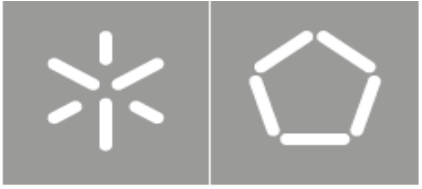
Universidade do Minho
Escola de Engenharia

Hadeel Mohamad-Ali Abdah

Selective Reprogramming of WSNs: Energetic Study and Functionality Optimization

Julho de 2016



Universidade do Minho
Escola de Engenharia
Departamento de Informática

Hadeel Mohamad-Ali Abdah

Selective Reprogramming of WSNs: Energetic Study and Functionality Optimization

Dissertação de Mestrado
Mestrado em Engenharia de Redes e Serviços Telemáticos

Trabalho realizado sob orientação de

Supervisor: Paulo Manuel Martins de Carvalho
Supervisor (in the Company): Emanuel Ribeiro Lima

Julho de 2016

Acknowledgments

First of all, I would like to thank the Erasmus Mundus program and the University of Minho for accepting me as a mobility student. It was a wonderful opportunity, full of new friends and experiences.

I would also like to thank the Engineering of Computer Networks and Telematics Services (MERSTel) program committee members for their help and contribution to my professional growth.

A special appreciation and thanks to my supervisors, Professor Paulo Carvalho and Eng. Emanuel Lima for all of their guidance and encouragements throughout the dissertation process.

Finally, I would like to thank my family for all of their love and support. I know it was hard, but it was worth it.

Abstract

Wireless sensors networks consist of large numbers of small, battery-powered, self-organizing computing motes. Nowadays, these networks are considered ideal candidates for a wide range of applications such as environmental monitoring, military operations and other application fields where it is hard to maintain a continuous presence of human beings.

Online remote reprogramming is usually carried out to update the code running on nodes due to factors such as changes in the environment or application. Remote reprogramming might be applied to the whole network or just to a subset of nodes (selective reprogramming), either way it is crucial to provide reliability for such procedure. Therefore, most of the approaches oriented to remote reprogramming resort to flooding the whole network, leading to a major waste of energy in network nodes.

When dealing with selective reprogramming, the waste of energy increases steeply even when just a small number of nodes need to get the update messages. These messages may be received and retransmitted from all nodes in the network resulting in a waste of resources.

This research identifies multiple scenarios for selective reprogramming and proposes a different energy-aware approach for each one trying to reduce energy consumption in the network by taking advantage of multiple and complementary solutions such as wise routing, clustering and the ability to manage nodes sleeping time instead of using the typical flooding approach.

These approaches were tested and compared with typical flooding and Deluge solutions. The results show a significant reduction of the power consumption, thus, making the selective remote reprogramming more energy-efficient.

Contents List

Figures List

Acronyms List

BP	Best Path
BPN	Bit Per Node
BS	Base Station
CH	Cluster Head
CN	Cluster Node
CP	Check Path
DCFR	Double Cost Function based Route
EECDA	Energy-Efficient Clustering and Data Aggregation Protocol
EEMRP	Energy-Efficient Multipath Routing Protocol
EEORP	Energy-Efficient Online Reprogramming Protocol
ESCFR	Exponential and Sine Cost Function based Route
HEED	Hybrid, Energy-Efficient, Distributed clustering approach
LEACH	Low-Energy, Adaptive Clustering Hierarchy
MCU	Micro-Control Unit
MEMS	Micro-Electro-Mechanical Systems
MNP	Multihop Network Reprogramming
MOAP	Multihop Over-the-Air Programming
NOH	Number Of Hops
NPN	Non-Path Node
PEGASIS	Power-Efficient Gathering in Sensor Information Systems
PN	Path Node
SASA	Structure-Aware Self-Adaptive
SEP	Stable Election Protocol
SN	Selected Node
SOS	Sleep Or Send
TDMA	Time Division Multiple Access
TTS	Time To Sleep
UW-sensor	Under Water sensor
WUSN	Wireless Underground Sensor Network
WSN	Wireless Sensor Network

1. Introduction

1.1 Introduction

Wireless Sensor Networks (WSNs) are networks of tiny, battery powered sensor nodes with limited on-board processing, storage and radio capabilities [1]. Nodes sense and send their reports toward a processing center called “sink”.

WSNs have gained worldwide attention in recent years, particularly with the proliferation in Micro-Electro-Mechanical Systems (MEMS) technology, which has facilitated the development of small sized, smart sensors. These sensors are inexpensive compared to traditional sensors and can operate for extended periods of time without physical intervention by humans [2].

Due to the potential to provide fine-grained sensing and actuation at a reasonable cost, these networks are considered ideal candidates for a wide range of applications such as industrial monitoring and military operations.

Program image updates, may also be referred to as code image or software updates, have become a requirement for long-lived networks due to changes in the functionality of the software running on the nodes, software updates, and security enhancements or for bug fixing after WSNs have been deployed. However, if WSNs are large scale or deployed in a harsh environment, it is impossible to reprogram manually all nodes. Online network reprogramming can remotely reprogram all nodes via wireless communication. Hence it becomes a promising technique [3].

Most proposed remote reprogramming protocols in WSNs take advantage of wireless nodes using dissemination strategies to reach all network nodes (remote reprogramming for the whole network). In some cases, just a few number of nodes might need to be reprogrammed (selective reprogramming). Either way, it is crucial to provide reliability for such procedure. However, for large WSNs, where the sink cannot reach every node through single-hop communication, updates can only be transmitted using multi-hop approaches, which normally consumes a significant amount of energy in the nodes and thus reducing the network lifetime.

When dealing with selective reprogramming, the waste of energy is magnified since only a small number of nodes needs to get update messages, while these messages might be received and retransmitted from all nodes resulting in a waste of resources.

The main goal for this research is to make selective reprogramming of WSNs as efficient as possible, reducing the energy consumption in the network while maintaining the reliability and versatility required for such procedure.

1.2 Motivations and objectives

Energy constrain is considered one of the main challenges facing wireless sensor networks(WSNs) since most sensor nodes still use battery as the main source of power.

Radio activities such as messages transmission/reception and idle listening are considered major factors of energy consumption in motes. These activities increase with the increase of reliability level required, hence it is usually achieved by broadcasting and rebroadcasting. Procedures such as remote reprogramming require high level of reliability leading to an increase in radio activities in the network, thus, more waste of energy. This waste increases steeply when dealing with selective reprogramming where only few nodes need to receive the code updates while most approaches used are dependent on flooding the whole network forcing all nodes to receive the code.

The main objective of this research is to study and optimize energy consumption in selective remote reprogramming procedure, presenting multiple scenarios and proposing a different approach for each to perform this procedure in an energy efficient way while trying to maintain reliability needed for this process.

The performance metrics we are focusing on the number of messages received and transmitted which is directly related to energy consumption in the nodes.

1.3 Contribution

Our contributions in this work consists of presenting multiple scenarios for selective reprogramming and propose a different approach for each, trying to reduce energy consumption in the network. We do this by taking advantage of multiple and complementary solutions such as wise routing, clustering and the ability to manage nodes sleeping time instead of using the typical flooding approach. In this thesis, we also propose an enhancement to Deluge extension for selective reprogramming in WSNs [4] in order to reduce the overhead in the reprogramming packets used in this protocol, thus, minimizing the number of packets needed.

We will consider two possibilities when performing selective reprogramming of a WSN depending on the centrality of the new code image to be disseminated. A first possibility is that the program image resides only in the base station and a specific node needs to be reprogrammed. In this scenario, flooding the whole network with code messages should be avoided, thus, we will try sending the code from the base station to the selected node following the path with the highest energy levels, while forcing other nodes outside this path to turn their radios off.

A second possibility is that when some of nodes deployed in the network already have the updated code image that needs to be routed to the selected node. In this case, two other scenarios need to be discussed. The first one is when the network has some sort of hierarchy in place, for instance, via node clustering. In this case, we suggest the use of multiple senders instead of only one to perform code dissemination. Thus, each sender can contribute by sending a number of code messages according to its residual energy. The other scenario is when the network is flat, i.e., with no hierarchy, and multiple nodes within vicinity to the selected node can play the role of sender.

1.4 Dissertation structure

This chapter describes the structure of the dissertation. Chapter 2 provides an overview of wireless sensors network, their characteristics, constrains, topologies and applications. Chapter 3 discusses energy issues in WSNs, including energy consumption and saving.

Chapter 4 focuses on remote reprogramming in WSNs, the challenges facing this procedure and the protocols proposed to do a whole or selective reprogramming.

Chapter 5 presents various scenarios of selective remote reprogramming and the suggested strategies to perform this procedure in an energy efficient manner.

Chapter 6 describes the proposed strategies implementation and the simulation results.

Finally, Chapter 7 is dedicated for conclusions and future work.

2. Wireless Sensor Network

This chapter presents a general overview of wireless sensor networks and is organized as followed: in section 2, some WSN characteristics and constraints are identified, section 3 describes the WSN topologies and paradigm. Finally, WSN applications are presented in section 4.

2.1 Wireless sensor network characteristics and constraints

A wireless sensor network (WSN) is a network made up of a large number of small sensors called sensor node or mote. This network, in general, also has one or several stations (sink) to put the data collected from the small sensors [5].

By providing fine-grained and unintrusive monitoring in real-time, WSNs allow tight integration of the physical world with a computing system infrastructure. WSNs have been demonstrated to play an important role in a rich variety of applications such as environmental and civilian infrastructure monitoring [6], wildlife tracking [7], military surveillance, etc.

Sensor nodes are able to capture various physical information such as barometric pressure, ambient temperature, atmospheric humidity, wind direction, wind speed, underground water level, and rainfall. The collaborative nature of sensor nodes brings several advantages over traditional sensing including greater fault tolerance, improved accuracy, large coverage area and extraction of localized features [8]. However, sensor nodes are usually battery-powered and deployed in a harsh or hostile environment where it is very difficult or even impossible to change or recharge the batteries [9]. Therefore, power consumption is a central design consideration in wireless sensor networks. However, since various sensor nodes often detect common phenomena, there is likely to be some redundancy in the data the various sources communicate to a particular sink so in-network filtering and processing techniques can help to conserve the scarce energy resources [10]. Sensor nodes are also highly limited in computation and storage capacities [9].

A WSN is usually designed and deployed for a specific application. The design requirements of a network change with its application. Hence the nodes are prone to physical damages or failures, and in addition to the dynamic environmental conditions require the system to adapt over time to change connectivity and system stimuli [11, 12].

No global identification is available for WSNs due to the large number of sensor nodes which makes impossible to build a global addressing scheme for a sensor network since it would introduce a high overhead for the identification [9, 12].

In most sensor network applications, the data sensed by sensor nodes flow from multiple source sensor nodes to a particular sink, exhibiting a many-to-one traffic pattern.

Sensors may be deployed in an ad hoc manner into the field then left unattended to perform monitoring and reporting functions. In an unstructured WSN, network maintenance is difficult since there might be so many nodes requiring that the system identifies and copes with the failure and connectivity loss between the nodes that it is necessary providing some automatic configuration and reconfiguration support (self-configuration) [11,13].

Security challenges in WSN are at many levels [13]. From the system point of view, it is critical that the information provided by the nodes be authenticated and the integrity verified, since this information provides the feedback loop to expensive equipment controlling power consumption depends on app. From the users' point of view, it is also critical that this information cannot be easily spoofed remaining protected in the back end processor, since it may affect the privacy of users.

2.2 Wireless sensor node topologies and paradigm

The most common four WSN data network topologies are Peer to Peer (also called Point to Point), Star, Tree and Mesh [14].

Peer-to-Peer networks allow each node to communicate directly with another node without needing to go through a centralized communications hub. **Star** topology needs a central hub where all communications must be routed through it. Each node is then a "client" while the central hub is the "server". The central hub is shown in Fig 2.1. **Tree** networks use a central node called a *Root* node as the main communications router. One level down from the *Root* node in the hierarchy is a *Central* hub. This lower level then forms a Star network. The Tree network can be considered a hybrid of both the Star and Peer-to-Peer networking topologies. Usually most routing protocols in WSNs are based on cluster tree topology where all cluster members route their data to a cluster head. **Mesh** networks allow data to "hop" from node to node which allows the network to be self-healing. Each node is then able to communicate with each other as data is routed from node to node until it reaches the desired location [14]. Fig 2.1 illustrates the different topologies in WSN.

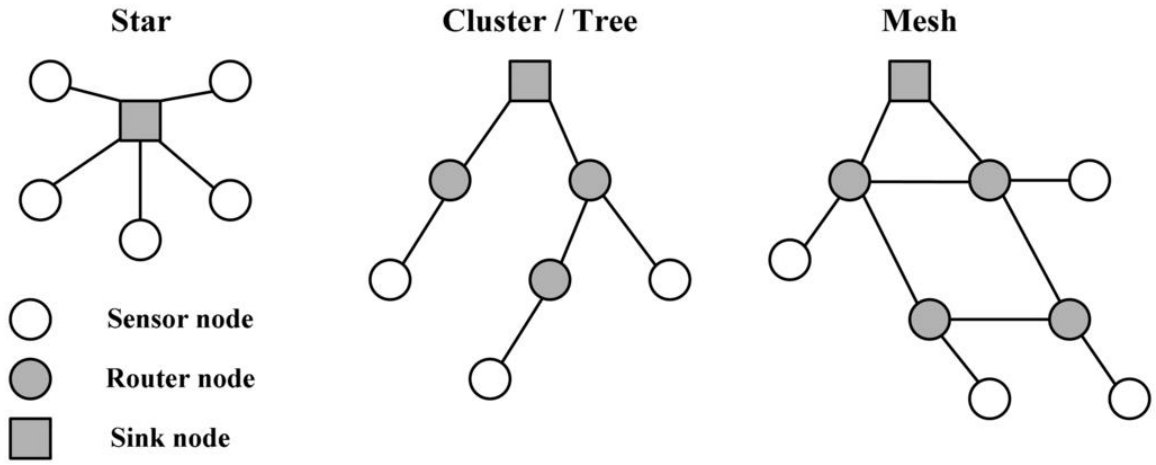


Fig 2. 1: WSN topologies [4]

Data aggregation is considered the essential paradigm for routing in sensor networks [15, 16]. The idea is to combine the data coming from different sources enroute, eliminating redundancy and minimizing the number of transmissions, thus saving energy. This paradigm shifts the focus from the traditional *address-centric* approaches for networking (finding short routes between pairs of addressable end-nodes) to a more *data-centric* approach (finding routes from multiple sources to a single destination that allows in-network consolidation of redundant data) [10]. The topologies used when applying data aggregation on WSN are usually the cluster and the tree topologies.

2.3 WSN applications

Wireless sensor networks (WSNs) have enabled numerous embedded wireless applications in different areas, such as military/battlefield, environmental monitoring, and intelligent building systems [17]. The following subsections will present some of WSNs applications.

2.3.1 Terrestrial applications

In this type of applications, hundreds to thousands of inexpensive wireless sensor nodes are deployed in a given area, either in an ad hoc or in a pre-planned manner. In ad hoc deployment, sensor nodes can be dropped from a plane and randomly placed into the target area [13].

In [18], a WSN was used for battlefield *surveillance* where the network was able to detect and classify multiple targets (e.g., vehicles and troop

movements) using inexpensive sensor capable of sensing acoustic and magnetic signals generated by different target objects. In [19], a real-time wireless physiological monitoring system for nursing centers was implemented. Its function is to monitor online the physiological status of aged patients via wireless communication channel and wired local area network. The collected data, such as body temperature, blood pressure, and heart rate, can then be stored in the computer of a network management center to facilitate the medical staff in a nursing center to monitor in real time or analyze in batch mode the physiological changes of the patients under observation. Other projects related to healthcare are [20] [21].

Environmental monitoring is carried out in [22] where a wireless sensor network is deployed on Great Duck Island, Maine. These networks monitor the microclimates in and around nesting burrows used by the Leach's Storm Petrel.

2.3.2 Underground application

Wireless Underground Sensor Network (WUSN) can be used to monitor a variety of conditions, such as soil properties for agricultural applications and toxic substances for environmental monitoring, Sensors are also successfully used to monitor the integrity of belowground infrastructures such as plumbing, landslide and earthquake monitoring are accomplished using buried seismometers [23]. In [24], a **S**tructure-**A**ware **S**elf-**A**daptive sensor system, SASA, is presented with the aim to address the challenges and provide a feasible framework for underground monitoring in coal mines. The design objectives of SASA are: i) rapidly detect the collapse area and report to the sink node; and ii) maintain the system integrity when the sensor network structure is altered.

There have been several works on tunnel monitoring such as [25] where the author proposes to utilize electro level systems to measure the structural variations in London underground tunnels.

2.3.3 Underwater application

Underwater WSNs consist of a number of underwater sensor nodes (UW-sensors) and vehicles deployed underwater. As opposite to terrestrial WSNs, underwater sensor nodes are more expensive and fewer sensor nodes are deployed. Autonomous underwater vehicles are used for exploration or gathering data from sensor nodes. Compared to a dense deployment of sensor nodes in a terrestrial WSN, a sparse deployment of sensor nodes is placed underwater. Typical underwater wireless communications are established through transmission of acoustic waves [26]. Fig 2.2 illustrates an underwater wireless sensor network.

