this application.

4.4.1 Learning Agents

The learning agents are based on a fuzzy-logic learning algorithm. Several optimization modules are built around the fuzzy-logic algorithm to refine its output.

Each node hosts one learning agent. The learning algorithm in each node gathers data from the entire network, and produces a set of rules representing different states of the network. Various optimization modules refine these rules.

A complete description of the learning agents can be found in Chapter 5 of this thesis.

4.4.2 Self-Organization Agents

The self organization agents are based on a specifically developed optimization algorithm. Each node hosts an agent of this type, which focuses on optimizing the position of its node in the network. The network optimization process is totally distributed and decentralized.

The learning agents provide the initial data which is used to determine the degree of correlation between two nodes. The agent uses this data in combination with an optimization algorithm to improve the cohesion of the network. The optimization agents also handle the transfer of the node from a cluster to another, via a specific protocol.

4.4.3 Policy Agents

The policy agents are the only centralized agents of this architecture. The core of this agent is an interpretative decision-making module. This module constantly analyses the current situation and the model (from the learning algorithms) while simultaneously examining a policy table to detect situations which match one of the policies. The policy agent then uses a specific module which converts the policy into an action strategy. Orders implementing this strategy are transmitted to the relevant nodes.

The development of the policy agent will be left for future studies of the concept.

4.4.4 Supervision and User Control

Although the Home Ambient Intelligent system is totally autonomous, a user interface is provided for troubleshooting, or any other exceptional situation requiring human interference. The bridging node is used to create a link between the WACNet and the LAN network to which the control interface terminal is connected. The remote control terminal can be any form of a computer running a JAVA Virtual Machine and capable of communication with the server running on the bridging node.

To access the control network, the user must simply establish a connection to a HTTP server, with an Internet browser. This HTTP server runs on the bridging node. A JAVA applet is downloaded from the server and launched by the browser. As soon as the applet is started, it opens a bidirectional link to the Bridging node. This link is used to transmit the data between the WACNet and the interface. The remote-control applet presents a control interface to the user. Custom interfaces can be developed to meet any application-specific requirement.

To perform remote control, two approaches can be considered:

- Main program on the bridging node: In this case, the Java applet is remotely controlled by the bridging node. The applet simply displays messages coming from the node, and sends the user's instructions. The XBee frames are created and processed by the bridging node.
- Direct approach: The applet is not only a machine-to-user interface, but also manages the communications, frame generation and control processes and remote access application. In this configuration, the bridging node acts like a pipe, directly forwarding the XBee frames created by the applet, to the WACNet and *vice versa*.

In this study, the direct approach has been chosen. It is envisaged that a computer powerful enough to display a JAVA interface will have adequate resources to run the control application. This solution reduces the load on the bridging node, which already runs a JAVA-based driver and an HTTP server.

Figure 4.8 illustrates the remote control and monitoring infrastructure. The bridging node (TINI BOARD), on the top-right, runs two applications; an HTTP server, and a driver in JAVA. The HTTP server is used to download the applet file. The driver acts as a pipe between the two networks. The InputStream of one interface is connected to the OutputStream of the opposing interface. The communication layer is ready for the development of an application on the bridging node. Currently the main program only logs the frames which transit through the bridging node.

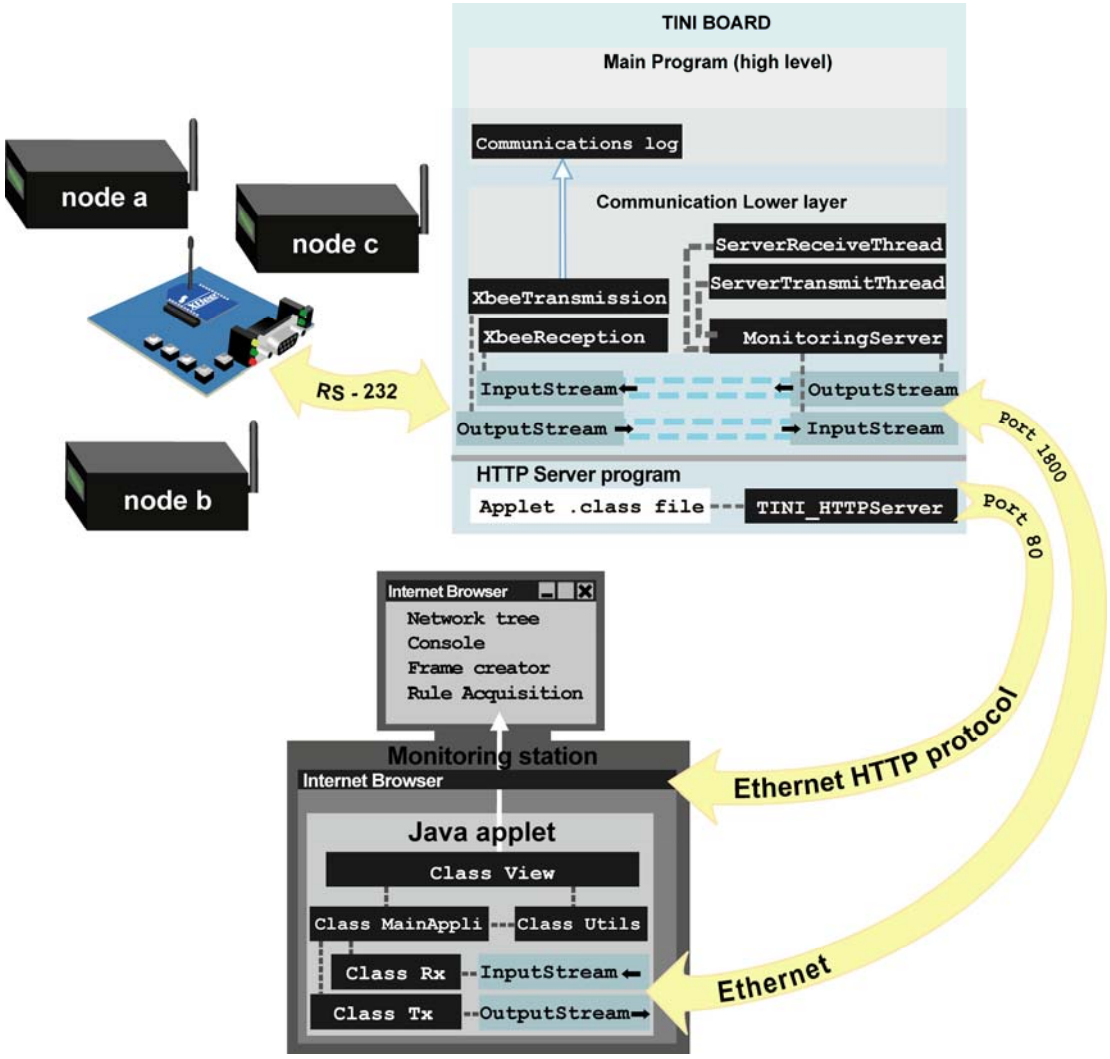


Figure 4.8: General structure of the remote control and monitoring system.

4.5 WACNet Organization for Home Ambient Intelligence

The last section of this chapter provides a high-level description (architecture and general organization) of the experimental setup. It presents how the nodes and software elements listed previously can be integrated to form a coherent solution. The choices made regarding global organization will be justified in relation to the criteria imposed by the application.

The general design of WACNet for a Home Ambient Intelligent system is presented. For each configuration, the constraints imposed by the application are described, and then the deployed solution is discussed.

4.5.1 Node Communication Considerations

This section focuses on the unique communication characteristics of the Home Ambient Intelligent system application. The constraints imposed by this application are presented, and solutions are proposed.

A. Requirements

The learning agents gathering sensory information generates the most significant network load. The rest of the communication will be a result of agent collaboration.

Each time a learning agent issues a query to obtain new learning data, the whole network replies. This allows the learning algorithm to build a comprehensive knowledge of the state of the system. It is expected that the network will be functioning under conditions with high bursts of traffic. When a node issues a data query, it will involve the simultaneous transmission of requests to every node on the network and immediately followed by a reply from the nodes queried.

For the learning to be accurate, the query process must be done with minimal latency. The time between the reading of the first sensor and the last one, as well as the overall reply delay must be kept minimal. When considering a large network, the synchronized management of such a task is a major challenge. The foremost difficulty comes from the fact that each burst of network traffic will inevitably create congestion and add delays to the reply.

The rest of the time, the network will be near idle. Indeed, the communications (other than learning data queries) will be much less frequent and use less structured paths. These communications will mainly consist of agent collaboration, dialogs between nodes wishing to change their position in the network, or communication between the monitoring station and the WACNet.

B. Proposed Solution

To obtain the best results with a high burst network, an appropriate architecture must be developed to minimize the use of a single path by many nodes. Using the whole range of links offered by the network significantly reduces the emergence of bottlenecks.

In this case, the formation of a bottleneck at the requesting node is almost inevitable. A distributed method for network query appears as the best solution to minimize congestion. The requesting node will not handle the full query and reply process. A node will seek the assistance of other nodes, amongst which the task will be allocated. These nodes also condense the information to reduce the size of the transmitted data. The network topology chosen for the WACNet will have to be compatible with task distribution. A study of the network topology is done in the next section.

4.5.2 Network Topology Considerations

The WACNet nodes can be set up with different levels of hierarchy, and the communications can be restricted between certain nodes. This configuration of the nodes in the network is called *network topology*. This section begins with a list of compulsory requirements for the network topology. An introduction to three standard topologies relevant to this application (star, tree and mesh) is provided and the proposed topology is presented.

A. Requirements

The chosen network topology must:

a) Operate with very few levels of hierarchy, ideally, none.
b) Support network self-organization and dynamic reconfiguration.
c) Handle the specific type of communications (one-to-all, all-to-one) generated by the learning process.
d) Facilitate task and query distribution.

B. Star Topology

Star-topology networks consist of various nodes connected to a central server, as shown in Figure 4.9. A star network is a centralized architecture, which is ideal for networks in which many nodes retrieve data from a central point. The main drawback is that it requires a powerful server to deal with the whole network of nodes simultaneously.

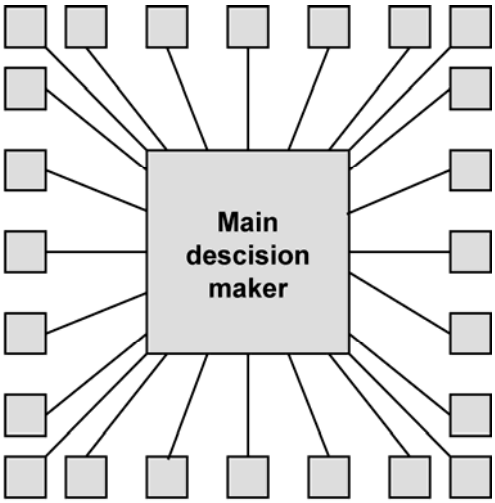


Figure 4.9: Example of star topology.

Considering the WACNet infrastructure, a star topology implementation is impossible for many reasons. First, it contradicts the fundamental principles of ad-hoc networking as it is highly structured and heavily centralized. Second, as the nodes of a WACNet all have a restricted computation power, a concept using only one of them to route and store the data exchanged on the entire control network is not feasible.

C. Tree Topology

The tree topology is an evolution of the star topology, with an increased level of hierarchy and distribution. An example is illustrated in Figure 4.10. Instead of using only one central server for the network, a central server communicates only to the head of each branch of the sub-network. In turn, the heads of each branch only communicates with nodes which are elected head of a sub-division. This topology favors vertical communication as master/slave dialog is faster than horizontal node-to-node communication. Horizontal communication requires an increasing number of hops as the communication occurs at the bottom of the hierarchy tree, especially for nodes which are on different branches.

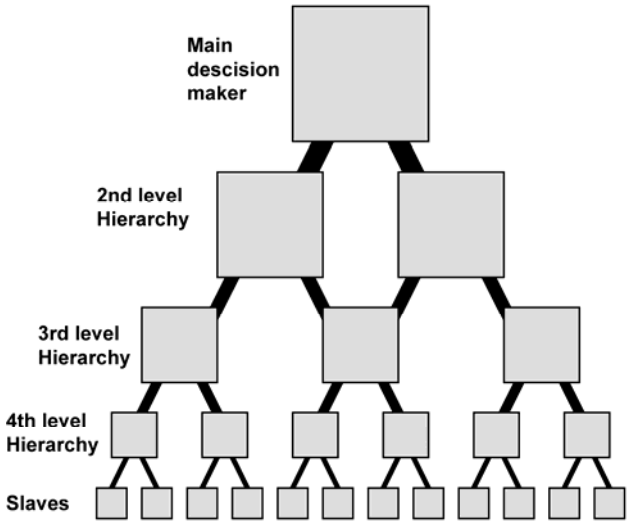


Figure 4.10: Typical tree topology control network.

This configuration is commonly used for control networks. It is particularly suitable for task distribution, but not decentralization. Indeed, a strict hierarchy exists: although the root cluster discharges its duties to lower levels of hierarchy, it is still in control of the whole process. This structure is also used for network self-organization and distributed address allocation, where the coordinator is responsible for the first *n* nodes to integrate the network. These nodes will then each handle a pool of *m* new nodes, etc.

This topology has one major shortcoming that restricts its use for WACNets. It is hierarchical and has some degree of centralization. If one of the nodes at higher levels of hierarchy fails, the whole sub-tree of nodes which depends on it is blocked.

D. Mesh Topology

Mesh networks are the most suited structure for ad-hoc networks. Mesh topology networks use all the existing links between the nodes to transmit messages with minimal hops, as shown in Figure 4.11. This topology is highly flexible due to its total lack of hierarchy. This makes it an ideal topology for dynamic self-organization. It is also robust and resistant to node or link failure, as it can establish an alternative route to bypass a faulty link/node.

Mesh topology is the topology of the Internet, at a global scale. The internet is composed of several Autonomous Systems (AS) with no central administration for routing. The Internet is composed of routers in each AS which independently determine the best path through the network.

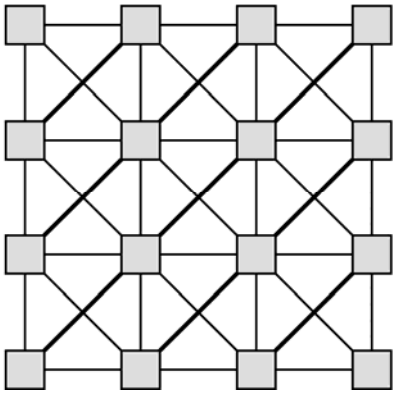


Figure 4.11: Example of mesh network.

In spite of all its advantages, mesh networks present a potential handicap. This topology attempts to involve every node in the communication process. The WACNet platform comprises of nodes which run on a very limited supply of power. As transmission consumes energy, those involving "weak" nodes should be strictly restricted to essential communications. For the considered application, it is preferable to use the nodes with a stable supply of energy as a network backbone.

E. Chosen Topology

The chosen topology for the network will be a hybrid of two of mesh and star-topology clusters as illustrated in Figure 4.12.

The WACNet is formed of groups of low-power nodes gathered around one node with the most reliable supply of energy. These groups form what is referred to as a *cluster*. In the cluster, the node with the most stable supply of energy becomes the *Cluster Head*. The cluster is a sub-network with a star topology. All the Cluster Heads will form an ad-hoc mesh network, which will become the backbone of the control network. Each cluster head is responsible for providing multi-hop bidirectional communication to the remaining *Cluster Nodes* in its cluster.

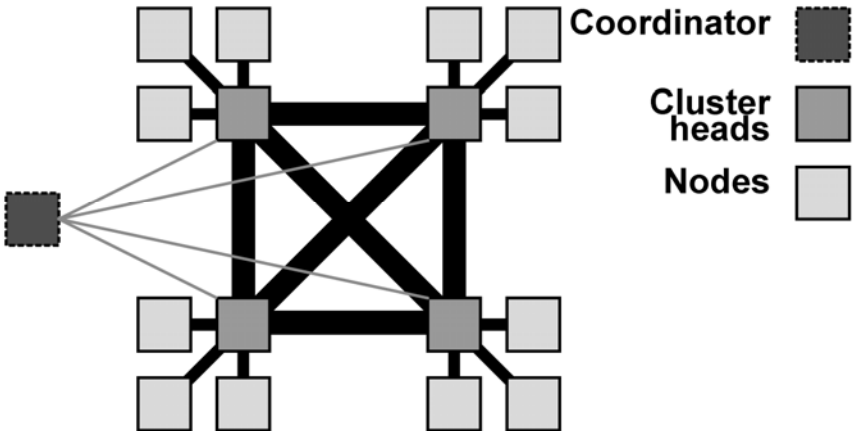


Figure 4.12: WACNet topology: mesh of clusters (including a coordinator).

This configuration requires very few levels of hierarchy and allows every node to communicate with minimal hops, independent from its rank. This architecture takes into account the autonomy of the nodes, preventing nodes with restricted supply of energy from taking part in the ad-hoc network. The clustered structure supports the distribution of the task by providing a pre-defined subdivision scheme for the network (clusters). The cluster heads can coordinate their cluster nodes to successfully manage task distribution among the existing clusters.

The network of clusters can only reach its maximum efficiency if the clusters are formed with functionally similar nodes. Indeed, the network topology is most efficient if the largest part of the communications remain in a single cluster. This means the network must be organized based on the task of each node.

For this specific application, cluster heads will play a central role in task distribution. To prevent weak nodes from having to manage the entire learning data query process, the relevant cluster head will distribute the task to all the other cluster heads on the network. In a parallel fashion, each cluster head will poll their respective clusters and collect the data. The sensory information of the whole cluster is collected and condensed by the cluster head. Once the data is acquired, a reply is issued to the source cluster head.

The cluster heads also contribute to minimizing the network traffic load by joining similar requests. The cluster head, managing the data acquisition of its cluster, will

only issue a request if the previous similar request has finished. This contributes to the reduction of bottlenecks and network congestion, while reducing the request reply time.

An example of WACNet organization is shown in the Figure 4.13

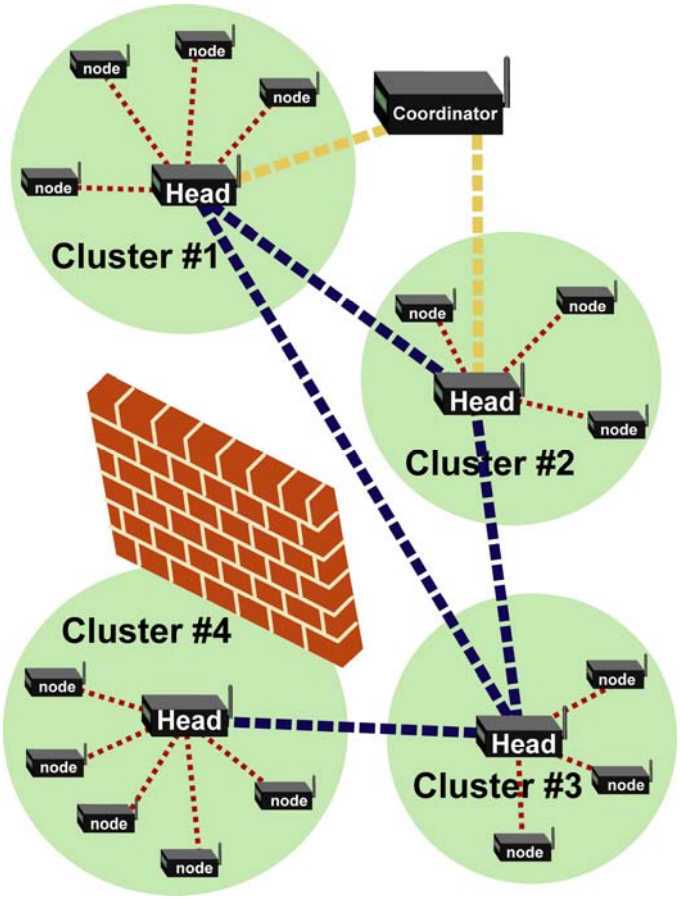


Figure 4.13 Example of WACNet configuration.

4.5.3 Network Discovery and Data Acquisition Protocol

The nodes are organized in a structured network and have the capacity to transmit messages. A protocol for communications is created for the nodes to recognize different types of messages. The communication protocol defines the sequence and contents of the messages which must be exchanged to carry out various queries. This section will present the protocols developed for network discovery and cluster formation, learning set queries and learning agent queries.

A. Node Characteristics Retrieval Protocol

The protocol defines a set of messages and a sequence of transmission to obtain the specifications of a node. This protocol is used at every network boot-up, or when a node changes its cluster. This procedure allows the cluster head to discover the characteristics of its surrounding nodes. This configuration data received from a node will be essential to read the frame correctly during the learning process.

- P query: The cluster issues a P message (P for "**P**resentation") to a node, to get a list of characteristics of this node.
- M frame: A node receiving a P query must reply with an M frame (M for "this is **M**e"). In the M frame, the node declares its number of sensors attached, and details about the local data conditioning (Fuzzy sets, see chapter 5).

The first part of Figure 4.14 illustrates the node discovery process.

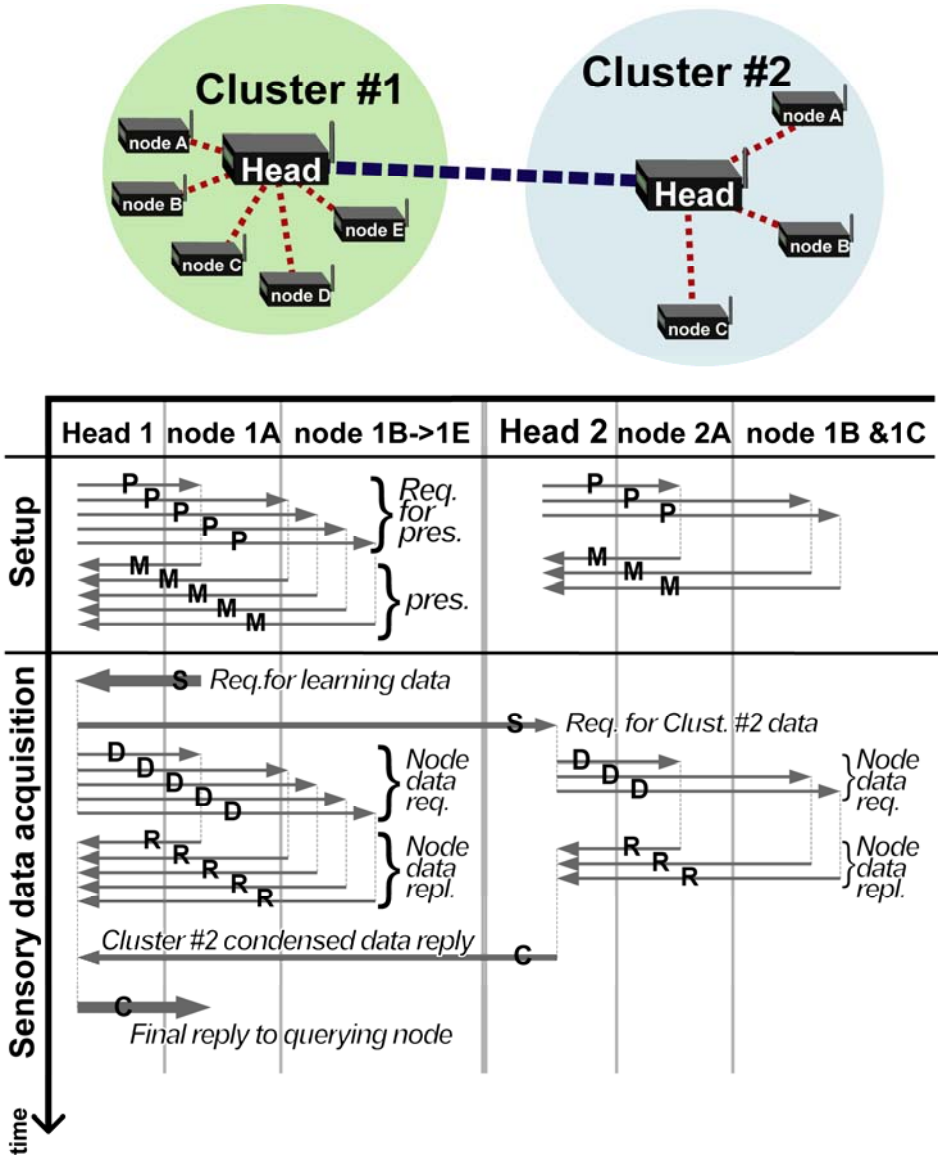


Figure 4.14: Protocol diagram for sensory data acquisition. The network considered is constituted of two clusters.

B. Sensory Data Query Acquisition Protocol

The query process involves transmission and reception over the entire network which overloads the capacity of a single node. Consequently, the sensory data collection process is distributed among the cluster heads. This reduces the polling delay and the load on the requesting node. Using the cluster heads as relays and coordinators offers many advantages. The nodes do not need to know all the network

addresses of every node on the network. The energy of the node and the overall time spent on the request are significantly minimized. Using the Cluster Heads to coordinate the acquisition of data also increases the availability of the data, by concentrating the information of the cluster into one node.

A specific protocol is implemented to distribute the learning data queries. The second part of the network communication diagram in Figure 4.13 illustrates this process. It portrays the sequence of messages transmitted to fulfill a sensory data query. Each type of message is detailed in the following list.

- S query: An S query sent to a Cluster Head is a request to provide a new learning data set. The S query is processed in two different ways, depending on the type of node which has issued the query including a neighboring Cluster Head, or a node within the cluster. An S query exchanged between two Cluster Heads is an inter-cluster request, whereas an S query originating from within the cluster is a full request.

