A wireless mesh network for smart metering

Geelen, D.J.M.

Award date:
2012

A Wireless Mesh Network for Smart Metering

D.J.M. (Daniël) Geelen

October 21, 2011

Technische Universiteit
Eindhoven
University of Technology

/ department of mathematics and computer science
System Architecture and Networking Group

MSc Thesis

A Wireless Mesh Network for Smart Metering

Author:
D.J.M. (Daniël) Geelen

Academic Supervisor:
prof.dr. A. (Antonio) Liotta

Industrial Supervisors:
G. (Gert) van Kempen
F. (Frans) van Hoogstraten

Eindhoven, October 21, 2011

Abstract

Worldwide there has been increasing interest over the past few years for so called ‘Smart Meters’, in academia, governments and in industry. Such smart-metering systems need a way to reliably and cost efficiently communicate the collected data to the backoffice for analysis. Several competing technologies exist and are in use world wide. Mesh-networks have been the winning technology in the USA and Australia for the past two years, and are gaining interest in Europe at he moment due to their reduced costs and increased reliability. In this thesis we present and evaluate a real-life implementation of a new routing protocol for use in smart-metering mesh-network grids. The routing protocol we present is designed to meet both technological constraints and legislative requirements as posed by the application area in mind. Our evaluation of the protocol is based on real-world experience and data collected from real-world devices, in combination with simulation studies of the protocol. Our evaluation shows that the protocol is a robust, reliable solution for communicating collected data in difficult scenarios, showing great resilience against both bit-errors and node-failures.

Preface

This master's thesis describes the work done on my graduation project at the Eindhoven University of Technology. I began my studies at the TU/e in 2004, and for much of my studies I was working on my bachelor and master programs in parallel. I only officially received my bachelor's degree in Technische Informatica (BTI) earlier this year, which has now quickly been followed by a master's degree in Computer Science and Engineering (MSCSE).

The project was carried out under the aegis of the System Architecture and Networking (SAN) group, which is an expertise area within the Department of Mathematics and Computer Science. The work involved an Industrial Collaboration between the university and Smart Dutch, a Dutch technology company. Smart Dutch specialises in the development of technologies for the smart grid, and in particular on RF-based smart-metering solutions. The smart grid will take on an increasingly important role, ensuring the reliable supply of energy to our homes, offices and factories. I believe the smart grid to be of critical importance to securing the future of energy distribution, playing a vital role in the integration of emerging green energy technologies.

First of all I would like to thank professor Antonio Liotta, who took on the role of both graduation supervisor and graduation tutor. His guidance proved invaluable over the course of the project, and especially in the final leg of the project his input helped tremendously in shaping up my thesis. I also would like to extend my gratitude to Gert van Kempen, Managing Director of Smart Dutch, for his advice, ideas, and encouragements. Further I want to acknowledge Frans van Hoogstraten, Development Engineer at Smart Dutch, for his assistance and support.

Finally I would like to thank my family—in particular my mother, my father, step-father and and step-mother, and my two younger brothers—and my friends, whom have supported me not only over the course of this project, but for the whole of my studies. Thank you, I could not have done it without your support.

Daniël Geelen
December 1, 2011

Contents

List of Figures

List of Tables

Listings

Chapter 1

Introduction

Energy, particularly electricity and gas, is something that we have grown accustomed to having around. Nearly every piece of household equipment nowadays needs electricity to function, and modern society would hardly function without it. Electricity has become something which is always at our disposal when we need it, so much so that we hardly ever stop to think about how it gets to our homes, offices, and factories. However recently awareness of where our energy is produced is increasing, and topics such as energy efficiency are on every one's mind, either because of monetary reasons, or out of concern for the environment. Electrical companies share these concerns, too, and invest heavily both in green-energy as well as in ways to reduce costs. Currently, once it has been generated at a power-plant, energy is transported to its destination through the electrical grid. The electrical grid is a large legacy network, and as new technologies emerge they give rise to problems which show the limitations of the current electrical grid.

The generation and transportation of energy is a complicated problem. Power generation and consumption are not constant throughout the day and fluctuate with the daily rhythm of our lives. Traditionally grid operators try to anticipate when extra power generation capacity is needed by watching for signals such as rapid drops in the voltage-levels and by taking into account our daily habits and major events. A certain level of power is always needed and has typically been provided by a power-plant which is capable of generating power efficiently at low cost over a longer period. Such a plant has high start-up and shut-down costs, and takes a long time to switch on or off (e.g. coal or nuclear). When more capacity is needed quickly, a second type of plant which can be turned on and off quickly and cheaply such as gas-turbines is used. These however are more expensive to operate. Consequently the price of energy also fluctuates slightly over the course of a day, to account for generation costs. However, many sources of renewable energy such as solar and wind do not fall in either category, and their production capacity at any given moment is beyond the control of the grid operator. To further complicate matters, these forms of energy generation may be installed by individual consumers, and in effect provide electricity back to the supplier.

To solve these problems grid operators are looking towards a *smart grid* [1]. The smart grid is an effort to leverage modern communications technologies to solve or alleviate these issues. By closely monitoring the individual users and producers of electrical power it is possible to respond to changes in the amount of electricity being used and generated more quickly, and more intelligently. The smart grid needs to enable two-way communications between suppliers and consumers of energy in order to do so. An example of the kind of advantages that can be obtained in this way is a technique known as *peak shaving* [2], which has a goal to reduce energy consumption during the most active periods. By taking advantage of the ability to send information to the subscriber, the grid-operator can influence daily usage patterns, for example by turning the washing-machine on at a time when the load on the network is lowest (or the amount of energy generated greatest). This technique can spread out energy consumption, and in turn, significantly reduce costs. This is especially relevant during peak-hours, when energy costs are at a premium [3]. Peak shaving benefits both the consumer and the grid operator. The latter has to spend less money and effort on short-term energy generation such as via gas-turbines. The customer benefits because, although the total amount of energy he has consumed

is the same, the price of the energy can be lower.

The key to a successful smart grid network is the monitoring and communication infrastructure. Currently standardisation efforts are ongoing world-wide, with a wealth of initiatives lead by the diverse standardisation bodies, industry and governments. The focus is on the exchange of information between parties and devices comprising the smart grid. Of particular importance to the success of the smart grid is a new type of electrical meter, the *smart meter.* Most current meter installations use an electromechanical meter which has to be manually read-out on a regular basis. Newer metering installations may use techniques such as Automated Meter Reading (AMR) or Automatic Meter Management (AMM) to facilitate remote meter reading. The next generation of smart electricity meters can be differentiated from modern metering techniques such as AMR or AMM by the amount of control it gives over the metering devices [4]. Besides offering the ability to read the meter remotely, smart meters are able to manage connected devices with short feedback and response times, as well enabling two-way communication between suppliers and consumers—a key requirement to the smart grid.

1.1 Problem statement

Many technologies are currently proposed, in use, or being evaluated for use in the smart grid and in smart metering communication. Such proposals range from technologies which aim to re-use the existing electrical grid infrastructure as a communications medium (so called power-line communications), to installations wherein each metering device is equipped with a mobile-telephony based communication module (GPRS or GSM). Many more candidate technologies exist, but in this thesis we focus on RF-based mesh networks, a technology which has the potential to affordably connect smart meters in urban areas to the smart grid.

In particular we focus on the analysis and evaluation of a proposed routing protocol for use in a newly developed smart metering system, which employs mesh networking [5] to allow many devices to share a single access-point. The protocol is optimised for deployment in devices with extremely limited hardware capabilities (little more than a kilo-byte of ROM and a few hundred bytes of RAM), and contains a novel implementation of source-routing.

At present the protocol exists only as a reference implementation running on a small number of metering devices in a case-study deployment in the Netherlands. The goal of this research is to study the current protocol implementation and evaluate the protocol's performance in different environments and under different conditions. To this end we need to complete the following tasks:

1. Develop a detailed specification of the protocol, starting from the reference implementation,
2. Design and implement a model of the protocol in a simulator,
3. Use the simulation model to evaluate the protocol's performance under a wide variety of conditions—i.e. beyond the small-scale trial testbeds available today.

The research goal can thus be summarised as follows:

Research goal: *Design and implement a simulation of an existing protocol for the purpose of evaluating the performance and behaviour of the protocol under a range of new, unknown, conditions.*

1.2 Research method

In order to complete the tasks previously listed we take the following approach:

1. We perform a literature study to give an overview of the current state-of-the art in communications technology directed towards smart metering. We briefly discuss and contrast various alternative approaches and routing protocols to the proposed method and protocol in chapter 2.

2. We receive the actual source-code used to build the firmware for the metering nodes in the case-study. We use this source-code to derive a detailed specification for the protocol. We present this specification and the workings of the protocol in chapter 3.

3. Using the specification of the protocol, and the knowledge of the workings of the metering nodes that we have obtained, we design and implement a simulation model in a simulator environment. We present our simulation model in chapter 5.

4. We receive data on the case-study installations, in the form of information on the actual layout of the topology of network, operational data, terrain, etc. This data is analysed to provide insights into the current performance and behaviour of the protocol.

5. We also use this data to implement a model of the existing installations in the simulation. Using the operational data we cross-validate our simulation. We detail the results of the analysis of this data, as well as the validation step in chapter 4.

6. Once the simulation has been shown to be able to successfully and accurately simulate the case-study installations, we design and implement a set of experiments to test the protocol under a variety of environmental conditions. We detail these experiments and simulation models in chapter 5.

7. The results of the experiments need to be analysed and interpreted. We present our experimental results in in chapter 6.

We recognise that no research project is ever fully finished. Invariably there are new questions raised over the course of the work, or new insights lead to new avenues of research which which cannot be immediately be pursued or which are of out of the scope of the current work. We present such future work alongside our conclusions and recommendations in chapter 7.

1.3 Contributions

This work provides the following contributions:

- Development of a formal specification of a protocol which can be used as a reference when implementing the protocol on a new hardware platform, to the share details on the operation of the protocol with potential interested parties, or to be quickly familiarised with the current protocol's implementation.
- Study and characterisation of an existing system, which can be used to identify strengths and weaknesses of the current system.
- Analysis of protocol behaviour and performance, both *a posteriori* using previously collected data which was handed to us for the express purpose of evaluating the protocol, and *a priori* performance analysis through simulation studies.

1.4 Deliverables

A number of tangible results were produced over the course of this project.

- Based on the initial work done on the simulation a conference paper was written, which has since been accepted.

  D. Geelen, G. Van Kempen, F. van Hoogstraten, A. Liotta, "A Wireless Mesh Communication Protocol for Smart-metering," *Proceedings of The Smart Grid: Telecommunication and Power Distribution Synergy,* Maui, Hawaii, USA, January 30 – February 2, 2012 (IEEE)

- Chapter 2 of this thesis forms the basis for a survey paper, which has been accepted.

  A. Liotta, D. Geelen, G. Van Kempen, F. van Hoogstraten, "A Survey on Networks for Smart-metering Systems," in *The International Journal of Pervasive Computing and Communication* (http://www.emeraldinsight.com/products/journals/journals.htm?id=ijpcc).

- The work on implementing the model of the protocol in the simulator lead to a simulation environment which is ready to use. Possible extensions to, or new versions of, the protocol can be developed and tested before deployment in the field.

- To interpret the data from the case-studies and the output of the the simulator, a number of tools were developed to process, visualise and analyse the data.

Chapter 2

Related Work: Networks for smart metering

In this chapter we will give an overview of the current state-of-the-art in communications technology directed towards smart-metering setups. We will start with a brief discussion of various alternative approaches to implementing a smart-metering system, e.g. various different communication mechanisms such as PLC, meshed networks, GPRS access points, etc. Then that we discuss a number of alternative implementations of smart-metering systems based on meshed networks. After that we will discuss various on-going standardisation efforts at the national and international level. Finally we will discuss a protocol which served as the basis for the protocol under discussion in this thesis.

2.1 Smart metering based on telecommunication networks

A number of possible technologies currently exist which can be (and are currently being) used for the communication between meters and a supplier's control centre. Notably there exist power-line communication (PLC), GPRS access points, GSM, telephone (land) lines, broadband Internet access and RF (both licensed and unlicensed radio communications). In this section we briefly describe these various technologies, as well as providing a summary of the respective advantages and disadvantages of each technology.

2.1.1 Power-line communications

Power-line Communications or Power-line Carrier (PLC) [6], is a communication technology which aims to make use of pre-existing electrical infrastructure (i.e. the power grid) to establish a communications network. The communication is relatively short range, typically only used to bridge the distance from the metering installation within a house to the nearest electrical substation. At each metering station a PLC modem is installed, which communicates with a single 'data concentrator' at the substation. At the substation still a different communications technology such as GSM or a GPRS access point is used to finally transport the data to the back office.

The main advantage of PLC is obvious; no additional infrastructure needs to be set up, it is possible to simply reuse the pre-existing electrical cabling that runs from the substation to each house as the communication medium. However PLC does suffer from a number of drawbacks which have hindered its widespread adoption. The first problem is that PLC has a relatively short transmissiuseon range due to high attenuation caused by transformers and interference from electrical appliances (lowering the signal-to-noise ratio) [4,7]. Even very simple electrical equipment such as light dimmers used by consumers may cause significant interference. In other cases it may not be other electrical equipment that shares the power lines that causes interference, sometimes the actual wiring itself maybe of such poor quality (especially in old neighbourhoods) as to render PLC unusable [8]. Further

complications arise due to the fact that regulatory authorities often allot only very limited bandwidth for PLC applications. In the European Union for example the usage of PLC has been standardised, and only a very small spectrum has been reserved for utility communications. The result is that realistic communication bandwidths only reach up to about 4 kbit/s [4]. The reason for these restrictions is in order to ensure that the devices communicating over PLC do not exceed electromagnetic interference (EMI) tolerance levels.

One of the first large scale roll-outs of smart meters using a PLC-based communications system was performed by the Italian *Enel SpA.* Enel SpA was one of the first electrical companies in the European Union to completely replace its entire installation base of electricity meters with remotely operable devices. Enel SpA replaced around 30 million metering devices, requiring an update to their information and communication infrastructure (now based on PLC technology), at an estimated cost of € 2 billion. The roll-out was expected to be very successful, as it was anticipated that the initial investment was earned back within 5 years [9]. However it remains unclear how often readouts occur, how much data is used and what level of remote interaction with the meters is possible. Smart metering requires the ability to quickly communicate with meters in order to achieve good peak shaving. Enel SpA's system is better classified as an AMR or AMM type system.

Similarly to the system under discussion in this thesis, a PLC type system like the one deployed by Enel SpA can be part of or directly connected to the electricity meter, drawing its power directly from the grid. This means that as long as there is power being used (and supplied), the metering device will keep operating. The prototype system would still be able to operate on reserve power and communicate the failure to the back office in the case of a power outage (e.g. using a battery backed or other type of uninterruptible power supply (UPS)). A PLC based system however might not be able to inform the back office of the power failure. If the outage is caused for example by a physical disruption of the power lines, the PLC modem will not be able to maintain communications with either the back office or the data concentrator. The prototype system on the other hand, since it does not rely on any physical connections, would be able to report the power outage to the electrical company, allowing for a quick and timely response. Also similar is the use of a dedicated endpoint to which each metering device connects in order to communicate with the back office. In the case of PLC technology this endpoint has a direct, shared, physical connection to the metering devices, similar to a large token-ring network. This means that the total available bandwidth has to be shared by all connected metering devices which are in range of the data concentrator. As mentioned before the requirements on EMI (limiting the maximum transmission power) and attenuation issues mean that PLC communication bandwidths are very low. The proposed system, using RF communication in the unlicensed spectrum, suffers no such drawbacks and can communicate at much higher speeds, allowing for an increased throughput volume of data and a more real time connection between the home and the back office.

2.1.2 Unlicensed RF communications

Radio frequency communications (RF) refers to a diverse set of wireless transmission technologies, standards and techniques. RF can operate on licensed and unlicensed frequencies. The difference between licensed and unlicensed RF is merely a distinction on the regulatory level, not any technical considerations. In licensed RF frequency-ranges are allocated to specific parties, which are granted exclusive use of their respective allocated bands. It is possible for frequency ranges to be allotted twice or to overlap for different usage, so long as it can be shown that this does not cause interference between the different applications. This could for example be the case when both applications would only be used in mutually exclusive geographical locations with limited broadcasting power. We will consider both licensed and unlicensed radio communications.

RF communications in the context of smart metering are mostly applied in combination with some other communication means. This is because while RF communication is an attractive solution for short to medium range communications, it is not well suited to long-range communications, which would be needed to for example communicate to the back office. In such scenarios wireless RF-communications are used to communicate among metering devices, and a technology such as GPRS or a broadband connection is then used to communicate with the back office. RF components can

be manufactured far cheaper than typical broadband modems or GSM/GPRS components, since they can be far simpler in design and do not require as strict and extensive verification of operating modes because they are not designed to inter-operate with devices (such as mobile phones) and have a far lower radiating power. By using RF communications for local communications (possibly over multiple hops), setting up a small-scale local network, it is possible for many metering devices to share a single (internet) access point. It is this sharing of a single, more expensive, communication device which gives RF communications the possibility to achieve dramatic reductions in deployment and operating costs compared to typical smart-metering solutions. In [4] a comparison between various technologies such as were discussed before (i.e. PLC, GSM, telephone (land) lines) and RF are evaluated on cost per meter. It is shown that, considering the hardware investment, RF is among the cheapest solutions, on par with the use of power-line communication and re-use of a pre-existing broadband connection.

Another advantage of RF communications is that it allows positioning all components of the smart-metering system out of harm's way. The RF module can be integrated into the metering device itself, whereas the external access point can be securely stored in an off-site enclosure (i.e. not on the subscriber's premises, but for example inside a nearby transformer housing). An RF module itself uses less power than other communication modules, such as those used in PLC or GPRS based systems. Low power usage is of great importance in the large scale roll-out of smart meters, as one of the goals of the smart grid is to achieve reduced overall power consumption. The communication latency of RF is lower than PLC, but also lower than that of GPRS when rolled out on a large scale. Thus RF communications are able to offer a more real time connection with the back office, again a key requirement to the smart grid.

Possible disadvantages to RF communications are issues related to the use of certain frequency bands. The preferred solution would be to secure a dedicated frequency band for use in smart-metering applications. However securing a frequency band for a particular use is a long and tedious process, involving many interested parties (e.g. governments, companies, NGOs, etc.), and it may not always be possible to secure a license for a particular frequency band. Since at present no such band has been allocated, currently most implementations of RF communications make use of the unlicensed bands.

The unlicensed spectrum is in principle free-for-all. However there are some rules and regulations set forth by various standards organisations (ITU, ETSI, IETF), which govern the use of these frequency bands. Typical restrictions include: setting limits on the amount of time that may be spend transmitting a signal (i.e. a transmitter has a limited duty cycle); limiting the effective radiated power (ERP) with which the signal may be emitted; and limits on the amount of bandwidth that may be used. However these guidelines may vary greatly between legislative areas, and usually apply only to each individual device. We discuss these legal matters in greater detail in chapter 3. However these restrictions do not govern how any two devices should coordinate their efforts to share the medium, and in fact devices are free to send at any time as long as they respect the aforementioned restrictions. This means that there always remains the possibility for interference with other devices, if for example two devices adopt a policy whereby they send without implementing any carrier sensing [10].

There are many wireless standards which are designed to operate in the unlicensed transmission bands (IEEE 802.15.4 [11], ZigBee [12], ISA100.11a [13], WirelessHART [14], RuBee [15], etc.), of which ZigBee is likely the most well known. The ZigBee standard is designed to operate in the unlicensed 2.4GHz (world wide), 915MHz (Americas) and 868MHz (Europe) ISM bands. The ZigBee standard offers many different features and operating modes, allowing great control over the configuration of a node. It is possible to choose between carrier sense, multiple access/collision avoidance (CSMA/CA) and Guaranteed Time Slots operating modes, whether or not devices should form a beacon-enabled network (where nodes periodically announce their presence to the network) and what type of network should be constructed. ZigBee supports constructing various different network topologies, such as star and tree type networks, as well as generic mesh networks. However whichever network topology is constructed it is always required to have a single master device. The routing protocols used in the ZigBee standard are based on the well known Ad hoc on-demand distance vector (AODV) routing [16]. However this abundance of features comes at a price. Even though the radio modules themselves are (relatively) inexpensive, the *ZigBee Qualification Process* involves

a full validation of the requirements of the physical layer, driving up costs [17].

2.1.3 Licensed RF communications

The GSM standard was originally developed by the European Telecommunications Standards Institute (ETSI) late in the 1980's. Since first publishing of the standard in 1990 improvements and new standards based on the original GSM specification have been released by different parties. Some such new specifications include standards such as General Packet Radio Service (GPRS), CDMA450 (a CDMA2000 type technology), Universal Mobile Telecommunications System (UMTS) and 3GPP Long Term Evolution/LTE Advanced (LTE). These technologies evolved in order to offer greater bandwidth and lower latencies. This development is predominantly driven by consumer-level applications such as SMS, MMS and mobile Internet-access. In the context of smart-metering applications there are a number of ways in which these technologies can be employed. The simplest and most straight forward way is to simply equip each and every metering device with its own dedicated transmitter and receiver. In this way a direct communication between the meter and the central system can be established relatively simply [18]. Another possibility which has been brought up previously in our discussion of PLC is to have only a single GSM-type device which provides a shared connection to which any number of metering devices may connect.

The main advantage of GSM-type connection is ease of use and general availability. These types of connections are in common use in the consumer market (e.g. cell phones, smart phones, and various other devices such as Internet dongles.). Because of the widespread use of mobile phones, GSM networks have excellent coverage in most parts of the world, with the exception of rural areas [4, 8]. Also the components required to connect to GSM networks are readily available and well tested. The downside of this technique when used to set up a direct line of communication is that the communication modules are still relatively expensive compared to alternative communications modules such as those for PLC or RF transmissions, making it too expensive to be used in a large scale roll-out of smart-metering installations. Currently GPRS is the preferred technology mostly due to the smaller scale deployments of (often) individual connections. Additionally usage of the service usually requires a subscription be acquired from a telecommunications company, which adds extra recurring costs [19]. It is worth noting that these recurring costs need not be fixed, as the price plan is determined by the telecommunications company. This dependency is not very attractive for a grid operator with millions of installed devices. Also although the reception may be characterised as 'excellent', reception is usually measured in above-ground situations, where people and mobile phones are located. However the reception in cellars and alike (where the meters are often located) is not guaranteed [4]. A further consideration regarding the coverage area of GSM networks is the target technology. As mentioned before there are many newer technologies which are based on GSM such as, but not limited to, GPRS, UMTS, LTE, etc. Although these technologies do offer huge bandwidths, this is more than required and comes at higher component costs. Moreover newer technologies take time to adopt and implement, and hence the expected coverage for a technologies such as UMTS and LTE can be considerably less than for older technologies such as SMS or GPRS. Further there may be an issue with the time frame for which any such particular technology will be maintained by telecommunications companies. Although SMS has been in heavy use for a long time already, and presumably will be for many years to come, the longevity of other such technologies is yet to be seen. As new standards are ratified in ever quicker succession it is not unthinkable that some of these standards may be decommissioned after only a few years. Choosing such a standard as the basis for a system designed to last for at least 15 years [4] could mean costly replacement equipment needing to be issued long before the originally projected end-of-life date of the smart-metering device. In fact this year was the first year where the number of SMS sent went down, with people more and more starting to use WhatsApp or Ping. It should be noted however that in the case of a large scale roll-out of smart-metering systems based on the same technology it becomes less likely that this technology would be decommissioned, even if the smart-metering network were to become its sole user[1].

[1] The costs however may increase dramatically in this case, as the electrical company will have to bear the costs for maintaining the network on its own.

Ultimately each of the different sub-technologies has their own strengths and weaknesses. For example SMS is very popular and will in all likelihood remain so for quite some time. However this same popularity also presents a weakness. Analysis of the reliability of the SMS service showed that under normal circumstances the average latency of SMS messages increased from several minutes (which is already quite high) to an hour during New Year's eve [20]. The same study also showed the nominal delivery rate for SMS messages to be only about 95%. Both high latencies (minutes) and high failure rates (over 5%) may not meet the requirements of a smart-metering system, which needs to be able to react and respond quickly to events in the network.

Although nearly a quarter of smart-metering installations in China appear to be based on some form of PLC, those installations only use PLC to construct a form of local communication network, i.e. to construct a local network which communicates with a data concentrator or similar access point. In order to actually communicate the data to the back office ('remote communication') most installations (70% of all metering installations) still rely on a GPRS connection [21]. This is similar to the proposed system which is the subject of this thesis; a local communication network connects metering devices within a short range, which then share a single GPRS module to connect to the electrical company's back office.

However there are still ample alternative smart-metering installations in use and being designed in China, as well as in other countries. We note a few pure GSM/GPRS based systems independently developed in different countries [22–24], which are all quite similar to each other in terms of overall design, although they differ greatly in capabilities and implementation details. By 'pure' it is meant that in these systems each individual metering device is equipped with a dedicated GPRS module, there is no other communications technology used, either among the metering devices or between a metering device and the back office. The GPRS network is then used to directly send and receive data and commands to and from the metering device or data centre. This is arguably a less efficient design than the previously discussed design in which many metering devices share a single GPRS module through a PLC or other type of connection such as is the case in the smart-metering protocol which is the topic of this thesis. Because each metering device is equipped with a GPRS module there is a considerable waste of resources, since each (individual) metering device only requires only a small amount of bandwidth to upload the collected data and download any pending commands, usually far less than a typical GPRS connection offers (10–100kbit/s). Further there can be some concern about the number of metering devices that can be expected to use the GPRS network at any one time leading to congestion of the network (as was observed in [20]), unless some form of coordination is used between the metering devices, or a form of back office initiated polling. In contrast the protocol in this thesis, by design, coordinates the nodes in the network to keep the network load low, ensuring a reliable connection.

2.1.4 Telephone (land) lines and (coaxial) cable

Telephone land-lines, or the Public Switched Telephone Network (PSTN), refers to the normal telephone line present in most households. There a number of ways to make use of the presence of a land-line such as dial-up Internet access, ISDN or a broadband connection provided by one of the various DSL-type technologies. These communication technologies provide a means to establish a two-way, point-to-point (Internet) connection between the metering device and the back office. Although the use of a pre-existing communications network for communication in the smart grid might seem natural, a number of shortcomings make them a less than ideal choice.

An initial consideration is the fact that the availability of a telephone line or other suitable communication cable at each meter is a requirement that cannot be always satisfied, especially in developing countries [25]. Even in the developed world more and more people are foregoing a dedicated land-line, favouring to make use of mobile communication means only as a means of saving money made possible thanks to advancements in mobile telephony and mobile Internet.

Next, in the case of small band communication over telephone lines (for example through a dial-up or ISDN type connection) it is not an option to keep the communications line open at all times (the consumer may themselves wish to make a phone call or connect to the Internet). Hence it is necessary

to only make a connection with the back office at certain pre-set times, either scheduled or when a report is due. There are two major downsides to this. Firstly this means that it will not be possible for the metering company to communicate with the metering devices at any given moment, instead it will be necessary to wait until the metering device initiates communication instead. Secondly it means that the device will have to initiate a connection by dialling in. This takes a significant amount of time (again reducing the real-time nature of the system), as well as possibility incurring additional costs associated with the use of the telephone or ISDN connection. Even if this usage is not billed to the subscriber directly, the metering company will need to be billed for the use of the line, which will ultimately be reflected in the pricing plan for the subscriber. Additionally there is still the issue of interference, be it either the metering device interfering with the normal operations of the subscriber's telephone line (which may suddenly be in use when the subscriber needs to use it, for example in case of an emergency), or the metering device finding the line in use by the subscriber.

In the case of a broadband connection dialling typically has to be done only once, and thereafter the connection can typically remain in an always-on state without issue. This is because these types of connections employ a multiplexing technique which allows them to share the phone line or coaxial cable (frequently used to transmit radio and television signals, too) without interfering with normal operations. However in spite of this the use of a pre-existing broadband connection, either over a phone land-line using a DSL-type technology or over coaxial cabling is still impeded by concerns about the sharing of a communication line with the subscriber. Again no such connection may be present.

Even though the broadband penetration in (western) Europe is among the highest in the world, it is still not sufficiently prevalent to assume a broadband connection will be available in every household or for every metering installation. E.g. even in countries with high levels of broadband penetration, adoption is still quite low among the elderly. This is especially true for countries outside of the European Union or in rural areas where it may be difficult to even obtain a broadband subscription. Hence this would entail setting up dedicated broadband connections in the cases where none is present yet. If a broadband connection is already setup, concerns may be raised about the sharing of the connection with the subscriber. For example since the connection is not owned by the metering company, it can exert no control over it. If the subscriber at any time, for any reason, decides to switch Internet Service Provider (ISP) or even cancel his subscription entirely, the connection to the metering device would be disrupted for an undefined period. This again is unacceptable for a service which must ensure a certain level of reliability. Forcing any (particular) broadband connection upon subscribers is neither a suitable option, and may even interfere with an existing broadband connection.

A final hurdle is the fact that, regardless of the choice of the above technologies, a connection needs to be established between the metering device and the broadband modem, which is often not in the same place as the metering device [4]. This would again entail the use of a third means of data transport.

As noted in [25], using the public switched telephone network for remote management and smart-metering purposes is an old proposal. A working system is described in [26] as far back as 1996. The system supports reading of up-to three utility meters (presumably a gas-, water- and electricity-meter), similar to the capabilities of the system for which the protocol under discussion in this thesis was designed, albeit the newer system facilitates connecting a large number of metering devices (up-to 255) through the use of an M-Bus (Meter-Bus, European standards EN13757-2 and EN 13757-4) connection.

Even though the idea to make use of the existing telephone connection on a subscriber's premises is a relatively old-fashioned idea, generally associated with various issues and introducing (some) discomfort for the subscriber. However this has not stopped people from developing new systems based on this technique. We note numerous recent efforts [27–29] in China alone. Unfortunately, due to pay-wall restrictions, we could not evaluate these systems. A brief description for one of the new systems is given in [25]. From this description we can conclude that due to the use of dual-tone multi-frequency (DTMF) signalling for transmission of metering data and commands, the telephone line will still be tied up for the duration of the communication.

2.1.5 Remarks about infrastructure based approaches

In this section we have presented some of the most common technologies used to implement smart-metering systems, highlighting the strengths and weaknesses for each technology and providing a comparison with the proposed system where possible. We have seen that, like the prototype system, most implementations take a dualistic approach, combining two or more technologies to implement a smart-metering system. This is done in order to improve the quality or reliability of the communications system and to reduce the cost of implementing, installing and operating the communications system. We have seen that the most systems tend to use a GSM/GPRS based solution at the data concentrator. This is because this currently seems the most flexible solution. Although it is possible to use many different technologies at the data concentrator, it requires the least amount of (testing and development) effort to implement a uniform solution in all metering installations.

2.2 Smart metering based on mesh networks

Routing protocols can be broadly classified in two categories: *precomputed routing* vs. *on-demand routing* (and, somewhat related, periodical update vs. event-driven update) [30]. In a precomputed routing protocol (proactive routing), routing tables are constructed before they are needed. That is the nodes in the network are always working to maintain an up-to-date view of the network or their immediate neighbourhood, even when the node would otherwise be idle. Some proactive routing protocols specifically select such idle periods to perform updates to the routing tables, so as not to interfere with the normal operations of the node. In an on-demand routing protocol on the other hand, routing information is not gathered until needed, i.e. when a packet from an upper or, possibly, lower layer is received and needs to be forwarded. The related periodical and event-driven update mechanisms describe at what moment a node publishes routing information to the network. Some protocols are continually communicating routing information in order to keep the view of the network current (and hence are always consuming a portion of the available bandwidth), whereas other protocols attempt to reduce the amount of bandwidth consumed by only announcing detected changes in the network's topology (e.g. links dropping, or new nodes joining the network).

From our literature study we can conclude that this division in proactive and reactive routing protocols seems to hold true in the field of RF communications for smart-metering purposes, too. Routing protocols used in smart metering may range from the very-simple ALOHA [31] like approaches such as [32], to intricately complex solutions such as [33], which uses an implementation of *collaborative transmission* as described in [34]. Next we discuss the most relevant approaches.

2.2.1 Gridstream

Gridstream™ is a commercial, complete, end-to-end, smart-metering solution developed by Landis+Gyr [35]. It is complete in the sense that Gridstream is a full suite of products, including metering hardware, a communications system, a software package, and more. The Gridstream Communications RF communications module was recently evaluated by Lichtensteiger et al. [32], via a simulation based on the OMNeT++ simulator. It was found that the system was a reliable solution, able to achieve a greater than 99.8% success rate in obtaining daily meter readings, for a large population of electricity meter endpoints in a utility's service area, at a read-out interval of 7.5 minutes.

The evaluated Gridstream system uses a time-synchronized slotted ALOHA scheme. This is a well known extension to the standard ALOHA broadcasting scheme, which yields a substantial improvement to the performance of the protocol. However the protocol remains based on ALOHA, and the maximum theoretical throughput remains low at only 36.8%. A clever way to improve the throughput of the network, which is employed by the Gridstream system, is by noting that the throughput is calculated per communications channel. Thus by increasing the number of available communications channels, the number of collisions on any one channel is reduced, thus yielding a better overall throughput. In this case the system utilises up-to 240 discrete channels, each of which has a 100kHz bandwidth, for a total bandwidth of 2.4MHz. Each node then uses a frequency hopping sequence

which is determined according to the nodes network identity. In contrast, the system of this thesis works on a single channel of the (already narrow) 868–868.6MHz band. This band provides only a total bandwidth of 600kHz, in addition to other restrictions. While it might technically be possible to employ this subdivision technique, this would yield only a few additional channels, far short of the 240 channels available to the Gridstream system.

The reason for the difference in available bandwidth between the proposed system and Gridstream is that the latter is targeted for use in the United States of America, where it can freely use the ISM bands. Our system targets deployments in the European Union, where regulations heavily restrict the available bandwidth. Another downside to the frequency hopping approach is that the nodes must coordinate the selected frequency among all nodes participating in a transmission. This complicates the design of the routing protocol considerably, since care must be taken that nodes in a forwarding chain can communicate with both the up stream and down stream nodes. Further this scheme requires more advanced (and hence, more expensive) transmission and reception hardware, capable of rapidly switching between many different communication channels.