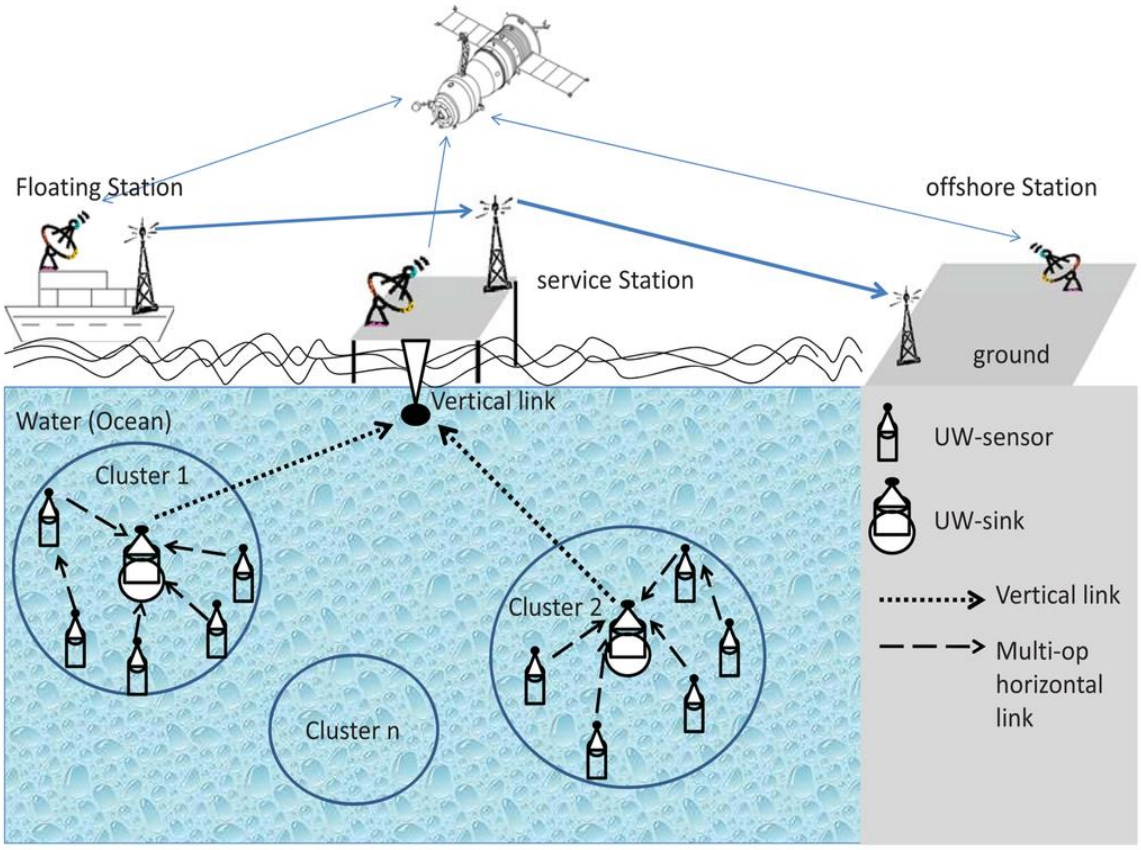


Fig 2. 2: Underwater wireless sensor network [27]

A promising application for underwater sensor networks is seismic monitoring for oil extraction from underwater fields. Frequent seismic monitoring is of importance in oil extraction. Studies of variation in the reservoir over time are called "4-D seismic" and are useful for judging field performance and motivating intervention.

Underwater equipment monitoring is a second example application. Long-term equipment monitoring may be done with pre-installed infrastructure. However, temporary monitoring would benefit from low-power, wireless communication.

A third underwater application for WSN is supporting groups of underwater autonomous robots. Applications include coordinating adaptive sensing of chemical leaks or biological phenomena (for example, oil leaks or phytoplankton concentrations) [28].

2.3.4 Multimedia application

Multimedia WSNs consist of a number of low cost sensor nodes equipped with cameras and microphones such as Panoptes [29], which is a system developed for environmental observation and surveillance applications, based on Intel StrongARM PDA platforms with a Logitech webcam as a video capture device.

In [30], a system whose objective is to limit the computation, bandwidth, and human attention burdens imposed by large-scale video surveillance systems is described. In-network processing is used on each camera to filter out uninteresting events locally, avoiding disambiguation and tracking of irrelevant environmental distractors.

2.3.5 Mobile applications

Mobile applications use sensor nodes that can move on their own and interact with the physical environment. A key difference is that mobile nodes have the ability to reposition and organize themselves in the network. A mobile WSN can start off with some initial deployment and nodes can then spread out to gather information. Information gathered by a mobile node can be communicated to another mobile node when they are within range of each other.

Other key difference is data distribution. In a static WSN, data can be distributed using fixed routing or flooding while dynamic routing is used in a mobile WSN. Challenges in a mobile WSN include deployment, localization, self-organization, navigation and control, coverage, energy, maintenance, and data process [13].

2.4 Summary

In this chapter, some WSN characteristics, topologies and applications were discussed.

As it has been highlighted in this chapter, WSN can be deployed in harsh environments where continuous presence of humans is impossible. Therefore, reducing energy consumption in WSN is a central design consideration since sensor nodes are usually battery-powered.

In the following chapter, power consumption in wireless sensor networks will be discussed thoroughly. After, the mechanisms used to save energy in WSN will be presented.

3. Energy Issues in WSN

This chapter will focus on the following issues: sensor nodes' types and architecture, sensor power sources, energy consumption and energy saving in WSNs. This will follow as sections.

3.1 Sensor node types and architecture

A sensor node, also known as a mote, is a node in a sensor network that is capable of performing some processing, gathering sensory information and communicating with other connected nodes in the network. A wireless sensor network (WSN) consists of spatially distributed autonomous motes to monitor physical or environmental conditions. Depending on the sensor nodes within the network, WSNs can be categorized into heterogeneous and homogeneous networks. Heterogeneous WSN consists of sensor nodes with different ability, such as different computing power and sensing range, whereas homogeneous WSN consists of identical sensor nodes [31].

There are many types of sensor nodes. The application where the wireless sensor network is applied is what determines what type of sensors to use. Some sensors can sense physical properties such as pressure, humidity, temperature, and flow. Other sensors can measure motion properties like position, velocity, angular velocity, and acceleration. Some sensors can measure proximity, distance, range, etc. A single sensor node might be able to sense several environmental metrics. Sensor nodes can also be classified based on their mobility to static and mobile nodes, or based on their capabilities to super nodes and low end nodes.

A typical sensor node usually consists of four sub-systems [32], as illustrated in Fig 3.1. These subsystems are:

a) A computing subsystem

Consists of RAM, flash memories, and a microprocessor that typically uses a set of analog-to-digital converters (ADCs) to obtain data from sensors and communications protocols [33]. The microprocessor can operate under various operating modes for power management purposes. However, shuttling between these operating modes involves consumption of power, so the energy consumption levels of the various modes should be considered while looking at the battery lifetime of each node [32]. This subsystem can also be referred to as Micro-Control Unit (MCU).

b) A communication subsystem

Consists of a short range radio which is used to communicate with neighboring nodes and the outside world. Radios can operate under the Transmit, Receive, Idle and Sleep modes. It is important to completely shut down the radio rather than put it in idle mode when it is not transmitting or receiving because of the high power consumed in this mode [32]. This subsystem can also be referred to as Micro Control Unit (MCU).

c) A sensing subsystem

Consists of a group of sensors and actuators that link the node to the outside world. Energy consumption can be reduced by using low power components and saving power at the cost of performance which is not required [32].

d) A power supply subsystem

Consists typically of two AA batteries. Expected lifetimes of batteries should range from months to years in wireless sensor network applications since it is undesirable to recharge or replace the batteries of thousands of sensor nodes frequently. When a node is depleted of energy, it can no longer fulfill its role unless the source of energy is replenished. Therefore, it is generally accepted that the usefulness of a wireless sensor (or wireless sensor node) expires when its battery runs out [17].

The amount of power drawn from a battery should be checked because if a high current is drawn from a battery for a long time, the battery will die even though it could have gone on for a longer time. The lifetime of a battery can be increased by reducing the current drastically or even turning it off often [32].
These components are illustrated in Fig 3.1

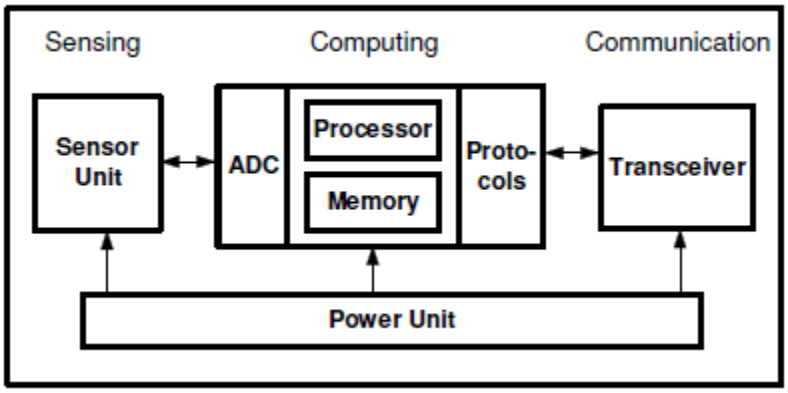


Fig 3. 1: Sensor node main component [33]

3.2 Sensor node power supply

Power is a scarce resource in WSNs. As an example, the capacity of the *Smart Dust mote* is 33mAh [34] as powered by a single size- 5 hearing aid battery. For MicaZ and Mica2 nodes [34], which use two AA batteries, the node capacity is limited to 1400–3400 mAh. Similarly, the recent SunSPOT platform uses a 750mAh lithium-ion battery [36].

Batteries continue to be the leading power source for sensor nodes. However, it is also possible to extend the lifetime of the WSNs through energy scavenging [37, 38], which is extracting energy from the environment. Solar cells offer excellent power density in direct sunlight. However, in dim office lighting, or areas with no light, they are inadequate. Power scavenged from thermal gradients is also considered. Nevertheless, it is difficult to find greater than a 10 ^{o}C[1] thermal gradient in a volume of 1 cm^3. Starner [39] has documented, that there is ample power to scavenge from the human body. Fuel cells also represent a potentially large improvement over batteries as an energy reservoir. Research to miniaturize fuel cells is currently underway [40]. Miniaturized fuel cells could extend the lifetime of a node up to several times when compared to battery-powered sources, and so would either require re-fueling, or would have a limited lifetime.

It should be noted that while the energy density of hydrocarbon fuels used in micro heat engines is very high, the output power of these devices is too high (on the order of 1–10 W) to be of practical use for low power wireless sensor nodes [41]. Furthermore, once started, they are not easily turned off. Their use would therefore necessitate a large energy storage reservoir to allow for lower power operation over a longer lifetime, thus negating their advantage.

As stated, non-rechargeable batteries remain the main source used for powering wireless nodes because of their higher energy densities, lower leakage rates and cost. For sensor node applications, battery lifetimes of at least a year are desirable, corresponding to 32 J per μW average power. Lithium based primary batteries provide 1400 – 3600 J/cc [42], so in principle a lifetime of several years is achievable for a battery well below 1 cc [43].

[1] The **degree Celsius**(**°C**) refer to a temperature on the Celsius scale

3.3 Power consumption evaluation inside the sensor node

First we must analyze and estimate power consumption characteristics of a wireless sensor node to be able to optimize and control energy consumption.

There are several methods to estimate the power consumption of a WSN node, including theoretical estimation, direct measurements, and the usage of simulations tools [44]. The best way to carry out the power consumption evaluation is directly on the physical hardware, by periodically measuring the remaining battery. This will provide a precise evaluation for power consumption, but it is very costly, tedious and may be unfeasible considering the (usual) large number of WSN nodes. Furthermore, due to the inherent dynamism of WSNs, the instrumentation required by measurement techniques makes difficult their use in several different scenarios [45].

The simulation of energy-aware WSNs using currently available tools has a number of shortcomings. First, most available simulators lack adequate support for the node's hardware (especially energy sources such as photovoltaic and vibration harvesters) and the needed environmental models (available environmental models are often too inflexible to allow easy integration of new hardware). Second, the node's simulated embedded hardware is structured largely around communications, giving little consideration to the implementation and interfacing of sensor processing and energy-aware algorithms [46]. One of the simulators used to evaluate energy in WSN is PowerTOSSIM [47], which has been built on top of TOSSIM simulator to estimate Mica2's energy consumption. Another widely used simulator is Avrora [48] which can simulate AVR-based microcontroller MICA2 sensor nodes, and supports energy consumption simulation. However, this simulator has several drawbacks. Avrora does not have GUI and it cannot simulate network management algorithms because it does not provide network communication tools. There is also eSimu, which is a software module that implements platform specific energy consumption models, and provides an estimation of the current absorbed by a node [49].

Recent analyses of WSN energy efficiency have been widely based on sensor node power consumption models [50] [51]. Such models may be analytically evaluated and/or simulated. Although modeling may provide less accurate results than measuring, it provides the designers the flexibility and agility to evaluate complex scenarios without interfering on the actual environment [45].

Power consumption in the node is distributed among the node's subsystems mentioned: MCU, communication subsystem, battery and the

sensing unit. While power consumption can be considered static or neglected in the sensing unit and battery, MCU and communication subsystem have a significant effect on the node's energy consumption characteristics.

The MCU used in the node is usually chosen based on the required performance levels and the node's power consumption characteristics. For example, the StrongARM microprocessor from Intel, used in high-end sensor nodes, consumes around 400 mW of power while executing instructions, whereas the ATmega103L AVR microcontroller from Atmel consumes only around 16.5 mW, but provides much lower performance. Thus, the choice of MCU should be dictated by the application scenario, to achieve a close match between the performance level offered by the MCU and that demanded by the application. Further, MCUs usually support various operating modes, including Active, Idle, and Sleep modes, for power management purposes. Each mode is characterized by a different amount of power consumption [52].

Power consumption measurements of the communication subsystem in sensor node devices reveal clear discrepancies between the widely cited power consumption models and the actual characteristics of real hardware implementations. For example, the measured power consumption of the receiving circuitry is often greater than the power consumption of the transmitting circuitry [53] [54] [55]. Similarly, the power consumption for baseband digital signal processing is found to be comparable to the power consumption of the combined transmit and receive circuitry [56].

A classical energy consumption model has been proposed by Heinzelman *et. al* based on the observation that the energy consumption would likely be dominated by the data communications subsystem [57]. Table 3.1 summarizes their consumption model.

Radio mode	**Energy Consumption**
Transmitter Electronics Receiver Electronics	50 *nJ/bit*
Transmit Amplifier	100 pJ/bit/m^2
Idle	40 *nJ/bit*
Sleep	0

Table 3.1: The classical energy consumption model

This model has not been verified against the behavior of a physical radio in a wireless sensor network.

Shih *et. Al* [58] presented energy model developed for a specific platform, the μAMPS Wireless Sensor Node. The platform has a StrongARM SA 1110

microprocessor with a clock speed from 59 Mhz to 206 Mhz. The model takes into consideration the energy consumed by the microcontroller, the energy lost due to leakage and the average consumption of the radio [58]. Table 3.2 summarizes the model characteristics.

State	SA-1110	Sensor A/D	Radio	Pk (mW)
Active	Active	Sense	Tx/rx	1040
Ready	Idle	Sense	Rx	400
Monitor	Sleep	Sense	Rx	270
Observe	Sleep	Sense	Off	200
Deep Sleep	Sleep	Off	Off	10

Table 3.2: Energy consumption model of μAMPS

The μAMPS model does not specify the power consumed in transmitting or receiving one bit. Nonetheless, the platform uses transmission rate of 1 Mbps, so the energies required for transmitting and receiving one bit can be calculated as follows:

Time to send or receive one bit = 1 / 1 Mbps = 1 μsec

*Energy = Power * Time (**)* where Power is in Watts and Time is in seconds.

Replacing the time value and the values taken from the table in the equation (**), it is possible to find the energy required to transmit one bit as followed:

$Energy_{Txonebit} = 1040 * 1 * 10^{-3} W * 1 * 10^{-6} sec = 1.04\ \mu J/bit$

Repeating the same method, we find:

$Energy_{RxonebitReadystate} = 0.4\ \mu J/bit$

$Energy_{RxonebitMonitorstate} = 0.27\ \mu J/bit$

In the present research work, we are going to benefit from modeling to evaluate the energy consumption in the sensor nodes. The corresponding process will be detailed in chapter 6.

3.4 WSN energy consumption due network traffic

In the previous sections, we discussed the main causes of energy consumption inside the sensor node. It is concluded that the transceiver unit is considered the main contributor to energy waste. Therefore, any additional network traffic or activities that affect this unit will lead to additional energy consumption. This section will present in-network activities that affect energy

consumption in WSN. Later, the mechanisms used to contradict this affect and to save energy in the network will be discussed in section 3.5.