  If the S request is an inter-cluster request, only the sensory data from the receiving cluster head is replied. If the requesting node is within the cluster, a full learning set is obtained from the entire network. Every time an S query is received, the Cluster Head immediately polls the whole cluster to acquire all the local sensory data. If the request is sent from within the cluster, a series of inter-cluster S requests are also issued to all the neighboring clusters.

  In Figure 4.14, the first S request (thick arrow from Node1A to CH#1) is a full request, as the query is sent to the Cluster Head #1, from a node within this cluster. This full request triggers one inter-cluster request (S, long thin arrow from Cluster Head #1 to Cluster Head #2). This will retrieve for CH#1 the data gathered from cluster #2 by its cluster head. Both of the S requests also initiate a local data poll (series of D queries).

- D query: When a Cluster Head receives an S query, it sends a D query to each node which is a member of the cluster. D queries are interpreted by a node as a request to acquire a process and send the sensory data (R frames).

- R frame: An R frame is the reply to a D query. The R frame is formed as follows. It begins with the 'R' identifier, followed by a payload field containing the node identifier, sensor identifier and the values for each sensor, as declared in the M-frame during cluster formation.
- C frame: a C frame is the final reply to an S query. Like the S queries, there are two different sorts of C frames. If it is a reply to a Cluster head (inter-cluster query), the frame will only contain the data harvested in the cluster. In the example given in Figure 4.14, it is the long thin 'C' arrow from CH#2 to CH#1. If the C frame is intended for the node which initiated the process, it contains the local data from the cluster, and also the data of all the surrounding clusters (last arrow on Figure 4.14, from CH#1 to Node 1A).

Distributing the polling process among the cluster heads allows parallel processing. This allows simultaneous processing of the query in each cluster, instead of one cluster at a time.

The diagram in Figure 4.15 illustrates the concept of task distribution and parallel processing to reduce the reply time.

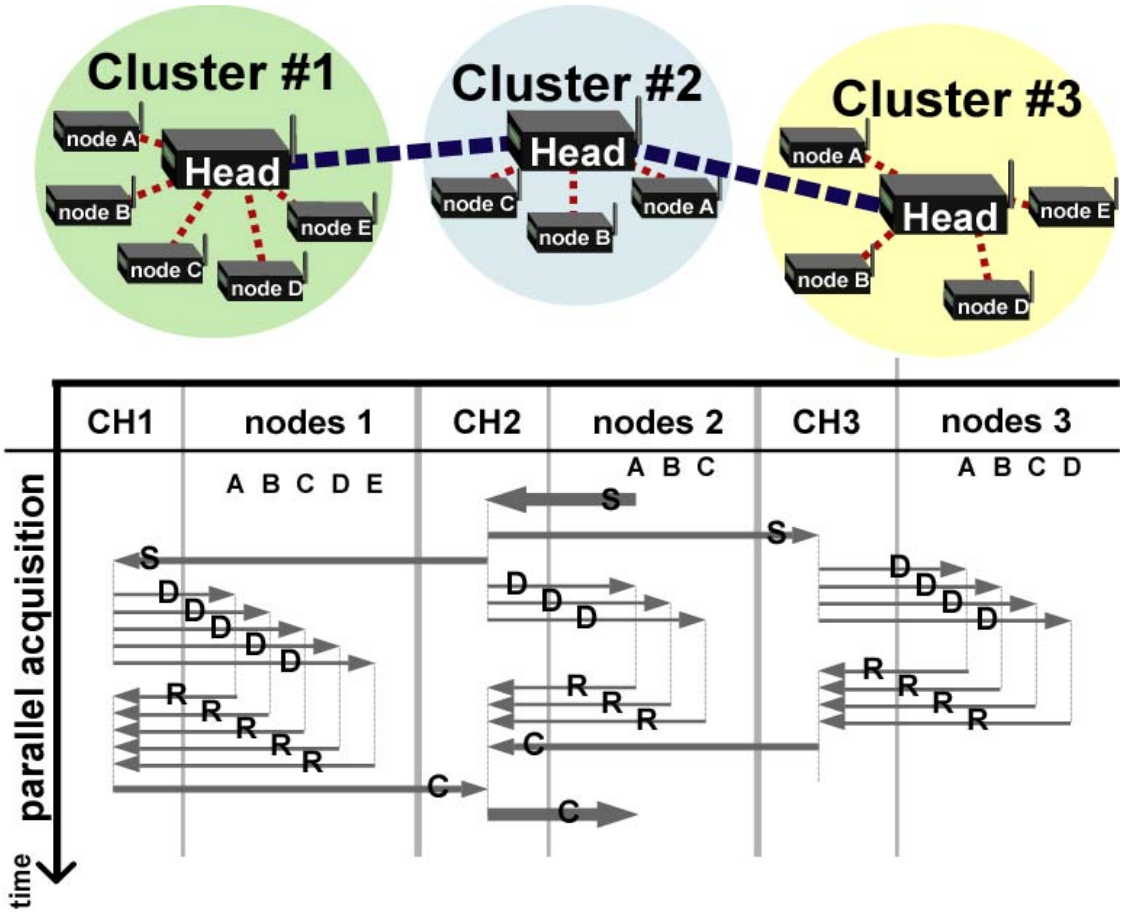


Figure 4.15 Parallel and localized data gathering on a three-cluster network.

This diagram demonstrates that the data acquisition of clusters #1, #2 and #3 is done simultaneously. Immediately after the cluster heads has received the S queries, the polling of the cluster begins. It also shows that the requesting node only sends and receives one message to acquire data from the entire network. The load is equally spread over the Cluster Heads. This prevents the formation of bottlenecks.

This technique reduces the reply time, but requires more messages than centralized polling. This is because Cluster Heads have to relay the information. While polling generates more messages than a direct "one-to-all" method, the Cluster Heads can reduce the network load by joining identical queries. Let us consider a specific type of query and refer to it as *Q*. Whenever a query *Q* reaches a cluster head, it systematically verifies that an identical *Q* query is not currently being processed. If

not, the query is initiated, and a lock is placed on this query. This lock prevents multiple identical requests from being processed at the same time. All identical queries are joined and placed in a specific queue. When the initial Q request finishes, all the nodes in the queue are replied to, and the lock is removed.

This technique prevents the accumulation of similar queries, and therefore decreases the node congestion and reduces the average reply latency. This system is most efficient when requests are made frequent enough to keep the queue full. To illustrate this phenomenon, let us compare accumulative and assimilative request processing. A situation where 5 similar queries arrive every t seconds is considered. One query takes $5t$ to be processed.

– Accumulative processing.

The Cluster Head processes the five identical requests independently, as if they were different. The Cluster Head must wait for a query to be completed, before a new one can be initiated. In this example, the five queries are replied with the following latencies: $5t$ for the first one, then $9t$, $13t$, $17t$ and finally $21t$ for the fifth query. The average reply time for these queries is $14.4t$. The average latency increases as the requests accumulate.

Figure 4.16 illustrates the example above. At t=0, 1, 2, 3 and 4, S queries are issued by each node in the cluster. The Cluster Head immediately begins with the first request, which is answered at t=5. The second query (received at t=1) is initiated at t=5 and returns at t=10, etc.

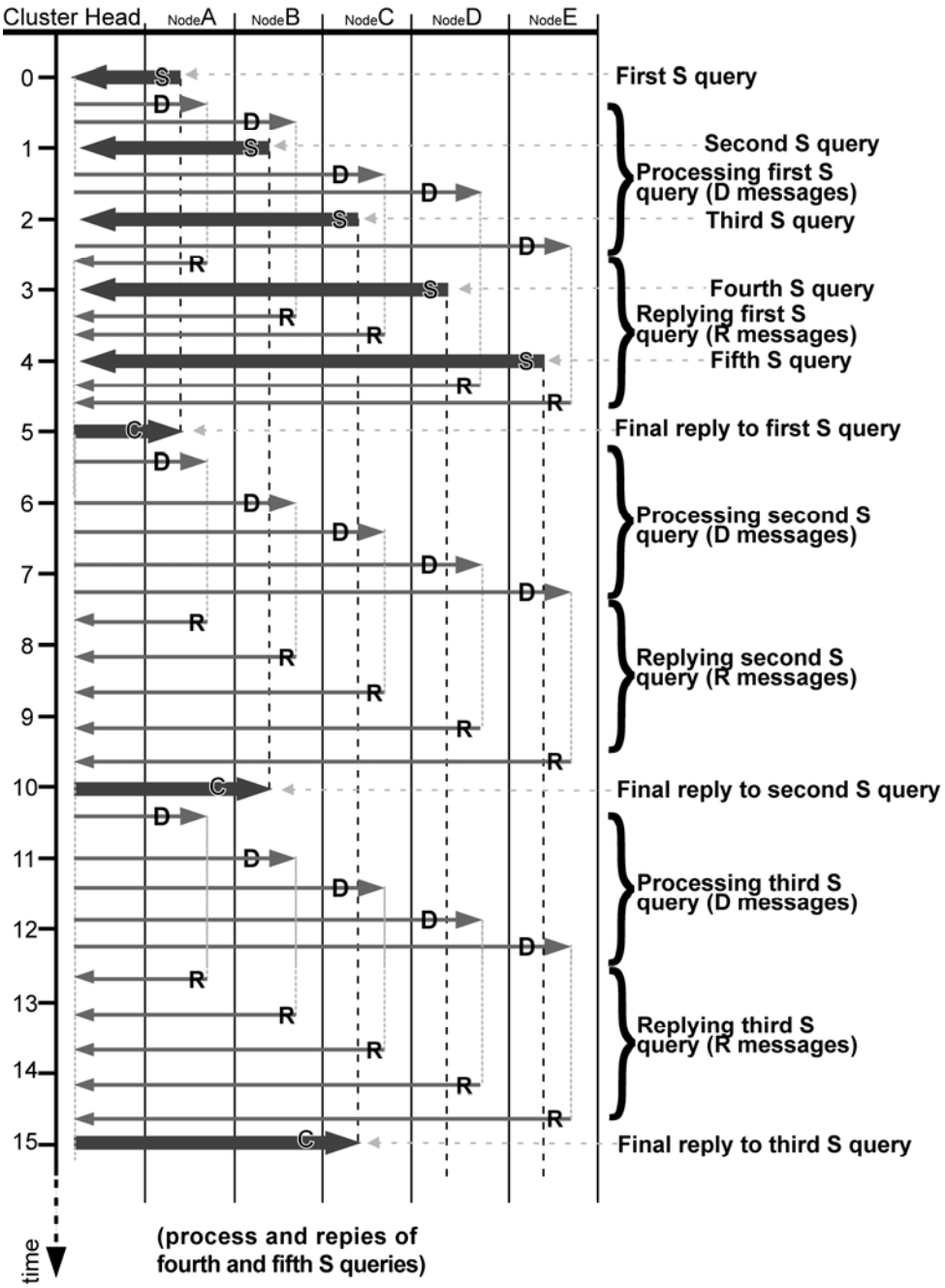


Figure 4.16 Message sequence diagram: One cluster, five requests issued each second.

– Assimilative processing.

The cluster head only performs the first query and queues the next similar ones. All the identical queries are answered simultaneously when the first

request is complete. The following example is similar to the previous one: Five queries issued every second. In this case, the first query, at t=0, will trigger a new polling process. The second, third, fourth and fifth query will be queued. When the reply to the first request arrives, 5t later, all of the nodes in the queue are replied to. The reply latency is 5t for the first, then 4, 3, 2, and finally 1t. for the fifth query. The average latency is 3 t.

Figure 4.17 illustrates this process.

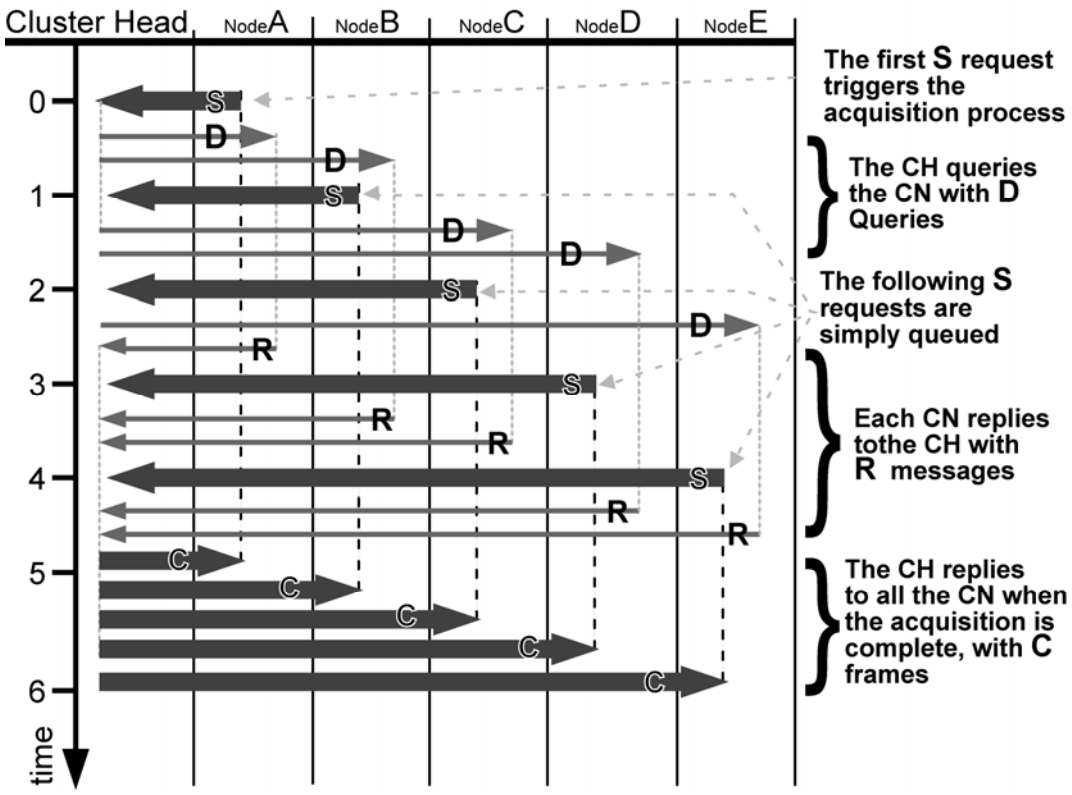


Figure 4.17 Five joined requests.

This technique does not accelerate the query process in itself. This method only improves the latency because identical queries "benefit" from the result of previous ones, hence reducing the reply time. If the queue is systematically empty, every successive request will trigger a new polling of the network.

C. Rule Acquisition Protocol

In order to obtain the data extracted by the learning agents, the protocol defines a sequence of messages to query a learning agent.

- T query: T queries are issued on request of an operator using a monitoring station. The T queries are created by the applet on the monitoring station, and emitted from the bridging node's ZigBee module. The query is addressed to a node hosting a learning agent, and holds a specific field for the identification number of the requested rule.
- F reply: This message is sent in reply to a T query. It contains all the relevant information concerning the requested rule. First, the rule number is included to avoid ambiguity, and is followed by the weight and firing occurrences. Finally the rule definition is transmitted.

Chapter 5 Learning algorithm

5.1 Introduction

This chapter will focus on the learning algorithm which forms the core of the learning agents. The purpose of the learning algorithm presented in this chapter is to determine and maintain a set of rules which represent the system's behavior. This is done by processing learning data acquired at regular intervals from the network of nodes.

This chapter is divided into two sections. The first section will present the underlying concepts and the principles of the learning algorithm. The second section will describe a preliminary test conducted to verify the concept.

A conclusion will be drawn from the preliminary tests, and propose enhancements for the final validation experiment. The final conclusions on this algorithm will appear in the *Validation* chapter (Chapter 7).

5.2 Learning Algorithm Concept

The learning algorithm provides the fundamental input to the automated process which creates the initial configuration of the system. The learning algorithm runs constantly to maintain an accurate knowledge of the system. The accuracy is maintained through progressive adaptation to changes in the environment.

5.2.1 Aim and Category of the Learning Algorithm

The learning algorithm considered is classified as a Multiple-Input, Single-Output (MISO) system. The inputs constitute all the sensors and meters in the network. The only output is a consumption prediction for a given appliance. The goal of the learning algorithm, as defined in [51] is to define the MISO function Φ in Equation 1:

$$\phi : X^P \rightarrow Y \tag{5.1}$$

where X are the *p* domains of input, and Y is the domain of output. The function Φ must verify all the learning sets. The learning data set Ω in Equation 5.2 is a collection of discrete samples $\omega(t_k)$ representing the state of the system in the space $\left(X^p \times Y\right)$ at a precise moment: t_k

$$\omega\left(t_k\right) = \left\{\left(x_1\left(t_k\right), x_2\left(t_k\right), \ldots, x_p\left(t_k\right)\right), y\left(t_k\right)\right\} \tag{5.2}$$

Where *x* are the *p* inputs and *y* is the output.

A fuzzy-logic learning is chosen for the following reasons:

- It can be implemented and run on a WACNet: it only requires simple arithmetic and logic that can be done by the node's microcontrollers, and uses a limited amount of memory, since it does not require storage of all datasets.
- It is the basic layer of several types of optimization algorithms, such as Neural Networks or Genetic Algorithms [53]. The agents developed in this work will be limited to the recording of fuzzy rules that match the system's behavior. Nevertheless, the output of this algorithm can be used in future work, when developing refinement algorithms.

5.2.2 Fundamental Concept: Fuzzy Sets

The learning agents will use a fuzzy logic-based learning algorithm. Before describing the algorithm, the underlying concept of Fuzzy Logic will be introduced.

Fuzzy Logic is based on the mathematical concept of Fuzzy sets. The theory of Fuzzy sets was introduced by Lofti Zadeh in 1965. It defines sets to which an element can have different degrees of membership. This is unlike classical sets, in which the membership is binary implying that an element can either belong or not belong to a set.

Fuzzy Logic is an extension of the classical Boolean logic. It is widely used in modern control systems. The main advantages are that it simplifies the modeling of complex problems and is tolerant to uncertainty. Fuzzy logic reduces the computation time while simplifying the configuration. Because it is simple yet powerful, fuzzy logic is the foundation of several categories of intelligent control systems.

5.2.3 Origin of Fuzzy Control: Human Perception

Fuzzy logic is very similar to how humans perceive their environment, and use these perceptions to make decisions. Unlike machines, humans can not obtain crisp values of the environment. The human mind deals only with perceptions. For example, when asked about the weather, most people describe their perception of the temperature using fuzzy terms such as quite cold, not warm, extremely hot, etc. rather than mentioning the precise temperature as a numerical value.

People usually grade their understanding of a perception as fuzzy subgroups. A qualifier is usually used in conjunction with the subgroup to define the degree of the perception. For example, for weather the sub-groupings include freezing, warm, hot, etc. The characterizing adjectives include a bit, slightly, fairly, very, completely, etc. The Perception of the weather for a particular individual can be illustrated by the diagram shown in Figure 5.1. For example, “freezing” is defined as temperatures between 0-10 degrees C.

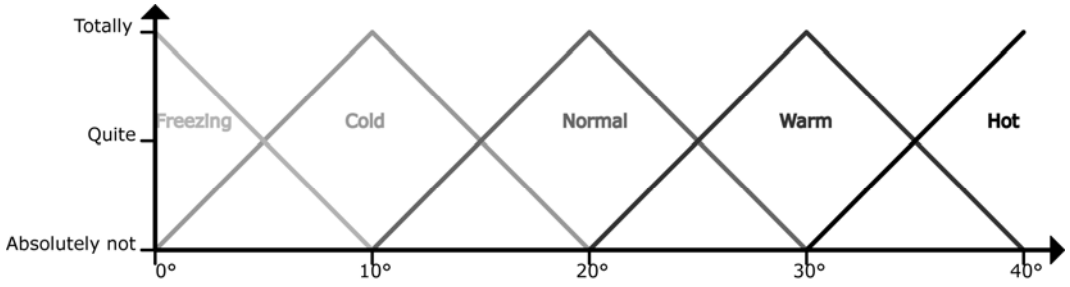


Figure 5.1 Generalization of how humans define their perception of temperature.

Such perceptions are also used in decision making using appropriate rules. Based on how well the conditions are met, the mind decides to which extent the rule should be applied. Provided that the individual has an accurate perception of the environment and the range of solutions, a few simple rules can be adequate to make correct decisions. This approach is very simple to apply, yet quite efficient. It can offer a wide range of responses, precisely adapted to any input value.

Considering the example of temperature, two rules suffice: "if it is *adjective* cold, wear *adjective* warm clothes" and an opposite one for hot temperatures. To make a decision, the subject considers how cold it is, and proportionally applies the solution proposed in the rule. If it is *extremely* cold, he or she will wear *extremely* warm clothes. If it is *not very* cold, then he or she will wear *not very* warm clothes. Obviously, a critical factor is the knowledge of the different ranges for "cold" and what corresponds to "warm clothes".

5.2.4 Fuzzy Logic Control

Fuzzy control is a control technique which uses fuzzy logic. The major steps to implement fuzzy control are:

(a) Conversion of various crisp inputs into compatible data.

(b) Application of fuzzy logic rules to the converted data. Each rule produces a *fuzzy* result.

(c) Calculating a precise control value based on the results produced by rules.

In the following sections, each of the above steps are described in more details.

A. Fuzzification of Crisp Values

Fuzzy logic uses the human approach of dealing with perceptions. The main concept is to divide the continuous output space of a sensor into a number of discrete sub-ranges. Having defined sub-ranges, any sensory value can be characterized with a set of numbers. Each number in the set represents the degree of membership that the sensory value obtains in each different sub-range.

Fuzzification is the term which refers to the process of converting a crisp sensor output value into a fuzzy value. In order to fuzzify a crisp value, the output space of a sensor must be broken down into well-defined overlapped sub-ranges known as *fuzzy sets*. Each fuzzy set is defined by a specific *mapping function* and is labeled with a letter or a number. The *truth value* of a crisp input represents the degree of membership to a given fuzzy set. A value of 1 corresponds to a full membership to a fuzzy set. Conversely, a value of 0 indicates no membership. A mapping function is used to fuzzify a crisp value.

As an example, the process of fuzzifying the temperature is described The output space of a temperature sensor (from 0 to 40°C) is partitioned into 5 fuzzy sets: *A*,*B*,*C*,*D* and *E*, as shown in Figure 5.2.

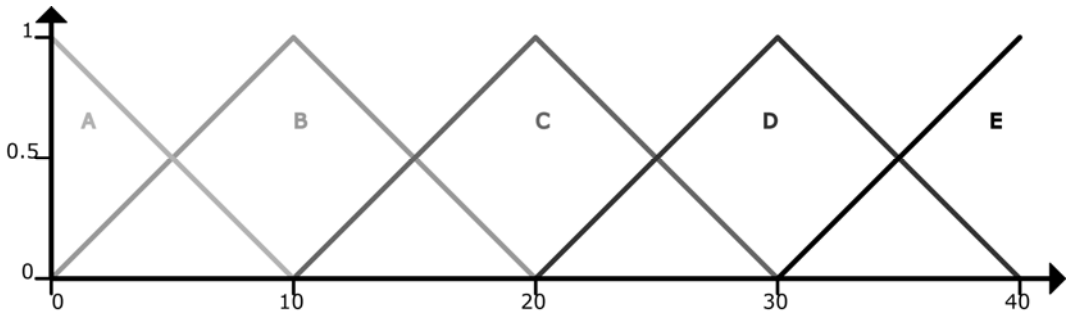


Figure 5.2 Partitions and mapping functions of a sensor's output space.

In this example, a fuzzified value of 5° would be described as follows:

0.5 in **A**	0.5 in **B**	0 in **C**	0 in **D**	0 in **E**

B. Evaluation of Fuzzy Rules

The next step is to evaluate the input data against a set of fuzzy rules. Fuzzy rules have a similar structure to conventional logic rules (IF-THEN statements). The *IF* part is called the *antecedent* and the *THEN* part is the *consequent*. The antecedent and consequent can have multiple conditions and effects, which are linked together by either an "OR" or an "AND".

An typical fuzzy rule has the following structure:

"IF **student_in_office** is **true** OR **student_at_home** is **true** AND **working** is **high**
THEN **productivity** is **high"**

This rule consists of three variables in the antecedent (**student_in_office**, **student_at_home** and **working**) and one variable in the consequent (**productivity**). Those variables can be defined by two or more fuzzy sets. These can consist of simple **true** and **false** sets or be more progressive, such as **null**, **low**, **nominal** and **high**.

Fuzzy rules apply in a similar way to Boolean logics rules. If all the inputs verify the conditions stated in the antecedent, then this rule applies to the current situation. All the rules which apply to a situation will influence the final output.

The output of the evaluation of a rule is fuzzy data. A variable, a resultant fuzzy set, and a corresponding truth value constitute the output of each evaluated rule. The truth value reveals how the rule matches the conditions stated in the antecedent. For any rule which applies to a data element, the truth value of the consequent equals the truth value of the antecedent. The truth value of an antecedent is calculated in different ways. If the rule has only one condition in the antecedent, the truth value of this condition becomes the truth value of the consequent. When a rule has more than one condition in the antecedent, the truth value of the consequent depends on whether the conditions are linked with "AND" or "OR".

OR implies that only one of the conditions needs to be matched for the rule to apply. Therefore the highest truth value for a variable in the antecedent becomes the truth value of the consequent.

To illustrate this, let us consider the rule:

IF **outside_temp** is **too_hot** OR **outside_temp** is **too_cold** THEN **going_outside** is a **bad_idea**

Considering a temperature of -15°, the truth values will be:

0.95 in **outside_temp** is **too_cold**	0 in **outside_temp** is **too_hot**

Only the highest value will be kept, in this case **too_cold**, and assigned to the consequent. Therefore, **going_outside** is a **bad_idea** with a strong truth value of 0.95 when the temperature is -15°.