The Gridstream system uses a geographical-routing protocol. Although [32] states that this ensures that the nodes communicate over the minimum number of hops, in fact no such guarantee can be given. Using a geographical-routing protocol ensures that nodes communicate over the shortest (physical) distance, but there is no guarantee that this is indeed causes the nodes to find an optimal route in terms of number of hops, low latency, high throughput, or other criteria. Consider for example the case where in the direction of the data concentrator there is an area of high attenuation, severely limiting the transmission distance. This might be the case for example if there are (one or more) backyards in between the transmitting node and the destination. This means that both the distance that can be covered per hop, as well as the reliability of the link over that hop, is reduced greatly. In this case it might be more advantageous to use a communication path with lower attenuation, which allows covering more distance per hop at a higher level of reliability. In fairness the effect of attenuation is lessened when transmitting with high transmission power, but again this is not always possible (e.g. either due to hardware restrictions, or due to regulations). In particular this could likely be an issue with the proposed system, which transmits at a mere 10mW.

Additionally, due to the nature of geographical routing, each node needs some way to know its own position (geographical coordinates, i.e. latitude and longitude), as well as the position of the target node. This implies either adding a GPS module, which increases the cost of each metering device and seems wasteful in a system which will spend its working life in a static location, or require extra work on the part of the installation personnel during commissioning. Both solutions are more costly, the former requiring additional hardware and the latter requiring more manual labour, with the added possibility for human error and associated rectification costs. On the other hand the approach of this thesis is fully self configuring, requiring a minimum of configuration, and no work on the part of the installation technician.

2.2.2 Routing Protocol for Low power and Lossy Networks

The Routing Protocol for Low power and Lossy Networks (RPL) is a newly developed protocol, destined for use in IPv6 enabled low-power wireless personal-area networks (i.e. 6lowpan [36] environments). It has been developed by the Internet Engineering Task Force's (IETF) Routing over Low Power and Lossy Networks (ROLL) working group, and is designed to deal with establishing and maintaining connectivity between nodes in Low power and Lossy Networks (LLNs) [37]. RPL is a gradient-routing protocol, and precomputes the routing information for the network only when necessary (i.e. it is a reactive protocol). RPL builds a special type of tree to capture routing information known as a Destination Oriented Directed Acyclic Graph (DODAG). The DODAG for the entire network is constructed from a single starting point which is the central node of the network. During the construction of the DODAG each node is assigned a weighting value, which is later used to make routing decisions. When a packet needs to be routed the node calculates the difference in weighting value between it and each potential next-hop neighbour node, and forwards the packet to the neighbour with the largest gradient. The calculation of weighting values happens according to a weighting function decided upon by

the central node.

As is the protocol has not (specifically) been designed for use in smart-metering scenarios, and it is thus not immediately obvious that RPL is suitable for use in smart-metering deployments. A recent study [38] highlights some of the potential issues with using RPL as the routing protocol in smart-metering (mesh) networks, suggests a number of improvements to the protocol in order to adapt to the different circumstances in smart-metering networks, and provides a performance evaluation of these enhancements. The suggested modifications are designed to improve the protocol's reactions to link loss, making it more robust for use in smart-metering networks which require a higher degree of reliability. The modifications extend RPL with a hybrid periodic/event-driven update scheme. Routing information is collected both upon detecting a loss of connectivity, as well as by periodically scanning for nearby data concentrators on alternate channels. This information is then stored so that a node can immediately switch to a secondary data concentrator without going through the process of route construction.

Compared to the routing protocol proposed in this thesis, RPL is a fairly complex protocol, relying on large data structures to calculate routing information. The nodes must have sufficient storage capacity and computational power to construct and maintain the DODAGs. While routing over the DODAG results in well-structured trees, there is no evidence that this results in superior routes. That is to say, there is no evidence that the routes constructed by the proposed routing protocol are less reliable or more prone to transmission failure. The proposed protocol implicitly constructs high-quality routes, although the result—when plotted—might look less appealing at first glance.

Further the RPL protocol performs regular probes for new routes, which could interfere with ongoing transmissions of nearby nodes. In this way the choice to scan for a new data concentrator when none is found is similar to the approach proposed herein, which only initiates a search for a communication endpoint when necessary. We employ a scheme which coordinates the times at which nodes are allowed to transmit, thus reducing the likelihood that any two nodes transmit simultaneously. RPL takes a more traditional approach: having no such provision each node simply attempts transmission and retries as necessary.

A bigger difference is in the way routing is ultimately performed. RPL is a typical packet-based routing protocol, where routing information for the network is present in all nodes. When a packet is received and needs to be forwarded, each node inspects *its own* routing table to determine which node the packet should be sent to. In the proposed protocol however a source-routing approach is used, cutting down on the amount of state that is required to be kept in each node. With the proposed changes RPL is capable of handling multiple data concentrators in the same network (thus improving redundancy). This is something that the current iteration of the proposed protocol does not yet (fully) support, although support for this feature is already planned for the next version.

2.2.3 IEC Synchronized Wireless Mesh Network

The Israel Electric Company is also investing in improving its infrastructure, with the goal to deploy an Automatic Meter Reading (AMR) system, as the first step towards setting up a full smart-metering network. A case study [33] was commissioned to investigate the feasibility of using a Wireless Mesh Network based on RF communications. A new routing protocol (referred to hereafter as the Synchronized Communication Protocol (SCP)) was developed and subsequently deployed in a field experiment. This showed that both the protocol and hardware functioned "beyond expectations".

The routing protocol is a time division multiple access (TDMA) type protocol, similar in spirit to the proposed protocol. As in the proposed protocol, time is divided in slices sufficiently large to allow any node in the network to communicate with the data concentrator, taking into account the time required to forward each 'message' over a pre-specified maximum number hops. The routing itself is different in that whereas the proposed protocol incorporates some intelligence as to deciding the next-hop destination for each message, SCP relies only on flooding [10]. Normally flooding is not considered as a viable communication technique, or indeed a routing protocol at all. However unlike the proposed protocol, which works like any ordinary wireless transmission, in SCP transmissions rely on a special feature of the hardware, which makes flooding practical.

By making use of the principle of collaborative transmission [34, 39], multiple nodes can be transmitting (forwarding) the same message *simultaneously*. This is done by synchronising the transmitters of all nodes on the bit level, so that all nodes are simultaneously sending the same radio wave. This is possible by the assumption that the communication channel has an "OR"-characteristic [39]. This synchronisation is done at the physical layer because of the strict requirements on synchronisation. Even a small deviation can cause severe interfere at the receiving nodes, leading to reception failure. Because at any given instance all nodes are cooperating to facilitate the communication between a single node (that node whose turn it is to report to the data concentrator), the protocol in fact lacks routing tables [40] on the assumption that nodes only need to communicate with a single pre-determined destination (e.g. the data concentrator). Although the proposed protocol shares this assumption in principle, it does still provide enough routing capabilities to allow nodes to communicate among themselves.

The downside to the approach taken by SCP is that it requires a high degree of cooperation between the hardware and the protocol. In particular the extremely tight synchronisation required by the protocol, coupled with the fact that this synchronisation needs to be implemented at the physical layer in order to be effective, restricts the choice of hardware. Thus in fact the reliance on the specific capabilities of the physical layer drives up the cost for the hardware. Although the protocol described in this thesis also requires a certain degree of time synchronisation between participating nodes, it does not rely on such precise timings. This means that the proposed protocol can be made to work with readily available hardware with relatively little effort, allowing the implementer to choose the most cost-effective solution.

A final distinction that can be made is the difference in who initiates the communication when a meter reading is to be taken. In the proposed protocol, each node can determine its designated communication slot based on its identifier. In SCP meter readings are initiated by the data concentrator, requiring to poll every metering device in sequence. Because of this each meter reading requires both more bandwidth and more time than needed. In [33] no mention is made of any attempt to limit the number of transmissions. Deploying the SCP system in the European Union could lead to regulatory limitations on the use of bandwidth being exceeded when a large amount of metering devices needs to be read. The protocol requires more messages to be send for each meter reading, and the whole network participates in the flooding of each message. Although this does rely on specifics of the amount of traffic, number of nodes, meter-readout frequency and other parameters which are not given, it is a scenario that needs to be taken into account before a large scale international roll-out can be considered. The proposed protocol is designed with these restrictions in mind.

2.2.4 Mesh networks based on ZigBee

Besides solutions being developed in-house [33, 35] or modifications to existing systems [38], there are also many smart-metering systems which aim to use a well established, standardised platform, such as ZigBee. ZigBee is one of the most popular standards for wireless RF communications, and has a large backing in industry. We have previously discussed ZigBee in section 2.1.2, we now briefly discuss two example smart-metering implementations based on ZigBee technology, hereafter referred to as system A [41] and system B [19] respectively. Both present a fully working system which has been developed and evaluated in the lab, using commercially available off-the-shelf (COTS) components.

Both systems make use of ZigBee's capabilities to form mesh topologies, appointing the ZigBee-coordinator device as the data concentrator. System A splits the metering device into two distinct components, similarly to the proposed system. It is assumed that the meter readings which are to be reported to the data concentrator for transmission to the back office are made available by one or more measurement devices. These then communicate with the main module of the metering system, which leverages a ZigBee transceiver-module to transmit data to and from the data concentrator. Each metering device synchronises its internal time to that of the data concentrator when it joins the network. Once synchronised, each metering device periodically transmits its meter readings. It is also possible to request an immediate reading from any meter in the network.

System B also proposes a split between the devices performing the meter readings and the device

responsible for establishing network connectivity, but does so for other reasons. It is suggested that converter devices be placed on existing metering installations as an upgrade. This would enable existing metering infrastructure, such as electromechanical electrical meters, to be easily incorporated in an AMR system. Although this does not constitute a smart-metering system, it is however also possible to connect modern meter-reading devices to the networking device, so it can be employed in a true smart-metering system, too. System B uses the Texas Instruments Z-Stack ZigBee [42] implementation as its basis. It is not entirely clear whether the routing protocol used in system B is one of those included in the Texas Instruments Z-Stack implementation by default or a modification thereof, although there are indications that the latter is the case. From the condensed protocol description given we can gather that it is similar in spirit to RPL (section 2.2.2). That is to say the protocol precomputes routing information by periodic updates, and actual routing occurs along a gradient. The updates occur in the form of broadcasts of node information to all neighbours, which include the node's distances in hops to the data concentrator. This is later used during routing to select a next-hop destination. One potential issue with the protocol for use in smart metering is that when a node is reporting a meter reading, it keeps retransmitting this reading until they it receives an acknowledgement. This means that it is easily possible for a single node to exceed the regulatory limits for use of bandwidth, not to mention to interfere with the operation of the rest of the network. This system, like system A and the proposed system, periodically sends meter readings on the node's initiative (i.e. there is no polling done by the data concentrator). It remains unclear if and how this timing is coordinated among nodes.

Both systems demonstrate the ease with which a ZigBee based communications system can be build. However both systems demonstrate some of the problems that come with relying on ready-built ZigBee technology. System A relies on the default ZigBee routing protocol to perform and scale well when deployed in a large scale smart-metering roll-out, which does not necessarily hold true. System B does demonstrate that it is possible to change the routing protocol, but this obviously does require using both hardware and software which supports this. Although Texas Instruments Z-Stack appears to offer a ZigBee software-stack which allows modifications to the software, this may not be compatible with all ZigBee hardware-modules. Further system B quotes a $5 (USD) price for the ZigBee module alone, which will drive up the price for a complete metering device by at least that amount.

2.2.5 Source Routing

The routing protocol proposed in this thesis belongs to the class of source-routing protocols. Perhaps the most well known and studied of the source-routing protocols is Dynamic Source Routing (DSR) [43, 44]. Because DSR is a well known routing protocol, we will use it to briefly explain the concept of source routing, and the manner in which these principles are applied in the protocol which is the subject of this thesis. An in-depth description of the protocol is included in chapter 3.

The main principle and innovation of source routing is the manner in which the route (or path) which a packet should follow through the network is specified. This method of specifying the route of a packet has long been known, and is even available in the Internet Protocol (IP) as the (optional) Strict Source and Record Route (SSRR) or Loose Source and Record Route (LSRR) options [45]. DSR adapts this technique by making it the primary means of routing, extending it with mechanisms to discover the ID of the destination node and reactions to loss of connectivity, all operating in an on-demand manner [5].

In DSR (or any source-routing protocol) the sending node needs to have all the routing information required to reach the intended recipient node. There are two basic ways in which a routing protocol can ensure that the sending node has this information. The first is by proactively sharing routing information about the whole network among all nodes. This would effectively mean that each node in the network could reconstruct—or devise an equivalent to—any given route, by looking at the source and destination addresses of a packet, and hence provides no apparent advantage over traditional routing. The second approach is to attempt to gather only that information about the network which is required to reach the target node. This can be done by querying neighbouring nodes for routing information to the destination. If necessary the neighbouring nodes propagate the query up in the

network, until eventually the destination node is reached. Both DSR and the proposed protocol take the latter approach, maintaining radio silence until there is data to transmit, at which time a search for the destination is started if no route is yet known.

In order to ensure error-free delivery of packets, some form of acknowledgement is needed in source routing, too. DSR uses a per-hop acknowledgement scheme, where each hop is responsible for ensuring the packet has been forwarded correctly. The proposed protocol uses an end-to-end acknowledgement approach. To prevent having to search for a route every time a packet needs to be transmitted, some way to remember previously discovered routes is required. Both protocols tackle this in a similar way. Instead of maintaining a routing table, nodes maintain a route *cache*. In this cache the nodes maintain a look-up table of previously used routes. In DSR there many ways specified in which a node can add or remove routes from the cache (e.g. when a node 'overhears' a nearby packet transmission, it may include (part of) the route in that packet's own route cache).

DSR has many more features designed to improve the resilience and performance of the protocol, especially in the later revisions of the protocol [44]. Ultimately DSR is a highly complex protocol, capable of providing full Internet connectivity to nodes in the mesh network. However this complexity no doubt comes at significant cost in terms of the amount of work (man-power) involved in implementing and testing the protocol on new wireless devices, the size of the final executable code (requiring more ROM storage) and the amount of RAM required during operation. Although this enables wireless devices to more easily inter-operate with hosts on larger scale networks, in the case of smart metering or even AMR or AMM this additional functionality is neither required nor desired. Smart meters need only to communicate with their controlling station (the data concentrator), which is a single host operated by the utility company, and as we have outlined in section 2.3.4, these resources are scarce in wireless nodes in general, and in a commercial smart-metering solution especially. The protocol in this thesis has been optimised to make the most of what limited hardware resources will be available on low-cost smart-metering devices. We will detail some of these hardware constraints in section 3.1.2.

2.2.6 Remarks about mesh network approaches

We have discussed a number of meshed-networking based smart-metering systems based on RF-communications infrastructure. We have seen that both very simple and very complex solutions have been deployed successfully, either in field tests, in commercial deployments or both. Given that the complexity of the system seems to have little impact on the effectiveness and reliability of the system, it seems obvious to choose the simplest solution. Complexity often comes at a cost, either as an initial investment in the construction of the system or the cost of its constituent components, or during maintenance or extending the functionality of the system.

There is a great deal of similarity in all of the systems discussed, including the approach of this thesis. This is to be expected of course, and the difference is in the details. Many of the presented systems have been designed for a singular purpose, that is to say they have been designed for deployment by a single company or in a single country. This can be a potential impediment to the widespread adoption of a smart-metering product when attempting to market a product on a multinational scale. Great care must be taken on many levels of product development, ranging from ensuring that the selected hardware can operate in the allotted frequency bands, to ensuring that the software observes the rules with respect to the usage of the wireless spectrum.

Finally we have seen that although it is possible to use standardised hardware and software such as IEEE 802.15.4-compliant transceivers and the ZigBee software stack, these too bring with them a set of considerations complicating any design. In fact many solutions in the field or under investigation [46–49] are derived from the approaches reviewed above.

2.3 Standardisation efforts

There is a growing interest in the development and deployment of smart-metering systems in order to realise what is known as the 'smart grid'. One of the key contributing requirements to the

widespread adoption of smart-metering installations and the takeoff of the smart grid is the development of industry-wide standards. This is achieved most effectively when backed by government initiatives. In the following we discuss the on-going standardisation efforts world wide, in the European Union and nationally in the Netherlands. We also discuss some of the other and related standardisation efforts, such as ZigBee and the drive to bring TCP/IP networking and Internet connectivity into all devices, even smart-metering devices.

2.3.1 Worldwide trends

Worldwide there is a vast difference in the level of on-going and planned standardisation. In the United States of America for example, until now "few attempts have been made to develop a written specification consummated with standards agreed upon by members of both coteries, due to lack of government support" [50]. In contrast in the European Union there is, and has been for some time now, an on-going process of government incited standardisation, both on a Union-wide level as well as on the national level.

In the United States of America there has recently been a push from the government to explore and develop interoperability standards for smart-grid systems. Currently the US National Institute of Standards and Technology (NIST) is working on a framework of smart-grid standards, with the first phase (a roadmap towards the specifications of a smart-grid interoperability standard) now completed [51]. However the lack of a comprehensive set of standards has not prevented electrical companies from developing and selling smart-metering systems. Gridstream is but one such example of a commercial offering of a smart-metering system. Interestingly there are also non-utility companies (such as Google, Microsoft and Cisco) which have entered the smart-metering market. The move by industry to develop smart-metering system shows that there is a will and a market for these systems. However the need for standardisation with rigid set of requirements became apparent recently in New Zealand, where a three-year report on the country's smart metering revealed numerous issue with the installed metering devices. The lack of agreed-upon standards was identified as the main cause for the failure of the smart-metering program [52].

The situation in the rest of the Americas is less clear. Brazil participates in the work of the International Electrotechnical Commission (IEC), it is not clear whether this has, or ultimately will, lead to the deployment of smart meters in Brazil. We do not know of any current interest in developing a smart grid in the rest of South America. The situation seems similar in much of the developing world, even though there is the possibility to effectively deploy smart-metering systems even in developing countries, with the benefits outweighing the costs of conventional meter installations [53].

Not only the Americas and European countries are seeking to build a smart grid. In Asia many countries are looking to develop a smart-metering infrastructure. In South Korea for example, authorities have initiated a long term three-phase plan to have a fully operational smart-grid infrastructure by 2030. However there does not seem to be any form of standardisation being included, with the project focusing instead on developing immediately applicable technologies [54]. In China, too, a large scale effort to develop a next-generation electricity distribution system was recently begun, under the "Strengthened Smart Grid" plan. In contrast however to the South Korean approach, the Chinese approach relies on the development of smart-metering standards. This work is being carried out by the State Grid Corporation of China (SGCC), taking into account, and cooperating with, many on-going standardisation efforts, both nationally (NIST in the United States of America, METI in Japan, and several German initiatives) and internationally (IEEE, IEC, ETSI, ITU) [51, 52, 54].

The standardisation work currently underway by the IEC seems to be the most comprehensive attempt at defining a complete set of standards covering all aspects of the smart grid. The collective work of the IEC on the topic is known as the Seamless Integration Architecture (SIA) and encompasses a large set of standardisation efforts for individual components of the smart grid. This ranges from defining the way in which information on customers and usage is exchanged (IEC 61970/61968: Common Information Model (CIM)) to the actual interfacing standard between different smart-metering devices (IEC 60870: Communication and Transport Protocols). The only current disadvantage is that the SIA is still a work-in-progress, hindering its widespread adoption.

2.3.2 European Union regulatory efforts

In the European Union the situation is very similar to the global picture. Where the global picture showed that both on a world-wide scale as well as on the international levels different groups are working on standards, the standardisation efforts in the European Union are also on-going on both a national and on a Union-wide scale. Therefore we will restrict to giving just a brief conceptual overview here.

European Union-wide standardisation efforts are initiated by decree of the European Parliament, under advisory of the European Commission. The most important of these is Directive 2006/32/EC of the European Parliament [55]. Although this directive does not mandate the deployment of smart-metering systems directly, it is of great influence on the decision process of Member States, with most states considering smart metering to be the best suited method to comply with the requirements of the directive [9]. In this way Directive 2006/32/EC is of direct influence on the behaviour and decision making process of the Member States, guiding them into the development of smart-metering solutions. However it can also be construed as a reaction to already on-going smart-metering deployments (Italy) and standardisation initiatives (Germany). In a similar vein there are efforts underway (Mandate CEN/CENELEC M/441 [56]) to standardise smart-metering interfaces in order to facilitate the development of smart grids. The general objective of this mandate is to create European standards that will enable interoperability of utility meters. However the vast majority of these standardisation efforts are focusing on PLC and GPRS technologies and protocols.

2.3.3 The case of The Netherlands

The Netherlands, being a Member State to the European Union, has to comply with European laws and mandates. It is however up to each individual Member State to determine the best way to fulfil these requirements. Further it is also always possible for Member States to strengthen any decision made by the European Parliament. In the Netherlands this has given rise to the Dutch NTA-8130 Directive [57] and Dutch Smart Meter Requirements (DSMR) document [58]. These specify the specific functionality which a smart-metering device should provide. This is foremost a practical description of the requirements most smart meters support, such as the ability to report meter readings to the utility company on demand, and without human intervention, the ability to remotely connect or disconnect customers from the electrical grid, etc. Further it is specified, as in Mandate CEN/CENELEC M/441 or IEC 60870, how metering devices should interconnect.

2.3.4 Smart metering communications over TCP/IP

There is ever present drive in industry as well as from academia to bring IP [45] technologies to wireless sensor networks. However pure IP communications are not often used because of the lack of guarantees this protocol offers. Therefore in order to make communications reliable the TCP protocol [59] is used on top of the IP protocol. However typical implementations of the TCP/IP protocol suite tend to be large and complex. Both the size of the executable code as well as the amount of memory required often reaches into the hundreds of kilobytes. These large quantities of memory (both ROM and RAM) are simply not available to small embedded systems. This makes it infeasible to directly make use of existing TCP implementations such as the popular Berkeley BSD TCP/IP implementation [60] or its derivatives [61–63]. Therefore a number of solutions and implementations have been proposed to bring IP connectivity to the world of small, embedded, wireless devices and the resource constricted environments in which they operate [62–64]. There even exist proposals to enable full IPv6 accessibility on small embedded devices. However as indicated in [64] these implementation often still require multiple tens of kilobytes of ROM and, more importantly, RAM. In fact indications are that the commercial offerings of TCP/IP implementations often outweigh their open counterparts in terms of resource utilisation. Further it can be seen that even protocols such as 6LoWPAN [36], which have been specifically designed for low resource utilisation still have troubles to fit even on commercial sensor nodes [65].

The main perceived advantage of the idea of 'IP everywhere' is that it becomes easier to interconnect devices and to develop applications to run on these devices, e.g. to cut back on the costs of development. However we argue that there is price to pay for ease-of-use. That price is the increase in resources required with increased abstraction, and the reliance on external components (black-box design), wherein one inherits all the flaws and shortcomings of the components, as well as the benefits. We argue that the price one has to pay in terms of additional resources and reliability outweighs the potential gains. Cross layer optimisation techniques in particular have the potential to exceed the savings made by using a ready-made TCP/IP solution. Although more expensive to develop initially, the long term savings on hardware and maintenance costs are significantly reduced. Therefore we concur with the view which is expressed in [61] that although it is certainly possible to make use of TCP/IP with relatively few resources, a superior alternative may be found in the use of a proxy or translation step in a gateway node, transparently translating between TCP/IP and an optimised protocol. The protocol proposed in this thesis will assume a similar translation step to occur at the data-concentrator node.

2.3.5 Remarks about standardisation

Although there are many standards under development, few are currently complete. Many countries specify a road map towards a smart-metering infrastructure or implementation of the smart grid, but often omit directives specifying specific standards. This holds true not only for the European Union or individual nations. Even though in the European Union there is the Mandate CEN/CENELEC M/441, this mandate still leaves portions of the specification of smart-metering devices open. This is similar to the IEC's SIA standard which, although it goes into great depth specifying all aspects of the smart grid and smart-metering infrastructure, also intentionally leaves certain aspects undefined. This is done in order to leave enough space for innovations and vendor specific implementations [51, 54]. This might be necessary because there can be no single universal technique which is equally well suited to every situation, and even within a certain type of technique there are aspects which can be tweaked to gain optimal performance for the intended application.

In particular there is the issue of how devices within a smart-metering installation are to communicate. This is not standardised, because there are many techniques to choose from, each with its unique strengths and weaknesses, which makes it well suited to a particular situation (we have discussed the most commonly suggested approaches in section 2.1). Therefore it is reasonable to assume that smart-metering installations within a particular geographical region (one building, a neighbourhood, or a whole town) will all make use of the same technologies, with all metering devices constructed by the same vendor.

Within the domain of the technique in which we are interested (smart metering over RF communications) there are trade-offs to be made, too. The choice between using standardised hardware and/or software, and creating a proprietary solution is mostly a trade-off between ease-of-construction and costs, both for the hardware, as well as for the software. We have seen that even when standardised hardware is used, it is still required to develop the necessary software (for example the systems discussed in section 2.2.4). Hence it makes sense, both from a marketing perspective as well as from a performance and efficiency perspective, to design and develop alternative approaches, algorithms, and protocols.

2.4 Conclusions

We have presented an overview of a diverse set of technologies which can and are used in the deployment of smart metering systems. We highlighted the strengths and weaknesses of the usage of existing infrastructure (power-line communications, land-lines). Although seemingly advantageous, there are significant drawbacks to these technologies. Licensed RF communications networks are a significant improvement, although they can be more expensive. Costs can be reduced by sharing a relatively expensive data uplink among many inexpensive metering devices. We argue that unlicensed

RF communication networks, in particular when used in mesh networks, are the most suitable candidate technology to realise such an approach. We have reviewed a number of existing and proposed mesh networks and protocols, showing approaches which result in complicated, expensive designs requiring custom hardware to implement protocol features or using a TCP/IP stack which consumes large amounts of resources. Existing smart metering systems make few provisions to comply with on-going with standardisation efforts. The proposed protocol and accompanying system make use of commodity components reducing hardware costs. In this thesis we will show that by using a mesh protocol optimised for use in smart-metering scenarios it is possible to construct and operate standards compliant smart meters while observing strict legal requirements.

Chapter 3

Protocol description

In this chapter we present the protocol as it was used in both the field-experiments (chapter 4), and in the simulation-experiments (chapter 5). Before we present the operational details of the protocol we first discuss what will be required of the protocol (section 3.1.3), and the constraints (sections 3.1.1 and 3.1.2) within which these requirements must be realised. After that we present the workings of the protocol in section 3.2 We conclude this chapter by highlighting some of the distinctive features of the protocol in section 3.3.

3.1 Goals and specifications

The specifications of a protocol entail the translation of the desires of many stake-holders, who each envision one or more goals to be achieved, into a concrete set of requirements. In the case of a routing protocol for use in smart-metering scenarios the stake-holders are legion, ranging from the ultimate users of the smart-metering devices (ordinary households), the utility companies, governments and regulatory bodies, standards organisations, manufacturers, and so on. Since each of the stake-holders has his own interests at heart, the challenge is to strike a balance and find a common ground that all can agree upon.

Thus specifications arise in the form of constraints, requirements and goals. Constraints are things that *can* or *can not*, or alternatively, *should* or *should not* be done. In this context the former are limitations imposed by the hardware platform, whereas the latter are limitations imposed by regulations and legislation. Both are important considerations in the design of the hardware and software, and indeed in the protocol. We have already touched on some of the regulatory issues in sections 2.1.2 and 2.3. In section 3.1.1 we explain in greater detail the issues at hand, and how they affect the design of the protocol and selection of the hardware. Section 3.1.2 details the limitations of the hardware platform that was used in the implementation of the protocol for the case-studies. We have attempted to condense the set of key requirements for this protocol in section 3.1.3.

3.1.1 Regulatory constraints

As mentioned before in chapter 1, the system under discussion in this thesis's primary deployment area will be the Netherlands, and the European Union thereafter. This deployment area must be taken into consideration in terms of regulatory constraints for a number of reasons. Firstly, as was discussed in section 2.3.2, in the European Union there are efforts underway (Mandate CEN/CENELEC M/441 [56]) to standardize smart-metering interfaces in order to facilitate a large-scale smart-grid. Similar efforts are taking place on a national level in the whole of the European Union. Particularly in the Netherlands there are efforts underway to provide standards to which utility-meters should adhere, such as for example the Dutch NTA-8130 directive or the Dutch Smart Meter Requirements (DSMR) document [58].

However complying with these standards is not sufficient to build a complete smart-metering system, including the protocol(s) used for inter-device communications. This is because these standardisation efforts are mostly focused on interoperability between different layers in the smart-grid. These standards specify the electrical connections and protocols between different components in a smart-meter (a single smart-metering unit may consist of multi-components of different vendors and utility companies, see section 3.1.2). How communications are to be done within a system or layer is left largely unspecified, so these standards have only a limited impact on the the routing and communications protocol. The routing and communications should be sufficiently flexible to ensure that they can (be configured to) accommodate the data that is to be communicated between layers, and provide any guarantees that are required by the standards, e.g. requirements on security or timeliness of data-delivery. Besides these requirements which are specific to smart-metering implementations, also more general restrictions also apply. Both in Europe and America, and indeed all over the world, there exists regulations which impose certain limitations on the use of radio frequencies. For example in the European Union there are regulations which affect all union members, as well as the possibility for individual members to amend these global rules. The European EFIS database[1] provides an overview of legislation governing the use of radio-frequencies on many levels, ranging from international agreements imposed by the International Telecommunication Union (ITU), down to amendments on the national level. Certain frequency bands may have been reserved for use by certain applications (e.g. medical usage, amateur-radio, etc), or may require a licensing agreement before they can be used. There are restrictions on the maximum allowed transmission power or Effective Radiating Power (ERP). There are restrictions on the amount of bandwidth that can be used or the amount of time a device is allowed to actively transmit (the active duty-cycle). The specific regulations and limitations that apply to a specific device thus depend on the frequencies and bandwidth chosen, and on the intended application area. The frequencies and bandwidth requirements in turn are influenced by the application area, and the capabilities of the hardware that is used. Section 3.1.2 details the hardware and its requirements, but for this discussion it is relevant that based on decisions made by the hardware development team, the smart-metering device operates in the 868MHz band.

In particular since the protocol will be developed and tested in the Netherlands it should adhere to European laws and regulations in general, and Dutch laws and regulations in specific. In the selected frequency band of 868.4MHz there are strict limits on the amount of available bandwidth and spectrum usage. This frequency band is part of the $868 - 868.6\text{MHz}$ band which has been allocated for use by 'non-specific' Short-Range Devices (SRDs) under the following conditions: no use is to be made of channel-spacing (however the whole stated frequency band may be used), the device must be limited to a maximum ERP of 25 mW using narrow/wide-band modulation, and the device must operate its transmitter such that it either has a worst-case duty-cycle of less than 1% or implements some form of Listen Before Talk (LBT). The term 'duty-cycle' is a measure of the percentage of time that a device is using its transmitter, and can thus potentially interfere with other users of the same frequency band. The duty-cycle is measured over a rolling one-hour period, and is calculated independently for each device. To calculate the duty-cycle one has to measure the total amount of time a device is transmitting using a one hour sliding window. Thus for any given device, if its transmission behaviour were measured for any period of one full hour, the total time spent transmitting should be strictly less than 1% of the time. Concretely this means that, combined with the given transmission characteristics of the transceiver (which does not support LBT), the protocol has to ensure that none of the participating metering- devices exceeds a 1% duty-cycle, at a maximum Effective Radiated Power (ERP) of 25mW.

3.1.2 Technological constraints

Figure 3.1 shows a prototype smart metering device as used in the field-trial deployments at Goes and Den Bosch (The Netherlands). Internally these metering-devices generally consist of two parts, a measurement station and a meshed transceiver. Often there is also the possibility to connect further measurement stations to the meshed-transceiver, e.g. a water- or a gas-metering unit. There are a number of reasons for splitting the measuring and reporting functionality in this manner.

[1] The EFIS database is available online through http://www.efis.dk/

The primary reason is practical in nature. The actual measurement-devices can be designed such that they provide just the functionality required by the grid-operator. However the reporting functionality needs to operate also outside of the confines of a single house, otherwise it would be impossible to get the measurements to the back-office. This does not mean that the actual device needs to operate under varying conditions, but rather that the communication should handle different operating conditions. For example there may be certain scenarios where the meshed-network approach does not work due to environmental circumstances (rural areas are one such example) and a different communication technology is required (such as a GPRS system for example). Rather than designing a completely new meter, the modular approach allows for easily changing to a different communication method when required. This design encourages competition, both among grid-operators as well as suppliers of communication modules. Thus future upgrades to newer technologies can be made more cost-efficient because one only needs to upgrade part of the metering-device.

Figure 3.1: Smart metering prototype-device.

As was discussed before the choice of hardware is of influence on which laws and regulations the final device must adhere to, and vice-versa. Although the current hardware design comprises only a prototype, this will be used a guideline for the construction of the final device, and thus gives good a base for what to aim for in the final product. An important requirement of the system was that it needs to be very cost efficient, build with widely available components offered by several suppliers, being very robust, reliable and resilient. After careful consideration the hardware-development team opted to adopt the Nordic Semiconductor nRF905 transceiver [66] for field-testing. This RF-transmitter chip is an affordable single-chip solution, capable of transmitting in one of 433MHz, 868MHz or 915MHz bands. Not all of these bands are available or well suited for use by smart-metering applications. Although the 915MHz band could freely be used in the Americas, EU-regulations don't allow it in the European Union, where the 915MHz band has been reserved for 'land mobile' use (i.e. GSM, DECT, emergency services, etc). The 433MHz band was rejected by the hardware development team, also for regulation related reasons. In the European Union, and indeed in the Netherlands, regulations governing the the 433MHz band are very liberal, allowing for many applications to make almost unrestricted use of this band. As a result this frequency band is crowded, with some applications transmitting for prolonged periods of time. There were concerns that this would lead to poor performance of the smart-metering system due to interference caused by such external influences. Hence the 868MHz was chosen not only because the other two bands were either unavailable or deemed unsuited, but because of the stricter regulations. In this frequency band the Nordic RF transmitter transmits on an approximately 200kHz wide channel, centred around 868.4MHz. This allows it to

transmit at an effective data-rate of at most 50kbit/s, or just over 6kB/s.

Another restriction of the RF-chip is that it is required that the chip knows in advance the length of the packet that is to be received. This implies that all packets used in the network need to have the same length, e.g. the same number of bytes. In the case of the nRF905 chip, the maximum packet-length is fixed at 32bytes by the hardware. Thus the protocol has only a very limited amount of space for carrying both protocol headers and data. For example TCP/IP adds 40 bytes of overhead and thus would not fit in the buffers used by the RF-chip for the reception and transmisson of packets.

Besides the RF-communications chip, the metering device is also equipped with a micro-controller which is used to perform regular measurements and run the routing protocol, as well as a variety of other tasks (coordination of external devices, controlling the RF-communications chip, etc). The specific micro-controller used in the prototype is the Microchip PIC16F887-I/PT[2]. This microcontroller offers 14kB of ROM storage, and just 368B of RAM. As discussed in section 2.3.4, even the smallest IP-suites typically require multiple tens of kilobytes of available ROM space, and several kilobytes of RAM. Clearly the prototype, and later possibly the final device will have far less resources available, again highlighting the need for a new routing protocol specifically targeted to smart-metering devices.