The different sources of energy waste that are related to network traffic can be classified into: idle listening, collisions, overhearing, and overhead. These categories are discussed in more details below [59] [60].

a) Idle Listening

Occurs when a node listens to the channel for a possible reception, but nothing is received. Idle listening has been identified as the major source of energy waste in a sensor node, mainly due to the low traffic loads commonly found in WSNs.

b) Collisions

When two (or more) nodes transmit simultaneously, the recipient may be unable to decode any of the packets involved. This problem implies that the senders waste energy through transmitting and that the receiver expends energy receiving without obtaining any benefit, as senders may eventually retry transmission. In the presence of hidden terminals, this effect is even more important because transmitters cannot sense each other.

c) Overhearing

This source of energy waste occurs when a sensor node wastes energy receiving a packet that is intended for a different destination.

d) Overhead

Data packets in WSNs are usually small; therefore, headers and other types of overhead (such as control messages) imply a high-level of energy waste for WSNs.

3.5 Power conservation mechanisms

Energy consumption reduction in WSN has attracted the interest of both the academic and the industrial worlds. The next subsections are devoted to present the mechanisms exploited to reduce energy consumption in WSNs.

Mechanisms used to reduce energy consumption in WSNs can be categorized into two main categories: passive mechanisms, and active mechanisms.

Passive power conservation mechanisms reduce the energy consumption of a sensor node by turning off its transceiver interface module when there is no communication activity [61]. Additional energy savings can be achieved by

optimizing the performance of the processor in an active state changing its operational frequency [62].

Active power conservation mechanisms differ from passive ones in that they achieve a reduction of the energy consumption by avoiding undesired events such as collision. For instance, adjusting the transmission power may help minimizing the probability of occurrence of a collision, an event that leads to higher power consumption due to the related detection and retransmission activities [44]. Energy-aware routing protocols are also exploited to reduce energy consumption in WSNs. Some researchers [63] have claimed that multi-hop network implementations consume less energy than an equivalent single-hop network. Conversely, other researchers [64, 65, 66] argue that single-hop implementations consume less energy in relaying data compared to equivalent multi-hop networks due to simpler routing protocols, lower communication overhead, and higher overall efficiency.

There is still much ongoing research on how to optimize power usage in battery-limited sensor networks. However, none of the proposed solutions is universally applicable. The main idea in energy-efficient mechanisms is to avoid the causes of energy waste such as idle listening, overhearing, and overhead.

We will focus on the power saving mechanisms which will be used in our approaches. These mechanisms are: (i) radio optimization; (ii) data aggregation; and (iii) energy-efficient routing, detailed in the following subsections.

3.5.1 Radio optimization

Since the radio module is considered the main component that causes battery depletion, researchers have tried to reduce energy waste due to wireless communication by optimizing radio parameters such as coding and modulation schemes, transmission power and antenna direction [67].

Idle states are major sources of energy consumption at the radio component. Sleep/wakeup schemes aim to adapt node activity to save energy by putting the radio in sleep mode for as much time as possible since sleep mode consumes substantially less energy than the other available modes (idle, transmitting or receiving). In sleep mode, a sensor node is not able to receive/transmit packets from/to the medium. This solution greatly reduces the idle listening energy waste. However, specific mechanisms should be defined to ensure that a sensor node will be awake if a node sends something to it [59].

The sleep/wakeup schemes can be categorized into: **a) Duty cycling schemes** which schedule the node radio state depending on network activity in order to minimize idle listening and favor the sleep mode. These schemes are

usually divided into three categories: *on-demand, asynchronous, and scheduled rendezvous*. In the on-demand approach, the node wakes up only when other node wants to communicate with it. In the *asynchronous* approach, the node wakes up at the same time as its neighbors according to a wakeup schedule then the nodes go to sleep until their next rendezvous. Finally, in the scheduled rendezvous, the node wakes up independently but its active period must overlap with its neighbors [68].

Duty-cycle based protocols are certainly the most energy-efficient but they suffer from sleep latency because a node must wait for the receiver to be awake. Moreover, in some cases it is not possible for a node to broadcast information to all of its neighbors because they are not active simultaneously [67]. **b) Passive wake-up radios:** where power radios are used to awake a node only when it needs to receive or transmit packets while a power-hungry radio is used for data transmission. Such approach is used in [69]. **c) Topology control:** when sensors are redundantly deployed in order to ensure good space coverage, it is possible to deactivate some nodes while maintaining network operations and connectivity. In such approach, nodes that are not necessary for ensuring connectivity or coverage can be turned off in order to prolong the network lifetime. In [70], this approach is used to save energy in the network and to prolong its lifetime.

3.5.2 Data aggregation

Data aggregation has been put forward as an essential paradigm for wireless routing in sensor networks. The idea is to combine the data coming from different sources eliminating redundancy, minimizing the number of transmissions and thus saving energy. This paradigm shifts the focus from the traditional *address-centric* approaches for networking (finding short routes between pairs of addressable end-nodes) to a more *data-centric* approach (finding routes from multiple sources to a single destination that allows in-network consolidation of redundant data) [71].

Data aggregation includes the following three approaches:

a) Tree-based approach

This approach performs aggregation by constructing an aggregation tree, which could be a minimum spanning tree, rooted at sink and considering source nodes as leaves. Each node has a parent node to forward its data. Flow of data starts from leaf nodes up to the sink and therein the aggregation is carried out by parent nodes [72].

b) Chain-based approach

The key idea behind chain-based data aggregation is that each sensor only transmits to its closest neighbor. This procedure continuous till the data reaches the sink or the base station [73]. PEGASIS protocol [74] uses this approach.

c) Cluster-based approach

In the cluster-based approach, as illustrated in Fig 3.2, the whole network is divided into several clusters. Each cluster has a cluster-head (CH), which is selected among cluster members. Cluster-heads do the role of aggregator, which consists of aggregating the data received from cluster members locally and then transmitting the result to the sink. The cluster approach is considered a special case of the tree approach [75]. This approach was adopted in many protocols such as LEACH [76], HEED [77], TEEN [78], EECDA [79], and SEP [80].

Cluster techniques have been proposed to enhance energy efficiency because they help to limit energy consumption via different means: (i) reducing the communication range inside the cluster ,thus, requiring less transmission power, (ii) limiting the number of transmissions thanks to fusion performed by the CH, (iii) reducing energy-intensive operations such as coordination and aggregation to the cluster head, (iv) providing the ability to power-off some nodes inside the cluster while the CH takes forwarding responsibilities; and (v) balancing energy consumption among nodes via CH rotation. In addition to energy-efficiency, cluster architectures also improve network scalability by maintaining a hierarchy in the network [81] [82].

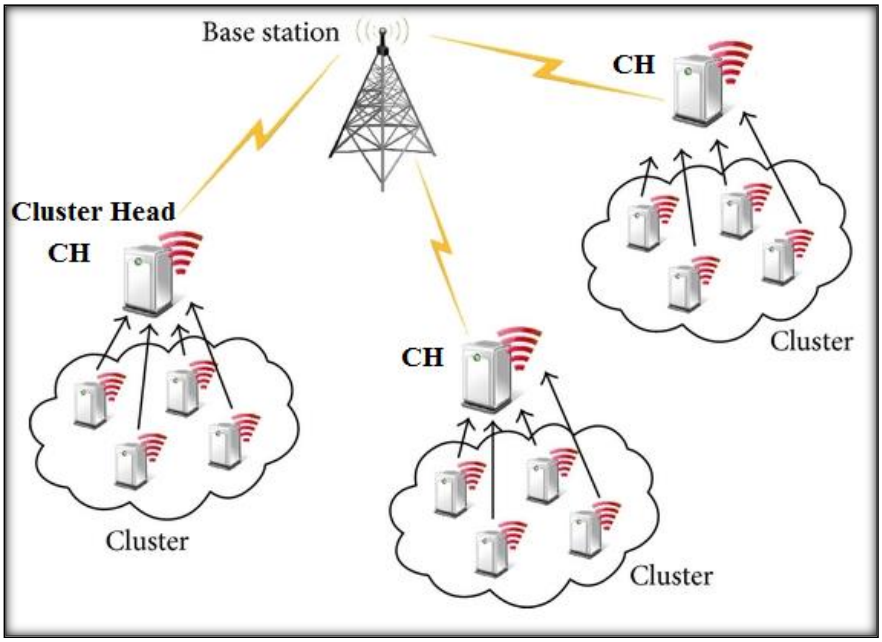


Fig 3. 2: Cluster-based aggregation

Routing can seriously drain nodes' energy. In particular, in multi-hop schemes, nodes closer to the sink have to route more packets. Therefore, their battery depletes faster. In what follows, we discuss some energy-saving mechanisms of different routing paradigms.

a) Energy as a routing metric

A solution proposed to extend the lifetime of sensor networks is to consider energy as a metric in the setup path phase. By doing so, routing algorithms do not only focus on the shortest paths but can select the next-hop based on its residual energy as shown in Fig 3.3. Recently, Liu et al. [83] introduced two new energy-aware cost functions. The Exponential and Sine Cost Function based Route (ESCFR) function can map a small change in the remaining nodal energy to a large change in the cost function value. By giving preference to sensors with higher levels of remaining energy during route selection, the function enforces energy balance. The Double Cost Function based Route (DCFR) protocol considers the energy consumption rate of nodes in addition to their remaining energy.

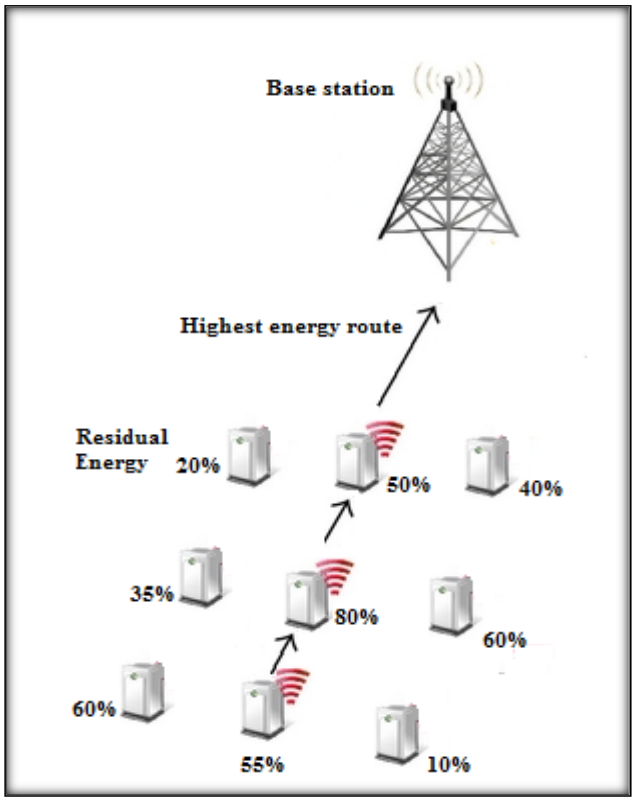


Fig 3. 3: Energy as routing metric

b) Multipath routing

Multipath routing enables energy to be balanced among nodes by alternating the forwarding nodes. As an example, the EEMRP (Energy-Efficient Multipath Routing Protocol) [84] discovers multiple node-disjoint paths using a cost function depending on the energy levels and hop distances of the nodes and allocates the traffic rate to each selected path. The EECA (Energy-Efficient and Collision Aware) protocol [85] constructs two node-disjoint and collision-free routes between a source and a sink. Multipath routing protocols also enhance network reliability by providing multiple routes, which enables the network to recover faster from a failure, whereas in single path schemes when a node runs out of power, a new route must be recomputed. Multipath routing is illustrated in Fig3.4.

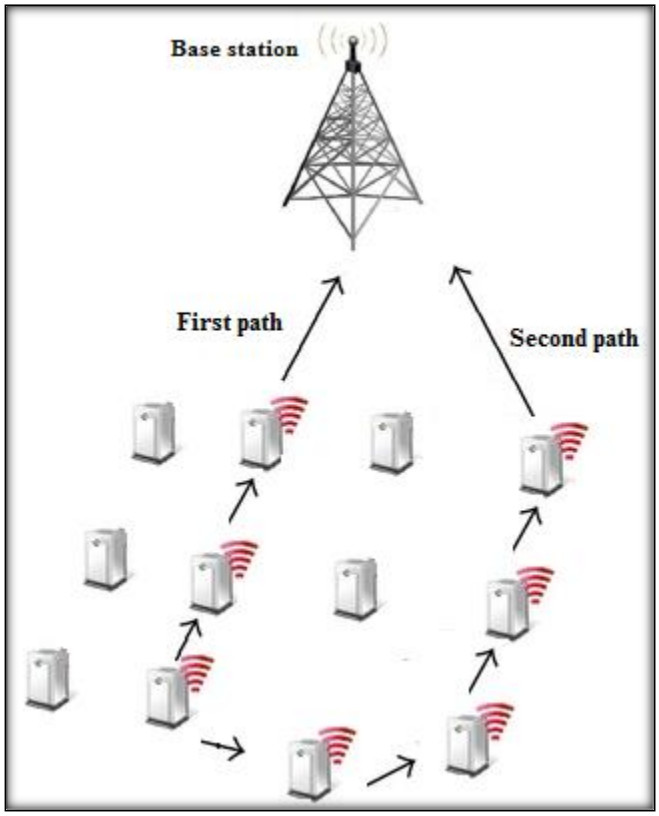


Fig 3. 4: Multipath routing

c) Relay node placement

The premature depletion of nodes in a given region can partition the network or create energy holes. Sometimes, this situation can be avoided thanks to the optimal placement of nodes through an even distribution or by adding a few relay nodes with enhanced capabilities. This helps to improve energy balance between nodes, avoid sensor hotspots and ensure coverage and connectivity [86].

Several works have focused on finding the minimum number of relay nodes or placing them optimally to prolong the network lifetime [87] [88].

d) Sink mobility

Most of the traffic in the sensor network is forwarded towards a sink, which increases the workload of the nodes closer to the sink. Thus, these nodes become hotspots and consume their energy quickly. The hotspot problem causes an early rupture of the network because the hotspot nodes no longer relay data for the farther nodes when they run out of their energy [89].

To increase network lifetime, it is possible to balance the load between nodes using a mobile base station, which moves around the network to collect node information. Sink mobility also improves connectivity in sparse architectures and enhances reliability because communication occurs in a single-hop fashion. Thus, it reduces contention, collisions and message loss [90]. When controllable, this mobile displacement can be studied to prevent high latency, buffer overflow and energy depletion [91, 92]. However, the sink mobility approach can create different problems. For example, in applications that gather data periodically, the purpose of employing a mobile sink is usually to improve flexibility and prolong network lifetime while gathering sensed data timely, reliably and efficiently. However, the sink cannot move in a random way, which is energy-unconscious and only a little beneficial to balancing energy consumption [93, 94]. Furthermore, a mobile sink that moves along fixed tracks [95-98] lacks flexibility and scalability because its moving path has to be redesigned for different networks. Autonomous moving schemes, in which a sink makes moving decisions according to the circumstances at that time, can provide reasonable adaptability to various types of network conditions.

3.6 Summary

In this chapter, we presented, sensor node's architecture, the power consumption sources inside the node and in the network, the methods adopted to estimate energy consumption in the node, and the different techniques proposed to save energy in WSN. These techniques, including radio optimization, data aggregation, and energy-efficient routing were discussed in detail.

In our attempt to make selective remote reprogramming as energy-efficient as possible, these three methods will be used. In the following chapter, remote reprogramming challenges and protocols will be discussed thoroughly to precede the explanation of the strategies proposed here to reduce power consumption in the selective remote reprogramming process.

4. WSN Remote Reprogramming

After deploying the network nodes, and as a result of change in the environment or in the application, it might be necessary to update the code running on sensor boards or to distribute management commands to sensors. However, due to the harsh hazards of the environments where these networks are deployed, it might be impossible to reprogram these nodes manually. Therefore, online or remote reprogramming is suggested as the best solution to achieve such modifications on the nodes [99]. Remote reprogramming might be applied to the whole network or just to subset of nodes (selective reprogramming), either way, it is crucial to provide reliability for such procedure. Therefore, most of the approaches proposed to perform remote reprogramming resort to flooding the whole network, leading to a major waste of energy in each node.

When dealing with selective reprogramming, the waste of energy increases steeply since only a small number of nodes need to get the update messages, while these messages might be received and retransmitted from all nodes resulting in a waste of resources.

As mentioned in Chapter 1, the present research aims to conserve energy as much as possible while performing selective reprogramming. Therefore, this chapter will discuss online remote reprogramming (RR) and selective remote reprogramming (SRR), the challenges and requirements that these procedures should satisfy, and the currently used protocols with their advantages and disadvantages.

4.1 Remote reprogramming in wireless sensor network

Software update research is a key topic in wireless sensor networks. The need to update the update mechanism itself, to reconfigure parameters, and to provide users with standardized features to manage online changes have also been identified, along with the need for overall update management [99].

The first step in reprogramming a sensor node is to decide when reprogramming is needed and which part of the code needs updating. This can be decided and initiated by the system user, or automatically (in a distributed manner) by the nodes themselves. In the former case, the system user issues a reprogramming command along with a set of attributes and the nodes operate in a slave-like mode. In the latter, the nodes should realize whether a code update is needed on their own. Coming to the second case, a straightforward solution

would be to periodically announce a profile of the code they run, so that their neighbors can compare the received information with the current version of the code to figure out whether a code update is needed or not [100].