If the temperature rises to 25°, the truth values will be:

0.05 in **outside_temp** is **too_cold**	0.10 in **outside_temp** is **too_hot**

The highest truth value found is **too_hot**, and the final result becomes: **going_outside** is a **bad_idea** with a weak truth value of 0.10.

AND requires all the conditions to be matched. Therefore, the lowest truth value found in the antecedent elements will be selected to become truth value of the consequent. To illustrate this, let us consider the rule:

IF **ambient_brightness** is **low** AND **presence** is **true** THEN **light_power** is **high**

Considering a dark room with no presence, the truth values will be:

0.95 in **ambient_brightness** is **low**	0 in **presence** is **true**

Only the lowest truth value is kept, in this case the value of **presence** is **true** and is assigned to the consequent. Therefore, **light_power** is **high** with a truth value of **0** when no one is in the room.

If someone enters the room, the truth values will change to:

0.95 in **ambient_brightness** is **low** 1 in **presence** is **true**

The lowest truth value becomes **ambient_brightness** is **low** and is assigned to the consequent. **Light_power** becomes **high** with a strong truth value of **0.95**.

C. Defuzzification

Actuators function with precise values. Consequently, the output of a fuzzy control system must be a crisp value. The crisp value is determined by the result of each evaluated rule. This process is called *defuzzification*. Many techniques exist, but most of them use a gravity-center computation to determine a crisp value. This produces a value which represents an average of all the rules.

5.2.5 Concept Critical Analysis

A. Fuzzy Logic over Boolean Logic

Boolean logic only supports two truth values (*true* or *false*). The major advantage of fuzzy logic over Boolean logic is that it tolerates various "degrees of truth". Fuzzy logic operates with any value within the continuous space between true (1) and false (0). This allows a finer control of the output. The output can be adjusted proportionally, depending on how well the situation matches an existing rule. Unlike fuzzy control, Boolean control can only have a finite number of output states, which reduces precision.

To illustrate this, let us consider the classic example of temperature regulation, with a system comprising of a heater with a proportional control and a temperature sensor.

- Boolean control:

A rule table to perform a very basic Boolean control of the heater would be as follows:

IF **room_below_20°** is **true** THEN **heater_on** is **true**

IF **room_below_20°** is **false** THEN **heater_on** is **false**

This control will simply activate the heater at its full power when the sensor reaches a threshold value, and stop it when the temperature is sufficient. This obviously results in a very rough "on/off" style control of the temperature.

The control can be refined by creating sub-ranges of temperature and considering different powers for the heater:

IF **room_below_10°** is **true** THEN **heater_on_100%** is **true**
IF **room_between_10°_20°** is **true** THEN **heater_on_50%** is **true**
IF **room_beyond_20°** is **true** THEN **heater_on_0%** is **true**

Even though this will produce a finer result, the output range will always be limited to a finite number of values.

- Fuzzy control:

In this example, the sensor input is partitioned over three fuzzy sets. An appropriate rule table to control this application would be:

IF **room_temp** is **cold** THEN **heater_power** is **on_high**
IF **room_temp** is **warm** THEN **heater_power** is **on_med**
IF **room_temp** is **hot** THEN **heater_power** is **off**

If the temperature is 20°, plausible truth values are:

0.1 in **room_temp** is **cold**	0.5 in **room_temp** is **warm**	0.2 in **room_temp** is **hot**

The three rules apply, which means that the fuzzy output value will be:

0.1 in **heater_power** is **on_high**	0.5 in **heater_power** is **on_med**	0.2 in **heater_power** is **on_high**

This output is then defuzzified to obtain a crisp value. A weighted average calculation will be performed, using the truth value as a weight. Unlike the output of a Boolean logic control system, there are no pre-defined values for the output of the defuzzification. The output space is continuous, which results in a fully proportional and fine control.

B. Fuzzy Control Over Non-logic Control

Compared to classic linear or non-linear control (integral, derivative, proportional functions, etc), fuzzy logic has the advantage of simplicity. Indeed, fuzzy control systems are faster to process and seldom require complex calculations. Analyzing and matching fuzzy rules is usually a simple and fast operation. It consists of finding a

minimum or maximum truth value. Most of the calculations are carried out during the fuzzification and defuzzification processes. However, these calculations usually remain quite simple. This is a major advantage over non-logic controllers. Non-logic controllers process complex mathematical equations to calculate an output. Although fuzzy control is simple to implement, it has no limits regarding behavior complexity. As long as there is a way to fuzzify an input, it can be used. This includes parameters such as derivatives or integrals.

Fuzzy control systems are also simpler to model and realize. The development of a fuzzy control system can be conducted in three steps. First; identification of the inputs and outputs, second; definition of mapping functions, and finally creation of a rule table. The conception of non-logic control systems requires identifying a single equation which incorporates all the inputs. The fuzzy control approach is often preferred when the model can not be correctly defined by an equation. The most critical task is to define and fine tune the fuzzy domains' mapping functions. Once this is done, the rules can be defined just by describing the desired output in a quite *natural* way (i.e.: IF **Temperature** is **Hot** THEN **Air-Conditioning** is **High**, etc.)

Fuzzy control also handles imprecision in the output. When an equation is applied to control the output, any slight variation or oscillation of the input creates an unstable output. This can be avoided with fuzzy control, provided that the mapping functions are well defined.

5.2.6 Basic Principle of the Learning Algorithm

The learning algorithm run by the learning agents is based on Fuzzy Logic. Fuzzy logic has been selected because it is well adapted to limited hardware platforms. It is relatively simple and therefore ideal for implementation on a microcontroller. Fuzzy logic also copes well with the imprecise and irregular behavior of human beings.

The algorithm learns through processing *data sets* which represent a sample of the state of every variable in the entire system, at a given time. The core concept of this learning algorithm is to contemplate every possible rule, and only mark the rules which match a learning data set. The algorithm assesses how a rule matches a learning set, by defining a firing strength. The rules considered by the algorithm

define a fuzzy set for each variable, and link the complete set of inputs with AND. The AND operator is required because the algorithm considers the state of the entire system, i.e. the state of all the variables at an instant of time.

The total number of rules the algorithm considers is *S*. The total number of fuzzy rules is the number of possible combinations of every variable's fuzzy sets (with a preset order) is obtained by applying Equation 5.3:

$$S = \prod_{n=0}^{p} n_{sets}(x_n) \tag{5.3}$$

where n_{sets} is the number of fuzzy sets which divide the output space of a sensor x_n.

The algorithm proceeds as shown in Figure 5.3.

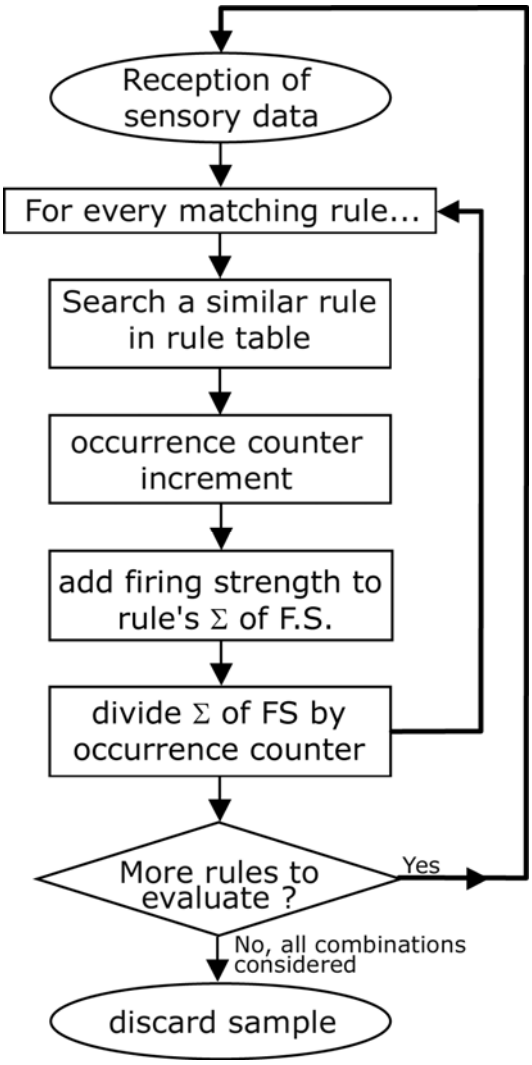


Figure 5.3 Learning algorithm sequence diagram.

Every time the learning agent acquires a learning data set, the algorithm compares the fuzzy data against all the possible rules. The general structure of the rules is: Var_1 is n_1 AND Var_2 is n_2 ... AND Var_k is n_k. Each time a rule corresponding to the data element received, the algorithm calculates the firing strength. The firing strength of a rule corresponds to the weakest firing strength found amongst the variables, as the variables are linked with the AND operator. If the firing strength is non-null, the rule is recorded with its firing strength. A count of matching occurrences is also kept. Once this is done, the algorithm moves on to the next data element.

Figure 5.4 Process of a dataset.

Let us consider the agent receiving a dataset of three sensors each with 3 fuzzy sets (as in Figure 5.4). Each row represents the truth values the sensor obtains, for the fuzzy sets listed in columns. The sensors are sorted in increasing identification number, and the fuzzy sets from the lowest to the highest. A rule corresponds to a combination of fuzzy sets from each sensor. In this example, the highlighted numbers correspond to 1**A** 2**B** 3**A**:

Sensor_1 is **Fuzzy_set_A** AND **Sensor_2** is **Fuzzy_set_B** AND **Sensor_3** is **Fuzzy_set_A**

The first rule considered by the algorithm corresponds to all variables in their lowest fuzzy set: 1**A** 2**A** 3**A**. In this case, this rule returns a truth value of zero, because the truth value of **Sensor_2** is **Fuzzy_set_A** equals zero. When a rule contains an element with a null truth value, the algorithm immediately moves to the next rule. The algorithm continues searching for a new rule until one is found with a firing strength not equal to zero.

The next rule is found by increasing the fuzzy set of the last sensor, e.g. 1**A** 2**A** 3**B**, then 1**A** 2**A** 3**C**. Once the highest value has been reached for the lowest senor, the subsequent one is increased e.g. 1**A** 2**B** 3**A**. A valid rule is one in which there are no null truth values. In this example, the highlighted numbers correspond to the first valid rule found in this dataset: 1**A** 2**B** 3**A**. The firing strength of the rule corresponds to the lowest truth value found. In this case, it is for **Sensor_2** is **Fuzzy_set_B**. This rule has a firing strength of 0.3.

If the rule is not in the table, the algorithm saves it. The rule is copied in the rule table, with its weight, and the occurrence count to one. If the rule is already in the table, it is updated. A rule update consists of incrementing the occurrence counter, and averaging the current firing strength with the previous ones.

Once this is done, the algorithm moves to the next rule with a non-null weight. In this example, this corresponds to following the arrow *(1)* and evaluate 1**A** 2**B** 3**C**. It will then move along arrow (2), and re-consider the two possible fuzzy domains for Sensor_3: the rules 1**A** 2**C** 3**A** and 1**A** 2**C** 3**C** which present a non-null truth value.

Once all the rules have been evaluated, the current data set is discarded and a new set is ready to be processed.

Each created rule can be regarded as a complete “snapshot” of the system. They depict a particular configuration or state of the system. The count of occurrence reveals how often this state appears. The firing strength is a metric for how accurately the rule matched the state.

The rule table summarizes all the states of the system encountered in the learning data. The rules obtaining the best scores (in terms of firing strength and occurrence count) are the ones which best describe the system. A complete rule table is adequate to replicate and predict the behavior of the system. An analysis of the rule table can also lead to the detection of correlated variables.

Occur	Firing Strg	S_1	S_2	S_3	S_4	S_5
3200	95%	1	3	1	1	1
356	75%	1	3	1	1	2
1750	40%	1	3	1	2	1

Figure 5.5 An example of rule table, which holds 3 rules.

The Figure 5.5 shows an example of rule table containing 3 rules. The rules are read by rows. The first rule has been matched 3200 times, with an average strength of 0.95. It corresponds to the Sensor#1 in the fuzzy set #1, the sensor#2 in the fuzzy set#3, etc. This table also reveals a correlation between the sensors 1, 2 and 3 as they systematically appear in the same state.

5.3 Preliminary Test and Verification

In this section, the algorithm previously introduced is tested with Matlab. This experiment will demonstrate the capacity of the algorithm to produce rules representing different states of the system. The experiment consists firstly of generating learning data with various levels of correlation among the inputs. The algorithm will then process the generated learning data. Finally the results will be verified.

5.3.1 Algorithm Implementation

- Inputs and fuzzy domains identification

Numbers are used to designate different Fuzzy domains and inputs. This is done to standardize the representation of the rules. Each input is attributed an identifier number. These identifiers form a consecutive suite of integers. For example, **0** will be attributed to the **light** meter, 1 to the **thermostat** sensor, 2 to the **washing_machine** flow meter, etc. Every fuzzy set, from lowest to highest, is also referred to by a consecutive suite of integers. For instance the integer 0 corresponds to *low*, 1 to

nominal and the number 2 represents the *high* fuzzy set. When a rule is created, the variables are implicitly sorted by increasing identifier numbers. Hence only the concerned fuzzy domain needs to appear. For instance, the rule coded as [0 2 1] must be interpreted as : **input_#0** is *Fuzzy_set_#0* AND **input_#1** is *Fuzzy_set_#2* AND **input_#2** is *Fuzzy_set_#1*. According to the examples cited above, this rule would be: **light** is *low* AND **thermostat** is *high* AND **washing_machine** is *nominal*.

- Rule identification

The rules are identified by an integer. The use of identifiers provides a simple and reliable way to deal with a large number of rules without writing them in memory. Each rule is attributed a unique integer, part of a consecutive suite from 0 to *S*-1 (eq. 3).

As the rule's fuzzy sets "increases", so does the rule identifier. The numbering technique is inspired from usual counting. The only difference is that each digit can only increase up to $n_{sets}(x_n)$. Indeed, $n_{sets}(x_n)$ corresponds to the fuzzy sets the variable x_n uses. The rule H_0 corresponds to the rule zero. H_0 is the rule which has all the inputs on the lowest fuzzy domains: [0 0 0 **…** 0]. Starting from H_0, the immediate next rule (H_{0+1}) in the list is obtained by incrementing the rightmost digit. When this digit has reached the highest possible value, the next digit to the left is incremented, and so on. The last possible rule is H_{S-1}. H_{S-1} is the rule in which all inputs are in their highest fuzzy set.

Considering a set of rules recognizing three inputs, each having three fuzzy domains, the 27 rules would be numbered as follows:

Table 5.1 All possible rules and their identifier numbers, for a 3 input system with three fuzzy sets per input.

H_0	[0 0 0]	H_9	[1 0 0]	H_{18}	[2 0 0]
H_1	[0 0 1]	H_{10}	[1 0 1]	H_{19}	[2 0 1]
H_2	[0 0 2]	H_{11}	[1 0 2]	H_{20}	[2 0 2]
H_3	[0 1 0]	H_{12}	[1 1 0]	H_{21}	[2 1 0]
H_4	[0 1 1]	H_{13}	[1 1 1]	H_{22}	[2 1 1]
H_5	[0 1 2]	H_{14}	[1 1 2]	H_{23}	[2 1 2]
H_6	[0 2 0]	H_{15}	[1 2 0]	H_{24}	[2 2 0]
H_7	[0 2 1]	H_{16}	[1 2 1]	H_{25}	[2 2 1]
H_8	[0 2 2]	H_{17}	[1 2 2]	H_{26}	[2 2 2]

The table is read in ascending numerical order from 0 to 26, which is column wise from left to right. H_{S-1} corresponds to H_{26}.

- Main loop

The program processes the data elements as follows. Each time a new sample of data is presented, the algorithm fuzzifies the inputs. The firing strength of each rule is calculated with respect to the conditions imposed by the AND operator. This means the lowest truth value found within the selected set becomes the firing strength of the rule.

- Storage of results

The results are stored in an array. The index of the array corresponds to the rule number. The array stores the average firing strength and the count of firing occurrences of each rule. At the beginning of the simulation, all the values in the result array are initialized to zero.

Each matched rule (i.e. its firing strength is non-null) is updated. The firing strength obtained by a sample is averaged with the previous ones, while the count of firing occurrences is incremented.

5.3.2 Experimental Setup

An experiment is conducted using learning data representing a plausible configuration. The setup consists of four input variables and one output. The inputs sensors are outdoor light brightness (sensor 0), presence in a room (sensor 1), washing-machine on/off (sensor 2) and the water consumption (sensor 3). The considered output of this system is the electrical consumption of a light (sensor 4).

Fuzzy sets are defined as follows. The outdoor light brightness and the level of water consumption are each defined by three fuzzy sets of *DARK* (fuzzy set id: 0), *MEDIUM* (1) and *BRIGHT* (2) for the brightness, *LOW* (0), *NORMAL* (1) and *HIGH* (2) for the consumption. The two remaining sensors have binary outputs. Their value can only be either [a truth value of 1 in *ON* (id: 0) **and** 0 in *OFF* (id: 1)] **or** [a truth value of 0 in *ON* **and** 1 in *OFF*]. The output has three fuzzy sets referred to as *LOW* (fuzzy set id: 0), *NORMAL* (1) and *HIGH* (2).

A dataset of 1000 samples is generated to test the algorithm. The learning data is generated to represent some realistic actions of the occupants in the environment described. The represented actions include the indoor light consumption which tends to increase as the daylight gets weaker and *vice versa*. The light is systematically turned off when the room is vacant (i.e. the presence detector is *false*). The two remaining variables (washing-machine electrical consumption and water consumption) are totally unrelated to the three other sensors, and to each other. The datasets of these variables are generated by a random function.

Two variables with no correlation with the output of the system (electrical consumption of a light) have been voluntarily introduced. This allows the study of the influence of unrelated variables on the output of the algorithm. The influence of unrelated variables is an important factor for future implementation. This is because the learning algorithm, once installed in a house, will deal with a large quantity of information with various degrees of relevance.

A selection of the learning data is plotted in Figure 5.6, 5.7 and 5.8.

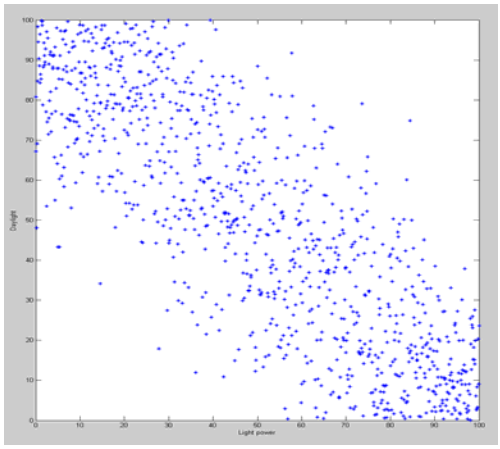


Figure 5.6: Daylight (Y axis) versus indoor light power(X).

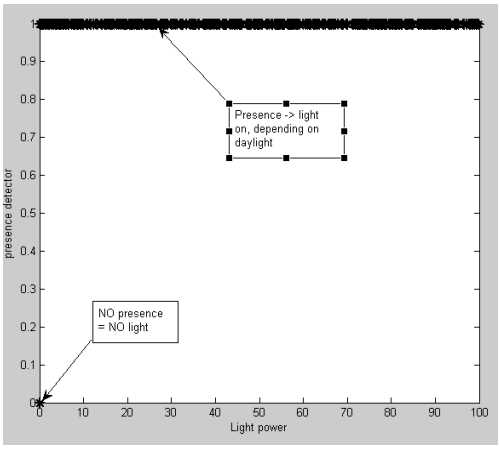


Figure 5.7: Indoor light power (X axis) versus presence(Y).

The Figure 5.6 confirms that this data is generated by a -1 coefficient linear function. Noise is added to represent the uncertainty in the decisions made by an individual.

The Figure 5.7 shows that when the room is vacant, the light is off. When it is occupied, the light is on, its power is determined by the daylight coefficient

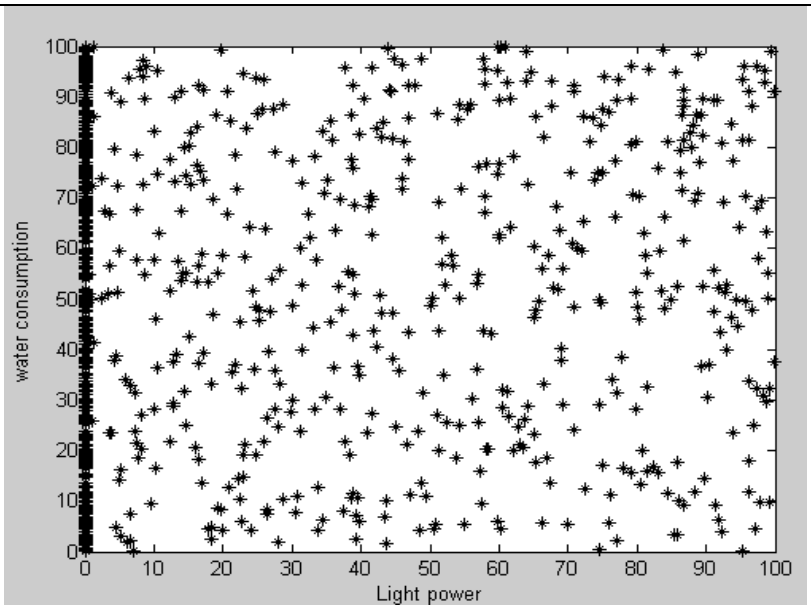


Figure 5.8: Water consumption (Y) versus indoor light brightness (X).

The graph on Figure 5.8 shows that there is no correlation between the two considered variables. The washing-machine on/off graph leads to a similar conclusion.

5.3.3 Experimental Results

This section presents the results obtained after running the algorithm with the data presented above.

Out of 108 possible rules, the results of the algorithm indicate that 61 rules have fired at least once. The firing strength obtained by each rule is shown in Figure 5.9.

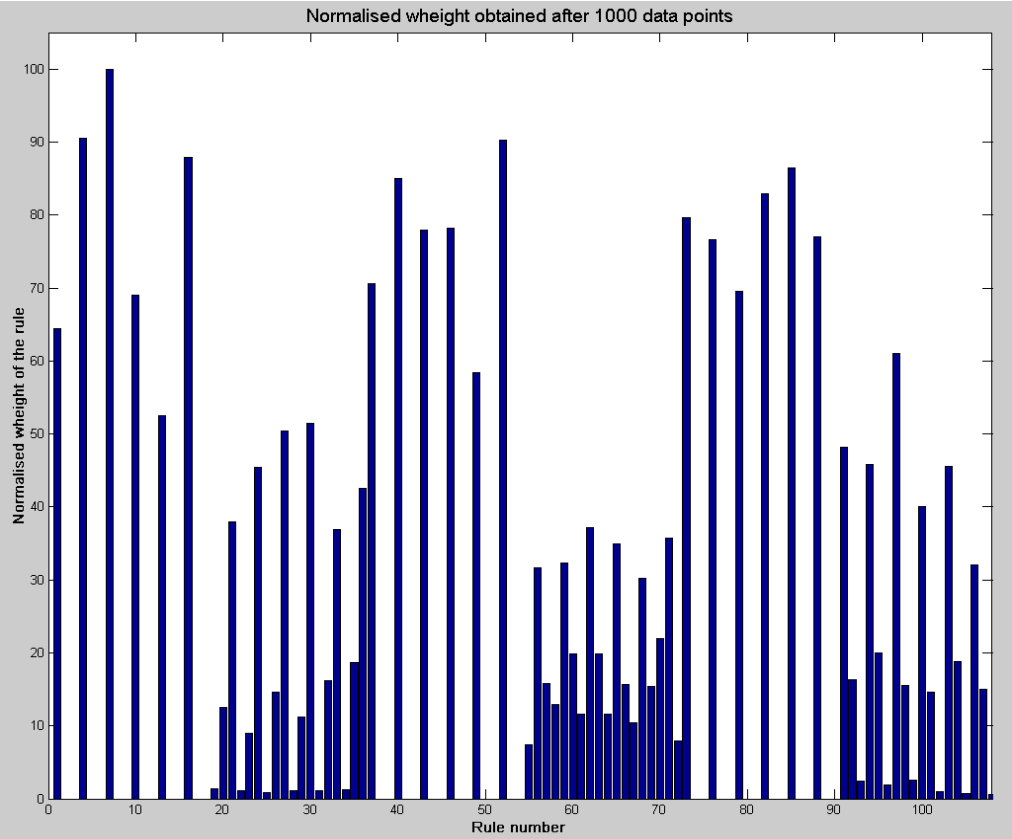


Figure 5.9: Raw output of the algorithm: rule ID number (X axis) and firing strength (Y).

- Accuracy

The accuracy of the algorithm can be determined through a comparison of the rules selected by the algorithm and the behaviors represented in the learning data. The results show that the algorithm has worked effectively. The rules selected match the rules used to create the learning data. In this case, the results show that the rules predicting the output to LOW when there is no presence obtain the highest ranks in terms of firing occurrences and firing strength. The rules predicting a rise in output consumption when the daylight fades (or a decrease when the daylight gets stronger) are also predominant.

- Optimal rules

Despite portraying accurate results, the raw output of this algorithm lacks refinement. Indeed, the results show a high level of redundancy. This can be observed by analyzing the rules related to the "turn the light off when no presence is detected" behavior. This simple behavior is defined by 18 rules in the output table. In each of the 18 rules, the two correlated variables (i.e. the presence detector and the light consumption) systematically appear, along with all the 18 possible combinations of the three fuzzy sets of the 3 remaining irrelevant variables (water consumption, washing machine on/off and daylight sensor).

