MSc Informatics Engineering
MSc Thesis
Final Report

Efficient Communication in Dense Networks

João Miguel Borges Subtil
jsubtil@student.dei.uc.pt

Advisor:
Marília Pascoal Curado
Co-Advisor
André Figueira Riker
July 3, 2017

FCTUC DEPARTAMENTO
DE ENGENHARIA INFORMÁTICA
FACULDADE DE CIÊNCIAS E TECNOLOGIA
UNIVERSIDADE DE COIMBRA

Acknowledgements

I would like to thank André Riker who provided me with crucial mentoring. Our discussions helped me greatly to keep experimenting and trying new approaches particularly when the work was not evolving according to plan.

I would like to express my gratitude to my advisor Marília Curado. Thank you for the guidance you provided me, your mentoring helped me keep focused on the important tasks. Your support was critical in achieving the final results.

I would also like to thank all my friends and colleagues who helped make this thesis a reality.

Finally I would like to thank my parents for all the support you gave through my academic career. Without you this work would not be possible.

Abstract

In the last few years the number of Internet of Things (IOT) networks has been increasing. In order to support Fifth Generation (5G), large-scale highly-dense networks will have to be deployed. Those networks will contain a massive number of low power, battery operated sensors, sensing and forwarding messages in dynamic topologies (star, mesh, ad-hoc). These highly dense networks can cause rapid exhaustion of a node's resources. As such they have to be as efficient as possible to operate as long as they are needed, while achieving reliable communications.

This work presents and examines state of the art mechanisms for energy measurements and data aggregation in Low power and Lossy network (LLN). Measuring the energy consumption of multiple sensor nodes is a complex task. This work presents some of the techniques used and opts by a software approach to obtain that metric. This work focuses on in-network data aggregation. The data aggregation is performed at every hop.

The core part of this work focuses on the development of a testbed environment. This environment consists of several physical boards communicating with each other and a gateway. The main focus of the testbed is measuring the energy consumption across different scenarios.

With these challenges in mind, this work presents a cross-layer approach to data aggregation. The main objective of the aggregation is to reduce the power consumption. The method is based on the creation of groups of nodes with similar properties, leveraging the similarity of the exchanged data. The final mechanism is capable of achieving up to 9.17% in energy savings when performing aggregation.

Keywords. Data aggregation, IOT, sensor networks, Low-power networks, Energy efficiency

This research was funded by a scholarship provided by the Portuguese Foundation for Science and Technology (FCT) though a grant: DenseNet project - PTDC/EEI-SCR/6453/2014 (POCI-01-0145-FEDER-016777).

FCT Fundação para a Ciência e a Tecnologia
MINISTÉRIO DA CIÊNCIA, TECNOLOGIA E ENSINO SUPERIOR

Cofinanciado por:
COMPETE 2020
PROGRAMA OPERACIONAL COMPETITIVIDADE E INTERNACIONALIZAÇÃO

PORTUGAL 2020

UNIÃO EUROPEIA
Fundos Europeus
Estruturais e de Investimento

Contents

List of Figures

List of Tables

Acronyms

5G Fifth Generation. v

6LoWPAN IPv6 over Low power Wireless Personal Area Networks. 6, 18

CCR Channel Check Rate. 35

CoAP Constrained Application Protocol. 2, 6, 12, 18, 19, 21, 24, 25, 31–33, 44

HTTP Hypertext Transfer Protocol. 18

IETF Internet Engineering Task Force. 11

IOT Internet of Things. v, vii, 1, 3, 6, 7, 9–13, 18, 29, 39

IP Internet Protocol. 3, 30

IPv6 Internet Protocol version 6. 4, 7, 18

LLN Low power and Lossy network. v, 3, 4, 11, 18

node Sensor Node. 11

NST Network Spanning Tree. 12

RDC Radio Duty Cycling. 35

RPL IPv6 Routing Protocol for Low-Power and Lossy Networks. 18

SLIP Serial Line Internet Protocol. 15

SNMD Sensor Node Management Device. 9, 10

UART Universal Asynchronous Receiver/Transmitter. 11

UDP User Datagram Protocol. 18, 30

USB Universal Serial Bus. 15

VM Virtual Machine. 4

Chapter 1

Introduction

This chapter presents the work produced during this Masters Thesis. It contains a state-of-the-art study about data aggregation and energy measurement in IOT sensor networks. It also contains the setup of an experimental testbed and the testing of the developed method of data aggregation.