Most remote reprogramming protocols are based on an algorithm named Trickle [101], which uses “polite gossip” to exchange code metadata with nearby network neighbors. Trickle breaks time into intervals, and at a random point in each interval, it considers broadcasting its code metadata. If Trickle has already heard several other motes gossip the same metadata in this interval, it politely stays quiet: repeating what someone else has said is rude. When a mote hears that a neighbor is behind the times (it hears older metadata), it brings everyone nearby up to date by broadcasting the needed pieces of code. When a mote hears that it is behind the times, it repeats the latest news that is aware of (its own metadata), following the first rule, this triggers motes with newer code to broadcast it [101].

4.2 Remote reprogramming challenges

The main challenge in remote reprogramming is that it requires reliable delivery. This involves two requirements: every node in the network must receive the program code, and it must be received in its entirety. This is very different from traditional sensor network applications, in which, occasional loss of data is tolerable [102]. High communication bandwidth is also needed in network reprogramming. For the vast majority of sensor network applications, the generated sensing data from an individual sensor node is small, usually of the order of bytes, and thus easily fits the low wireless radio bandwidth. However, delivering the entire program image, of the order of kilobytes over low-bandwidth wireless radio, as required in network reprogramming, requires significant bandwidth [102].

The problem of concurrent senders might appear in remote reprogramming because every node that has the new code image is a potential sender. Thus, it is likely that too many senders are transmitting at the same time, causing collisions, congestion and possibly failure of reprogramming. Energy consumption is also considered a problem due to sensor nodes power supply limitation, therefore, the amount of energy consumed during the reprogramming process may directly affect network lifetime [102].

Additionally, a reprogramming protocol must take into consideration the large amount of data needed to be transferred compared to the memory available on the node [100], and the time required to distribute the data for

reprogramming should be minimized [100]. Sufficient support for incremental upgrades should also be provided since program data often evolves slowly [100].

4.3 Energy consumption sources in reprogramming process

Several key factors affect reprogramming process efficiency and reprogramming lifetime of a WSN, including the transferred code size, the loading cost, and the reprogramming voltage requirement [103]. First, the transferred code size greatly impacts on transmission overhead. For example, to make a WSN reprogrammable, Deluge [2] disseminates the entire program image, including the application code, the TinyOS kernel, and the reprogramming protocol. Second, the loading cost for executing the new code needs to be minimized. In particular, hardware reboot should be avoided as it wastes energy as well as dropping data. A system normally stores many state data such as routing tables [104], synchronization tables [105], dissemination keys, etc. After rebooting, some nodes may take hours to fully come back online. Third, it is beneficial to avoid flash writes during reprogramming as writing to flash requires a much higher voltage than the minimum operational voltage commonly used, for instance, in TelosB nodes [103].

There are also other factors that affect energy consumption without being particularly related to the reprogramming process such as idle listening (which is a major source of energy waste [106]), message collision, overhearing, control message overhead and the number of messages sent and received.

4.4 Remote reprogramming protocols

A large number of remote reprogramming protocols have been proposed recently. Depending on their main focus, these protocols can be classified as reliable protocols, fast protocols or energy-efficient protocols. The protocols these categories encompass are illustrated in Table 4.1, and discussed in the following subsections.

Reliable RR protocols	Fast RR protocols	Energy-efficient RR protocols

Reliable RR protocols	Fast RR protocols	Energy-efficient RR protocols
• Flooding • Deluge [2] • XNP [110]	• SYNAPSE++ [112] • Splash [113]	• Stream [120] • MOAP [119] • MNP [102] • Freshet [118] • Hermes [121] • Rsync [123] • Zephyr [122] • Infuse [116] • Elon [103]

Table 4.1: Remote reprogramming protocol categories

4.4.1 Reliable remote reprograming protocols

Many remote reprogramming approaches were proposed having reliability of code dissemination as their main concern, depending heavily on broadcasting to achieve that and leading to solutions that still evince limitations regarding energy efficiency.

A straightforward example of these protocols is classic flooding: a base node broadcasts new codes to its neighbors. Upon receiving the data, each node stores it and then rebroadcasts to its neighbors. However, in dense large-scale WSNs with limited energy, classic flooding is particularly costly resulting in serious redundancy, contention, and collision. This is called the “broadcast storm” problem [107], and it is mainly caused by two deficiencies: data redundancy and sender redundancy where some senders are redundant to cover a desired area [108].

Deluge [1] is also considered one of the main reliable protocols available but with no consideration for energy efficiency. It is a NACK-based protocol that relies on periodic advertisements to keep nodes informed of their neighbors' states; it uses a three-stage handshake protocol consisting of advertisement, request and data, where updated nodes advertise their code version and outdated nodes will request these nodes for the new code image. Deluge [1] provides high reliability, robustness and support for multi-hop network reprogramming while being simple to implement. However, it requires radio to be always on causing nodes to be in idle listening mode during the reprogramming process. This leads to a considerable amount of energy waste since idle listening is one of the major

sources of energy consumption in WSNs[2]. It also suffers from the hidden terminal problem [109], which occurs when two senders out of range of each other transmit packets to the same receiver at the same time, thus causing collisions at the receiver.

Broadcasting is also used extensively in XNP protocol [110]. To distribute the program image in XNP, the source broadcasts the entire image and then queries nodes for the lost packets. Nodes make requests if necessary. The source then rebroadcasts every packet requested by some node, allowing other nodes to snoop and fill their own missing slots [2]. This guarantees the delivery of the program image (the default code distribution scheme in TinyOS). The problem in XNP is that it only operates over a single hop and does not provide incremental updates of the code image [3]; it also requires all nodes to be within bidirectional communication range of the source [2].

Although these protocols guarantee one hundred percent reliability, they fail to accomplish reprogramming process in energy-efficient manner.

4.4.2 Fast remote reprogramming protocols

This type of protocols usually uses a mechanism called pipelining to reduce reprogramming completion time. In this mechanism, the code image is divided into several segments. A segment consists of many packets. The packet is the transferred object. Dividing code image aims to conserve memory needed to maintain the received data in the node and to allow the use of spatial multiplexing. Spatial multiplexing is the most important issue here, as it affects the completion time. In spatial multiplexing, different segments can be transferred simultaneously. These parallel transmissions lead to shorter completion time [111].

An example of fast reprogramming protocols is SYNAPSE++ protocol [112], which reduces reprogramming completion time significantly using an original pipelining strategy, coupled with a novel and distributed channel access mechanism called soft Time Division Multiple Access (soft TDMA). The authors improved the Fountain Code (FC) implementation of SYNAPSE by a joint design with the forwarding mechanism in order to maximize the number of errors that are corrected through overhearing, thus limiting the number of explicit retransmissions. SYNAPSE++ [112] features advanced boot loader and memory management modules, which allow the dissemination of binary images written in any operating system and make application and reprogramming software completely independent in terms of memory and variables. This protocol reduces the time required for reprogramming significantly. Pipelining is also used in Splash [113], which is scalable to large, multi-hop sensor networks and built upon

two recent works: Glossy [114] and PIP [115]. Splash eliminates the need for contention resolution by exploiting constructive interference and channel diversity to effectively create fast and parallel pipelines over multiple paths that cover all the nodes in a network. This is called tree pipelining. In order to ensure high reliability, Splash also incorporates several techniques, including exploiting transmission density diversity, opportunistic overhearing, channel-cycling and XOR coding. Evaluation results showed that Splash is faster than state-of-art dissemination protocols and achieves a reduction in data dissemination time by a factor of more than 20 compared to DelugeT2.

4.4.3 Energy-efficient remote reprogramming protocols

As discussed in the previous chapter, the mechanisms usually used to save energy in WSN are: radio optimization, data aggregation and energy-efficient routing. Most energy-efficient remote reprogramming protocols rely mainly on radio optimization to conserve energy. The techniques used are: putting nodes to sleep, reducing the size of transferred code or reducing the number of senders. For example, Infuse [116] takes advantage of turning off node's radio to reduce energy consumption. It disseminates data in TDMA slots assigned to each sensor. To recover from lost messages, *implicit acknowledgments* (received by listening to the transmissions of the successors of a sensor) are used. The authors presented two possible recovery algorithms based on modified sliding window protocols to use implicit acknowledgments. Infuse [116] recovers from random message losses caused by varying link properties and message corruption. It disseminates data in an energy-efficient manner reducing the number of message transmissions and receptions. Since Infuse uses a TDMA-based MAC protocol, sensors only need to listen to the radio in the slots assigned to their neighbors. In the remaining slots, sensors can turn off their radio. The main disadvantage of Infuse is that the two recovery algorithms the authors present have weaknesses. The first one (Go-Back-N) has a critical window size that must be satisfied, otherwise the latency will be increased. In the second recovery algorithm (Selective retransmission), the authors observed that the dissemination latency increases considerably in the presence of failed sensors [116].

Putting nodes to sleep to save energy is also used in MNP [102]. This protocol also tries to reduce power consumption by reducing the number of senders. The sender selection algorithm attempts to guarantee that in a neighborhood there is at most one source transmitting the program at a time. Furthermore, the sender selection process is greedy as it tries to select the sender that is expected to have the most impact. It also uses pipelining to enable fast data propagation. MNP [102] reduces collisions, the hidden terminal problem

and the active radio time of a sensor node by putting the node into “sleep” state when its neighbors are transmitting a segment that is not of interest. Although MNP saves energy, the code propagation process takes long time [117].

Another protocol that puts nodes to sleep is Freshet [118], which operates in three phases for each new code image: blitzkrieg, distribution, and quiescent. It aggressively conserves energy by putting nodes to sleep between the blitzkrieg and the distribution phases as well as the quiescent phase. Freshet [118] uses location information and reduces control message overhead, but it requires the node to keep data about the network topology, which is a problem in fast changing topology networks.

Reducing the number of senders in order to minimize the communication in the network is also used in Moap protocol [119], where the program image is divided into segments no larger than a communication packet. Nodes with a new program image advertise and other nodes make requests as necessary. Nodes make requests to specific nodes to help minimizing the number of senders. A NACK-based approach for requesting dropped packets is used to minimize state in the sender [2]. Moap supports multi-hop reprogramming but it does not leverage the pipelining effect with segments of the code image [3]; it disseminates code in a hop-by-hop fashion in which a node has to receive the entire program image before starting advertising and no sender selection mechanism is considered [102].

Many RR protocols tried to conserve energy by reducing the size of the code transmitted. The Stream protocol [120], developed in 2007, was based on this particular idea to save energy while reprogramming. Using the facility of having multiple code images on a node and switching between them, Stream pre-installs the reprogramming protocol as one image and the application program equipped with the ability to listen to new code updates as the second image. For a sample application, Stream reduces the size of the program image by 10 pages (48 packets/page) when compared to Deluge. Stream [120] reduces the reprogramming time, the number of bytes transferred, the energy expended, and the usage of program memory. There are increasing advantages of Stream over Deluge in presence of larger network sizes [120] as it avoids transferring the entire reprogramming component everytime a code update is carried out, as it was in Deluge. However, being a non-incremental reprogramming protocol, the whole binary image must be transferred every time reprogramming is accomplished.

The Hermes protocol [121] also reduces the size of transferred data by transferring only the *delta* between the old and new software, letting the sensor nodes rebuild the new software using the received delta and the old software. It

reduces the delta resorting to techniques to mitigate the effects of function and global variable shifts caused by software modifications. Then it compares the binary images at the byte level with a method to create a small delta that needs to be sent over the wireless network to all the nodes. Hermes [121] supports multi-hop incremental reprogramming. The experiments carried out proved that Hermes transfers up to 201 times less information than Deluge.

Transferring only the image or code difference is used in Zephyr [122] and Rsync [123]. Rsync [123] computes the difference between the old and new program images and divides the binary data into fixed size blocks; both sender and receiver compute the pair (Checksum, MD4) over each block. The problem in Rsync is that it can only reprogram a single hop network and does not mitigate the effects of function and global variable shifts resulting in a large delta and sensor nodes cannot afford to perform expensive MD4 computation [121]. Even more, Zephyr [122] reduces the size of the difference by using application level modifications to mitigate the effect of software component shifts and it also uses an efficient byte level comparison that compares the binary images to produce a small difference. Sensor nodes build the new image from the difference and the old image. Both Zephyr and Rsync transfers relatively small amount of data, therefore, reducing reprogramming time and energy.

The Elon protocol [103] introduces a new concept of *replaceable* component to reduce the transferred code size significantly. It also avoids the cost of hardware reboot with a novel software reboot mechanism. Moreover, it significantly prolongs the reprogramming lifetime by avoiding flash writes for TelosB nodes. Experimental results show that Elon transfers less information than Deluge, and Stream. The software reboot mechanism in Elon reduces the rebooting cost in terms of beacon packets and of unsynchronized nodes. In addition, Elon prolongs the reprogramming lifetime by a factor of 2.3 for TelosB node compared to existing approaches (including Deluge [2] and Stream [120]). The overhead of Elon is acceptably small for real world applications in terms of reprogramming cost, memory overhead, and execution overhead.

A different approach is used in EEORP [3] to reduce the code size. EEORP separates the reprogramming protocol as a single program image and implements the application as other program image equipped with the ability to listen to new code updates. During program image dissemination, only the application program image needs to be disseminated so the transmitting energy is reduced**.** EEORP improves the dynamic adjustment mechanism of advertisements transmit rate to preserve energy. Being based on Deluge, it can easily be integrated into Deluge and has good prospects.

Table 4.2 shows the different mechanisms used to save energy in each energy-efficient remote reprogramming protocol.

RR Protocol	Put nodes to sleep	Reduce code size	Reduce number of senders
Stream [120]		X	
MOAP [119]			X
MNP [102]	X		X
Freshet [118]	X		
Hermes [121]		X	
Rsync [123]		X	
Zephyr [122]		X	
Infuse [116]	X		
Elon [103]		X	

Table 4.2: Energy conserving mechanisms used in remote reprogramming protocol

4.5 Selective Remote reprogramming protocols

Despite the undeniable relevance of performing selective reprogramming in WSNs, only few protocols were proposed covering this facility. The following protocols are the most prominent. Socially-aware Dissemination of Code Updates [124] was applied in a real-world scenario with animals and humans being monitored using wireless sensor network (WSN) devices. The selective reprogramming for several nodes was carried out taking advantage of animals and humans social behavior. Code updates are relayed opportunistically from one individual to the other upon contact. Unlike existing approaches that propagate updates to the entire network, the authors limit dissemination as much as possible to the target nodes, taking advantage of a characteristic common to many mobile WSNs scenarios and the fact that the monitored individuals exhibit social behavior. The implicit structure of social interactions, once elicited, provides an effective tool for steering efficient routing decisions. This approach was able to reduce the network overhead but it is applied in limited scenarios.

Two enhanced versions of Deluge were proposed to perform selective reprogramming. the first is a Deluge extension for selective reprogramming of WSNs [4], which allows reprogramming the network in two modes: (i) reprogramming the entire network (as in Deluge [2]); and (ii) reprogramming

nodes selectively depending on the type of platform or nodes IDs. The protocol extension is fully compatible with the default Deluge [2] operations, has minimal impact on network traffic and offers a high degree of reliability. Despite the new features the protocol extension brings in, being the dissemination process based on Deluge, reducing overhead and optimizing energy are still open design issues. As an example, the type of packets used in reprogramming (DE and DF packets) introduces an additional overhead, which is considered a limitation regarding energy savings. These packets will be explained fully in the next chapter.

Another Deluge-based approach for selective reprogramming is called Multicast Deluge (MDeluge) [125], which can be used to disseminate the code image into a designated subnet of a wireless sensor network. MDeluge disseminates code in the sensor network using a tree which is formed when the sensor nodes send code request messages. A micro server keeps the code and sends it based on the requests received. The simulation of MDeluge shows that MDeluge performs better than Deluge to disseminate the code into designated sensor nodes. However, it is not suitable for all network scenarios as the solution requires too many operational assumptions.

The MCP protocol [126] also enables programming a designated group of nodes. Each node in MCP maintains a small table to record information of interest of known applications. The table enables sending out multicast-based code dissemination requests so that only a subset of neighboring sensors contributes to code dissemination. Compared to broadcasting based schemes, MCP greatly reduces signal collision and saves both the dissemination time and reduces the number of dissemination messages.

4.6 Summary

This chapter presented, remote reprogramming protocols, their categories, and the mechanisms used to conserve energy in them. Then an overview on the current selective remote reprogramming protocols was made.

As discussed in this chapter, most selective reprogramming approaches were based on Deluge, which is designed for a whole network reprogramming, making these approaches inefficient regarding energy consumption. Therefore, in the next chapter, we will present new methods to optimize selective remote reprogramming procedure in order to make it more energy-efficient while trying to fulfill remote reprogramming protocols' typical requirements such as reliability and compatibility.

5. Strategies to enhance selective reprogramming

When performing selective reprogramming in WSNs, two scenarios can be considered depending on the location in the network of the new code image to be disseminated. The first scenario is when the user wants to reprogram a subset of nodes for the first time and so it is necessary to send the new code image from the base station (BS) to the selected nodes. In this scenario, flooding the whole network with code messages should be avoided. Instead, the code image should be routed from the BS to the selected nodes through the path with the highest energy levels. During this process other nodes not involved in routing should turn their radios off, saving energy.

