UNIVERSIDADE DE LISBOA
FACULDADE DE CIÊNCIAS
DEPARTMENTO DE INFORMÁTICA

EVALUATION OF WSN TECHNOLOGY FOR DEPENDABLE MONITORING IN WATER ENVIRONMENTS

Mestrado em Engenharia Informática
Especialização em Arquitetura, Sistemas e Redes de Computadores

Carlos Alexandre Aleixo Nascimento

Dissertação orientada por:
Prof. Doutor António Casimiro Ferreira da Costa
Prof. Doutor Pedro Miguel Frazão Fernandes Ferreira

2019

Agradecimentos

Este trabalho foi apoiado pela Fundação para a Ciência e a Tecnologia (FCT) através da unidade de investigação LASIGE (UID/CEC/00408/2019) e do projeto AQUAMON (PTDC/CCI-COM/30142/2017).

Resumo

Há vários problemas que surgem quando se monitoriza uma área geográfica recorrendo a uma rede de sensores sem fios sem fiabilidade para a leitura de dados e a disseminação dos mesmos. Estes problemas manifestam-se sob forma de qualidade da transmissão, medições com valores errados ou a alteração do valor inicial durante a transmissão ou até a omissão de um valor esperado, podendo comprometer o sistema. A motivação por detrás da criação da metodologia proposta nesta tese é a construção e instalação de uma rede de sensores de fios para monitorização aquática. Como tal, também é necessário proceder à escolha de uma tecnologia sem fios que consiga satisfazer os requisitos dessa mesma rede. Tal rede incorposa-se num projecto chamado Aquamon.

O Aquamon é um projecto que surge como iniciativa de reposta aos problemas anteriormente mencionados no que toca à monitorização confiável usando redes sem fios. No contexto deste trabalho, o principal desafio é a escolha de uma tecnologia sem fios no meio aquático. A imprevisibilidade da ondulação a somar ao movimento de maré amplificam as faltas sentidas na comunicação dos dados. A rede que será instalada situar-se-á na baía do Seixal, na margem sul do rio Tejo. Os objectivos do projecto passam pela instalação de uma rede de sensores sem fios, a recolha de amostras da qualidade da água e o estudo do caudal e restantes movimentos aquáticos causados pelas marés.

Quanto às tecnologias sem fios usadas em redes de sensores sem fios, as limitações normalmente encontram-se na Qualidade de Serviço e na disponibilidade que fornecem. Estas limitações normalmente encorrem de vários requisitos aplicacionais e/ou funcionais que têm de ser satisfeitos para garantir a operacionalidade correcta da rede. Durante a escolha da tecnologia sem fios a ser usada na rede é imperativo ter estes requisitos em conta. Este processo de escolha da tecnologia é normalmente feito de forma não metódica, manualmente e sem padrões definidos. É, portanto, um processo feito de maneira *ad-hoc*, com base no conhecimento (ou nas preferências) do designer da rede onde os prós e contras de cada tecnologia são comparados e revistos com os requisitos aplicacionais e funcionais da rede em mente. Mesmo em redes simples e de pequena escala há uma enorme sobreposição de requisitos, quer sejam funcionais ou não. A enorme complexidade que advém da conciliação entre requisitos, as diversas tecnologias sem fios e todas as possíveis soluções resultantes da combinação dos dois primeiro factores faz com que a tarefa de encontrar uma solução óptima seja difícil, se não impossível. Acrescenta-se

que nem sempre é fácil perceber o que realmente constitui uma solução óptima. Todos estes problemas no processo de escolha de uma tecnologia sem fios até agora referidos serviram de motivação para esta tese.

Uma nova categoria de tecnologias sem fios foi criada, de nome LPWA (Low Power Wide Area) e nela se encontram notoriamente três tecnologias: NB-IoT, Sigfox e LoRaWAN. Estas novas tecnologias foram criadas para serem usadas em aplicações IoT e portanto tomam relevância no contexto deste trabalho. Visto que são três tecnologias recentes e o seus graus de maturidade são baixos, um dos objectivos da tese é o estudo destas três tecnologias. Serão estudadas de forma teórica, não havendo experimentação práctica com as três tecnologias. As tecnologias serão estudadas em relação à sua autonomia, custo (tanto operacional como de aquisição), arquitectura, cobertura geográfica e qualquer outra propriedade de relevo. Não só será feita uma comparação entre as três mas também com uma tecnologia mais madura, o ZigBee.

O trabalho feito para concretizar o primeiro objectivo da tese, tal como a compreensão dos problemas inerentes que advêm da instalação de uma rede de sensores sem fios, servem como fundação para o objectivo final da tese. O segundo objectivo é conseguir criar uma metodologia que contraria a subjectividade presente na escolha de uma solução óptima para a rede de sensores sem fios. Com isto quer-se a criação de um processo standard, automatizado que remove factores externos não técnicos do processo de decisão.

Antes de criar a metodologia estudou-se as tecnologias LPWA que poderão ser alternativas a qualquer rede de sensores sem fios, no entanto, a metodologia também tem de ter em conta os requisitos da rede. É precisamente por isto que foi feito um levantamento dos requisitos mais frequentemente usados em redes de sensor sem fios e as suas respectivas definições. Depois de ter sido tomado conhecimento sobre os detalhes técnicos das tecnologias, os problemas inerentes a rede de sensores sem fios e os requisitos influenciam a rede pode-se começar a criar a metodologia.

A metodologia tem em conta os vários requisitos aplicacionais e funcionais, as características técnicas das várias tecnologias consideradas (NB-IoT, Sigfox, LoRaWAN e ZigBee) e retorna um espaço de solução que melhor satisfaz todos os requisitos. Não só oferece uma metodologia que atinge objectivamente o melhor conjunto de soluções óptimas, como também foi pensada de maneira a automatizar todo o processo de escolha minimizando ao máximo o papel do designer da rede.

Uma das inovações desta metodologia é a utilização de um grafo como representação da rede física. Os vértices representam nós da rede e arestas a ligação física entre dois nós. O designer da rede estabelece os requisitos para cada vértices. No entanto os requisitos das arestas são impostos pelos requisitos dos vértices. A metodologia está dividida em 4 passos: estabelecimento dos vértices, ligação dos vértices, sobreposição de grafos e finalmente a definição do espaço de solução. Os três primeiros passos servem para a

construção do grafo final que contém toda a informação acerca dos requisitos e tecnologias consideradas. Esta informação é depois extraída e é gerado um espaço de solução. Este espaço é por norma constituído por um número massivo de soluções que cresce exponencialmente com o aumento linear de nós na rede. O designer da rede define critérios de avaliação, estes critérios dão valor a cada uma das soluções. A definição destes critério servem também para não só dar poder de decisão ao designer da rede mas também para lhe ser possível definir aspectos que são relevantes para aquele caso de uso em específico. Nem todas as soluções são óptimas, para filtrar o espaço de solução e conseguir obter um sub-espaço de soluções óptimas, as soluções são passadas por um algoritmo da Parede de Pareto. Após esta filtragem, o designer da rede pode finalmente escolher a sua solução.

Foram escolhidos dois parâmetros de avaliação da metodologia: eficácia e performance. A eficacia deve-se à capacidade da metodologia conseguir reduzir suficientemente o espaço de solução de modo a que o designer da rede consiga analisar as soluções óptimas e escolher uma. A performance está relacionada com o tempo em que a metodologia consegue produzir resultados.

Estes dois parâmetros foram avaliados de diferentes maneiras. Para ambos foram usados os mesmo casos de uso: 4 casos de uso genéricos e o caso de uso do Aquamon. Desta maneira conseguimos o contraste entre casos de uso genéricos e um caso de uso que será realmente instalado no terreno de forma operacional. A percentagem de redução do espaço de solução gerado para o sub-espaço de soluções óptimas (após o algoritmo de Pareto) foi a medida usada para avaliar a eficácia. Quanto à performance, a ferramenta que implementa a metodologia executou dez vezes sobre cada caso de uso. A média aritmética do tempo de execução (em segundos) da ferramenta foi usado para medir a performance.

Finalmente são dadas listadas algumas sugestões de melhoramento da metodologia como também algumas deficiências da metodologia que poderão motivar trabalho futuro.

Palavras-chave: lwpa, methodology, comparative study

Abstract

Few problems arise when trying to reliably monitor a surrounding environment by the use of sensors and a wireless network to disseminate the information gathered. In the context of an aquatic environment, the undulation and the low predictability of the surrounding environment could cause faults in the transmission of data. The initial motivation for the work developed in this thesis was the Aquamon project.

Aquamon is a project that has as objective the deployment of a dependable Wireless Sensor Network (WSN) for the purposes of water quality monitoring and the study of tidal movements. Therefore, Aquamon, like any other WSN will have to go through the process of choosing a technology that meets its application requirements as well as the requirements imposed by the deployment environment.

WSNs can have constraints when it comes to the Quality of Service and availability it can provide. These networks generally have a set requirements that need to be satisfied. Thus, there needs to be a selection of one (or multiple) wireless technologies that can satisfy said requirements. This selection process is usually done in a ad-hoc way, weighting the advantages and disadvantages of different possible solutions with respect to some requirements, often using empirical knowledge or simply dictated by the designer's preference for some particular technology. When several functional and non-functional requirements have to be addressed, finding an optimal or close to optimal solution may become a hard problem. This thesis proposes a methodology for addressing this optimisation problem in an automated way. It considers various application requirements and the characteristics of the available technologies (including Sigfox, LoRa, NB-IoT, ZigBee and ZigBee Pro) and delivers the solution that better satisfies the requirements. It illustrates how the methodology is applied to a specific use case of WSN-based environmental monitoring in the Seixal Bay.

Keywords: lwpa, methodology, comparative study

Contents

List of Figures

List of Tables

Chapter 1

Introduction

With the introduction of LPWA (Low Power Wide Area) technologies the available scope of wireless technologies for the deployment of WSN (Wireless Sensor Network) increased. There are now diverse technologies to choose from which differ at various technical levels. Where some technologies excel, others have limitations. This is because different technologies were created to be used in different situations and to address different application requirements.

Therefore, there is no single technology that is appropriate to address the outstanding requirements of every possible use case. Moreover, selecting one or a set of technologies that will appropriately address the application requirements is a hard problem. This is not only because the amount of possible options tends to be large, namely when considering possible combinations of technologies to serve in different parts of a WSN, but also because there are situations in which it is not obvious which solution is optimal, and finally because there are human aspects that often influence the selection process based only on empirical knowledge that is sometimes not substantiated. For instance, managerial decisions or personal bias can influence the selection process leading to either sub-optimal or downright wrong selection choices. In this thesis we propose a solution to address this problem. More precisely, we introduce a methodology to automate the process of selecting one or multiple LPWA technologies while satisfying a set of application requirements. By automating the process, we not only facilitate the selection process but also provide a way to reach optimal or close to optimal solutions, removing subjectivity, personal bias, and possible assessment mistakes from the technology selection process.

Our proposed methodology consists of four steps. They go from the gathering of information regarding the application requirements (e.g., concerning functional aspects like throughput or communication distance as well as non-functional aspects like cost or network homogeneity) to the provision of a set of possible solutions for the problem, including two steps whose purpose is to eliminate solutions that do not meet requirements. We argue that this solution hits the sweet spot in solving the problem of finding the right solutions for the network technologies to be used, which satisfy the application require-

ments and, whenever possible, still allow some degree of flexibility in the final choice.

1.1 Motivation

The initial motivation to start the working on the proposed methodology came from the need to design and choose a technology to be used in a WSN for the Aquamon project. The Aquamon project has as objective the deployment of a WSN that is reliable and satisfies real-time constraints. Thus, a study on the available wireless technologies is required. There are many wireless technologies suitable for WSNs, they differ in many technical aspects such maturity, energy consumption, data throughput, area coverage and perhaps most importantly, optimal use case. The first choice for the technology to be used in the deployment of Aquamon's WSN was ZigBee. Although ZigBee is a mature technology, a new category of wireless technologies appeared. These technologies are denominated LPWA (Low Power Wide Area) and were designed for the deployment of large IoT applications.

These new technologies still suffer from some problems that stem from the lack of reliable delivery of the data, insufficient coverage and throughput, in some cases the lack of flexibility in the deployment of such networks. In addition, the necessity of balancing energy consumption with the amount of transmission time cannot be ignored. Consequently, a situation is created where the selection of the technology is in the hands of the network designer, making her responsibility to find the optimal solution (technology) that satisfies all the requirements. Reaching an optimal solution is not an easy task and there is no common practices that are employed. Moreover, it is not always clear what is the best definition of an optimal solution.

Not only there are technical aspects that constrain the choosing process, there is also the human factor. Human bias, lack of knowledge and managerial decisions remove objectivity from the final solution. Unfortunately, there aren't many answers to this problem.

In essence, there is not a concrete pattern, methodology or automated process that is standard when it comes to picking a wireless technology for a WSN.

1.2 Objectives

The thesis has two main objectives. The first one is the study of the LPWA technologies such as NB-IoT, Sigfox and LoRaWAN and how they compare to not only each other but also ZigBee.

This background work, together with the understanding of the inherent problems of deploying WSNs, serves as foundation for the main objective of the thesis. The second objective is to bring forth a solution that aids in the process of choosing a technology (or more) to be used in a WSN. The objective is to provide a solution that establishes a

standard automated process which removes any external human biases and provides the most objective solution space according to the requirements of the WSN.

1.3 Structure

Firstly, in Section 2.1, an overview of the several related works that have been done in regards to the monitoring of water bodies and the deployment of WSN in different use cases. Following that, Section 2.2 gives an introduction and a detailed discussion on the LPWA technologies. Section 2.3 describes how ZigBee differs from LPWA technologies and provides a foundation to be used in Section 2.4 where LPWA technologies and ZigBee are compared. Since the scope of this work is the deployment of WSNs, Section 2.5 lists several requirements that are used to define any WSN.

Chapter 3.3 describes the solution found for the problem of choosing a wireless technology in the deployment of a WSN. It starts by giving a problem definition followed by a methodology overview. Afterwards it focuses in explaining the methodology. Chapter 4 gives insight to the implementation of the tool and the various algorithms and procedures that constitute the methodology.

The evaluation of the methodology is done in Chapter 5. First it presents the evaluation objectives and how the evaluation will take place. The analysis of the results is done in Section 5.3.

Finally, Chapter 6 lists several suggestions to improve the methodology as well as relevant issues that be built upon in future work.

Each chapter concludes with a small summary of what was discussed which also serves as a segway for the next chapter.

Chapter 2

Background work

Firstly, a survey of the past and current work that is being done on the subject of deployment of WSNs is shown in Section 2.1. The content of the featured works showcase the use of new LPWA technologies in several use cases, but also a demonstration of several methodologies that aid the process of the deployment of WSN.

This chapter contains a survey and a review of the most recent category in wireless technologies and which technologies are part of it. This paradigm of wireless technologies is called LPWA and its most notable technologies are NB-IoT, Sigfox and LoRaWAN. Not only introduces new concepts like LPWAs (Section 2.2) but also a more mature technology like ZigBee (Section 2.3). After a detailed review of both LPWA technologies and ZigBee, Section 2.4 compares the two on a technical scale with the aide of Table 2.1. Moreover, some of the comparison points present in the table are discussed, providing more insight and detailed information to complement what is presented on the table.

Section 2.5 lays forth relevant background knowledge to be applied in Chapter 3.3 in regards to wireless communication and application requirements.

2.1 State of the art in WSN deployment and LPWA studies

Recent work on real-time monitoring of water bodies has been able to not only show practical results of the use of LPWAs and ZigBee but also to help general population and authorities. LoRa was used in India where a system was deployed to monitor water quality and to alert the authorities in real-time whenever the water quality dropped below a certain level. They improved water quality, established alert systems to help the population and farmers.[3]

In Malaysia, ZigBee was used to monitor water quality. A monitoring software with GUI was developed to be able to interact with the base station. Thanks to the reliability of the ZigBee communication, it was deemed a satisfactory monitor system.[4]

Not only there is practical work but theoretical as well. The interoperability between

systems is compromised by the lack of standards and proper models. To tackle this issue, in Spain, a detailed architecture and modelling system were created to help define the integration of IoT concepts in water monitoring systems. A simple water management system was made to show off the model in practice.[5]

Mac et al[6] identify two different application use cases and a taxonomy is established for the network requirements. Some of these requirements were also considered in the methodology presented in this thesis.