Table 5.1: Extract of the rule table: 18 rules representing the "When presence is FALSE the light is LOW" behavior.

rank	W	rule											
1	100%	If daylight IS	BRIGHT	AND pres. is	FALSE	AND w-mach IS	OFF	AND water IS	HIGH	THEN light IS	LOW
2	93%	If daylight IS	DARK	AND pres. is	FALSE	AND w-mach IS	OFF	AND water IS	LOW	THEN light IS	LOW
3	89%	If daylight IS	MED	AND pres. is	FALSE	AND w-mach IS	ON	AND water IS	NORM	THEN light IS	LOW
4	88%	If daylight IS	MED	AND pres. is	FALSE	AND w-mach IS	OFF	AND water IS	LOW	THEN light IS	LOW
5	87%	If daylight IS	DARK	AND pres. is	FALSE	AND w-mach IS	ON	AND water IS	HIGH	THEN light IS	LOW
6	81%	If daylight IS	BRIGHT	AND pres. is	FALSE	AND w-mach IS	OFF	AND water IS	LOW	THEN light IS	LOW
7	76%	If daylight IS	DARK	AND pres. is	FALSE	AND w-mach IS	OFF	AND water IS	HIGH	THEN light IS	LOW
8	75%	If daylight IS	MED	AND pres. is	FALSE	AND w-mach IS	OFF	AND water IS	HIGH	THEN light IS	LOW
9	74%	If daylight IS	MED	AND pres. is	FALSE	AND w-mach IS	ON	AND water IS	HIGH	THEN light IS	LOW
10	73%	If daylight IS	MED	AND pres. is	FALSE	AND w-mach IS	OFF	AND water IS	NORM	THEN light IS	LOW
11	69%	If daylight IS	BRIGHT	AND pres. is	FALSE	AND w-mach IS	ON	AND water IS	LOW	THEN light IS	LOW
12	69%	If daylight IS	DARK	AND pres. is	FALSE	AND w-mach IS	ON	AND water IS	LOW	THEN light IS	LOW
13	68%	If daylight IS	BRIGHT	AND pres. is	FALSE	AND w-mach IS	ON	AND water IS	NORM	THEN light IS	LOW
14	67%	If daylight IS	BRIGHT	AND pres. is	FALSE	AND w-mach IS	OFF	AND water IS	NORM	THEN light IS	LOW
15	65%	If daylight IS	MED	AND pres. is	FALSE	AND w-mach IS	ON	AND water IS	LOW	THEN light IS	LOW
16	64%	If daylight IS	DARK	AND pres. is	FALSE	AND w-mach IS	OFF	AND water IS	NORM	THEN light IS	LOW
17	63%	If daylight IS	DARK	AND pres. is	FALSE	AND w-mach IS	ON	AND water IS	NORM	THEN light IS	LOW
18	62%	If daylight IS	BRIGHT	AND pres. is	FALSE	AND w-mach IS	ON	AND water IS	HIGH	THEN light IS	LOW

Each of these 18 rules describes the same behavior of "when the presence is false, the light is dark." The algorithm has selected 18 rules for this behavior, so that one

systematically applies no matter what the configuration of the three remaining variables.

5.3.4 Discussion

The observation of the results shows that the proposed learning algorithm has provided a solution for this simple problem. The data is processed fast using simple operations. Despite the noise and imprecision in the experimental learning data, the algorithm has promoted the most relevant rules, with the highest firing strengths and firing occurrences.

The results also highlight the fact that this algorithm severely lacks the assistance of a rule optimization algorithm. This experiment was set up with few variables using a restricted number of fuzzy sets. This configuration could only generate a maximum of 108 different rules. The algorithm selected 61 rules, when in fact an engineer could describe the system in less than a dozen.

This section will first discuss the issues related to the rule redundancy, and will then comment on the problem of insufficiently relevant/frequent rules.

A. Rule redundancy

The results show that the algorithm does not detect the uncorrelated inputs. The presence of uncorrelated inputs in the learning data causes the selection of several redundant rules, when only one would suffice. This phenomenon is caused by a lack of flexibility of the rules. Indeed, irrelevant elements can not be withdrawn from a rule. The *AND* operator, compels every single element of the rule to be matched, for the rule to apply. If the learning set contains unrelated data, the algorithm will have to promote rules which apply for any state of the uncorrelated inputs. The redundant rules assist in fully applying corresponding rules to a pattern of only a few inputs independent from the state of the inputs.

This necessity for redundant rules results in the inflation of the rule table. In fact, based on a configuration obtained with only correlated inputs, each added irrelevant variable will multiply the size of the rule table by $N_{sets}(x_n)$; the total number of fuzzy domains.

Let us consider a practical example. A system is set up, and the algorithm runs with only correlated inputs. The learning algorithm produces three laws. If an unrelated input "*light switch*" is added, the system will have to promote rules applying for each fuzzy set of the light switch. In this case, the light switch only has two fuzzy sets. Therefore a set of rules will be promoted to apply when the light switch is on, and another set to apply when it is off. This will create two sets of redundant rules. As a result, the rule table is expanded by two because of this uncorrelated inputs.

In this case, the simulation results are correct but not optimal, as the amount of selected laws is more than five times of what is necessary. This improper use of memory space is not acceptable, especially for a platform with limited computational capacity and memory size.

For the learning algorithm to produce expandable results, it must be completed with a rule-optimizing algorithm. This algorithm should be capable of freeing the rules from the influence of uncorrelated inputs. It should also detect and merge all the redundant rules. The rule optimization algorithm will have to create rules with a variable size. A variable-length storage technique, allowing the use of either *OR* or *AND*, will have to be developed.

The 18 rules in Table 5.1 would be an ideal target for rule merging. All 18 rules have the same effect of predicting low light energy consumption when the room is vacant. The 18 rules have clearly been selected to apply this policy no matter what the state of the three other variables. These rules carry a relatively high firing strength of between 60 and 90%. Those rules could be merged into a single rule: IF pres. is **FALSE** THEN light is **LOW**. The optimized rule would be given a firing strength representing the average of the rules which verify it. The count of occurrences would be the sum of all the counters of the rules from which the merged rule is derived.

Many methods can be applied to optimize the rule table. Genetic algorithms are the most efficient optimizing algorithms for these applications. The drawback of such algorithms is that they require a great deal of computation and memory, usually referring back to past data sets, and can be slow to converge. Such algorithms can not be considered here, because they will not meet the constraints of the platform. A

much simpler approach must be taken. A well-configured rule-based algorithm could sufficiently improve the rule table. The design and development of such an algorithm will be left for further studies.

B. Insufficiently relevant rules

By observing Figure 5.9, it appears that several rules obtain a low firing strength. The rules with a firing strength 75% lower than the strongest rules can be considered as *weak*. The weak rules are those which do not accurately match a situation and do not significantly influence the final output. If a level of imprecision is tolerated, these rules could be omitted with limited consequences. This would ensure that memory is used to store only reasonably relevant rules, depending upon the level of imprecision allowed.

C. Insufficiently frequent rules

The results also reveal the existence of rules which apply to less than 5% of the data sets. These rules are called *singular* rules. Singular rules represent irregularities in the behavior of the inhabitants. These rules do not represent the most common situations encountered during the learning process. Depending on the requirements of the future application, *singular* rules could be deleted from the rule table.

D. Challenges

To appropriately remove irrelevant rules, further research must be undertaken. The influence of the selection metrics must be determined. The two metrics available for rule selection are average firing strength and count of firing occurrences. The influence of these two metrics, used as selection criteria, will be analyzed in the experiment with the intention of validating the learning algorithm.

5.3.5 Conclusion of the Preliminary Tests

The preliminary experiment conducted in this section proves that the learning algorithm succeeded in selecting rules corresponding to the behaviors represented by the learning data.

It has also proven that despite a correct result, the output must be optimized to be deployed practically.

This preliminary test shows that the fuzzy learning algorithm requires two additional modules of rule selection and rule optimization for efficient operation.

If the amount of redundant or weak rules is not reduced, this will become a significant problem when the algorithm runs in a limited memory embedded system, occupied with many other tasks. For example, a house fitted with 20 sensors (proportional and On/Off values) and a time management module would create several millions of potential rules.

The algorithm which will validate the concept will take advantage of the observations and conclusions of the preliminary test. The rules will be stored with a flexible structure, and a module will be developed and tested to select only the most relevant rules.

Chapter 6 Validation

6.1 Introduction

The theoretical concepts presented in this study will be validated in this chapter. This will be done through a series of experiments designed to demonstrate the feasibility and potential of the platform and the concepts developed.

The chapter comprises of three sections. The first will present the results of the benchmark test of a node. It will focus on the following parameters: computational power, message transmission rate and delays. The second section will provide the results of the simulation of a large WACNet. The results will illustrate the effectiveness of the network topology and the data acquisition process presented formerly. The final section presents the results of a practical implementation of the learning algorithm on real WACNet nodes.

The chapter will unfold by providing a description of the experimental setup. The parameters studied in the simulation will be described and their significance will be highlighted. The results of the experiment will be presented and critically analysed.

6.2 Platform Benchmarks

The first step of this validation process is to verify the performance of the upgraded node. This is achieved by setting a benchmark test using two nodes. This will quantify parameters such as the transmission delay, data throughput, computational speed of the microcontroller, error rate and others.

6.2.1 Node Benchmark Setup

The aim of this benchmark test is to acquire the characteristics of the nodes as the basic element of the WACNet framework. The benchmark test uses two WACNet nodes. The microcontrollers are programmed to carry out relevant tasks for measurement. A digital oscilloscope is used to measure the success rate, execution

times and transmission delays. The inputs of the oscilloscope are connected to dedicated output pins of the node's microcontroller. The oscilloscope's trigger is set so that the acquisition begins when the pin's voltage rises. Using specific instructions in the program, the state of these output pins is changed when an event of interest occurs. The success of the operation is confirmed by the presence of rising and falling voltages on the oscilloscope's display. The oscilloscope also allows the measurement of delays and execution times, by measuring the time interval between the rise and fall of the pin's voltage.

This approach is inaccurate, as a small number of clock cycles are required to change the state of the output pin. Since the microcontroller's clock period T is considerably smaller than the delays measured, the bias is insignificant. Indeed, setting an output at a high level takes less than 62.5 ns, while the smallest transmission time measured in this benchmark is 17ms. Computation delay measurements are the most sensitive to this bias. Since these delays are very small, they carry a higher relative error rate, which has been measured to be around ±2%.

The tests consist of repeatedly sending messages with defined payloads at the fastest possible frequency before exceeding the capacity of the communication link. In practice, this happens when the transmission buffer overruns, causing the node to crash. The microcontroller is set to acquire the computation time, time of the start and completion of transmission and reception, and finally the arrival time of the acknowledgement frame.

6.2.2 Node Benchmark Results

6.2.2.1 Computation Delay

The computation delay is monitored using the following protocol. The microcontroller raises the voltage of the dedicated pin when the frame-making function is called, and lowers the voltage when the function returns. The delay is the time interval between rising and falling peaks.

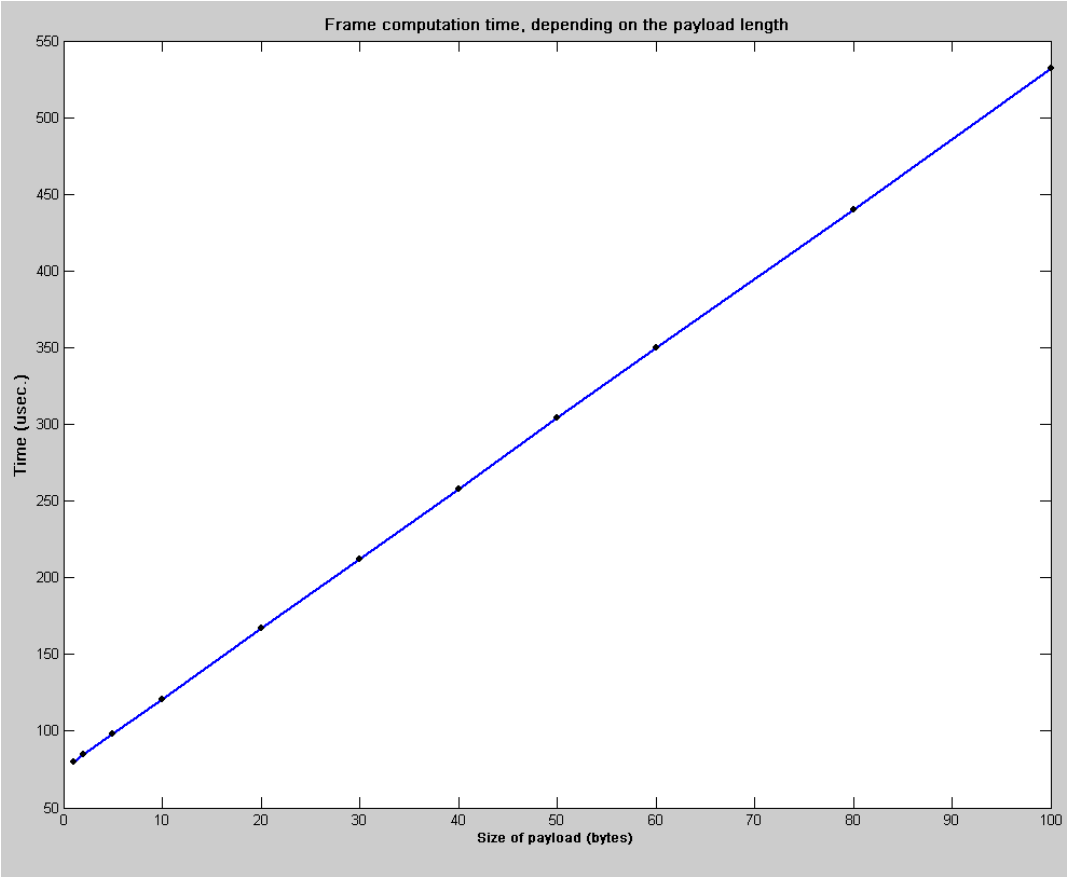


Figure 6.1: Computation delays (Y, µs), versus payload size (X, bytes).

Figure 6.1 shows the computation delay against the size of the frame's payload. This a linear function with a 80µs offset. The offset corresponds to the initialization and header computation delays. The computation delay reaches 530µs for the largest allowed payload size (100bytes).

Comparison with transmission delays reveals that computation time is insignificant. Considering a 100 byte message, only ≈0.2% of the total transmission time is used to compute the message. While one byte is transmitted through the UART (≈1ms), the microcontroller can compute up to 230 bytes of frames.

The graph shown in Figure 6.2 illustrates the total computation time required to transmit 1,000 bytes of data. The graph has a shape similar to a $1/kx$ function. Although according to Figure 6.1, the smaller frames require less computation time,

the computation time for a specified amount of data increases with smaller payloads as shown by Figure 6.2.

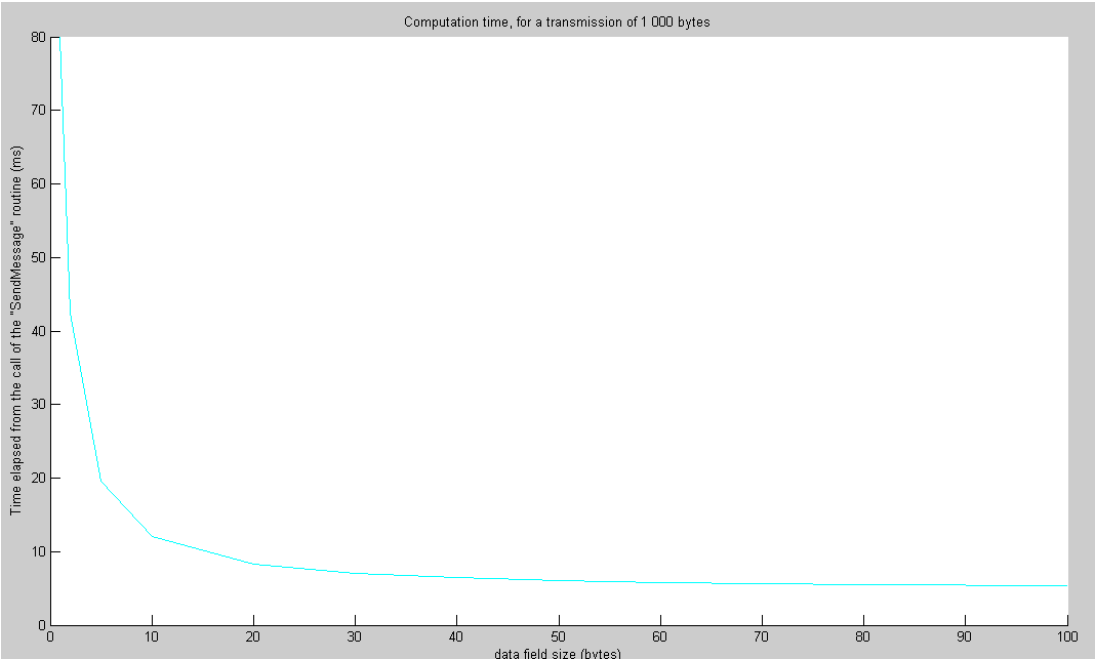


Figure 6.2: Total amount of time taken to compute the frames (Y,msec), for a 1,000 bytes-message, depending on the payload size.

When the payload is smaller, more frames are required to transmit a message. For each transmitted frame, 80µs of computational time is required to compute the frame's header and checksum. As a result, the total computation time is the highest when the payload size is 1 Byte, as it requires the computation of 1,000 frames. On the opposite extreme, the computation time is the smallest when the payload is 100bytes, as it only requires the computation of 10 frames.

6.2.2.2 Transmission Delays

A. Data Acquisition

This section will present all the delays involved in communication. The delays are acquired using a technique similar to the one described previously (6.1.1). The events monitored on the oscilloscope are the start and completion of transmission through

the UART, the start and end of the reception process at the receiving node's UART, and reception of an acknowledgment at the transmitting node.

B. Results

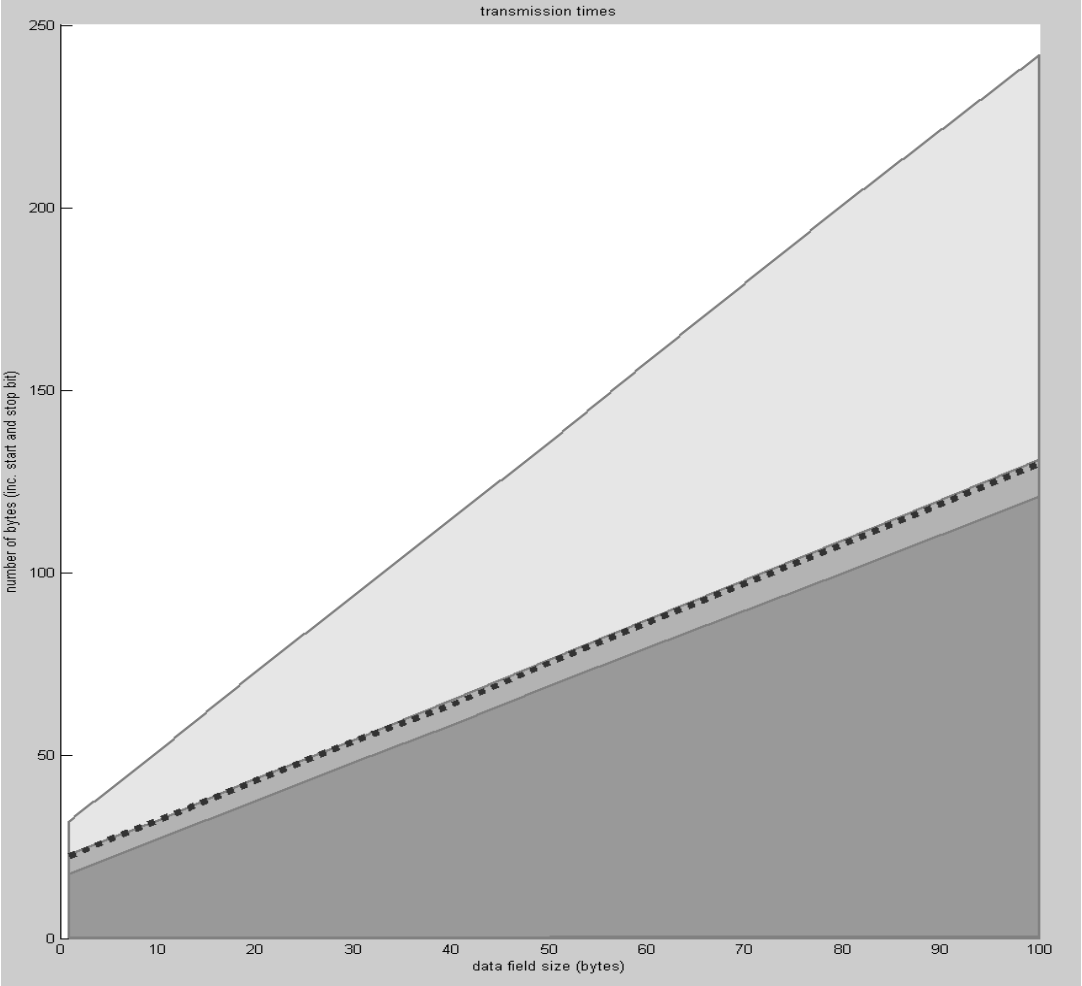


Figure 6.3: Transmission delays (Y) for one frame, depending on the size of the payload (X).

Figure 6.3 presents all the delays in a single graph. The plot represents delay (Y-axis, from 0 to 250 ms) versus payload size (X-axis, from 1 to 100 bytes). Each area corresponds to a step of the transmission process. They represent from the darkest to the lightest region the transmission to the ZigBee module (at source node), the ZigBee wireless transmission, and reception from the ZigBee module (at destination node), respectively. The initial computation time is also plotted, but it is too small to

be visible. The dashed line corresponds to the instant at which the acknowledgement is received.

C. Transmission

The transmission delay depends on the payload size $N_{databytes}$, and H, the total number of non-payload bytes (header, address, sequence number, checksum, etc.). The transmission delay t_{TX} (in ms) is calculated using Equation 6-1

$$t_{TX} = (N_{databytes} + H) \times 1.05 \pm 5\% \quad (6\text{-}1)$$

If the XBee chips use the ZigBee standard, H will be 18 and the computation time is insignificant and will be ignored. Using only two nodes, 100% of the messages are successfully transmitted and acknowledged.

D. Reception

The first reception interruption on the receiving node is triggered from 5 to 10 ms after the end of transmission. This delay increases with the size of the packet. The frame is completely written in the node's buffer and ready to be read at the destination after a delay t_{RX} (in ms) which is determined using Equation 6-2.

$$t_{RX} = (N_{databytes} + H) \times 2.1 \pm 5\% \quad (6\text{-}2)$$

E. Acknowledgement

The acknowledgement is received about 5 to 10ms after the last byte of the frame has been sent. The delay to receive completely the acknowledgement t_{ACK} (in ms) can be calculated using equation 6-3:

$$t_{ACK} = (N_{databytes} + H) \times 1.09 + 27.5 \quad (6\text{-}3)$$

F. Discussion

As expected, the graph in Figure 6.3 shows linearity within the data. The transmission time of a message cannot be below 30ms because of the overhead for both transmission and reception. The computation time is clearly insignificant. The transmission time between the two ZigBee modules is also very small. In fact, the delay is mainly due to the transfer rate from the microcontroller to the ZigBee

module, through the RS-232 link. A summary of the communication events is presented in Figure 6.4.

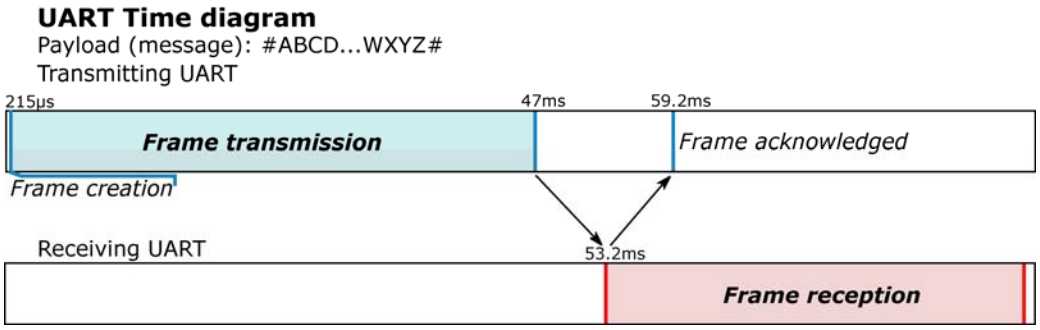


Figure 6.4: Time Diagram of Transmission. Top: transmitting node, bottom: receiving node.

6.2.2.3 Byte Rate

A. Acquisition

In this experiment, the maximal transmission speed is acquired with payload size as a parameter. The data is obtained by transmitting frames with a set payload. The delay between two transmissions is gradually decreased, until the frames are no longer received by the other node. When packets are no longer received, the network has reached its maximum throughput. The goodput (rate at which useful bits are transmitted, excluding overhead) is then calculated by removing the overhead. The graph in Figure 6-5 shows the goodput.