The second scenario is when some nodes in the network already have the new code image. In this case, the code image needs to be routed from these nodes to the selected nodes for reprogramming. In this scenario, we discuss two approaches: (i) when clustering is used in the wireless sensor network; and (ii) when the network is flat.

5.1 Revisiting selective reprogramming within Deluge

For the strategies discussed in this section we will take advantage of Deluge extension for selective reprogramming [4]. In this protocol, the authors propose two types of packets (DE, DF) in order to combine the normal operation of Deluge [2] with selective reprogramming. A DE packet corresponds to the packet originally sent in reprogramming, with minor changes in order to specify the type of reprogramming, the cardinality of the set of nodes to be reprogrammed using that packet, and the corresponding Node IDs. A DF packet, oriented to selective reprogramming, is designed to carry Nodes IDs in excess, i.e., Node IDs that cannot be transported in a DE packet.

As shown in Fig 5.1, the field Reprogram Type in DE packets determines whether the selective reprogramming is carried out through Node ID or platform type. Although these features increase the flexibility of selective reprogramming,

the format of DE and DF packets do not optimize the way node IDs are handled. In fact, the protocol reserves two bytes to identify each node ID, which leads to an increase in packet overhead and, consequently, to an increase in the number of DE and DF packets required for selective reprogramming. This increase is more significant when the list of node IDs to reprogram is long, urging for a more efficient handling of node IDs.

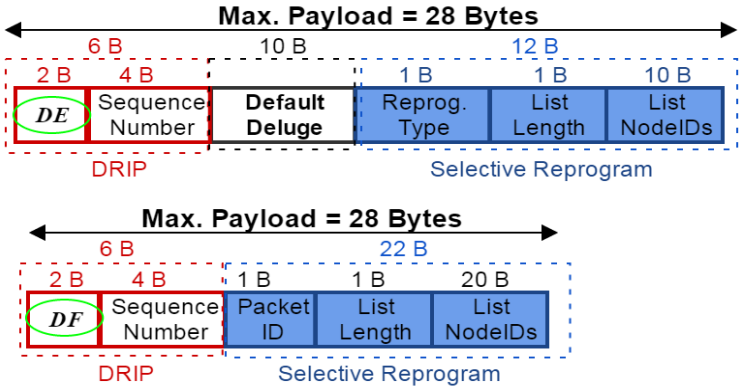


Fig 5. 1: Structure of DE and DF packets [3].

In order to make this procedure more energy-efficient, we suggest making the number of bits reserved to represent a node ID adjustable. This is accomplished defining a 4-bit long packet field called BPN (bits per node), allowing a maximum length of 16 bits to represent node IDs. This allows reducing DE and DF packets overhead in scenarios where only a small number of nodes with limited range of IDs is present. In practice, the number of DE and DF packets exchanged in the network will be reduced. The structure of the modified version of a DE packet is illustrated in Fig 5.2.

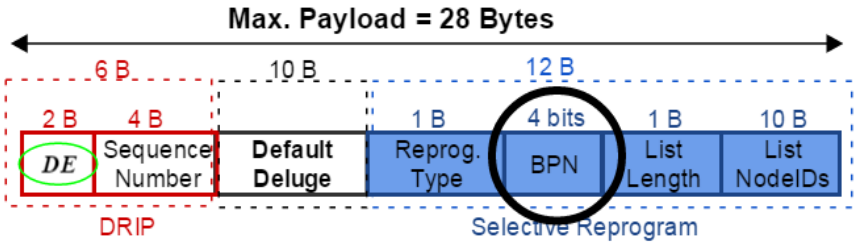


Fig 5. 2: DE packet modified structure.

The proposed strategies also require synchronization among sensor nodes to help determining the sleeping and wakeup periods for these nodes accurately.

In WSNs several metrics are commonly matter of concern, namely, the energy consumption in the nodes, the network lifetime, and the number of messages received and transmitted per node. The following subsections will

detail the two reprogramming scenarios mentioned above, assuming the use of Deluge extension for Selective Reprogramming [4] due to the flexibility it brings to Deluge.

5.2 Selective reprogramming using the BS

In this scenario, we assume that the selected node (SN) might be several hops away from the BS and the network is flat (no hierarchy). During the process of forwarding the code image from the BS to the SN, we propose an energy-aware algorithm for performing selective reprogramming divided in three phases: (i) Discovering possible paths; (ii) Choosing the best path; and (iii) Disseminating code, as shown in Fig 5.3.

Discovering possible paths (Phase 1) - In the first phase, the BS will sense the network to discover possible delivery paths up to the selected node. For this, each node in a path shall report its current energy to assist the routing decision. Thus, initially, the BS sends a specific message type named CP (Check Path), with the structure illustrated in Fig 5.4. CP messages, apart from allowing sensor nodes to perform energy and route pinning, allow the BS to inform the network on the number of packets required to send the new code image (code packets). This is information is relevant to evaluate the sleeping time some nodes may undergo for saving energy.

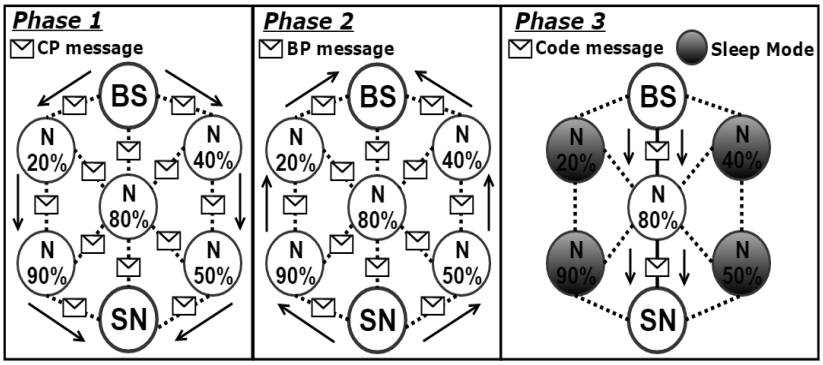


Fig 5. 3: Disseminating the code from the BS to the selected node.

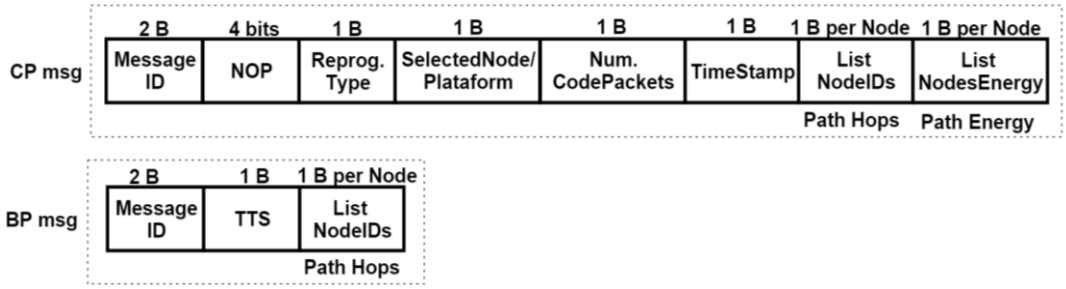


Fig 5. 4: CP and BP messages structure

As shown in Fig 5.4, the first two bytes in a CP message are used to define the message identifier. The NOH field is used to specify the maximum allowed number of hops that a path may have. This allows controlling the dissemination scope, i.e., any path exceeding NOH will be ignored. As mentioned before, the RepType field identifies the type of reprogramming, namely, if reprogramming will be based on the node identifier or on the platform type. When the BS sends a CP message, both the list of hops in the path and corresponding list of energy levels are empty. One byte is reserved for each element in these lists, allowing a total of ten NodeIDs as hops in the path. The number of CodePackets identifies the number of code packets that the BS should send to reprogram the node, while the TimeStamp field indicates when the CP message was originated. These fields are used in the second phase of the algorithm.

The first phase involves checking all possible paths in the network; thus, CP messages are flooded. When a node receives a CP message, it checks if it matches either the SelectedNode field containing the NodeID of the node to be reprogrammed or the platform type. If the node is not the selected one, then it checks the list of NodeIDs to verify if its NodeID is already in the list of hops in the path, before inserting it. The node also inserts its residual energy in the list of NodesEnergy field (Path energy) to allow for subsequent path decision. After that, the node will broadcast the modified CP message to its neighbors, which will handle the message in the same way. This procedure will be repeated in every hop so that nodes in the path to the selected node indicate their NodeIDs and residual energy. A node will drop a CP message if its own NodeID is already in the path to the selected node.

Choosing the best path (Phase 2) - The selected node will receive multiple CP messages reporting different paths and corresponding energy status. Several decision-making rules can be defined for helping the selected node to choose the best path. A straightforward approach is to establish a minimum energy threshold, below which a path is not eligible. For instance, the selected node may eliminate paths containing at least a hop with residual energy below 20%, and then choose the path with the maximum percentage of energy, on

average. This avoids the selection of a path including energy-constrained nodes. After choosing the best path, the selected node is expected to send a BP message (Best Path message) reporting that choice, see Fig 5.4. A BP message is also designed to convey relevant timing information so that nodes outside the best path enter into sleeping mode for a correct amount of time.

In more detail, as shown in Fig 5.4, the first two bytes of a BP message are used to identify the message type. The selected node can then set the sleeping time (TTS field) for the nodes outside the best path. This value is evaluated based on the number of code packets that the BS is expected to send, previously announced in the CP message. Thus, the nodes are able to turn their radio off for a certain time. This time will correspond to the number of code packets sent by the BS times the time a message takes to reach the selected node. This amount of time is calculated using TimeStamp value in the CP message, and corresponds to the time elapsed from sending a CP message and receiving it in the selected node

.

When a node receives a BP message it checks if its NodeID is in the list of NodeIDs of the message. If this is the case, the node knows it is in the best path; therefore, it will forward the BP message to its neighbors and will keep its radio on to forward the code packets to the node. Nodes that do not belong to the chosen path will forward the message and immediately turn their radio off according to TTS.

Disseminating code (Phase 3) - This phase starts as soon as the BS receives the BP message. Although the code dissemination can be performed using any dissemination protocol such as Deluge, the energy costs will be significantly reduced as the nodes that do not belong to the best path do not receive and transmit any messages.

5.3 Selective reprogramming using updated nodes

5.3.1 Selective reprogramming with clustering

In large scale WSNs, some level of hierarchy is expected to be present, being data aggregation the most common mechanism used for this purpose. In data aggregation, the network is divided into groups or clusters, and instead of making each node forward its data to the BS, data is sent to a group leader node, usually named cluster head (CH). The selection of a CH is dependent on the clustering protocol that is being used. Clustering protocols such as HEED [77] and Dynamic Multi Level Hierarchical Clustering [127] choose a CH depending on its residual energy and number of neighbors. In our approach, these protocols

are preferable due to their efficiency although choosing any other clustering protocol is also acceptable. Our objective is to take advantage of clustering to enhance selective reprogramming. We advocate the use of multiple senders instead of only one to perform code dissemination so that sender can contribute by sending a number of code messages according to its residual energy.

In the clustering scenario, we assume that the code to be disseminated already resides in other nodes within the same cluster as the SN. This assumption reflects a scenario which may well occur in practice, therefore, selective reprogramming should take advantage of existing up-to-date code versions distributed in the WSN. The proposed algorithm is divided into three phases: (i) Selective Reprogramming Setup; (ii) Selection of Senders; and (iii) Code Dissemination, as illustrated in Fig 5.5.

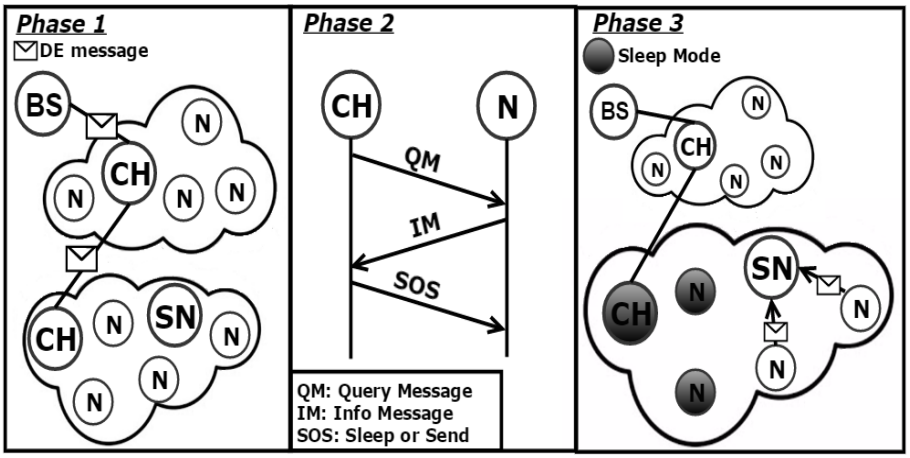


Fig 5. 5: Code dissemination in a WSN with clustering.

Selective Reprogramming Setup (Phase 1) - In this phase, the BS sends packets to the CH in order to identify in which clusters the selected nodes for reprogramming are located. This can be accomplished resorting to DE and DF packets [4].

Selection of Senders (Phase 2) - When a CH receives a DE or DF packet it will send a Query message to the nodes in its cluster asking for their identifiers, residual energy and the version of code image they are running. This control message has a minimal structure, as only a message type, and packet identifier are required. In response to Query messages, nodes will send an Info message to the CH. Based on the received messages; the CH determines which nodes in the cluster require reprogramming and which nodes can be selected as senders. If reprogramming is needed inside the cluster, the CH will divide the cost of updating the selected node by multiple senders. For this process, we propose a mechanism where the CH starts to eliminate potential senders with

energy less than a specific threshold if better candidate senders are in place. For example, any node with a new code version but with residual energy less than 20% of its initial energy can be neglected as a sender. This extends the lifetime of nodes that have a small amount of energy, as they will not be involved in the updating process. Then, the CH calculates the number of packets containing the new image that each sender is expected to send during the updating process. For this calculation the CH takes in consideration the energy of each sender and the number of packets that each sender should sent, i.e.:

$$\frac{\textbf{Sender residual energy} \times \textbf{Number of code packets needed for reprogramming}}{\textbf{Summation of the potential senders energy}}$$

The total number of code packets that need to be sent for updating a selected node can be taken from the DE packet, where the field "DefaultDeluge" (see Fig 5.1) contains information concerning the structure of the code image. Using the proposed mechanism, the number of code packets that each sender has to send is proportional to its remaining energy. It must be noted that all nodes residual energy is taken as a percentage of their initial energy. Fig 5.6 depicts the structure of Info message and Sleep or Send (SOS) messages that are used by the CH to control the updating process of the selected node.

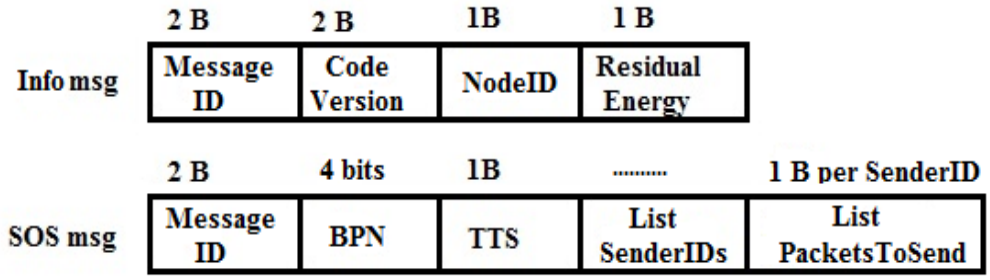


Fig 5. 6: Info message and SOS message structure.

The TTS (Time To Sleep) field in SOS messages is used to inform passive nodes, i.e., nodes neither being selected senders nor undergoing update, about the amount of time they should turn their radio off. The CH can determine this time by multiplying the number of code packets needed to update the node by their delivery time to the selected node (an approximation of the time between sending the Query message and receiving the last Info message). Each sender knows the number and order of packets to be sent depending of its NodeID position in the List of SenderIDs of the SOS message. For instance, if the first NodeID in the List of SenderIDs is 23 and the number of packets in the first position of the List of PacketsToSend is 10 means that node 23 must send the first 10 packets of the code image, i.e. both lists are co-indexed.

Code Dissemination (Phase 3) -In the third and last phase, the designated senders will start using Deluge extension [4] or Default Deluge [2] to disseminate the code image while the nodes that are not participating in the updating process will be in sleeping mode.

In presence of WSN clustering, the strategy proposed above is expected to improve the energy efficiency during selective reprogramming. The proposed solution takes advantage of: (i) using multiple collaborative senders, allowing the nodes to share the energy cost of updating a selected node; and (ii) putting the nodes that are not involved in reprogramming in sleeping mode during selective reprogramming.

5.3.2 Selective reprogramming in flat networks

In the previous scenario, we handled selective reprogramming in presence of clustering, assuming that the update code version resides in other nodes of the same cluster as of the SN. A similar approach can be used when the SNs in need of reprogramming are in a flat network, i.e., without clustering. In this case, the SN can play the same role as the CH in the previous scenario. When the SN node receives the modified version of a DE or DF packet it will send a Query message to its neighbors, and waits for Info message responses. The SN can then determine the best senders and the number of packets each one is expected to send, and will send SOS messages to its neighbors asking them for either turning their radio off or playing the role of senders.