3.1.3 Requirements

The most important requirements that the protocol should adhere to and implement arise from the legal-, regulatory- and technological-constraints, as well as the requirements of the various stake-holders. Here we give the most import requirements that we have identified.

- The protocol should first and foremost adhere to the legal constraints, in order to be marketable. The most important regulatory constraint with respect to the routing protocol is the limitation on the amount of bandwidth that is available, and the total time that a metering-device is allowed to transmit (i.e. the active duty-cycle). In the given application, the metering device will attempt to report its own measurements at most 4 times per hour, thus any single node will only need to perform 4 short transmission bursts per hour. Therefor the routing protocol is the primary source of transmissions, and thus it is the routing protocol which should ensure that none of the participating metering-devices exceeds the stated duty-cycle.

- The protocol should enable two-way communication with the utility's back-office in order to facilitate the reporting of energy and *reverse-energy*[3], as well as gas- and water-readings from the different meters to the utility's back-office. Also the protocol should enable the delivery of notifications to the meters, e.g. to inform the meters and other connected devices of changes in prices of the delivered services. This will require the protocol to enable two-way communication between the utility's back-office and the metering-devices.

- The protocol should be self-configuring as much as possible, and should not require expert knowledge to setup. Minimal preparations of the protocol for each device (for example setting an identifier for each device) are allowed, but should be performed beforehand on an off-site location or by the manufacturer. This is in order to facilitate the installation of the metering devices by unskilled labor. After installation of a metering-device, the protocol setup should be completely automatic.

[2] http://www.microchip.com/wwwproducts/Devices.aspx?dDocName=en026561

[3] Reverse-energy refers to energy generated at the metering site, e.g. energy from solar cells or wind power.

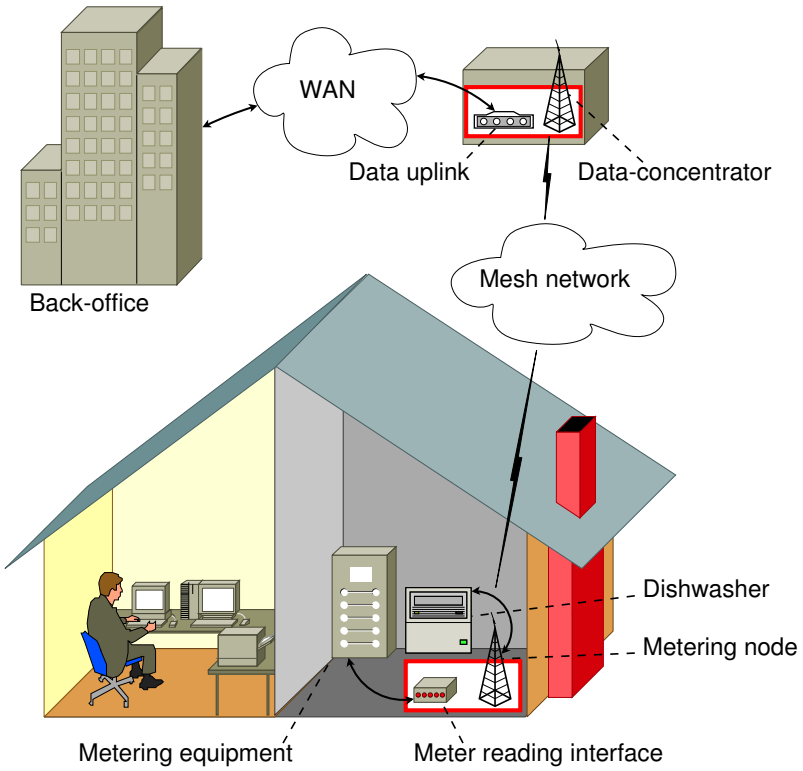


Figure 3.2: Overview of the of the system. This figure shows the place of the system in the smart grid infrastructure. In this figure the system is represented by the components in the red rectangles. The data-concentrator node provides a means for the nodes to connect to the back-office through the mesh network. Metering equipment and devices at the consumer's premises are connected to the metering node.

3.2 Protocol description

We give a full description of the protocol in sections 3.2.1 to 3.2.3. We review the most important aspects and phases of the protocol in sections 3.2.4 and 3.2.5.

3.2.1 Packet format

In the protocol there are two types of nodes, namely 'metering' nodes, and 'concentrator' nodes. Both nodes operate in almost the same way as far as most of the communication protocol is concerned. There are some minor differences in operation which are detailed below. Metering nodes are nodes which are attached to a gas-, water- and/or electrical meter. These nodes take measurements and can be considered the source-nodes for the network. Concentrator nodes are nodes which are connected to an internet connection, for example though a GPRS uplink. These collect the measurements for transmission to the back-office, and can be considered as the sink-nodes in the network. All nodes can operate as an intermediate node in multi-hop communication.

The protocol tries to keep as much intelligence in the source-nodes as possible. This means that there is no polling of the nodes done by the concentrator, in contrast to for example [33]. Instead the metering-nodes themselves initiate communication on certain preset times, and lasts for a brief moment only. That is to say each node has its own communication *slot*. The communication for each node is for the most part restricted to it's slot (we discuss out-of-slot communication later). When

the nodes are installed no special attention is paid to any relation between their node-id and their physical positioning. This means that even in the event that two nodes lose time synchronisation, it is less likely that their communications will overlap, since they are likely to be in different areas of the network.

The protocol uses a proprietary packet format which is optimised for use in meshed communications among a limited number of nodes, each which require only a small amount of data to be delivered. In Figure 3.3 the packet structure is depicted. Each field has been annotated with a short-hand, the meaning of wich, and the intended use of the field are detailed in Table 3.1. Each packet is exactly 32 bytes long, regardless of whether or not those bytes are actually fully used. This is due to a limitation of the RF-receiver, as was mentioned prior in section 3.1.2. The reason for this limitation is due to on-chip CRC verification of the packet for proper reception. The RF-chip does not notify the micro-controller of a reception unless the packet's CRC is correct. In order for the RF-chip to be able to calculate the CRC, it is required that the length of a packet is known beforehand.

Figure 3.3: Layout of bytes in a packet.

Short-hand	Bytes	Meaning	Function
PT	1	Packet type	Type of the packet. For example a search or a direct communication.
OID	1	Origin ID	ID of the sending node.
DID	1	Destination ID	ID of the destination node.
N	1	Number of hops	The maximum distance (in hops) a packet is allowed to travel.
PID	1	Packet ID	ID of the packet. Used to detect duplicate packets.
RFL	1	Route flag(s)	Array of bit-flags.
RN0-4	5	Route node	IDs of intermediate nodes on a multi-hop path.
RSV	3	Reserved	Reserved for future use.
P0-17	18	Payload	Data to be transmitted. For example the Electricity measurement.

Table 3.1: Fields in a packet.

The protocol imposes a 14 byte overhead, leaving 18 bytes free for data communications. Of these 14 bytes three are currently not used and reserved for future use, or they might be dropped in a future version if no longer needed. Thus the true overhead for the protocol is just 11 bytes. For comparison, a raw 802.15.4 header is at least 9 bytes[4], but at that size does not offer distinction of packet type, hop limitation, or routing information. When 6LoWPAN is also used (necessary for routing), this increases the size of the header by 7 bytes [36], for a total of 16 bytes.

The first field in the packet is the PT field, and is used to indicates the type of packet. At present the protocol defines only the following 3 packet types:

1. PT_DIR, used for *direct communication*,
2. PT_SCH, used when initiating a *search* for a node, and
3. PT_MESH, used to send data via *meshed communication*.

Additionally each packet has a corresponding acknowledgement packet type. The data format as used in this protocol is easier to parse, requiring less (no) complex data-structures, a minimum of memory, no compression algorithms and no unpacking. This makes it faster to implement on new hardware platforms, and easier to debug and verify correct operation. Further, because all packets are

[4] http://lia.deis.unibo.it/research/WANDA/architecture/amipv6.html#ipv6

of equal size, transmission times for each packet are predictable. Calculating the required bandwidth is also more straightforward, since each packet takes a fixed amount of bandwidth, regardless of the contents of the payload field `P0-17`. The only potential drawback to this packet format is that, in order to communicate with individual nodes in the network from outside (e.g. over the Internet), it will be required to install a conversion layer at the concentrator.

3.2.2 Handling and routing packets

In order to communicate with and route packets to other nodes, many routing protocols maintain a routing table. Source-routing is different in that instead of relying on each node in the network to know where to send a packet, only the sending node needs to maintain this information. To this end nodes do not keep a routing table in the traditional sense, instead each node keeps a record of the nodes with which it has communicated in the (recent) past, and the route that was used for that communication. This *route-cache* is only used when communication is required with a node outside of direct radio range, and there is a need to use meshed communication [5]. When there is no need to use meshing for communication this is called *direct communication*, and is used as described in sections 3.2.4 and 3.2.5.

When a node wants to communicate with another node, for example in order to transmit it's measurements to the concentrator, it first checks its routing table to see if it knows a route to the intended target node. If a route is known then that route is used for the communication. If the node does not have a route stored for the destination node there are three possible reasons for this: 1) the node has just been turned on and does not yet know any routes, 2) the route that the node was using has failed, or 3) the destination node is within radio range and no route is required for communication. In this case the node initiates an attempt at direct communication. If the node is not able to communicate directly with its target node, or the node does not receive a timely reply by using the stored route, its next step will be to initiate a search. If the node finds a route to its target node with the search, it will store this in its routing table, replacing the previously stored route. It will then proceed to communicate normally, using multi-hop meshed communication. If the search-process fails to discover a route to the target node, the current attempt at communication is aborted. The next time a packet needs to be transmitted (e.g. at the next scheduled measurement interval) the node will start with a fresh attempt at communication, i.e. it 'forgets' the failed attempt at communication.

Upon reception of a packet, nodes process the packet to determine whether this must be forwarded, handled locally, or discarded. This process is depicted in figure 3.4. The node starts by checking whether or not this packet is intended for it, i.e. whether or not its node-id equals the packet's `DID`. If the packet is intended for the node, there are different possible scenarios:

- If there is a direct communication, the node processes the packet and immediately sends an acknowledgement back.
- If it is a search, the node enters a waiting phase. During this phase further searches are ignored. This ensures that potential duplicate search packets which arrive via a different route are ignored. When the waiting time is over an acknowledgement is sent back via the route contained in the packet.
- If it is a mesh data packet, the node processes the packet, and then enters a waiting phase, similar to the waiting phase for a search (see above). When the waiting phase is over an acknowledgement is sent back via the route contained in the packet.

On the other hand, if the packet is not intended for the node, there are different possibilities:

- If the packet is an attempt at direct communication (`PT_DIR`), the packet is discarded and no further action is taken.
- If the packet is a search (`PT_SCH`), the node will attempt to forward the search, provided that (a) it did not initiate the search itself (`OID` $\neq$ node-id), (b) it has not forwarded this search before. A node can determine this by inspecting the packet's route-list, (c) propagating the search does

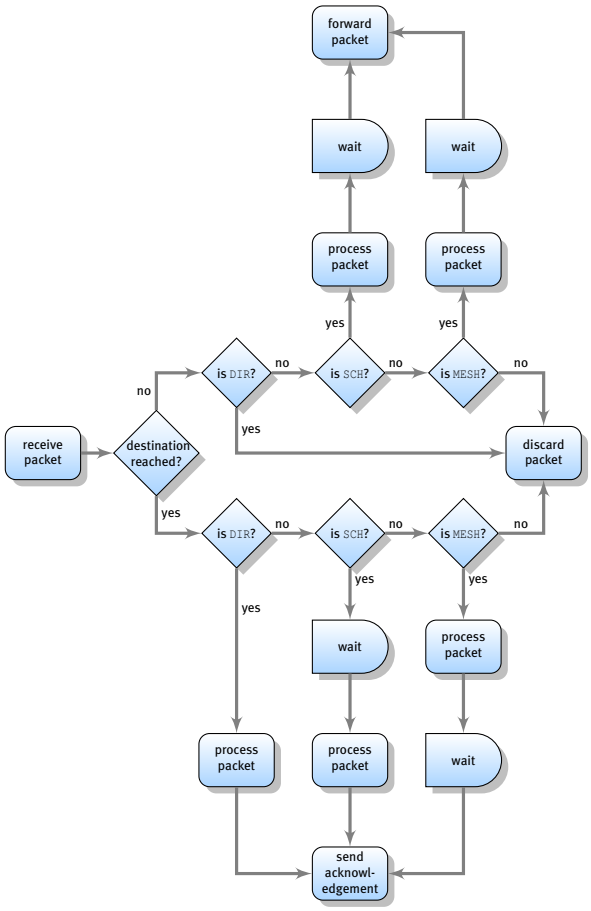


Figure 3.4: Decision process upon receiving a packet.

not cause the search to exceed the specified hoplimit, and (d) it is not already in the process of forwarding another packet. In the current implementation a node can forward at most one packet at a time, no queueing of packets is implemented. If any of these conditions do not hold the packet is discarded and no further action is taken.

- If the packet is any other type of packet, and the node's id is present in the packet's route-list but the corresponding route-flag is not set, the packet is forwarded. If the node's id is not present in the route-list, or the corresponding route-flag is already set, then the packet is discarded and no further action is taken.

3.2.3 Forwarding of searches and meshed-data

Forwarding a packet in this protocol is usually a very simple operation. Once a route has been established it requires only a single bit to be flipped for each packet. Direct communication packets and their acknowledgements do not require any house-keeping to be done and can be processed immediately. Search packets (but not the acknowledgements to a search) require some more work before the packet can be retransmitted.

This is possible by the assumption that routes will have a fixed maximum length of at most 5 intermediate nodes, or 6 hops. In fact the initial protocol specifications allowed for up-to 8 intermediate nodes (header-bytes 12–14, now marked as 'reserved'). However during the case study it was determined that a maximum distance of at most 5 intermediate nodes, or 6 hops, was enough to cover a total distance of up-to 125 meters. In the case-study the average maximum transmission distance—

the *average maximum transmission distance* means to take the average over all nodes' observed maximum transmission distance—was approximately 46 metres, with an average transmission distance of almost 29 metres. Due the random nature of forwarding, the average transmission distance is lower than the average maximum distance transmission distance a node may be capable of. Thus it should be possible to bridge distances of $9 \times 29\text{m} = 260\text{m}$, up-to $9 \times 46\text{m} = 414\text{m}$, so between 250 and 400 metres. Hence in areas with sufficiently dense distribution of nodes paths of up-to 9 hops should be enough, as this gives a range which should be sufficient to reliably cover an area up-to 0.5km^2 around the concentrator node, provided that a sufficient number of nodes is available.

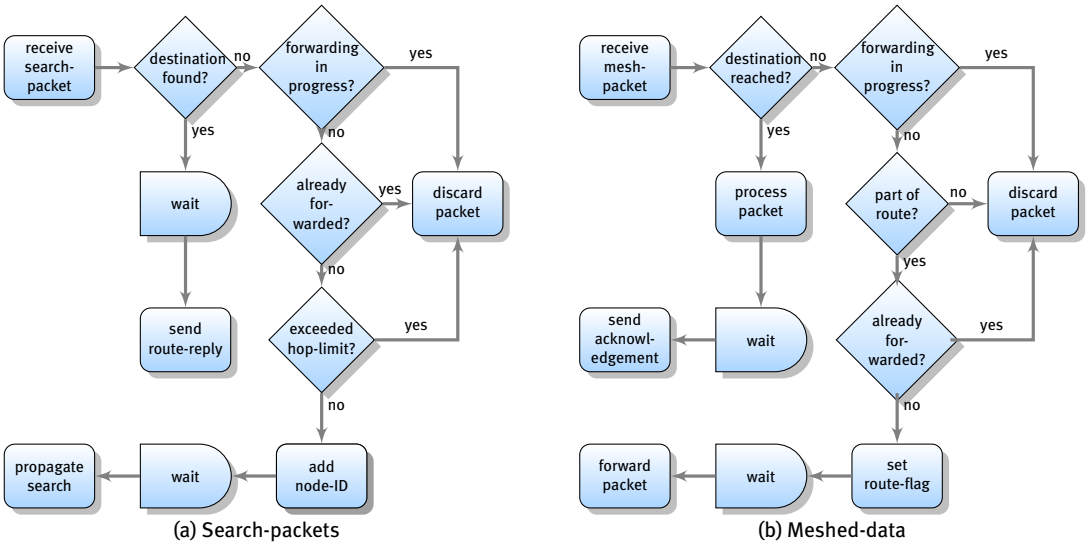


Figure 3.5: Forwarding different types of packets.

As stated, forwarding a packet requires only a single bit flip. Although a node will need to calculate which bit to flip, this procedure is straightforward. Figure 3.5 shows the over-all process for forwarding search- and mesh-type packets. As can be seen the high-level desciption of the procedure for forwarding a search-type packet (figure 3.5a) is actually quite similar to the procedure for forwarding a mesh-type packet (figure 3.5b). The main differences are in the way in which a node checks whether it needs to handle the packet, and the specifics of each step. Packet bytes 7–11 (`RN0`–`RN4`) hold the route that a packet has (in the case of a search) or will travel (all other packet types). These fields should be filled from lowest to highest. Any position which is not filled should be set to be an invalid node-id. During the search-process each forwarding node inserts its own node-id in the first unused slot. A node can decide whether or not a search will exceed the hop-limit imposed for a packet by counting the number of filled positions in the route-list. Now when a packet is to be routed to a destination node, the sending node fills in the route that the packet is to take in the route-list, and clears the route-flag field, `RF`. When a node receives a packet that is neither a search packet, nor a direct communication packet (i.e. a packet which is to be routed), it will inspect the route-list to see if its node-id is included in the route. If this is the case it checks the route-flags field, `RF`. Each bit in `RF` corresponds to a node-id entry in the route-list, with the following meaning. Bit i of `RF` is set iff this packet has been forwarded by the node at position i in `RN0-5`. If the bit corresponding to the index at which the node's id was stored in `RN0-5` is unset, the bit is flipped and the packet is then scheduled for transmission. Otherwise the packet is discarded.

In figure 3.6 the forwarding process is illustrated using an example. This shows how different nodes react to the reception of a mesh-type packet. In this example two nodes (not depicted) are using mesh-type communication. One step of the forwarding process is detailed. Assuming nodes `A`, `P`, `Q` and `B` are part of a previously established route between the two communicating nodes, we can show how every node processes the forwarding request. Nodes `R` and `S` serve to help illustrate the different states that a node may be in that cause it to fail to forward a packet. Suppose that node

A has received the forwarding request from node Q in a prior step (note that the nodes in figure 3.6 are not placed in any particular order). Node A has determined that it needs to forward the packet, and does so. Each node starts processing the forwarding request as soon as it it received. Node S does not forward the packet because it already has a packet in its transmission queue waiting to be send. Node R could potentially forward the packet as it could be in range of B, but since it is not in the packet's hop-list it does not. Node Q does not forward the packet because it has determined that it has already done so in the past by inspecting the packet's route-flags – note that this holds for node A too, but we omit the processing steps going on in node A for clarity. In the end, the only node which is now eligible to forward the packet is node P, and it does so after a short delay. This process repeats as many times as necessary to reach the packet's destination (which could be, for example, node B).

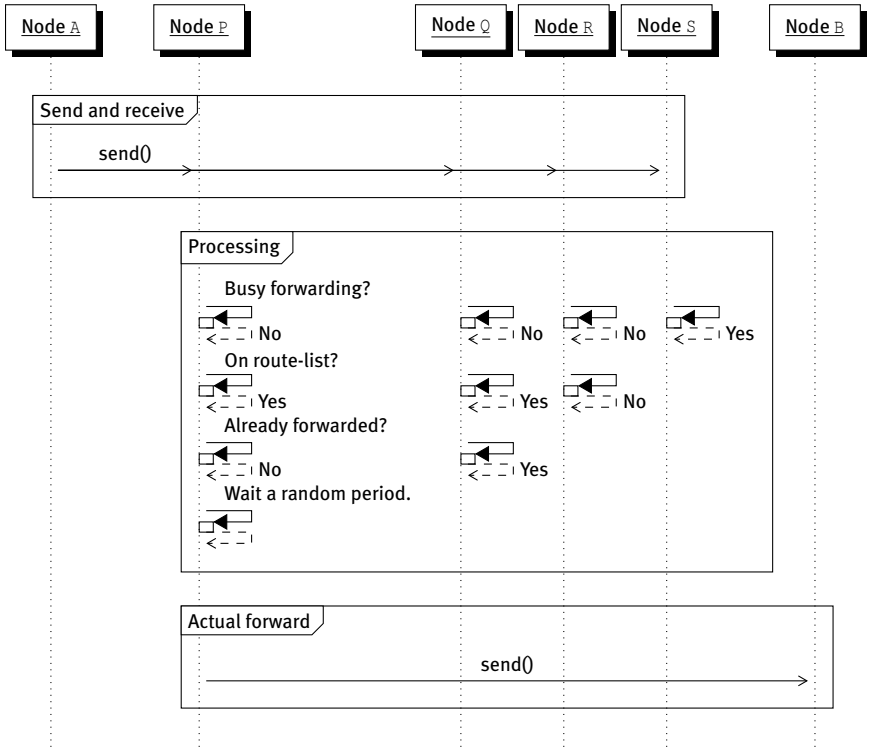


Figure 3.6: Node A forwards a packet on behalf of another node.

Before forwarding a packet, nodes always wait for a short, random, amount of time. This delay is a random delay between 0 and 56 ms, in 8ms increments, i.e. the delay is always of the form $i \cdot 8\text{ms}$, for some random i in the range 0 to 7. By including a small random delay between forwarded messages the protocol avoids excessive collisions between nodes attempting to forward search packets. Similarly the node which is the target of a search will wait a short amount of time before replying to the search with an acknowledgement, in order to allow for the forwarding of search-packets to die-out.

3.2.4 Initial behaviour

As per the requirements, the protocol is fully self configuring and starts operating as soon as the power to the node is turned on.

The process of finding a concentrator node is a straightforward search process which uses an increasing search radius to minimise the overall length of communication-paths. There are currently three different search distances defined; 0, 3 and 5 intermediate hops. Nodes start out with attempting to reach the concentrator node directly, i.e. without using any intermediate hops. In order to account for lost packets this is attempted up-to three times, with a 250ms time-out inbetween. If no

reply is received before the time-out of the last attempt expires the node initiates a controlled flooding search in order to obtain a route for use with meshed communication. The controlled flood starts with a maximum search distance of three intermediate hops. Searches use a time-out which is dependent on the number of hops that the packet is allowed to travel. Typically the timeout is set to 0.9s for a three hop search distance, and 1.5s for a 5 hop distance. If still no reply is received before the time-out expires the node attempts a second search with three-hop distance. If this second search also does not lead to route-discovery the node attempts a single search with a 5-hop distance.

3.2.5 Steady state

Over time all nodes will have acquired a route to the concentrator node, either through a direct connection or via a meshed communication route. Once a node has found a route (any route) to the concentrator node it will store this route for future communications. Note that only routes for use in meshed-communications are cached. It is not necessary to cache the route for direct communications because in the case that a node does not have a route to the concentrator node in its cache it will always act as if it were just powered on, and start its communications by attempting communicating directly with the concentrator node before proceeding with any search.

When a node wishes to communicate with the concentrator node it will first search its route-cache for a suitable route, and if a route is found it will use this to initiate a meshed-communication with the concentrator node. Sending a packet via meshing is attempted up-to three times, with the same time-out values as are used when searching for the concentrator node. If for any reason the meshed communication fails three times in a row the node will drop that route from its cache and give up on transmitting the current packet. The next packet to be transmitted however will need to find a new communication path, and goes through the process described in section 3.2.4.

3.3 Distinguishing features

Finally we conclude this description by briefly reiterate some of the unique features of the protocol.

3.3.1 Quick and easy

The protocol has a simple packet structure, allowing for easy and fast parsing of the packet's headers. We have shown that discovering nodes and forwarding mesh-packets is accomplished through just a few checks required and a minimum on manipulations of the packet's data-structures. This means that even on slow micro-processors the handling of packet headers is unlikely to become a bottleneck. In turn, it is easier to debug any issues that may occur during realisation of the protocol, since it is simple to follow the actions of individual nodes by using a packet-sniffer to capture packets, since all knowledge required to parse and interpret a packet is present within that packet.

3.3.2 Good things in small packages

Besides the protocol's simplicity there is one more feature which makes it well suited for use in microcontrollers and embedded hardware. The protocol requires only a very small amount of free RAM for operation. This allows to equip hardware nodes with very small amounts of (relatively) expensive RAM, cutting back on hardware costs. In the case of the device and protocol as used in the case-studies the total RAM consumption of the software stack running the nodes is just 345 bytes[5]. This includes memory required for communicating with the metering-equipment. Of these 345 bytes the vast majority is used for various communication buffers. E.g. the seven most memory consuming buffers already account for 193 bytes of memory. This includes the transceiver's send and receive buffers which are both 32 bytes long.

[5] Assessment based on assembly output of the CC5X compiler for the PICmicro family.

The reason for this low memory requirement is that the protocol does not require that nodes maintain any state or store information about ongoing meshed-communications in which they participate. That is there are no routing tables being maintained by the nodes or concentrator node, nor do nodes keep any historical record of past or ongoing communications. Instead all of the information necessary for communication is stored within the packets themselves. Thus the memory space necessary for holding the send and receive buffers doubles as a temporary storage location for the current state. Upon receiving a packet a node will inspect the packet to see how it needs to be handled and act accordingly. For example a packet is eligible for forwarding if it has the correct type (e.g. an acknowledgement to a search or a mesh-type packet) and has not yet been forwarded by the node.

3.3.3 Redundant replies

Redundant replies can be useful, and provide an additional level of redundancy. In traditional source-routing protocols, a strict following of the hops listed in the route recorded in the packet is enforced. This means that even though the packet is received by many nodes, some of which are potentially on the forwarding list, only one node can forward a packet at a time. In contrast, in the proposed protocol any node which is part of the route can forward the packet, out of order. This enables the protocol to make use of the times at which the conditions for transmission and reception are better to forward packets more quickly, by transmitting them over greater distances, while still maintaining the same route in case conditions deteriorate again. This is possible because the destination will ignore any duplicate packet (as in most other routing-protocols), and because of the random waiting periods in-between transmissions. Figure 3.7 illustrates both (one possible way) how this process gives rise to duplicate packets, as well as and how these are handled. Suppose for the sake of example that

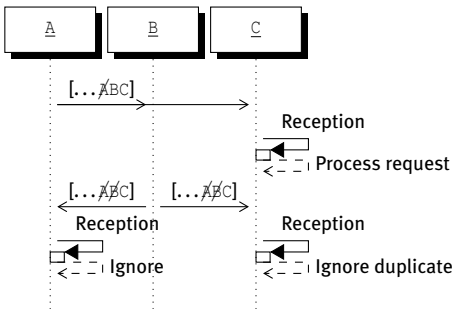


Figure 3.7: Handling of duplicates.

nodes A, B, and C are the last three nodes in a route. When node A forwards the packet, it is in range of the destination C, too. Thus C can immediately begin processing the packet. Node B, which has also received the packet and is on the hop-list, needs to wait a random, short amount of time before it can forward the packet again. Even if the waiting time was zero, node C would still be able to discard the packet as a duplicate. When node B forwards the packet to C, C identifies the transmission as a duplicate and discards the packet.

Chapter 4

Field experiments

Prior to the work presented in this thesis, the proposed routing-protocol was already undergoing field-testing in two different geographical areas. From these field-experiments operational-data was extracted, which was used to analyse the behaviour of the protocol, and verify the accuracy of the implementation of the simulation. In this chapter we detail the work done on the analysis of the extracted operational-data. We first give an overview of the environments used and the challenges presented in sections 4.1.1 and 4.1.2. Thereafter we present our analysis of the operational data in section 4.2. We describe how the operational-data was obtained in section 4.2.1, and the analyses we have performed on this data in sections 4.2.2 and 4.2.3. Finally we present our conclusions in section 4.3.

4.1 Case studies

The field experiments were part of a case-study commissioned by three out of the four largest Dutch grid operators. In the case-study, real-life installations of smart-meters were placed in two neighbourhoods in different cities in the Netherlands. The locations were chosen to be representative of typical expected deployment scenarios. The first scenario is a large apartment-building situated in Den Bosch's 'Binnenstad' neighbourhood. Figure 4.1 shows the building as it appears in Google Earth. The second location represents a typical sub-urban environment, situated in the 'Goese Polder' neighbourhood in Goes. An overhead view of the area is presented in figure 4.2.

4.1.1 Worst case scenario (Den Bosch)

The apartment-building has a somewhat angular U-shape. The metering-devices in this building are located in the basements-boxes of the apartments. Several apartments share one basement-box, so there are a number of areas where potentially many metering-devices are concentrated in a small space. Typically in this scenario around 10 to 12 metering-devices are placed in each basement-box. In total 97 metering-devices were used in this case-study. Note that not all apartments in this apartment-building took part in the case-study. As can be seen in figure 4.3 only about one third of the apartments participated in the case-study. Hence the area covered by the metering-devices in this installation is smaller than the actual lot size of the apartment-building, at an area of approximately 90×40m. Because this particular apartment-building is of an older type of construction it features thick, strong, re-enforced concrete walls which support the structure of the building and act as separating walls between the different basement-boxes. There are two main things which are tested in this scenario. Firstly because of the combination of thick, re-enforced concrete and soil separating the metering-devices this is considered a good test of the penetrating power of the transceiver. Secondly the many metering-devices placed in a small area tests the interference among the metering-devices themselves. This is considered the worst-case scenario which the metering-devices are expected to encounter in a real-world deployment, and which they should be capable of handling.

Figure 4.1: Part of the 'Binnenstad' in Den Bosch.

Figure 4.2: Part of the 'Goese Polder' in Goes.

4.1.2 Typical sub-urband environment (Goes)

In the second scenario the meters are spread across a larger geographical area measuring approximately 167×89m. Here each metering-device was placed at an individual location (i.e. one metering-device per house). The average distance between two neighbouring devices is relatively small, between 5 to 6 meters or 9 to 12 meters, depending on where the houses are placed in relation to each-other, and within each house the metering-device is placed. Whereas the scenario in Den Bosch was picked as the worst-case urban scenario, the scenario in Goes represents the average sub-urban environment that the metering-devices are expected to be deployed in. It has various distinctly sub-urban features such as the placement of, and distances between, houses. The houses featured are mostly semidetached houses. These represent a middle-ground between terraced housing and detached housing. Further features include a street crossing with metering-devices placed in houses on opposite sides of the street, as well as a stretch of gardens containing large quantities of trees and various shrubbery. One unusual feature of this scenario, and an other reason for choosing this particular area, is that a large television tower operates in the vicinity. At the request of the grid-operator this location was choosen to test whether such antennae would interfere with the system (since e.g.

COAX partially uses the same frequencies).

Both the street and the gardens pose unique challenges for the metering-devices. Whereas the gardens present an area of relatively high but constant attenuation, the attenuation levels for transmission crossing the street may vary both in the short and in the long term. Consider for example the difference in attenuation between cars which are parked along the street during the evening and night, which present a source of attenuation which is present or absent for long periods of time, versus a lorry or van which is simply passing through during the course of the day and presents a fleeting source of interference.

4.2 Field experiment results

As the field results were conducted prior to our involvement we can only provide an a posteriori analysis of the results of these experiments. We first give a short description of the process of how the operational data was collected over the course of the field trials in section 4.2.1. The remainder of this section is then devoted to providing the results of two analyses we have performed on the historical data. In section 4.2.3 we present an analysis of the packet-arrival times and response rates, thus providing an estimate on the average communication latencies in the current network. Thereafter we explore the various communication paths used in the network (section 4.2.2).

4.2.1 Data collection

During the course of the case-study there was ongoing monitoring of the visible end-results of the protocol's operation (i.e. the monitoring of the measurements delivered at the grid-operator's endpoint). Although informative, this monitoring process only gives insight into the final result, and gives little information about what is going on *within* the network of metering-devices. Over the course of the case-study there have been several times when an intermediate inspection was performed on the operation of the protocol. During these inspections an RF-sniffer device was employed to gain insight into various aspects of the operation of the protocol. Using the RF-sniffer device it is possible to capture any packet which is 'in flight' within the RF-sniffer device's antenna range. By analysing this captured traffic it is possible to provide answers to numerous questions regarding the operating efficiency of the metering-devices (e.g. 'Which modules are within reach of each other?' or 'What routes are selected within the network'), and to fine-tune the protocol or fix possible errors. The logs from these have also been used for the construction and verification of the simulation. To illustrate how the data obtained from the RF-sniffer device looked like when we received it, we have included a small section of this data as table 4.1. This data was handed to us in multiple large excel-sheets full of timestamped, raw packet dumps which were captured at a position relatively close to the data-concentrator node using the RF-sniffer device. Therefore these logs give a good indication of what the network looked like from the point of view of the data-concentrator at a given time.