B. Result

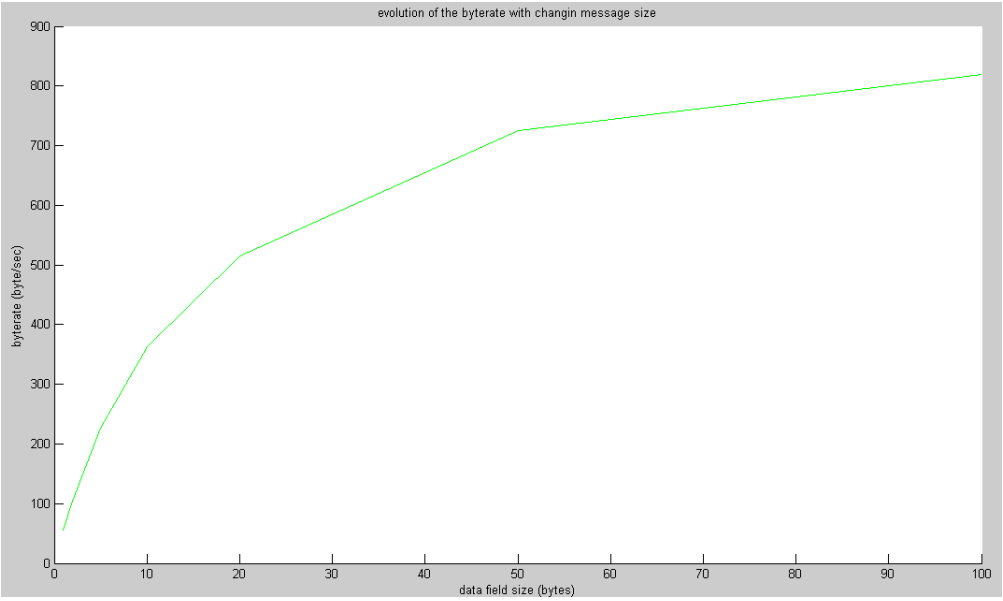


Figure 6.5: Goodput byte rate versus payload size.

The X-axis represents the payload size from 1 to 100 bytes, and the Y-axis is the goodput byte rate, from 0 to 900 bytes per second.

C. Discussion

The graph clearly demonstrates that the best goodput is reached when the payload is the largest. Indeed, the maximum rate of 800bytes per seconds is attained when the payload size is the highest value of 100 bytes. The function on this graph is approximated by Equation 6-4

$$G_{(Bytes/\sec)} = \frac{p_{bytes}}{(H_{bytes} + p_{bytes})} \times T_k \qquad (6\text{-}4)$$

This equation implies that the goodput G depends directly on the ratio of the length of payload p_{bytes} and the length of the total frame (H_{bytes}+p_{bytes}). In order to illustrate this, let us consider a frame composed of 1 byte of payload plus the necessary 18 bytes of overhead. In this case, only 5% of the transmission is actual data and the remaining 95% is the overhead. On the other hand, if the frame payload size is maximum, only 18 bytes of overhead are required for every 100 bytes of data. In this configuration, 85% of the transmission is useful bytes.

Considering this from a message-oriented point of view, the total amount of bytes Tx_{bytes} necessary to transmit a message of n_{bytes} can be calculated by applying Equation 6-5:

$$Tx_{bytes} = n_{bytes} + \frac{H_{bytes} \cdot n_{bytes}}{p_{bytes}} \quad (6\text{-}5)$$

This equation is plotted in the Figure 6-6, with n_{bytes} set to 1,000.

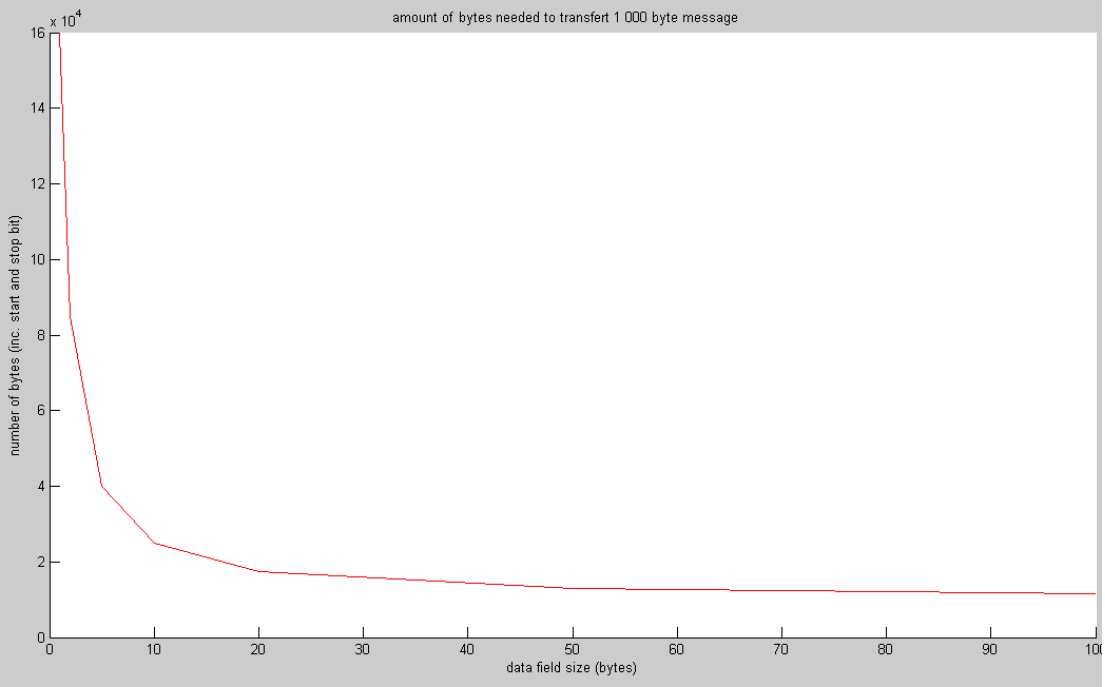


Figure 6.6: Real amount of bytes transmitted to send 1,000-bytes message (Y, in Bytes·10^4/sec), depending on the message payload size (X).

Figure 6-6 represents the number of bytes that must be transmitted in order to send 1,000 bytes of information. The X-axis represents the payload size (variable p_{bytes} in equation 6-5) from 1 to 100 bytes, and the Y-axis is the amount of transmitted bytes (Tx_{bytes} in equation 6-5) from 0 to 160,000. The plot has a shape similar to $y=1/kx$ function, which matches equation 6-5. It demonstrates that reducing the payload size increases the total amount of bytes required to transmit a message.

D. Conclusion

The capacity of the link is fixed by the throughput rate of the UART. The goodput depends on how much information is sent in frames. As the amount of overhead to transmit increases, less information can be transferred in a given time, which results is a lower goodput byte rate. This explains why the overhead ratio minimization plays a determining role in achieving the greatest throughput.

Another way to improve the throughput is to increase the baud rate of the nodes' UART. The relevance of this method will depend on the requirements of the application. Raising the baud rate will increase the flow of information through the network and will reduce delay. At the same time, it will also reduce the availability of the media which will be more prone to transmission failures and errors.

6.2.3 Discussion

This benchmark has revealed that the transfer time of a given amount of data is a linear function with offset. The latency increases with the size of the payload. From this strict point of view, the best solution to achieve low latencies appears to be sending frames with a small payload. Small frames containing limited amount of sensory values can achieve latencies below 40ms, and possibly even lower with a higher UART Baud rate.

When comparing the throughput and delay experiments, an optimization problem appears; delay increases when goodput decreases. Goodput is maximized when the frames are set at their full capacity, but this also induces larger delays.

This optimization problem can only be solved with a clear knowledge of the requirements of the system. Indeed, the application will define the importance of priority over fast and efficient transmission.

A buffering method could also be used, to prevent the individual transmission of very small messages. Using a buffer, small messages can be sent in one frame, which will reduce the number of transmitted frames and will increase the latency. This technique is applied when the XBee module is used in "cable replacement" transparent mode. In this condition, packets are only sent when a minimum of *n* bytes have been uploaded to the module.

6.3 WACNET Network Topology Simulation

Due to the limited amount of physical nodes, a simulator has been built to generate statistical data representing network communications. This data will allow a comparative analysis of the performances of the three possible WACNet topologies of mesh, star and cluster. The simulation will also determine the improvements resulting from the data-sharing and query-distributing protocol, described in Chapter 4.

6.3.1 Presentation

The proposed WACNet simulator is a Discrete-Event Simulator (DES), programmed in JAVA language. The simulator reproduces the flow of information generated by a WACNet acquiring data for the learning agents presented in Chapter 5. The four primary parameters of this simulator are the number of nodes, number of clusters, the frequency at which data queries are issued (or the delay between them) and finally, the duration of the simulation.

6.3.2 WACNet Simulator Description

6.3.2.1 Simulation method

Discrete Event Simulation is based on the execution of a schedule of discrete events, controlled by a discrete clock. In this case, the events are the transmission and reception of messages. A specific module of the simulator, referred to as *clock*, manages the simulation time. The modules are notified at each new "tick" of the clock. They perform the events scheduled at this *tick*, and update their list of events. Throughout the simulation, the events of interest are recorded. At the end of the simulation the events record is processed and presented as simulation results.

6.3.2.2 Programming Language

The simulator uses the JAVA Object-Oriented (OO) programming language. The Object-oriented programming languages are ideal to implement DES because they ease the modelling of complex systems. In order to create the DES, a few management classes are developed for time and simulation control. The remaining

classes describe the behaviour of the objects which will be simulated. The simplicity and convenience of the JAVA programming language also reduces development time.

6.3.2.3 Global Architecture

This section will present the general organization of the WACNet DES. It will begin with the central part of the DES (time and network control), and will be followed by a description of the simulated objects.

A. Core Class

The fundamental class and central coordinator of the Discrete Event Simulator is the Core class. This class instantiates every node, sets up the initial configuration of the network and finally controls the clock. It also retrieves the statistics from all the Node objects at the end of the simulation, to process simulation statistics.

The Core class proceeds in the following order:

- At the beginning of the simulation, the Core class instantiates a series of Node objects. The Node objects replicate the behavior of WACNet nodes. All the Nodes objects instantiated are stored in a JAVA ArrayList.
- The Core class sets up the network associations and assigns nodes to clusters, as defined in the simulation setup section.
- The Core class then begins the simulation. The clock uses a variable to keep track of the time. Each tick of the clock is advertised to the nodes which check their events schedule for the arrival or departure of packets. The time is incremented when all the nodes have updated their events list.
- Once the simulation clock reaches a value specified as a parameter, the simulation is interrupted. The Core class notifies all the Node objects to transmit their statistics. The statistics are compiled and printed to the console in a text file.

B. Node Object

The Node object represents a WACNet node and controls and coordinates the peripheral objects. It can act as a Cluster Head or a Cluster Node. Upon instantiation, Node objects create two dependant objects; a ZigBeeModule and a Stats. The ZigBeeModule replicates the ZigBee modules and manages the event schedule for

communication. The Stats object monitors different variables of interest during the simulation. The Node objects also have access to a common class which generates sensory data.

Depending on its status (CH or CN), the Node controls frame creation, local and remote sensory data acquisition, and cluster formation. The node communicates with respect to the protocol described in Chapter 4. The simulation begins with the presentation and cluster formation protocol, followed by cluster query and inter-cluster query processes.

Each time an event of interest occurs, the Node calls a specific function of its Stats object. The Stats objects are presented in D.

C. ZigBeeModule Objects

The ZigBeeModule object replicates the ZigBee module attached to every node. It is responsible for scheduling of transmissions and receiving of messages, with respect to the transmission delay model.

The performance of the Maxstream XBee chips could not be measured under large network circumstances. Therefore, there is no defined model implemented in the simulator to take into account heavy network traffics, networks with numerous nodes or deployed over a large geographical area. The ZigBeeModule object recreates the delays observed when the node's microcontroller is set to communicate at a speed of 9600kbps.

Due to the lack of a precise model, the time-related variables obtained through simulation can only be considered most accurate for small network simulations, and with nodes transmitting at a rate which is well below the maximum rating. The effect on the variables used to compare the different network topologies and configurations (such as efficiency and total amount of data per sample) is marginal.

D. Stats Object

Each node is provided with a Stat object, which records the values of different parameters during simulation. It monitors variables such as peak buffer size, amount of data transmitted, and also the number of requests issued and replies received.

For example, at every increase or decrease of buffer size the `Stats.rXBufferSize()` function is called. This method keeps track of the maximal buffer size throughout the simulation.

Each Stat object is called by the Core class at the end of the simulation to collect the value of each variable. The results of each individual node are processed by the Core class to produce global statistical data.

E. Visualization Object

The Core class instantiates a Visualization object at the beginning of the simulation. This class draws a simple Graphical User Display (GUI) which represents the current cluster configuration of the simulation. This is achieved through the JAVA standard graphics interface.

6.3.3 Simulation Setup

A network of 24 nodes exchanging data is simulated. The nodes issue queries which conform to the protocol in Chapter 4.

A total of 152 simulations are carried out, with two different parameters; the delay between two data queries (19 values), and the number of clusters in the network (8 values).

The cluster configurations consists of 1 cluster of 24 nodes (Star topology), 2, 3, 4, 6, 8 and 12 clusters (cluster topology), and finally 24 clusters of one node (mesh topology). These values will allow the comparative analysis of various clustered topologies against star and mesh topologies. The values chosen for the *number of clusters* parameter correspond to all the layouts permitting a homogeneous distribution of the nodes among the clusters (i.e. identical number of nodes in each cluster). These values can be obtained with Equation 6-6:

$$REM(n_{nodes}, n_{clusters}) = 0 \tag{6-6}$$

REM() is the *remainder* function, and n_{nodes} and $n_{clusters}$ correspond to the number of nodes and the number of clusters in the network, respectively.

The simulated polling time intervals range from 2 seconds up to 1600seconds (26”40’), with a decreasing resolution as delay increases. This corresponds to polling frequencies from $6.25 \cdot 10^{-4}$ to 0.5Hz.

This set of experiments aim at determining an optimal configuration which produces the largest quantity of data, while minimizing the traffic. This configuration is determined by monitoring the following variables:

- Average amount of learning datasets received by each node during the simulation.
- Average amount of data received and transmitted.
- Maximal size of the buffers throughout the simulations.

6.3.4 Simulation Results

The results obtained in the simulation of the following variables are presented in this section: reception and transmission throughputs, transmission and reception efficiency (amount of bytes required to obtain a data set), and finally the total amount of data sets acquired during the simulation.

6.3.4.1 Reception and Transmission Throughput

The following section presents the results of the raw throughput variable, for Cluster Heads and Cluster Nodes.

A. Calculations

The average throughput is expressed in bytes per seconds, and is obtained by performing the following calculation:

$$T_{avg} = \frac{n_{bytes}}{t_{sim}} \tag{6-7}$$

Where T_{avg} is the average throughput (bytes/sec), n_{bytes} the number of bytes sent during the simulation and t_{sim} the length of the simulation in seconds. When possible, the data is averaged over all the nodes of similar rank (either Cluster Heads or Cluster Nodes).

The byte rates considered represent the average flow of data at the links marked on Figure 6.7: “*Raw throughput (reception)*” and “*Raw throughput (Transmission)*”.

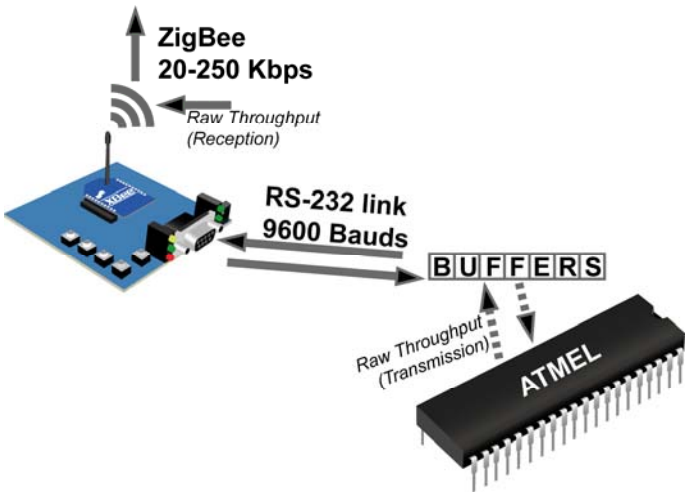


Figure 6.7: Different links involved in communication.

The byte rate graphs shown in Figure 6.8 only display values from 0 to 1,066 bytes per second. The higher values are truncated at 1,066Bytes/s. These configurations are not stable as they exceed the 9,600Bauds limit. These unstable configurations are irrelevant to this study and therefore do not require to be sketched.

B. Transmission and Reception Throughputs of the Cluster Heads

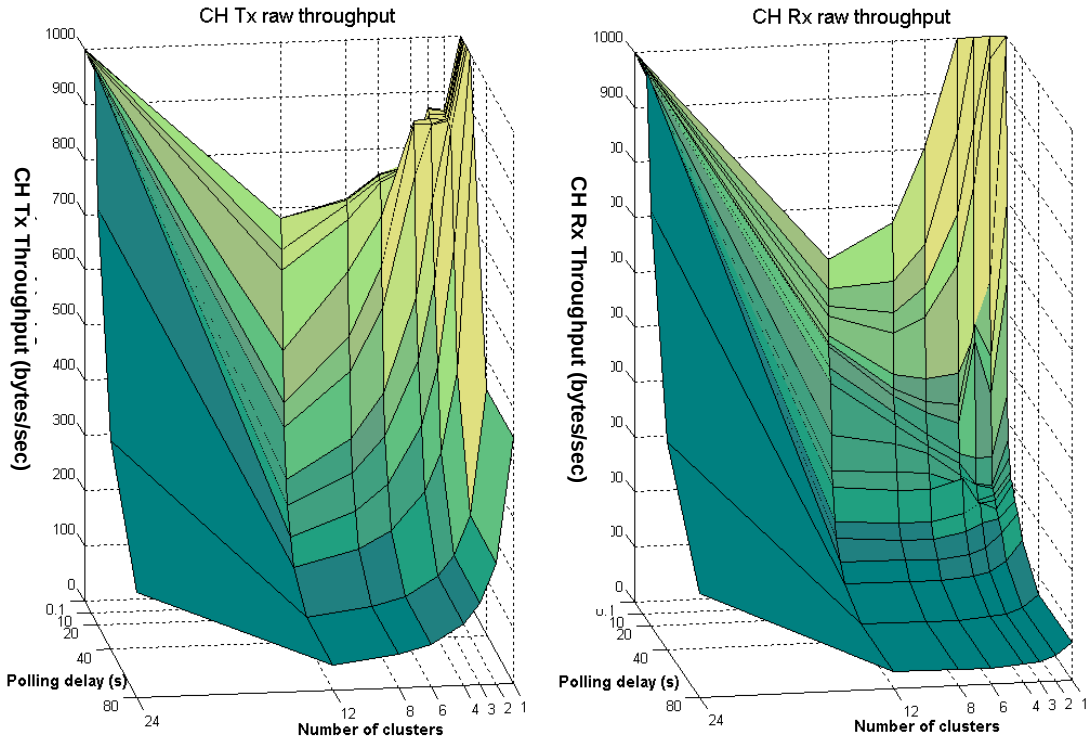


Figure 6.8: Raw Transmission (Tx) and Reception (Rx) throughputs for the Cluster Head nodes.

The left and right graphs of the Figure 6.8 respectively represent the transmission throughput and the reception throughput of a Cluster Head node.

The two graphs present a similar shape. This data clearly shows that star and mesh network topologies require larger throughputs than clustered topologies. Regarding polling frequency, the graph shows that issuing queries more frequently increases the network traffic, as expected.

The lower throughputs observed for the clustered topologies is attributed to the data-sharing protocol, which is active only in clustered topologies. As the clusters enlarge, the benefits of clustered task distribution tend to fade. In large clusters, the Cluster Heads have to communicate with many Cluster Nodes, which consequently raises the throughput. As the clusters become smaller, the Cluster Heads

communicate with fewer nodes. This reduces the throughput. Mesh topology requires the Cluster Heads to query individually the whole network of Cluster Heads, without distribution. This is why this configuration requires such high throughputs.

Overall, this graph shows that star or mesh network configurations are not viable solutions. This is because these configurations require throughputs which are beyond the 9600Bps maximal transmission rate of the nodes. Running the network in these configurations will create congestion at the RS-232 link, which will eventually overflow the buffers of the node or the ZigBee module.

C. Transmission and Reception Throughputs of the Cluster Nodes

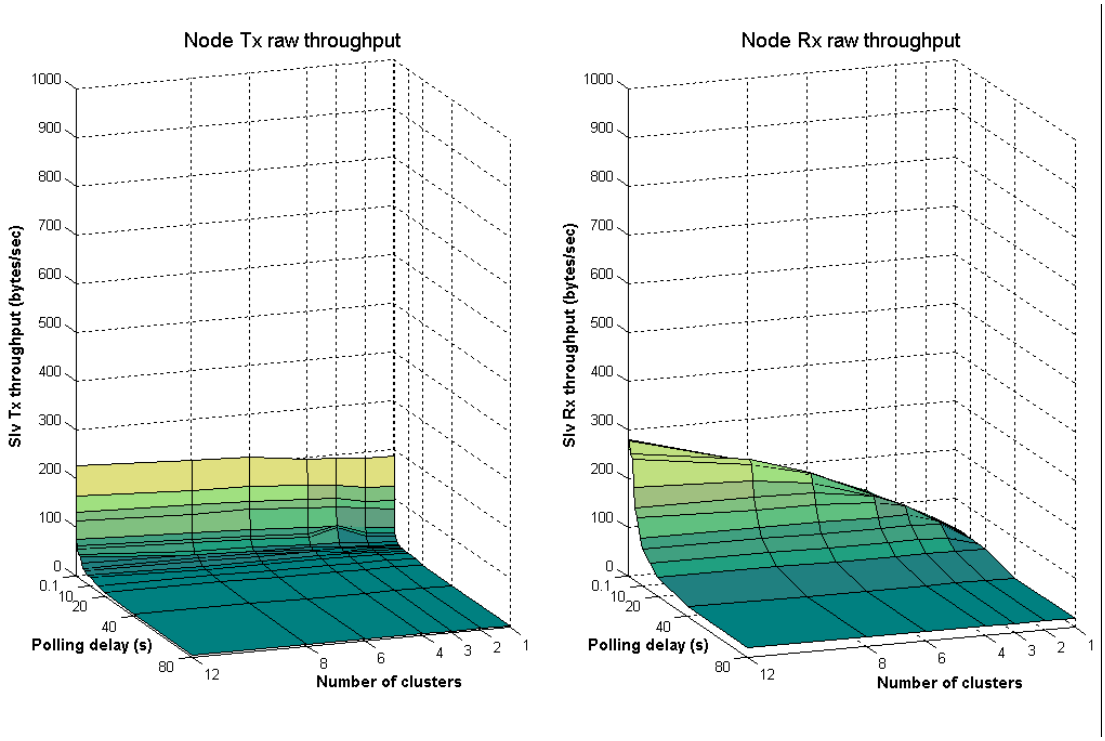


Figure 6.9 : Raw Transmission (Tx) and Reception (Rx) throughputs for the Cluster Nodes.

These graphs show the average network traffic faced by the Cluster Nodes. They do not display the result of the mesh topology experiment, because there are no Cluster Nodes in this configuration; only Cluster Heads.

The outbound traffic of Cluster Nodes consists of small packets which contain either:

- replies to sensory information requests, or
- requests to obtain the sensory values of the entire network of nodes.

The inbound traffic consists of two types of messages:

- small packets containing requests for new sensory data, or
- replies to network poll requests (usually very large packets, containing data samples from all the nodes on the network).

Globally, the nodes' transfer rate constantly remains well below the maximal rates. This is attributed to the request queuing performed by Cluster Heads. The first advantage of request queuing is that it automatically adapts the Cluster Head's polling frequency to the capacity of the Cluster Nodes. Indeed, when the nodes cease to reply new queries are not issued but queued. The second advantage is that one set of information is used to reply several queued queries. This re-uses available data, hence generating less traffic, while replying faster to a larger number of queries.

As expected, the two plots increase as queries are issued more often. When the number of queries rises, the nodes are requested to send information more often, which ultimately raises the throughput. Concerning the reception traffic, the results also appear to be depending on the cluster configuration: the reception throughput decreases when the cluster becomes larger. This is because the nodes receive less learning data sets, as this configuration yields fewer results (see 6.2.4.4).

6.3.4.2 Analysis of the Transmission and Reception Efficiency

The following graphs represent the communication efficiency. The metric corresponds to the ratio between the number of learning sets obtained and the total amount of bytes transmitted or received:

$$\eta_{Tx} = \frac{n_{sets}}{n_{TxBytes}}, \eta_{Rx} = \frac{n_{sets}}{n_{RxBytes}} \tag{6-8}$$

η_{Tx} and η_{Rx} are the transmission and reception efficiency, n_{sets} is the total of learning sets received, $n_{RxBytes}$ and $n_{TxBytes}$ are the amounts of bytes received and transmitted, respectively.

This ratio is interpreted as efficiency because it is proportional to the cost of data acquisition. Indeed, the number of bytes sent and received has a direct correlation with the energy consumed by the communication module. Therefore this graph permits the comparison of different topologies for their communication energy efficiency.

In Figure 6.10, higher values represent the beter efficiencies, while low values indicate a higher cost to obtain information.