5.4 Summary

In this chapter, strategies to improve energy-efficiency in remote reprogramming were proposed and discussed in detail. These strategies cover the two possible cases: (i) when the code update is disseminated for the first time from the base station; and (ii) when the code is present in updated nodes. A different strategy to disseminate the code in each case was proposed depending on how the network is structured (flat or clustered).

The following chapter will be dedicated to describe the implementation of these strategies, after which, the simulation results will be presented and conclusions will be deducted.

6. Strategies implementation and simulation results

To evaluate the performance of the proposed strategies, they were implemented and compared with both flooding and Deluge [2], using OMNeT ++ [128]. This choice was based on the various facilities OMNeT ++ provides such as the ability of creating user-defined messages and nodes. In our approaches several new messages have been defined and more than four different nodes have been reprogrammed.

In order to facilitate comparison among these different approaches, the sensor nodes used in the simulations are static. Adding mobility will be considered in future work.

As we discussed fully in Chapter 3, radio subsystem is the major source of energy waste in the sensor node. Hence, the performance metric considered for evaluation is related to the radio activity in a node, expressing the number of messages received and sent in that node. Further explanation will be presented in the following sections.

Deluge protocol was implemented in OMNeT++ by simulating Trickle algorithm with a minimum interval size (Imin) of 3 simulation time unit and a maximum interval size (Imax) of 5 simulation time unit. To get more accurate results, the results' values related to Deluge are taken as the average of ten different runs to accommodate the randomness Deluge is characterized by.

The following sections are ordered as followed. The first section will present the results on selective reprogramming using the base station, a strategy explained fully in (5.2). Three nodes will be taken into consideration for evaluation.

The second section is to present result on selective reprogramming using updated nodes, a strategy discussed in (5.3).

6.1 Results on selective reprogramming using the BS

For the first strategy (see 5.2) - *selective reprogramming using BS* –the network is flat, and the code is transmitted for the first time from the base station (BS) to the selected node (SN). A multihop scenario was considered including fifteen static sensor nodes with different percentages of residual energy. The

number of neighbors for each node varies from three neighbors to a maximum of eight depending on the node location. The nodes were programmed to forward only three CP messages with a maximum number of hops (NOH) set to six hops. Note that these values vary extensively according to the type of application where the WSN is used, where the network manager assessment plays a major part in determining these values. These particular numbers were chosen to suit the simulation limited scenario. This scenario is illustrated in Fig 6.1.

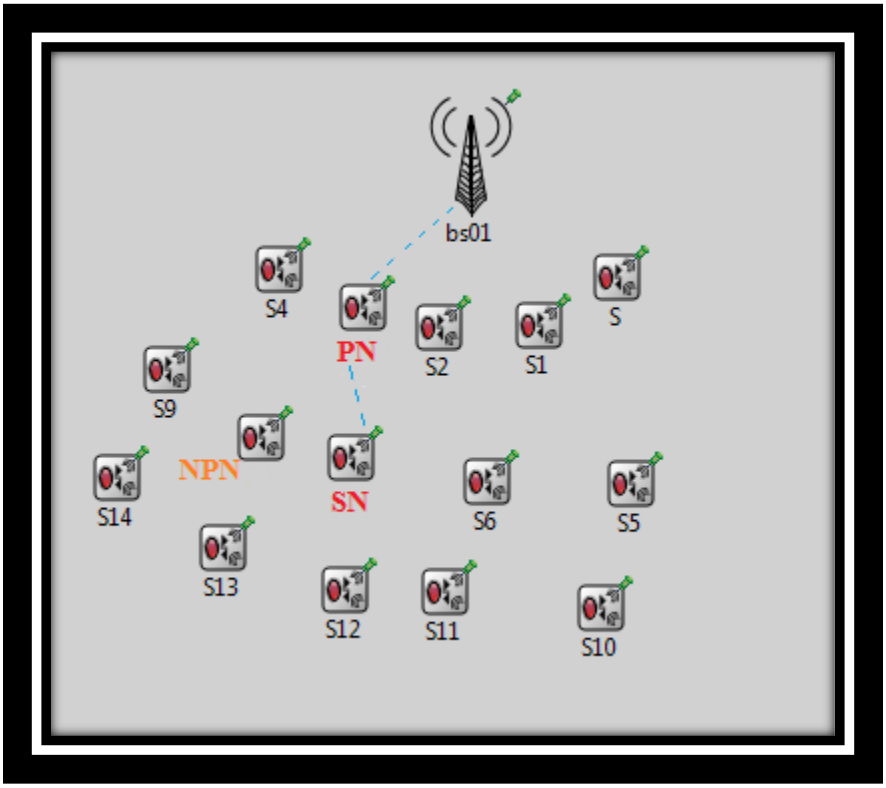


Fig 6. 1: Selective reprogramming using BS scenario.

The radio activity in nodes was analyzed for: the selected node (SN); a neighboring node that is outside the estimated best path (NPN); and finally, a node that belongs to the best path (PN). The locations of these nodes are shown in Fig 6.1.

Radio activity in these nodes was studied while varying the number of code packets needed to be transferred to the selected node. As mentioned before, the mean values were considered in order to obtain accurate results for Deluge simulation. The number of messages sent and received in the SN, NPN and PN are illustrated respectively in Figs 6.2, Fig 6.3, and Fig 6.4. The simulation results show a significant improvement when applying our strategy over both flooding and Deluge in terms of reducing radio activity in all three nodes, thus, reducing

power consumption in them. This improvement is mostly evident in the node that is not included in the best path (NPN) node, as indicated in Fig. 6.3.

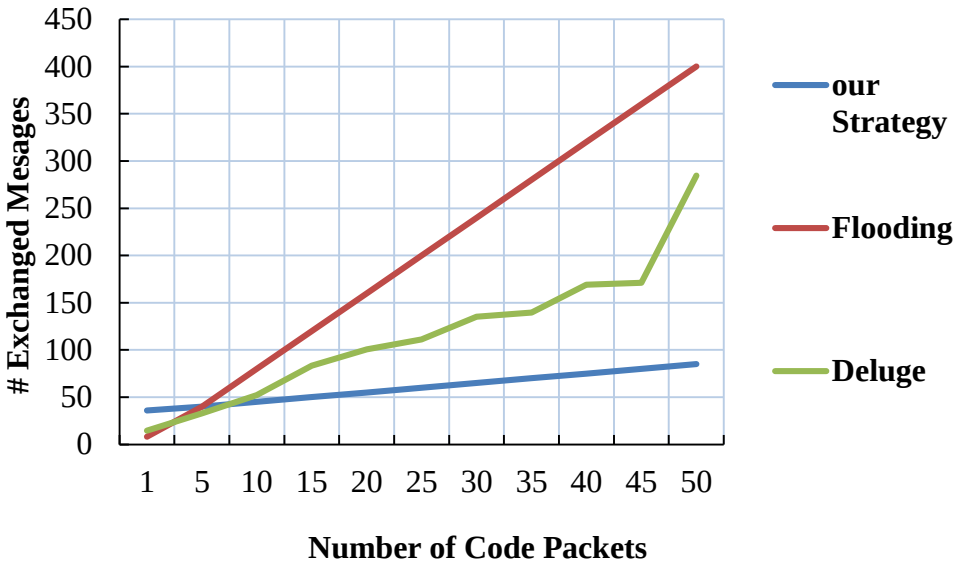


Fig 6. 2: Messages sent and received at SN.

As illustrated in Fig 6.2, the number of messages sent and received in the selected node SN increases proportionally with the increase of the number of code packets in both flooding and Deluge. Most of these exchanged messages are redundant, whereas applying the proposed strategy adds only an unchangeable number of packets as an overhead in the first phase (Discovering possible phases). This overhead can also be managed and reduced drastically changing the NOH field in the CP message. However, by reducing the number of hops, some high energy paths might be ignored, thus, determining the value of this field is very critical.

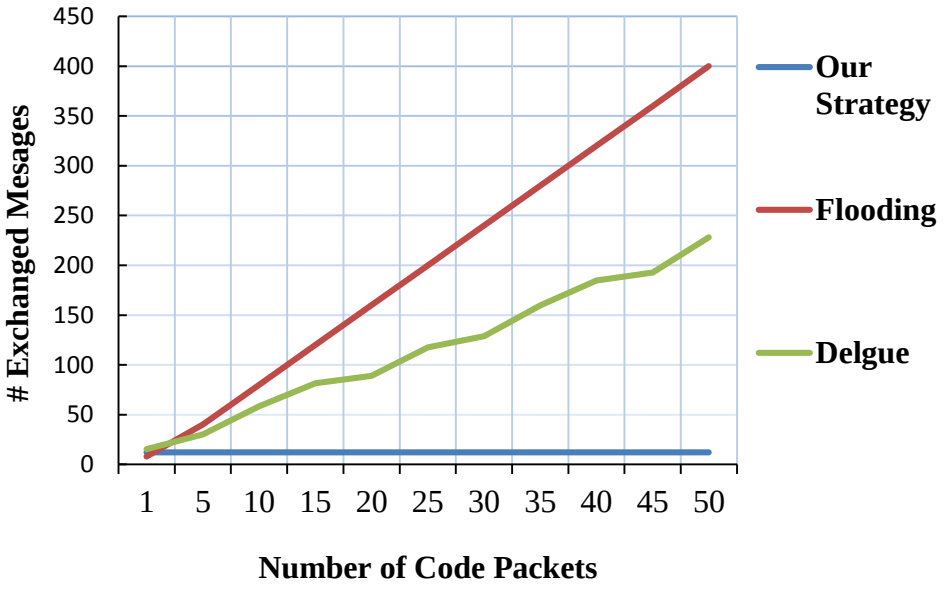


Fig 6. 3: Messages sent and received at NPN.

In the nodes that do not belong to the best path NPN (see Fig 6.1), the number of messages exchanged is static and represent the first phase's overhead. After this phase, these nodes are put to sleep for a specific amount of time; therefore, no further exchanges related to the reprogramming process are carried out in such nodes.

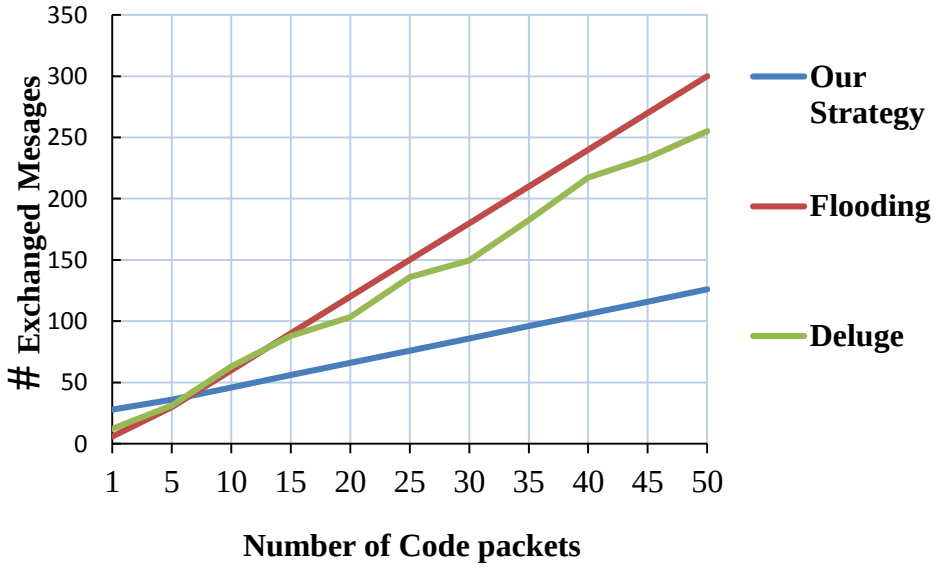


Fig 6. 4: Messages sent and received at PN.

As expected, for PN nodes (see Fig 6.4), the radio activity increases with the increase of code packets, however, the increase is still low in comparison to flooding and Deluge.

Since the third phase (Disseminating the code) can be accomplished using any of the dissemination protocols such as Deluge, the reliability of the reprogramming process using this strategy depends solely on the chosen protocol.

According to the classical energy consumption model or the energy consumption model of μAMPS discussed previously in chapter 3 (see Table 3.1 and Table 3.2), the energy required for one-bit transmission is equal to the energy required for reception in case the radio is in the Active mode. Therefore, in case of standardized messages size, we estimate energy consumption model in the radio subsystem for each of these nodes to be similar to the graphs representing radio activity, since all nodes' radio will be in the active mode when a new version of the code is disseminated for the first time using either Deluge or flooding approach.

It must be noted that this strategy works well in the case of reprogramming a single selected node. This is not advisable when reprogramming multiple nodes since each selected node might have a different best path, thus causing conflicts when putting nodes to sleep.

To solve this problem, two solutions are possible. The first solution is repeating this strategy for each node which is not a recommended solution since it will increase the reprogramming time proportionally to the number of selected nodes. The second solution is to use this strategy to reprogram one selected node, while the remaining selected nodes can be reprogrammed using the second strategy (selective reprogramming using updated nodes), since from that point on, two code sources exist: the base station and the recently updated node.

6.2 Results on selective reprogramming using updated nodes

For the second strategy (see 5.3.1) - *selective reprogramming using update nodes* - the network is clustered and some neighboring nodes already have the code to be transferred to the selected node SN. Fig 6.5 represents the clustered network used to simulate this strategy

In the initial simulation scenario, each network cluster is comprised of five cluster members and one cluster head (CH). These nodes are in bidirectional communication with each other. The cluster structure is illustrated in Fig 6.6.

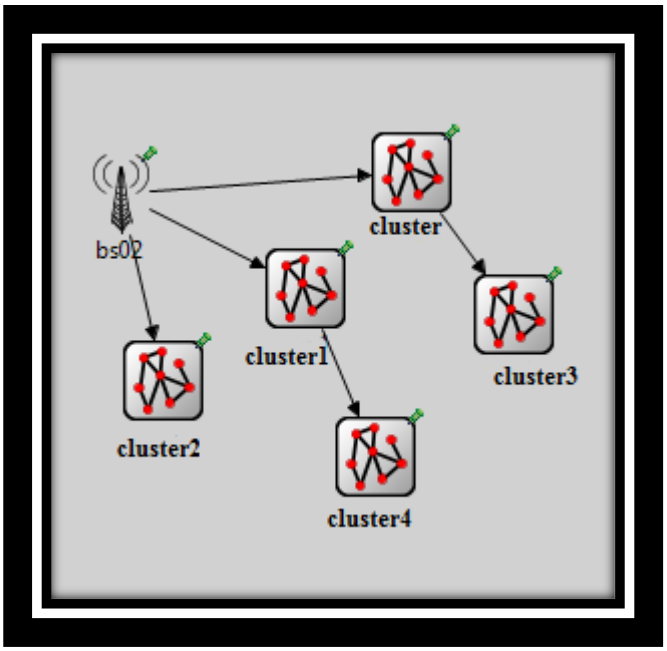


Fig 6. 5: Selective reprogramming using updated nodes scenario.

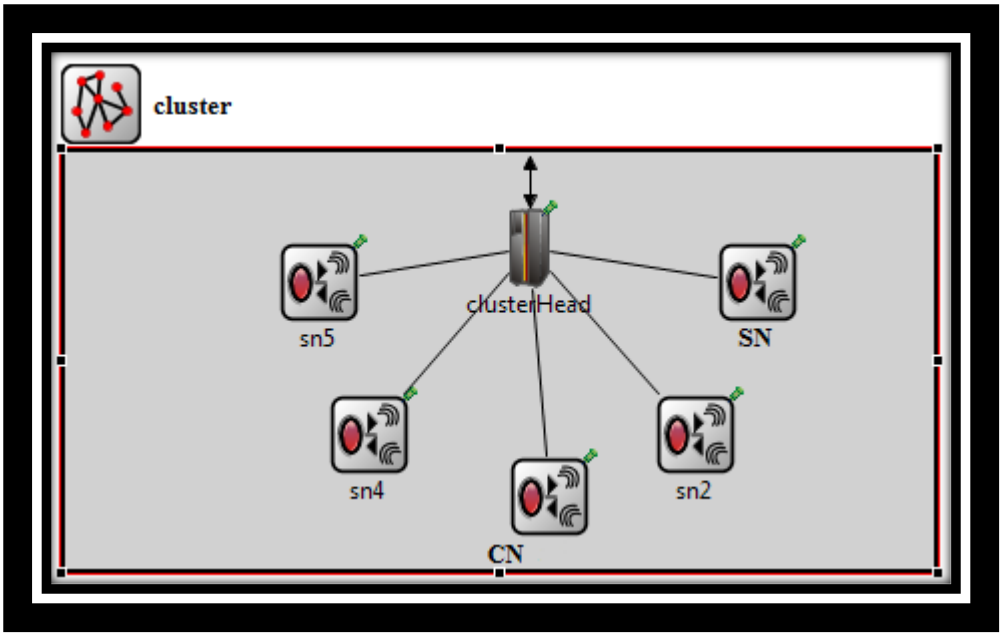


Fig 6. 6: The cluster structure in the simulation scenario.