Date	PT	OID	DID	N	PID	RFL	RN0	RN1	RN2	RN3	RN4	RSV	RSV	RSV	P0	P1	P2	P3	P4	P5	P6	P7	P8	P9	P10	P11	P12	P13	P14	P15	P16	P17
03 mrt 17:50:16:125	MSH_3	50	1	3	102	0x01	90	0	0	0	0	0	0	0	0x03	0x00	0x32	0x30	0x37	0x31	0x30	0x30	0x30	0x36	0x30	0x31	0x2E	0x39	0x37	0x30	0x00
03 mrt 17:50:16:453	MSH_A_3	1	50	3	102	0x00	90	0	0	0	0	0	0	0	0x08	0x44	0x00	0x00	0x00	0x00	0x00	0x00	0x00	0x00	0x00	0x00	0x00	0x00	0x00	0x00	0x00	0x00
03 mrt 17:50:16:515	MSH_A_3	1	50	3	102	0x01	90	0	0	0	0	0	0	0	0x08	0x44	0x00	0x00	0x00	0x00	0x00	0x00	0x00	0x00	0x00	0x00	0x00	0x00	0x00	0x00	0x00	0x00
03 mrt 17:50:35:719	SCH_3	52	1	3	19	0x00	56	50	90	0	0	0	0	0	0x00	0x00	0x00	0x00	0x00	0x00	0x00	0x00	0x00	0x00	0x00	0x00	0x00	0x00	0x00	0x00	0x00	0x00
03 mrt 17:50:35:766	SCH_3	52	1	3	19	0x00	40	22	78	0	0	0	0	0	0x00	0x00	0x00	0x00	0x00	0x00	0x00	0x00	0x00	0x00	0x00	0x00	0x00	0x00	0x00	0x00	0x00	0x00
03 mrt 17:50:36:000	SCH_A_3	1	52	3	19	0x00	40	22	0	0	0	0	0	0	0x00	0x00	0x00	0x00	0x00	0x00	0x00	0x00	0x00	0x00	0x00	0x00	0x00	0x00	0x00	0x00	0x00	0x00
03 mrt 17:50:36:093	SCH_A_3	1	52	3	19	0x02	40	22	0	0	0	0	0	0	0x00	0x00	0x00	0x00	0x00	0x00	0x00	0x00	0x00	0x00	0x00	0x00	0x00	0x00	0x00	0x00	0x00	0x00
03 mrt 17:50:36:951	MSH_A_3	1	52	3	19	0x00	40	22	0	0	0	0	0	0	0x00	0x00	0x00	0x00	0x00	0x00	0x00	0x00	0x00	0x00	0x00	0x00	0x00	0x00	0x00	0x00	0x00	0x00
03 mrt 17:50:54:907	MSH_3	54	1	3	38	0x01	84	26	36	0	0	0	0	0	0x03	0x00	0x32	0x30	0x37	0x31	0x30	0x30	0x30	0x32	0x36	0x36	0x33	0x2E	0x30	0x32	0x38	0x00
03 mrt 17:50:55:001	MSH_3	54	1	3	38	0x05	84	26	36	0	0	0	0	0	0x03	0x00	0x32	0x30	0x37	0x31	0x30	0x30	0x30	0x32	0x36	0x36	0x33	0x2E	0x30	0x32	0x38	0x00
03 mrt 17:50:55:219	MSH_A_3	1	54	3	38	0x00	84	26	36	0	0	0	0	0	0x00	0x00	0x00	0x00	0x00	0x00	0x00	0x00	0x00	0x00	0x00	0x00	0x00	0x00	0x00	0x00	0x00	0x00
03 mrt 17:50:55:281	MSH_A_3	1	54	3	38	0x01	84	26	36	0	0	0	0	0	0x00	0x00	0x00	0x00	0x00	0x00	0x00	0x00	0x00	0x00	0x00	0x00	0x00	0x00	0x00	0x00	0x00	0x00
03 mrt 17:50:55:297	MSH_A_3	1	54	3	38	0x04	84	26	36	0	0	0	0	0	0x00	0x00	0x00	0x00	0x00	0x00	0x00	0x00	0x00	0x00	0x00	0x00	0x00	0x00	0x00	0x00	0x00	0x00
03 mrt 17:50:55:375	MSH_A_3	1	54	3	38	0x05	84	26	36	0	0	0	0	0	0x00	0x00	0x00	0x00	0x00	0x00	0x00	0x00	0x00	0x00	0x00	0x00	0x00	0x00	0x00	0x00	0x00	0x00
03 mrt 17:50:55:391	MSH_A_3	1	54	3	38	0x07	84	26	36	0	0	0	0	0	0x00	0x00	0x00	0x00	0x00	0x00	0x00	0x00	0x00	0x00	0x00	0x00	0x00	0x00	0x00	0x00	0x00	0x00
03 mrt 17:51:19:056	MSH_3	58	1	3	14	0x01	38	0	0	0	0	0	0	0	0x03	0x00	0x32	0x30	0x37	0x31	0x30	0x30	0x30	0x30	0x38	0x32	0x38	0x2E	0x34	0x30	0x34	0x00
03 mrt 17:51:19:383	MSH_A_3	1	58	3	14	0x00	38	0	0	0	0	0	0	0	0x00	0x00	0x00	0x00	0x00	0x00	0x00	0x00	0x00	0x00	0x00	0x00	0x00	0x00	0x00	0x00	0x00	0x00
03 mrt 17:51:19:477	MSH_A_3	1	58	3	14	0x01	38	0	0	0	0	0	0	0	0x00	0x00	0x00	0x00	0x00	0x00	0x00	0x00	0x00	0x00	0x00	0x00	0x00	0x00	0x00	0x00	0x00	0x00
03 mrt 17:51:35:295	MSH_3	60	1	3	100	0x01	90	0	0	0	0	0	0	0	0x03	0x00	0x32	0x30	0x37	0x31	0x30	0x30	0x30	0x33	0x39	0x38	0x32	0x2E	0x34	0x30	0x35	0x00
03 mrt 17:51:35:607	MSH_A_3	1	60	3	100	0x00	90	0	0	0	0	0	0	0	0x00	0x00	0x00	0x00	0x00	0x00	0x00	0x00	0x00	0x00	0x00	0x00	0x00	0x00	0x00	0x00	0x00	0x00

Table 4.1: Example of raw sniffer-data. The data in this format follows the layout of the packet as discussed in section 3.2.1 (see figure 3.3 and table 3.1).

4.2.2 Analysis of trace-data (communication paths)

We have developed a tool to visualise the communication paths used by the nodes in the network. This tool is able to take a log of the communication in the network (such as the data collected with the RF-sniffer device) and construct a visual representation of the data paths. This is possible because within each packet the full routing information for that packet is contained, i.e. the intended route for each packet is stored as an in-sequence list of hops. As discussed in section 3.2.1, the sequence of hops for each packet is contained in `RN0-RN4`. We can visualise this list for each packet by drawing a line between any two nodes which are adjacent in any packet's hop-list to show a communication-link. The thickness of the line dependents on the number of packets that travel over the same link, i.e. it serves as a measure of the amount of traffic between nodes. Regular metering nodes are shown as circles, whereas the data-concentrator is represented by a trapezium. Figures 4.3 and 4.4 show the result of running the logged data from the RF-sniffer device through this tool. This clearly shows

Figure 4.3: Communication paths in Den Bosch.

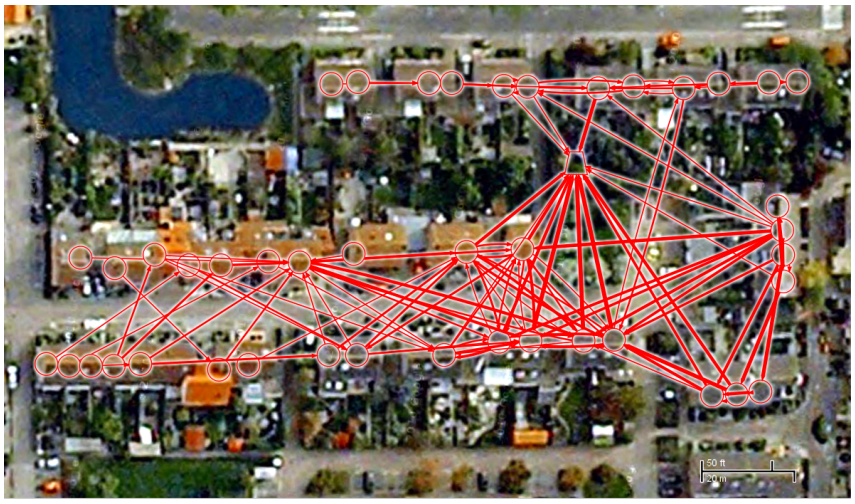


Figure 4.4: Communication paths in Goes.

the general trends in communication, which nodes are within communication range of one-another, which nodes are used as relay-nodes more often, potential bottlenecks, etc. As can be seen there is a small number of nodes close to the data-concentrator node which act as the final relay nodes for the

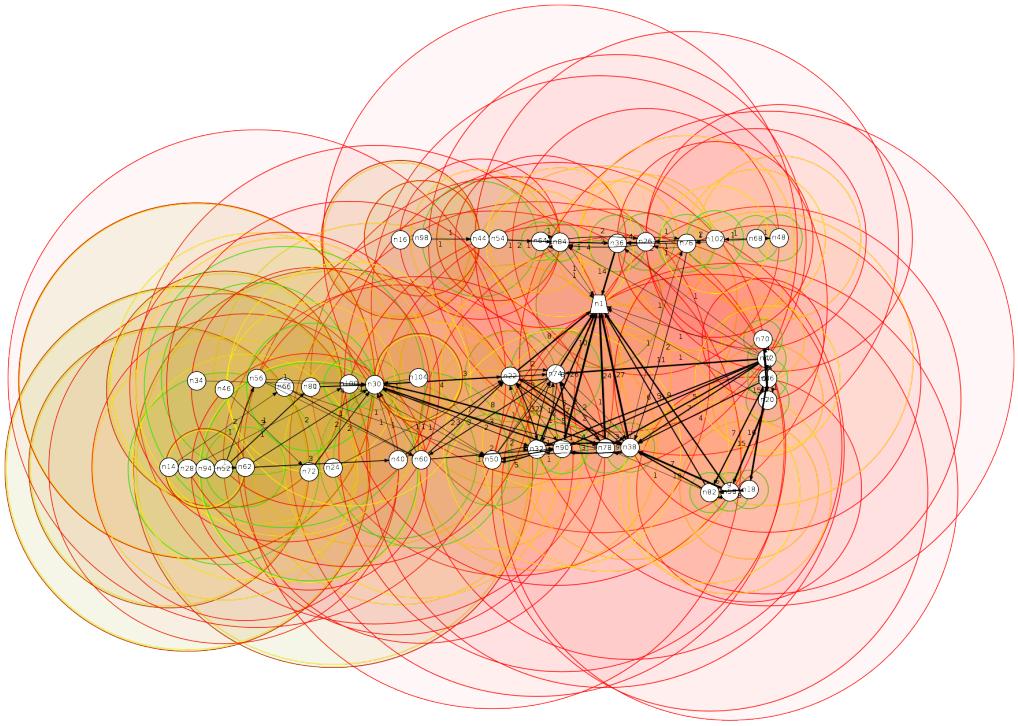


Figure 4.5: Visualising transmission distances in Goes.

rest of the network. Also visible is the trend for paths towards the end-point at the data-concentrator node, despite the random selection of next-hop nodes in the route-discovery phase. This is especially clear in the lower left area of figure 4.4.

Generating these visualisations requires some rudimentary analysis of the data, for example to determine the thickness of the links. These statistics can be visualised, too. Figure 4.5 shows a colour-coded overview of the minimum (green), maximum (red) and average (yellow) observed transmission distances for each node[1]. This figure uses the same visualisation elements as figures 4.3 and 4.4. It is also possible to run the visualisation tool as a simple analysis tool, outputting some simple statistics. Table 4.2 gives an example of the statistics that can be obtained for the two scenarios used in the field experiments. Some of these values, such as for example the average per-hop distance, have been used to adjust some of the simulation parameters to better match with the real-world results, as well as to verify that the routes used by nodes in the simulation are sane. This process is described in chapter 5.

	Nr. routes	Avg. nr. hops	Avg. distance	Avg. distance/hop
Goes	532	3.08 ± 1.08	$89.18\text{m} \pm 38.92\text{m}$	$28.95\text{m} \pm 15.25\text{m}$
Den Bosch	11183	2.87 ± 1.02	$15.09\text{m} \pm 9.68\text{m}$	$43.29\text{m} \pm 23.45\text{m}$

Table 4.2: Statistics on communication distances.

Besides being able to read the data extracted using the RF-sniffer device, this visualisation tool is also able to generate the same visualisation from the data output by the simulation. As an example, figure 4.6 shows the result of running the simulation output of a simulation of the Goes topology through the visualisation tool.

[1] This image uses a colour-coded representation to represent minimum, maximum, and average distances and may be difficult to see in grey-scale reproductions of this work.

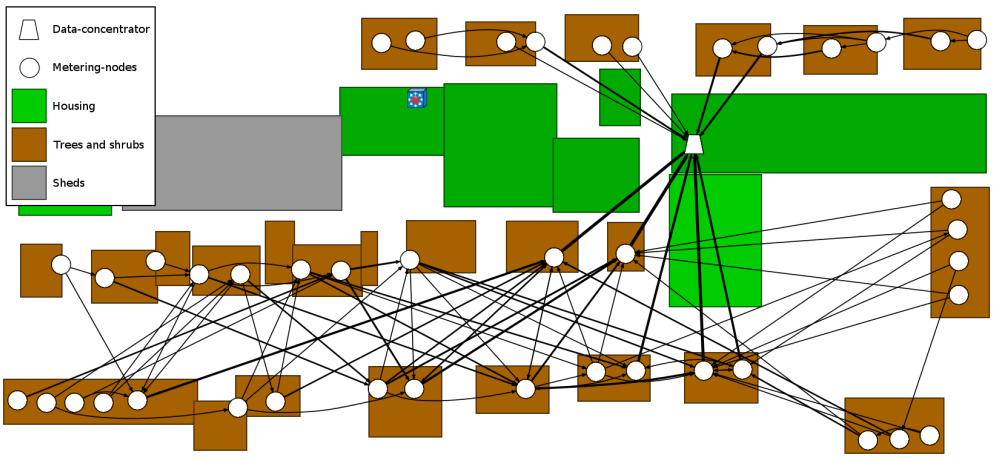


Figure 4.6: Simulation of the Goes topology.

4.2.3 Performance estimation based on analysis of latency values

Using the historical logs available from the intermediate-inspections performed at the two test locations, we have performed an analysis to gain some insight into the expected behaviour of the protocol before proceeding to the simulations. As discussed in section 4.2.1, the RF-sniffer device is able to capture any packet which is transmitted within antenna-range of the RF-sniffer device. However as we were working with the captured packet logs it was discovered that the range of the RF-sniffer device was quite limited. The cause for this is twofold. On the one hand there is the limited transmission power of the metering-devices, which in combination with the physical environment (terrain, obstacles) causes the RF-signals to attenuate quickly, before reaching the RF-sniffer device. On the other hand there is the RF-sniffer device itself, which was constructed from a repurposed metering-device. This meant that the RF-sniffer device inherited the properties of a normal metering-device. This means that the RF-sniffer device's antenna is no more sensitive than a normal metering device. It also means that, like the normal metering devices, it can not receive packets which have been damaged (due to the on-chip CRC verification, as discussed in section 3.2.1), either due to external influences or due to collisions, nor can it capture more than one packet simultaneously.

Our analysis was further complicated somewhat by the fact that the exact positioning of the RF-sniffer device during its operation was not known. However it was known that the device remained mostly stationary for the duration of the sniffing-sessions, and that it was 'placed near to the data-concentrator node'. This latter fact can also be deduced from the logs which indeed do contain packets which are transmitted directly by the data-concentrator node, thus indicating that the RF-sniffer device was within one-hop distance of the data-concentrator. Therefore we have based our analysis on a reconstruction of the communications of nodes which are relatively close to both the data-concentrator and the RF-sniffer device. This means that it is possible for certain packet types to be overrepresented in our analysis. Packets which reach the data-concentrator node in fewer hops, for example, can be expected to show up more often than packets which travel a greater distance. This is partly because packets that travel longer distance have a higher chance of being damaged during the forwarding process simply because they are forwarded more often, and partly because depending on the exact positioning of the RF-sniffer device it may only pick up transmissions on one 'side' of the data-concentrator, which, depending on the shape of the network could lead to a different ratio of packet types being sniffed versus that actually in the network.

In order to estimate the end-to-end delay introduced by the protocol we have looked at packets for which we can be sure of the time of transmission and/or reception. There are several conditions which indicate that we can be sure of the initial transmission of a packet. Firstly packets which are acknowledgements, but which have not yet been forwarded (this includes DIR packets, which are never forwarded), must have just been transmitted by the data-concentrator node. Since we know that

the only communication in the network in the logged sniffer-data is between metering-devices and the data-concentrator node, we know that such a packet marks the half-way point in communication. Such packets can be distinguished by the fact that their route-flag is set to `0x00`. For packet types other than acknowledgements determining whether they have just been transmitted depends on the type of packet. For search packets the route-flag never gets set. Instead we need to inspect the list of intermediate hops, and determine if it is empty. If so, we can say that the packet has just been transmitted. For meshed-communication packets we can again look at the route-flag for determining the transmission times.

In order to compute an estimate on the actual transmission times we employ a similar method as above to filter on the data logged using the RF-sniffer device. As it turned out we had relatively few packets to work with. Table 4.3 shows how many packets were available for our analysis in both case-studies. As can be seen in the case of Goes there are still relatively many packets which pass the filter criteria, whereas in the Den Bosch scenario only very few packets are suited for analysis. This is counter to our expectations, because there is more logged data available from the scenario in Den Bosch. Moreover we know from reports that for this particular scenario there is recorded data for a full re-initialisation of the network. So it is certainly reasonable to expect more usable data to be available. Possible reasons for the lack of (usable) data may be that the RF-sniffer device was positioned in such a way that it captured mostly traffic from the nodes, but not the replies from the data-concentrator node. Although there is some evidence to suggest this (in the Den Bosch scenario only 36% of all captured packets originated at the data-concentrator node, versus almost 55% for the Goes scenario), we are unsure if this is sufficient to explain the large difference between Den Bosch and Goes.

	Total values	Direct	Indirect	Search	Mesh
Goes	406	227	179	10	169
Den Bosch	49	19	30	2	28

Table 4.3: Number of packets available for latency analysis.

After the list of packets was filtered, it was ordered by time. Next, communication latencies were estimated by taking the time between a request and the corresponding acknowledgement, by applying some knowledge about the operation of both the protocol as well as knowledge about the internal-workings of the metering-devices and the RF-transmitter. For example we know that for a direct-communication message the protocol attempts to send a response as quickly as possible, and we know that there is a small (but measurable) delay in processing the message and preparing the response. This needs to be accounted for when estimating the one-way communication delays. Because there is only communication initiated from the metering-devices to the data-concentrator node, we can only estimate the communication delay from nodes to the data-concentrator. However because replies always follow the same path that they arrived on—only in reverse—it can be assumed that on average this delay is symmetrical. Table 4.4 shows the estimated (two-way) communication delays that have been calculated for the scenarios in Goes and in Den Bosch. This data can be used to verify the accuracy of the implementation of the simulation. If the simulation is sufficiently realistic, an analysis of the output traces from the simulation should yield similar latencies as in the case-study scenarios.

	Minimum	Maximum	Average	σ
Goes	0.015s	0.296s[a]	0.031s	0.055s
Den Bosch	0.015s	0.19s	0.038s	0.035s

[a] Actually the maximum value here was $1.016s$. However this is such an extreme outlier and that we consider it an erroneous value. The value shown here is the second-largest value.

Table 4.4: Communication latency estimations for Goes and Den Bosch.

4.3 Conclusions

We have given a brief overview of the conditions under which smart-devices are typically expected to operate, by means of two field trials. We explain why these particular scenarios were picked for the case-studies, which aspects of the protocol they are meant to exercise. These scenarios provide a good base for developing thet simulations, as describe in chapter 5. We have presented a tool for visualising the activity in the network. We show how visualising the trace-data we have obtained from a number of real-life examples gives insight into the behaviour of the routing-protocol. We have shown that the same tool can be used to perform a rudimentary analysis of the network, able to extract path-lengths, hop-counts, per-hop distances, and more. We have performed an analysis of the average communication latencies between nodes, and we have showed how the trace-data can be used to verify the accuracy of the simulator by comparing the routes in the simulation to those in the real world.

Chapter 5

Design and realisation of the simulation framework

In this chapter we detail the work done on building the simulation of the proposed routing-protocol, as well the tools to support the generation of scenarios and the verification of the simulation. In section 5.1 we first discuss the various choices of simulation environments there are, which simulator we have chosen as a starting point, and why. Next we present the overall architecture of the simulation that we have constructed in section 5.2, before giving an in-depth description of the program-code that has been developed for the simulation of the protocol, as well as the various additions and modifications that were required to existing parts and components of the simulator in section 5.3. This includes a description of the attenuation modelling that was done (section 5.4) in order to come to a faithful representation of the real-world scenarios. Once we were able to simulate the protocol and environments, we verified the accuracy and correct of the simulations. This process is detailed in section 5.5. Finally we briefly talk about the simulation environment itself, we discuss how the execution of simulations was automated and reflect on the performance of the simulation (section 5.6).

5.1 Choice of simulator

The first step in setting up the simulation experiments was to decide which simulator we would use. This decision was made early on in this project, in order to minimise the time spent assessing and familiarising with various simulators, and hence to maximise the available time for the actual implementation of the protocol. At that point very little was known about the exact details of the protocol, and what would be required of the simulator in order to develop a proper simulation. For example because we did not yet have access to the protocol's documentation nor it's implementation, one of the criteria used in our decision was the flexibility of the simulator. For example a simulator which could perfectly simulate an IP network but lacked the flexibility to simulate other types of network might be a good choice if the communication protocol was build on top of existing IP network. However if the protocol used a different underlying network protocol (or indeed none at all), a different simulator might provide a more suitable environment, and hence be the better choice.

We realise that given the vast array of different simulation software in existence today, it is very difficult to give a proper, in-depth, analysis of each individual simulator. Therefore this section not meant as an exhaustive survey of available simulators, but rather aims to provide a rationale for *our* choice of simulator, OMNeT++. We have considered a number of well-known and highly regarded network simulators, as well as some that are less common. We also discuss some more general classes of simulators, and their suitability to this project.

5.1.1 OMNeT++, ns-2, QualNet, and OPNET

In terms of the protocols these simulators support we found a detailed matrix-overview in [67], covering ns-2, QualNet, OPNET, and OMNeT++. Here it can be seen that at the higher layers of the network stack (the transport layer and up) OMNeT++ appears to offer less default functionality than the other three simulators. However because we are interested in implementing a simulation of a novel routing protocol this should not prove a hindrance, especially so because we will evaluate the routing protocol in the context of smart-metering applications. This means that we will be running our evaluations using workloads typically found in a smart-metering scenario, and the simulator's ability to conveniently simulate a video-conferencing scenario will not be used anyway. At the lower levels (network layer and down) OMNeT++ seems to be more-or-less comparable to ns-2, albeit each having more options available in different areas. E.g. ns-2 provides more scheduling and queueing algorithms, while OMNeT++ has support for IPv6. It was felt that having the ability to simulate IPv6 was important option to keep open. Finally it should be noted that this particular survey examined an older version of the OMNeT++ simulator (version 3.2), while at the time of writing version 4.1 was already released, which included more features. Also it was not entirely clear which extensions to OMNeT++ were considered in this survey, since for example the overview of supported features on the official the INET Framework-status website [1] shows more protocols are in-fact present in the INET Framework. Besides the INET Framework there are other frameworks too, which provide more features and protocols. So from this survey we can conclude that at the time of writing both OMNeT++ and ns-2 were good candidate simulators, although QualNet and OPNET would seem even better. However even-though both QualNet and OPNET have a richer feature-set, they were ultimately rejected for use in this project because of their commercial nature (all other simulators mentioned in this survey are either available free of charge, or make an exception for academic use).

5.1.2 ns-2 and NS-3

Based on the previous survey we concluded that both OMNeT++ and ns-2 were good candidate simulators. Especially ns-2 is, and has been for a longtime, recognised as the most widely used network simulator. However in spite of it's popularity, over the years a number of drawbacks have been identified in ns-2 by various reviewers and surveys. In [68], which surveys a large number of simulators comparing and assessing their suitability in simulating (wireless) sensor networks it is noted that 'Packet formats, energy models, MAC protocols, and the sensing hardware models all differ from those found in most sensors. Also simulation of interactions between the application level and the network protocol level not possible. It is however recognised that one of the reasons makings ns-2 the *de facto* choice of simulator is its extensibility. Further [68] states that ns-2 has an object-oriented design, allowing for straightforward creation and use of new protocols. On the other hand [69] notes that 'The simulator has a complex structure. This makes adding new components a hard task since it requires a good knowledge of the simulator.

These two statements seem hard to reconcile, highlighting the subjectiveness of such surveys. What is a common theme however is that over time many extensions and modifications to the ns-2 simulator have been developed by various individuals and research groups. Unfortunately it appears that often these enhancements are not contributed back to the main ns-2 project, but are instead discarded, or in the best case maintained on a voluntary basis by the original developers. A large list[2] of contributed modules is kept at the ns-2 wiki-website. It is unclear wether this list is actively pruned for links which are no longer available, but a cursory survey of the links showed that a number of these extensions were no longer available at the listed addresses.

These reasons led to the creation of the NS-3 project. The NS-3 project is the successor to the ns-2 network simulator and represents a complete re-write of the ns-2 network simulator. NS-3 is written entirely in C++, whereas ns-2 was written in a combination of C++ and OTcl. Compared to C++, OTcl is a relatively obscure and difficult programming language. This makes it easier to start working with

[1] http://inet.omnetpp.org/index.php?n=Main.Status
[2] http://nsnam.isi.edu/nsnam/index.php/Contributed_Code

NS-3 than ns-2 because it does not require one to learn a new programming language. The goal of the NS-3 programming language is to collect and combine all of the individual patch, improvements and extensions to ns-2 into one program [70]. In this way it is hoped to create a large community of people working together on the simulator. However fairness demands to say that we did not consider the NS-3 project because at the start of the project there wasn't sufficient literature about the maturity of NS-3. Further, to date the ns-2 website makes no mention of the availability of a newer version of the simulator.

In the end the decision to favour OMNeT++ over ns-2 was based on the desire to use a more recent, easier to use simulator. Various inquiries around the department indicated that ns-2 was more difficult to master. The various surveys did not conclusively indicate the suitability of ns-2 for this particular project. In my estimation the amount of work in implementing the protocol would be about equal between OMNeT++ and ns-2, with OMNeT++ (which is entirely C++ based) likely being slightly easier due to having some previous working experience using C++, whilst having no experience with OTcl.

5.1.3 JiST/SWANS

Our evaluation of the JiST/SWANS simulator (Java in Simulation Time/Scalable Wireless Ad hoc Network Simulator) revealed that this was rather old and no longer being maintained (the latest update to the JiST/SWANS website was in early 2005, nearly six years ago). Nevertheless it boasts some impressive statistics on its website[3]. It supposedly is many times faster than competing solutions such as ns-2 and GloMoSim, while using far less memory. However closer inspection revealed some oddities, such as the fact that the performance tables showed that GloMoSim used only 1.2% of the memory that Parsec[4] required for simulating the same scenario. However the GloMoSim simulation library is built on top of the Parsec simulation library. Therefore it seemed quite odd that GloMoSim would use less memory than it's parent simulator. However the reason for ultimately rejecting the JiST/SWANS simulator were not these strange benchmark results, but rather the fact that it was no longer updated, and provided relatively few and simple network simulation protocols and features. In order to avoid spending time implementing features which were already present in other simulators, JiST/SWANS was rejected as a candidate simulator.

5.1.4 GloMoSim

Although we rejected the JiST/SWANS simulator, our evaluation did point us to another possible network simulator, GloMoSim. In our research we found a number of recent surveys [68, 69, 71–73] which mention GloMoSim as one of the 'popular' and 'widely used' network simulation tools. However we found it quite hard to find concrete information on the actual simulation capabilities provided by GloMoSim. This may be due to the fact that according to [68] the development of GloMoSim was discontinued in the year 2000. This was because at that time the developer behind GloMoSim had launched his own company, and started to sell a commercial product based on the GloMoSim simulator, QualNet. Because of this it was feared that likely GloMoSim had been surpassed in features and quality of simulation by other network simulators. Indeed, according to [72] GloMoSim can only simulate IP based networks. This would mean that, unless the protocol to be simulated is based on top of IP, using GloMoSim to implement the simulation could either be considered impossible, or yield a highly unrealistic simulation. However it should be noted that according to the official website[5], GloMoSim is build using a layered approach, in an effort to '... allow the rapid integration of models developed at different layers by different people.'. Thus it might in fact be possible to implement the protocol as a routing protocol in GloMoSim after all. In the end our choice to reject GloMoSim was based on a combination of factors. First of all there was uncertainty about the present day usefulness of GloMoSim due to it not being maintained for over ten years. This could give problems on a number

[3] http://jist.ece.cornell.edu/
[4] http://pcl.cs.ucla.edu/projects/parsec/
[5] http://pcl.cs.ucla.edu/projects/glomosim/

of levels, ranging from the question of whether we would still be able to compile the software on a modern (Linux) system, down to the available features in the software which like have been surpassed by newer simulators. Secondly there were some doubts about the amount of work that would be involved in implementing the protocol in this simulator. Because the simulator was aimed at simulating IP based networks, implementing the protocol in this simulator might involve more work compared to doing the implementation in a different simulator. This because due to the fact that for a given network layer only a small number of implementations are by default present in the simulator. Therefore multiple layers in the simulator might need to be changed or even completely rewritten.

5.1.5 Other simulators and emulators

Many more simulators were encountered during our inventarisation of the availability simulation options. The abilities, goals and type of simulators varied greatly. There are many network simulators with varying degrees of specialisation. Some are based on existing simulators, but add some specialisation particular to one field of interest. An example of this type of simulator is SensorSim, which is based on ns-2. The disadvantage of this type of simulators is that they are often based on older versions of their parent-simulator, and hence miss-out on any updates or improvements that are made to that simulator. More over they inherit the flaws that may be present in their parent simulator, staring from flaws or drawbacks in the initial design down to particularities in the implementation.

We also encountered another highly specialised type of simulator. These programs are designated better as emulators rather than simulators. The main difference between a simulator and an emulator in this distinction is the level of detail in the simulation. Whereas a simulator generally perform a higher level simulation than an emulator, which generally performs a very low level simulation. That is to say a simulator *simulates* the general behaviour of a device or protocol, whereas an emulator attempts to recreate the actual device or implement the actual protocol as it would appear in reality. As an example of the former consider the simulation which we present in this thesis. Our simulation attempts to capture as much of the real world as possible, but the actual implementation of the protocol, although it captures the behaviour of the protocol faithful as possible, differs from that in the actual devices at a few details (see footnote 14 for a detailed overview of these differences). TOSSIM [74] is an example of the latter. TOSSIM is a TinyOS[6] 'mote' simulator which simulates the execution of actual machinecode on MICA[7] sensor nodes. The MICA 'mote' is a wireless sensor node hardware module, and TinyOS It allows to run programs compiled for TinyOS directly, so that developers can test not only their algorithms, but also their implementations [8].

Although we encountered these emulators in the field of (wireless) sensor networks, they are likely to exist for other areas as well. We opted not use any of these simulators and emulators because we felt that they were too specifically aimed at one very particular type of simulation. At the start of the project it was not entirely clear exactly what type of simulation would be required, or indeed even the exact definition of the device we would simulate. TOSSIM for example would be a poor choice if the device we were simulating was not based on a MICA (or very similar) mote. Therefore it was necessary to choose a simulator which allowed for greater flexibility in the eventual implementation.

5.1.6 OMNeT++

Most of the advantages and reasons for choosing the OMNeT++ simulator have already been mentioned before, in the comparisons with the various other simulators that we investigated. We therefore give only a brief re-cap of our main perceived advantages and reasoning for choosing the OMNeT++ simulator environment. Firstly the OMNeT++, and its supporting software-libraries are *free*. There no charge for using OMNeT++ for academic and non-profit use, and, most importantly OMNeT++ is *free software* (open-source), published under the Academic Public License (APL). This holds

[6] http://www.tinyos.net/
[7] http://web.archive.org/web/20020212062204/http://www.xbow.com/Products/Wireless_Sensor_Networks.htm
[8] http://www.cs.berkeley.edu/~pal/research/tossim.html

for the additional libraries that we expected to require (MiXiM and Mixnet), too. Further our investigation revealed a large quantity of seemingly high-quality documentation. OMNeT++ sports a comprehensive manual, several step-by-step tutorials introducing the OMNeT++ modeling workflow, a set of example programs, extensive doxygen-documentation of the source-code. Several surveys quote OMNeT++ as being among a popular choice of simulator, and even the OMNeT++ website mentions its rise in popularity. Being more popular should mean that it is easier to find answer to any eventual questions, as well as ensure that fewer issues remain in the simulator. Our final argument for choosing the OMNeT++ simulator over any of the other options is that it is a modern approach, based on object-orientated concepts, with a well structured, layered architecture. Being written in the C++ programming-language ensures that it is relatively simple to get good performance from the simulation. Finally the author is relatively familiar with C++, making it easier to pick up and write code for this simulator.

5.2 Architecture of the OMNeT++ simulator

Before we proceed with a detailed description our implementation of the protocol and simulation (section 5.3), we first give an architectural overview of the simulation as it was implemented using the OMNeT++ simulation environment. Figure 5.1 shows at a very high level how we have implemented the simulation of the protocol, as well as how the components of our implementation are related. We have used an object-oriented approach for our implementation, leveraging inheritance where possible. In figure 5.1 the gray package marked *Omnet++* lists only those components of OM-

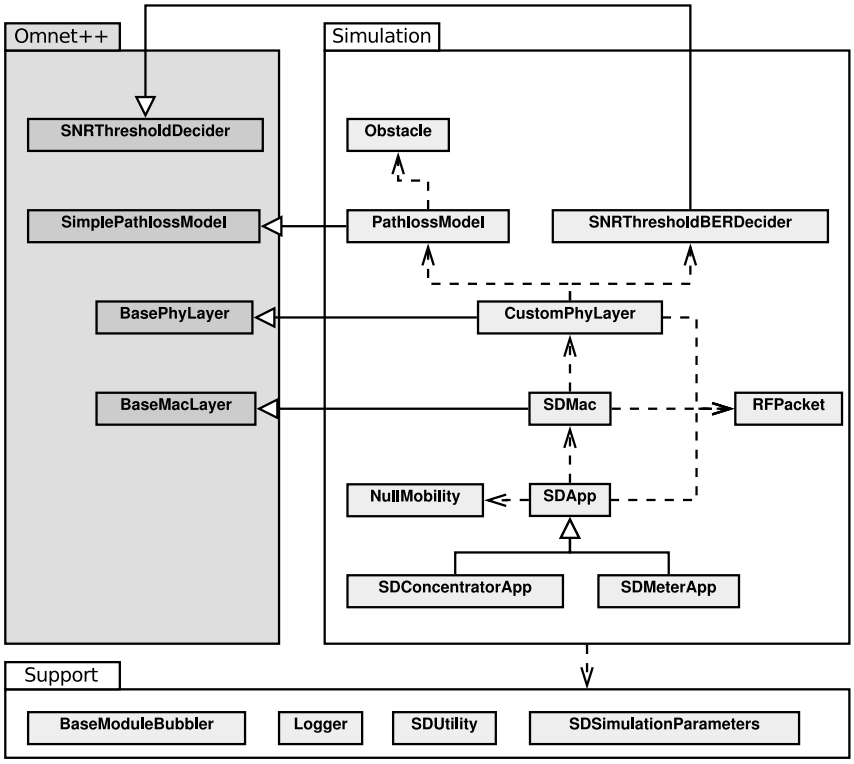


Figure 5.1: Architectural overview of the simulation implementation.

NeT++ that we have either had to make some modifications to, and omits standard and unmodified components. The *Simulation* and *Support* packages all are components that are or contain original work. Components under the Simulation heading are directly related to the simulation of the protocol and/or environment. Components under the Support heading exist merely to support the Simulation

components. We now proceed to give a brief description of each component and its purpose.

- Supporting components.