Díaz and coworkers [7] describe a seven step methodology that goes from requirement gathering to deployment and maintenance of a WSN for agricultural applications. While the methodology includes several steps, similarly to the methodology presented in this thesis, these steps correspond to the life cycle of a WSN application and not specifically to the process of selecting an appropriate solution for the adopted technologies. Furthermore, the thesis is focused on a specific application area.

In [8] the authors try to solve the problem of selecting the best technology for a set of requirements in a WSN, presenting a two step methodology. Firstly, the available technologies are reduced to only those that satisfy the gathered requirements. Then, the result obtained in the first step is further refined by taking into account cost requirements. This work considers two use cases to practically demonstrate the methodology and how it helped to solve the deployment of two very different WSNs: one to monitor containers in a port and the other to manage parking lots. The methodology introduced in this thesis is more elaborate in the intermediate steps to prune inappropriate solutions.

2.2 LPWA Technologies

With the increased demand of connected devices to the Internet or to each other forming huge networks performing several tasks such as smart agriculture, smart metering, smart grids, smart city, the monitoring of environments a new problem arose: the lack of a technology (or a bundle of them) that could answer the needs that these applications demand. The technology needed to satisfy the requirements imposed by these use cases is one that allows for the possibility to employ an ubiquitous, scalable long range wireless network, reliable, with good quality of service. The throughput stops being a priority being replace by an efficient way of managing energy.[9]

Some technologies exist that answer to some of the above-mentioned requirements, some more than others. 4G/LTE are a group of technologies that seem to be able to be used for IoT applications, unfortunately they lack in several points that are deal breakers. They were planned for applications that require large amounts of data for long periods of time with little latency, this is not ideal to IoT applications where the pattern of communication is normally defined by sporadic exchanges of small amounts of data and latency is not a priority[10, 9]. The use of great amounts of energy (in the context of IoT) is also

a concern. This becomes a nuisance as it makes the upkeep of the devices extremely difficult, specially in rural areas. In contrast, LPWAs can have a battery life in the order of a decade[11].

Since the most suitable use cases for LPWA technologies require a large, scalable, energy efficient network, the cost of each device should be affordable in large quantities.

Under the LPWA umbrella three technologies take the spotlight: NB-IoT, Sigfox and LoRaWAN. Even though they are contained in the same category of wireless technologies, they still differ between each other and offer solutions to different use cases. More detailed information on these technologies will be given from Section 2.2.1 to Section 2.2.3.

2.2.1 NB-IoT

Narrowband IoT is a standard defined by 3GPP on Release 13 which runs on top of current GSM/LTE networks. It uses a low frequency licensed spectrum in the 700MHz-900MHz range. The reason as to why it was implemented was to create an alternative technology that had be thought out from the ground up to serve IoT applications, satisfying their requirements that GSM/LTE networks could not.

Its communication protocol is similar to that of LTE, in fact, NB-IoT is LTE with some tweaks (restricting some parts and extending others) to better mould it to what the necessities of IoT applications[10].

One of the first production test deployments of a NB-IoT is being done in Belgium[9].

Technical Details

Thanks to the nature of these low frequencies that NB-IoT uses to transmit its data, NB-IoT gains in terms range of coverage and indoor and deep-indoor penetration, technologies with higher frequencies such as WiFi have difficulties in both these capabilities.

NB-IoT uses a licensed spectrum[12]. The fact that the frequencies are licensed, frees them from the noise that is common in unlicensed frequencies, this improves greatly the quality of service[12].

There are three ways to deploy NB-IoT[10, 13]

- In-band operation: uses one or more resource blocks from the LTE carrier, this means that the base station is allocating resource both for LTE and NB-IoT. This sharing of resources does not impact the performance of either.

- Guard band operation: uses resource blocks from the LTE carrier that aren't being used for guard band. As a result the transmission power is greater since it shares the same amplifier power as the LTE channel.

- Stand alone operation: uses an isolated spectrum. The base station allocates all its resources to NB-IoT which results in a wider coverage. In the future this will

overtake GSM carriers.

NB-IoT's communication is synchronous which has an effect on it is battery life as it can not indefinitely sleep, on the other hand, it provides lower latency. Long duration communications are difficult because there is not an indication of signal deterioration.To mitigate this terminal devices adjust themselves to three coverage levels: 144dB, 154dB and 164dB. Bigger the dB, bigger the area covered.

Architecture

The architecture is divided in three main components. Firstly we have the NB-IoT modules that are coupled with IoT devices, these then connect to the base stations which then redirect the data to the cloud where they can be analysed and processed as seen fit.

Due to the fact that NB-IoT is a standard close to LTE and utilises the same infrastructure (base stations for example) it is then only natural that GSM,4G/LTE providers are the ones that will supply NB-IoT. Little work is necessary for a base station to be compliant with NB-IoT, it only needs an antenna capable of transmitting the required frequencies and in some cases a small firmware update.[13, 14, 10] It's up to the providers to manage the infrastructure.

Cost

NB-IoT modules are cheap. Another argument supporting why existent GSM providers will be the ones to provide NB-IoT is that the cost to transmit in licensed spectrum is vastly too great for someone that hasn't an infrastructure yet.

2.2.2 Sigfox

Sigfox is a proprietary technology created by a private company. Provides long range coverage and a small energetic footprint. Unlike NB-IoT it transmits over an unlicensed spectrum (868MHz in Europe), which makes Sigfox comply with duty cycle[15], therefore it is subject to noise and interference from other devices that may be transmitting and the restriction of time it may transmit.

Normally Sigfox partners with another company to manage the infrastructure. The client is only responsible for the terminal Sigfox modules, the base stations and the Sigfox Cloud are closed to them. This gives the client peace of mind and decoupling from the management of such big infrastructures but puts them in the hands of the Sigfox which may experience temporary hiccups in availability, for example, leaving the client unable to receive their data[16].

Technical Details

Although bidirectional communication is supported, the number of Downlink messages is restricted to subscription plans. Messages are 12 bytes long and the maximum of allowed messages is 140 a day, this number goes down depending on the subscription plan[15, 1]. To ensure reliability Sigfox uses a process called Random Access where 3 messages are sent in 3 different time periods which then are received by the Sigfox base stations, no acknowledgement is made by the base stations[1], this saves battery.

Architecture

Sigfox's infrastructure is closed, the client only has access to terminal devices, applications must use the Sigfox Cloud API to retrieve the data and to communicate with said terminal devices[10].

Its architecture is divided in three parts, in all of them the communication protocols are closed and the client has no power over it. Even so, the client can choose to use it is cryptographic algorithms instead of the default closed Sigfox cryptography[17]. Terminal devices connect to Sigfox Base Stations which before sending them to the Sigfox Cloud perform several operations on the data like deletion of redundant messages for example. To retrieve the data from the Sigfox Cloud the client needs to use the Sigfox API.

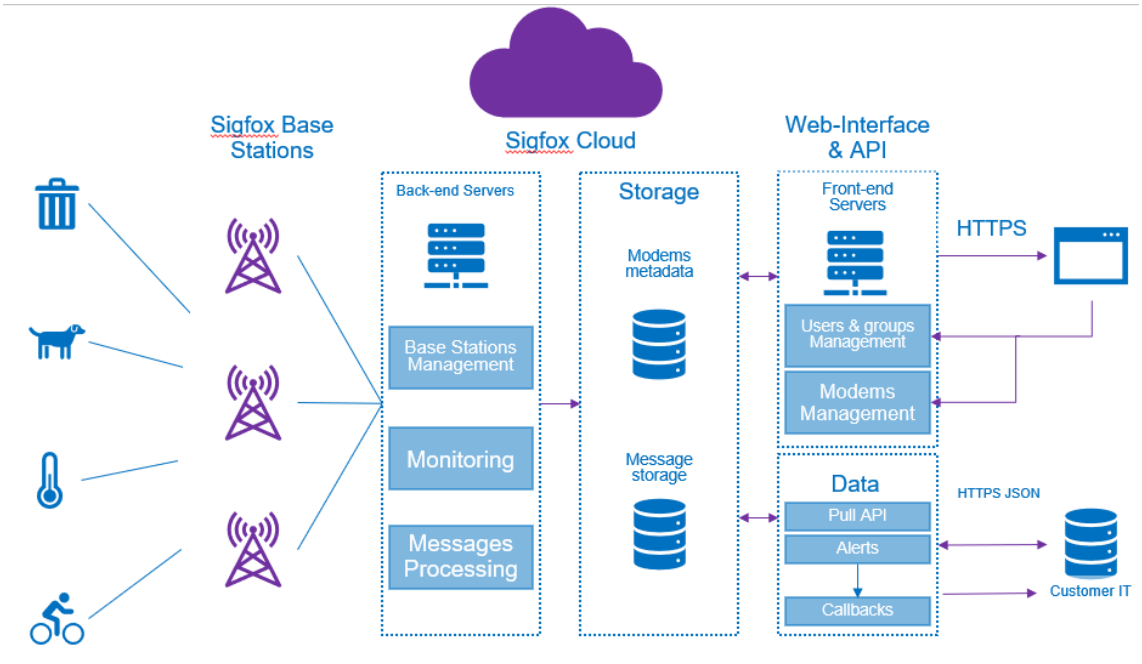


Figure 2.1: Overview of the Sigfox architecture.[1]

Cost

As referred already, the number of messages transmitted per day depends on the subscription plan chosen by the client. Different plans offer different message thresholds for Uplink and Downlink alike. On the other hand the client does not need to worry about the management of such big infrastructures.

Another thing that can not be neglected is the restrictive nature of the Sigfox API which makes the developing of an application using Sigfox very strict and not easily portable to any other technology. The Sigfox modules are cheap.

2.2.3 LoRaWAN

LoRaWAN is a LPWA communication standard which is maintained by LoRa Alliance, an open and non-profit association. It's backed by big industry names like Orange, Cisco, IBM, etc and already has large deployed networks in several countries, The Netherlands is one example. For this reason, it can be said that LoRaWAN has a maturity level higher than that of NB-IoT.

Like Sigfox, LoRa transmits over an unlicensed frequency therefore it is susceptible to the same shortfalls as Sigfox when it comes to the interference of other devices and the need to comply with the duty cycle[18]. Differently though, LoRaWAN as it is an open protocol (except the physical layer) it has no maximum number of message exchanges.

Technical Details

As before-mentioned the access to the medium layer is proprietary and closed but the rest of the protocol is open[2]. The LoRa gateway receives the data from the terminals via radio and converts them to IP packets which are then forwarded so the central server. The second part of the communication can be by Ethernet, 3G, WiFi for example, the reverse is done when it is a downlink communication[2, 16, 15].

LoRa modules are divided in three classes:[2, 18]

- Class A - Less Energy, Bidirectional: It's the default class. The communication is started at the terminal device and it is asynchronous. Thanks to that the device can sleep, saving energy. Despite this, the communication can still be bidirectional as there is a window where the gateway can answer after receiving a message.

- Class B - Bidirectional with deterministic downlink latency: In addiction to the downlink transmission windows, the communication is synchronous. There are predefined temporal instances where a heartbeat is sent which allows downlink transmissions deterministically at the expense of slight more energy consumption.

- Class C - Less latency, bidirectional: Provides reduced latency by maintaining the receiver awake when not transmitting. Therefore downlink connections can be done

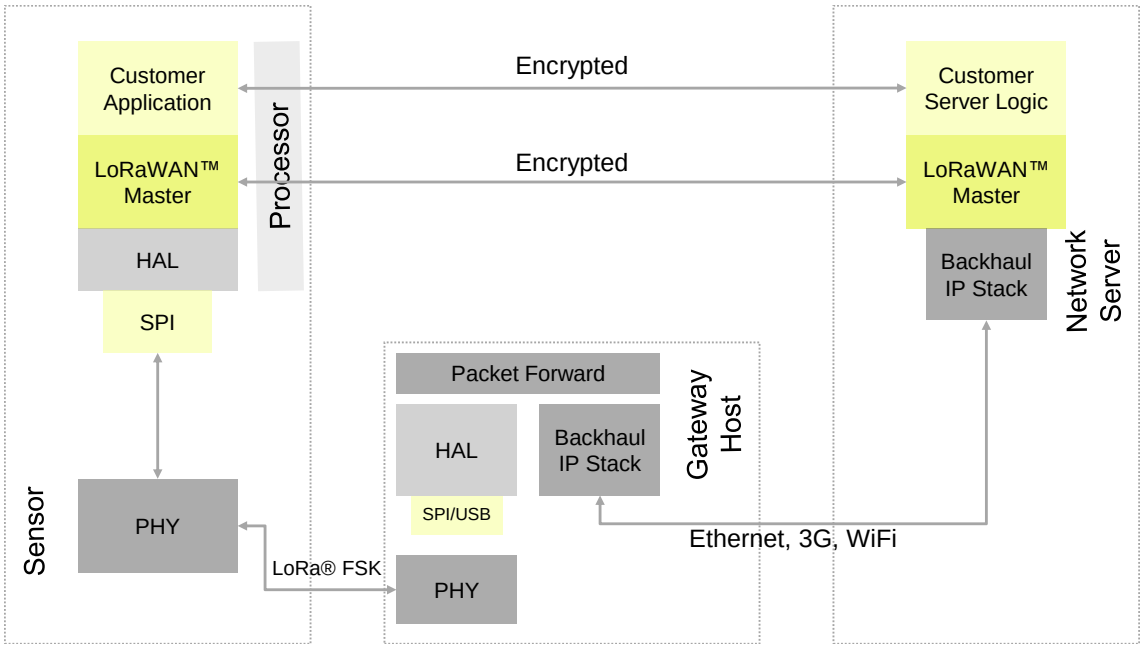


Figure 2.2: Overview of LoRaWAN's layers.[2]

with very low latency at the expense of a bigger energy consumption. Terminal devices can change between Class A and Class B as convenient.

Packet payload vary from 19 to 250 bytes with 12 bytes of overhead. LoRa's coverage is conditioned by the link budget, as in, playing with several variables such as bandwidth, coding scheme, transmission power and carrier frequency we can change the effective range of the device[2].

Gateways establish individual data rates for each terminal device, compromising range for transmission time. Different data rates do not interfere with each other, that way gateways can manage and balance used energy with transmission capabilities[2].

Scalability in the IoT context is a valid concern. The way that LoRa addresses this is in complement to the adaptive data rate, LoRa gateways have a multi-channel transceiver that permits the reception of several messages at the same time even with if they come in different data rates. The gateway also treats each device efficiently, it increased the data rate of closer devices to free up space for distant devices to also transmit[2].

Architecture

In contrast with both technologies already introduced, LoRa does not have a provider, it is the client's job to design, implement and deploy their LoRaWAN infrastructure. In more detail, the way gateways talk to the central server or cloud service, the deployment of said server and the client application that retrieves the data are responsibility of the client.

LoRaWAN is divided in four parts: the terminal devices, the gateways, the central server and the client application. The LoRa modules on the terminal devices connect to

the one or several gateways that then forward the data to the central server/cloud[2]. Gateways do not communicate among each other but multi-hop of end devices is possible[19]. A terminal node serves as relay for another one, this relay is used to further the coverage of the network which would not have been possible otherwise since the terminal device would be out of the gateway's range. LoRa also supports broadcast messages.