A. Cluster Heads Transmission and Reception Efficiency

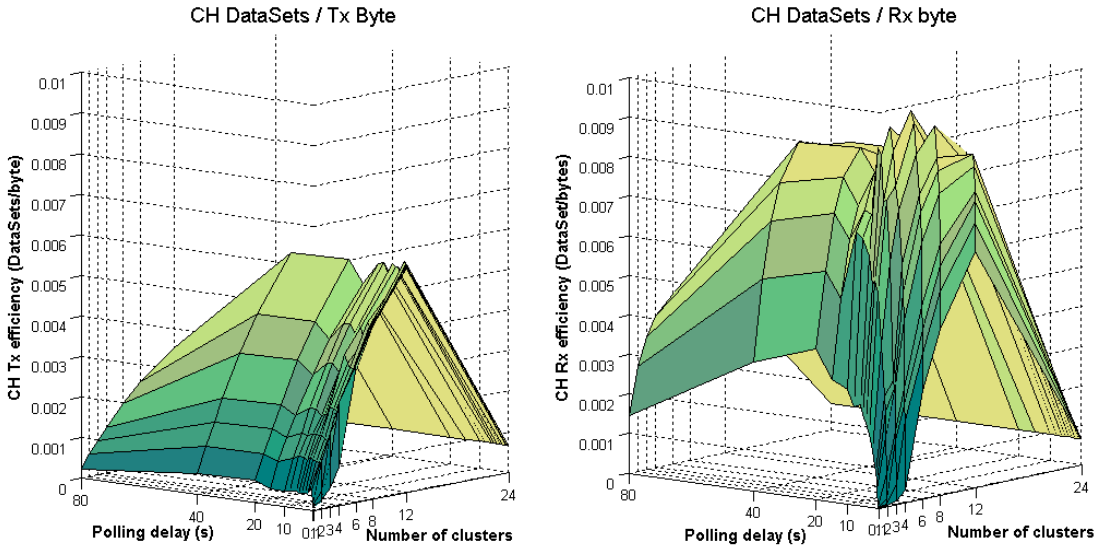


Figure 6.10 Data sets obtained per byte transmitted (left, Tx) or received (right, Rx), for Cluster Head (CH) nodes.

The two graphs in Figure 6.10 represent the cost of information acquisition for the Cluster Head nodes. The graph on the left shows the transmission efficiency and the one on the right efficiency for reception. The efficiencies are plotted against polling frequency and cluster distribution.

The two graphs display a similar shape, but at different scales. Globally, the efficiency increases with polling frequency. When the delay between each query becomes smaller than 40s, the efficiency ceases to increase.

This graph also proves that star topology is clearly the least efficient solution. The efficiency decreases as the clusters become larger, and reaches a minimum when all the nodes form a single cluster (star topology). The best efficiency is obtained by clusters with two nodes. The efficiency results obtained for mesh topology are also very low.

Comparison with the graphs shown in Figure 6.8, shows that the configurations generating small flows of data are also the most efficient ones. Either too large or too small clusters result in a poor efficiency. The best configurations to minimize the cost of data acquisition are the ones with between 4 and 2 nodes per cluster (ie. 6, 8 or 12 clusters for a network of 24 nodes).

The decreasing efficiency observed when delays between queries exceed 40 seconds is due to the fact that the requests are becoming sufficiently spaced to be treated individually. This means that each individual query triggers one complete data acquisition process and the data is not shared, as the queue is empty. This lack of information-sharing reduces efficiency.

When delays are smaller than 40 s, the efficiency reaches a plateau value. This is because smaller delays between queries tend to overwhelm the Cluster Heads. This means that queries are received faster that they can be answered, which results in a greater waste of transmissions and consequently limits the efficiency. Very small intervals between polling queries generates a higher traffic, without improving the efficiency.

B. Cluster Nodes Transmission and Reception Efficiency

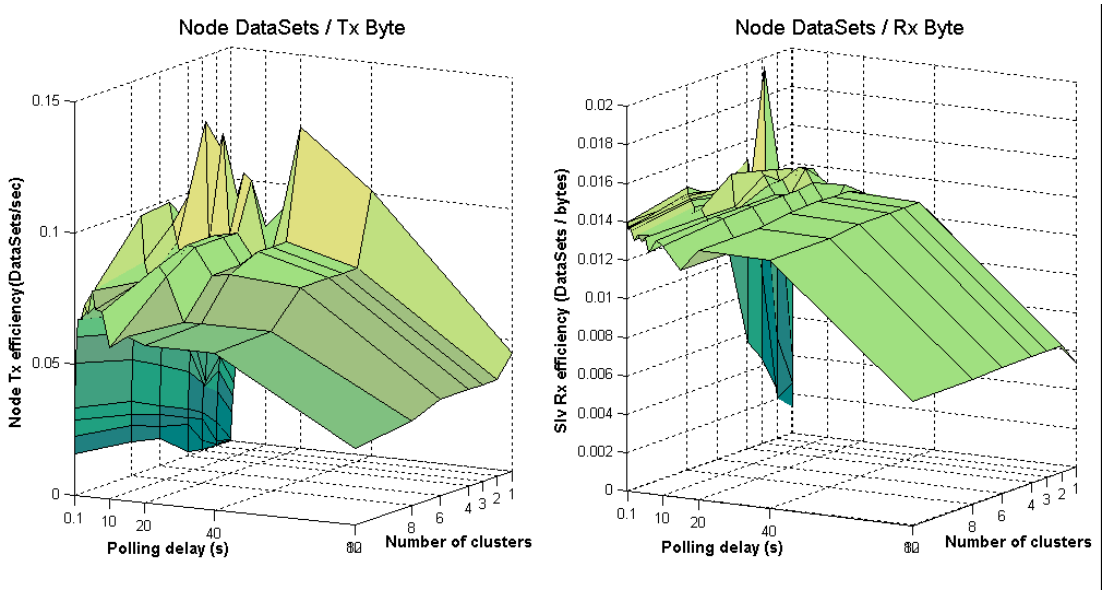


Figure 6.11: Data sets obtained on data transmitted (left, Tx) or received (right, Rx) ratio, for cluster nodes.

The two graphs in Figure 6.11 represent the cost of information acquisition for the Cluster Nodes. The diagram on the left shows the efficiency metric for transmission and on the right the graph for reception. In order to present the best view of the 3D surface graph, the point of view has been changed by reversing the scale of the two parameters compared to Figure 6.10.

Globally, the graphs reveal the same trends as the Cluster Head graphs. The efficiency is ameliorated, because the Cluster Nodes only carry out a minor part of the data acquisition process. The efficiency decreases when the delay between two queries becomes too short or too long. The influence of network configuration, though, is less visible.

The decrease of efficiency on the transmission graph is justified by the performance of the Cluster Head. When the polling delay is reduced, the Cluster Head tends to be overwhelmed by queries. This initiates the transmission of more requests, but still a limited number of data sets are obtained. This is the justification for the decrease in efficiency. On the contrary, when the polling frequency increases

too far, the queries are treated individually, and therefore take longer to complete and are less shared.

On the reception graph, the nodes are protected from excessive queries by the grouping and queuing strategy adopted by their Cluster Heads. Similar to the reception graph, the efficiency remains relatively stable. The only significant decrease occurs with very long or very short polling intervals, particularly in large clusters.

6.3.4.3 Analysis of the Amount of Learning Sets Received

In this section, the results obtained previously are summarized. The diagrams shown in Figures 6.12 and 6.13 display the same information from two different points of views. They present the total number of distinct learning sets acquired over a time of 800 seconds (surface) and the peak efficiency for each network configuration (stems). The variables are plotted against the number of clusters and polling intervals.

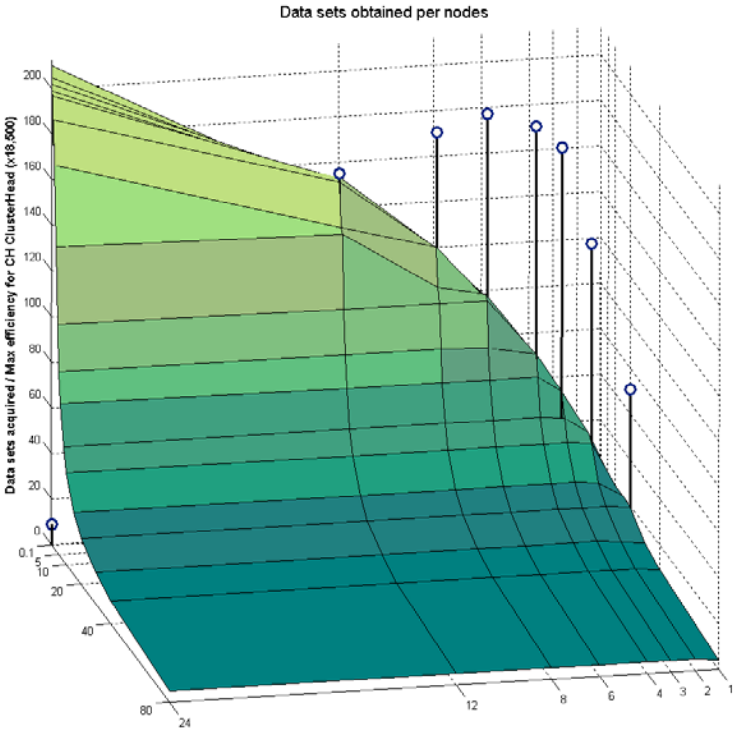


Figure 6.12: Total amount of data sets obtained per Cluster Node (surface) and peak values for data acquisition efficiency.

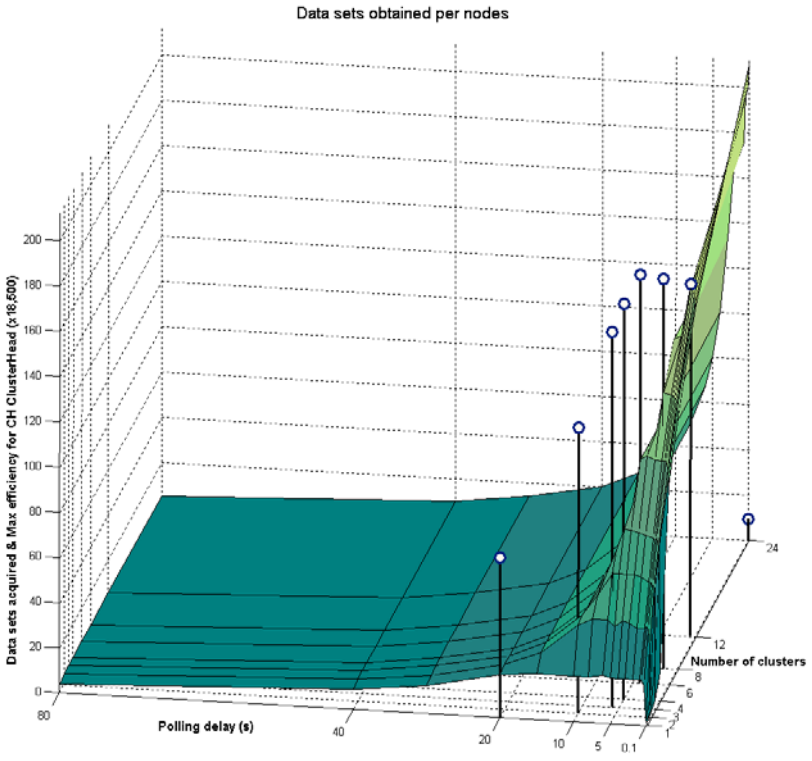


Figure 6.13: Same graph as above, presented from a different angle.

The analysis of the throughput graphs (in 6.2.4.2) clearly revealed that some configurations were beyond the capacity of the system. The simulator produced results because it allowed its buffers to expand without limits. Some results displayed by this graph are not achievable with a real WACNet. The areas of instability are at high polling frequencies, with a very high or very low amount of clusters. These configurations can not be considered.

A. Influence of the *Polling Delay* Variable on the Amount of Learning Sets Collected

Observing the surface graph along the *polling delay* variable's axis, the amount of learning data sets obtained increases as the delay decreases. The diagram peaks between 0.5 and 10 sec between each query, depending on the network configuration. A steep decrease is observed after the peak value, clearly showing that the limits of the system have been reached. When the polling delay becomes too small, the network is flooded with requests which prevent the transmission of information.

When the polling frequency is below the capacity of the system, the data can not be shared by the Cluster Heads and therefore these configurations yield poor results.

B. Influence of the *Number* of *Clusters* Variable on the Amount of Learning Sets Collected

Considering the surface plot along the *number of clusters* variable's axis, it clearly appears that smaller clusters perform better. Nevertheless, it is also important to note that the configuration which yields the best result; the mesh network (24 clusters), is not stable in the areas where its optimal results are observed.

Configurations with large clusters yield the worst results. Indeed, due to an insufficient distribution, the Cluster Heads in charge of large clusters are rapidly overwhelmed. The Cluster Heads of smaller clusters are able to share data and collaborate, hence withstanding higher polling frequencies.

C. Stem Graph and Surface Graph

The stem graph generally shows that the maximal efficiency is inversely proportional to the cluster size, and also occurs at higher polling frequencies when the cluster size decreases. A strong correlation is observed between the maximal efficiency for a network configuration (stems) and the best results in terms of data collection (highest points of the surface). The configurations marked by stems correspond to minimal energy consumption, and also correspond to maximal amounts of data collected.

The stem graph shows that mesh topology has a very low efficiency. Despite this potential to perform well, this configuration would require a larger bandwidth to cope with its inefficiency. The *stable* polling frequencies for this configuration yield results which are well below the results of the other configurations. Since efficiency is more important than the amount of data acquired, mesh topology can not be considered as an ideal configuration.

The graph shows that the configuration acquiring data at the lowest "cost" is the one forming clusters of 4 nodes, with a polling delay of 4 seconds. In this configuration, a Cluster Head must send 158 bytes of data and receive 104Bytes to

acquire a learning set (average values). A new data set is obtained approximately every 8 seconds.

The configuration which collects the largest amount of learning datasets is with clusters of 2 nodes, at a polling delay of 3/4 of a second. It requires the transmission of 175 bytes and the reception of 118 bytes to obtain a dataset. It collects 50% more data sets than the most efficient configuration, at the cost of a 10% drop in efficiency. In this setup, a new set of learning data is acquired, on the average, every 5.5 sec.

6.3.5 Discussion

This section will compile all the observations made previously in order to describe the optimal solution.

The simulation results proved that the polling frequency and number of nodes per cluster have a considerable influence on the efficiency and the throughput of the WACNet.

Large clusters, while being relatively energy-efficient, fail to match the performance of networks with a higher degree of distribution (typically with 3 to 8 nodes per cluster). Clusters of one node (mesh) are to be avoided at all cost, as they generate a potentially overwhelming amount of network communication, and are very inefficient.

The optimal polling frequency depends on the size of the clusters. Therefore it must be adjusted to match the individual characteristics and requirements of the network. Generally, a low polling frequency results in poor transceiver energy efficiency. On the other hand, an excessive polling frequency will overwhelm the Cluster Heads and possibly the entire ZigBee network. Generally, the optimal delay between two queries is from 1 to 10s, for a network of 24 nodes. The general rule is that larger clusters are more efficient when larger delays between two queries are observed.

From the analysis of the previous results, it appears that the best configuration which maximizes both efficiency and total amount of data made available is:

- A network configuration with a cluster size of 4 to 8 nodes,
- A delay between two polling requests between ¾s and 4s.

The simulation model is simplified compared to a real physical system and hence some differences can be observed in its performance. Nevertheless, the simulator illustrates that performance of WACNet for such a task depends widely on the network configuration and the demand for information quantity. This simulator has also validated the data-sharing strategy presented, by proving that it takes advantage of the clustered topology of this network.

Further studies may refine these results, using a simulator which is able to replicate finely the behaviour of the modules used by the nodes to communicate. This would provide accurate data, considering various link speeds, packet loss rates, delays, etc.

6.3.6 Conclusion

In most cases, centralization would generate less network traffic and lower delays, compared to decentralized architectures. This is because it transmits the information directly to the concerned node, without relay. On the other hand, mesh topologies are very scalable, because they maximize the independence and autonomy of the node.

For WACNet, the most important requirement is to minimize the energy consumption of the Cluster Nodes. Maximal throughput and low delays are secondary objectives. The energy consumption of the Cluster Nodes is reduced by decreasing their traffic. Since the WACNet uses a distributed architecture with a protocol supporting data sharing, some part of the network load and task can be shifted to the Cluster Heads. The Cluster Heads can withstand higher network loads, as they are expected to run on a permanent source of energy. Since data is concentrated in the Cluster Heads, not only are the Cluster Nodes less solicited, but also the reply time is reduced. The reply time is reduced because the data is concentrated where it can be rapidly accessed, without systematically initiating a full query process.

The distribution of the data-acquisition process in a clustered network is also very scalable. The results of the simulation demonstrate the incapacity of centralized and decentralized topologies to support large networks. In the star-topology configuration, the central Cluster Head is rapidly overwhelmed. On the other hand, mesh-topology configurations prove to be too complex and too decentralized for the

limited Cluster Nodes. The high degree of individualism of the nodes prevents them from dividing and distributing the querying task to neighboring nodes. Therefore, none of these configurations can scale to large networks. Inevitably, the system will become too large for a single node to query the entire network.

The simulation has shown that, using clusters of variable size, the task can be distributed in a simple and natural way among the clusters. It also highlights the importance of determining the optimal compromise between centralization and decentralization:

- Clusters that are too large overwhelm the Cluster Head. An exaggerated example being the star topology. In a star topology, the Cluster Head deals with a cluster which *is* the entire network.
- clusters that are too small induce a larger number of clusters. This results in more queries between Cluster Heads. In this case, the exaggerated example is the mesh topology. In mesh topology, a node has to query the entire network of nodes, without assistance.

This experiment has proven the benefits of adopting a clustered network topology for a WACNet. It also validated the concept of a data sharing and request queuing protocol.

6.4 Results of the Learning Algorithm

In this section, an experiment using a set of WACNet nodes is set up to demonstrate the feasibility and efficiency of the learning algorithm under real-life conditions.

The results of this test are obtained from a set of nodes which execute the learning algorithm. The nodes are provided with a pre-determined input, assimilated to sensory data. At the end of each experiment, the results are uploaded to a PC using the bridging node, to analyse the results.

This section will begin with a description of the experimental setup. It will be followed by a description of each experiment. The results of each experiment will be

discussed in two separate sections. Finally, conclusions and comments will be made on the learning algorithm and the influence of the parameters used.

6.4.1 Experimental Setup

This section will begin with a general description of the experimental platform and protocol. It will be followed by a description of all the experiments which have been conducted.

6.4.1.1 Implementation of the Learning Algorithm

The results of the Matlab algorithm simulation and other issues raised in preliminary tests have been taken into account when implementing the algorithm on the target. One of the major hindrances of the algorithm simulated in Matlab was its lack of flexibility in the definition of rules. This problem has been solved in this implementation. The problem associated with inability to distinguish relevant from irrelevant rules has also been addressed. Different techniques for rule selection are tested in this experiment.

Although the basic principle of the learning algorithm is the same, the rule table memory organization allows more flexibility. When a new rule is created, sufficient memory is allocated to hold, for each variable, a unique identifier of the sensor and the fuzzy set. This allows a differentiation of each sensor, so that any change in the network configuration has no impact on the acquired rules. This implementation also allows rules with a variable length. Consequently, it leaves a possibility to later implement a rule-optimizing algorithm that can delete unrelated inputs, merge rules, as well as add a new sensor to an incomplete rule. The trade-off for this evolution is that it uses more memory, as the identifier ($@S_x$ in Table 6.1) of each sensor must appear in every occurrence, as shown in Table 6.1.

Table 6.1: Rule table in memory: The address of each sensor is included, and the length is variable.

Occur	Weight	Rule							
3200	95%	@s_1	1	@s_4	9	@s_7	3	@s_4	2
356	75%	@s_6	4	@s_3	7	@s_1	4	@s_9	1
1750	40%	@s_4	1	@s_5	2				

A rule-filtering system has been implemented using a double-stage rule table. A "probation zone" is set up to store the rules temporarily. The probation zone momentarily memorizes an incoming rule, while the system verifies its strength and occurrence frequency. The most relevant rules are selected for promotion to the "reliable zone" of the rule table. The reliable zone stores the valid rules and prevents them from being overwritten by newly-acquired and unverified rules.

6.4.1.2 Data Acquisition

The method used to acquire data in this experiment complies with the network protocol described in Chapter 4.

A. Network Boot-up

Every experiment begins with a network boot-up phase. Although the cluster organisation is pre-determined, the nodes still apply the protocol described in Chapter 4. During this stage, a node is promoted to Cluster Head, while the others remain Cluster Nodes. Following this, the Cluster Nodes advertise to their Cluster Head the type of information they will send (number of sensors and number of fuzzy sets).

B. Transmission of Information and Remote Acquisition

The data is acquired using the data-acquisition protocol described in Chapter 4. The node executing the learning algorithm acquires the learning sets through the

Cluster Head by issuing S requests. The cluster Head polls the entire cluster, and returns the learning set to the requesting note in a single frame.

C. Data Upload

A request is added to the protocol in order to obtain a specific rule from the table. A frame with the symbol 'T' is sent to the node along with an identifier number *n*. The node fetches the n^{th} rule in the table, and writes it to an outbound frame. The frame is then forwarded to the requesting node.

The rule table requests are ordered from the monitoring station, and the T frames are sent from the bridging node. The reply is printed on the monitoring station's interface and is also written in the log file of the bridging node. At the end of each experiment, a routine collects the rules one by one and writes them in a log file on the Bridging Node. The log file can be downloaded to the PC using FTP to analyze the results.

6.4.1.3 Benchmark Description and Network Configuration

The algorithm presented in the previous section is implemented on a WACNet consisting of 4 Cluster Nodes and one Cluster Head.

An environment is recreated in order to provide sensory data to the nodes. The nodes acquire data from their sensor port. The sensor port of each node is connected to an emulator. The emulator is a microcontroller which is programmed to replicate one sensor for each node. The total of four sensors emulated comprise of two binary "on/off" switches, and two devices returning values between 0 and 254.

The environment emulator executes a code which produces a semi-random set of states. A new state is generated every 2 seconds. The total number of combination of the system's variables formulates 36 fuzzy rules. Among these 36 possible rules, only a limited number match the few states emulated by the microcontroller. The different states presented to the system carry different probabilities and match rules with various strengths. This is done to evaluate the reaction of the learning system to rare events or events which can not be clearly defined.

A timer triggers the transmission of S frames every two seconds. Since there is only one cluster, the Cluster Head simply forwards the request to all the nodes in the

cluster. When requested, the Cluster Nodes read the data present at their sensor port, process it to fuzzy data, and transmit this fuzzified data to the Cluster Head.

When all the nodes in the cluster have replied, the Cluster Head compiles all the sensory data into a single frame. This frame is sent back to the requesting node which issued the request.

The queuing system described in Chapter 4 (and also used in the simulation) is implemented here. Only one full query can be processed at a time. Similar requests arriving before a full query is completed, are queued. When the data is acquired, all the nodes in the queue are replied to with the same data set. The Cluster Head is also set to abandon a request if one or many nodes fail or do not reply in time. This reply timer prevents the process from infinitely waiting for a reply, when a node becomes silent.

6.4.1.4 Experimental Description

A. Control Test

In order to determine:

- the probability of an event firing a specific rule, and
- its associated strength,

a control test is performed. This test will serve as a reference when evaluating the results of the algorithm running in selective mode.

The system is left running without memory space constraints until a total of 3,000 global rule-match occurrences are reached. During this control test, every matching rule is stored in the rule table, along with its specific metrics.

B. Selective Mode Experiment

Two experiments are conducted in selective mode operation. This means that the rule table is limited in size, and the rules created by the algorithm are filtered. The aim is to select only the most relevant and frequent rules, while discarding the rules which do not significantly represent the system. This selection process is based on competition. The rule selection system determines the best rules by letting them compete for a slot in the reliable zone. The strongest rules are protected, while the

weakest ones are discarded. This concept accelerates convergence, whilst leaving the system open to new rules reflecting changes in the environment.

In this experiment, the learning agent allocates "memory slots" for up to 20 rules. In order to perform selection, a two-level rule table is introduced. Among the twenty available slots, all new or recent rules will stay in one of the 12 slots available in the probation zone. The rules in probation zone must reach a threshold value of firing occurrences (or strength) to access the reliable zone. If the rule fails to reach the threshold value, it will inevitably be overwritten by a new incoming rule. In this case, the threshold value for access to the reliable zone is 15 firing occurrences. The rules which have reached the reliable zone will be considered by the system as relevant.

The diagram in Figure 6.14 illustrates the major steps of this procedure.