  `BaseModuleBubbler`
  The OMNeT++ simulation environment comes with a graphical user interface for running simulations. This is useful when simulating small networks and for debugging, as it allows for a visual inspection of the operation of the simulation. One can see the connection between nodes, as well as packets traversing the network. One can go through the simulation step-by-step, one event at a time, and there are facilities for inspecting the state of objects in the simulation, such as network-nodes or packets. One particularly useful aspect of this visualisation is that it allows network nodes to present a message to the user, for example to notify the user that a message is about to be sent or has been received. This is called 'bubbling' in OMNeT++ parlance, because the message is displayed in a speech-balloon attached to the originating entity. `BaseModuleBubbler` implements a simplified version of this interface, and every component under the Simulation heading extends the `BaseModuleBubbler` class. In this way each component can display a message to the user in a convenient manner. The primary purpose hereof was to facilitate in debugging the simulation.

  `Logger`
  The `Logger` class serves a similar purpose to the `BaseModuleBubbler`, only its job is to provide an interface for recording certain events in the simulation to permanent storage. OMNeT++ offers a similar feature, but the implementation provided by OMNeT++ logs also messages sent to the user, or from other components in the simulation that may not be relevant for our purposes. By implementing our own logging interface we can provide faster processing of messages, since we only log exactly that which is relevant, eliminating a filtering step that would otherwise be required. Further we can output directly to a properly named file in the same directory that other simulation results are stored, which makes for a more tidy directory structure, as well as eliminating one more step in the processing of simulation results.

  `SDSimulationParameters`
  The `SDSimulationParameters` is not a class, but rather a collection of parameters which are of influence on the simulation. These are specified in the form of constants (C++ `#define`s) rather than as run-time configurable variables. Although these could, or rather, should be specified as parameters to the simulation using a configuration file, for performance reasons or simply because it would be technically more dificult to do so, they are not. This includes things such as the average processing-delay in a node and how to handle requests for bubbling or logging.

  `SDUtility`
  The `SDUtility` component contains a number of small, but useful algorithms pertinent to the simulation, as well as a number of classes that could perhaps be placed under a different heading. E.g. we have implemented classes implementing vectors and lines, and various operation thereon, which are used by our obstacle implementation (section 5.4.1). However because they have a supporting role only we have opted to categorise them as utilities.

- Components that are part of the core of the simulation.

  `Obstacle`
  The obstacle class is used to represent various types of obstacles in the simulated environment, ranging from housing to plants and trees. We detail our obstacle model in section 5.4.1.

  `PathlossModel`
  The `PathlossModel` class implements the pathloss-model that we have used in our simulations, the Free-Space Path-Loss (FSPL) model. The path-loss model calculates, given

sender and receiver parameters, as well as other parameters such as the strength of the transmitted signal, the signal level at the receiver. It is based on the `SimplePathlossModel` provided by MiXiM, but has been extensively modified to incorporate losses due to obstacles, as well as to reduce the computations involved in determining the amount of signal attenuation, thus speed up the simulation. See also section 5.3.2.

`SNRThresholdBERDecider`
The `SNRThresholdBERDecider` implements a required part of the OMNeT++ simulation, which is responsible for the final decision on whether or not a reception was successful. We have based our Decider model on the `SNRThresholdDecider` provided by OMNeT++. This Decider model takes into account only the received signal strength and noise level to determine reception. We have extended this model with a simple bit-error rate simulation in order to be able to evaluate the protocol under varying bit-error rates, rather than varying Signal-to-Noise Ratios (SNR), which is more difficult to implement in OMNeT++. See also section 5.3.2.

`CustomPhyLayer`
The `CustomPhyLayer` is a simple extension of the OMNeT++ provided `PhyLayer`. In order to make use of our custom Decider and PathlossModel it was required to make a small change to this class. See section 5.3.2 for details.

`SDMac`
The `SDMac` class implements the functionality of the MAC layer, providing an interface between the tranceiver-hardware and the application layer. We have based the implementation of our MAC layer on MiXiM's `BaseMacLayer`. We discuss our modifications in section 5.3.2.

`RFPacket`
The `RFPacket` is used to represent the packets that are being transmitted by the nodes in the simulation. It is a hybrid-class, in the sense that it is based on a class generated by OMNeT++ based on a description of a packet. The generated code was sub-classed in order provide additional functionality such as simpler manipulation of the route-list and hop-flags (see section 3.2.1).

`NullMobility`
The `NullMobility` class is little more than a stop-gap measure. The MiXiM framework assumes that all nodes will be mobile, and hence provides only classes that implement simulations of certain movement patters. However the nodes in our simulation are firmly attached to an electrical meter, and hence do not move. Apparently a similar situation arose in the INET Framework, since we found an implementation of a mobility class which does effectively nothing there [9]. At first we used this class as-is, but we soon found that it was lacking some necessary functionality (such as for example the reporting of a node's position). Since we had to implement this functionality, and the existing code was only an empty shell, we have listed it as original work.

`SDApp`, `SDConcentratorApp` and `SDMeterApp`
These last three components are the most interesting, and implement the core functionality of the routing-protocol, as well as the behaviour of the metering devices. The reasons for combining the protocol implementation and metering-behaviour into a single class are two-fold. Firstly this mirrors the way that this functionality is implemented in the actual devices, and secondly the functionality required of the metering-devices in order to test the protocol is so small (only generating a packet once in a while) that there is little point in separating this into its own class. The base-class `SDApp` implements most of the shared functionality of metering-nodes and the data-concentrator nodes such as the caching of routes and the handling of duplicate packet detection, as well as many of the things that are shared from a simulation perspective such as the collection of statistics and such.

[9] Our `NullMobility` class is based on the INET Framework's `NullMobility` class, which can be found in the headers and source files in `INET-OverSim-20100505/src/mobility/`.

The `SDMeterApp` implements the behaviour of metering-nodes, and `SDConcentratorApp` that of the data-concentrator. A detailed explanation of the differences between these two nodes is given in section 5.3.1.

5.3 Protocol implementation

For the purpose of evaluating the performance of the protocol we have created an implementation of the protocol in the OMNeT++[10] network-simulator. This implementation is based on the existing C implementation which runs on the real nodes in the case study.

Although OMNeT++ is geared towards simulation of high level protocols through the use of various frameworks and packages such as the INET Framework[11] (simulation of commonly used protocols such as UDP, TCP, IP, IPv6, etc), OverSim[12] (a peer-to-peer network simulation framework) and MiXiM[13] (a modeling framework for mobile and fixed wireless networks), it also possible to fore-go these frameworks and program directly to the lower network layers such as the MAC layer or even the Physical layer. We have taken an intermediate approach, in that we use both features from a high-level framework, as well as implementing (almost) at the MAC layer.

5.3.1 Implementation details

For our experiments we model the protocol as it is described in chapter 3. While creating our model we have used the actual C-code which runs on the real-world device as a reference, and created a new implementation of that code as an OMNeT++ module. This code, as we received it, implements two distinct modes of operation for the devices. In the first mode the device acts as a metering-station, expecting to be coupled to a metering-module which provides measurements. The implementation and interaction of this separate metering-module falls outside the scope of this work, and we assume that measurements are available when they are required by the protocol. Further we shall not be concerned with the actual values of the measurements, since the payloads which are to be delivered have no influence on the operation of the protocol. In the second mode of operation the device acts as a sink-node in the network, and expects to be coupled to an external uplink. The same assumptions again apply here.

In the original C-code, both operational modes (of the protocol) are defined in the same set of files, and a compile-time switch determines which mode the compiled binary will operate in. Apparently this was done in order to achieve a form of code re-use. The implementation of both modes share much code, and differ only on certain query/response patterns and external connections. In our implementation we implement the two different modes using C++ objects and inheritance. To do this the shared part of the protocol is implemented in a base-class, and the different modes are implemented as derived classes, implementing only the behaviours expected from the device in each mode. This however leads to a minor difference in the observable behaviour of the simulated devices, where the actual device may respond slightly differently in certain situations. For example in the real hardware the sink-nodes still contain the code necessary to respond to query packets directed at a metering-device, and will reply with an acknowledgement. During the course of normal operation this should never occur, but it could be useful in the field to communicate with any given node in the network. One could for example imagine using this to remotely (without entering a residence and physically connecting to the device) query a device about operating statistics or perform techical troubleshooting. However because we are only interested in simulating the behaviour of the protocol during normal operation this difference should not have an observable impact on the results of the simulation.

[10] http://omnetpp.org/
[11] http://inet.omnetpp.org/
[12] http://www.oversim.org/
[13] http://mixim.sourceforge.net/

5.3.2 Modifications to the simulator

As mentioned before we use a number of standard components provided with the OMNeT++ environment. During the course of developing our simulation we have had to make some minor changes to some of these component layers. We briefly touch on some of the other changes which we have made to the simulation environment in order to perform our experiments.

For our implementation we make use of the MiXiM framework. MiXiM provides radio wave propagation models, interference estimation, MAC protocols and more. While it is possible to use MiXiM purely as a traditional library, providing ready-made functionality, it is also possible to implement customised models. Customised models can be either based on models provided with MiXiM, or created from scratch. We have used several of MiXiM's 'simple' models as a base for implementing our own models in cases where features which were either unavailable or where the more specific modules MiXiM provides (such as those for simulating IEEE 802.11 style networks) were unsuitable for our purposes and too complex to modify. For example we have extended MiXiM's `SimplePathlossModel` with our own obstacle model, and although an obstacle model is advertised in [75], it is at present not yet available in MiXiM. Therefore we have developed a simple obstacle model for use in OMNeT++. We detail these changes in the remainder of this section.

MAC layer

The modifications to the MAC layer were small, so we only give a succinct description of the changes made. Because of the simple nature of the tranceivers used in the field-trials, we opted to mirror this simplicity in the simulation by use the simplest available MAC layer. However we encountered what appears to be a bug in the `BaseMacLayer` as it is presently implemented in MiXiM, or at least an odd design choice. The issue is that the `BaseMacLayer` does not properly handle the switching of the simulated tranceiver hardware state. When a simulated hardware component (physical layer) changes states it notifies interested components by sending a notification message. However not all message are properly handled by the `BaseMacLayer` as it is currently implemented. In fact at present any notification except for 'transmission over' is ignored. The result of this is that for example the sending component may get stuck in certain states waiting for a reply, or exhibit otherwise unexpected behaviour. In particular we have had to implement proper support for switching between sending (TX) and receiving (RX) modes for the simulated hardware. We do not use (and hence did not implement) the sleep-state, as the device which we are simulating is always either in TX or in RX mode and hence does not require a sleep state. Note that the hardware used in the prototype *does* support such a sleep state, it is simply not used in this application.

Physical layer

We have used MiXiM's `PhyLayer` physical layer, in combination with the `SNRThresholdDecider` decider layer and `SimplePathlossModel` analogue layer as a basepoint for our simulation. As discussed before the physical layer is composed of two individual components. The only modification to the `PhyLayer` consist of adding the ability to load the other two modified layers. We could have made this modification in-place to the existing components (in the MiXiM source directory), but doing it this way has two advantages. First of it is a cleaner approach as opposed to muddling in the MiXiM sources. Secondly (and more importantly) in this way we retain the ability to switch back to the older, known-working versions of the component. The `SimplePathlossModel` was extended with adding the ability to place obstacles, and have them affect the pathloss of the signal. The `SNRThresholdDecider` was extended with the ability to (crudely) simulate bit-errors, and drop the received packet if any bit-error was detected. This simulates the behaviour of the hardware used, as the transceiver on the real hardware has an on-chip CRC verification. The RF-receiver does not notify the microcontroller of a reception unless the CRC of the received packet is correct. Adding and checking the CRC occurs on the hardware level, outside of software influence. We used the most simple Decider as a base because many of the other decider were designed for simulating errors in more complex forms of transmissions than the device we wish to simulate.

5.4 Attenuation modelling

5.4.1 Obstacle model

We would like to start-off by noting that there already exists a similar extension[14] to the MiXiM framework by Christoph Sommer, which is to be included into MiXiM and the INET Framework. However, even though this extension was written by someone who has prior involvement in developing for OMNeT++, MiXiM and the INET Framework and could hence save use time and effort as a ready-made simulation solution, we have opted to develop our own obstacle model for a number of reasons. First off, at the time of writing this proposed extension was not yet integrated into these other frameworks. In order to use this extension is would be necessary to obtain a development version of the OMNeT++ simulator and attempt to integrate the development version of this extension ontop of that. At this point we already had an existing implementation of our simulation working on the then-current version of OMNeT++. Therefore we felt that attempting to switch to a new, untested, in-development version of OMNeT++, and adding this new, untested, in-development extension to that posed to great a risk. It was not clear to us how much work this would be, and what the chance on success would be, if we would need to adapt our existing implementation to suit the new versions of OMNeT++, MiXiM and this extension. Therefore we have opted instead to implement our own simplified obstacle model.

In our obstacle model an obstacle is represented by an OMNeT++ module. This means that it has a specification in NED-script, and an implementation as a C++ module. The NED specification has a number of properties among which is the shape, size and orientation of the obstacle, as well a specification of the attenuation behaviour of the obstacle. For each obstacle there is a fixed attenuation cost associated with entering and leaving the space occupied by the obstacle. This attenuation is given in dB, and can for example be used to represent a signal passing through a wall. Further each obstacle has a attenuation cost associated with traversing the space occupied by the obstacle, e.g. to represent the attenuation the signal would undergo by passing through the mass of the obstacle. This attenuation cost is specified in dB/m.

At present the NED specification of the obstacles is actually less generic than the underlying C++ implementation of the OMNeT++ module would allow. The underlying C++ implementation uses an abstract `Obstacle` class which in turn uses an abstract `Shape` class to represent the actual shape of the obstacle. In this way users of the obstacle class (e.g. the physical layer) do not need to know about the actual obstacle's shape, and can just use a clearly defined interface to query an obstacle to obtain a list of intersection points or the attenuation, given a line-segment representing the signal's path. Each type of obstacle (2-dimensional or 3-dimensional, square, or round, etc.) can thus be implemented simply by creating a new implementation and adding it to the NED definition. However due to time constraints we currently only implement abitrarily orientated, 2-dimensional, rectangular obstacles. Because at present we only support one type of obstacle, the NED-specification is kept simple and does not show the modular approach to obstacles that is going on 'behind the scenes'.

In comparison to the obstacles extension to the INET Framework and MiXiM as proposed by Christoph Sommer, there are a number of similarities and differences between the two implementations. First of all our implementation, like the implementation by Christoph Sommer supports different attenuation costs to be associated with just 'hitting' an obstacle as opposed to the attenuation cost of the signal passing through the obstacle. We noticed that this feature was offered by the implementation offered by Christoph Sommer, and could easily be added to our implementation as well. This was because at this point we had already implemented the calculations which estimate the attenuation for a signal passing through the obstacle, and hence adding an additional fixed cost to the result of this computation was easy. A key difference between the two implementations is the way in which the actual shape of the obstacles is defined in each. In our implementation the shape of an obstacle is separated from the other properties of the obstacle. This allows for different implementations for different shapes, each most suited and optimised for that particular shape. In this way it is possible to precisely define the shape of an obstacle, regardless of whether this is a 2-dimensional square, rectangle or even a 3-dimensional spheroid. This is in contrast to the implementation offered by Christoph Sommer, where

[14] http://www7.informatik.uni-erlangen.de/~sommer/omnet/obstacles/

there is no such distinction between the definition of the obstacle and its shape. Here the shape of the object is simply defined as a two-dimensional enclosing polygon. This means that defining certain shapes is difficult, e.g. defining a circle can only be done by approximating its shape. Defining a 3-dimensional object is impossible. It can be noted that this mode of defining objects could easily be added to our current implementation of obstacles, as this already closely resembles our implementation of rectangular objects, which are defined as a set of four enclosing lines. Extending this to allow an obstacle's shape to be defined by arbitrary sets of lines would closely mimic the obstacle definition of the implementation by Christoph Sommer. One final note to make about our implementation of obstacles is that because we know that for the scenarios which we will be simulating nodes will remain stationary, we pre-compute all possible signal-obstacle intersections once during initialisation of the physical layer, and store the associated signal-attenuation for each pair of communicating nodes in a look up table. Then during the simulation run we do not constantly need to re-compute the signals' intersections with obstacles. Note however that this optimisation is merely an implementation detail in order to reduce the simulation time.

5.4.2 Simulation of materials

Once the obstacle model (section 5.4.1) was implemented, we needed to supply it with actual attenuation values. During the development phase of the obstacle model, values were used that were obtained through an iterative process. By refining some of the attenuation values it was attempted to match the visual output of the simulation to that of the case-studies. Although this process certainly has its merits, and indeed did give a good indication for the range of valid or at least believable values, we now wanted to supply the simulation with more accurate attenuation values. We distinguished two major types of obstacles that would occur in our simulations, namely the housing wherein the metering-devices are placed, and the outside world, where it was decided that tree, plants and shrubs were likely to represent the major causes of attenuation.

Flora (Plants, Bushes and Trees)

On of the first things that we wanted to the simulation was Flora, and in particular trees. These are especially important in the modelling of the Goes area, where they are effectively responsible for splitting the network in two semi-independent sub-nets. Although we had simulated this by now, using the new obstacle-model, the attenuation value we used was arbitrarily picked to some high value.

We have performed a literature-study into the modelling of attenuation due to plants and trees. We found that although some work has been done in this area, but did not find much recent work. This does not pose much issue, after all a tree is a tree and the attenuation behaviour of a tree with respect to radio-waves will not soon change. However, much work focuses on very specific instances of attenuation. We note for example work that tracked the attenuation behaviour of a single tree or very small group of trees over the course of a year, detailing how the loss of leafs in fall and how the addition of falling-snow in winter affects signal propagation. We also found quite a few studies which investigated the attenuation properties of specific types of plants. What often was done was to position a sender and receiver on opposite sides of the subject (i.e. a tree or plant) at a fixed distance, and measure the attenuation of the signal, sometimes for a number of different frequencies.

The trouble with these types of research is that, although very valuable in their own right, these types of attenuation values cannot simply be interpreted in our situation. In our case we do not have enough information to model individual tree (we have to go by the information provided by the images of the areas provided to us by the protocol's developers). These images are hardly clear enough to distinguish individual trees, let alone to determine a particular tree's genus or height. In fact since we are looking at locations where the foliage is mostly concentrated in people's (back-) yards, a wide variety of tree and shrubs can be expected to be planted here. Therefore, what we are looking for is a more general description of the attenuation behaviour of various types of flora. Luckily we finally located a work which contained information on many types of tree and foliage, including average

values [76]. Surprisingly these attenuation values were surpisingly close to the values that we had originally worked out through experimentation. The values that we use are 1.3dB/m while passing through an obstacle (trees and shrubs), and 0dB for hitting an obstacle (threes and shrubs do not have a clearly defined 'wall').

Housing

Obtaining proper attenuation values for the housing in the simulations was equally difficult. Our situation seems to be fairly unique, in that we were interested in the attenuation that a signal undergoes as it passes from inside a house to the outside, and vice-versa. However most research efforts into defining the attenuation properties of housing and/or urban areas seems to focus on either the radio-wave propagation inside a room or house, e.g. 'office-conditions', or entirely outside of a house, e.g. to model the propagation of radio-waves for use in GSM-type applications. Attempting to locate information on the behaviour of radio-waves as they cross different types of (housing)-materials did not prove successful either, since here we faced a similar problem to the one we faced with the trees—we do not know the exact materials used in building the houses or flats in our scenarios. Again finding an average or expected value proved difficult. We did locate some interesting research into the behaviour of radio-waves crossing from under-ground into above-ground which may be relevant in future work, since these smart-metering devices are often placed inside the metering-cabinet, which can also be located under-ground [77–79].

Ultimately we decided to simply use the same attenuation values as the model that is under development for the MiXiM framework by Christoph Sommer. We use these values in good faith that they provide reasonable approximations to real-world attenuation values. The values that we settled on using are 0.1dB/m while passing through an obstacle (houses), and 5dB for hitting an obstacle (walls).

5.5 Verification of the simulation

Before we can use the simulation to perform an analysis of the protocol we must make sure that the simulation is a faithful representation of the real-world, and that the results that are produced by the simulation are reliable in the sense that they are what could be expected from the actual devices. We do so using a combination of visual inspections of the protocol's operation, as well as by performing a quantitative comparison to the operational data. We are in a unique position where we have access to packet-data captured from a real-world, working installation for the metering-devices, which can be used to verify the accuracy of the simulation. We discuss visual verification of the protocol in section 5.5.1, and our approach to a quantitative verification is detailed in section 5.5.2.

5.5.1 Visual verification

Visual inspections are performed on two levels. Firstly there is the normal inspection which can be performed using the OMNeT++ simulator simply by setting up a small network which should exhibit a known behaviour, and verifying that the nodes act as expected. Stepping through the simulated events it can easily be seen that nodes react to received packets in the correct manner. However such an approach quickly becomes infeasible as the number of nodes increases. Further more this approach does not allow one to see effects which play on a larger scale, affecting the entire network (or parts thereof).

In order to accommodate the visual verification of the protocol on a larger scale we have also created a tool for visualising the routes nodes use to communicate in the network. Paths in the network can be deduced from logs of communication activity, so long as the captured packets carry acknowledgement information. This way we can visualise both real-world communications using the logged data collected during the intermediate inspections using the RF-sniffer, as well as traffic paths in the simulated network. Figure 5.2 shows an example of visualising the various routes used by nodes

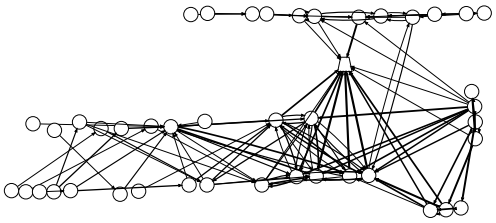


Figure 5.2: Visualisation of packet paths in the real-world.

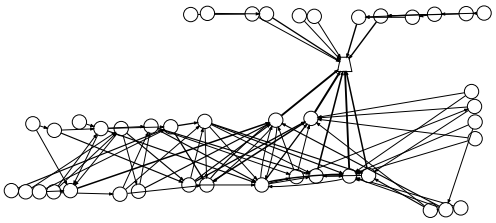


Figure 5.3: Visualisation of packet paths in the simulation.

during one of the intermediate inspections at the Goes scenario. Figure 5.3 shows routes used by nodes in one of our simulations. The output of figure 5.2 and figure 5.3 indeed look very similar. As can be seen the network is more or less split into a set of communicating nodes above and below the concentrator node. This split is because people's backyards are located in this area, as can be seen in figure 4.4. The backyards form a strip of trees and shrubs in between these nodes, blocking the wireless signal. The main reason for developing our obstacle model is to simulate the signal attenuation caused by this foliage.

5.5.2 Quantative verification

In order to perform a quantitative comparison of the simulated to the real-world behaviour, we have created a similarity index. This index is used to compare the routes used by nodes in the real-world versus those generated by the simulation. We have based our approach on Levenshtein edit-distance [80]. The Levenshtein distance between two text strings is defined as the minimum number of edits required to transform one string into the other, with an edit considered to mean the insertion, deletion, or substitution of a single character. The Levenshtein distance between a pair of strings thus is a measure of similarity.

Routes are not regular text strings, and due to the random selection of hops in the protocol a direct comparison of routes using a standard algorithm or metric such as Levenshtein-distance is poised to fail. We did attempt a direct application of Levenshtein-distance and various derivatives thereof, but we quickly found that these were unable to distinguish between the routes generated by the simulation, versus routes that were essentially completely random. The routes were similar in the sense as that when visualised the result is instantly recognisable as similar to the case-study scenario from visual inspection, whereas the randomised routes were not. The problem stems from the fact that these metrics take into account only the IDs of nodes, but not their geographic location. As such using an approach based purely on string edit-distance neglects the fact that nodes are more likely to transmit packets to nodes which are geographically closer. As an example consider that we are to compare a path $\mathcal{P} = [s,b,p,t]$ in the sniffed-data to a path generated by the simulation, and a randomised path. In this paths s and t represent the source- and destination-node, respectively, and can be considered 'fixed'. Let the path used by the simulation be $\mathcal{P}' = [s,d,q,t]$, with d and q neighbouring nodes to b and/or d, and the randomised path $\mathcal{P}'' = [s,x,y,t]$, with x and y nodes elsewhere in the network, but not in the neighbourhood of any of the nodes in $\mathcal{P}$. In this case metrics such as Levenshtein-distance consider $\mathcal{P}'$ and $\mathcal{P}''$ to be *equally different to* $\mathcal{P}$ (at distance 2). However it is clear that in a random protocol a path which passes through nodes which are close to the nodes in the original path is actually quite similar indeed. The problem is aggravated by the fact that also adding or removing a node from a path is considered equally different, too.

Therefore we have attempted to come up with a metric which acknowledges that a route can be similar in a geographic sense, rather than only considering similarities in the actual intermediate hops that is uses. The final iteration of the comparison algorithm on which we settled tries to compare paths in a way which not only takes the geographic location of nodes into account, but also attempts to take into account differences in the lengths of routes. We have dubbed it '2-Neighbourhood' for

want of imagination. The algorithm roughly works as follows. We assume that there is a upper-limit on the number of intermediate hops allowed. The protocol indeed satisfies this assumption, allowing at most 5 intermediate hops, thus there are a total of at most 7 nodes in a path. When comparing two paths we ensure that both paths are at the same length, 'stretching' both paths to length 7 by repeating hops as neccessary. We then iterate over all intermediate hops (e.g. *not* the source and destination nodes) in both paths, calculating for each node a in the first path, the the distance between a and the all nodes b in the second path, and the square of the distances of the nodes before and after b in the path. We take the sum of the minimum-value of each iteration as the value of the metric. Listing 5.1 shows the pseudo-code for this algorithm. In particular this listing shows how the algorithm performs the equalisation of the path-lengths.

```
|[  var A,B : 𝒫 → ν ; δ ∈ ν × ν → ℝ ; i,j,a,b ∈ ℕ ; s,score,l_a,l_b ∈ ℝ
 |  score := 0
    l_a,l_b = length(A), length(B)
    if  l_a › 3 ∧  l_b › 3 →
        i := 0
        do i < 7 →
            s := ∞
            j := 0
            do j < 7 →
                a := ⌊i · (l_a/7) + (2 · l_a)^-1⌋
                b := ⌊i · (l_b/7) + (2 · l_b)^-1⌋
                if  b > 0 ∧ b < l_b →
                    s := s ↓ δ(A[a],B[b−1])² ↓ δ(A[a],B[b]) ↓ δ(A[a],B[b+1])²
                fi
                j := j + 1
            od
            score := score + s
            i := i + 1
        od
    fi
    return  score
]|
```

Listing 5.1: Algorithm to calculate a similarity index for two paths A, B.

Table 5.1 shows the result of comparing the sniffed routes from case-study installation against the same sniffed routes, against a set of simulated routes, and against a set of randomly generated routes. This comparison is done against the Goes scenario. As can be seen for the purely edit-distance based approaches, it is very difficult to conclude whether or not the simulation is in any way more similar to the real-world than the random traces. Even when matching the real-world data to *itself* (i.e. when trying to see if the routes in the real-world sniffer-data could be generated in the real-world) the results not, as one might expect, zero. This again due to the random nature of the protocol, which means that for the same destination, routes may use different routes at different times. This is aggregated in this case because the sniffer-data that was used in this comparison was of a test of the devices where the data-concentrator node was put offline for an extended period of time. This represents a worst-case scenario for the edit-distance based approaches because now each node is almost guaranteed to have more than one path to the data-concentrator. However in the case of our similarity index it can be seen that indeed the real-data is self-similar, and also the simulated routes are quite similar, whereas the random routes are much more different. This indicates that simulated nodes are using routes which are similar to those used by the real-world nodes. Using our similarity index we are thus able to show that the simulated nodes are discovering and using routes which are relatively similar (with respect to the random nature of the protocol) to those used by the real-world

nodes.

Goes (Real) vs	Levenshtein [80]	Levenshtein/Dice [81]	Strike-a-match [82]	2-Neighbourhood
Goes (Real)	0.62 ± 0.56	1.60 ± 0.36	0.44 ± 0.47	7.37 ± 15.15
Goes (Simulation)	0.90 ± 0.30	1.44 ± 0.19	0.09 ± 0.23	11.76 ± 14.16
Random (small)[a]	0.97 ± 0.21	1.40 ± 0.14	0.02 ± 0.10	29.58 ± 36.79
Random (large)[b]	0.98 ± 0.20	1.39 ± 0.14	0.02 ± 0.08	29.59 ± 36.94

[a] Sample size (number of traces) was about equal to the sniffer-data.
[b] Sample size was considerably larger.

Table 5.1: Comparing various similarity indexes

5.6 Simulator performance and lessons learnt

We conclude this chapter by briefly reflecting on the performance of the final implementation of the simulation. All experiments were conducted on a single blade-type server. This server was fairly powerful, containing an Intel® Xeon ® X3430 CPU. This CPU has 4 cores running at 2.4GHz each. This meant that we could (and in fact did) execute up-to 4 independent simulation-runs in parallel, provided that sufficient memory and hard-disk space was available. This greatly reduced the total time required for doing the simulations, although it did require a significant amount of additional work on our part. Further the server was equipped with just 4GB of memory, and a relatively small (141GB) harddisk. Both proved, at times, to be a limiting factor in the amount of simulations that could be run in parallel. Especially the small size of the harddisk was troublesome at times, because the OMNeT++ simulator generates very large (up to tens of gigabytes) intermediate files over the course of a simulation run. The operating system that was installed on the server was the Ubuntu 10.04.1 LTS release (Lucid Lynx). This is not a typical server-OS, but seems to be a requirement of the OMNeT++ simulator. We attempted to install OMNeT++ on two other flavours of Linux distribution (namely CentOS and Gentoo), but we were unable to get the OMNeT++ simulator to work under these operating systems. It is not entirely clear why this is (since OMNeT++ comes with the full source), but we suspect that the simulation-editor environment, which is based on the Eclipse framework, which is in turn based on Java, requires Java Native Interfaces (JNI) to specific library versions which are not present, or rather the wrong version is present, on alternative Linux distributions.

The intermediate files generated by OMNeT++ are so large because OMNeT++ keeps a record of *every* event that occurs in the simulation, in order to provide a means of analysing this data later using the OMNeT++ analysis suite. This means that one must devise a way to process this while it is being generated, or otherwise run the risk that the disk fills up prematurely, ending a simulation run. We solved this by running an extraction process upon completion of each simulation, designed to keep only that data which was relevant to our analysis. This greatly reduced the amount of diskspace required, since we are not interested in many of the events that are recorded. For example we are interesting in all events involving a packet reception or transmission, but we do not care about the timer-events in the simulator.

Simulation itself was quite fast, often reaching speeds between 100 and 200 times real-time. As an example consider the case of one of the earlier experiments on the Goes area. In this particular experiment we tested the response of the protocol to different bit-error rates. We tested a wide range of bit-error rates, in total 38 different data-points were considered. In order to compute each data-point, the average result of 15 simulated 8 hour runs were used. Thus in total $38 \times 15 \times 8\text{h} = 4560$ hours worth of time was simulated for this experiment. The total time it took to run this experiment was just under 7 hours, or 650 times faster than real-time. However, recalling that we perform up-to 4 simulations in parallel, this means that each individual simulation was running, on average, at about 160 times real-time. The time to run a single iteration (which was done often during development of the simulation) was just a few minutes. In this experiment for example the running time of a single iteration would be just under three minutes. This meant that during development relatively much time

was spent waiting for the simulation to complete, and tracking down a bug, which might take many runs, took a long time. Once the simulation was more-or-less working properly we would therefore try to run full-length simulations over-night as much as possible, using the day-time to work on the simulation using shorter simulations.

The speed of simulation was however heavily dependent on the number of nodes, and in particular the number of connections between nodes. This is not entirely unexpected of course, but it did influence the running times for the simulations. For some of the larger experiments that we wanted to run this was becoming a limiting factor, because the simulation speed had dropped to just 10–20 times real-time (an order of magnitude slower). At this point we spent some time optimising the simulation. For example we implemented the pre-calculation of attenuation values for each pair of nodes. We also spent some time making sure that the simulation was working properly at higher levels of compiler optimisation, as well as tuning the compiler flags for the target machine. The end result was a significant increase in simulation speed, which now averaged around 70 times real-time in the heavier (more nodes and connections) scenarios. For less demanding scenarios this also meant an increase in simulation speed, which allowed us to perform more simulations. One of the largest simulations that we have performed in a single sitting was an experiment involving 15 scenarios, 20 different settings for the variable under study, with each setting repeated 15 times for an average measurement. Again the simulated time duration was set to 8 hours. Thus in this experiment 4500 individual simulations were performed, totalling 36000 hours worth of simulated time. Due to technical reasons we cannot give an exact measurement of the elapsed time, but we recall starting the experiment on a Friday, leaving the experiment to run over the weekend, with the experiment still running the following Monday-morning.

5.7 Conclusions

We have performed a brief literature-survey of available simulation packages, and stated our reasons for using the OMNeT++ simulator. We have discussed the architecture of the simulation, as well as some of the relevant implementation details. In particular we explain how we modelled the attenuation behaviour of obstacles in line-of-sight between nodes. We have presented our approach to verifying that the simulation accurately simulates the real-world circumstances, producing results similar to those obtained from the case-study installations. We have presented a novel algorithm for comparing routes used by nodes in the simulation and in the real-world, for the purpose of validating the simulation. We have presented the various experiments that we have carried out, the goals for each experiment and what we can learn from it, and how each experiment was set up. Finally we have given some performance figures for the simulation, as well as some of the technical challenges that we faced during our work on the simulation experiments.

Chapter 6

Simulation results

We have discussed the design and realisation of the simulation framework in chapter 5. We now present the results of the simulation studies we have performed to assess the protocol. We first discuss the setup and goals of the scenarios we have used in our simulation experiments in section 6.1. Broadly we have performed two types of experiments, each with two distinct assessment goals. We present the results of the first, experiments using real-world topologies, in section 6.2. The results of the experiments using random topologies are presented in sections 6.3 and 6.4. In the interest of brevity we only detail the results on the Goese Polder. Next we review some miscellaneous findings in section 6.5, which are not as structured as the other results, but interesting nonetheless.

6.1 Simulation scenarios

After the simulation was completely implemented, verified to operate correctly, and generating results consistent with the real-world scenarios, we designed and carried out a number of different experiments. In this section we describe the setup and goals of each experiment. We present the results of these experiments in chapter 6. Each experiment ran over one or more physical layouts, e.g. either a single setting such as Goes or Den Bosch, or a number of randomly generated environments, where metering-nodes and obstacles are placed randomly into the environment. In order to come to a balanced result every experiment was run a total of 15 times, and the average of these 15 iterations were calculated. Each experiment simulated the operation of the protocol over an eight-hour period, including the initial start-up of the protocol. This is sufficient time to allow the protocol to settle into a stable state and gather information on all stages of the protocol.