The results below are for two cluster nodes, the selected node SN and a cluster member node (CN) with residual energy of 25%, making of it a potential sender. These nodes are shown in Fig 6.6.

Fig 6.7 and Fig 6.8 illustrate respectively the number of messages sent and received at both SN and CN while varying the number of code packets.

The results show that when applying the strategy based on cooperative senders, the radio activity of SN, in terms of the number of messages sent and received, is similar to the radio activity of this node in Deluge. However, the radio activity in other cluster nodes is reduced sizably when this strategy is implemented. It must be also noted that using (5.3) approach, where the code packets are sent via multiple senders, the corresponding power consumption is distributed among multiple nodes instead of draining a specific single node. The approach of having a single sender is adopted in all previous remote reprogramming protocols, where transmitting all code packets is assigned to only one specific node leading to a rapid depletion of this node and creating a hotspot problem.

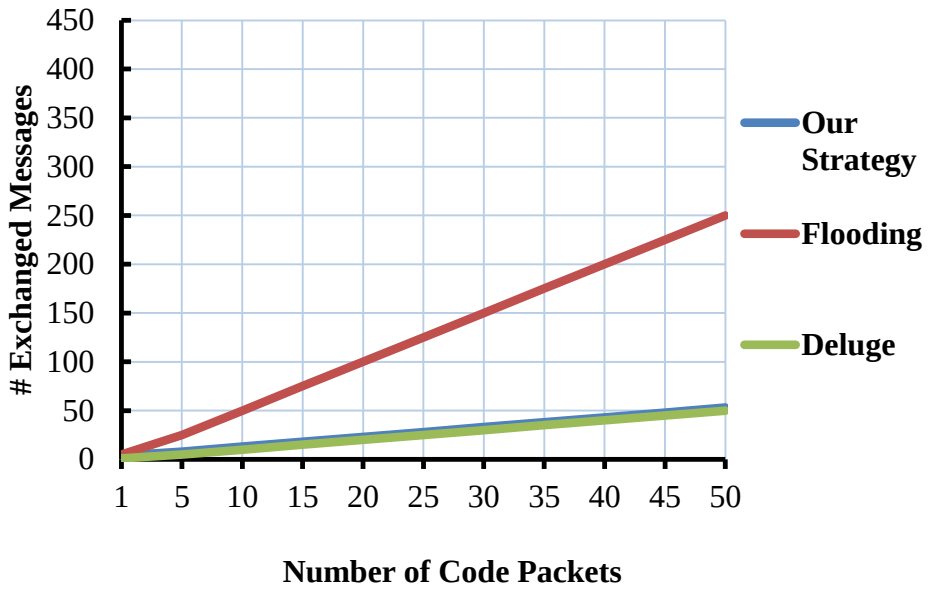


Fig 6. 7: Messages sent and received at SN.

The third phase of the strategy proposed is also based on Deluge [2] or Deluge extension [4] which makes reliability of the reprogramming process unaffected by the strategy proposed but related to the reprogramming protocol used. This also enables pipelining, thus, speeding up the reprogramming process.

As mentioned before, this strategy can also be used in a flat network with no clustering, whereas the selected node can play the rule of the cluster head (CH). This will increase the number of messages sent and received in the selected node slightly and the power consumption accordingly, hence the second phase of the strategy proposed which is the selection of senders will be assigned to the SN instead of the CH, increasing the activity in both CPU and radio unit. Meanwhile, the radio activity of the other cluster members will be identical to their peers in the clustered scenario.

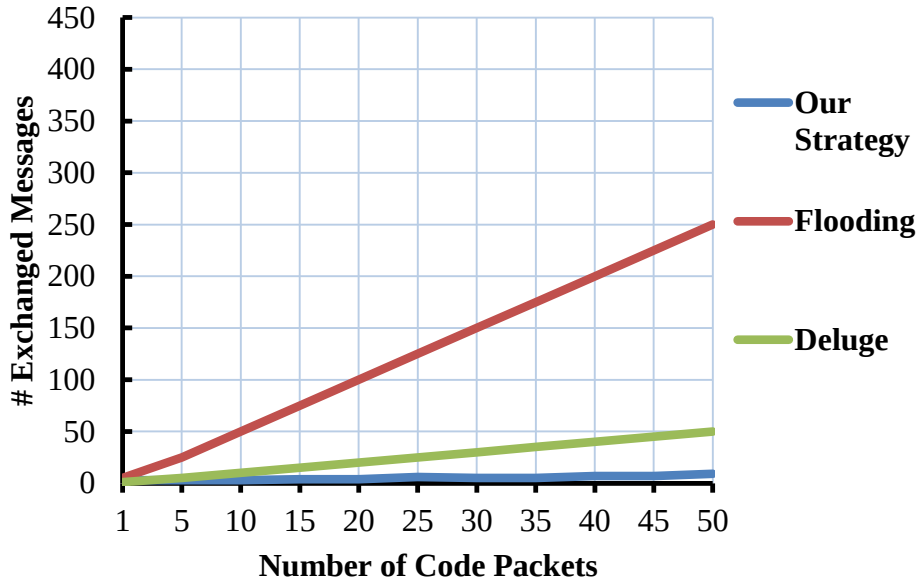


Fig 6. 8: Messages sent and received at CN.

6.3 Summary

In this chapter, the strategies proposed in chapter 5 to enhance selective remote reprogramming procedure, were implemented and compared with Deluge [2] and flooding approaches using OMNet++. The first strategy is for transmitting the code image from the BS to the SN, while the second strategy reuses an existing code image located in a nearby node. The obtained results show a significant reduction in the number of messages received and sent in the network nodes, leading to reduction of the power consumption and making the selective remote reprogramming more energy-efficient.

7. Conclusions and future work

In this thesis, new strategies have been proposed to improve selective reprogramming of WSNs in order to make the process more energy-efficient. Reducing energy consumption can be achieved avoiding blind flooding throughout the network and turning the radio off in specific nodes. Two scenarios were analyzed and discussed: (i) transmitting the code image from the BS to the SN; and (ii) reusing an existing code image located in a nearby node. In the first scenario, the proposal is to choose the path with the highest energy levels to transfer the code, putting nodes that do not belong to this path asleep for the whole reprogramming period. The second scenario is analyzed both for cluster-based and flat WSNs. The proposed solution is the use of multiple senders to transmit the code to the SN while forcing other nodes to sleep during the reprogramming process, thus, avoiding unnecessary reception of code messages in these nodes. In addition, dividing the code dissemination among multiple senders will help eliminating single sender exhaustion since transferring the code image may require the transmission of a large amount of data. The suggested approaches will also help avoiding the problem of having concurrent senders.

These approaches were tested and compared with typical flooding and Deluge solutions. The results show a significant reduction in the number of messages received and sent in the nodes, leading to reduction of the power consumption and making the selective remote reprogramming more energy-efficient.

Although Deluge [2] has become a standard reprogramming protocol for WSNs, the present proposals can be understood as contributions toward an energy-aware deployment of Deluge or Deluge extension for selective reprogramming [4].

As future work, the plan is to extend the study to networks with larger number of nodes, and analyze the impact of applying these strategies on the time needed to accomplish the reprograming process. Enhancing these strategies' adaptability to network topology changes is also considered as future work.

8. References

[1] Ahlawat, M., Mittal, A.: Different Communication Protocols for Wireless Sensor Networks: A Review. Ijarcce. 4, 213–216 (2015).

[2] Chlipala, A., Hui, J., Tolle, G.: Deluge: data dissemination for network reprogramming at scale. Univ. California, Berkeley, Tech. Rep. (2004).

[3] Xie, M.: EEORP: Energy-Efficient Online Reprogramming Protocol for Wireless Sensor Networks. [J]. Applied Mathematics & Information Sciences (AMIS), (2011)

[4] Lima, E., Carvalho, P., Gama, O.: A protocol extension for selective reprogramming of WSNs. In: Software, Telecommunications and Computer Networks (SoftCOM), 2015 23rd International Conference on. pp. 280–284. IEEE (2015).

[5] Zhu, X., Ma, Y., Yang, P.: On The Shortest Recycle Path Algorithm Of Wireless Sensor Node. 48, 1244–1247 (2013).

[6] Cerpa, A., Elson, J., Estrin, D., Girod, L., Hamilton, M., Zhao, J.: Habitat monitoring: Application driver for wireless communications technology. ACM SIGCOMM Comput. Commun. Rev. 31, 20–41 (2001).

[7] Juang, P., Oki, H., Wang, Y., Martonosi, M., Peh Li-Shiuan, P., Rubenstein, D.: Energy-Efficient Computing for Wildlife Tracking: Design Tradeoffs and Early Experiences with ZebraNet. Proc. 10th Int. Conf. Archit. Support Program. Lang. Oper. Syst. (ASPLOS 2002). 96–107 (2002).

[8] Ye, D., Gong, D., Wang, W.: Application of wireless sensor networks in environmental monitoring. In: In Power Electronics and Intelligent Transportation System (PEITS), 2009 2nd International Conference. pp. 287–291 (2009).

[9] Tripathy, A.K., Chinara, S.: Comparison of Residual Energy-Based Clustering Algorithms for Wireless Sensor Network. ISRN Sens. Networks. 2012, 1–10 (2012).

[10] Krishnamachari, L., Estrin, D., Wicker, S.: The impact of data aggregation in wireless sensor networks. In: Proceedings 22nd International Conference on Distributed Computing Systems Workshops. pp. 575–578 (2002).

[11] Estrin, D., Girod, L., Pottie, G., Srivastava, M.: Instrumenting the world with wireless sensor networks. in ICASSP. 2033–2036 (2001).

[12] Jain, N.: Energy Efficient And Cluster Based Routing Protocol for Wireless Sensor Network: A Review. International Journal of Advance Technology & Engineering Research. 22-24 (2011).

[13] Srivastava, N.: Challenges of Next-Generation Wireless Sensor Networks and its impact on Society. J. Telecommun. 1, 128–133 (2010).

[14] Mobility, B.S.: Mesh networks: Delivering IP-Based Seamless Mobility in Municipal and Ad Hoc Wireless Networks. Whitepaper. (2006).

[15] Eskandari, Z., Yaghmaee, M.H., Mohajerzadeh, A.H.: Energy Efficient Spanning Tree for Data Aggregation in Wireless Sensor Networks. 2008 Proc. 17th Int. Conf. Comput. Commun. Networks. 1–5 (2008).

[16] Intanagonwiwat, C., Govindan, R., Estrin, D.: Directed Diffusion: A Scalable and Robust Communication. Proc. 6th Annu. Int. Conf. Mob. Comput. Netw. (MobiCom '00). 56–67 (2000).

[17] Wenqi, G., Healy, W.: Power Supply Issues in Battery Reliant Wireless Sensor Networks: A Review. International Journal of intelligent control and systems vol. 19, NO. 1, 15-23 (2014)

[18] Bokareva, T., Hu, W., Kanhere, S.: Wireless sensor networks for battlefield surveillance. Proc. of The Land Warfare Conference, LWC 1–5 (2006).

[19] Krishnamachari, B., Estrin, D., Wicker, S.: The Impact of Data Aggregation in Wireless Sensor Networks. In Proc: 22nd International Conference on Distributed Computing Systems, p.575-578 (2002).

[20] Chung, W.-Y., Lee, Y.-D., Jung, S.-J.: A wireless sensor network compatible wearable u-healthcare monitoring system using integrated ECG, accelerometer and SpO2. In: 2008 30th Annual International Conference of the IEEE Engineering in Medicine and Biology Society. IEEE. pp. 1529–32 (2008).

[21] Jovanov, E., Milenkovic, A., Otto, C., de Groen, P.C.: A wireless body area network of intelligent motion sensors for computer assisted physical rehabilitation. J. Neuroeng. Rehabil. 2, 6 (2005).

[22] Mainwaring, A., Culler, D., Polastre, J., Szewczyk, R., Anderson, J.: Wireless Sensor Networks for Habitat Monitoring. Proc. 1st {ACM} Int. Work. Wirel. Sens. Networks Appl. 88–97 (2002).

[23] Kosmerchock, S.: Wireless sensor network topologies. [online] http://www.k5systems.com/TP0001_v1.pdf. (2014).

[24] Heidemann, J., Silva, F., Intanagonwiwat, C., Govindan, R., Estrin, D., Ganesan, D.: Building efficient wireless sensor networks with low-level naming. ACM SIGOPS Oper. Syst. Rev. 35, 146 (2001).

[25] Cheekiralla, S.: Wireless Sensor Network-Based Tunnel Monitoring. In Proc of: Workhop on Real-World Wireless Sensor Networks. (2005).

[26] Yick, J., Mukherjee, B., Ghosal, D.: Wireless sensor network survey. Comput. Networks. 52, 2292–2330 (2008).

[27] Khalil, I.M., Gadallah, Y., Hayajneh, M., Khreishah, A.: An adaptive OFDMA-based MAC protocol for underwater acoustic wireless sensor networks. Sensors (Switzerland). 12, 8782–8805 (2012).

[28] Heidemann, J., Wills, J., Syed, A.: Research challenges and applications for underwater sensor networking. IEEE Wirel. Commun. Netw. Conf. 2006. WCNC 2006. 1, 228–235 (2006).

[29] Feng, W., Code, B., Kaiser, E., Shea, M., Feng, W., Bavoil, L.: Panoptes: Scalable Low-Power Video Sensor Networking Technologies. Acm. 562–571 (2003).

[30] Chu, M., Reich, J., Zhao, F.: Distributed attention in large scale video sensor networks. In: Intelligent Distributed Surveilliance Systems, IEE. pp. 61–65 (2004).

[31] Wu, C., Chung, Y.: Heterogeneous Wireless Sensor Network Deployment and Topology Control Based on Irregular Sensor Model. In: Advances in Grid and Pervasive Computing: Advances in Grid and Pervasive Computing: Second International Conference, GPC 2007, Paris, France, May 2-4, 2007. Proceedings. pp. 78–88 (2007).

[32] Raghunathan, V., Schurgers, C., Park, S., Srivastava, M.B.: Energy-aware wireless microsensor networks. IEEE Signal Process. Mag. 19, 40–50 (2002).

[33] Kazienko, J.F., et al.: Practical evaluation of a secure key-distribution and storage scheme for wireless sensor networks using TinyOS. CLEI Electronic Journal (2011).

[34] Pottie, G.J., Kaiser, W.J.: Wireless integrated network sensors. Commun. ACM. 43, 51–58 (2000).

[35] Crossbow MicaZ mote specifications. http://www.xbow.com.

[36] SunSPOT mote specifications. http://www.sunspotworld.com.

[37] Rabaey, J.M., Ammer, M.J., da Silva, J.L., Patel, D., Roundy, S.: PicoRodio supports ad hoc ultra-low power wireless networking. Computer (Long. Beach. Calif). 33, 42–48 (2000).

[38] Roundy, S., Wright, P., Rabaey, J., 2004: Energy scavenging for wireless sensor networks: with special focus on vibrations. (2004).

[39] Starner, T.: Human-powered wearable computing. IBM Syst. J. 35, 618–629 (1996).

[40] Sim, W.Y.S.W.Y., Kim, G.Y.K.G.Y., Yang, S.S.Y.S.S.: Fabrication of micro power source (MPS) using a micro direct\nmethanol fuel cell (/spl mu/DMFC) for the medical application. Tech. Dig. MEMS 2001. 14th IEEE Int. Conf. Micro Electro Mech. Syst. (Cat. No.01CH37090). 00, 341–344 (2001).

[41] Mehra, A., Zhang, X., Ayón, A.A., Waitz, I.A., Schmidt, M.A., Spadaccini, C.M.: Six-wafer combustion system for a silicon micro gas turbine engine. J. Microelectromechanical Syst. 9, 517–527 (2000).

[42] Brodd, R.J., Bullock, K.R., Leising, R. a., Middaugh, R.L., Miller, J.R., Takeuchi, E.: Batteries, 1977 to 2002. J. Electrochem. Soc. 151, K1 (2004).

[43] Yeatman, E.M.: Advances in power sources for wireless sensor nodes.pdf. Proc. BSN 2004. 2 (2004).

[44] Moschitta, A., Neri, I.: Power consumption Assessment in Wireless Sensor Networks. ICT - Energy - Concepts Towar. Zero - Power Inf. Commun. Technol. 203–224 (2014).

[45] Dâmaso, A., Freitas, D., Rosa, N., Silva, B., Maciel, P.: Evaluating the power consumption of wireless sensor network applications using models. Sensors (Basel). 13, 3473–500 (2013).

[46] Merrett, G. V., White, N.M., Harris, N.R., Al-Hashimi, B.M.: Energy-aware simulation for wireless sensor networks. 2009 6th Annu. IEEE Commun. Soc. Conf. Sensor, Mesh Ad Hoc Commun. Networks, SECON 2009. (2009).

[47] Shnayder, V., Hempstead, M., Chen, B.-R., Allen, G.W., Welsh, M.: Simulating the power consumption of large-scale sensor network applications. In: Proceedings of the 2nd international conference on Embedded networked sensor systems - SenSys '04. p. 188 (2004).