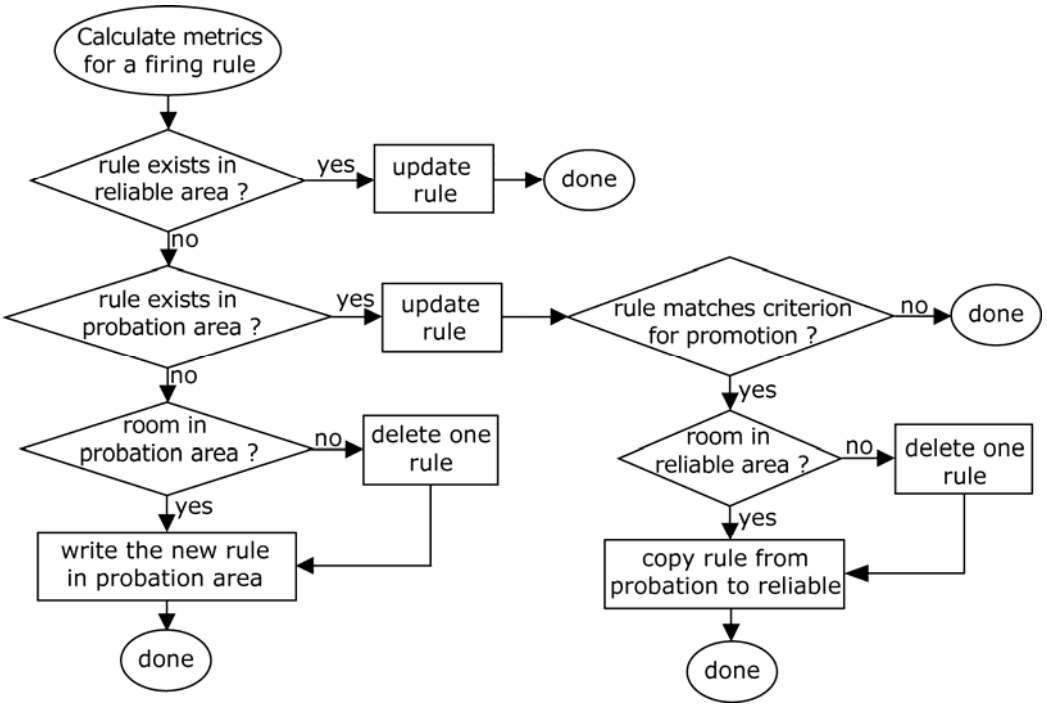


Figure 6.14: Rule acquisition process flow diagram.

Two metrics can be used to determine the relevance of a rule. These metrics are firing strength and number of firing occurrences. The effects of using these metrics as eviction criterion are compared in the two following experiments. The experiments are terminated when over 1,000 rule matches occurred.

6.4.2 Results Description

This section will present, describe and justify the results obtained from the experiments described above.

6.4.2.1 Control Test

The reading of the control test in Table 6.2 shows that all the 36 possible rules were fired at least once. *Valid rules* are rules which match one of the events generated by the emulator. The 19 rules with a darker background in Table 6.2 correspond to valid rules. The valid rules are classified into the three following categories:

Category 1. A unique state perfectly matching a single rule (firing strength of 97%), which fires very frequently (probability = 0.15). The rule matching this state is the first one in Table 6.2.

Category 2. A set of events corresponding to 2 categories, each potentially matching 9 rules. The rules in this group are fired less often ($0.04<p<0.06$) and have a lower average firing strength (from 15% to 50%). These events are considered as the most common state of the system. Indeed, the global probability to fire one of the rules in this group is 0.75.

Category 3. A rare and unique state with a probability below 0.02, but which also perfectly matches a single rule, with an average firing strength of 90%.

Apart from valid rules, this control test also marked rules that do not apply to any state programmed in the event emulator. This occurs when the nodes do not receive the order to read their sensor at the same time. If the state of the system changes during the acquisition process, the results will be incorrect. Some nodes will read the value just before the change, and some will read just after. The result is a learning dataset which represents an erratic combination of two distinct states. These undesirable data sets fire the 16 remaining fuzzy rules (called *error rules*). They correspond in Table 6.2 to the rows in white background. They sometimes reach a significant weight (up to 47%), but usually remain quite rare. Indeed, all error rules (except one) show a firing probability inferior to 0.002. The overall probability to acquire a sample which fires an error rule is 0.08.

Table 6.2: The 36 rules acquired during the control set.

weight	Occur.	Prob.	S1	S2	S3	S4
97%	447	14.7%	0	0	1	1
52%	190	6.3%	0	0	0	1
47%	186	6.1%	2	0	0	0
38%	185	6.1%	0	2	0	0
35%	166	5.5%	2	2	0	1
24%	146	4.8%	0	0	0	0
18%	128	4.2%	0	1	0	0
26%	127	4.2%	1	0	0	0
18%	127	4.2%	0	2	0	1
19%	122	4.0%	2	2	0	0
24%	121	4.0%	1	1	0	0
18%	121	4.0%	2	0	0	1
17%	121	4.0%	1	2	0	0
24%	120	4.0%	0	1	0	1
24%	119	3.9%	1	0	0	1
24%	119	3.9%	2	1	0	0
17%	119	3.9%	1	2	0	1
24%	116	3.8%	1	1	0	1
16%	115	3.8%	2	1	0	1
90%	57	1.9%	0	0	1	0
47%	11	0.4%	2	0	1	1
22%	8	0.3%	1	0	1	1
7%	7	0.2%	0	1	1	1
7%	7	0.2%	0	2	1	1
34%	6	0.2%	2	0	1	0
18%	5	0.2%	0	2	1	0
12%	5	0.2%	1	0	1	0
8%	5	0.2%	1	1	1	1
6%	5	0.2%	1	2	1	1
13%	4	0.1%	0	1	1	0
9%	4	0.1%	2	1	1	1
7%	4	0.1%	2	2	1	1
2%	3	0.1%	1	1	1	0
2%	3	0.1%	1	2	1	0
2%	3	0.1%	2	1	1	0
2%	3	0.1%	2	2	1	0
total	**3035**					

Table 6.2 summarizes the results of the control set. The rules are displayed on a row. The first value in a row is the average firing strength. The second column displays how many times this rule was fired (ordering column). The next value is the probability of firing (in percentage). The probability of firing is obtained from the division of the value in the previous column by the total number of occurrences (3,035). The four remaining values correspond to the fuzzy set (0→low, 1→average, 2→high) of the four sensors: S1, S2, S3 and S4.

The darkest row in Table 6.2 is the rule describing the category 1 event, the grey rows correspond to the rules matching the various category 2 states, and the light grey row represents the rule fired by the category 3 events. The remaining rows are error rules.

6.4.2.2 Rule Weight Selection Experiment

Table 6.3 presents the rules found in the reliable zone after processing learning sets using rule weight as selection criterion. The rule number (field Rule #) is an identifier attributed to the rule by order of appearance. The table is sorted by descending weight. The coloured boxes represent data from the control set for comparison purposes.

Table 6.3: Printout of the reliable zone of the rule table, after running an experiment with rule firing strength as selection criterion.

Rule #	Weight	Weight (ctrl)	Occur	Rule				Freq. (ctrl)	Case
1	98	97	244	0	0	1	1	14,7%	case1
7	95	90	31	0	0	1	0	1,9%	case4
4	55	52	97	0	0	0	1	6,3%	case2
3	49	47	85	2	0	0	0	6,1%	case3
5	43	38	96	0	2	0	0	6,1%	Case3
2	39	35	82	2	2	0	1	5,5%	Case2
0	33	24	10	1	0	0	1	3,9%	Case2
6	31	24	13	2	1	0	0	3,9%	Case3

This version of the algorithm selects rules which match most accurately the states of the system, regardless of how often they appear. Indeed, rules with a probability as low as 0.019 (rule #7) have been selected. Rules with a firing strength below 39% (Rule #2) do not appear in the results.

6.4.2.3 Rule Strength Selection Experiment

Table 6.4 presents the rules found in the reliable zone after running the learning algorithm using firing occurrences as selection criterion.

Table 6.4: Printout of the reliable zone of the rule table, after running an experiment with firing occurrences as selection criterion.

Rule #	Weight	Weight (ctrl)	Occur	Rule				Freq. (ctrl)	Case
1	97	97	396	0	0	1	1	14,7%	case1
5	52	52	159	0	0	0	1	6,3%	case2
3	51	47	151	2	0	0	0	6,1%	case3
4	39	38	151	0	2	0	0	6,1%	case3
2	43	35	159	2	2	0	1	5,5%	case2
0	17	18	12	0	1	0	0	4,2%	case3
7	16	18	97	0	2	0	1	4,2%	case2
6	25	24	97	0	1	0	1	4,0%	case2

This selection criterion selects only the most frequent rules, regardless of the weight. The lowest weight is 16% (Rule #7). The minimal weight in the previous experiment is almost twice as large. Rules which are fired less than 4% of the time are filtered. This value is the double of the one observed in the rule weight selection experiment.

6.4.2.4 Reliability and Delay Tests

An experiment is set up to verify the reliability of the system. In this experiment, a Cluster Head is set to issue a total of 30 queries, at the rate of one query every 2s. The replies are counted, and the delay is measured. This experiment is repeated several times to get the most reliable results, but a standard value proved to be difficult to determine, because of a high standard deviation.

The results of these experiments reveal that the reply delay for an S query generally varies between 500 and 800ms.

Failures to reply occur in a totally unpredictable way. The average success rate of queries is measured around 96%.

6.4.3 Discussion

6.4.3.1 General Accuracy

The results of the experiments using rule selection systems are compared against the results of the control test. Both of the experiments measured correct values, and accurately performed a selection based on the metrics defined (rule firing strength or rule firing occurrence). The results show that the system is able to determine and select which rules are relevant, given a metric to assess relevancy.

6.4.3.2 Influence of the Selection Metric

The test using the firing occurrences as a metric for rule eviction only promotes the rules which fire strictly more than 4% of the time. In this case, it selects rules with a firing strength of 18%, which are not the most significant for result computation. It mostly consists of the rules from the categories 1 and 2, while the category 3 rule has not been promoted.

The test using rule weight as the eviction criterion keeps only the rules with a weight greater than 25%. This test also selects the category 3 rule, despites its very rare frequency of occurrence.

Using the rule weight or rule firing occurrence as a metric for rule eviction gives the system a specific orientation:

- Rule firing occurrences selection focuses on the most common situations. It is capable of defining accurately the most frequent states of the system. On the other hand, it is incapable of "remembering" rare events.
- Rule weight selection builds a rule table which is better capable of dealing with all situations found in the learning data sets. Generally it will give slightly more imprecise results, but the accuracy will depend less on the event's probability. In other words, this rule table will always provide a "rough" definition of most scenarios it encounters.

For the application considered (*i.e.* human behavior learning), the use of rule weight for selection is preferable. Precision in the forecast is not a major requirement of the system. This is because the consumption values are expected to fluctuate, depending on the human's rather unpredictable behavior. It is certainly more important to be able to predict if a significant consumption of resources is involved with *any* scenario, rather than being capable to quantify precisely the consumption only for the most usual cases. Therefore the policy agents will prefer a rule table allowing them to react to any event, even if the tradeoff is the accuracy. The system has to handle every single behavior of the users, even the less common ones.

6.4.3.3 Influence of Data Acquisition Latency

The control test reveals that the algorithm generates rules which do not represent any state produced by the emulator. This is attributed to non-uniform transmission delays, which causes the nodes to receive the polling request at a slightly different time. Although this type of error has a very low probability of occurring and is successfully filtered in both experiments, these learning sets are incorrect data to the learning algorithm.

The dual zone rule table has proven to be an efficient filter against these erratic rules. Indeed, not a single error rule appeared in the reliable zone. This is because the errors do not happen often enough to even secure a stable slot in the *probation* zone.

Although the filtering appears to be efficient in this experiment, it is important to minimize the probability of acquisition of erratic datasets. This can only be done by ensuring that the data acquisition order is received simultaneously by all the nodes. On larger networks, the clustered topology and task distribution scheme will allow a parallel processing of the request which will minimize the effects of non-uniform latencies.

6.4.4 Conclusion and Further Requirements

This test has proven that the learning algorithm can acquire a set of rules which describe a system, after a short period of learning. It has also proved successfully that it is suitable for microcontroller targets. This selection system correctly filters the

“noise”, assimilated with undesirable rules. The rule selection technique retains only the most relevant rules, thus simplifying the computation of an output and saving memory space while converging faster and maintaining an acceptable accuracy.

Yet this test reveals the importance of latencies for the accuracy of learning. The impact of latency should be considered on larger multi-cluster networks.

From the early definition of this algorithm, the large memory constraints have been highlighted. The flexible rule table introduced in this chapter also increases the memory size requirements. It is highly likely that the microcontroller’s memory will be insufficient as the network grows in complexity and size. Solutions including the use of external memory should be considered. Techniques to merge and optimize the rules must also be considered and tested.

Finally, one of the major hindrances of this test is the lack of a real data set. Although the learning algorithm is tested on a real network, the dataset generated by the emulator is artificial and pre-determined. The learning data encountered by an agent in a real situation will has a more complex pattern and features and less frequent variations than the one used for the simulation. Further studies should consider implementing the algorithm within a real-life experimental set up.

Chapter 7 Conclusions

7.1 Introduction

The concept of WACNet was introduced in 2004 towards creating a new generation of highly distributed wireless control networks for applications beyond factory automation. Further study and experimentations were carried out in this thesis to realize the concept and to demonstrate its viability and effectiveness.

During the course of this study, the WACNet framework as a platform for Home Ambient Intelligent systems was validated. In spite of extensive experimentation, both the application and the platform will require further work before a commercially viable proposal is developed.

The primary focus of this chapter is on critically evaluating the work carried out and the outcome produced. In particular, the strengths and limitations of the findings of the study on the WACNet concept and its application in a home ambient control system will be reviewed. Conclusions will be drawn and suggestions for future development of the project will be provided.

7.2 Feasibility of the Concept for Large, Real-World Applications

A brief review of the control networks history was carried out in the thesis, with a special focus on the new emerging control networks. The large number of research projects reported in the literature with a focus on control networks demonstrate the significance and popularity of this field of research.

Through comparison with other concepts and systems, it was realised that the Wireless Ad-hoc Control Networks offered easier installation and upgrade, lower capital and maintenance cost, and a more efficient and simpler architecture, overall resulting in more efficient implementation of a control system.

It was also shown that Wireless Ad-hoc Control Networks were a better alternative to wired networks for low-cost applications or in cases where wiring was not cost efficient or simply impossible. WACNets effectively compensate their relatively slow rate of communication by adopting an optimal organization, and distributing the work load in an intelligent way. By maintaining the communication at a optimal level, the power consumption by the nodes is minimised. WACNets are particularly fit for environment monitoring and control, especially in buildings, agricultural fields, rescue and safety systems, and more generally applications where the installation and maintenance of a network infrastructure is not practical.

While the clustered topology proved to minimize the traffic in any configuration, the impact of the number of nodes on the ZigBee network could not be studied due to a limited number of nodes available. A description of the principle of operation and fundamental modules of a Home Ambient Intelligent system has been proposed, but only the most fundamental and critical modules have been studied and validated due to the limited scope of this project.

7.3 WACNets for Home Ambient Intelligent Systems

The real-world application developed in this work is designed to use the WACNet at its full capacity. The Home Ambient Intelligent system which was used as a case study for WACNet required intensive computation to perform the learning algorithm, and constant communication in a one-to-many points. The WACNet successfully provided the resources necessary for the execution of this demanding application. The major strengths of the WACNet for this application were identified as follows:

1. Low cost of the overall solution, especially with the use of “off-the-shelf” components.
2. Simplicity of installation and use, achieved through learning algorithms, self-organization and self-healing network.
3. Clustered topology allowing a natural distribution of the task, faster processing and battery lifetime maximization.
4. Ability to function without a central coordinator.

This study has not considered the constraints of consumption and size of the nodes. The nodes developed for this research were designed for convenience of experimentation, and not to minimise their physical size. There were no tests carried out to measure the power consumption of the nodes. This was not practical in the deployed experimental rig as the nodes were designed for study of the temporal aspects of the WACNets rather than power consumption.

7.4 Validation Process

A series of computer simulations were carried out to examine characteristics of the WACNet concept. The capabilities of the nodes and the effectiveness of the selected network topologies were verified during validation. In the final stage of the process, the performances of the learning algorithm (normally intended for resourceful systems) on a group of WACNet nodes was verified.

The validation process was carried out in three stages. The first one was a benchmark test of the upgraded node, using two nodes and various measuring instruments. This test mainly focused on determining the performances of the ZigBee modules. The second test used a purpose-built JAVA Discrete Event Simulator to verify the validity of the WACNet topology. The program ran simulations of the WACNet in clustered and other common topologies, for comparison purposes. The final test was a real-life test of the WACNet, using five nodes developed for this project, fulfilling the learning task required to realize a Home Ambient Intelligent system.

The validation process proved that the ZigBee standard is indeed well suited for WACNets. It offers adequate communication speed and is designed for cluster topology networks. ZigBee is also a standard open protocol, supporting the heterogeneous nature of this type of network. Another main advantage over Bluetooth is the full implementation of all aspects of routing in the protocol stack.

The simulations proved that the clustered topology is more efficient than the two other topologies considered (i.e. star and mesh topology), for the task presented. The results of the simulation validated the cluster topology, and also determined the

influence of the *number of node per clusters* variable on network efficiency and throughput. The experiments thoroughly explained the influence of the number of nodes per cluster on the network's performances and concluded that the value of this parameter had to be determined and adjusted based on the requirements of each particular application.

Finally, the real-life application proved the suitability of the platform to run an algorithm requiring significant levels of communication and computation considering a microcontroller-based platform. The network performed well for several hours, continuously communicating and performing the algorithm with a relatively high refreshing rate. This test also revealed interesting aspects of running soft computing algorithms on a distributed network of nodes with a limited computational capacity and small memory size.

7.5 Future Work

The complete development of a commercially viable Wireless Ad-Hoc Control Network is a long term project, and many questions still remain unanswered at the end of this study. The work carried out in this thesis has provided essential information on the potential of WACNets for real-life applications. Nevertheless, in order to reach a complete and commercially-viable system, further study should focus on the following areas.

a) The performances of ZigBee under heavy network loads should be assessed. This can be done with a network simulator. This step should focus on metrics such as throughput, delay per hop and packet losses with size and configuration of the WACNet as parameters.
b) This study has demonstrated that the memory size of the WACNet nodes is not suitable for control tasks which require the storage of large amounts of data. Further development should be done to overcome this hardware limitation.

c) A miniaturized WACNet node must be developed. This will determine whether or not a WACNet node can be reduced to the size of a low-cost, small-size chip to be fitted in various appliances.
d) The battery lifetime of a miniaturized WACNet should be studied. A major requirement of the WACNet is to necessitate minimal maintenance. A section of the WACNet research should be dedicated to quantization and possibly reduction of the node's energy consumption.

This study also introduced the concept of Home Ambient Intelligent system for energy consumption management purposes. The fundamental concepts and architecture were presented, but there are further developments required before a commercially viable system is produced, including:

a) The performance of the learning algorithm with a larger number of nodes and real-life data should be verified instead of machine-generated data.
b) Policies and policy agents should be developed.
c) The Home Ambient Intelligent system in a real-life setup should be examined.

References

[1] Jeremy Colandairaj, William Scanlon, George Irwin "*Understanding wireless networked control systems through simulation*", IEE computing and engineering, april/may, 2005.

[2] Martin Leopold, Mads Bondo Dydensborg, Philippe Bonnet, "*Bluetooth and sensor networks: a reality check*", Conference On Embedded Networked Sensor Systems, Proceedings of the 1st international conference on Embedded networked sensor systems (2003).

[3] Ramamurthy, H.; Lai, D.; Prabhu, B.S.; Gadh, R. **"***ReWINS: a distributed multi-RF sensor control network for industrial automation*", IEEE Wireless Telecommunications Symposium, 2005.

[4] "control system." Encyclopædia Britannica. 2007. Encyclopædia Britannica Online. 16 Dec. 2007 http://www.britannica.com/eb/article-9026081.

[5] Shengrong Bu Naghdy, F. (2005) *Wireless ad-hoc control networks,* 3rd IEEE International Conference on Industrial Informatics (INDIN '05). pp 839- 844.

[6] Shengrong Bu Naghdy, F. (2005) *Service Discovery in Wireless Ad-hoc Control Networks*, Proceedings of the 2005 International Conference on Intelligent Sensors, Sensor. Networks and Information Processing Conference, 2005. pp: 157- 162.

[7] "XBee® ZNet 2.5 OEM RF Modules – Digi International – " Digi White paper http://www.digi.com/pdf/wp_zigbee.pdf. 2007.

[8] Jianliang Zheng Lee, M.J. "*Will IEEE 802.15.4 make ubiquitous networking a reality?: a discussion on a potential low power, low bit rate standard*" IEEE Volume: 42, Issue: 6 pp: 140- 146, June 2004.

[9] Callaway, E. Gorday, P. Hester, L. Gutierrez, J.A. Naeve, M. Heile, B. Bahl, V., "*Home networking with IEEE 802.15.4: a developing standard for low-rate wireless personal area networks*" IEEE Communications Magazine, Volume: 40, Issue: 8 pp: 70- 77, Aug 2002.

[10] "Z-WaveAlliance.org - Alliance -- Start"Z-Wave alliance homepage, http://www.z-wavealliance.org/ . 2007.

[11] "Zensys – Zensys" Zensys homepage, http://www.zen-sys.com/

[12] DENNY Mark "*Watt steam governor stability*", European journal of physics vol. 23, n°3, pp. 339-351, 2002.

[13] Chee-Yee CHONG, Srikanta P. Kumar "*Sensor Networks: Evolution, Opportunities, and Challenges*" Proceedings of the IEEE, Vol. 91, No.8, August 2003

[14] Robert Patzkea , "*Fieldbus basics*" ,Computer Standards & Interfaces Volume 19, Issues 5-6, 15 October 1998, Pages 275-293.

[15] Jansen, Dirk; Buttner, Holger, "*Real-Time Ethernet the EtherCAT solution*", Computing & Control Engineering; Feb/Mar2004, Vol. 15 Issue 1, p16-21.

[16] Jack Quinn, "*Serial interfaces*", Digital Data Communication, pp104, Chp 6, 1995.

[17] "*IEEE Standards Wireless Zone –overview–*" http://standards.ieee.org/wireless/overview.html. 2007.

[18] Nick Baker "*Zigbee and Bluetooth, strengths and weaknesses for industrial applications*", IEE Computing and Control engineering, April/May 2005, pp 20-25.

[19] Geer, D "*Users make a Beeline for ZigBee sensor technology*" IEEE Computer, Volume 38, Issue 12, Dec. 2005 pp:16 – 19.

[20] G Ferrari, P Medagliani, S Di Piazza, M Martalò "*Wireless sensor networks: performance analysis in indoor scenarios*", EURASIP Journal on Wireless Communications and Networking, 2007.

[21] ZigBee Alliance Newsletter "*New ZigBee Smart Energy Profile Delivers Efficiency and Savings*", January 22, 2008, http://zigbee.org/en/press/zigbeealliancereleases.asp, 2008.

[22] ZigBee alliance, **"***ZigBee Specification*", ZigBee Alliance Download page: http://zigbee.org/en/spec_download/zigbee_downloads.asp, 2008.

[23] Alan Mainwaring, David Culler, Joseph Polastre, Robert Szewczyk, John Anderson "*Wireless sensor networks for habitat monitoring*" Proceedings of the 1st ACM international workshop on Wireless sensor networks and applications, 88 – 97, 2002.

[24] G. J. Pottie, W. J. Kaiser "*Wireless integrated network sensors*" Communications of the ACM, Volume 43 , Issue 5 pp: 51 – 58, May 2000.

[25] Merrill, W.M., Newberg, F., Sohrabi, K., Kaiser, W., Pottie, G., "*Collaborative networking requirements for unattended ground sensor systems*", Proceedings of the IEEE Aerospace Conference, Volume: 5, pp: 5_2153- 5_2165, March 8-15, 2003.

[26] Deborah Estrin, Ramesh Govindan, John Heidemann, Satish Kumar, "*Next century challenges: scalable coordination in sensor networks*", Proceedings of the 5th annual ACM/IEEE international conference on Mobile computing and networking, pp 263 - 270, 1999.

[27] Ramamurthy, H. Lai, D. Prabhu, B.S. Gadh, R., "*ReWINS: a distributed multi-RF sensor control network for industrial automation*" proceedings of the IEEE Wireless Telecommunications Symposium, pp 24-33, 2005.

[28] KAHN Joseph M.; KATZ Randy Howard; PISTER Kristofer S. J. "*Emerging challenges : Mobile networking for Smart Dust*" Journal of communication and networks, vol. 2, n^{o}3, pp. 188-196, 2000.

[29] M.C. Monteleone, H. Yeung and R. Smith, "*A review of Ancient Roman water supply exploring techniques of pressure reduction*" Water Science & Technology: Water Supply Vol 7 No 1 pp 113–120 © IWA Publishing 2007 .

[30] Bushby S.T. "*BACnetTM: a standard communication infrastructure for intelligent buildings*", Automation in Construction, Volume 6, Number 5, , pp. 529-540 September 1997.

[31] Damsker, D.J. "Towards advanced concurrency, distribution, integration, and openness of a power plant distributed control system (DCS)", IEEE Transaction on Energy Conversion, Volume: 6, Issue: 2 pp: 297-302, Jun 1991.

[32] Delfino, B., Pinceti, B., "*Fieldbus applications for electrical industrial systems*", Conference Record of the 1993 IEEE Industry Applications Society Annual Meeting, 2084-2090, vol.3, 2-8 Oct 1993.

[33] Nitaigour Premchand Mahalik "*6.8 Contemporary Programmable Logic Controllers*" in "Fieldbus Technology: Industrial Network Standards for Real-Time Distributed Control", Springer, pp.152-3, Published 2003.

[34] Open System Interconnection standard at International Institute for Standardization, http://standards.iso.org/, 2007.

[35] "IEEE Application Guide for Distributed Digital Control and Monitoring for Power Plants", IEEE Std 1046-1991 , http://ieeexplore.ieee.org/iel1/2291/4098/00159083.pdf?tp=&isnumber=4098&arnumber=159083, 2007.

[36] Min, R. Bhardwaj, M. Seong-Hwan Cho Shih, E. Sinha, A. Wang, A. Chandrakasan, A., "*Low-power wireless sensor networks*" Fourteenth International Conference on VLSI Design, pp: 205-210, 2001.

[37] "*Ember Press Release: ZigBee slashes hotel energy costs by up to 40 percent*" Ember press release, http://ember.com/press_release.html?id=94, 2008.