6.1.1 Simulation of the real-world setup

The simulations of the real-world served a two-fold purpose, as discussed before. The first goal was to make sure that the simulation was capable of replication the results from the case-studies. The second goal was to assess how the protocol would react to changes in the environmental conditions. In order to be able to make a fair comparison of the results, here the environmental conditions refer to the conditions for radio-wave propagation, and not to the layout of housing and obstacles or the attenuations thereof.

Verification phase

The first simulations involved simulating the real-world scenario in Goes in order to assess the performance of the protocol under varying operating conditions, using the known environments as a baseline for varying the transmission conditions. We used both of the available real-world scenarios to verify the simulation, however our tests indicate that our simulation of the Goes scenario was more accurate. The reasons for this are presently unknown. An attempt was made to perform an inspection

of the site, in an effort to determine why the scenario in Den Bosch was not matching the real-world as well as the scenario in Goes. Many theories were proposed, but due to administrative reasons we did not get to investigate this further. Therefore we decided to focus on the Goes scenario for the experiments. Figure 6.1 shows the model of the Goes area in the simulator, cf. figure 4.4. The brown rectangles represent the housing, the green rectangles represent areas with dense vegetation, and the grey area represents a combination of garden-sheds and shrubbery, which we assumed to have different attenuation properties to the areas containing only plant-life. The settings used for these simulations were as close as possible to the real-world. That means using average attenuation values for the houses and trees, and a BER of $3.2 \cdot 10^{-06}$. This is the BER that according to the development team of the hardware-nodes was measured during the case-studies.

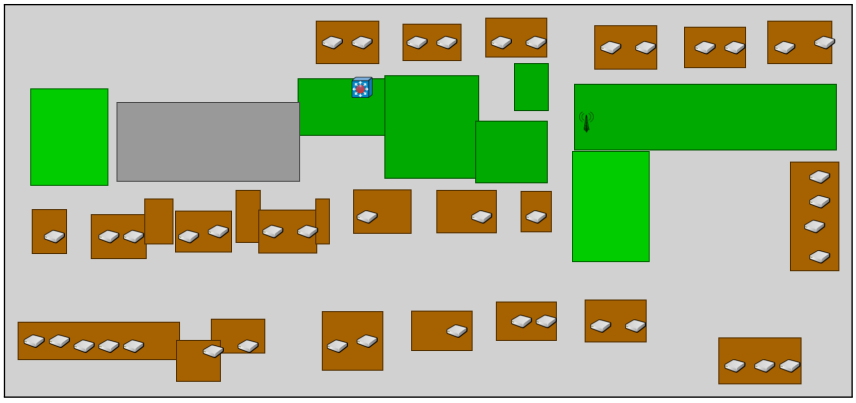


Figure 6.1: The Goes area modelled in the simulator.

Experiment 1

In the first experiment the channel-state (bit-error rate) was varied between extremely good ($\mathcal{P}_{\text{Error}} = 1^{-9}$) and extremely poor ($\mathcal{P}_{\text{Error}} = 7.75^{-1}$). The goal of this experiment was to investigate the protocol's ability to deal with and recover from packet loss by various causes, either from outside interference or poor equipment. This experiment thus aims to see what happens when for example other equipment is active on the same frequency band, and what level of interference the protocol can tolerate. This experiment can also be used to investigate what happens when environmental conditions cause reception errors, such as for example caused by microwave radiation or electrical discharges during a thunder-storm.

A range of values was chosen to cover as large as possible a range of bit-error rates whilst still completing the simulations in a reasonable amount of time. In total 25 different bit-error rates were simulated. In order to get the best spread for graphing the results, the BER values (which would become a data point on the plot each) were chosen according to the following formula.

$$\mathcal{P}_{\text{Error}}(x) = 10^{-\frac{x}{4}} \text{ for } 12 \leq x < 37$$

Additionally we tested the protocol's behaviour when $\mathcal{P}_{\text{Error}} = 0$ and $\mathcal{P}_{\text{Error}} = 1$. The other parameters were kept the same as during the verification phase.

Experiment 2

The second experiment that was carried out varied the number of active nodes. The goal of this experiment was to investigate the robustness of the protocol and its ability to deal with node failure, e.g. to route around 'gaps' in the network. Such gaps may be caused by equipment malfunction, or by changes in the environment which (temporarily) cause attenuation towards particular neighbour nodes to increase dramatically. For this particular experiment we varied the percentage of active

nodes between 100% and 10%, in steps of 2.5%. Below 10% active nodes insufficient nodes remained active to sustain a viable mesh-network.

6.1.2 Simulation of random scenarios

After simulating the Goes scenario we wished to investigate the performance of the protocol under unknown environmental-conditions, to predict how the protocol may perform on different topologies.

Experiment 1

This actually consisted of two sub-experiments, based on the simulations done for the real-world as descibed above. What we did was to repeat the above two experiments, i.e. the bit-error and node-failure rates were again varied, but now using a *randomised representation* of the Goes scenario. By 'randomised representation' it is meant that the random scenarios used were generated in such a way as to have similar properties to the Goes scenario. That is to say, the random-scenarios were generated such that each scenario had the same number of nodes (metering-devices) as the Goes scenario, placing the metering-devices with similar inter-device distances, and giving obstacles in the simulation similar properties (i.e. attenuation values for the walls and interior regions) to the obstacles in the Goes scenario. An obstacle was placed over each meter-device was to represent the houses, and patches of vegetation scattered in between that in proportion to the vegetation present in Goes. The nodes were placed an a area of equal size (width and height wise) as the real-world, e.g. 167m by 89m. The placement of nodes was random, so there are no regular patterns (e.g. placement of nodes along a street), but the variation in inter-node distances is similar to that in the real scenario. Once the nodes have been placed a portion of the surface area is covered by 'flora'. Figure 6.2 shows one such a randomised scenario. The goal of this experiment was to see whether the results obtained in the Goes simulation would hold true for similar scenarios. This would re-enforce the validation of our simulation, since it is reasonable to expect similar results for similar scenarios.

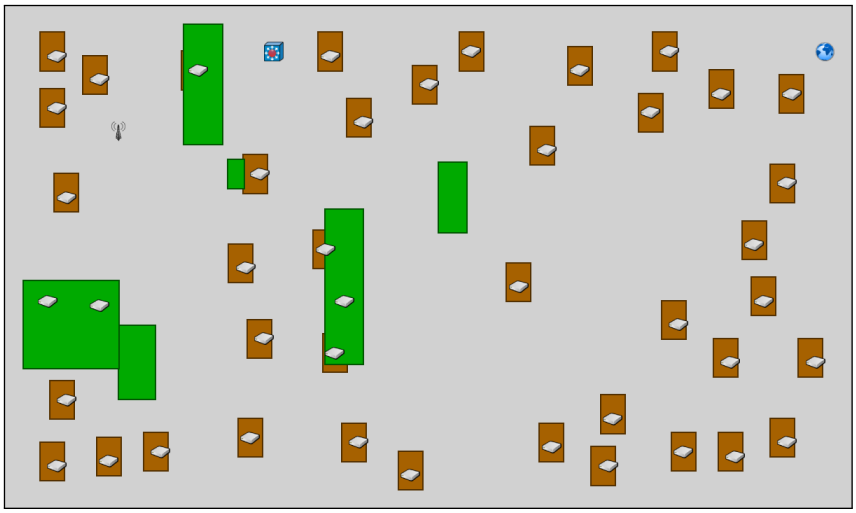


Figure 6.2: A randomised representation of the Goese Polder scenario.

Experiment 2

The final set of experiments consisted of again the same type of experiments as before, e.g. an experiment wherein the bit-error rate is varied in order to assess the ability of the protocol to deal with packet losses, and a second experiment wherein the number of active (or enabled) nodes is varied in order to assess the ability of the protocol to deal with nodes failing. The difference with

respect to the previous random experiment is that we now generated a large number of completely random terrains, i.e. now all aspects of the environment were varied. The number of metering-devices was varied between 10 and 100 an the size of the simulated area was varied between a very small $64\text{m} \times 64\text{m}$ patch of land, up-to an are one square kilometre in size. We attempted to fill a 'reasonable portion' of the environment with plants. The size of the housing was also varied within reasonable parameters, ranging from very small 8m by 5m houses as in the Goes scenario, to very large 700m^2 buildings. For each iteration of the simulation (recall that each data point is simulated up-to 15 times) a different scenario was used in order to come the average behaviour of the protocol over a large range terrain types. Examples of these types of random environments are figures 6.25 and 6.26. Figure 6.26 is a model of a small, densely populated area with relatively much greenery, figure 6.25 represents a much larger, wider area. The goal of this final set of simulations was to give some predictions regarding the protocol's general, overall behaviour. E.g. a prediction as to what can generally be expected for the protocol's performance given various conditions.

6.2 Experiments using real-world topologies

The experiments with real-world topologies were performed using the topologies of the *Goese Polder* in Goes and of the *Binnenstad* in Den Bosch. These topologies were modelled as accurately as possible in the simulator, and then used first to verify that the simulation results were accurate, and secondly to see how the protocol behaves under varying conditions in these scenarios. We have already discussed the results obtained for both the Goese Polder and the Binnenstad when we detailed the verification of the simulator in section 5.5.2, so we omit those results here. We have performed two experiments using the topology of the Goese Polder area. In the first experiment the bit-error rate of the channel was varied, in the second experiment the number of nodes participating the routing-protocol was varied. We will only briefly recapitulate the gist of each experiment and focus on the results, as the goals and setup for each experiment have been discussed in section 6.1. Each plot in this section 6.2 show both the value of interest as well as the standard deviation of the result. The standard deviation is plotted a shaded area surrounding the plot-line.

6.2.1 Sensitivity to bit-error rate

By varying the bit-error rate of the channel an estimate can be made of the protocol's ability to cope with interference from outside-sources which may be sharing the same channel. The chance on a bit erroneously being received in these experiments was varied between 0 and 10^{-3}, as described in section 6.1.1. Recall that the typical bit-error rate as encountered in the field experiments was $3.2 \cdot 10^{-06}$.

One of the key requirements of the protocol is that it should ensure that the duty cycle for each node remains below the 1% mark at all times. In figure 6.3 the duty cycle of the data-concentrator and metering nodes is plotted. As can be seen in this figure, neither the data-concentrator nor the metering-nodes exceed the 1% duty-cycle limit. Because it is the communication target of every node in the network the data-concentrator uses substantially more of the available bandwidth, reaching a duty-cycle almost 5 times higher than any other node in the network. However even at very high bit-error rates (and thus a maximum of required retransmissions), even the data-concentrator's duty-cycle does not exceed even one tenth of the imposed limit, staying well below 0.1%. Especially at a bit-error rate of $3.2 \cdot 10^{-06}$ the duty-cycles are within the limit . It can be observed at the very extreme end of the tested bit-error rates that whereas the graph shows a large increase in the duty-cycles for metering-nodes, the concentrator's load decreases slightly. This is because at this point the channel conditions are so poor that nodes are no longer able establish even one-way communication with the data-concentrator.

Figure 6.4 shows the combined delivery rates of gas-, water-, and electricity-measurements, as well as reverse-energy measurements. As can be seen the protocol's ability to deliver measurements to the concentrator node is largely unaffected by bit-error rate. Only at high bit-error rates does the

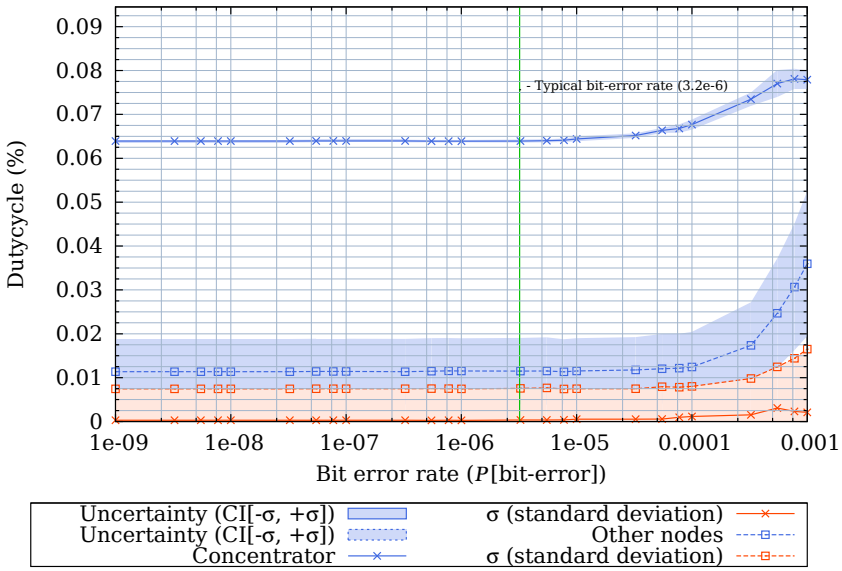


Figure 6.3: Duty-cycles in Goes under varying bit-error rates.

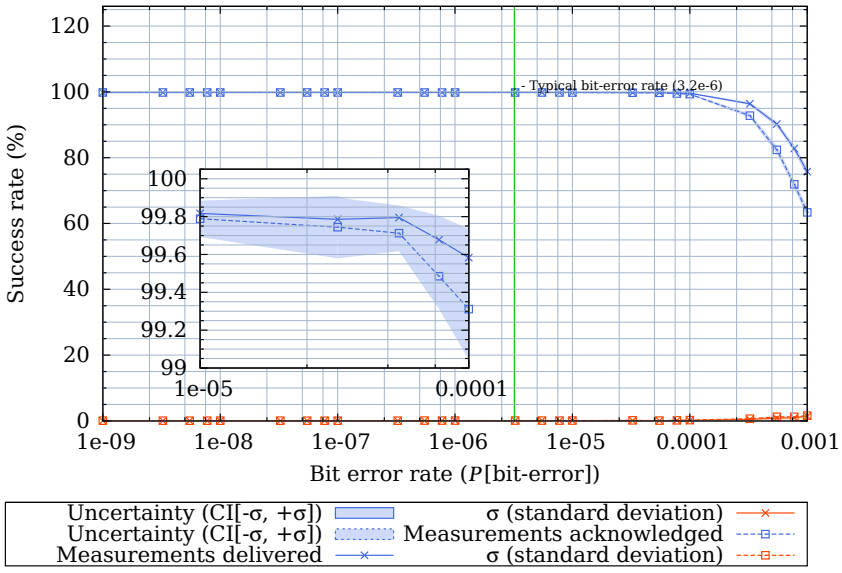


Figure 6.4: Delivery rates in Goes under varying bit-error rates

packet-loss rate start to negatively affect the protocol's ability to reliably deliver packets. At the bit-error rate observed in the real-world conditions the protocol works almost as well as under ideal conditions. Even at a bit-error rate of 10^{-4} the protocol still delivers more than 99.5% of all measurements to the concentrator. It can also be seen that even though a node may itself believe to have failed to deliver the measurement due to not receiving an acknowledgement, even at very high bit-error rates of $5.5 \cdot 10^{-4}$ (expected packet loss rate at 314 bits per packet $(1-(1-(5.5 \cdot 10^{-4}))^{314}) \cdot 100 > 15.8\%$), still more than 90% of all measurements are delivered at the concentrator. This shows that the protocol continues to operate robustly even in the presence of high-packet loss rates, and can be expected to operate even in noisy environments.

Figures 6.5 to 6.7 show three measures for the length of the routes. Figure 6.5 shows the average

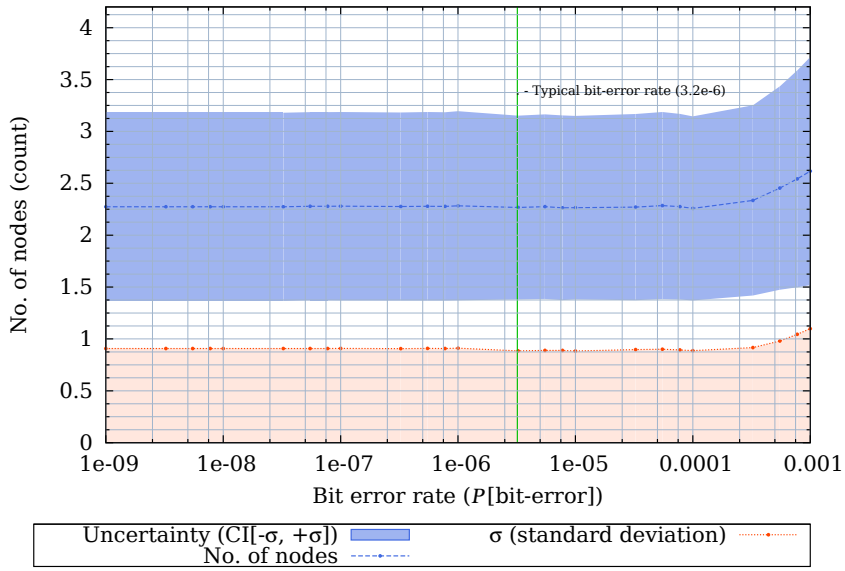


Figure 6.5: Path-length (in no. of hops) in Goes under varying bit-error rates.

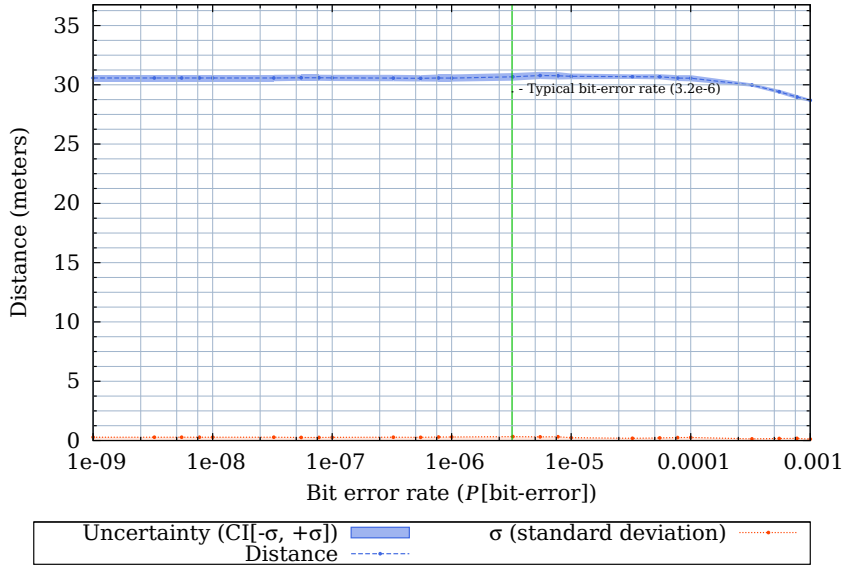


Figure 6.6: Mean transmission distance (single-hop) in Goes under varying bit-error rates.

number of hops for each route, figure 6.6 shows the average hop-to-hop transmission distance and figure 6.7 shows the average distance covered by a route. The average transmission distance and route lengths is measured in meters. The hop count is the number of (re)transmissions of a packet. This shows that routes are fairly unaffected by changes in the bit-error rate. The routes that are used are fairly constant in length, although there is some variation in the number of hops between individual hops. Only at very high bit-error rates a change behaviour can be observed. It also shows that, counter-intuitively, at higher bit-error rates nodes tend to discover communication paths that are longer, e.g. seemingly less direct. This is because each individual hop covers a shorter distance.

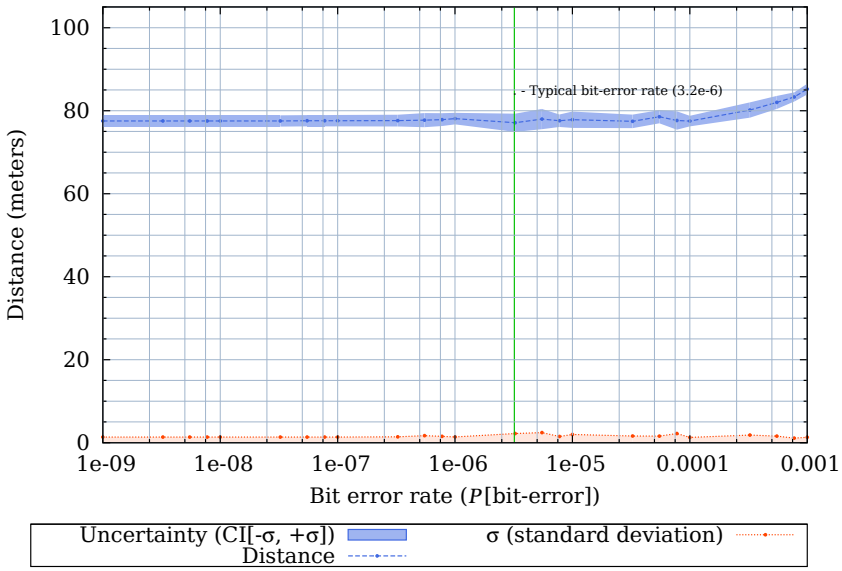


Figure 6.7: Mean transmission path length (multiple-hops) in Goes under varying bit-error rates.

6.2.2 Sensitivity to number of active nodes

By varying the number of active nodes in the simulation we can simulate nodes failing over time. This gives some insight into the response of the protocol to losses in connectivity due to node failure. The percentage of active nodes was varied between 100% and 0%.

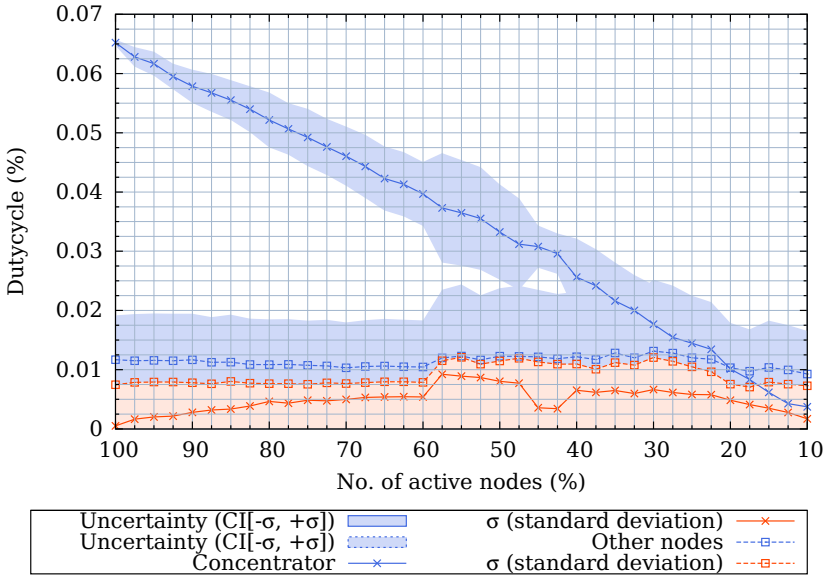


Figure 6.8: Duty-cycles in Goes under varying node availability.

Figure 6.8 shows that, as the number of active nodes decreases, so does the load on the data-concentrator. This is be expected, since the number of acknowledgements that need to be send by data-concentrator decreases. Interestingly the load on the remaining nodes remains mostly stable initially, but when the number of nodes remaining in the network drops below 60%, there is a small, but noticeable increase in the load on the remaining nodes. A possible explanation for this is that at this point the network is at a 'breaking point'. In the network certain nodes may function as an important intermediate node offering the best route to the concentrator. At this point the chance that a number of such nodes fail at the same time greatly increases. This causes nodes to consider alternative routes, or in the worst case a partitioning of the network. Thus some part of the network may no longer be able to reach the data-concentrator, leading to excessive, futile, searches.

In figure 6.9 again the packet delivery rate is plotted. This shows the same pattern as before. Measurement delivery-rates remain fairly stable until only 60% of the nodes remain active. Between 100% and 60% active nodes, on average between 99.8 % and 96.7% of measurements are delivered at the data-concentrator. The reason why we do not achieve a 100% data delivery rate even when all nodes are active is probably because of the bit-error rate, which as we discussed before is still quite high in this scenario. Nevertheless, the protocol is able to bypass the defective nodes with an efficiency that drops only down to 97.7%. When 60% of the nodes remains active the network remains functional in most cases (standard deviation 13.2%). Also in present in this figure is a transitional region at 60%. In this case it is present in the form a small drop in the average succes-rate ($97.7\% \rightarrow 93.3\%$), and a substantial increase in the uncertainty with which this level of performance can be met, increasing almost two-fold to 23.2%. Beyond this point performance degrades rapidly.

Figures 6.10 to 6.12 depict how the number of available nodes influences the length of the routes. As can be seen the routes are very similar in terms of distance covered and number of hops used to those in the bit-error rate experiment (figures 6.5 to 6.7). The distances and hop-counts remain fairly constant up-until the point where more than half of all nodes are offline. This shows that even nodes that are relatively far away from the data-concentrator still stand a good chance of communicating their measurements, even when an extensive portion of the network fails.

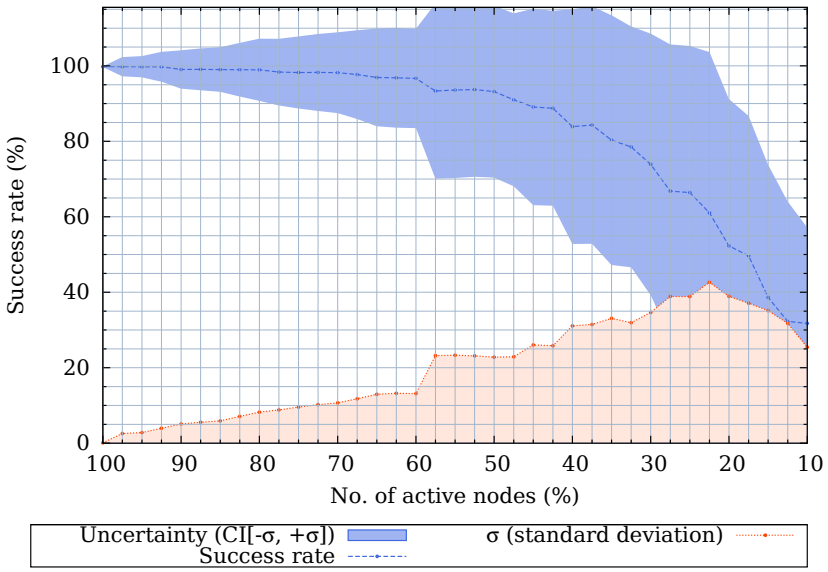


Figure 6.9: Delivery rates in Goes under varying node availability (considering data that is both delivered and acknowledged).

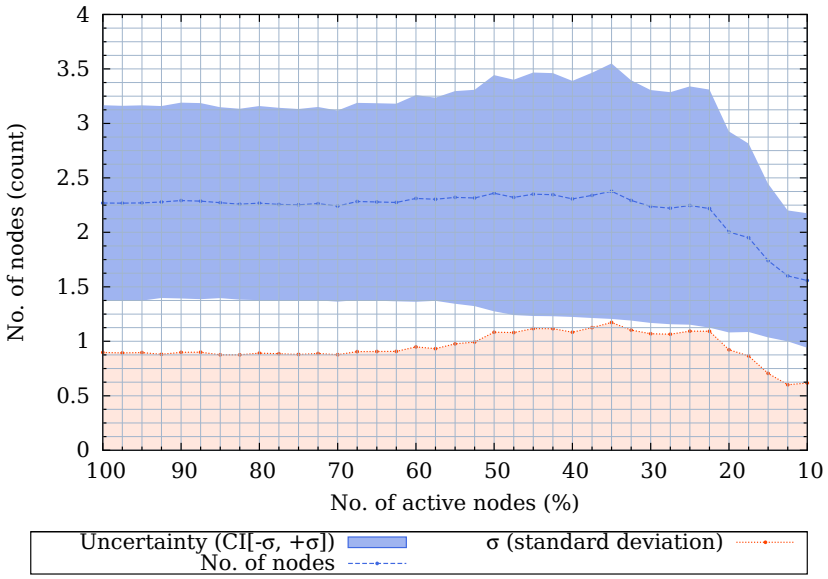


Figure 6.10: Path-length (in no. of hops) in Goes under varying node availability.

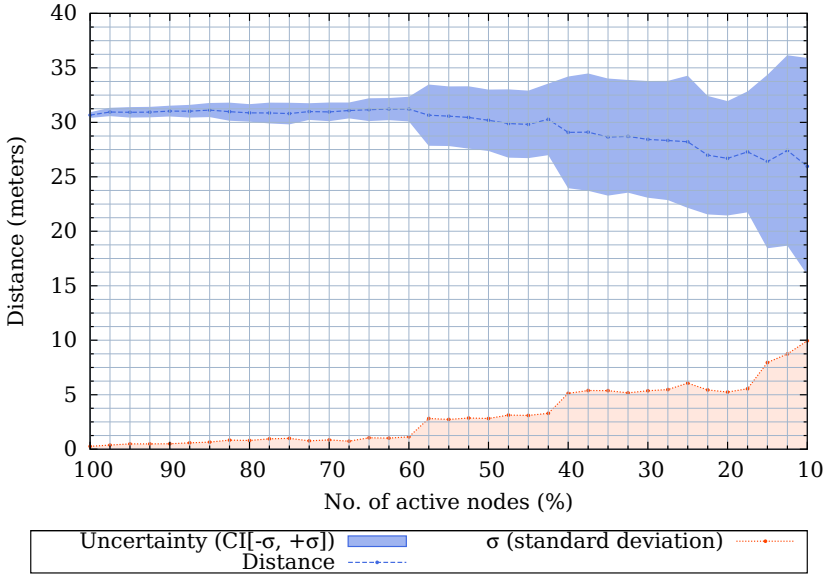


Figure 6.11: Mean transmission distance (single-hop) in Goes under under varying node availability.

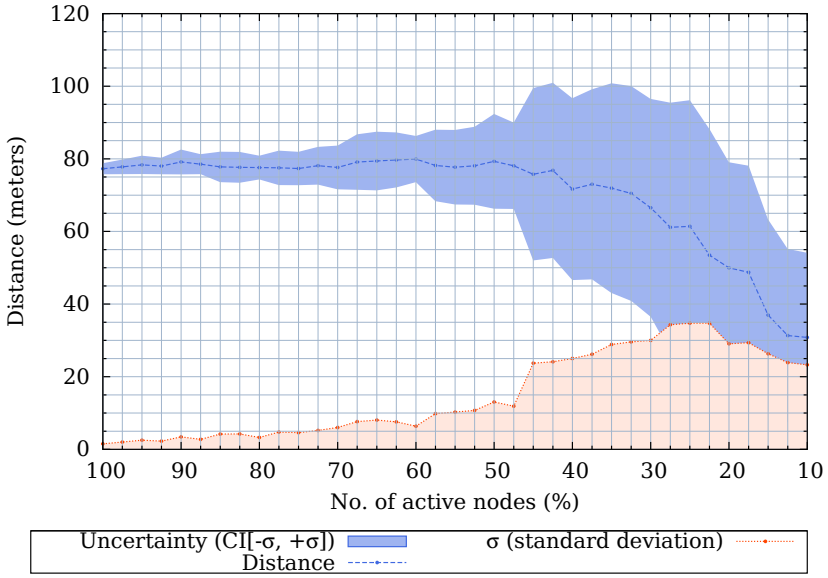


Figure 6.12: Mean transmission path length (multiple-hops) in Goes under under varying node availability.

6.2.3 Conclusions

We previously established in section 5.5 that the simulation of the Goes scenario is a faithful representation of its real-world counterpart. In this chapter we use simulations to show that the protocol is able to achieve a high level of reliability on a large range of bit-error rates, delivering over 99.8% of measurements under reasonable channel conditions. This is in line with field measurements. The protocol is able to achieve such a performance without exceeding the maximum allowed amount of bandwidth and transmission time, staying far below the 1% duty-cycle limit. Lastly we found that in the Goese Polder scenario the protocol is far from reaching the limit on the number of hops, using little more than 2 hops on average. From our experiment on the tolerance to node failure we additionally conclude that the protocol does also not exceed the duty-cycle limitation when node failures force the use of additional, possibly longer, routes involving more hops. The protocol is resilient against node failure, and continues to operate reliably even when large portions (up-to half) of the network are offline.

6.3 Experiments of randomised real-world-like scenarios

The experiments using random topologies were performed using two different sets of random topology types. The first set is based on the characteristics of the Goese Polder area, and is meant to represent sub-urban environments. The goal of this experiment was to investigate how the protocol would behave under environmental conditions similar to those in the Goese Polder. To this end the placement of nodes and obstacles was randomised, but the parameters for each were set to similar values as used in the Goes model. We here present the results of an experiment where the number of available nodes was varied using the first set of topologies. We use the Goese Polder data as benchmark.

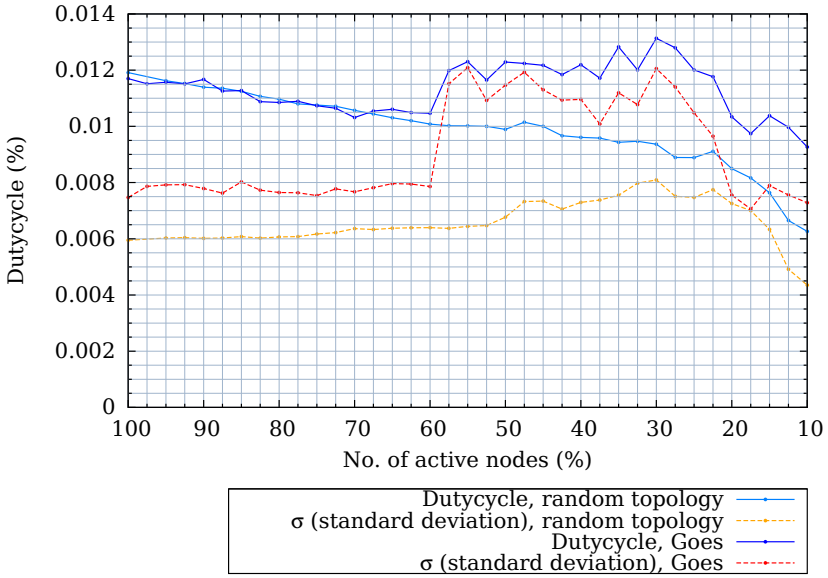


Figure 6.13: Duty-cycles for metering-nodes in randomised real-world-like topologies vs. Goes under varying node availability.