[48] Shu, L., Hauswirth, M., Chao, H.-C., Chen, M., Zhang, Y.: NetTopo: A framework of simulation and visualization for wireless sensor networks. Ad Hoc Networks. 9, 799–820 (2011).

[49] Fournel, N., Fraboulet, A., Feautrier, P., Compsys, I., Lyon, E.N.S. De, France, L.F.-: Embedded Software Energy Characterization: using non-intrusive measures for application source code annotation. Computing. 3, 1–15 (2009).

[50] Heinzelman, W.R., Chandrakasan, A., Balakrishnan, H.: Energy-efficient communication protocol for wireless microsensor networks. In: Proceedings of the 33rd Annual Hawaii International Conference on System Sciences. pp. 3005–3014 (2000).

[51] Heinzelman, W.B., Chandrakasan, A.P., Balakrishnan, H.: An application-specific protocol architecture for wireless microsensor networks. IEEE Trans. Wirel. Commun. 1, 660–670 (2002).

[52] Raghunathan, V., Schurgers, C., Park, S., Srivastava, M.B.: Energy-aware wireless microsensor networks. IEEE Signal Process. Mag. 19, 40–50 (2002).

[53] Polastre, J., Szewczyk, R., Culler, D.: Telos: Enabling ultra-low power wireless research. In: 2005 4th International Symposium on Information Processing in Sensor Networks, IPSN 2005. pp. 364–369 (2005).

[54] Chipcon, SmartRF CC1000 Single Chip Very Low Power RF Transceiver.

[55] Chipcon, SmartRF CC2420, 2.4GHz IEEE 802.15.4/ZigBee-ready RF Transceiver

[56] Hempstead, M., Tripathi, N., Mauro, P., Wei, G.-Y., Brooks, D.: An Ultra-Low Power System Architecture for Sensor Network Applications. ACM SIGARCH Comput. Archit. News. 33, 208–219 (2005).

[57] Heinzelman, W.R., Chandrakasan, A., Balakrishnan, H.: Energy-efficient communication protocol for wireless microsensor networks. In: Proceedings of the 33rd Annual Hawaii International Conference on System Sciences. pp. 3005–3014 (2000).

[58] Shih, E., Cho, S.-H., Ickes, N., Min, R., Sinha, A., Wang, A., Chandrakasan, A.: Physical layer driven protocol and algorithm design for energy-efficient wireless sensor networks. Proc. 7th Annu. Int. Conf. Mob. Comput. Netw. - MobiCom '01. 272–287 (2001).

[59] Cano, C., Bellalta, B., Sfairopoulou, A., Oliver, M.: Low energy operation in WSNs: A survey of preamble sampling MAC protocols. Comput. NETWORKS. (2011).

[60] Kredo, K., Mohapatra, P.: Medium access control in wireless sensor networks. Comput. Networks. 51, 961–994 (2007).

[61] Srisathapornphat, C., Shen, C.: Coordinated Energy Conservation for Ad hoc Networks. Communications. (2002).

[62] Im, C., Kim, H.: Dynamic Voltage Scheduling Technique for Low-Power. Electr. Eng. 1–6 (2001).

[63] Rabaey, J.M., Ammer, J., Karalar, T., Otis, B., Sheets, M., Tuan, T.: PicoRadios for wireless sensor networks: the next challenge in ultra-low power design. 2002 IEEE Int. Solid-State Circuits Conf. Dig. Tech. Pap. (Cat. No.02CH37315). 1, 200–201 (2002).

[64] Haenggi, M.: Twelve reasons not to route over many short hops. IEEE 60th Vehicular Technology Conference 2004 VTC2004Fall (2004).

[65] Haenggi, M., Puccinelli, D.: Routing in ad hoc networks: a case for long hops. IEEE Commun. Mag. 43, 93–101 (2005).

[66] Zhong, L.C., Rabaey, J.M., Wolisz, A.: Does proper coding make single hop wireless sensor networks reality: The power consumption perspective. In: IEEE Wireless Communications and Networking Conference, WCNC. pp. 664–669 (2005).

[67] Rault, T., Bouabdallah, A., Challal, Y.: Energy efficiency in wireless sensor networks: A top-down survey. Comput. NETWORKS. 67, 104–122 (2014).

[68] Anastasi, G., Conti, M., Di Francesco, M., Passarella, A.: Energy conservation in wireless sensor networks: A survey. Ad Hoc Networks. 7, 537–568 (2009).

[69] Ba, H., Demirkol, I., Heinzelman, W.: Passive wake-up radios: From devices to applications. Ad Hoc Networks. 11, 2605–2621 (2013).

[70] Misra, S., Pavan Kumar, M., Obaidat, M.S.: Connectivity preserving localized coverage algorithm for area monitoring using wireless sensor networks. Comput. Commun. 34, 1484–1496 (2011).

[71] Krishnamachari, B., Estrin, D., Wicker, S.: The Impact of Data Aggregation in Wireless Sensor Networks. In Proc: 22nd International Conference on Distributed Computing Systems, p.575-578 (2002).

[72] Lee, M., Wong, V.W.S.: An energy-aware spanning tree algorithm for data aggregation in wireless sensor networks. PACRIM. 2005 IEEE Pacific Rim Conf. Commun. Comput. signal Process. 2005. 0–3 (2005).

[73] Rajagopalan, R., Varshney, P.K.: Data-aggregation techniques in sensor networks: A survey, (2006).

[74] Lindsey, S., Raghavendra, C.S.: PEGASIS: Power-efficient gathering in sensor information systems. In: IEEE Aerospace Conference Proceedings. pp. 1125–1130 (2002).

[75] Dasgupta, K., Kalpakis, K., Namjoshi, P.: An efficient clustering-based heuristic for data gathering and aggregation in sensor networks. In: IEEE Wireless Communications and Networking Conference, WCNC. pp. 1948–1953 (2003).

[76] Heinzelman, W.B., Chandrakasan, A.P., Balakrishnan, H.: An application-specific protocol architecture for wireless microsensor networks. IEEE Trans. Wirel. Commun. 1, 660–670 (2002).

[77] Younis O., F.S.: HEED: a hybrid, energy-efficient, distributed clustering approach for ad hoc sensor networks. IEEE Trans. Mob. Comput. 3, 366–379 (2004).

[78] Manjeshwar, a., Agrawal, D.P.: TEEN: a routing protocol for enhanced efficiency in wireless sensor networks. In: Proceedings 15th International Parallel and Distributed Processing Symposium. IPDPS 2001. pp. 2009–2015 (2001).

[79] Kumar, D., Aseri, T.C., Patel, R.B.: EECDA: Energy Efficient Clustering and Data Aggregation Protocol for Heterogeneous Wireless Sensor Networks. Communications. VI, 113–124 (2011).

[80] Smaragdakis, G., Matta, I., Bestavros, a.: SEP: A stable election protocol for clustered heterogeneous wireless sensor networks. Second Int. Work. Sens. Actor Netw. Protoc. Appl. (SANPA 2004). 1–11 (2004).

[81] Kumar, D., Aseri, T.C., Patel, R.B.: EEHC: Energy efficient heterogeneous clustered scheme for wireless sensor networks. Comput. Commun. 32, 662–667 (2009).

[82] Li, H., Liu, Y., Chen, W., Jia, W., Li, B., Xiong, J.: COCA: Constructing optimal clustering architecture to maximize sensor network lifetime. Comput. Commun. 36, 256–268 (2013).

[83] Liu, A., Ren, J., Li, X., Chen, Z., Shen, X.S.: Design principles and improvement of cost function based energy aware routing algorithms for wireless sensor networks. Comput. Networks. 56, 1951–1967 (2012).

[84] Lu, Y.M., Wong, V.W.S.: An energy-efficient multipath routing protocol for wireless sensor networks. Int. J. Commun. Syst. 20, 747–766 (2007).

[85] Wang, Z.W.Z., Bulut, E., Szymanski, B.K.: Energy Efficient Collision Aware Multipath Routing for Wireless Sensor Networks. 2009 IEEE Int. Conf. Commun. 1–5 (2009).

[86] Younis, M., Akkaya, K.: Strategies and techniques for node placement in wireless sensor networks: A survey. Ad Hoc Networks. 6, 621–655 (2008).

[87] Ergen, S.C., Varaiya, P.: Optimal placement of relay nodes for energy efficiency in sensor networks. IEEE Int. Conf. Commun. 8, 3473–3479 (2006).

[88] Misra, S., Majd, N., Huang, H.: Approximation Algorithms for Constrained Relay Node Placement in Energy Harvesting Wireless Sensor Networks. Comput. IEEE Trans. PP, 1 (2013).

[89] Bi, Y.: Moving Schemes for Mobile Sinks in Wireless Sensor Networks. 101–108 (2007).

[90] Di Francesco, M., Das, S.K., Anastasi, G.: Data Collection in Wireless Sensor Networks with Mobile Elements: A Survey. ACM Trans. Sen. Netw. 8, 7:1–7:31 (2011).

[91] Sugihara, R., Gupta, R.K.: Optimizing energy-latency trade-off in sensor networks with controlled mobility. Proc. - IEEE INFOCOM. 2566–2570 (2009).

[92] Liang, W., Luo, J., Xu, X.: Prolonging network lifetime via A controlled mobile sink in wireless sensor networks. GLOBECOM - IEEE Glob. Telecommun. Conf. (2010).

[93] Luo, J., Hubaux, J.-P.: Joint mobility and routing for lifetime elongation in wireless sensor networks. IEEE 24th Annu. Jt. Conf. IEEE Comput. Commun. Soc. 3, 1735–1746 (2005).

[94] Vincze, Z., Vass, D., Vida, R., Vidács, A., Telcs, A.: Adaptive sink mobility in event-driven multi-hop wireless sensor networks. Proc. first Int. Conf. Integr. internet ad hoc Sens. networks. 13 (2006).

[95] Kansal, a., Rahimi, M., Estrin, D., Kaiser, W.J., Pottie, G.J., Srivastava, M.B.: Controlled mobility for sustainable wireless sensor networks. 2004 First Annu. IEEE Commun. Soc. Conf. Sens. Ad Hoc Commun. Networks, 2004. IEEE SECON 2004. 00, 0–5 (2004).

[96] Chakrabarti, A., Sabharwal, A., Aazhang, B.: Communication power optimization in a sensor network with a path-constrained mobile observer. ACM Trans. Sens. Networks. 2, 297–324 (2006).

[97] Bozdag, D., Ekici, E., Ozguner, F.: Partitioning based mobile element scheduling in wireless sensor networks. 2005 Second Annu. IEEE Commun. Soc. Conf. Sens. Ad Hoc Commun. Networks, 2005. IEEE SECON 2005. 00, 386–395 (2005).

[98] Somasundara, A.A., Ramamoorthy, A., Srivastava, M.B.: Mobile element scheduling for efficient data collection in wireless sensor networks with dynamic deadlines. Proc. - Real-Time Syst. Symp. 296–305 (2004).

[99] Brown, S., Sreenan, C.J.: Software update recovery for wireless sensor networks. In: Sensor Applications, Experimentation, and Logistics. pp. 107–125. Springer (2009).

[100] Leligou, H.C., Massouros, C., Tsampasis, E., Zahariadis, T., Bargiotas, D.: Reprogramming wireless sensor nodes. 1–8 (2011).

[101] Levis, P., Patel, N., Culler, D., Shenker, S.: Trickle: A Self-Regulating Algorithm for Code Propagation and Maintenance in Wireless Sensor Networks.

Proc. First USENIX/ACM Symp. Networked Syst. Des. Implement. 15–28 (2004).

[102] Kulkarni, S.S., Wang, L.W.L.: MNP: Multihop Network Reprogramming Service for Sensor Networks. Proc. 25th IEEE Int. Conf. Distrib. Comput. Syst. (ICDCS '05). 7–16 (2005).

[103] Dong, W., Liu, Y., Wu, X., Gu, L., Chen, C.: Elon: Enabling Efficient and Long-Term Reprogramming for Wireless Sensor Networks. *Proc. SIGMETRICS*. (2010).

[104] Gnawali, O., Fonseca, R., Jamieson, K., Moss, D., Levis, P.: Collection tree protocol. Proc. 7th ACM Conf. Embed. Networked Sens. Syst. - SenSys '09. 1 (2009).

[105] Maróti, M., Kusy, B., Simon, G., Lédeczi, Á.: The flooding time synchronization protocol. SenSys'04 - Proc. Second Int. Conf. Embed. Networked Sens. Syst. 39–49 (2004).

[106] Razaque, A., Elleithy, K.M.: Energy-efficient boarder node medium access control protocol for wireless sensor networks. Sensors (Basel). 14, 5074–5117 (2014).

[107] Tseng, Y.C., Ni, S.Y., Chen, Y.S., Sheu, J.P.: The broadcast storm problem in a mobile ad hoc network. Wirel. Networks. 8, 153–167 (2002).

[108] Wang, Q., Zhu, Y., Cheng, L.: Reprogramming Wireless Sensor Networks: Challenges and Approaches. 48–55 (2006).

[109] Hui, J.W., Culler, D.: The dynamic behavior of a data dissemination protocol for network programming at scale. SenSys '04 Proc. 2nd Int. Conf. Embed. networked Sens. Syst. 81–94 (2004).

[110] Hase, I.N.Y.P.R.P.: Mote In-Network Programming User Reference. 1–8. http://www.tinyos.net/tinyos-1.x/doc/Xnp.pdf, (2003).

[111] Miyamaru, T., Mineno, H., Terashima, Y.: State-Based Pipelining for Reprogramming. 531–538 (2007).

[112] Rossi, M., Bui, N., Zanca, G., Stabellini, L., Crepaldi, R., Zorzi, M.: SYNAPSE++: Code dissemination in wireless sensor networks using fountain codes. IEEE Trans. Mob. Comput. 9, 1749–1765 (2010).

[113] Doddavenkatappa, M., Chan, M.C., Leong, B.: Splash: Fast Data Dissemination with Constructive Interference in Wireless Sensor Networks. Proc. 10th USENIX Symp. Networked Syst. Des. Implement. (NSDI 13). 269–282 (2013).

[114] Ferrari, F., Zimmerling, M., Thiele, L., Saukh, O.: Efficient network flooding and time synchronization with Glossy. Proc. 10th ACM/IEEE Int. Conf. Inf. Process. Sens. Networks. 73–84 (2011).

[115] Raman, B.: TDMA-based MAC for High Throughput Bulk Transfer. Proc. 8th ACM Conf. Embed. Networked Sens. Syst. 15–28 (2010).

[116] Kulkarni, S.S., Arumugam, M.: Infuse: A TDMA Based Data Dissemination Protocol for Sensor Networks. Int. J. Distrib. Sens. Networks. 2, 55–78 (2004).

[117] De, P., Liu, Y., Das, S.K.: Energy-efficient reprogramming of a Swarm of mobile sensors. IEEE Trans. Mob. Comput. 9, 703–718 (2010).

[118] Krasniewski, M.D., Panta, R.K., Bagchi, S., Yang, C.-L., Chappell, W.J.: Energy-efficient on-demand reprogramming of large-scale sensor networks. ACM Trans. Sen. Netw. 4, 1–38 (2008).

[119] Stathopoulos, T., Heidemann, J., Estrin, D.: A Remote Code Update Mechanism for Wireless Sensor Networks. [Technical Report]. (2003).

[120] Panta, R.K., Khalil, I., Bagchi, S.: Stream: Low overhead wireless reprogramming for sensor networks. Proc. - IEEE INFOCOM. 928–936 (2007).

[121] Panta, R.K., Bagchi, S.: Mitigating the effects of software component shifts for incremental reprogramming of wireless sensor networks. IEEE Trans. Parallel Distrib. Syst. 23, 1882–1894 (2012).

[122] Panta, R.K., Bagchi, S., Midkiff, S.P.: Zephyr: Efficient incremental reprogramming of sensor nodes using function call indirections and difference computation. Usenix Atc. 32–32 (2009).

[123] Culler, D.: Incremental network programming for wireless sensors. 2004 First Annu. IEEE Commun. Soc. Conf. Sens. Ad Hoc Commun. Networks, 2004. IEEE SECON 2004. 00, 25–33 (2004).

[124] Pásztor, B., Mottola, L., Mascolo, C., Picco, G. Pietro, Ellwood, S., Macdonald, D.: Selective reprogramming of mobile sensor networks through social community detection. In: Wireless Sensor Networks. pp. 178–193. Springer (2010).

[125] Xiao, Z., Sarikaya, B.: Code dissemination in sensor networks with MDeluge. 2006 3rd Annu. IEEE Commun. Soc. Sens. Adhoc Commun. Networks, Secon 2006. 2, 661–666 (2007).

[126] Li, W., Zhang, Y., Childers, B.: MCP: An energy-efficient code distribution protocol for multi-application WSNs. Lect. Notes Comput. Sci. (including Subser.

Lect. Notes Artif. Intell. Lect. Notes Bioinformatics). 5516 LNCS, 259–272 (2009).

[127] Tomar, G.S., Verma, S.: Dynamic Multi-Level Hierarchal Clustering Approach for Wireless Sensor Networks, (2009).

[128] OMNeT++ Home Page. http://www.omnetpp.org