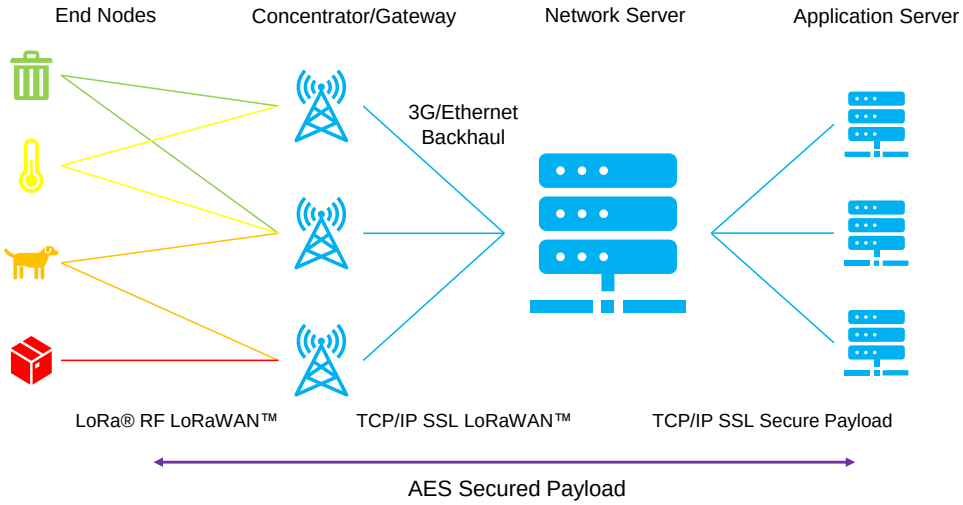


Figure 2.3: High level overview of the LoRaWAN architecture.[2]

Cost

LoRa modules are cheap and gateways are found at an accessible price. The real cost is the process of designing, implementing and deploying a network, adding to that we then have to account for the management and upkeep of the infrastructure. In a way, this unrivalled freedom and flexibility that LoRa provides can be seen as a double edged sword.

2.3 ZigBee and ZigBee PRO

LPWAs do not necessarily replace existing technologies. There are applications where the use of Bluetooth seems more plausible than LoRaWAN for example, in the same train of thought, ZigBee continues to have a niche group of use cases that makes it more apt than any other LPWA technology. Although this might be true for most cases, special applications that now use ZigBee might change to LPWAs, this is because at the time no other alternative existed which forced the use of ZigBee, even if not optimal.

The biggest advantage ZigBee has over LPWAs is that even though LPWAs are equipped with several different deployment methods, ZigBee allows for point to point communication where there is not a need for a central point in the network. This mesh like topology is not possible in LPWAs which invalidates them from application that require the use of such topology and the benefits that comes with it.

In terms of coverage ZigBee can not compete with the great distances LPWA cover, on the other hand, it can reach farther than WiFi and Bluetooth. On that note, ZigBee does not offer the same energy efficiency as LPWAs, but once again, excels against PAN and WAN technologies. Regarding data throughput, ZigBee surpasses both Sigfox and LoRa rivalling NB-IoT. ZigBee is also a largely more mature and tested technology comparing to LPWAs.

Two use cases where ZigBee is used are the following:

Smart parking In this use case we talk how ZigBee was used on a smart parking application. It is to be considered that at that time no LPWA existed, but even if there were, they would not have been used since this use case used multihop mesh topology. A repeater was put in place every 5-10 nodes and a gateway every 100-150 parking nodes. The gateways used a 3G modem or an Ethernet cable as a way to connect to the Internet[20].

Smart home We saw that one of the factors in favour of ZigBee comparing to any other LPWA techonology is that it is possible to deploy a mesh topology network, despite this, a star topology was used[21].

This was smart home application therefore we are to expect lots of devices throughout the home/building communicating via ZigBee and some complementary RFID. The latency and data rates are important in this context as it is crucial for the usability of the application without frustrating the people that inhabit the room/building. These latency and throughput requirements and also the constant need to send and receive data would invalidate Sigfox and LoRa. NB-IoT would also be undesirable since its architecture would over-complicate the application; data would be sent to the base station and then accessed through the cloud.

2.4 Comparison between the different technologies

After a brief introduction to these three LPWA technologies, a comparison between them will take place regarding some aspects that are relevant to IoT applications.

Table 2.1 gives an overview of the three technologies, then, a more detailed explanation will follow.

Table 2.1: Comparison between LPWAN technologies.

	NB-IoT	LoRaWAN	Sigfox	ZigBee	ZigBee PRO
Frequency	700-900 MHz	868 MHz	868 MHz	2.4 GHz 915 MHz 868 MHz[22]	2.4 GHz 915 MHz 868 MHz[23]
Licensed	Yes	No	No	No	No
Data rate	190-250 kbps[11]	250bps-50 kbps[2]	<100-600 bps[1]	250 kbps	250 kbps (at 2.4 GHz) 10 kbps (at 915 MHz) 100 kbps (at 868 MHz)[23]
Payload	1600 bytes[24]	19-250 bytes 12 bytes overhead[2]	12 bytes[1]	100 bytes (no security) 82 bytes (with security)[22]	variable: 1 octet with payload size[23]
Topology	Star	Star	Star	Star/Mesh/Tree[22]	Star/Mesh/Tree[23]
Coverage	1-8km (urban) 25km (suburban)[11] <35km[12]	2-5km (urban) 15km (suburban)[11] <15km[12]	3-10km (urban) 30-50km (suburban)[11] <15km[12]	100 m	average: <300 m (line of sight) 75-100 m (indoor) <1km for sub GHz channels[23]
Battery life	10 years[10]	>10 years[11]	8-10 years[11]	-	-
Base cost	-	<30€	70€(w/ 1 year subscription)	<30€	<45€
Subscription cost	Yes	No	Yes	No	No

Architecture

The three technologies are similar in terms of architecture. Neither of them offer a point to point communication, all the data is converged to a central point on the network, the cloud. In retrospect, there are three parts of the architecture that are common: the terminal devices, the cloud and the client application. Noting this, these will be the subjects of our comparison.

Out of the three the one that offers more liberty in deployment in LoRaWAN, not only because it permits having several gateways but also because multi-hop can also be achieved. The LoRa standard does not establish directives on how to process data on the cloud which grants more flexibility in deployment.

In contrast, Sigfox is the one that has the most closed environment. The client does not have any power where their data is stored and how it is sent. To retrieve its data, the client need to use the Sigfox API to interact with the Sigfox Cloud. Not only this but we are in the hand of Sigfox, if for whatever reason some part of the Sigfox infrastructure loses temporary availability there will not the data being transmitted and/or being retrieved.

When NB-IoT is compared with LoRaWAN, one of the conclusions is that NB-IoT is the one that grants a better Quality of Service. This is due to the transmission over licensed frequencies and a provider maintaining the infrastructure. As LoRaWAN varies synchrony model from NB-IoT (meaning it is asynchronous) and uses unlicensed frequencies, it is recommended that LoRaWAN be used in applications where quality of service, latency, data integrity and data throughput are not a big factor. If they are, NB-IoT should be the technology chosen.[12]

Throughput

Data throughput is not a priority for LPWAs hence the low data rates that all of them have.

As can be seen from Table 2.1, NB-IoT provides the most data rate of the three. Moreover, as it uses a licensed spectrum, it is not restricted by duty cycle meaning it can transmit at any time.

The other two technologies are bound to the unlicensed spectrum. Per Table 2.1, LoRaWAN has a huge discrepancy in data rate values, this is due to its channel disposition. LoRaWAN has ten channels, eight of them have speeds varying from 250bps to 5.5kbps, one has the speed of 11kbps and the last of 50kbps[2]. The disposition of these channels changes with the geographic region, the values presented are the European Union norm.

Sigfox markets its platform saying we can transmit 36 seconds of each hour due to the duty cycle. Subscription plans also limit the number of messages per day and consequently the quantity of data that is transmitted.

As is with NB-IoT, LoRa does not limit the number of messages to be sent having only to oblige to the duty cycle. Different classes of LoRa modules influence the quantity of data transmitted between the module and the gateway. As an example, a device could be running in Class A and change to Class C to be able to more quickly receive a Downlink firmware update[2].

Coverage

The way these three technologies provide coverage for their network varies from each other.

In NB-IoT's case since that it is a LTE version, where currently exists 3G/4G/LTE coverage there should also be NB-IoT coverage in the future. For the base station to support NB-IoT it only needs the installation of an antenna capable of transmitting on the desired frequencies (700 to 900MHz), in some cases, a small firmware update might be needed. So there is high hopes that NB-IoT gains global coverage.

In regards to LoRa it is up to the client to deploy the gateways in the areas he sees fit since it is the gateways that provide the coverage. This has an advantage over both Sigfox and NB-IoT in that the client is not limited to the only areas of coverage a third party deems relevant, leaving other areas with poor to none coverage.

Sigfox outsources the management of it is infrastructure to a partner, and so, the areas that will have coverage are responsibility of that partner which might cause limitations to the client. Figure 2.4 shows the current coverage of the Sigfox network in Portugal (mainland).

Cost

The cost for the terminal modules is similar in all three technologies. The real cost, technically speaking, varies with each technology.

Since Sigfox is a closed and proprietary platform, applications are bound to the Sigfox API raising the cost of migration to another platform. We can not dismiss the subscription plans which have to be bought to access the Sigfox platform, different plans allow and limit the client to a certain number of messages per day, Uplink and Downlink alike.

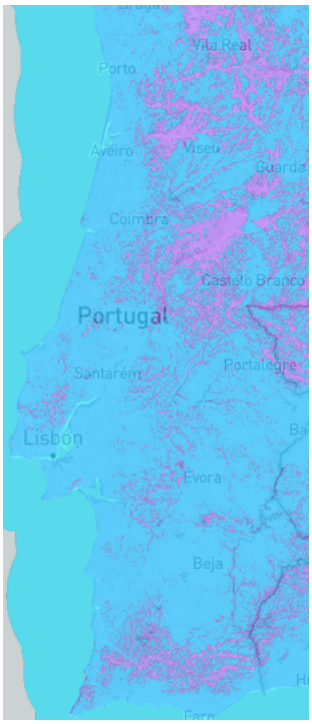


Figure 2.4: Coverage of the Sigfox network in Portugal (mainland).

If Sigfox's cost is due to its closed nature, LoRa's cost is precisely due to the opposite. As it was said already, the prices of the modules are low however the cost of operating a LoRa network is in the planning, implementing and managing all of its infrastructure. On a small scale this is achievable and sometimes wins over other technologies but as the network expands it rapidly becomes unfeasible without a great personal investment.

2.5 Application requirements

This section lists several application requirements and their definitions. These requirements were deemed the most relevant out of many more. The list of requirements was divided into two groups: the explicit requirements and the implicit requirements wherein there is a subgroup of requirements which come from a managerial stand point.

The reason for the separation of requirements in two is that some requirements differ in how they are defined. Explicit requirements, as the name entails, need to be specified by the network designer. They are attributes given to the network and not something inherent to it. Latency, data throughput, minimum packet loss are requirements that need to be specified and cannot be deduced by the network designer. Same goes for every requirements in this category.

In contrast, implicit requirements can be inferred from the network. They characterise the network by defining how big the network is, how many nodes, what the surrounding environment of the nodes is.

2.5.1 Explicit network requirements

The identified requirements featured in this section are availability, latency, packet loss and data throughput.

Availability

The network is composed by several types of components like end-nodes, gateways and central servers, all of which may fail in ways that compromise the ability to communicate. Availability requirements express the need for the network to be operational with some probability. Availability is expressed by the ratio of uptime with the total time of operation.

Latency

Communication latency refers to the time needed for a message to be sent from a source to a destination, usually expressed in time units (e.g., milliseconds). Defining the synchrony model for the application is the first important step. Applications can either be synchronous or asynchronous. The synchronous model can have real-time constraints which are classified differently: hard and soft. The guarantees provided by real-time systems are known deadlines (timely delivery of events and timely response to events) that need to be respected.

For hard real-time applications, latency is bounded and must be strictly satisfied. However, in soft real-time applications, latency bounds can be exceeded and still have the network in operation.

In an asynchronous system, latency is not bounded.

Packet loss

Packet loss may occur due to several reasons, like interference or fading. It has a negative impact on the performance of applications and is defined as a rate between lost packets and total packets sent. Packet loss manifests itself by the loss or re-transmission of data.

These events can be due to a faulty link corrupting or losing packets. Packet loss has negative impacts on the performance of the system. Packet lost becomes more relevant as the requirements for Quality of Service grow more strict. For use cases that require reliable data transmission, packet loss rate or packet loss mitigations (like error corrections or redundant bytes) are important to be known.

Data Throughput

Data throughput characterises the amount of data that can be, or must be, transferred within a certain time frame, hence being usually expressed as a rate of bits per second (bps). The channel data rate, which is the maximum amount of data that a channel can

transfer per unit of time, must be larger than the data throughput required by the application.

2.5.2 Implicit network requirements from application characteristics

In this section the requirements that are defined are size, autonomy, mobility, the geographical scale and the network scalability.

Size

The application size, measured in terms of the number of sensor nodes and sensor data produced by each node, implicitly imposes requirements on the network technology to be used. Based on the application size it may be possible to determine throughput requirements for the network.

Autonomy

Autonomous or unattended operation of sensor nodes impose constraints on energy consumption. Therefore, WSN applications involving autonomous operation impose an implicit requirement of energy-efficient networking to maximise the lifetime of energy sources.

Autonomy is coupled to the network usage. It only applies to certain devices in the network. Gateways or coordinator nodes are typically required to be provide continuous service and therefore they are likely externally powered. In terms of end-nodes, autonomy is coupled with the use of the network. These nodes are normally powered by batteries meaning they require an efficiency management of their energy.

Mobility

Some applications require network devices to physically move during operation. This translates to a constant mutation in the network topology and to autonomy requirements, which impose constraints to the networking technologies that may be used. As a result, it might not be possible to ensure at deployment time that all nodes remain within a certain distance of other nodes. Thus, upper bounds for distance must be considered when selecting and configuring a network. Mobility also brings forth the problem of dynamic routing which adds further constraints.

Mobility can be classed in two ways: short range and long range. Long range is defined when the host of the device travels long distances that span outside the cut off distance. This cut off distance is associated with the technology's gateway transmission coverage which then would mean a transfer in the gateway responsible for the transmission. Since this process is not trivial it needs to be taken into account.

The speed at which the device moves is not negligible, specially if the use case requires frequent low latency communication.

Geographical scale

WSN applications need to cover a physical area, which may be significantly wide. Knowing the span of the WSN is fundamental for choosing a technology that can provide the required coverage. Depending on the technology, coverage can be solved by using multi-hop or large distance single-hop transmission.

Surrounding environment

Characterises the physical environment surrounding the nodes on the network. It has an effect in the transmission of data. Buildings, interference from other transmitting devices and trees, among many other natural obstacles, influence the reliability of the transmission and hence impose additional requirements on transmission power and transmission distance.

Network scalability

With the rapid growth of inter-connectivity and ubiquitous systems, applications need to have scalability into account.

Scalability requirements of a WSN application express the needs for seamless addition of new nodes without affecting the network operation. If scalability is needed, then this might affect the decision on the technology to be selected.

2.5.3 Implicit network requirements from a management perspective

Still under the implicit requirements group, there is a subgroup of requirements that differ in the way they are defined. These are requirements defined not by the network designer to answer to technical or deployment (environment) constraints, but by the point of view of a project manager. Homogeneity and cost constitute this subgroup.

Normally cost and homogeneity are viewed as a global metric and not so much with the granularity of point to point communication, hence why cost and homogeneity belong in this group. In terms of cost, when establishing a financial budget for the deployment of a network, what is considered is the overall cost of the network and not each communication hop.

Homogeneity

Homogeneity of a network has impacts on its cost and on interoperability. The interoperability provided by an homogeneous network allows for simpler maintenance and

management of the infrastructure. Another positive aspect is that personnel only needs to master that specific technology. Naturally, homogeneity imposes constraints on the technology selection.

Unfortunately homogeneity is sometimes hard to achieve. To overcome this, understanding what parts of the network are more critical and making that part as homogeneous as possible could help reduce the complexity of the overall network. It could be argued that the level of homogeneity desired is proportional to the complexity of the network.