This first chapter starts by stating the motivation behind this work, followed by the objectives and expected contributions. Finally the thesis structure is presented.

1.1 Motivation

In today's world people are more connected than ever. Everyday new technological products appear, many of which can be connected to the Internet. IOT is following the trend and rapidly expanding. It is difficult to estimate the role it will have in the next few decades.

Cisco estimates [1] that by the year 2020 the Internet will contain more than 50 billion devices among which the majority of them will have constraints in terms of memory, processing power or energy. With the number of use cases for IOT products expanding, future technologies will have to deal with very dense networks and saturated wireless channels. Therefore it is of the utmost importance that an effort is made not only to reduce battery usage but also to decrease the amount of data flowing within those networks.

1.2 Objectives and Contributions

This work focuses on the issues of dense networks. When energy-constrained devices communicate in dense networks it is necessary to use energy efficient mechanisms to avoid a quick exhaustion of the nodes energy. The main objective of this work is to reduce the energy consumption in those networks. In order to do this it is necessary to measure the energy consumption across many devices simultaneously. It is also necessary to develop energy efficient mechanisms. This work focuses on data aggregation.

During the development of this thesis several contributions have been made.

First it was necessary to correctly create a testbed environment to test the developed data aggregation techniques. In order to set up the testbed several tasks had to be performed. First it was necessary to find an accurate method of measuring the energy consumption. The simplest solution to measure the energy consumption on several nodes simultaneously was using the software Energest.

For this energy measurement software to be used it was necessary to measure its accuracy. Several tests were performed comparing the energy measurements obtained by Energest against those obtained by a hardware tool. The measured difference was acceptable and the Energest was selected to measure the energy consumption.

Having the testbed capable of measuring energy consumption the next step followed. It was necessary to test if it was possible to reduce the energy consumption by performing data aggregation at each node. The first tests showed that it was possible to save energy by performing aggregation. A paper was written with the results and submitted to *The 20th ACM International Conference on Modeling, Analysis and Simulation of Wireless and Mobile Systems* (MSWiM 2017).

Finally a data aggregation mechanism based on groups was developed. This method included a light-weight layer combined with extra fields added in each Constrained Application Protocol (CoAP) message that enabled a node to perform data aggregation. This method differed from the one first tested, groups were created that enabled internal aggregation, between group members. External aggregation was also possible, between different groups. This mechanism was tested in an environment similar to the one from the first tests. It showed an improvement in energy consumption.

1.3 Thesis Structure

This Thesis is divided as follows: Chapter 2 introduces ContikiOS, the operating system that will be used on the testbed, Chapter 3 introduces techniques to measure energy consumption and data aggregation methodologies. Chapter 4 describes the experimental work performed in order to prepare the testbed. Chapter 5 contains the information regarding the first experimental tests. Chapter 6 presents the developed data aggregation technique based in groups and contains a discussion about the results. Chapter 7 illustrates and describes the work plan and Chapter 8 presents the conclusions and future work.

Chapter 2

ContikiOS Platform

This chapter introduces ContikiOS. First an overview of the platform is presented as well as important aspects of its architecture. Then the included energy estimation software is presented followed by the reasons behind selection ContikiOS amongst others.

2.1 Introduction

ContikiOS is an operating system for the IOT. It connects tiny low-cost, low-power microcontrollers to the Internet. Being an operating system it provides a lot of commodity features to speed up and facilitate program development. Those features include a network simulator (Cooja) [2], useful to test programs before the deployment, power awareness (Energest), for measuring energy consumption or full Internet Protocol (IP) networking for communication, amongst many others. Like other other operating systems it has device drivers to interact with specific hardware, that means it is possible to have the same program running on a network of heterogeneous hardware. It also includes many Internet standard protocols that will be discussed latter.

ContikiOS dates back to 2003, when the creator Adam Dunkels [3] released it as an open source project. It had several features that contributed to its success, the original version included application level multithread support, run time application loading as well as an IP protocol version called μIP designed to run on 8 and 16 bit microcontrollers.

2.2 Contiki Overview

Typically the hardware used in LLN has very high restrictions at both energy and resource levels that do not necessarily decrease as technology improves, as such multi-threading support could not follow the traditional architectures. The earlier versions of Contiki used an event based kernel with

a preemptive multi-threading library available only to applications that required it. Preemption was implemented using timer interrupts that saved processor registers onto the stack an later switched to the kernel stack. The model was later abandoned due to the need of having one separate stack per thread which consumed valuable resources. It was later replaced by *protothreads* which will be discussed in the next section.