Figures 6.13 and 6.14 compare the duty-cycles in the resilience experiment between the original simulation of the Goes scenario and its randomised topology. We show the duty-cycles of the metering-nodes (figure 6.13) separately from those of the data-concentrator (figure 6.14), in order to show that the decrease in overall load is not due to the decrease in load on the data-concentrator, but to a decrease in load throughout the network. This also allows us to better show the behaviour of the

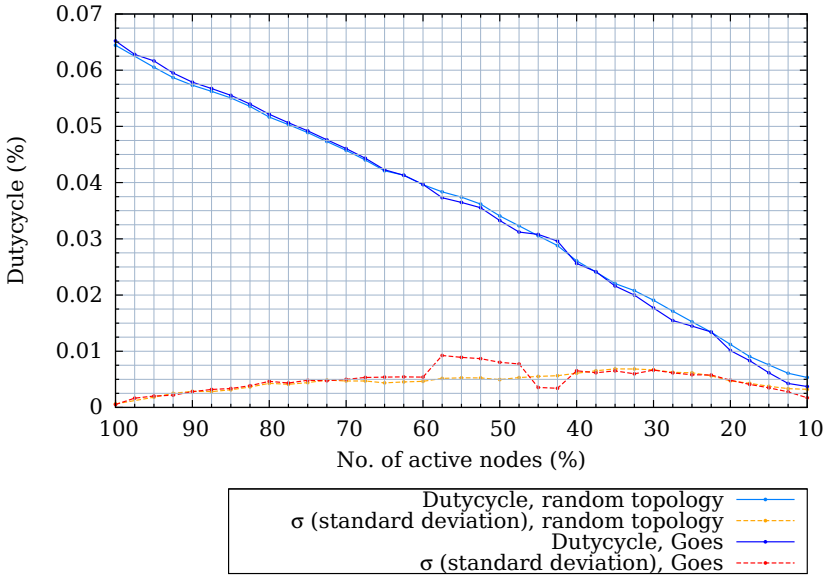


Figure 6.14: Duty-cycles for the data-concentrator in randomised real-world-like topologies vs. Goes under varying node availability.

metering-nodes (cf. figure 6.8). It can now be seen that the metering nodes show a slight decrease in load too, whereas this was not apparent in the combined plot. In the case of the original experiment we noticed that the duty-cycles increased dramatically after about 40% of the nodes were rendered inactive. We speculated that this might be related to particular topology of the scenario. E.g. if some critical connection could no longer be reliably made due to an insufficient number of nodes being active, this would force the remaining nodes to route around the affected area. This increases the length of the paths used for communication, which in turn requires more nodes to transmit more message, thus increasing the load on the network. As can be seen in figure 6.13 this effect is indeed limited to the particular topology of the Goes scenario, and disappears when averaged over a large number of scenarios. Futher it can be seen that for the initial part of figure 6.13 (between 100% and 60% active nodes), and for the whole of figure 6.14, the simulations of the randomised Goese Polder scenario match very well with the simulation of the Goese Polder scenario.

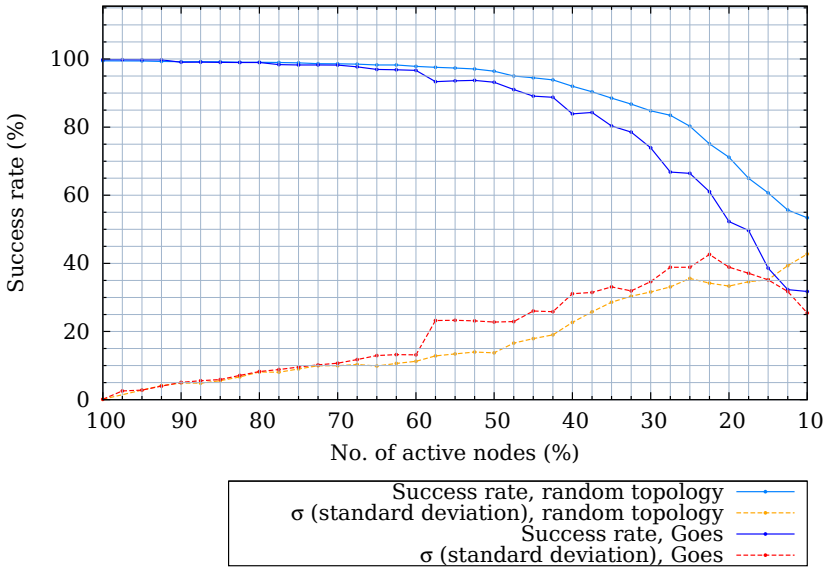


Figure 6.15: Delivery rates in randomised real-world-like topologies vs. Goes under varying node availability.

Figure 6.15 shows how the delivery rates of measurements to the concentrator node differ between the two experiments. This shows that the delivery rates in the Goes scenario behave almost identical to comparable scenarios up to the point where the Goes scenario was becoming vulnerable to disconnections in the connectivity of the network (around 60% remaining active nodes). It shows that under different circumstances the protocol can maintain good connectivity (still delivering ›95% of measurements) even when only half of the nodes remain active. Beyond that point the protocol continues to deliver more measurements to the data-concentrator. This shows that in situations with similar environmental conditions to the Goese Polder scenario the protocol can maintain good connectivity even after many nodes in the network have failed, provided that the topology is sufficiently rich to support the network. An issue in the Goese Polder is that the network is already almost partitioned at the data-concentrator (recall section 5.5.1, figure 5.2).

Figures 6.16 to 6.18 show that even though the metering devices were carefully placed with similar inter-node distances, the average per-hop distance and total path lengths are slightly bigger than they were in the Goes scenario. There are two possible contributing factors for this. First (this is probably the main contributing factor) we have used a slightly large inter-node distance in the generated scenarios than we used in Goes. This was necessary because the random nature of the generated topologies was causing some of the obstacles used to represent the housing surrounding the nodes was overlapping. Therefore we added a minimum distance between the nodes. However the mini-

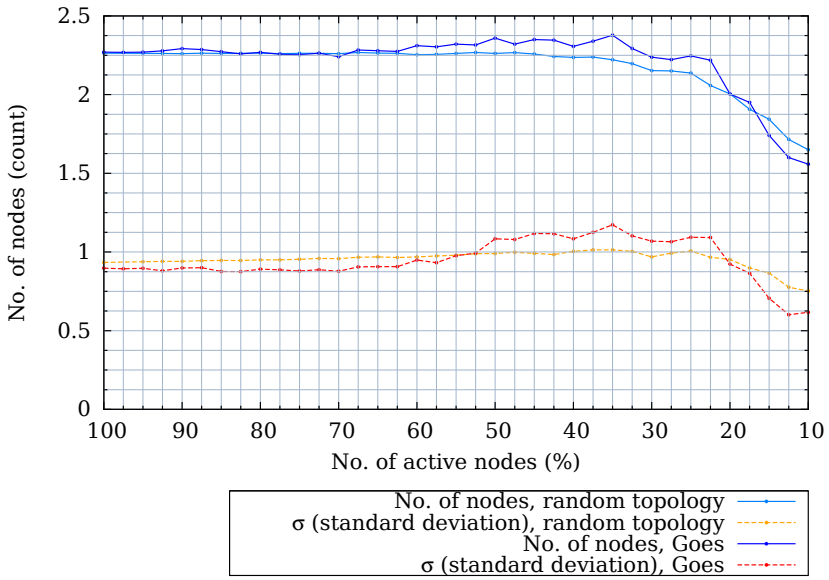


Figure 6.16: Path-length (in no. of hops) in randomised real-world-like topologies vs. Goes under varying node availability.

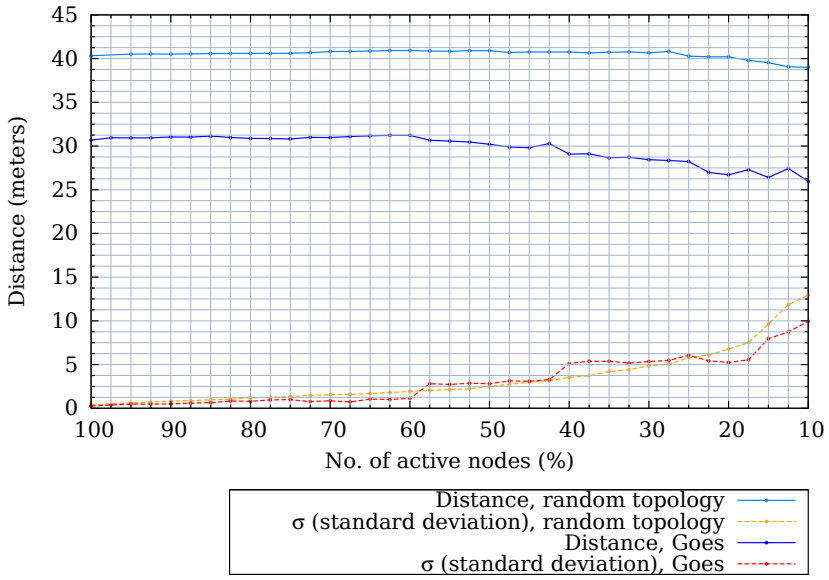


Figure 6.17: Mean transmission distance (single-hop) in randomised real-world-like topologies vs. Goes under varying node availability.

mal required distance to avoid overlap was bigger than the smallest distances in the Goes scenario ($\pm$10m vs. $\pm$6m). The distances between nodes in the Goes scenario are often quite small because many of the houses are semi-detached. However our tool, at present, only generates detached housing. A second possible contributing factor is that in the Goes scenario the nodes were placed in a rigid grid-like formation, since they are placed inside of houses in a neighbourhood. Likewise the obstacles which were placed were meant to represent the various flora in (back)gardens. The way the randomised topologies are generated is just that, random, without any specific order to where nodes or obstacles are placed. This means that in the randomised scenario a node is likely to find less inter-

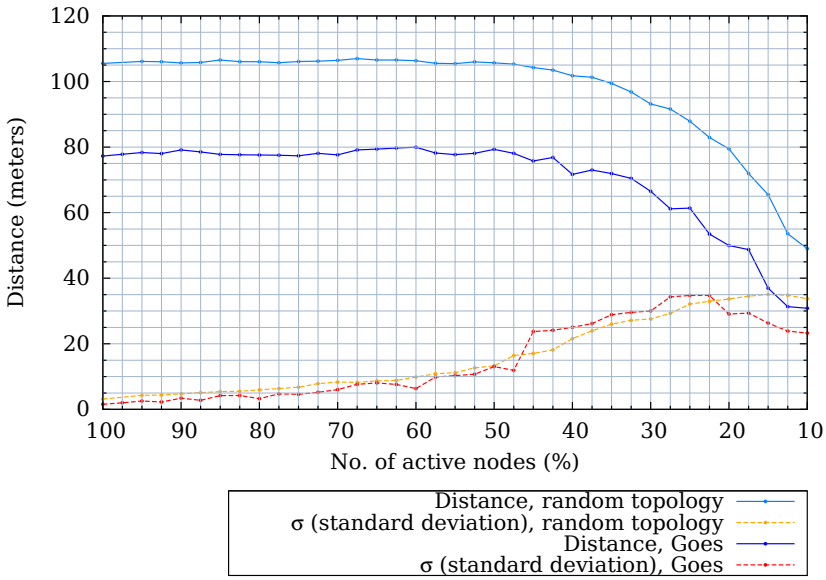


Figure 6.18: Mean transmission path length (multiple-hops) in randomised real-world-like topologies vs. Goes under varying node availability.

mediate nodes (which have obstacles representing housing surrounding them) on its communication path, and hence on average nodes are able to communicate with nodes which would normally be out-of-range. This indicates that path lengths are more a function of the positions of nodes than of any direct influence of the protocol. As can be seen the number of hops does follow the same pattern in both the Goes scenario and its randomised approximations, indicating that the number of hops is a function of the protocol and is independent of the physical distribution of the nodes.

6.4 Simulation of genuinely random topologies

The final experiments that were performed were aimed at assessing the performance of the protocol on a variety of widely different scenarios. These environments range‘ from small, densely populated areas to spread out topologies. The parameters for the size and composition of buildings and plant-life were varied much more than before. This allows to make some predictions on the expected behaviour of the protocol before it is deployed in new surroundings. We again we benchmark against the Goese Polder data.

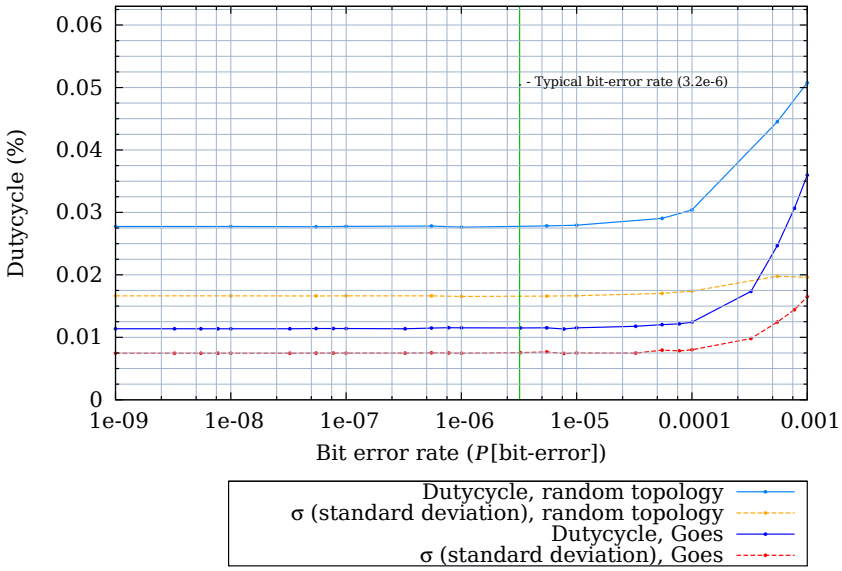


Figure 6.19: Duty-cycles for metering-nodes in random topologies vs. Goes under varying bit-error rates.

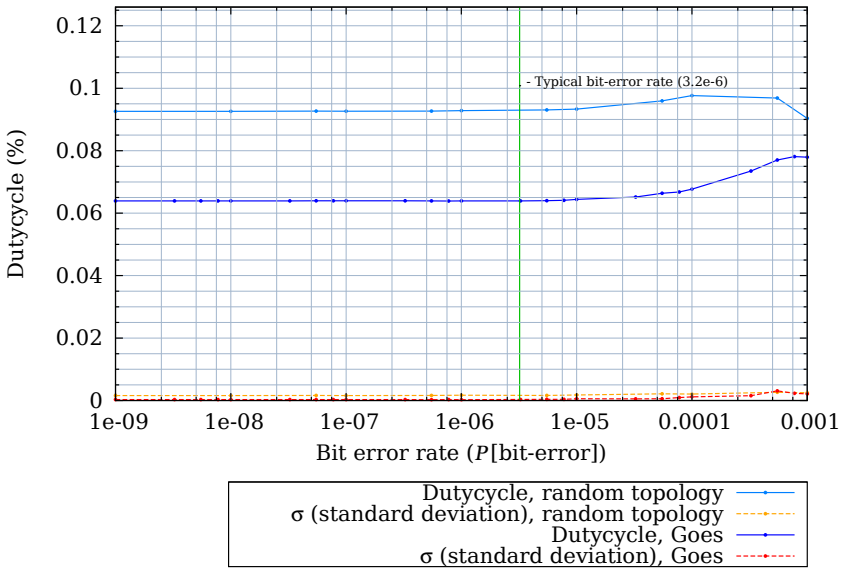


Figure 6.20: Duty-cycles for the data-concentrator in random topologies vs. Goes under varying bit-error rates.

Figures 6.19 and 6.20 show that the most active node's (the concentrator node) duty-cycles are on average still far below the imposed 1% limit, as required. In fact the highest recorded duty-cycle level for a concentrator in any of the scenarios in this experiment was still less than 0.14%. We know from the previous experiments that a reduction in the number of active nodes will mostly decrease the concentrator node's load, but that occasionally there is an increase in load. However the additional amount of load is generally relatively small compared to the normal duty-cycles levels. That is to say that, although the relative increase in the duty-cycles may be quite substantial (as in figure 6.13, where there is an almost 15% jump in dutycycles around 60% active nodes), the additional increase beyond the initial duty-cycle level of 0.0129% is only 2%. Similar results have been observed in various of the simulated areas. Therefore it is reasonable to conclude that the duty-cycle limit will never be exceeded, because enough leeway remains before hitting the 1% duty-cycle mark.

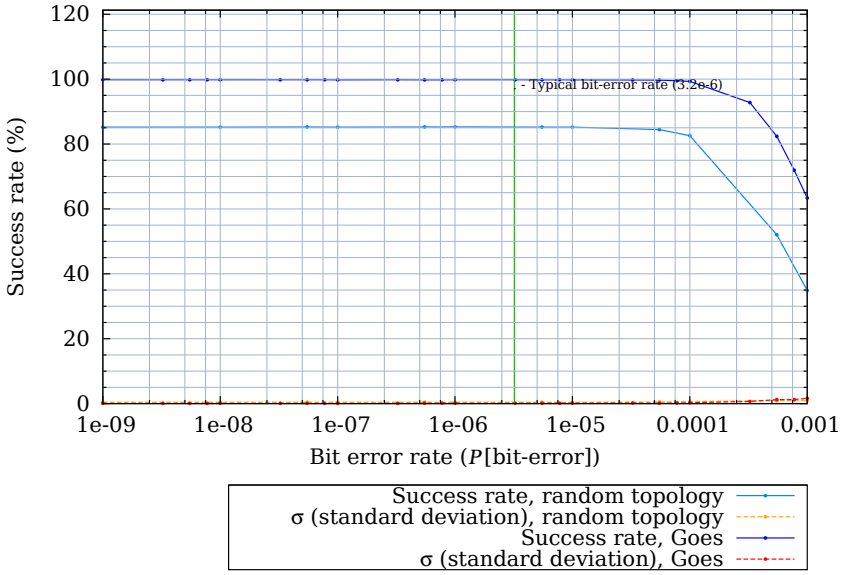


Figure 6.21: Delivery rates in random topologies vs. Goes under varying bit-error rates.

In figure 6.21 the delivery-rates in the Goes scenario and the average delivery-rates in the random scenarios are plotted. As is readily apparent from this figure, the delivery rates in the random-scenarios are not nearly as good as in the Goese Polder scenario. However as it turns out the reason for this is both an issue with the generation of the random topologies used in this scenario, as well as with the protocol.

The first problem is that due to the placement of nodes in larger areas, occasionally some nodes and obstacles will get placed in such a way that the nodes are prevented from communicating. An example is when a particularly dense patch of plant-life is placed between nodes. There are currently no provisions to detect and prevent this from happening, as this would entail re-implementing (part) of the simulator logic in the scenario generator. This did not occur in the simulations of the Goes scenario because of the dense placement of nodes, ensuring availability of routes, and the (usually) lower attenuations caused by housing and plant-life. For the generation of the scenarios in this experiment a range of attenuation values around those found in the Goes area was used. This means that both the attenuation due to walls, as well as due to a patch of greenery can be significantly higher.

The second issue is that some of the generated scenarios exceed the capabilities of the protocol in terms of the number of hops required to connect all nodes to the data-concentrator. For example, some of the generated scenarios present extremely elongated topologies, where nodes are placed in a narrow (64m) strip, several hundred meters long. In these cases the 5-hop limit ensures that a good portion of all nodes cannot connect to the data-concentrator.

Therefore we unfortunately can not make a strong conclusion about the performance of the proto-

col on genuinely random settings, other than to conclude that the plot does follow the same general form of the experiment done on the Goes scenario, and thus would show a similar tolerance to bit-error rates in the general sense. By super-imposing the two plot-lines, it can be shown that in this experiment the performance of the protocol was slightly worse, have a lower tolerance to bit-error rates. This does show the importance of proper placement of nodes, and influence of the environment. The fact that the protocol, on average, did not perform as well, or rather, experienced a degraded performance more quickly, suggest that the topology does play a role in the performance of the protocol. Not all environments are equally well supported. It is thus worth-wile to investigate any potential installation area for possible signal-obstructions, and carefully consider the placement of metering-nodes, as well of the data-concentrator. In some cases some kind of repeater might be required, or the output power of the nodes could be increased. The current model is based on a transmitter capable of at most 10mW output. However regulations allow up-to 25mW of output power, which should significantly boost the range of the devices, alleviating some of these issues.

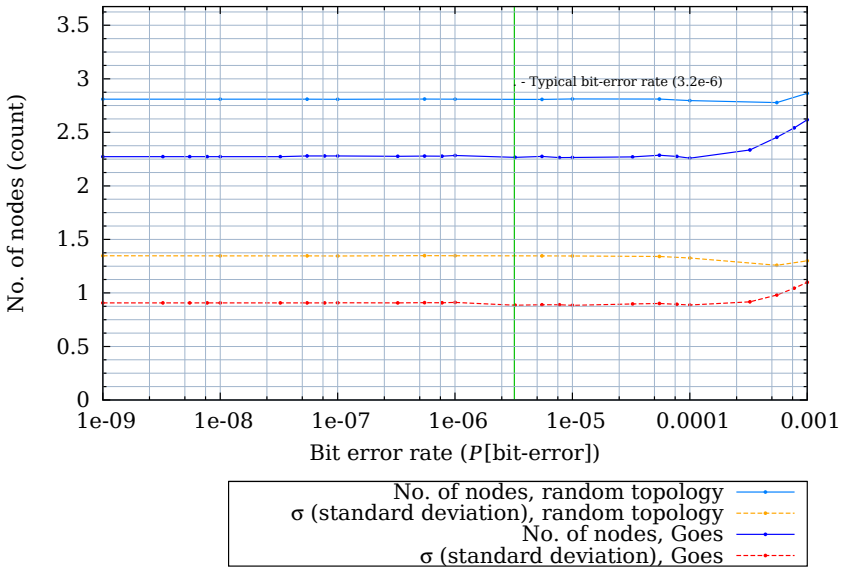


Figure 6.22: Path-length (in no. of hops) in genuinely random topologies vs. Goes under varying bit-error rates.

Figures 6.22 to 6.24 show that the average transmission distances in the random scenarios are much longer than they were in the Goese Polder scenario. This is a natural consequence of the fact that the scenarios that are tested in this experiment are much larger, up to a square kilometre. In section 3.2.3 we theorised that the protocol should be able to cover an area of up-to 0.5km^2 around the data-concentrator. This means that the protocol should be able to reliable cover a distance of 400m, using at most 5 intermediate hops. From figure 6.24 this seems unlikely, given that the average distance that the protocol covers is just about 175m, even in large scenarios. However as we will discuss in section 6.5.2, this figure does not give an accurate impression of the situation. What we can conclude from figures 6.22 to 6.24 is that again the route lengths are very stable, showing little variation even in high bit-error rates.

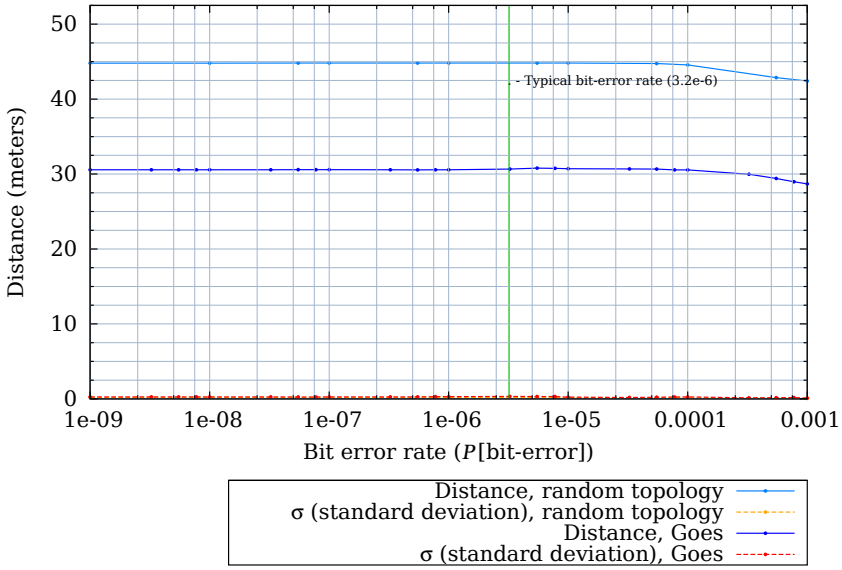


Figure 6.23: Mean transmission distance (single-hop) in genuinely random topologies vs. Goes under varying bit-error rates.

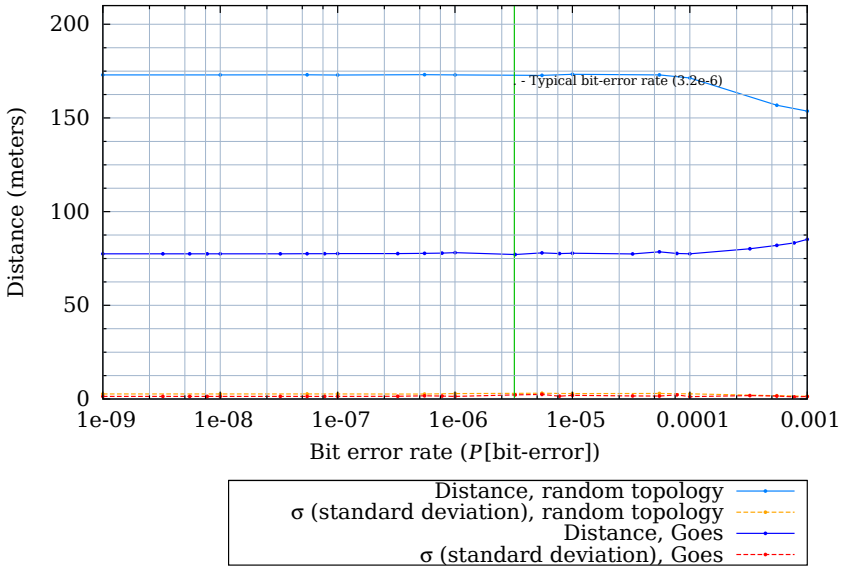


Figure 6.24: Mean transmission path length (multiple-hops) in genuinely random topologies vs. Goes under varying bit-error rates.

6.4.1 Conclusions

We have shown that when the protocol is deployed in scenarios which are like the Goes scenario, having similar environmental parameters, the behaviour of the protocol closely matches with that in the Goes scenario, showing similar responses to changes in the topology under varying node availability. By changing the topologies but keeping leaving other parameters unchanged, we show that the increase in load on the network is indeed, as we proposed, an artifact of the topology in Goes and disappears when averaged over many topologies. We also show that on average it is reasonable to expect better performance in terms of the number of measurements that are delivered to the data-concentrator than was demonstrated in the Goes area.

The close matching between the two experiments indicates that we may be able to use our simulation as a prediction model, estimating how the protocol will behave and what level of performance can be expected in any given situation. We explored this use case in section 6.4 with a simulation targeting a wide variety of topologies and attenuation parameters. We again showed that the protocol did not exceed the duty-cycle limit, showing that regardless of bit-error rate, topology and attenuation conditions it is highly improbable for the protocol to ever cause any individual node to exceed a 1% duty-cycle.

6.5 Miscellaneous findings

We conclude this chapter by sharing some of the interesting findings that were not part of the formal experiments.

6.5.1 Connectivity in the network under extreme BERs

The connectivity in the network slowly decreases and recedes as the bit-error rate approaches the critical threshold. This is interesting because it clearly shows how the spread of nodes and the positioning of the data-concentrator can affect the reliability of the network. It can clearly be seen how the positioning of the patches of greenery interferes with communication in the network at high bit-error rates. Figure 6.25a shows the model as it appears in the simulator. The data-concentrator is located in the lower-left. It shows that the patches of greenery are more-or-less evenly distributed over the area of the simulation. However after running the simulation we found that at high bit-error rates the network forms a distinctive shape around the obstacles in the centre. Evidently these obstacles cause much higher attenuation than the other obstacles. At bit-error rates over $5.5 \cdot 10^{-3}$ communication in the network slowly breaks down. Figures 6.25b to 6.25d show the progression of communication failure in the network. As can be seen the nodes nearest to the data-concentrator are the last to disconnect from the network. Also clearly visible is how the protocol routes around the obstacles, utilising all available connections to maximise the number of nodes reached—such as the single connection top-center, which is among the last connections to drop and provides a significant amount of connectivity to the top-left area of the network.

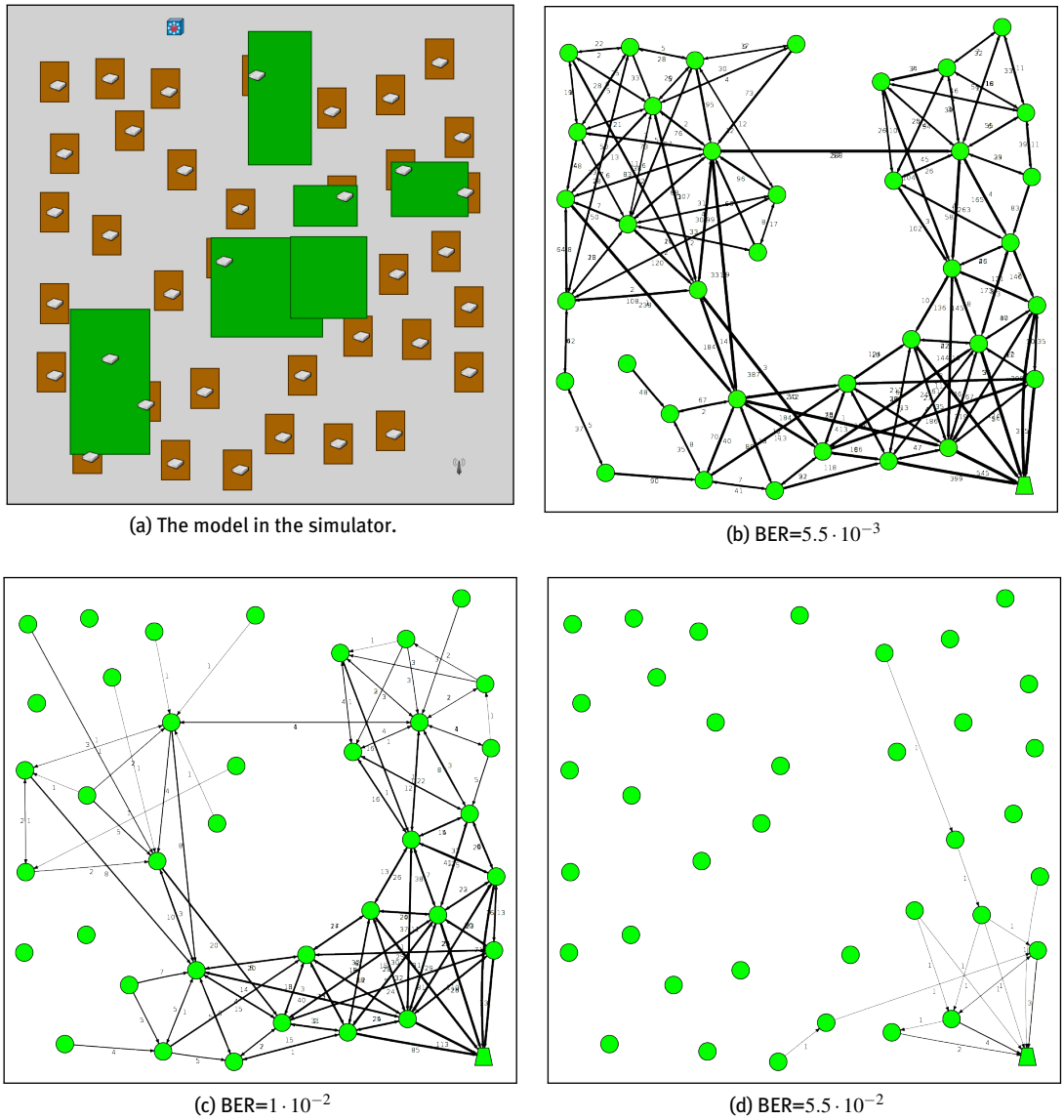
6.5.2 Limitations on the coverage-area

Some of the random scenarios that we have simulated showed that the protocol failed to connect some of the nodes that were situated at the greatest distance from the data-concentrator. In particular scenarios where the size of the environment was set to a length or width greater than 512m showed the limitations in the range of the system. We have measured the distances that *were* bridged in these instances, and have been able to confirm our original hypothesis we postulated in section 3.2.3, where we theorised that the system should be able to cover an area of up-to 0.5km^2. In order cover such a large area it should be possible for a metering-metering node up-to 400 m away from the data-concentrator to report its measurements.

Table 6.1 shows for each scenario with a width or height greater-than or equal-to 512m the maximum distance that the system was able to successfully cover in that scenario. The distances provided are measured as the crow flies, and does not represent the actual distance covered by a packet which may be greater. In some scenarios the longest path reached up-to the edge of the simulated area, and so we provide only a lower-bound. In the other scenarios the protocol did manage to build a network covering all nodes. In these cases we list the maximum observed distance. As shown the average distance covered in the experiments was 390m. This is already quite close to the required 400m distance. However because some of the measurements are only a lower bound, and more-over the table lists at 3 entries where the distance cover exceeded 400m, it is shown that the system indeed is able to cover an area of up-to 0.5km^2.

6.5.3 Dutycycles

We did spent time on building a tool to visualise the duty-cycle level of each node, in the expectation that some of the experiments were going to cause the duty-cycle limit to be exceeded. Because we wanted to be able to show where this happened, in order to identify topological features which might be indicative of causing excessive load on the network under certain circumstances. Although the duty-cycle limit was not exceeded in any of our experiments, this can still be used to show which parts of the network experience the highest load. Figure 6.26 shows one of the models from the last experiment (the largest, $1\text{km} \times 1\text{km}$ model). Figures 6.26b and 6.26c show a visualisation of the load



(a) The model in the simulator.

(b) BER=$5.5 \cdot 10^{-3}$

(c) BER=$1 \cdot 10^{-2}$

(d) BER=$5.5 \cdot 10^{-2}$

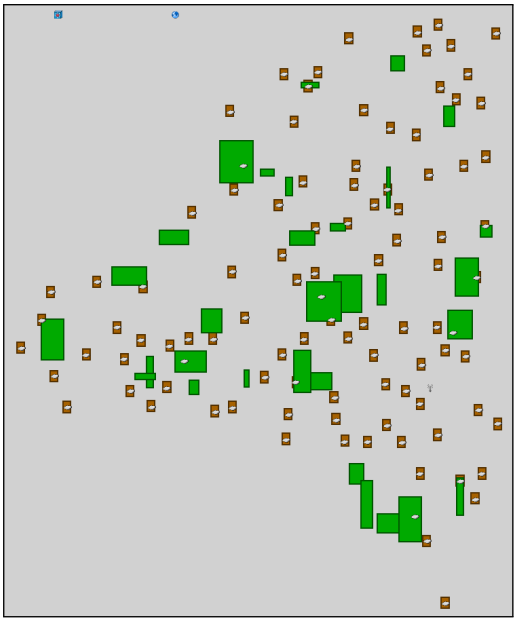
Figure 6.25: random topology.

of the nodes in this network, where green is used to indicate (very) low load, and red is used for (very) high load, nearing the duty-cycle limit. Additionally it is possible to show nodes which exceed the limit in a different colour altogether (not shown here). As can be seen in figure 6.26b all of the nodes in the network are well below the dutycycle limit, all having nearly the same shade of green. When we scale the visualisation's colour scheme to match the range of the load-values we can still see that some areas of the network experience more load than others. Figure 6.26b shows that it is not the nodes that are closest to the data-concentrator that experience the highest load in the network (after the data-concentrator itself), but some of the nodes that are in-between the nodes on the periphery of the network and the data-concentrator. The visualisation in figures 6.26b and 6.26c are based on a simulation with relatively high bit-error rate, however this does not fully explain why these particular nodes are experiencing such elevated levels of load in comparison to the rest of the network. We can unfortunately offer no explanation of this phenomena at this time, and can only recommend further study.