Cost

Many factors can influence cost when choosing a wireless technology for a WSN. The hardware, building and maintaining the infrastructure, training of personnel and service subscription fees, all weight on the final decision. Depending on the context, cost can either be seen as a one-time expenditure or continuous over time.

2.6 Summary

The first section of the chapter dove into the state of the art of Wireless Sensor Networks and the current studies being done on LPWAs, the new category of wireless technologies. It was learned that there are indeed work being done in trying to understand how to design and deploy a WSN network in a more objective and cost efficient way. Despite its novelty, LPWAs are the centre of a large volume of work and studies. They have as objective characterise LPWAs and how their practical use and performance fare against its theoretical models by the use of use case implementations.

Afterwards, LPWA technologies (NB-IoT, Sigfox and LoRaWAN) are introduced and explained in detail. Although they differ in various technical parameters, three common aspects between the three technologies were identified: they are comprised by end-nodes, these end-nodes communicate with a gateway, the gateway communicates with the network server (cloud).

Given that LPWAs are considered the main solutions for wireless communication in IoT for the future, it was necessary to know which technologies were there before. A quick overview of ZigBee was given and it was understood that perhaps there are still use cases where ZigBee wins over LPWAs. In the comparison between LPWAs and ZigBee, the difference between the two was noticeable. ZigBee provides larger throughput, less autonomy and coverage and more flexibility in deployment, in contrast, LPWAs have as priority long battery life and large geographical coverage for any deployment environment. Therefore, they have serious limitations in data throughput.

Shifting the focus from the wireless technologies used in a WSN towards the general view of the entire network, the most used requirements that characterise a network are listed. The requirements were divided in two groups, explicit and implicit requirements.

This distinction stems from the way the requirements are defined. Explicit requirements are specified to the network and implicit requirements are inherent to it. This final section also helps to put in perspective the methodology presented in the next chapter.

Chapter 3

Wireless Technology Selection Methodology

This chapter will explain the methodology that was created to answer to the problems expressed in Chapter 1. It gives a complete understanding of the methodology.

The structure of this chapter is as follows: Section 3.1 gives the problem definition. Afterwards, Section 3.2 presents an overview of the methodology followed by Section 3.3 which describes the methodology in detail.

3.1 Problem Definition

The problems that the methodology tries to address are:

- The selection a wireless technology for a WSN that can address every application and functional requirement.
- The selection of a wireless technology in every use case.
- Remove non-technical influences from the choosing process.
- No standard or automated way of choosing a wireless technology for a WsN.

A methodology that could solve all there problems would be ideal but extremely complex. In the hopes of simplifying some of the identified problems a few assumption were made before devising the methodology:

- The approximate position where individual devices should be deployed is known.
- The technologies to be considered as possible alternatives is limited to those introduced in Section 2.2.
- There is at least one central point in the network (gateway) that is the destination of the data flow. Mesh like topologies are not considered. Mobile devices will also not be considered.

- The methodology may not provide a single final solution, it presents a solution space ranked by the criteria given.
- The methodology is agnostic to the application layer.

3.2 Methodology Overview

The proposed methodology is meant to be used from the application requirements definition up to the deployment phase. The methodology is automated but leaves some leeway for the network designer to choose his preferred solution. The result of the methodology is a solution space where a solution can be chosen from.

3.3 The Methodology

This Section describes the methodology and the steps and procedures that constitute it. It is divided in Sections that define four distinct steps: establishing the vertices, linking the vertices, superposing partial graphs and defining the solution space.

Firstly, in Section 3.3.1, a few basic concepts regarding the methodology are introduced. There on, a detailed description of each step is given.

Throughout the Section there will be a recurring example which exemplifies visually the various processes described.

3.3.1 Basic concepts

The methodology utilises a graph to convey the physical network topology. A graph is used because it is the most straightforward way of representing a network.

Taking advantage of using a graph as a way to represent the network, it is possible to the utilise its edges and vertices as holders of requirements that correspond to its physical counterparts.

A vertex in the graph represents an individual node on the network and contains the pertinent information about the requirements imposed by that node. Implicit requirements are stored in the vertices.

Edges, on the other hand, represent the physical links between nodes in the network. They contain the explicit requirements that characterise the communication between those two points on the network as well as the difficulties/obstacles (transmission environment) to data transmission.

The first three steps can be viewed as a graph construction process in which the output is used in the definition of the solution space. After the explanation of these first three steps, Section 3.3.5 takes the raw information produced by the final graph and moulds it in a way that can be easily manipulated to create the solution space.

3.3.2 Establishing the vertices

The first step in building the graph is to add the vertices that represent each physical point on the network. The first set of requirements that are analysed are implicit requirements. These requirements give a superficial, non-technical and general understanding of the network. Translating this to the graph would mean annotating the vertex with information such as: the amount of data sent and received by the node, its surrounding environment, the level of autonomy needed and its purpose in the network (gateway, end node, relay node).

Although this information is still not enough to rule out candidate technologies, it provides a solid foundation to build the remaining graph. Depending on the information given, it could also already hint possible solutions.

The first occurrence of the example is demonstrated in Figure 3.1. Applying the first step of the methodology discussed in this section we end up the following:

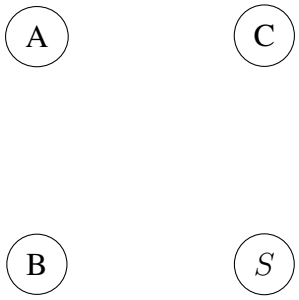


Figure 3.1: Graph with annotations

Table 3.1: Annotation of the requirements in each vertex.

Vertex	Purpose	Mobility	Data Scale	Autonomy	Surr. Env
A	Sensor	None	40 bytes	No power 2 weeks	Open
B	Camera	None	480p photos	No power 2 weeks	River side
C	Sensor	None	50 bytes	No power 2 weeks	Open
S	Gateway	None	-	Power source	Suburban

3.3.3 Linking the vertices

The second step consists in establishing edges by connecting the vertices until a sink is reached. The edges need to be annotated with the information regarding the communication between the two vertices that it connects. The edge requirements are imposed

by the vertices' requirements. Edges hold the maximum expected latency, the minimum throughput required, the maximum accepted packet loss, the physical distance of the link and if there is a line of sight (obstacles).

This process to connect the vertices is described in Algorithm 1:

Algorithm 1 Linking Vertices

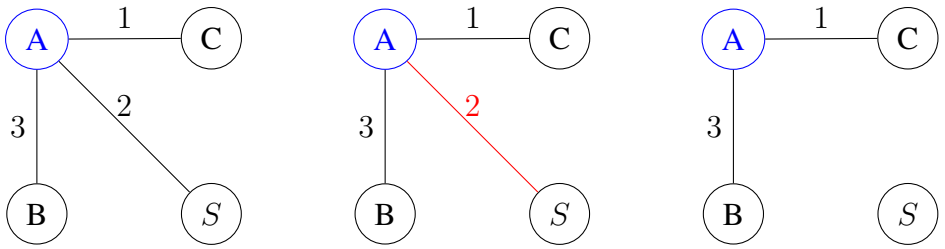
```
1: let adjacency_list
2: let edge_map
3: let paths ← {}
4: procedure LINKVERTICES(array_of_vertices, adjacency_list, paths, edge_map)
5:     for starting_vertex in array_of_vertices do
6:         let starting_vertex_adj_list ← GetAdjacencyListByVertex(
               starting_vertex, adjacency_list)
7:         let curr_path ← {}
8:         LinkVerticesAux(starting_vertex_adj_list, curr_path, paths,
               adjacency_list, edge_map)
9: procedure LINKVERTICESAUX(starting_vertex_adj_list, curr_path, paths,
   adjacency_list, edge_map)
10:    let i ← 0
11:    for vertex in starting_vertex_adj_list do
12:        let already_processed ← CheckEdgeProcessedState(
               starting_vertex_adj_list[i], vertex, curr_path)
13:        if already_processed then
14:            continue
15:        let edge ← GetEdgeByVertices(starting_vertex_adj_list[i], vertex)
16:        AnnotateEdgeWithRequirments(edge, curr_path)
17:        let was_pruned ← PruneEdge(edge)
18:        if was_pruned then
19:            continue
20:        curr_path.Append(edge)
21:        if vertex.IsGateway() then
22:            paths.Append(curr_path)
23:            return
24:        else
25:            let starting_vertex_adj_list ← GetAdjacencyListByVertex(vertex,
                   adjacency_list)
26:            LinkVerticesAux(vertex, curr_path, paths, adjacency_list)
27:        i++
```

By holding the information related to the explicit requirements in each edge it is possible to achieve finer granularity, allowing decision making for each particular edge. Therefore, the two nodes that are connected and their respective requirements become more relevant than if we did not have this detailed view of the network.

It is important that the information on the vertices and edges be detailed and concrete. If detailed information cannot be given, then at least an estimation of the requirements

should be provided. A solution for this would be categorising requirements into a scale, according to the characteristics of the considered technologies.

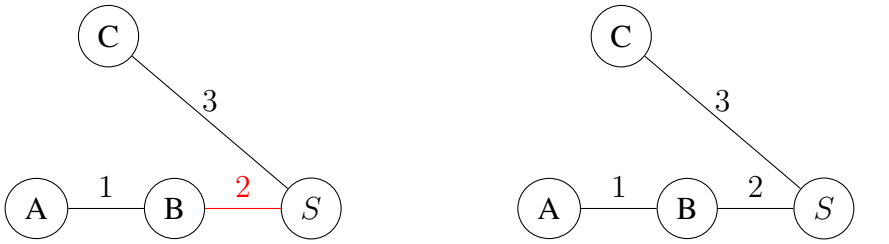
A step by step visualisation of Algorithm 1 is shown in Figure 3.2:



(a) Start (line x). (b) Edge removed (line y). (c) End iteration (line z).

Figure 3.2: Demonstration of one iteration of Algorithm 1 for one vertex (line x-z).

Not only does this approach allows for micro decision making between two nodes, but it also brings forth the notion of inherited requirements. Inherited requirements shine light on a problem that might go unnoticed and cause problems to the WSN if not caught at this early stage. Figure 3.3 demonstrates the problem encountered when ignoring inheritance.



(a) Without requirement inheritance. (b) With requirement inheritance.

Figure 3.3: Visual showcase of the impact requirement inheritance has on the network.

Let us consider four vertices: A, B, C and S, and three edges: edge 1 (connecting A to B) and edge 2 (connecting B to S) and edge 3 (connecting C to S). In the case that vertex A requires the transmission of large amounts of data, edge 1 would have to have the requirements necessary to reliably transmit that data. In turn, if both vertices B and C don't require to transmit as much data, edge 2 and 3 would have weaker requirements than edge 1. Thus, in this case, edge 2 would become a performance bottleneck as well as a link that would weaken the network's operability in relation to the desired Quality of Service. This is represented in Figure 3.3(a) by colouring edge 2 in red. On the other hand, in Figure 3.3(b), if the requirements present in edge 1 are passed down to edge 2, edge 2 will no longer pose constraints in the operability of the system.

3.3.4 Superposing graphs

The result of the process described in Section 3.3.3 is a set of partial graphs that can either have the same or different paths to the sink.

In the step described in this section, the graphs are superposed into a single layer, thereby creating a single graph.

Finally, Figure 3.4 shows a typical graph that has undertaken the building process up until this point. It is the final graph that will be used in Section 3.3.5.

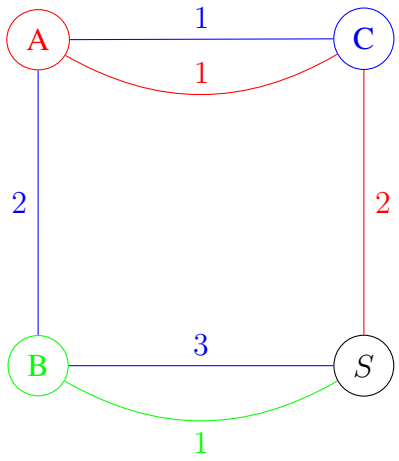


Figure 3.4: Superposition of partial graphs.

3.3.5 Definition of the solution set

After having arrived at the final graph, this section explains the second phase of the methodology. It defines a process to interpret any received graph built using the steps described in Section 3.3.2 to Section 3.3.4 and how to reliably generate a solution space. This stage is composed of two parts: extracting the information from the graph and taking that information and plotting the final solution space to a scatter plot.

Any edge of the superposed graph can have more than one set of requirements depending on the path previous taken. Simultaneously, for each edge there is also at least one or more technologies available. This poses two questions: (1) which set of requirements do we choose to cover? and (2) which technology do we choose for each edge?

These questions are not directly answered but are dependent on the final solution chosen. Before choosing a solution, it is necessary to extract the information stored in the graph and use it to establish the solution space. The first part in the definition of the solution space is as follows:

1. Pick a vertex not previously chosen;
2. Pick an outwards edge from the previously chosen vertex;
3. Of the available set of requirements, pick one not previously picked;
4. Of the available set of technologies, pick one not previously picked;

5. Repeat from step 2. until a sink is reached;
6. Add the resulting path to the set of paths starting in the vertex chosen in step 1.;
7. Repeat from step 2. until all possible combinations of requirements and available technologies from all edges have been exhausted;
8. Go to step 1. until all starting vertex have been subject to the algorithm.

The output of this algorithm is all the possible combination of paths from each vertex to the sink. In other words, for each vertex there is a corresponding set of paths starting on that vertex and ending on the sink vertex.

Algorithm 2 demonstrates the second and final step in creating the solution space. All these paths (extracted from the previous step) are then combined, generating several graphs. The idea is that a graph contains one path from each vertex to the sink. All the possible combinations are considered, which means that the total number of combinations is directly dependent on the number of paths that connect each vertex to the sink.

The procedure for the second part of the solution space definition is defined in 2.

The problem is that the solution space is massive and not ordered by any sort of metric that would allow for an objective choice to be taken. To solve this, graphs are rated according to criteria. These criteria can be anything the network designer deems relevant (e.g. in Chapter 5 the three criteria used are homogeneity, cost and the average hops in the graph). There could be various ways of defining the same criteria, thus making criteria subjective.

As mentioned beforehand, the solution space might become massive. To counter this, graphs are rated and then passed into an algorithm that computes the Pareto Frontier. The Pareto Frontier receives the final criteria rating list and filters the criteria ratings that are deemed optimal by the algorithm. This is done with the purpose of sieving a small subset of only optimal solutions from the original large solution set. After all, it is one of the objectives of the methodology to find a solution as optimal as possible and remove unnecessary solutions helps the decision making.

If the criteria rating of a certain graph becomes part of the Pareto Frontier then it is added to the scatter plot. Otherwise, it is discarded for being sub-optimal. Given that each graph can be rated by a multitude of criteria is it not only possible to have a single multi-dimensional space plot (Pareto Frontier allows multi-dimensional space), but also to have several plots each one featuring pairs of criteria. This allows the network designer to have a plot with the overall ratings of each graph and then several auxiliary plots for specific comparison between criteria.

Finally, a solution, as in, a point on a plot, must be chosen. It is the network designer that has the responsibility of picking the solution that exhibits a desirable or acceptable compromise among the criteria. Although this final solution is rated by human defined

Algorithm 2 Second step of the definition of solution space (Path combination).

```
1: let paths_from_linkproc
2: let vertices
3: let graphs ← {}
4: procedure COMBINATIONPROCESS(path, paths_from_linkproc, graphs, vertices)
5:     for path in paths_from_linkproc do
6:         CombinationProcessAux(path, paths_from_linkproc, graphs, vertices, 0)
7: procedure COMBINATIONPROCESSAUX(path, paths_from_linkproc, graphs,
   vertices, index)
8:     let curr_graph ← {}
9:     curr_graph.Append(path)
10:    if curr_graph.Size() == 1&&curr_graph[0].Size() == vertices.Size() then
11:        graphs.Append(curr_graph)
12:    else
13:        for index; index < paths_from_linkproc; index + + do
14:            let candidate_path ← paths_from_linkproc[index]
15:            if ArePathsEqual(curr_path, candidate_path) then continue
16:            if CheckRepeatedVertices(curr_graph, candidate_path, vertices) then
17:                continue
18:            curr_graph.Append(candidate_path)
19:            SortPathsInGraph(curr_graph)
20:            if DoesGraphAlreadyExist(curr_graph, graphs) then
21:                graphs.Remove(curr_graph)
22:                continue
23:            if IsGraphCompleted(curr_graph, vertices) then
24:                graphs.Append(curr_graph)
25:            CombinationProcessAux(curr_path, paths_from_linkproc, graphs,
                   vertice, index + 1)
```

criteria and ultimately chosen by a human, it does not fall short on the objectives that this methodology was set out to achieve. When the graph is completed in Section 3.3.4, all the present paths are sure to comply with the requirements given. Therefore, the solution space previously defined is comprised of every graph that satisfies both application and functional requirements.