In a deployed network which can have many nodes over a very widespread area it becomes exceptionally laborious to reprogram the nodes individually. Contiki introduced run-time remote node reprogramming. Now, tasks such as bug removal or the addition of new features could be done in a reduced amount of time with minimal overhead.

In older systems it was necessary to send the full binary image of the system, including all the libraries, the kernel and the application itself, which caused a huge impact on the networks with low throughput. By enabling just the application to be sent, not only was the time to reprogram the network reduced but the energy spent doing so as well, since the amount of data was several times smaller compared to the full system.

An important feature of Contiki was the μIP protocol stack, also developed by Dunkels [4]. It was adopted by several big companies like Atmel and Cisco and updated to support Internet Protocol version 6 (IPv6) [5]. Contiki can also be distributed in a Virtual Machine (VM) with all the tools needed for development. The VM has a simulator called Cooja which can simulate a LLN deployment, it is a very usefull feature for quick prototyping and testing.

2.3 ContikiOS Architecture

The ContikiOS can be divided in two parts, the first contains the loaded program, called application, which can be changed dynamically in run-time. The second is the core, which contains the kernel, services, libraries an the loader. Bellow is an image depicting the Contiki partitioning. The communication between processes always goes through the kernel but applications can directly control hardware.

The kernel consists of an event scheduler that dispatches events to running processes and executes calls to the processes polling handlers The polling mechanism is used to manage high priority events called synchronous events, these are scheduled between low priority events, called asynchronous events. All processes share the same stack. Event scheduling is done at the same level and processes cannot alter that order unless an interrupt occurs. Interrupts can be generated by hardware.

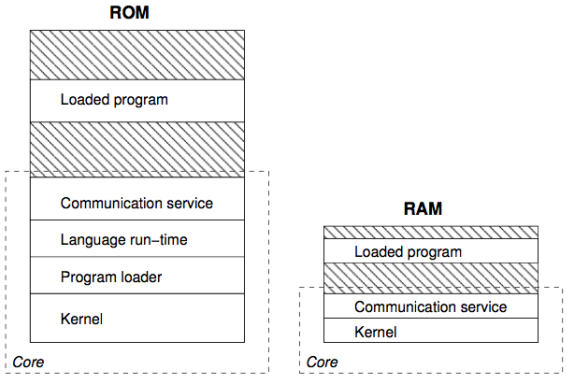


FIGURE 2.1: Contiki [3]

Program loading is done by a run-time re-location function. When the program is loaded the loader attempts to allocate enough memory for it, if it fails the program loading is aborted. If the memory is available the loader calls the processes main function.

Services are processes whose functions can be used by more than one program. Services can also be loaded during run-time, they are accessed by an interface which has the list of running services. If one program wants to access a service, it uses a stub of that interface to search for the service by using a string with the service name. Then the interface returns the process ID of the desired service.

The kernel implements only the basic mechanisms to manage process handling and CPU. Other tasks are managed by libraries that can either be dynamically or statically lined with processes.

The communication stack is implemented as a service and can be altered in run-time. When a packet is received an event is generated to deliver the packet to the corresponding application. If the application replies a reply packet is placed in the output buffer.

In the first versions, Contiki had a multi-threading model where each thread had its separate stack, however this behaviour sometimes caused exhaustion of the hardware resources in some weaker systems. To solve that problem the Contiki creator launched a system called *protothreads*.

Protothreads[6] are a simplified thread system, here threads share the same stack and context changes are done by re-wounding the stack, they include locking mechanisms. Compared to the previous system protothreads only run on the function which implements them, and so, if a function using protothreads calls other function, the secondary function will not have locking mechanisms. In order to provide conditional locking all functions need to run in a separate protothread.

2.4 Energy Estimation Mechanism

Contiki has an internal mechanism [7], that is able to estimate how much energy a node is consuming. Energest is a service that maintains a table entry for every component, the CPU and transceiver on the node. Each entry contains the total time, expressed in system ticks, that particular component has been active. When the component is turned on the energy estimation module produces a time stamp, when the component is turned of the module computes the time it spent active.Finally the table entry is updated with the difference.

The energy estimation mechanism uses a linear model for the sensor node energy consumption. The energy consumption is calculated by the formula:

$$\frac{E}{V} = I_m t_m + I_l t_l + I_t t_t + I_r t_r + \sum_i I_{c_i} t_{c_i} \tag{2.1}$$

where V is the supply voltage,I_m the current draw of the microprocessor when running, t_m the time in which the microprocessor has been running, I_l and t_l the current draw and the time of the microprocessor in low power mode(there is no low power mode available in the transciever), I_t and t_t the current draw and the time of the communication device in transmit mode. Ir and tr the current draw and time of the communication device in receive mode, and I_{c_i} and t_{c_i} the current draw and time of other components such as sensors and LEDs.