[38] "*WiSuite | The Freedom and Simplicity of Wireless Control*" WiSuite product website http://www.wisuite.com/, 2007.

[39] "*ZigBee August External Newsletter*", ZigBee alliance newsletters page, http://www.zigbee.org/en/newsletters/2007_08-Newsletter/0807-external.asp#spotlight, 2007.

[40] "*Ember Press Release: Siemens launches Ember-enabled wireless building automation system*", Ember press release http://ember.com/press_release.html?id=62, 2007.

[41] "*ZigBee October External Newsletter*", ZigBee alliance newsletters page, http://www.zigbee.org/en/newsletters/2007_10-newsletter/1007-External.asp#spotlight, 2007.

[42] Otsuka, Y. Shimizu, N. Yagiu, R., "Home network remote controller using the appliance integrated script engine", International Conference on Consumer Electronics, 2006 (ICCE '06). Jan. 2006 pp: 249- 250.

[43] IEEE 1451, Draft Standard for a Smart Transducer Interface for Sensors and Actuators, http://ieee1451.nist.gov/intro.htm, 2007.

[44] Shengrong bu "*Wireless Ad-hoc Control Networks*", MSc thesis, pp: 13-28, Chp. 2. 2005.

[45] Michael H. Coen, 1997. "*Building Brains for Rooms: Designing Distributed Software Agents*". In Proceedings of Ninth Conference on Innovative Applications of Artificial Intelligence.

[46] Ueli Rutishauser, Josef Joller, Rodney Douglas 2005 "*Control and Learning of Ambience By an Intelligent*", IEEE Transactions on Systems,Man and Cybernetics, Part A, Vol. 35, No 1.

[47] J. K. W. Wong, H. Li and S. W. Wang January 2005 "*Intelligent building research: a review*", Automation in Construction, Volume 14, Issue 1, Pages 143-159.

[48] Robathen P., "*Intelligent buildings guide*", Intelligent Buildings Group and IBC Technical Services Limited, 1989.

[49] S. Sharples, V. Callaghan, G. Clarke, May 1999, "*A Multi-Agent Artchitecture For Intelligent Building Sensing And Control*". International Review Journal, Volume 19, Number 2, pp. 13-14(2).

[50] "*Data Radio Modems and Radio Modules - Digi International*", MaxStream website http://maxstream.net, 2007.

[51] Botia,J.A., Barbera H.M. and Skarmenta A.F.G, "Fuzzy Modeling in a Multi-Agent Framework for Learning in Autonomous Systems". *In* Russo M., Jain L.C. *Fuzzy learning and applications*, pp.93-145, 2001.

[52] "*ERE CO.,LTD*", ERE manufacturer website. Refer to the 'EMBMEGA32C16' product. http://ere.co.th/ . 2007.

[53] Ilona Jagielska, Chris Matthews, Tim Whitfort, "An investigation into the application of neural networks, fuzzy logic, genetic algorithms, and rough sets to automated knowledge acquisition for classification problems". Neurocomputing, Volume 24, Number 1, February 1999 , pp. 37-54

Appendix A: Human to Machine Interface

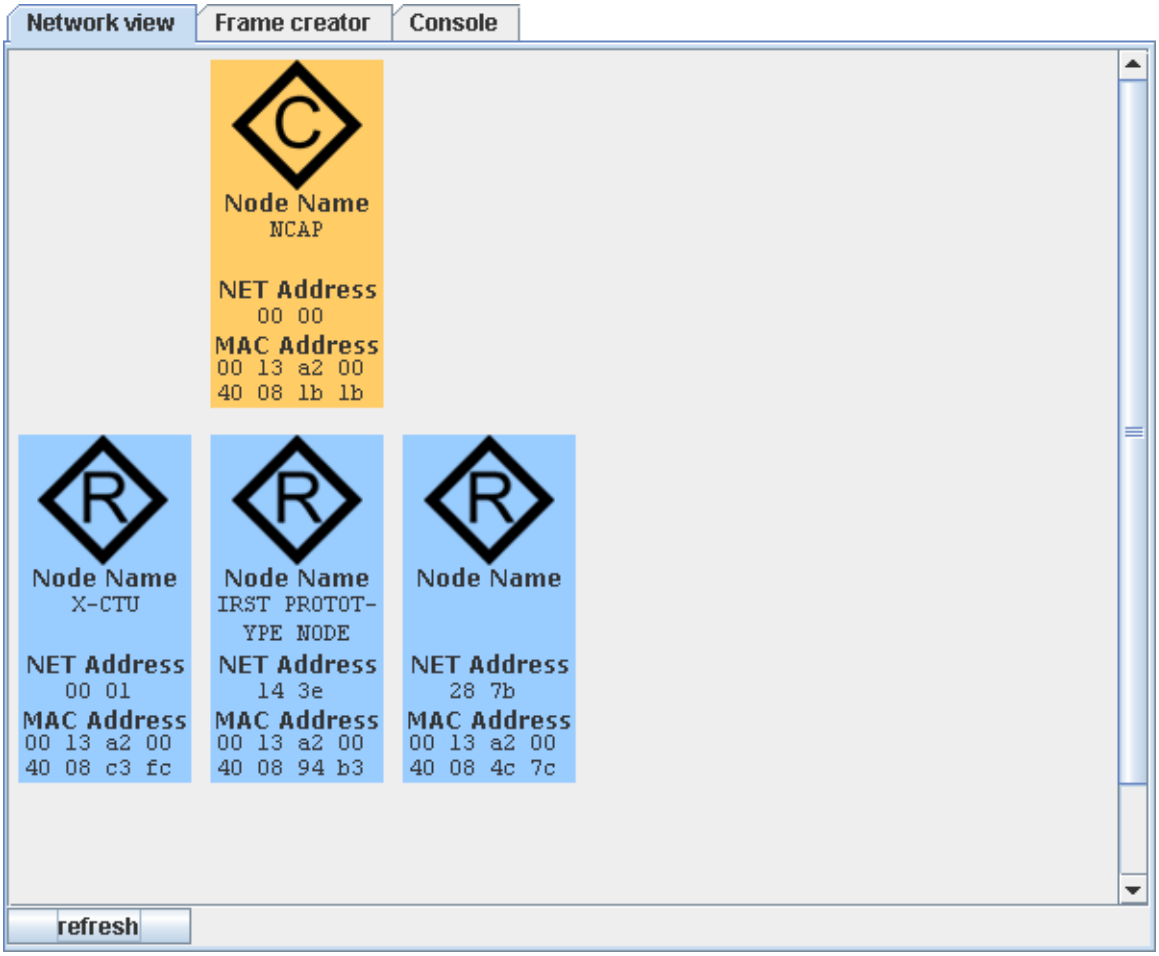


Figure A-1: View of the WACNet nodes on the HMI applet

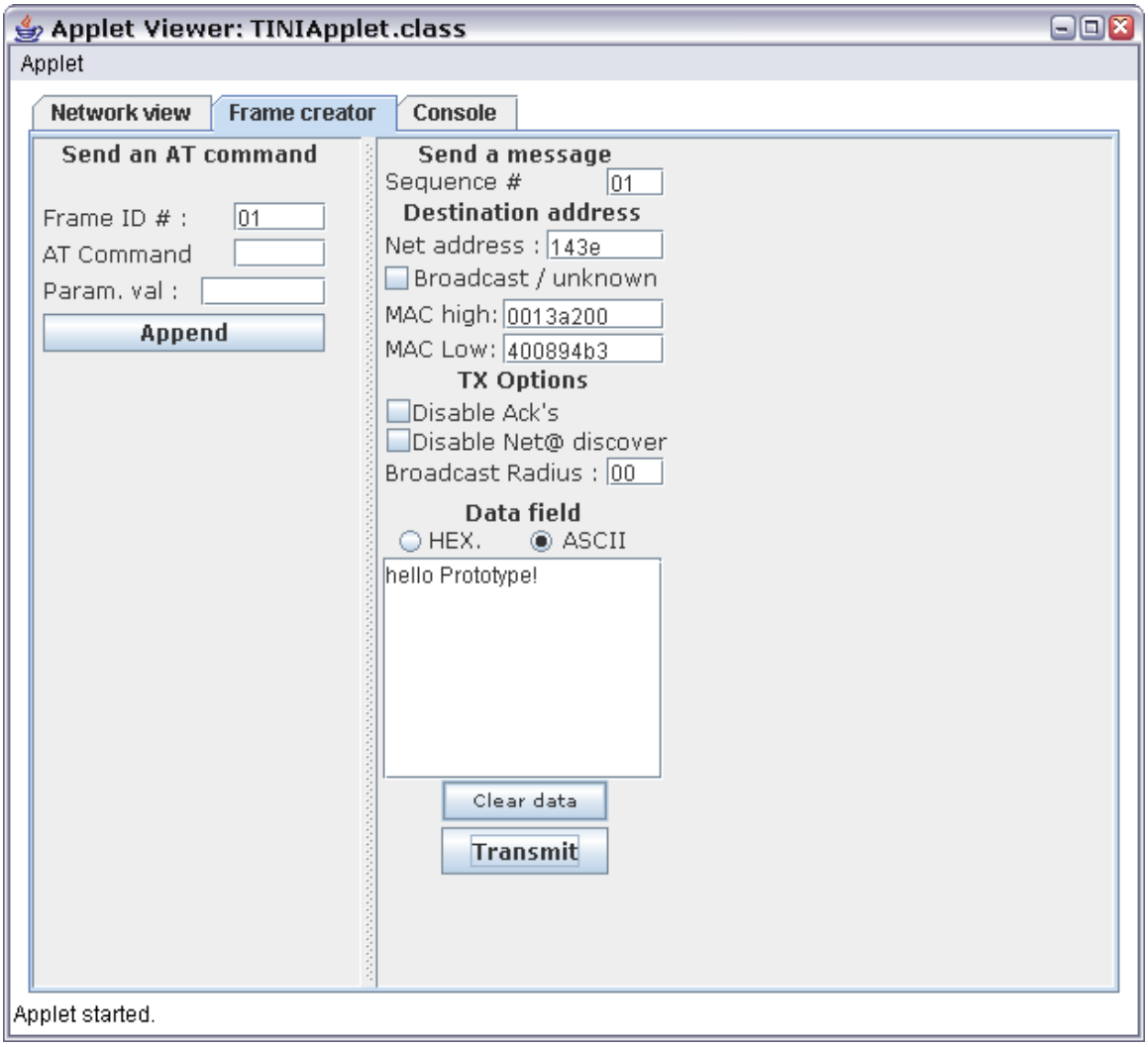


Figure A-2: Screenshot of the "Frame creator" utility on the HMI applet.

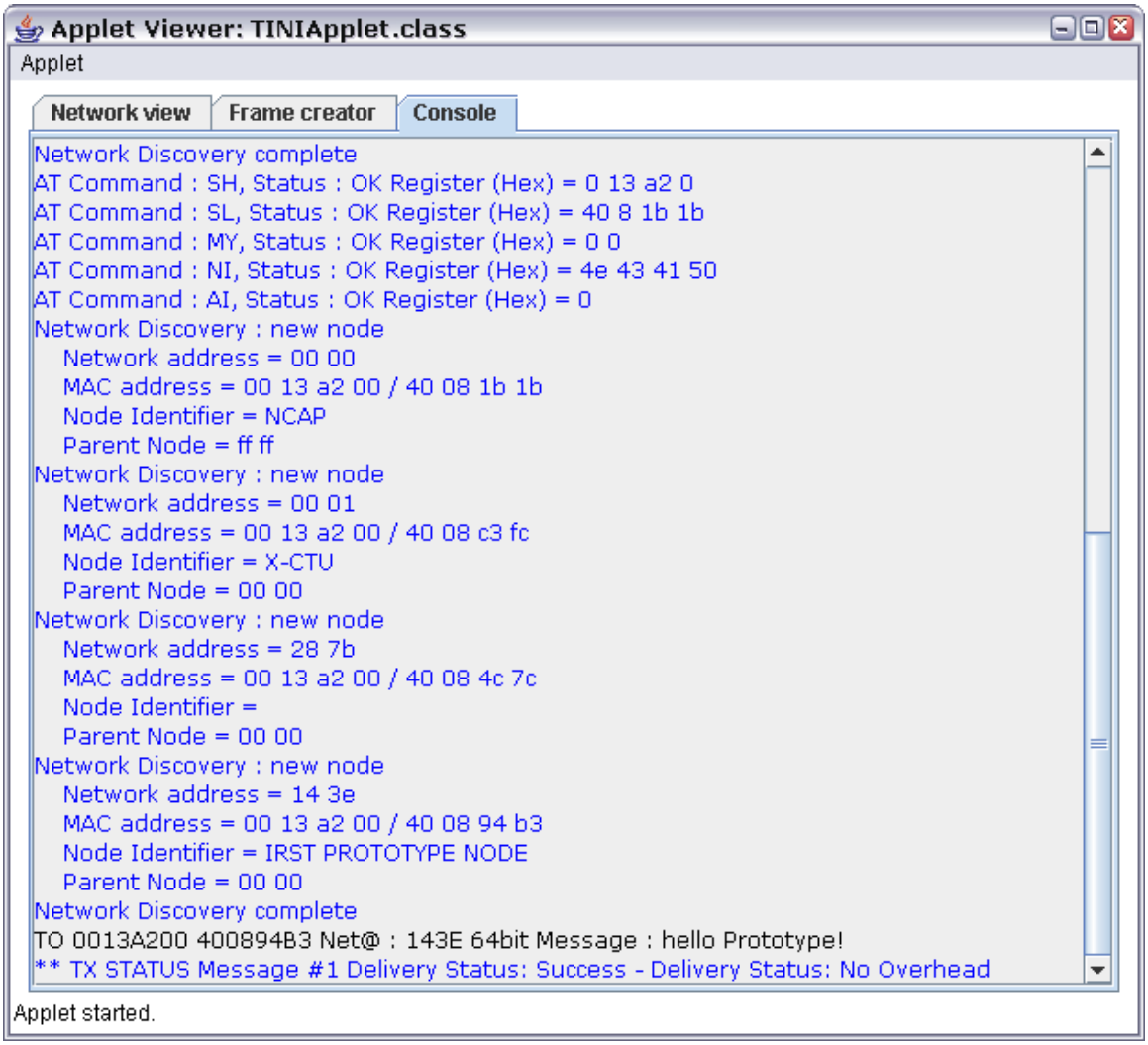


Figure A-3: View of the HMI's console. This displays and explains all the frames transiting through the bridging node

Appendix B XBee ZigBee Chips Datasheet

1. XBee/XBee-PRO OEM RF Modules

The XBee and XBee-PRO OEM RF Modules were engineered to meet IEEE 802.15.4 standards and support the unique needs of low-cost, low-power wireless sensor networks. The modules require minimal power and provide reliable delivery of data between devices.

The modules operate within the ISM 2.4 GHz frequency band and are pin-for-pin compatible with each other.

1.1. Key Features

Long Range Data Integrity

XBee
- Indoor/Urban: up to 100' (30 m)
- Outdoor line-of-sight: up to 300' (100 m)
- Transmit Power: 1 mW (0 dBm)
- Receiver Sensitivity: -92 dBm

XBee-PRO
- Indoor/Urban: up to 300' (100 m)
- Outdoor line-of-sight: up to 1 mile (1500 m)
- Transmit Power: 100 mW (20 dBm) EIRP
- Receiver Sensitivity: -100 dBm

RF Data Rate: 250,000 bps

Low Power

XBee
- TX Current: 45 mA (@3.3 V)
- RX Current: 50 mA (@3.3 V)
- Power-down Current: < 10 µA

XBee-PRO
- TX Current: 215 mA (@3.3 V)
- RX Current: 55 mA (@3.3 V)
- Power-down Current: < 10 µA

ADC and I/O line support

Analog-to-digital conversion, Digital I/O

I/O Line Passing

Advanced Networking & Security

Retries and Acknowledgements

DSSS (Direct Sequence Spread Spectrum)

Each direct sequence channels has over 65,000 unique network addresses available

Source/Destination Addressing

Unicast & Broadcast Communications

Point-to-point, point-to-multipoint and peer-to-peer topologies supported

Coordinator/End Device operations

Easy-to-Use

No configuration necessary for out-of box RF communications

Free X-CTU Software (Testing and configuration software)

AT and API Command Modes for configuring module parameters

Extensive command set

Small form factor

Free & Unlimited RF-XPert Support

1.1.1. Worldwide Acceptance

FCC Approval (USA) Refer to Appendix A [p57] for FCC Requirements. Systems that contain XBee/XBee-PRO RF Modules inherit MaxStream Certifications.

ISM (Industrial, Scientific & Medical) **2.4 GHz frequency band**

Manufactured under **ISO 9001:2000** registered standards

XBee/XBee-PRO RF Modules are optimized for use in the **United States, Canada, Australia, Israel and Europe**. Contact MaxStream for complete list of government agency approvals.

1.2. Specifications

Table 1-01. Specifications of the XBee/XBee-PRO OEM RF Modules

Specification	XBee	XBee-PRO
Performance		
Indoor/Urban Range	up to 100 ft. (30 m)	Up to 300' (100 m)
Outdoor RF line-of-sight Range	up to 300 ft. (100 m)	Up to 1 mile (1500 m)
Transmit Power Output (software selectable)	1mW (0 dBm)	60 mW (18 dBm) conducted, 100 mW (20 dBm) EIRP*
RF Data Rate	250,000 bps	250,000 bps
Serial Interface Data Rate (software selectable)	1200 - 115200 bps (non-standard baud rates also supported)	1200 - 115200 bps (non-standard baud rates also supported)
Receiver Sensitivity	-92 dBm (1% packet error rate)	-100 dBm (1% packet error rate)
Power Requirements		
Supply Voltage	2.8 – 3.4 V	2.8 – 3.4 V
		If PL=0 (10dBm): 137mA(@3.3V), 139mA(@3.0V)
General		
Operating Frequency	ISM 2.4 GHz	ISM 2.4 GHz
Dimensions	0.960" x 1.087" (2.438cm x 2.761cm)	0.960" x 1.297" (2.438cm x 3.294cm)
Operating Temperature	-40 to 85° C (industrial)	-40 to 85° C (industrial)
Antenna Options	Integrated Whip, Chip or U.FL Connector	Integrated Whip, Chip or U.FL Connector
Networking & Security		
Supported Network Topologies	Point-to-point, Point-to-multipoint & Peer-to-peer	
Number of Channels (software selectable)	16 Direct Sequence Channels	12 Direct Sequence Channels
Addressing Options	PAN ID, Channel and Addresses	PAN ID, Channel and Addresses
Agency Approvals		
United States (FCC Part 15.247)	OUR-XBEE	OUR-XBEEPRO
Industry Canada (IC)	4214A-XBEE	4214A-XBEEPRO
Europe (CE)	ETSI	ETSI (Max. 10 dBm transmit power output)*
Japan	n/a	005NYCA0378 (Max. 10 dBm transmit power output)**

* When operating in Europe: XBee-PRO RF Modules must be configured to operate at a maximum transmit power output level of 10 dBm. The power output level is set using the PL command. The PL parameter must equal "0" (10 dBm).

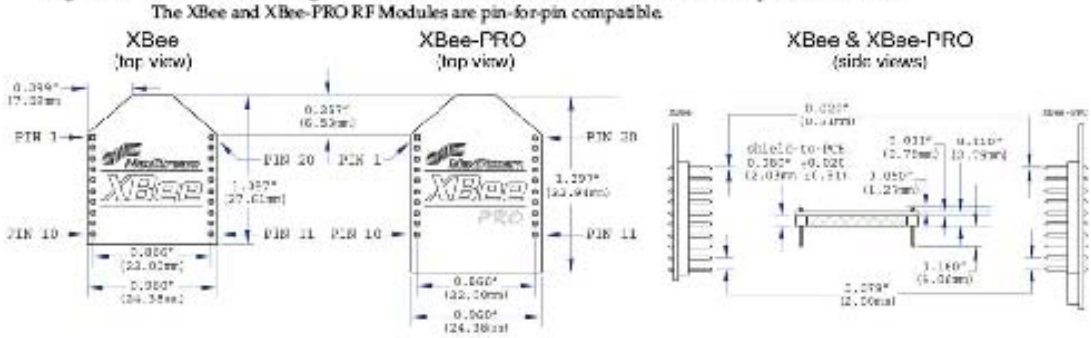
Additionally, European regulations stipulate an EIRP power maximum of 12.86 dBm (19 mW) for the XBee-PRO and 12.11 dBm for the XBee when integrating high-gain antennas.

** When operating in Japan: Transmit power output is limited to 10 dBm. A special part number is required when ordering modules approved for use in Japan. Contact MaxStream for more information [call 1-801-765-9885 or send e-mails to sales@maxstream.net].

Antenna Options: The ranges specified are typical when using the integrated Whip (1.5 dBi) and Dipole (2.1 dBi) antennas. The Chip antenna option provides advantages in its form factor; however, it typically yields shorter range than the Whip and Dipole antenna options when transmitting outdoors. For more information, refer to the "XBee Antenna" application note located on MaxStream's web site (http://www.maxstream.net/support/knowledgebase/article.php?kb=153).

1.3. Mechanical Drawings

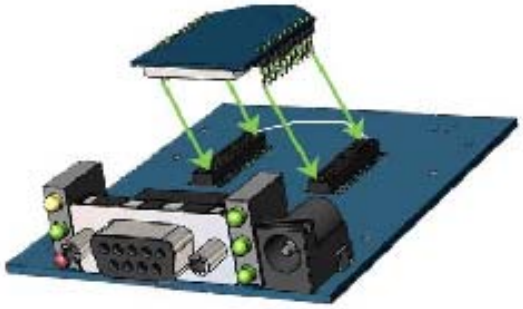
Figure 1-01. Mechanical drawings of the XBee/XBee-PRO OEM RF Modules (antenna options not shown)
The XBee and XBee-PRO RF Modules are pin-for-pin compatible.



1.4. Mounting Considerations

The XBee/XBee-PRO RF Module was designed to mount into a receptacle (socket) and therefore does not require any soldering when mounting it to a board. The XBee Development Kits contain RS-232 and USB interface boards which use two 20-pin receptacles to receive modules.

Figure 1-02. XBee Module Mounting to an RS-232 Interface Board.



The receptacles used on MaxStream development boards are manufactured by Century Interconnect. Several other manufacturers provide comparable mounting solutions; however, MaxStream currently uses the following receptacles:

- Through-hole single-row receptacles -
  Samtec P/N: MMS-110-01-L-SV (or equivalent)
- Surface-mount double-row receptacles -
  Century Interconnect P/N: CPRMSL20-D-0-1 (or equivalent)
- Surface-mount single-row receptacles -
  Samtec P/N: SMM-110-02-SM-S

MaxStream also recommends printing an outline of the module on the board to indicate the orientation the module should be mounted.

Appendix C ATMEL ATMEGA 32 Datasheet

Features

- High-performance, Low-power AVR® 8-bit Microcontroller
- Advanced RISC Architecture
  - 131 Powerful Instructions – Most Single-clock Cycle Execution
  - 32 x 8 General Purpose Working Registers
  - Fully Static Operation
  - Up to 16 MIPS Throughput at 16 MHz
  - On-chip 2-cycle Multiplier
- Nonvolatile Program and Data Memories
  - 32K Bytes of In-System Self-Programmable Flash
    Endurance: 10,000 Write/Erase Cycles
  - Optional Boot Code Section with Independent Lock Bits
    In-System Programming by On-chip Boot Program
    True Read-While-Write Operation
  - 1024 Bytes EEPROM
    Endurance: 100,000 Write/Erase Cycles
  - 2K Byte Internal SRAM
  - Programming Lock for Software Security
- JTAG (IEEE std. 1149.1 Compliant) Interface
  - Boundary-scan Capabilities According to the JTAG Standard
  - Extensive On-chip Debug Support
  - Programming of Flash, EEPROM, Fuses, and Lock Bits through the JTAG Interface
- Peripheral Features
  - Two 8-bit Timer/Counters with Separate Prescalers and Compare Modes
  - One 16-bit Timer/Counter with Separate Prescaler, Compare Mode, and Capture Mode
  - Real Time Counter with Separate Oscillator
  - Four PWM Channels
  - 8-channel, 10-bit ADC
    8 Single-ended Channels
    7 Differential Channels in TQFP Package Only
    2 Differential Channels with Programmable Gain at 1x, 10x, or 200x
  - Byte-oriented Two-wire Serial Interface
  - Programmable Serial USART
  - Master/Slave SPI Serial Interface
  - Programmable Watchdog Timer with Separate On-chip Oscillator
  - On-chip Analog Comparator
- Special Microcontroller Features
  - Power-on Reset and Programmable Brown-out Detection
  - Internal Calibrated RC Oscillator
  - External and Internal Interrupt Sources
  - Six Sleep Modes: Idle, ADC Noise Reduction, Power-save, Power-down, Standby and Extended Standby
- I/O and Packages
  - 32 Programmable I/O Lines
  - 40-pin PDIP, 44-lead TQFP, and 44-pad QFN/MLF
- Operating Voltages
  - 2.7 - 5.5V for ATmega32L
  - 4.5 - 5.5V for ATmega32
- Speed Grades
  - 0 - 8 MHz for ATmega32L
  - 0 - 16 MHz for ATmega32
- Power Consumption at 1 MHz, 3V, 25°C for ATmega32L
  - Active: 1.1 mA
  - Idle Mode: 0.35 mA
  - Power-down Mode: < 1 µA

8-bit AVR® Microcontroller with 32K Bytes In-System Programmable Flash

ATmega32
ATmega32L

2503J–AVR–10/06