This final solution also dictates the final topology of the network. It is important to note that this does not violate the objectives that this methodology was set out to achieve. The process up to the point of criteria definition is completely objective and has no human input, it solely relies on technical details, application and functional requirements. The subjectivity of the criteria and the fact that the choice is ultimately made by a human serves as leeway. It provides a certain degree of flexibility where although the solution space is enforced, the final solution is not.

3.4 Summary

The main contribution of the thesis was presented in this chapter. It consists of a two phase methodology that solves the problem of choosing a wireless technology for a WSN. The methodology uses a graph to convey the physical network. There are three distinct steps to build the graph: establishing the vertices, linking the vertices and superposing the graphs. The output of each step servers as input for the next. The vertices are annotated with the node's requirements and linked to each other. The edges are then given the link's requirements, if these requirements can not be met by the considered technologies, the edge is pruned. Lastly, all the created graphs are superposed into a single graph.

The second phase of the methodology takes resulting graph from the last phase as input. The final solution space results in a process that takes the paths from the graph and adds each one as points into a plot.

Comparing to other methodologies already seen in Section 2.1, one of the main differences that can be seen in the methodology described in this chapter is that it aims to be an automated process where the only required interactions from the network designer is the requirement annotation in the first two steps of the first phase of the methodology. With the desired automation, human bias is mitigated and only objective solutions are suggested, this is because the application and functional requirements are the only factors that affect the choices made each step. This chapter had a theoretical approach, brushing over each step giving a detailed but abstract idea of each step of the methodology.

The next section will have a more practical approach to the problem, an implementation of the methodology will be done.

Chapter 4

Implementation

It is in Section 4 where the steps described in Section 3.3 take a more concrete form by giving a technical explanation.

The implementation of the methodology was done in C++ without multithread support. The program requires interaction with JSON files, therefore, Niels Lohmann's Json library[25] was used. Two other helper Python scripts were used - one for the Pareto Algorithm[26] and the other to generate the scatter plots with the help of Matplotlib[27].

In Section 4.1 an overview of the implementation architecture is given as well as an explanation of the data structures used. Afterwards, in Section 4.2, code from the annotation and pruning procedure is shown and explained.

4.1 Program Architecture and Data Structures

The program requires a Json setup file. This setup file needs to be filled by the network designer with relevant information such as the requirements of each node, information about each link between nodes and the considered technologies' technical details. The setup file used in the Aquamon use in Chapter 5 can be seen in Listing 4.1.

```
1  {
2    "num_nodes" : 5,
3    "vertices": [
4      {"id": "A", "purpose": 1, "surrounding_environment": 1, "data_scale
           ": 30, "mobile": false},
5      {"id": "B", "purpose": 1, "surrounding_environment": 1, "data_scale
           ": 250, "mobile": false},
6      {"id": "C", "purpose": 1, "surrounding_environment": 1, "data_scale
           ": 30, "mobile": false},
7      {"id": "D", "purpose": 1, "surrounding_environment": 1, "data_scale
           ": 30, "mobile": false},
8      {"id": "S", "purpose": 2, "surrounding_environment": 2, "data_scale
           ": 0, "mobile": false}
9    ],
10   "edges": [
11     {"id": "AB", "distance":  450, "obstacle_level": 1},
12     {"id": "AC", "distance":  580, "obstacle_level": 2},
```

```
13      {"id": "AD", "distance":  1460, "obstacle_level": 2},
14      {"id": "AS", "distance":  880, "obstacle_level": 2},
15      {"id": "BC", "distance":  1000, "obstacle_level": 3},
16      {"id": "BD", "distance":  1900, "obstacle_level": 3},
17      {"id": "BS", "distance":  1280, "obstacle_level": 3},
18      {"id": "CD", "distance":  934, "obstacle_level": 1},
19      {"id": "CS", "distance":  545, "obstacle_level": 2},
20      {"id": "DS", "distance":  1140, "obstacle_level": 2}
21    ],
22    "technologies": [
23      {"id": "lorawan", "max_urban_distance": 10000, "
          max_suburban_distance": 15000, "data_throughput": 200, "
          packet_loss":  0.1, "cost": 45.5},
24      {"id": "nbiot", "max_urban_distance": 10000, "max_suburban_distance
          ": 20000, "data_throughput": 250, "packet_loss":  0.02, "cost":
           40.0},
25      {"id": "sigfox", "max_urban_distance": 5000, "max_suburban_distance
          ": 7000, "data_throughput": 50, "packet_loss":  0.15, "cost": 7
          0.15}
26    ]
27  }
```

Listing 4.1: Example of a setup file.

Some requirements require a scale, either to identify the obstacle level between nodes, to specify the node's purpose on the network or to characterise the surrounding environment. The `surrounding_environment` goes from 1 to 2 where 1 is urban environment and 2 is suburban environment. `purpose` goes from 1 to 4: (1) sensor, (2) gateway, (3) multimedia and (4) relay. Finally, the `obstacle_level` starts at 0 up to 3, 0 being line-of-sight or open environment and 3 being high density of obstacles.

The information is gathered from the setup file and populated into the data structures to be used in the runtime of the program. These are `struct Technology`, `struct Requirements`, `struct Vertex`, `struct Edge` and `struct Path` and `struct Graph`.

```
struct Technology{
    std::string id;
    float       max_urban_coverage;
    float       max_suburban_coverage;
    uint32_t    max_data_throughput;
    float       percentage_of_pl;
    double      cost = 0.00f;
};
```

Listing 4.2: Technology struct.

```
struct Vertex{
    std::string id;
    bool        mobile;
    int32_t     data_scale;
    int32_t     surr_env;
    int32_t     purpose;
```

```
};
```

Listing 4.3: Vertex struct.

```
struct Requirements{
    int32_t min_acceptable_packet_loss;
    int32_t data_throughput;
    float   packet_loss;
};
```

Listing 4.4: Requirements struct.

```
struct Edge{
    Requirements            reqs;
    int32_t                 obstacle_level;
    float                   distance;
    Vertex                  from;
    Vertex                  to;
    std::vector<Technology> accepted_technologies;
};
```

Listing 4.5: Edge struct.

```
struct Path{
    std::vector<Edge>   edges;
    double              cost = 0.00f;
};
```

Listing 4.6: Path struct.

```
struct Graph{
    uint32_t            id;
    std::vector<Path>   paths;
    double              cost = 0.00f;
    double              homogeneity_ratio = 0.00f;
    double              hops_average = 0.00f;
    uint32_t            num_of_edges = 0;
};
```

Listing 4.7: Graph struct.

Not only are the structs created and initialised, the adjacency list of each node are also created. The adjacency list takes form as matrix of `Vertex` where the first position of each vector is the vertex and the remaining of the vector its adjacent neighbours.

A map of `std::pair<Vertex,Vertex>` to `Edge` is created to hold the edges that connect the nodes. This map is used in the linking process to retrieve the edges.

The output of the main program is a `output.dat` file with the criteria ratings of all the graphs generated. This file will then be the input to the first Python helper script which in turn outputs a file with the criteria ratings that belong on the Pareto Frontier. The second Python script reads both files and generates the scatter plots. An overview of the use of the tool can be seen in Figure 4.1.

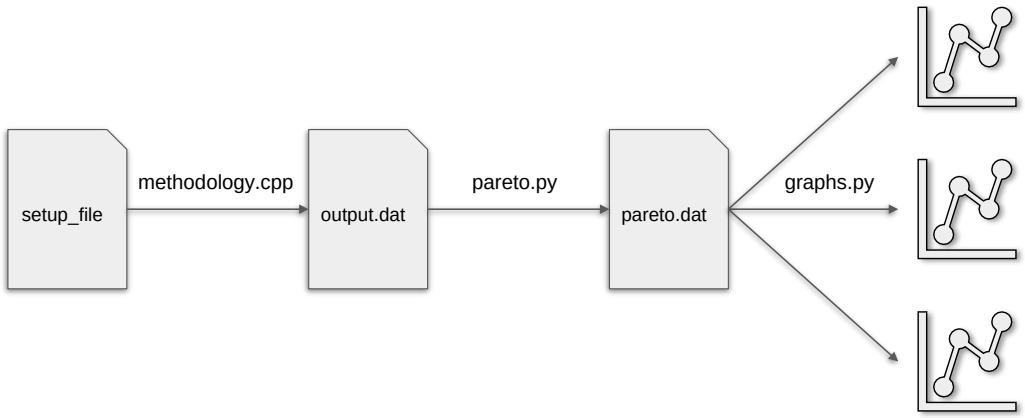


Figure 4.1: Runtime stages from the setup file input to the final scatter plots output.

4.2 Code

With the initial setup completed the automated process starts. During the linking process the annotation of the requirements on the edge is shown in Listing 4.8. The annotation process is rather trivial and in this version of the implementation only the throughput is considered. Emulating packet loss and quality of service is complex so these were not used in the linking phase.

Throughput is calculated based on the amount of data the node needs to send. It is calculated as the data was sent in a single sitting, periodic transmission of data is not considered in this version of the implementation.

Since paths have multiple nodes and the data flows from the starting vertex up to the gateway, the nodes along the path need to account for the data received from previous nodes. Therefore, in Line 12 from Listing 4.8, the throughput is incremented to account for the transmission of data from the starting vertex up to that current vertex.

```
1  void annotate_edge_with_requirement(Edge &e, Path &path_from_traversing
       , const std::map<vpair_string, Edge> &edge_map) {
2
3       Requirements new_req_set;
4       int32_t data_throughput = e.from.data_scale;
5       new_req_set.data_throughput = data_throughput;
6       e.reqs = new_req_set;
7
8       if(path_from_traversing.edges.empty()){
9           return;
10      }else{
11          Edge previous = path_from_traversing.edges.back();
```

```
12          e.reqs.data_throughput += previous.reqs.data_throughput;
13      }
14  }
```

Listing 4.8: Procedure to annotate the edge's requirements.

With the requirements annotated the pruning procedure can start (Listing 4.9). Line 4 tries to deal with problem of diverging architecture between the technologies. NB-IoT and Sigfox connect directly to a base station and do not allow for point to point communication. The way used to circumvent this problem is to prune edges that do not connect to the gateway if they are using NB-IoT or Sigfox.

Pruning simply compares the amount of throughput and coverage of each technology to their counterparts annotated in the edge (Line 7 to Line 9). A ratio is calculated to simulate the problems often encountered when the transmission between nodes is big and close to the max distance possible; the possibility of obstacles is also accounted for. The calculation of said ratio can be seen in Lines 11 and 12. The max coverage of the technology is divided by the distance between the nodes, in turn, the result is divided by the obstacle level between the two nodes. Naturally, as the obstacle level increases, the resulting value decreases. If the ratio is below 1 then the edge is pruned because it is deemed unstable - distance too great in relation to the max coverage of the technology and the obstacles present.

```
1  bool prune_edge(Edge &e, const std::vector<Technology> &technologies){
2      for(const auto &tech : technologies){
3          //skip NB-IoT and Sigfox if target vertex isn't a gateway
4          if(e.to.purpose != 2 && ((tech.id == "sigfox") || (tech.id == "
               nbiot"))){
5              continue;
6          }
7          bool throughput = tech.max_data_throughput >= e.reqs.
               data_throughput;
8          bool suburban_coverage = tech.max_suburban_coverage >= e.
               distance;
9          bool urban_coverage = tech.max_urban_coverage >= e.distance;
10
11         bool urban_obstacle_level = (tech.max_urban_coverage / e.
               distance / e.obstacle_level) >= 1;
12         bool suburban_obstacle_level = (tech.max_suburban_coverage / e.
               distance / e.obstacle_level) >= 1;
13
14         if(throughput){
15             if(e.from.surr_env == 1){
16                 if(urban_coverage && urban_obstacle_level){
17                     e.accepted_technologies.push_back(tech);
18                 }
19             }else{
20                 if(suburban_coverage && suburban_obstacle_level) {
21                     e.accepted_technologies.push_back(tech);
22                 }
23             }
24         }
```

```
25      }
26      return e.accepted_technologies.empty();
27  }
```

Listing 4.9: Procedure to verify if an edge needs to be pruned.

4.3 Summary

The Chapter started with an overview of the architecture and the data structures used in the implementation of the methodology described in Chapter 3.3. These were followed by brief explanations on the choices made while implementing. Some shortcomings of the implementation were indicated and will be expanded on Chapter 6.

Afterwards, a showcase of the annotation and pruning procedures was done complemented by some comments on the implementation of each procedure.

In the following Chapter the methodology will be put to test by the means of a use case and its results analysed.

Chapter 5

Evaluation

Chapter 3 theoretically explained the methodology and the steps that constitute it, afterwards, in Chapter 4, a practical implementation of the methodology was done. But the creation and implementation of the methodology is not enough to prove its virtues. In this Chapter an evaluation of the methodology will take place: firstly by defining what will be evaluated, how it is going to evaluated and finally a discussion of the results.

5.1 Evaluation Objectives

In Section 3.3.5 it was explained how the solution space can take a massive size. The size of the solution space is reduced by using the Pareto Frontier algorithm resulting in a smaller sub-set of optimal solutions. The main objective of this Chapter is to understand, based on practical experimentation, if the methodology holds up to its theoretical effectiveness. In other words, if the results of practical uses of the methodology can reduce the possible optimal solution space enough that a human can analyse the results and choose the solution he deems best for the that particular use case.

The evaluation of the methodology's performance and which factors weight on it is also important. The number of possible solutions grows exponentially with each added node rendering the use of the methodology useless, thus, making the methodology's performance a relevant evaluation metric.

In short, effectiveness and performance of the methodology are the two factors evaluated in this Chapter.

5.2 Method of Evaluation

Effectiveness and performance differ in the ways they are measured. Effectiveness is measured by comparing the resulting sub-set of optimal solutions to the total number of solutions in the solution space. Therefore, the level of effectiveness of the methodology is tied to the percentage of reduction of the solution space. The methodology is deemed

effective if the optimal solution space is drastically smaller than the solution space. This way, it can be possible for the network designer to analyse and pick his solution.

Performance is measured by executing the tool on the same setup file 10 times and getting the average execution time.

To evaluate the methodology several general use cases will be used. The running environment where the tests will take place is the following: `Linux Kernel: 5.0.0-29, CPU: AMD Ryzen 5 2600X, gcc 8.3.0`.

In practical terms, four generic use cases and their respective setup files will serve as input to the methodology and its results analysed in Section 5.3. Afterwards and to contrast the generic use cases, a real world use case is also used.