Finally,to obtain the energy consumption it is only necessary to know the power consumption of each component by checking the datasheet of the selected hardware and. Despite that energy measurements should be conducted to verify, and possibly correct, those values. This methodology has been evaluated [8] and was found to be a valuable alternative to more complex, expensive and difficult solution such as specialised hardware.

2.5 Why ContikiOS

When searching for an operating system for IOT there were a few options available. The major criteria that lead to the rejection of most of them was the inclusion of standard IOT protocols such as CoAP and IPv6 over Low power Wireless Personal Area Networks (6LoWPAN). The IOT community has been converging to certain protocols and makes sense in the context of this work to propose data aggregation on top of those protocols. Other factor such as project maturity, included features, the developer community support and hardware support also weighed in the decision.

In the end it came down to Contiki and TinyOS[9]. ContikiOS ended up being the chosen platform. It included the above mentioned protocols and more, it has a large and active community of developers, with a reasonable

amount of reading material available not only to learn about, but also to troubleshoot potential problems. There are also companies using and supporting Contiki. Companies such as Zolertia, a hardware manufacturer or Cisco also contributed to the platform with drivers and IPv6 related code respectively. There are also big projects using Contiki [10] ,[11], [12], which contribute to its improvement and growth. Also many referenced papers in this Thesis make use of Contiki, and although there are no surveys about IOT operating system usage, it is commonly referred as the most popular project.

2.6 Summary

This section started by introducing ContikiOS in the context of this project, followed by an overview of relevant features that contributed to its success. Then the system architecture was described in a more in-depth way, followed by the description of the embedded energy estimation software. This was done in preparation for the next chapter because energy management is a big component of this thesis. Lastly the reasoning behind the choice of this platform amongst others was given.

Chapter 3

Energy Efficiency and Data Management in IOT

This chapter comprises two fundamental points that are the core of this thesis, energy efficiency and data management. Section 3.1 of this chapter will introduce energy measurement techniques essential to compare energy efficiency between different protocols. Section 3.2 discusses energy usage reduction when applying data aggregation schemes.

Energy efficiency in IOT environments is one of the base pillars of a successful real-world deployment. Node networks have very limited resources not only in terms of processing but also in terms of energy supply, as they frequently use batteries and can be deployed for a very long time. It then becomes imperative to develop applications that are built with energy saving mechanisms but are also capable of reliable and scalable communications.

3.1 Energy Measurement

Measuring energy consumption is a very important task when designing and managing a IOT network. There are several ways energy can be measured, some rely on software and others rely on hardware.

The Sensor Node Management Device (SNMD) [13] is a hardware device that plugs into sensor nodes, and is capable of measuring energy consumption of sensor nodes in a distributed fashion as well as handle management tasks. The device is capable of completely controlling the energy supply of the sensor node. It can use a real battery to supply power as well as simulating a real one. In this case a special circuit controls the voltage in a way that resembles a real world depletion. Since the real world behaviour is very difficult to simulate, hardware has very specific characteristics that are not considered in simulation scenarios. This may lead to untrustworthy results of energy consumption estimation in the simulation results.

The MoteLab testbed, from Werner-Allen et al. [14], consists of a set of permanently deployed sensor nodes connected to a central server that handles scheduling, reprogramming nodes, and data logging. In addition, each

node in MoteLab is connected to a network-connected digital multi-meter, which helps to continuously monitor the energy usage of the node.

Similarly to the previous measurement techniques Sheu et al. [15] also propose a hardware device called E-MCU (hardware device with energy measurement capability) that plugs into the sensor node to measure energy consumption. The method consists on modifying the TinyOS libraries to signal each state change through the board pins. Then the E-MCU measures the time in each state. It takes into account processor, radio, LED and other board components such as resistors and transistors bonded to a single state. The energy levels for each component is measured with a multimeter. To calculate the total energy consumption the time on each state is multiplied by the supplied voltage.

Contrary to previous energy measurement methods this one is software based, Dunkels et al. [7] use a method similar to Sheu et al. [15] but with software. The Energest software module was previously explained in the Contiki, Chapter 2.4.