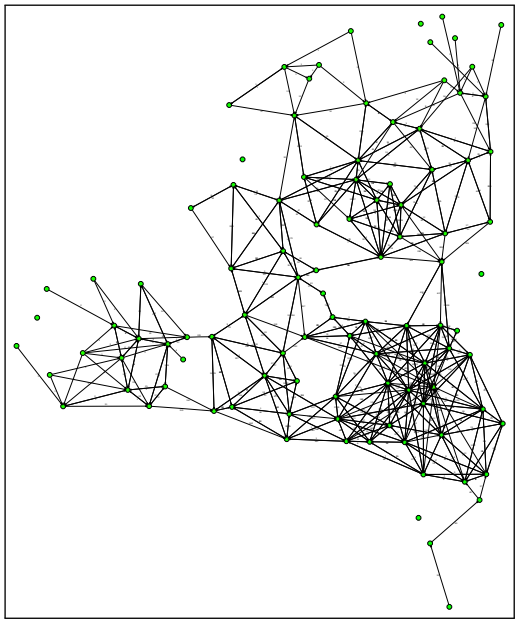
Size	Distance
64m × 512m	306m
64m × 1024m	363m
128m × 1024m	388m
256m × 1024m	406m
256m × 512m	›387m[a]
512m × 512m	456m
512m × 1024m	›394m[a]
1024m × 1024m	›411m[a]
Average	390m

[a] Lower-bound only.

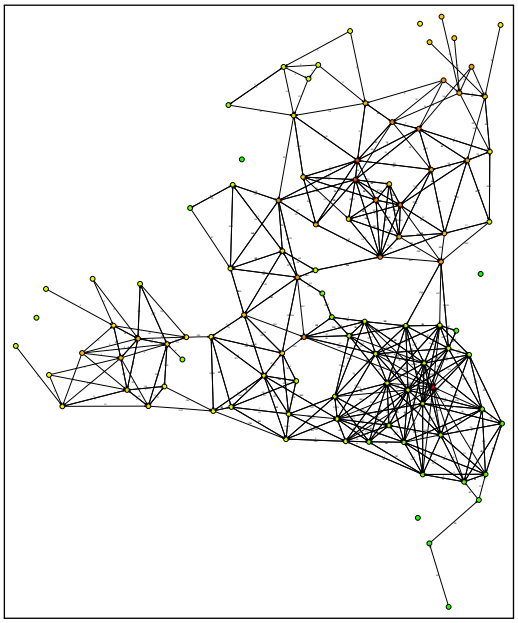
Table 6.1: Communication distances.



(a) The model in the simulator.



(b) Visualising duty-cycle load, mapping load-values to colours as-is.



(c) Visualising duty-cycle load, mapping load-values to colours using scaling to cover the full range of the visualisation.

Figure 6.26: random topology.

6.6 Conclusions

We have shown that the simulation of the Goes scenario is a faithful representation of the real-world. The simulation yields data which correlates with the data collected during the intermediate inspections. We have verified this both by giving intuitive arguments and pictures, as well as by giving a quantative analysis comparing the simulation to the real-world data. This shows that we have created a (relatively) accurate simulation of the Goes scenario. We show that these results hold under other topologies, and that changing the layout of nodes helps to eliminate some of the artifacts that we saw in the simulation of the Goes scenario, yielding data that can be applied to a wide range of sub-urban environments. A study of a wide range of topologies under varying environmental conditions showed that the protocol did not exceed the duty-cycle limitation under any of the conditions tested. It showed that there is sufficient headroom with regards to the limit to state that it is highly unlikely that the 1% duty-cycle limit will ever be exceeded by the current protocol. Thus there is room to extend the protocol, allowing for more communication and features to be supported. Finally we have shown that the distance at which can be communicated over multiple hops using the current protocol and hardware, is currently in the 400+m range. This was achieved on randomised topologies, which may offer better connectivity than in certain real-world scenarios. However we believe that with some planning and mindful placement of metering-devices this is achievable in reality, too.

We have also showed some of the limitations of the current protocol. At present nodes will give up attempts at communication after three failed attempts. This was done to avoid consuming too much bandwidth. However as we have showed there is ample headroom, especially at moderately high bit-error rates. By a simple modification of the protocol, allowing more attempts at communication, the reliability of the protocol could be greatly enhanced without sacrificing too much bandwidth. However because the protocol has been finely tuned throughout to the current scheme—the time-out values used throughout the protocol are tuned to the number of retransmissions permitted for example—it is recommended to perform further simulation studies to investigate this possibility.

Chapter 7

Conclusions and future work

In this thesis we have presented the work done on the analysis and assessment of an existing routing-protocol for use in smart-metering environments. We now briefly summarise the results of our analysis in section 7.1, and a list of contributions in section 7.2. As with any project, not all the issues that we came across and ideas that we had over the course of could be fully solved or explored. In section 7.3 we give an overview of possible future work.

7.1 Results

Our analysis of the data captured in the real-world installations showed that the routing protocol constructed rich network topologies, which reliably transported measurement-data to the data-concentrator and is able to distribute the network load over all participating nodes. We have written a detailed description and specification of the protocol, based on the actual program-code used to program the actual hardware devices. We found no major bugs of flaws in the program-code, although we recommend a cleaner approach to achieving modularity, in order to improve readability and ensure the future maintenance and extendability of the software. Using this protocol specification we have designed and implemented a simulation environment for the protocol. We have analysed the output of the simulation, and verified that it its results closely resemble real-world data, thus the simulation can be used to analyse the current behaviour of the protocol, as well as to test possible future improvements without the need for a full-scale roll-out. We have performed a set of experiments aimed to assess the protocol's behaviour under both known and unknown conditions. Our simulations showed that the protocol can reliably deliver measurements to the data-concentrator (and hence, presumably, to the back-office) under high bit-error rates. Our testing of the protocol's resilience to node-failure showed that the protocol can effectively route around node-failures in a wide range of topologies, and showed high tolerance to dropped nodes in the network, able to deliver measurements of remaining nodes with a minimum of available network-nodes. In all we show that a meshed-networking approach, using inexpensive radio-components and a relatively simple implementation of a source-routing protocol, is able to achieve good results when applied to smart-metering. Our protocol is to function in a wide range of bit-error rates and topologies, and is tolerant to node failure. Simplicity in the protocol is demanded by the constraints of the hardware, yielding a very close integration between application and routing-protocol. Nevertheless the combined design of the protocol and application ensure that regulatory constraints are met, as was confirmed through our simulation-experiments.

7.2 Contributions

This work provides the following contributions.

- *Literature studies and simulator survey.*
  We highlight state of the art in smart-metering, presenting and contrasting different possible underlying technologies to the smart-grid in chapter 2. Within the field of RF-communications in particular we have discussed alternative implementations such as ZigBee as well as a number of proprietary implementations. In section 3.1 we have discussed the various strengths and weaknesses of a large number of simulation solutions. Our analysis of the state-of-the-art will be useful to those who intend to advance the area of communications for smart grids.

- *Development of a formal specification of the protocol.*
  Although the protocol is an existing protocol, no formal specification existed at the beginning of this project. We have developed a specification of the protocol based on the source-code used for the case-study installations. We have presented our specification in chapter 3, highlighting the most important phases of the protocol separately, and providing an in-depth description of the operation of the protocol.

- *Study and characterisation of an existing system.*
  The routing-protocol that is the topic of this thesis was deployed in a case-study deployment in two topographically different areas in the Netherlands. We have presented an *a posteriori* analysis of protocol behaviour and performance, using previously collected data which was handed to us for the express purpose of protocol evaluation in chapter 4.

- *Design and implementation of a simulation environment including tools.*
  We have designed and implemented a simulation environment based on the popular OMNeT++ simulator. This simulator can be used to developed and test possible extensions to (or new versions of) the protocol before deployment in the field. Several improvements and extensions to the OMNeT++ simulator were made in order to support, and improve the accuracy of, the simulation of the protocol. We have performed a cross-validation of the simulation through the use of visual inspections, as well as a quantitative analysis of the accuracy of the simulation using a novel path-matching approach based on string edit-distance metrics. We have detailed the work on the simulation in chapter 5.

- *Analysis of protocol behaviour and performance.*
  Using the simulation environment we developed, we have performed an *a priori* performance analysis through simulation of the protocol under new and previously unexplored conditions. In the experiments we presented in section 6.1 both environmental conditions and network topologies were varied. We have discussed the results of these experiments in chapter 6.

- *Accepted conference paper.*
  D. Geelen, G. Van Kempen, F. van Hoogstraten, A. Liotta, "A Wireless Mesh Communication Protocol for Smart-metering," *Proceedings of The Smart Grid: Telecommunication and Power Distribution Synergy,* Maui, Hawaii, USA, January 30 – February 2, 2012 (IEEE)

- *Accepted journal paper.*
  A. Liotta, D. Geelen, G. Van Kempen, F. van Hoogstraten, "A Survey on Networks for Smart-metering Systems," in *The International Journal of Pervasive Computing and Communication* (http://www.emeraldinsight.com/products/journals/journals.htm?id=ijpcc).

7.3 Future work

The work presented in thesis lays a solid basis for future development and analysis of the protocol. Although we have already performed several experiments and answered many questions, there is always a next level. Examples of the future work developments are given below.

- As mentioned before, during the development of the simulation environment we noted that our simulation of the Goese Polder area was more accurate than that of the Binnenstad scenario in Den Bosch. We had planned to perform an inspection of the site in order to determine what was causing the mismatch between the simulation and the real world behaviour. However due to organisational reasons this was not possible. Still it might be worth-wile to investigate the Binnenstad to further improve the accuracy of the simulation.
- One interesting improvement would be to further enhance the simulation. This could be done within the established OMNeT++ framework, or perhaps by using a different simulator. The current simulation focused on simulating the protocol and the interactions with the environment. To this end some abstractions and short-cuts were required. For example multi-path propagation and interference are not taken into consideration. One intriguing possibility would be to use an advanced simulator capable of very accurately simulating the propagation characteristics, coupled with a highly detailed environmental model. One of the reasons speculated to contribute to the difference in behaviour between the simulation and the Den Bosch area was due to limitations in the obstacle model, which did not allow to express the complexities of the building. Using a simulator with more advanced modelling would be a possible solution to this.
- During the implementation of the protocol in the simulation environment we thought of a number of small changes and improvements to the protocol. The following are some recommendations:
  - During the meshing of packets through the network, each hop delays the forwarding of the packet by a short, random amount of time. This is evidently done to ensure that two nodes who both receive a packet with their node-id in the route-list do not simultaneously attempt to forward the packet. However it may be possible to reduce this time-out without impacting the performance of the protocol, since only a limited number of nodes will actually be attempting to forward the packet at any one time.
  - Similarly, the data-concentrator uses a 250ms timeout, even for packets of type `PT_DIR`. This again may be reduced, for example to twice the time needed to send, receive and process a packet.
  - The current implementation of the protocol is very small and optimised for the given hardware platform. However it may still be possible to include some small optimisations present in more elaborate source-routing protocols. One extension which immediately came to mind was for nodes to attempt to optimise the route a packet takes by searching their route-cache for the packet's destination to see if the node knows a shorter route. This reduces the number of hops each packet has to travel, as well as the end-to-end communication latencies. However this means that some nodes in the network must handle more traffic than before, as the load is no longer spread evenly over all nodes.
  - Currently, when a route is dropped for any reason (e.g. one of the intermediate nodes fails), the protocol will start by attempting direct-communication. However it can infer that this attempt will fail, because otherwise it would not have stored a route before. Instead the node should immediately go to the route-discovery phase. This should reduce the duty-cycle in noisy conditions or in unreliable environments.
  - During our simulations of random topologies we encountered a number of scenarios where the limited number of hops became an issue. This is something that should be taken into account in future versions of the protocol, and could easily be developed and tested using our simulation environment.

Bibliography

[1] S. Massoud Amin and B. Wollenberg, "Toward a smart grid: power delivery for the 21st century," *Power and Energy Magazine, IEEE*, vol. 3, no. 5, pp. 34–41, 2005. [Online]. Available: http://ieeexplore.ieee.org/xpls/abs_all.jsp?arnumber=1507024

[2] S. Caron and G. Kesidis, "Incentive-based energy consumption scheduling algorithms for the smart grid," in *Smart Grid Communications (SmartGridComm), 2010 First IEEE International Conference on*. IEEE, 2010, pp. 391–396. [Online]. Available: http://ieeexplore.ieee.org/xpls/abs_all.jsp?arnumber=5622073

[3] M. Erol-Kantarci and H. Mouftah, "The impact of smart grid residential energy management schemes on the carbon footprint of the household electricity consumption," in *IEEE Electrical Power and Energy Conference (EPEC), Halifax, NS, Canada*, 2010. [Online]. Available: http://b-dig.iie.org.mx/BibDig/P10-0777/epec10/papers/1764_CR.pdf

[4] G. Deconinck, "An evaluation of two-way communication means for advanced metering in Flanders (Belgium)," *2008 IEEE Instrumentation and Measurement Technology Conference*, pp. 900–905, May 2008. [Online]. Available: http://ieeexplore.ieee.org/lpdocs/epic03/wrapper.htm?arnumber=4547164

[5] A. Liotta and G. Exarchakos, *Networks for Pervasive Services: Six Recipes to Upgrade the Internet*. Springer, 2011, vol. 92.

[6] K. Dostert, *Powerline communications*. Prentice Hall PTR, 2001. [Online]. Available: http://www.lavoisier.fr/livre/notice.asp?id=OR2WO3A62K3OWY

[7] R. Olsen, "Technical considerations for broadband powerline (BPL) communication," in *EMC Zurich Symposium*, 2005, pp. 1–6. [Online]. Available: http://preview.pserc.wisc.edu/documents/publications/papers/2005_general_publications/olsen_bpl_paper_feb2005.pdf

[8] S. Keemink and B. Roos, "Security analysis of Dutch smart metering systems," *University of Amsterdam, Amsterdam*, 2007. [Online]. Available: http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.159.6314&rep=rep1&type=pdf

[9] J. Vasconcelos, "Survey of regulatory and technological developments concerning smart metering in the European Union electricity market," 2008. [Online]. Available: http://cadmus.eui.eu/handle/1814/9267

[10] J. Kurose and K. Ross, *Computer networks: A top down approach featuring the internet*. Pearson Addison Wesley, 2005. [Online]. Available: http://www.just.edu.jo/~fhawad/Teaching/courses/CPE442/CPE442CoursePolicySummer0607.pdf

[11] IEEE Communication Society, "802.15.4 IEEE standard for Information Technology-Part 15.4: Wireless Medium Access Control (MAC) and Physical Layer Specifications for Low Rate Wireless Personal Area Networks (LR-WPANS)," 2003.

[12] Z. Alliance, "Zigbee specification," *ZigBee document 053474r06, version*, vol. 1, 2005. [Online]. Available: http://scholar.google.com/scholar?hl=en&btnG=Search&q=intitle:Zigbee+specification#0

[13] ISA Standard, "Wireless systems for industrial automation: process control and related applications," *ISA-100.11 a-2009*.

[14] J. Song, S. Han, D. C. Al Mok, M. Lucas, M. Nixon, and W. Pratt, "Wirelesshart: Applying wireless technology in real-time industrial process control," in *IEEE Real-Time and Embedded Technology and Applications Symposium*. IEEE, 2008, pp. 377–386.

[15] IEEE Communication Society, "Standard for Long Wavelength Wireless Network Protocol," 2006.

[16] C. Perkins, E. Belding-Royer, and S. Das, "Ad hoc on-demand distance vector (AODV) routing," no. February 1999, 2003. [Online]. Available: http://ssrnet.snu.ac.kr/course/cs4541_682-06-01/presentation/AODV_MBH_ver_White.pdf

[17] R. Bienert, "ZigBee Certification Frequently Asked Questions." [Online]. Available: http://www.zigbee.org/imwp/idms/popups/pop_download.asp?ContentID=6200

[18] K. De Craemer and G. Deconinck, "Analysis of state-of-the-art smart metering communication standards," *status: published*, pp. 1–6, 2010. [Online]. Available: http://scholar.google.com/scholar?hl=en&btnG=Search&q=intitle:Analysis+of+State-of-the-art+Smart+Metering+Communication+Standards#0

[19] J. Zhu and R. Pecen, "A Novel Automatic Utility Data Collection System using IEEE 802 . 15 . 4-Compliant Wireless Mesh Networks," *Networks*, 2008.

[20] X. Meng, P. Zerfos, V. Samanta, S. Wong, and S. Lu, "Analysis of the reliability of a nationwide short message service," in *INFOCOM 2007. 26th IEEE International Conference on Computer Communications. IEEE*. IEEE, 2007, pp. 1811–1819. [Online]. Available: http://ieeexplore.ieee.org/xpls/abs_all.jsp?arnumber=4215793

[21] Q. Liu, B. Zhao, Y. Wang, and J. Hu, "Experience of AMR systems based on BPL in China," in *Power Line Communications and Its Applications, 2009. ISPLC 2009. IEEE International Symposium on*. IEEE, 2009, pp. 280–284. [Online]. Available: http://ieeexplore.ieee.org/xpls/abs_all.jsp?arnumber=4913443

[22] A. Mahmood, M. Aamir, and M. Anis, "Design and implementation of AMR smart grid system," in *Electric Power Conference, 2008. EPEC 2008. IEEE Canada*. IEEE, 2008, pp. 1–6. [Online]. Available: http://ieeexplore.ieee.org/xpls/abs_all.jsp?arnumber=4763340

[23] L. Yujin, G. Liwei, and Z. Deyi, "On remote automatic meter reading system based on GPRS technology," in *Control Conference (CCC), 2010 29th Chinese*. IEEE, 2010, pp. 5731–5733. [Online]. Available: http://ieeexplore.ieee.org/xpls/abs_all.jsp?arnumber=5572414

[24] A. Al-Omary, W. El-Medany, and S. Al-Irhayim, "Design and Implementation of Secure Low Cost AMR system using GPRS Technology," *ipcsit.com*, vol. 5, pp. 138–143, 2011. [Online]. Available: http://www.ipcsit.com/vol5/25-ICCCM2011-A064.pdf

[25] T. Khalifa, K. Naik, and A. Nayak, "A Survey of Communication Protocols for Automatic Meter Reading Applications," *IEEE Communications Surveys & Tutorials*, pp. 1–15, 2010. [Online]. Available: http://ieeexplore.ieee.org/lpdocs/epic03/wrapper.htm?arnumber=5473879

[26] S.-w. Lee, C.-s. Wu, M.-s. Chiou, and K.-t. Wu, "Design of an automatic meter reading system [electricity metering]," *Proceedings of the 1996 IEEE IECON. 22nd International Conference on Industrial Electronics, Control, and Instrumentation*, vol. 1, pp. 631–636, 1996. [Online]. Available: http://ieeexplore.ieee.org/lpdocs/epic03/wrapper.htm?arnumber=571031

[27] S. K. Kim, "Automatic meter reading system and method using telephone line," 2006.

[28] M. Shen and Y. Dai, "Design of Management Terminal for Remote Centralized Meter Reading System Based on PSTN [J]," *Low Voltage Apparatus*, vol. 16, 2007.

[29] G. A. O. Tian, J. Zhan-rong, Y. Yan, and W. Qi, "Design of the Remote Meter Reading System Based on the PSTN SMS," *Microprocessors*, 2007.

[30] X. Zou and B. Ramamurthy, "Routing Techniques in Wireless Ad Hoc Networks Classification and Comparison," *Proceedings of the Sixth World*, 2002. [Online]. Available: http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.96.9871&rep=rep1&type=pdf

[31] N. Abramson, "THE ALOHA SYSTEM: another alternative for computer communications," in *Proceedings of the November 17-19, 1970, fall joint computer conference*. ACM, 1970, pp. 281–285. [Online]. Available: http://portal.acm.org/citation.cfm?id=1478502

[32] B. Lichtensteiger, B. Bjelajac, C. Muller, and C. Wietfeld, "RF Mesh Systems for Smart Metering: System Architecture and Performance," *Smart Grid*, pp. 379–384, 2010. [Online]. Available: http://ieeexplore.ieee.org/xpls/abs_all.jsp?arnumber=5622071

[33] L. Hardy and M. Gafen, "A new highly-synchronized wireless mesh network model in use by the Electric Company to switch to automatic meter reading: Case study," *2008 5th International Conference on Networked Sensing Systems*, pp. 31–34, Jun. 2008. [Online]. Available: http://ieeexplore.ieee.org/lpdocs/epic03/wrapper.htm?arnumber=4610892

[34] A. Krohn, M. Beigl, C. Decker, and T. Riedel, "Syncob: Collaborative Time Synchronization in Wireless Sensor Networks," *2007 Fourth International Conference on Networked Sensing Systems*, pp. 283–290, Jun. 2007. [Online]. Available: http://ieeexplore.ieee.org/lpdocs/epic03/wrapper.htm?arnumber=4297432

[35] Landis+Gyr, "GridStream." [Online]. Available: http://www.landisgyr.com/en/pub/products_and_services/gridstream.cfm

[36] J. Hui and D. Culler, "Extending IP to low-power, wireless personal area networks," *IEEE Internet Computing*, pp. 37–45, 2008. [Online]. Available: http://www.computer.org/portal/web/csdl/doi/10.1109/MIC.2008.79

[37] T. Winter, P. Thubert, A. Brandt, T. Clausen, J. Hui, R. Kelsey, P. Levis, K. Pister, R. Struik, and J. Vasseur, "Rpl: Ipv6 routing protocol for low power and lossy networks," *Work In Progress), http://tools. ietf. org/html/draft-ietf-roll-rpl-11*, 2010.

[38] P. Kulkarni, S. Gormus, and Z. Fan, "A Self-organising Mesh Networking Solution Based on Enhanced RPL for Smart Metering Communications," *ieeexplore.ieee.org*. [Online]. Available: http://ieeexplore.ieee.org/xpls/abs_all.jsp?arnumber=5986178

[39] M. Ringwald and K. Romer, "BitMAC: a deterministic, collision-free, and robust MAC protocol for sensor networks," *Proceeedings of the Second European Workshop on Wireless Sensor Networks, 2005.*, vol. 00, pp. 57–69, 2005. [Online]. Available: http://ieeexplore.ieee.org/lpdocs/epic03/wrapper.htm?arnumber=1461999

[40] L. Hardy and M. Gafen, "A fixed AMR for an Electric Company, using a new type of wireless mesh network: A Case Study," *2008 5th International Conference on Networked Sensing Systems*, pp. 223–223, Jun. 2008. [Online]. Available: http://ieeexplore.ieee.org/lpdocs/epic03/wrapper.htm?arnumber=4610862

[41] S.-W. Luan, J.-H. Teng, S.-Y. Chan, and L.-C. Hwang, "Development of a smart power meter for AMI based on ZigBee communication," *2009 International Conference on Power Electronics and Drive Systems (PEDS)*, pp. 661–665, Nov. 2009. [Online]. Available: http://ieeexplore.ieee.org/lpdocs/epic03/wrapper.htm?arnumber=5385726

[42] Texas Instruments, “Z-Stack.” [Online]. Available: http://www.ti.com/tool/z-stack

[43] D. Johnson and D. Maltz, “Dynamic source routing in ad hoc wireless networks,” *Mobile computing*, pp. 153–181, 1996. [Online]. Available: http://www.springerlink.com/index/QG8843V474571123.pdf

[44] D. Johnson, D. Maltz, J. Broch, and Others, “DSR: The dynamic source routing protocol for multi-hop wireless ad hoc networks,” *Ad hoc networking*, vol. 5, pp. 139–172, 2001. [Online]. Available: http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.131.5263&rep=rep1&type=pdf

[45] J. Postel, “Internet protocol (RFC 791),” STD 5, RFC 791, September, Tech. Rep., 1981.

[46] B. Reid, “Oncor Electric Delivery Smart Grid initiative,” *2009 62nd Annual Conference for Protective Relay Engineers*, no. Figure 1, pp. 8–15, Mar. 2009. [Online]. Available: http://ieeexplore.ieee.org/lpdocs/epic03/wrapper.htm?arnumber=4982500

[47] H. Gharavi, “Multigate mesh routing for smart Grid last mile communications,” *Communications and Networking Conference (*, pp. 275–280, 2011. [Online]. Available: http://ieeexplore.ieee.org/xpls/abs_all.jsp?arnumber=5779140

[48] T. Iwao, K. Yamada, M. Yura, and Y. Nakaya, “Dynamic Data Forwarding in Wireless Mesh Networks,” *), 2010 First IEEE*, pp. 385–390, 2010. [Online]. Available: http://ieeexplore.ieee.org/xpls/abs_all.jsp?arnumber=5622074

[49] A. Ghassemi and S. Bavarian, “Cognitive radio for smart grid communications,” *Smart Grid Communications (*, pp. 297–302, 2010. [Online]. Available: http://ieeexplore.ieee.org/xpls/abs_all.jsp?arnumber=5622097

[50] C. Bennett and D. Highfill, “Networking AMI Smart Meters,” *2008 IEEE Energy 2030 Conference*, no. November, pp. 1–8, Nov. 2008. [Online]. Available: http://ieeexplore.ieee.org/lpdocs/epic03/wrapper.htm?arnumber=4781067

[51] S. Rohjans, M. Uslar, R. Bleiker, J. González, M. Specht, T. Suding, and T. Weidelt, “Survey of Smart Grid Standardization Studies and Recommendations,” in *Smart Grid Communications (SmartGridComm), 2010 First IEEE International Conference on*. IEEE, 2010, pp. 583–588. [Online]. Available: http://smartgrid.ieee.org/publications/latest-ieee-xplore-publications/plug-in-hybrid-electric-vehicle/2771-survey-of-smart-grid-standardization-studies-and-recommendationshttp://ieeexplore.ieee.org/xpls/abs_all.jsp?arnumber=5621999

[52] F. yuan Xu, L. Zhou, Y. Wu, and Y. Ma, “Standards, policies and case studies in smart metering,” in *Power and Energy Society General Meeting, 2010 IEEE*, vol. 02. IEEE, 2010, pp. 1–5. [Online]. Available: http://ieeexplore.ieee.org/xpls/abs_all.jsp?arnumber=5590103

[53] P. Rámila and H. Rudnick, “Assessment of the Introduction of Smart Metering in a Developing Country,” q mimeo, Tech. Rep., 2009. [Online]. Available: http://web.ing.puc.cl/~power/paperspdf/RamilaRudnick.pdf

[54] M. Uslar, S. Rohjans, R. Bleiker, J. González, M. Specht, T. Suding, and T. Weidelt, “Survey of Smart Grid standardization studies and recommendationsâĂŤPart 2,” in *Innovative Smart Grid Technologies Conference Europe (ISGT Europe), 2010 IEEE PES*. IEEE, 2010, pp. 1–6. [Online]. Available: http://ieeexplore.ieee.org/xpls/abs_all.jsp?arnumber=5638886

[55] P. UNION, “DIRECTIVE 2009/28/EC OF THE EUROPEAN PARLIAMENT AND OF THE COUNCIL,” *Official Journal L*, pp. 64–85, 2009. [Online]. Available: http://old.zgwrp.org.pl/ue/poradnik/dokumenty/01_prawo/srodowisko/Zarzadzanieodpadami/2000-53-EC/tekst-ang/Directive.dochttp://dpea.scotland.gov.uk/Documents/qJ10165/J115136.PDF

[56] European Commission, "Standardisation Mandate to CEN, CENELEC and ETSI in the Field of Measuring Instruments for the Development of an Open Architecture for Utility Meters Involving Communication Protocols Involving Interoperability," 2009.

[57] Nederlands Normalisatie-instituut, "Netherlands Technical Agreement NTA-8130," vol. 8130, no. August, 2007.

[58] KEMA Consulting, "Smart Meter Requirements - Dutch Smart Meter specfication and tender dossier," 2008.

[59] J. Postel, "Transmission control protocol," STD 7, RFC 793, September, Tech. Rep., 1981.

[60] M. K. McKusick, K. Bostic, M. J. Karels, and J. S. Quarterman, "The Design and Implementation of the 4.4 BSD Operating System.pdf," 1996.

[61] Z. Shelby, P. Mahonen, J. Riihijarvi, O. Raivio, and P. Huuskonen, "NanoIP: the zen of embedded networking," *IEEE International Conference on Communications, 2003. ICC '03.*, pp. 1218–1222, 2003. [Online]. Available: http://ieeexplore.ieee.org/lpdocs/epic03/wrapper.htm?arnumber=1204570

[62] M.-l. Chiang and Y.-c. Li, "LyraNET: A Zero-Copy TCP/IP Protocol Stack for Embedded Operating Systems," *11th IEEE International Conference on Embedded and Real-Time Computing Systems and Applications (RTCSA'05)*, pp. 123–128, 2005. [Online]. Available: http://ieeexplore.ieee.org/lpdocs/epic03/wrapper.htm?arnumber=1541068

[63] A. Dunkels, "Towards tcp/ip for wireless sensor networks," *Malardalen University Licentiate Thesis*, no. 45, 2005. [Online]. Available: http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.62.3794&rep=rep1&type=pdf

[64] L. van der Ploeg, "Small TCP / IP stacks for micro controllers," Ph.D. dissertation, Universiteit Twente.

[65] B. Cody-Kenny, D. Guerin, D. Ennis, R. Simon Carbajo, M. Huggard, and C. Mc Goldrick, "Performance evaluation of the 6LoWPAN protocol on MICAz and TelosB motes," *Proceedings of the 4th ACM workshop on Performance monitoring and measurement of heterogeneous wireless and wired networks - PM2HW2N '09*, pp. 25–30, 2009. [Online]. Available: http://portal.acm.org/citation.cfm?doid=1641913.1641917

[66] Nordic Semiconductor, "nRF905 Single chip 433/868/915MHz Transceiver Product Specification."

[67] P. Truchly, G. Marek, F. Tomas, G. Radoslav, and L. Michal, "Simulation of IMS using current simulators," in *ELMAR, 2008. 50th International Symposium*, vol. 2, no. September. IEEE, 2008, pp. 545–548. [Online]. Available: http://ieeexplore.ieee.org/xpls/abs_all.jsp?arnumber=4747562

[68] D. Curren, "A survey of simulation in sensor networks," *project report (CS580), University of Binghamton*, 2005. [Online]. Available: http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.94.2626&rep=rep1&type=pdf

[69] M. Köksal, "A Survey of Network Simulators Supporting Wireless Networks," *Middle East Technical University Ankara, TURKEY October*, vol. 22, 2008. [Online]. Available: http://www.ceng.metu.edu.tr/~e1595354/ASurveyofNetworkSimulatorsSupportingWirelessNetworks.pdf

[70] R. Hornig and A. Varga, "AN OVERVIEW OF THE OMNeT++ SIMULATION ENVIRONMENT," *Proceedings of the First International ICST Conference on Simulation Tools and Techniques for Communications, Networks and Systems*, 2008. [Online]. Available: http://eudl.eu/?id=3027

[71] V. Mhatre, "Enhanced wireless mesh networking for ns-2 simulator," *ACM SIGCOMM Computer Communication Review*, vol. 37, no. 3, p. 69, Jul. 2007. [Online]. Available: http://portal.acm.org/citation.cfm?doid=1273445.1273455

[72] M. Mekni and B. Moulin, "A Survey on Sensor Webs Simulation Tools," *2008 Second International Conference on Sensor Technologies and Applications (sensorcomm 2008)*, pp. 574–579, 2008. [Online]. Available: http://ieeexplore.ieee.org/lpdocs/epic03/wrapper.htm?arnumber=4622722

[73] S. Mehta, N. Ullah, M. H. Kabir, M. N. Sultana, and K. S. Kwak, "A Case Study of Networks Simulation Tools for Wireless Networks," *2009 Third Asia International Conference on Modelling & Simulation*, pp. 661–666, 2009. [Online]. Available: http://ieeexplore.ieee.org/lpdocs/epic03/wrapper.htm?arnumber=5072065

[74] P. Levis, N. Lee, M. Welsh, and D. Culler, "TOSSIM: accurate and scalable simulation of entire TinyOS applications," in *Proceedings of the 1st international conference on Embedded networked sensor systems*. ACM, 2003, pp. 126–137. [Online]. Available: http://portal.acm.org/citation.cfm?id=958506

[75] D. Willkomm, S. Valentin, and T. Parker, "Simulating Wireless and Mobile Networks in OMNeT++ The MiXiM Vision," *Proceedings of the First International ICST Conference on Simulation Tools and Techniques for Communications, Networks and Systems*, 2008. [Online]. Available: http://eudl.eu/?id=3031

[76] J. Goldhirsh and W. Vogel, "Handbook of Propagation Effects for Vehicular and Personal Mobile Satellite Systems," *NASA Reference Publication*, vol. 1274, 1999. [Online]. Available: http://scholar.google.com/scholar?hl=en&btnG=Search&q=intitle:Handbook+of+Propagation+Effects+for+Vehicular+and+Personal+Mobile+Satellite+Systems#0

[77] A. Silva and M. Vuran, "Empirical evaluation of wireless underground-to-underground communication in wireless underground sensor networks," *Distributed Computing in Sensor Systems*, pp. 231–244, 2009. [Online]. Available: http://www.springerlink.com/index/Y468214K719226U0.pdf

[78] L. Li, M. Vuran, and I. Akyildiz, "Characteristics of underground channel for wireless underground sensor networks," *Proc. Med-Hoc-NetâĂŹ07*, pp. 92–99, 2007. [Online]. Available: http://gtk.hopto.org:8089/220.pdf

[79] I. Wassell, "Wireless sensor network: Water distribution monitoring system," *2008 IEEE Radio and Wireless Symposium*, pp. 775–778, Jan. 2008. [Online]. Available: http://ieeexplore.ieee.org/lpdocs/epic03/wrapper.htm?arnumber=4463607

[80] V. I. Levenshtein, "Binary codes capable of correcting deletions, insertions, and reversals," in *Soviet Physics Doklady*, vol. 10, no. 8, 1966, pp. 707–710.

[81] S. Chapman, "String similarity metrics for information integration," *University of Sheffield*, vol. 39, 2006.

[82] S. White, "How to strike a match," *Development Cycles*, 2004.