Firstly, Listing 5.1 corresponds to a network with five nodes with lax requirements and considerable distances between the nodes.

```
{
  "num_nodes" : 5,
  "vertices": [
    {"id": "A", "purpose": 1, "surrounding_environment": 1, "data_scale": 20, "mobile": false},
    {"id": "B", "purpose": 1, "surrounding_environment": 2, "data_scale": 64, "mobile": false},
    {"id": "C", "purpose": 1, "surrounding_environment": 1, "data_scale": 10, "mobile": false},
    {"id": "D", "purpose": 1, "surrounding_environment": 1, "data_scale": 100, "mobile": false},
    {"id": "S", "purpose": 2, "surrounding_environment": 1, "data_scale": 0, "mobile": false}
  ],
  "edges": [
    {"id": "AB", "distance":  200, "obstacle_level": 3},
    {"id": "AC", "distance":  410, "obstacle_level": 1},
    {"id": "AD", "distance":  410, "obstacle_level": 1},
    {"id": "AS", "distance":  180, "obstacle_level": 2},
    {"id": "BC", "distance":  1200, "obstacle_level": 2},
    {"id": "BD", "distance":  1200, "obstacle_level": 1},
    {"id": "BS", "distance":  100, "obstacle_level": 2},
    {"id": "CD", "distance":  188, "obstacle_level": 1},
    {"id": "CS", "distance":  188, "obstacle_level": 1},
    {"id": "DS", "distance":  388, "obstacle_level": 1}
  ],
  "technologies": [
    {"id": "lorawan", "max_urban_distance": 10000, "max_suburban_distance": 15000, "data_throughput": 200, "packet_loss":  0.1, "cost": 45.5},
    {"id": "nbiot", "max_urban_distance": 10000, "max_suburban_distance": 20000, "data_throughput": 250, "packet_loss":  0.02, "cost": 40.0},
    {"id": "sigfox", "max_urban_distance": 5000, "max_suburban_distance": 7000, "data_throughput": 50, "packet_loss":  0.15, "cost": 70.15}
  ]
}
```

Listing 5.1: Generic use case - setup file with 5 nodes.

Secondly, Listing 5.2 represents a six node network with lax vertices requirements and large distances between nodes.

```
{
  "num_nodes" : 6,
  "vertices": [
    {"id": "A", "purpose": 1, "surrounding_environment": 1, "data_scale": 30, "mobile": false},
    {"id": "B", "purpose": 1, "surrounding_environment": 2, "data_scale": 36, "mobile": false},
    {"id": "C", "purpose": 1, "surrounding_environment": 1, "data_scale": 29, "mobile": false},
    {"id": "D", "purpose": 1, "surrounding_environment": 1, "data_scale": 34, "mobile": false},
    {"id": "E", "purpose": 1, "surrounding_environment": 2, "data_scale": 48, "mobile": false},
    {"id": "S", "purpose": 2, "surrounding_environment": 1, "data_scale": 0, "mobile": false}
  ],
  "edges": [
    {"id": "AB", "distance":  200, "obstacle_level": 3},
    {"id": "AC", "distance":  410, "obstacle_level": 1},
    {"id": "AD", "distance":  412, "obstacle_level": 1},
    {"id": "AE", "distance":  400, "obstacle_level": 1},
    {"id": "AS", "distance":  180, "obstacle_level": 2},
    {"id": "BC", "distance":  1100, "obstacle_level": 2},
    {"id": "BD", "distance":  1037, "obstacle_level": 1},
    {"id": "BE", "distance":  100, "obstacle_level": 2},
    {"id": "BS", "distance":  87, "obstacle_level": 2},
    {"id": "CD", "distance":  188, "obstacle_level": 1},
    {"id": "CE", "distance":  139, "obstacle_level": 1},
    {"id": "CS", "distance":  113, "obstacle_level": 1},
    {"id": "DE", "distance":  30, "obstacle_level": 1},
    {"id": "DS", "distance":  51, "obstacle_level": 1},
    {"id": "ES", "distance":  119, "obstacle_level": 1}
  ],
  "technologies": [
    {"id": "lorawan", "max_urban_distance": 10000, "
       max_suburban_distance": 15000, "data_throughput": 200, "
       packet_loss":  0.1, "cost": 45.5},
    {"id": "nbiot", "max_urban_distance": 10000, "max_suburban_distance": 20000, "data_throughput": 250, "packet_loss":  0.02, "cost":  40.0},
    {"id": "sigfox", "max_urban_distance": 5000, "max_suburban_distance": 7000, "data_throughput": 50, "packet_loss":  0.15, "cost": 70.15}
  ]
}
```

Listing 5.2: Generic use case - setup file with 6 nodes.

Finally, Listing 5.3 represents a similar network to that of Listing 5.2 but with an added node. Listing 5.4 is based on the original seven node network but with laxer requirements

and shorted distances between nodes. The reason why a second seven node network with laxer requirements is used is because, due to the pruning procedure logic, the less strict the requirements are the bigger solution space becomes. In short, the second use case is a worst case scenario of the original use case. These three use cases and their slight differences between each other serve to understand the influences that small changes in requirements and the distances between nodes make.

```
{
  "num_nodes" : 7,
  "vertices": [
    {"id": "A", "purpose": 1, "surrounding_environment": 1, "data_scale
       ": 30, "mobile": false},
    {"id": "B", "purpose": 1, "surrounding_environment": 2, "data_scale
       ": 36, "mobile": false},
    {"id": "C", "purpose": 1, "surrounding_environment": 1, "data_scale
       ": 29, "mobile": false},
    {"id": "D", "purpose": 1, "surrounding_environment": 1, "data_scale
       ": 34, "mobile": false},
    {"id": "E", "purpose": 1, "surrounding_environment": 2, "data_scale
       ": 48, "mobile": false},
    {"id": "F", "purpose": 1, "surrounding_environment": 1, "data_scale
       ": 80, "mobile": false},
    {"id": "S", "purpose": 2, "surrounding_environment": 1, "data_scale
       ": 0, "mobile": false}
  ],
  "edges": [
    {"id": "AB", "distance":  200, "obstacle_level": 3},
    {"id": "AC", "distance":  410, "obstacle_level": 1},
    {"id": "AD", "distance":  412, "obstacle_level": 1},
    {"id": "AE", "distance":  400, "obstacle_level": 1},
    {"id": "AF", "distance":  310, "obstacle_level": 1},
    {"id": "AS", "distance":  180, "obstacle_level": 2},
    {"id": "BC", "distance":  1100, "obstacle_level": 2},
    {"id": "BD", "distance":  1037, "obstacle_level": 1},
    {"id": "BE", "distance":  100, "obstacle_level": 2},
    {"id": "BF", "distance":  1200, "obstacle_level": 2},
    {"id": "BS", "distance":  87, "obstacle_level": 2},
    {"id": "CD", "distance":  188, "obstacle_level": 1},
    {"id": "CE", "distance":  139, "obstacle_level": 1},
    {"id": "CF", "distance":  144, "obstacle_level": 1},
    {"id": "CS", "distance":  113, "obstacle_level": 1},
    {"id": "DE", "distance":  188, "obstacle_level": 1},
    {"id": "DF", "distance":  139, "obstacle_level": 1},
    {"id": "DS", "distance":  144, "obstacle_level": 1},
    {"id": "EF", "distance":  188, "obstacle_level": 1},
    {"id": "ES", "distance":  139, "obstacle_level": 1},
    {"id": "FS", "distance":  144, "obstacle_level": 1}
  ],
  "technologies": [
    {"id": "lorawan", "max_urban_distance": 10000, "
       max_suburban_distance": 15000, "data_throughput": 200, "
       packet_loss":  0.1, "cost": 45.5},
    {"id": "nbiot", "max_urban_distance": 10000, "max_suburban_distance
       ": 20000, "data_throughput": 250, "packet_loss":  0.02, "cost":
```

```
        40.0},
38    {"id": "sigfox", "max_urban_distance": 5000, "max_suburban_distance
        ": 7000, "data_throughput": 50, "packet_loss":  0.15, "cost": 7
        0.15}
39  ]
40 }
```

Listing 5.3: Generic use case - setup file with 7 nodes.

```
1  {
2    "num_nodes" : 7,
3    "vertices": [
4      {"id": "A", "purpose": 1, "surrounding_environment": 1, "data_scale
         ": 10, "mobile": false},
5      {"id": "B", "purpose": 1, "surrounding_environment": 2, "data_scale
         ": 26, "mobile": false},
6      {"id": "C", "purpose": 1, "surrounding_environment": 1, "data_scale
         ": 19, "mobile": false},
7      {"id": "D", "purpose": 1, "surrounding_environment": 1, "data_scale
         ": 14, "mobile": false},
8      {"id": "E", "purpose": 1, "surrounding_environment": 2, "data_scale
         ": 38, "mobile": false},
9      {"id": "F", "purpose": 1, "surrounding_environment": 1, "data_scale
         ": 11, "mobile": false},
10     {"id": "S", "purpose": 2, "surrounding_environment": 1, "data_scale
         ": 0, "mobile": false}
11   ],
12   "edges": [
13     {"id": "AB", "distance":  100, "obstacle_level": 3},
14     {"id": "AC", "distance":  210, "obstacle_level": 1},
15     {"id": "AD", "distance":  300, "obstacle_level": 1},
16     {"id": "AE", "distance":  240, "obstacle_level": 1},
17     {"id": "AF", "distance":  140, "obstacle_level": 1},
18     {"id": "AS", "distance":  180, "obstacle_level": 2},
19     {"id": "BC", "distance":  1000, "obstacle_level": 2},
20     {"id": "BD", "distance":  900, "obstacle_level": 1},
21     {"id": "BE", "distance":  100, "obstacle_level": 2},
22     {"id": "BF", "distance":  1200, "obstacle_level": 2},
23     {"id": "BS", "distance":  30, "obstacle_level": 2},
24     {"id": "CD", "distance":  40, "obstacle_level": 1},
25     {"id": "CE", "distance":  57, "obstacle_level": 1},
26     {"id": "CF", "distance":  34, "obstacle_level": 1},
27     {"id": "CS", "distance":  62, "obstacle_level": 1},
28     {"id": "DE", "distance":  128, "obstacle_level": 1},
29     {"id": "DF", "distance":  119, "obstacle_level": 1},
30     {"id": "DS", "distance":  134, "obstacle_level": 1},
31     {"id": "EF", "distance":  18, "obstacle_level": 1},
32     {"id": "ES", "distance":  136, "obstacle_level": 1},
33     {"id": "FS", "distance":  117, "obstacle_level": 1}
34   ],
35   "technologies": [
36     {"id": "lorawan", "max_urban_distance": 10000, "
         max_suburban_distance": 15000, "data_throughput": 200, "
         packet_loss":  0.1, "cost": 45.5},
37     {"id": "nbiot", "max_urban_distance": 10000, "max_suburban_distance
         ": 20000, "data_throughput": 250, "packet_loss":  0.02, "cost":
```

```
            40.0},
38      {"id": "sigfox", "max_urban_distance": 5000, "max_suburban_distance
            ": 7000, "data_throughput": 50, "packet_loss":  0.15, "cost": 7
            0.15}
39    ]
40  }
```

Listing 5.4: Generic use case - setup file with 7 nodes (laxer requirements).

Aquamon is a project that aims to monitor vast water bodies and to provide easy and open access to the information gathered. The Tagus' estuary is the first choice for the deployment of a WSN. It covers a vast area, washes over several urban places and is a place for a wide variety of aquatic activities. This WSN will be deployed in the Seixal Bay (southern bank of the Tagus river) with the purpose of collecting water related measurements and to study various aspects of tidal movements.

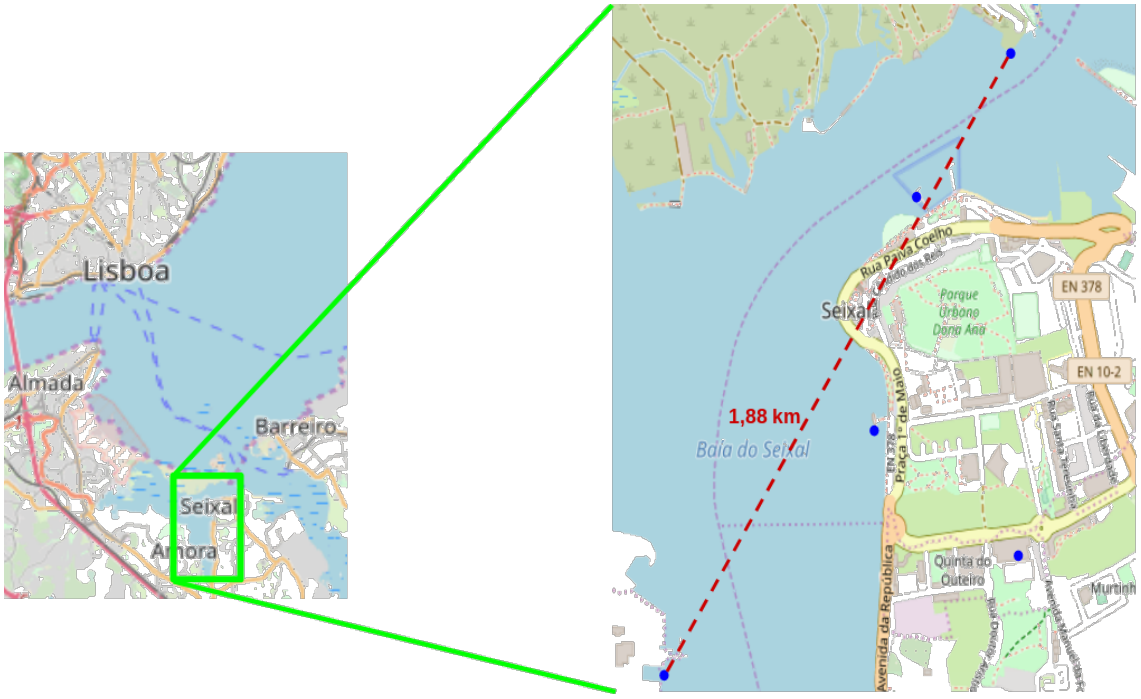


Figure 5.1: Map overview of the deployment area - Seixal Bay.

This project provides a good use case to apply and evaluate the proposed methodology. As can be seen in Figure 5.1, the area covered by the WSN contains all sorts of environments: from open waters to dense urban environments, nodes with and without line of sight between each other, some at large distances from others (the largest distance between two nodes is approximately 1,9 km).

Not only is a place that encompasses many phenomenons that can happen in aquatic environments, it also has several urban settings which give us a range of diverse deployment environments that allows for a more thorough experimentation and diverse solutions. All these factors combined make this the right choice as our test bed.

Initially, it will consist in four distinct nodes in fixed positions along the waterside and a gateway node in the centre of the city. Similarly to the deployment environment, the network is heterogeneous as well. Nodes have different purposes and operate accordingly to them. Some nodes require the periodical transmission of low amounts of sensor data while others require the reliable transmission of video. This heterogeneity of requirements makes it harder to select convenient communication technologies as well as defining an optimal topology.

Therefore, this use case is appropriate to evaluate the applicability of the proposed methodology with the objective of understanding which solutions are possible and if an optimal solution can be found.

The purpose of testing the methodology within this use case is to have a concrete idea of the practical use of the methodology and usability beyond a theoretical one. The methodology will be applied as demonstrated in Chapter 3.3 to the five nodes of the to-be-deployed WSN. The results will then be evaluated in two ways: first by analysing its optimal solutions and justifying why they seem optimal in relation to the defined criteria and secondly by comparing the obtained solutions with a standard rationalisation and decision making that would occur if the methodology did not exist.

The first step in applying the methodology to the use case in question is to establish the vertices, resulting in Figure 5.2, and annotate them (Table 5.1):

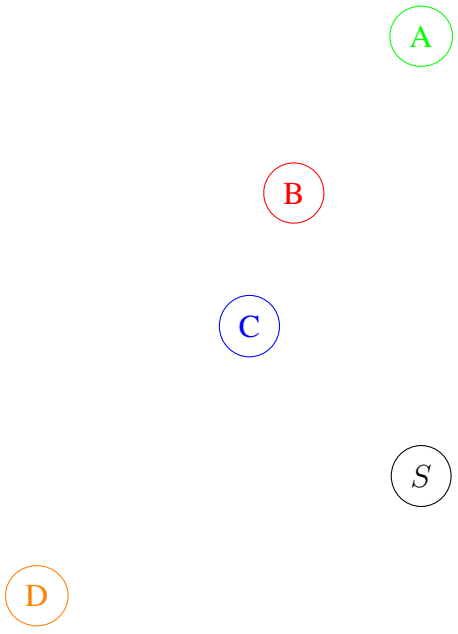


Figure 5.2: Establishing the vertices according to their physical layout on the map.

Table 5.1 gives an overview of the requirements for each vertex. Vertices A,C and D are nodes which hold sensors and their purpose is to transfer the data gathered to the gateway. The size of the data gathered is small and the nodes are immobile. Since these nodes will be placed riverside, power sources can not be reached, forcing the use of batteries.

Although two weeks is probably not enough to deplete the batteries, frequent maintenance of the sensors is needed (cleaning, re-positioning) thus a change of batteries can be done. Vertex B has a camera that takes pictures periodically. Although the node is close to the water it is within reach of a power source. The node is immobile.

Finally, vertex S is the gateway and is located inside a building in the middle of the city, it has access to a power source as well a stable internet connection.

Table 5.1: Annotation of the requirements in each vertex.

Vertex	Purpose	Mobility	Data Scale	Autonomy	Surr. Env
A	Sensor	None	30 bytes	No power 2 weeks	Open River side
B	Camera	None	480p photos	Power Source	Open River side
C	Sensor	None	30 bytes	No power 2 weeks	Open River side
D	Sensor	None	30 bytes	No power 2 weeks	Open River side
S	Gateway	None	-	Power source	Urban

After having annotated each vertex with the implicit requirements, the automated process can start. First the linking process followed by the superposition of the partial graphs.

With the conclusion of the building phase, the next automated step is the generation of the graphs. It is a this step that the solution space is generated, possibly taking massive dimensions. The criteria chosen are:

1. Homogeneity: the percentage of the most used technology in the graph.

2. Cost: the incremental cost of each hop in the graph.

3. Average number of hops per path in the whole graph.

Homogeneity and cost are common concerns in any type of network which explains their use as criteria. To reiterate, these definitions of criteria are completely subjective and can differ from person to person and from use case to use case. The third criteria tries to filter long paths in the graph, prioritising small paths and therefore small graphs. The logic behind this criteria is that long paths generate longer graphs which in turn raise the complexity of the network as well as its maintenance cost (both monetary cost as well as in terms of human resources).

The rating of the graphs takes place, afterwards, the ratings will serve as input to a Pareto Frontier algorithm and plotted to a scatter plot.

5.3 Analysis of the results

The methodology provides the network designer with the most optimal solutions according to the criteria he defined. Theoretically, this approach should provide the best solutions for each specific use case.

The results will be analysed and the methodology will be evaluated according to its effectiveness (Section 5.3.1) and its performance (Section 5.3.2).

5.3.1 Effectiveness

In this Section the effectiveness of the methodology will be measured according to the evaluation objectives defined in Section 5.1. Firstly, an analysis of the results from the execution of the methodology on the generic uses cases will take place. Afterwards, the same will be done on the results obtained from the Aquamon use case.

Generic use cases

The purpose behind using these generic use cases is to gather more information about the overall effectiveness of the methodology.

The five node use case is the first to be analysed. Having ran the methodology, 348 graphs were generated, and after the execution of the Pareto Frontier algorithm only 44 graphs remain. As a result, a 87% reduction from the generated solution space could be achieved. The two points on the Pareto Frontier are $(50, 171, 1.3)$ and $(75, 165.5, 1.3)$ with 33 and 11 graphs with the same criteria rating respectively. Although 44 graphs is still a big solution space for a human to analyse and take a decision, the fact that the Pareto Frontier only has two points allows the network designer to chose which point is the best for his use case and, therefore, reducing even more the number of graphs.

The same analysis process can be done to the use case with six nodes. The execution of the algorithm generates 3024 graphs of which only 304 are on the Pareto Frontier. This indicates a 90% reduction in solution space.

The last two use cases to be analysed are the two seven node networks. The first use case has a solution space of 20008 graphs, 1071 being on the Pareto Frontier, resulting in the reduction of 95% of the solution space. On the other hand, the second network with laxer node requirements results in 32176 but, interestingly, also has 1071 graphs in the Pareto Frontier (97% reduction). The less strict nature of the nodes' requirements led to a bigger solution space (as expected) but the result is the same as the original network. On top of this, both networks share the same point in the Pareto Frontier: $(50, 256.5, 2)$. It can be concluded from the results of these two use cases that the methodology can arrive at the same optimal solutions independently of slight changes in the vertices' requirements.

Real world use case - Aquamon

After the execution of the methodology, 75 graphs are generated, of these, 7 are in the Pareto Frontier (90.5% reduction in solution space). In Figure 5.3 the most optimal solutions take form of blue triangles with the following coordinates: $(50, 171, 1)$, $(75, 165.5, 1)$ and $(100, 160, 1)$.

Figure 5.7 and Figure 5.8 illustrate the seven optimal solutions to this use case. When looking at each graph and their criteria ratings it appears Figure 5.8c is the best solution. It can be seen that is reaches 100% homogeneity, has the lowest cost of every other graph and has one hop per path in the graph. According to the criteria and their definitions given in Section 5.2 it is clear that this graph is correctly graded and a viable choice.

The most optimal solution uses NB-IoT as technology. The definition of cost used leaves out the cost of operating the network and the subscription cost to the provider needed to use NB-IoT. If this was to be taken into account then an alternative like LoRaWAN would have been a better choice (in regards to cost) as there woudld not be the added cost of subscription.

In total four scatter plots were created (Figure 5.3 and Figure 5.4 to Figure 5.6), the first being a tri-dimensional plot which takes all the criteria in consideration and the remaining three being two-dimensional scatter plots resulting from the combination of the three criteria.

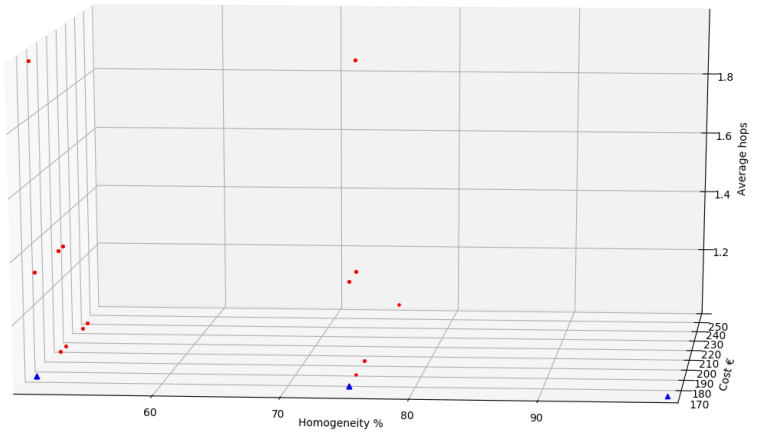


Figure 5.3: View of the 3D scatter plot over the x axis (Homogeneity).

Now focusing on the amount of hops in each path, the best solution seems to be of a single hop per path, independently of the technology used. The reason comes from every considered technology being able to satisfy the distances of each node to the gateway and therefore it can be argued that adding extra hops would add complexity and cost to the graph/network, an example of this is the graph in Figure 5.9. Perhaps a more subjective reason would be that it does not make sense to extend a path if the starting node can communicate directly with the gateway. Figure 5.5 also justifies the use o less hops to

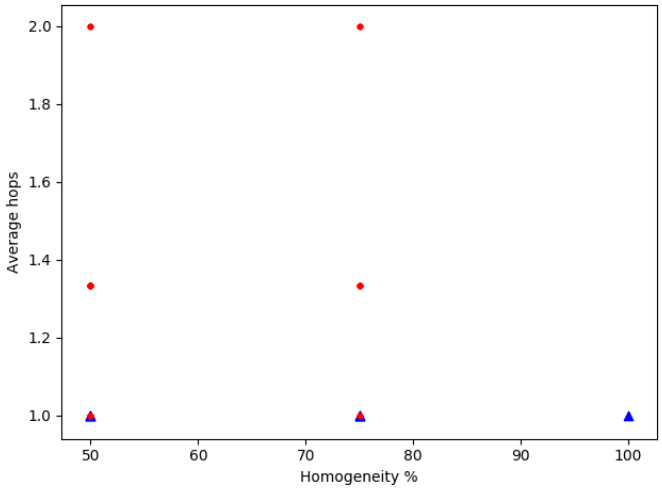


Figure 5.4: Average hops in function of homogeneity.

achieve a lower cost.

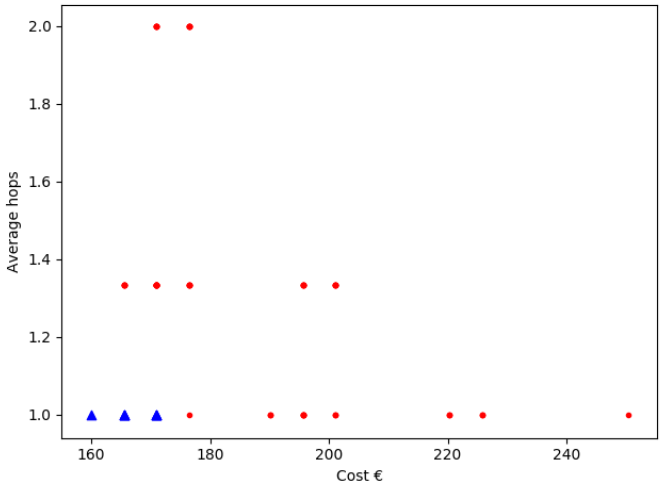


Figure 5.5: Average hops in function of cost.

Figure 5.6 shows that there is a solution that is both the most homogeneous one as well as the cheapest (solution illustrated at Figure 5.8c). Generally cost and homogeneity are orthogonal to each other and it is not always possible to reach a solution where they converge. In this use case and with the definitions given to homogeneity and cost it could be achieved.

Entertaining the idea that the methodology was not available, a possible solution would always have the minimum amount of hops (for the same reasons described beforehand) and use a technology that the personnel responsible for maintaining the network is comfortable with. This would minimise maintenance cost and work. Weighing in cost, LoRaWAN would be the best solution since it does not require subscription to a provider and can be entirely deployed as a private network. If cost was not a problem then NB-IoT

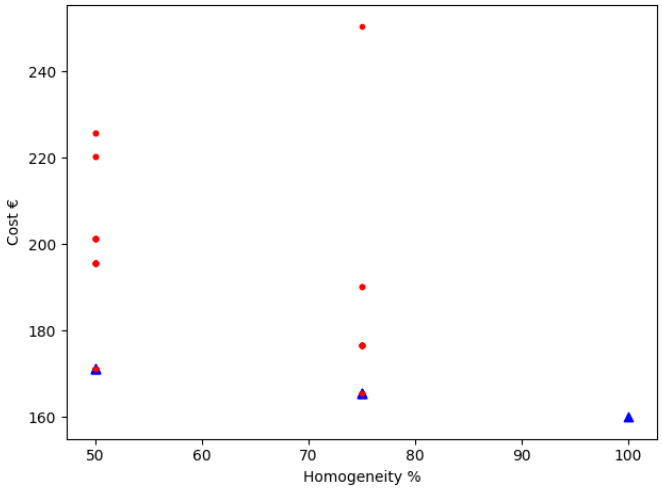


Figure 5.6: Cost in function of homogeneity.

would be the obvious choice since it grants good quality of service, large coverage and good throughput.

Extrapolating the analysis done to Figure 5.8c to other solutions generated by the methodology, it can be argued that the methodology does indeed provide solutions that seem justifiable and fit for practical use after deployment.

Discussion

Compiling the results from every use case used to evaluate the methodology it can be extrapolated that, on average, the proposed methodology can reduce the generated solution space up to 92%. The bigger starting solution spaces is the more it gets reduced. This can be backed when looking at the six, seven and seven (laxer requirements) use cases: the percentage of reduction in solution space rises with the generated solution space. This means that the bigger the generate solution space the smaller the optimal solution space is, which reinforces the effectiveness of the methodology.

Although the optimal solution set gets reduced drastically, in networks with many nodes it is not possible for the network designer to manually review every single solution. In terms of effectiveness, the methodology is successful. It provides optimal and satisfiable solutions no matter how big the original solution set is.

Even though the optimal solution set can be big, the Pareto Frontier is normally composed by few points. Follows that although all graphs are different, it can happen that their evaluation according to the criteria is the same, specially in bigger networks. Some graphs share the same rating and, therefore, the same place in the graph. Choosing a point based on its criteria ratings can help further diminish the size of the solution set.

5.3.2 Performance

The tool was executed 10 times on each use case's setup file. In the same train of thought as the evaluation done in Section 5.3.1, this Section will first evaluate the generic use cases in terms of performance and lastly the Aquamon use case.

The average execution time for each use case can be seen in Table 5.2.

Table 5.2: Runtimes of each execution for each use case (in seconds).

Execution #	5 nodes	6 nodes	7 nodes	7 nodes (lax)
1	0.003	0.094	59.881	92.261
2	0.003	0.094	59.978	92.127
3	0.003	0.094	72.61	91.302
4	0.003	0.094	66.14	90.59
5	0.003	0.094	59.839	90.612
6	0.003	0.094	59.602	88.598
7	0.003	0.094	62.37	91.415
8	0.003	0.094	65.72	92.737
9	0.003	0.094	66.92	88.336
10	0.003	0.094	65.49	88.128
Average	0.003	0.094	63.855	90.61

Reviewing the execution times shown in Table 5.2 it is very evident the sudden exponential climb in execution time from the six node network to the seven node one. The first two use cases (five and six node networks) have negligible execution times that do not even reach a full. Figure 5.10 illustrates the relation between network size and execution time. It can also help in visualise the difference in execution time from the six node to the seven node use case. By comparing the times between the original seven node network and the one with less strict requirements it is clear that shorter distances between nodes and laxer requirements impact the runtime performance. This is expected due to the increased solution space laxer requirements promote.

The Aquamon use case features a five node network. As seen in Figure 5.2, five node networks have a negligible execution time, which also stands for this particular use case. The execution time for the Aquamon use case averages 0.002 seconds.

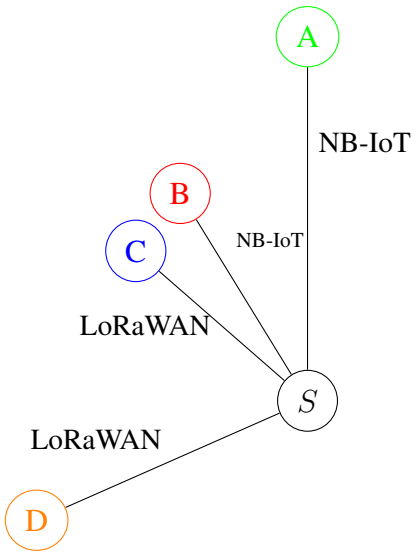
To note that, as mentioned in Chapter 4, the tool is single-threaded and the methodology procedures take a brute force approach, thus, the performance of the tool could be improved further. In Chapter 6 these constraints will be briefly discussed. Nevertheless, the runtime executions remained within a satisfiable time-frame, therefore it can be argued that the methodology can be positively evaluated in terms of performance.

5.4 Summary

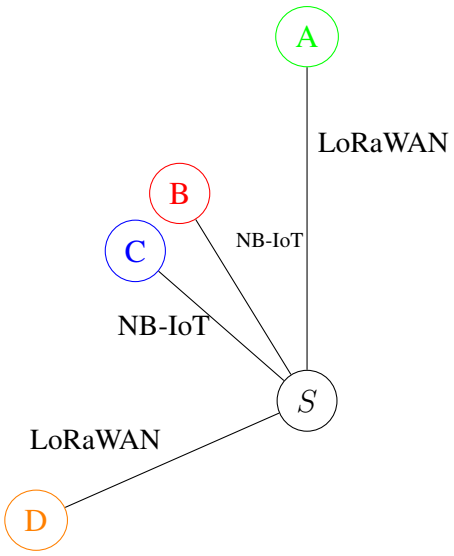
In this Chapter the evaluation objectives (effectiveness and performance) and the methods for evaluating the methodology were defined. Several use cases were used but in particular a real world use case. The Aquamon project was introduced and was used as a test bed for the proposed methodology. Its diversity in environments and the necessity to deploy a 5 node WSN for the purpose of tidal movement study and the retrieval of water samples provides the methodology with a great test of its viability.

Finally, the results outputted by the methodology were evaluated. A short rationalisation about the results was done to understand if the results were justified or not.

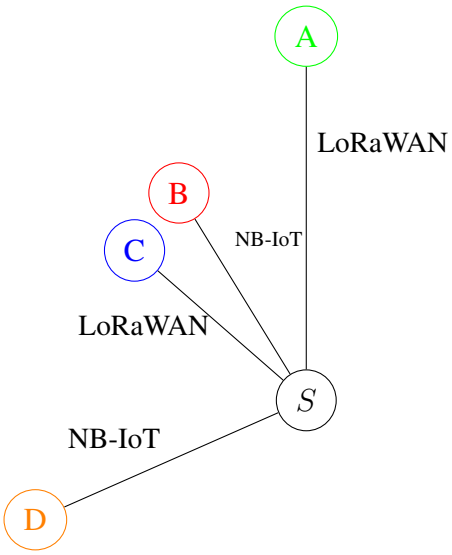
In the following and final Chapter the main open issues of the methodology are listed and briefly discussed.



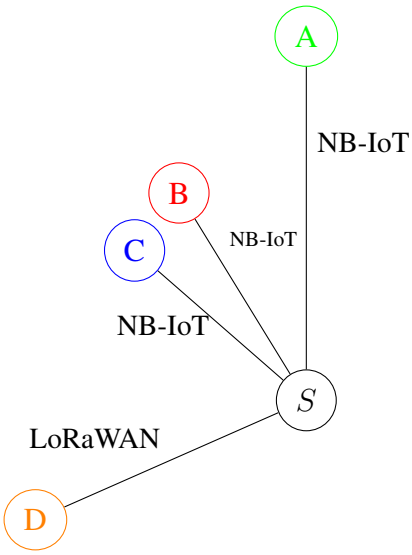
(a) Optimal graph with rating $(50, 171, 1)$.



(b) Optimal graph with rating $(50, 171, 1)$.

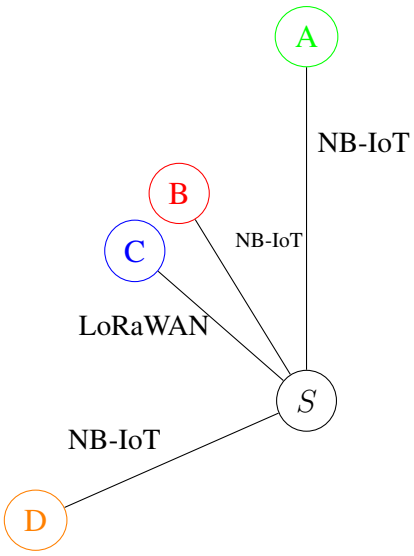


(c) Optimal graph with rating $(50, 171, 1)$.

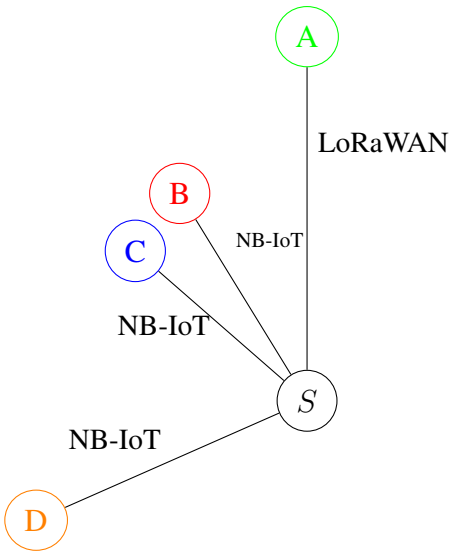


(d) Optimal graph with rating $(75, 176.5, 1)$.

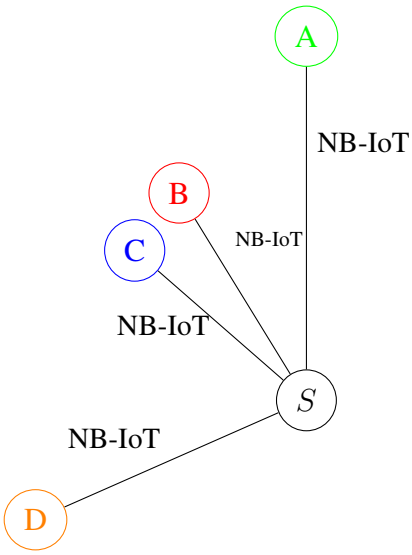
Figure 5.7: Examples of graphs according to the criteria rating.



(a) Optimal graph with rating $(75, 165.5, 1)$.



(b) Optimal graph with rating $(75, 165.5, 1)$.



(c) Optimal graph with rating $(100, 160, 1)$.

Figure 5.8: Examples of graphs according to the criteria rating.

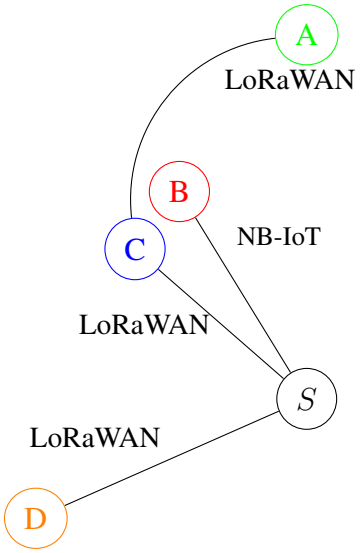


Figure 5.9: Sub-optimal graph with rating $(75, 176.5, 1.3)$.

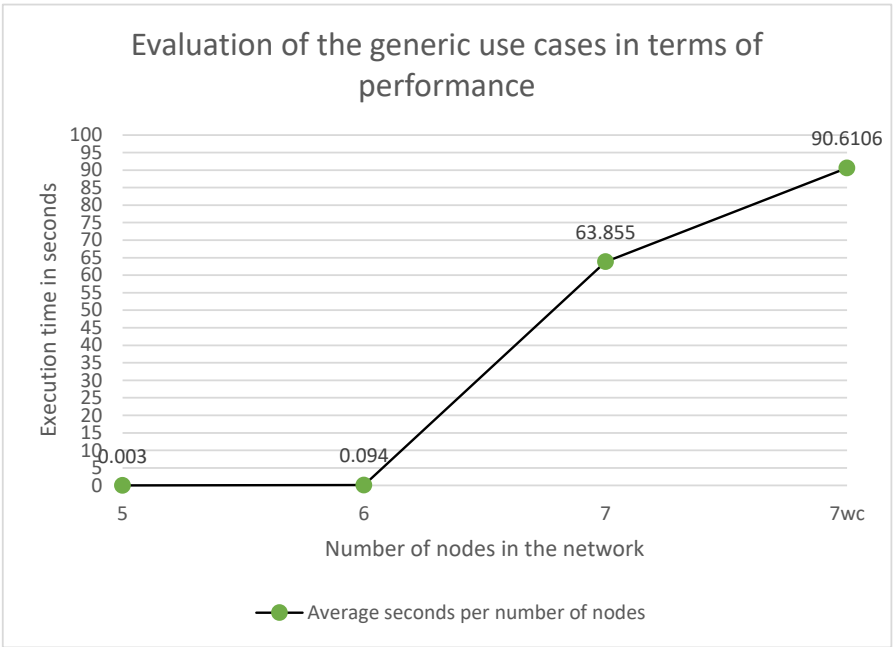


Figure 5.10: Average execution time of each generic use case (in seconds).

Chapter 6

Open issues

As seen in Chapter 5, the proposed automated methodology is already capable of taking nodes of a network as input and output a viable set of solution with the topology and the wireless technologies to use.

Despite this, there are still many issues and improvements that can be made to the methodology that can inspire future work.

Firstly, the methodology currently has no way of dealing with mobile nodes. It also does not have a way of distinguishing technologies that have different architectures. NB-IoT and LoRaWAN are an example, both these technologies differ in their architecture and the methodology, at this point, does not have the means to store that information as well as use it.

Having a balance between a completely automated methodology and one where the network designer has complete control is a strong point of this methodology. But it can also be a double edged sword. The way the methodology imposes the vertex requirements onto each edge is done in a trivial way and not yet based on physical models to support a good representation of reality. This is a big issue as the final solution space is dependent on the linking process and the pruning of edges.

One of the main concerns is the scalability of the methodology. Currently a brute force method is employed. Future work should look into attributing weight to each edge based on its requirements and use algorithms present in literature to solve the optimisation problem in a non brute force way. The reason why this was not done is because defining the weight for the edge is a complex procedure and it fell out of scope from this thesis.

Finally, to complement and reinforce the solutions the methodology provides, the methodology should be part of a deployment cycle. In short, the network designer would benefit in creating a deployment cycle where the methodology would be provide the solutions, these solutions would then be tested for their viability in the real world and afterwards, according to the practical results, the methodology should be applied again with the feedback received from the practical tests.

Chapter 7

Conclusion

The first objective of this thesis was the study of the new spectrum of wireless technologies called LPWA. Chapter 2 provides a comprehensive description of each technology as well as a comparison between each of the technologies, thus, the first objective of the thesis was achieved.

The final objective is to provide a solution for the several problems identified in Section 3.1 when it comes to the selection of a wireless technology for a WSN. The methodology created comes in a form of a four step methodology and introduces the novelty of using graphs as a way of translating, analysing and outputting a solution. It removes non-technical aspects from the decision making and it can be applied to any use case with any considered technologies and defined requirements. But most importantly, the methodology outputs solution space where every single solution satisfies the application and functional requirements and a single (or set) technology is chosen. Therefore, the second objective was also reached.

This work not only provided an automated methodology, it did it in a way that could accommodate and give relevance to the network designer's wishes. A further contribution of this methodology is that it achieves a sweet spot between full automation and full human control (the current paradigm). Up to the point of criteria definition the solutions in the solution space are sure to comply to every requirement defined up to that point. The criteria allows the network designer to specify which proprieties and aspects he deems relevant for his current use case. Our automated process works in such a way that, in its final stage, allows the network designer to choose in which criteria to base his final solution in.

The methodology was evaluated according to two metrics: effectiveness and performance. Meaning, the capacity to reduce the solution space down to a small enough size that the network designer can analyse the solutions and the overall runtime of the methodology when applied to a use case, respectively. According to the result discussion and analysis done in Section 5.3, it was concluded that the methodology passed both metrics.

Bibliography

[1] Sigfox. Sigfox whitepaper. Technical report, Sigfox, 2017.

[2] LoRa Alliance. Lorawan a technical overview of lora and lorawan. Technical report, Huawei.

[3] M Saravanan, Arindam Das, and Vishakh Iyer. Smart water grid management using lpwan iot technology. In *Global Internet of Things Summit (GIoTS), 2017*, pages 1–6. IEEE, 2017.

[4] Zulhani Rasin and Mohd Rizal Abdullah. Water quality monitoring system using zigbee based wireless sensor network. *International Journal of Engineering & Technology*, 9(10):24–28, 2009.

[5] Tomás Robles, Ramón Alcarria, Diego Martín de Andrés, Mariano Navarro, Rodrigo Calero, Sofía Iglesias, and Manuel López. An iot based reference architecture for smart water management processes. *JoWUA*, 6(1):4–23, 2015.

[6] Ronan Mac Ruairí, Mark T Keane, and Gerry Coleman. A wireless sensor network application requirements taxonomy. In *2008 Second International Conference on Sensor Technologies and Applications (sensorcomm 2008)*, pages 209–216. IEEE, 2008.

[7] Soledad Escolar Díaz, Jesús Carretero Pérez, Alejandro Calderón Mateos, Maria-Cristina Marinescu, and Borja Bergua Guerra. A novel methodology for the monitoring of the agricultural production process based on wireless sensor networks. *Computers and electronics in agriculture*, 76(2):252–265, 2011.

[8] Frederic Vannieuwenborg, Sofie Verbrugge, and Didier Colle. Choosing iot-connectivity? a guiding methodology based on functional characteristics and economic considerations. *Transactions on Emerging Telecommunications Technologies*, 29(5):e3308, 2018.

[9] Orange. Orange iot and lpwa connectivity. Technical report, Orange.

[10] Joseph Finnegan and Stephen Brown. A comparative survey of lpwa networking. *arXiv preprint arXiv:1802.04222*, 2018.

[11] Jiming Chen, Kang Hu, Qi Wang, Yuyi Sun, Zhiguo Shi, and Shibo He. Narrowband internet of things: Implementations and applications. *IEEE Internet of Things Journal*, 4(6):2309–2314, 2017.

[12] Rashmi Sharan Sinha, Yiqiao Wei, and Seung-Hoon Hwang. A survey on lpwa technology: Lora and nb-iot. *Ict Express*, 3(1):14–21, 2017.

[13] Yiming Miao, Wei Li, Daxin Tian, M Shamim Hossain, and Mohammed F Alhamid. Narrowband internet of things: Simulation and modeling. *IEEE Internet of Things Journal*, 5(4):2304–2314, 2018.

[14] Mads Lauridsen, Huan Nguyen, Benny Vejlgaard, István Z Kovács, Preben Mogensen, and Mads Sorensen. Coverage comparison of gprs, nb-iot, lora, and sigfox in a 7800 km^2 area. In *Vehicular Technology Conference (VTC Spring), 2017 IEEE 85th*, pages 1–5. IEEE, 2017.

[15] Benny Vejlgaard, Mads Lauridsen, Huan Nguyen, István Z Kovács, Preben Mogensen, and Mads Sorensen. Coverage and capacity analysis of sigfox, lora, gprs, and nb-iot. In *Proceedings of the 2017 IEEE 85th Vehicular Technology Conference (VTC Spring), Sydney, Australia*, pages 4–7, 2017.

[16] Matthias Herlich and Ferdinand von Tüllenburg. Introduction to narrowband communication.

[17] Sigfox. Sigfox security whitepaper. Technical report, Sigfox.

[18] Ferran Adelantado, Xavier Vilajosana, Pere Tuset-Peiro, Borja Martinez, Joan Melia-Segui, and Thomas Watteyne. Understanding the limits of lorawan. *IEEE Communications magazine*, 55(9):34–40, 2017.

[19] José Dias and António Grilo. Lorawan multi-hop uplink extension. *Procedia computer science*, 130:424–431, 2018.

[20] Sergey Andreev, Olga Galinina, Alexander Pyattaev, Mikhail Gerasimenko, Tuomas Tirronen, Johan Torsner, Joachim Sachs, Mischa Dohler, and Yevgeni Koucheryavy. Understanding the iot connectivity landscape: a contemporary m2m radio technology roadmap. *IEEE Communications Magazine*, 53(9):32–40, 2015.

[21] Dae-Man Han and Jae-Hyun Lim. Design and implementation of smart home energy management systems based on zigbee. *IEEE Transactions on Consumer Electronics*, 56(3), 2010.

[22] *ZigBee Specification*.

[23] *ZigBee PRO specification*.

[24] Kais Mekki, Eddy Bajic, Frederic Chaxel, and Fernand Meyer. A comparative study of lpwan technologies for large-scale iot deployment. *ICT Express*, 2018.

[25] Niels Lohmann.

[26] Matthew J Woodruff and Jon Herman.

[27]

[28] Huawei. Huawei whitepaper. Technical report, Huawei.

[29] Marco Centenaro, Lorenzo Vangelista, Andrea Zanella, and Michele Zorzi. Long-range communications in unlicensed bands: The rising stars in the iot and smart city scenarios. *IEEE Wireless Communications*, 23(5):60–67, 2016.

[30] Wen Li and Sami Kara. Methodology for monitoring manufacturing environment by using wireless sensor networks (wsn) and the internet of things (iot). *Procedia CIRP*, 61:323–328, 2017.