Another software energy measurement method is Powertrace, developed by Dunkels et al. [16]. This method however differs from the previous one as it can measure energy consumption from network-level activities such as packet transmission or reception. Internally it uses structures called energy capsule which records the time of the desired activity whether it is a packet reception, packet transmission or idle wake up. It can be seen as a fine-grained extension to Energest. Where the latter measures hardware components energy as a whole, Powertrace can measure individual activities inside those components. Energy capsules can also be combined into capsule aggregates. Those aggregates contain the sum of the energy from all its capsules. They can later be used to attribute power consumption to network protocols. Powertrace also enables applications or protocols to subscribe to its capsule feed. When the activity of a capsule has been terminated the control module informs Powertrace that then distributes those values to the subscribers. This enables the creation of power aware protocols and applications. By using those feeds those protocols can make energy-aware decisions.

Lastly one very important fact to consider are the inaccuracies on the production of electronic components, in some cases the difference in energy consumption between the same hardware may differ more than 4% according to Hurni et al. [8].

All of the methods mentioned above have pros and cons. SNMD [13] is probably the best method to measure energy consumption from the methods described above. It allows a distributed energy measurement and management of the device. The drawback of this method however is cost, according to [8] the cost is about 300$ per unit which makes it a convenient tool for lab environments but too costly for large deployments.

Similarly in Werner-Allen et al. [14] the usage of a network connected digital multimeter has a very high degree of accuracy but the cost of the equipment is very high. In Sheu et al. [15] the cost of the equipment is lower because the E-MCU is a rather simpler device and it is shown its accuracy only differs 2.6% from measurements with a multimeter. However the main drawback of this solution is that it cannot be used in a real test scenario. Mainly because the device sends its data via Universal Asynchronous Receiver/Transmitter (UART), that would mean each node would have to be connected via cable to a PC in order to extract the data.

The software method cited is Energest, as Powertrace is Energest based. This method accuracy was already evaluated on Hurni et al. [8], and it has an overhead of 0.7% [7], however in order to obtain valid results the consumption on each state has to be validated with a measurement device as the real consumption may differ from the datasheet.

One other method of measuring the energy consumption is using using an oscilloscope. Oscilloscopes are very precise devices capable of measuring the energy consumption of nodes. However when dealing with a network of several nodes far from each other using this type of equipment becomes a complex and costly endeavour.

3.2 Energy Consumption Optimisation

Energy conservation has become a central focus in IOT research. Work in energy conservation has been considered at several levels of the protocol stack. From the design of the physical layer, to protocols developed with energy savings in mind. Protocols developed for energy savings take advantage of overhearing and scheduling to allow nodes to sleep when they are not transmitting or receiving messages [17] [18]. At the routing level [19], an Internet Engineering Task Force (IETF) standard has been designed to operate in an energy efficient way across a wide range of network types with focus on LLN. Other techniques such as energy harvesting [20] take into consideration solar powered node when building the routing trees and traffic will preferably go through the nodes equipped with solar panels.

Many different solutions have been presented to reduce energy consumption at all levels. Radio modules are typically the greatest consumers of energy [21]. The energy consumption during data transmissions is directly proportional to the number of bits sent through the network. An important aspect to keep in mind is that aggregation is in some cases a trade-off. It can achieve a smaller number of transmissions but it can add extra delay or increase the in Sensor Node (node) processing.

The Grid-based Routing and Aggregator Selection Protocol (GRASS), Al-Karaki et al. [22], is a cluster based protocol and has two levels of data aggregation. First a group is created and a Local Aggregator and Cluster-head

are elected. This selection is performed according to a set of rules designed to maximise network lifetime, one node can have those two functions. Then Master Aggregators are elected and they will aggregate data from smaller groups.

One method, called Lifetime Balanced data Aggregation(LBA) [23] proposes an aggregation scheme under an application-specific end-to-end delay requirement. LBA adaptively adjusts the aggregation delays between parent-child nodes to balance lifetime. Once a collection tree has been established each node needs to set Self-Aggregation Delay (SAD) and Forwarding-Aggregation Delay (FAD) parameters. SAD is the time a node waits to transmit a packet. After that time expires all data produced in that node is aggregated, to reduce the amount of data sent to the parent node. Similarly the FAD will wait, a certain amount of time, for packets from its children, if it has any. Once those parameters have been set up, each child-parent relationship will adjust those values to maximise node lifetime while maintaining delay requirements.

Two-tier aggregation for multi-target applications (TTAMA) [24] performs data aggregation within group communications. The aggregation structure is constructed in two phases. In the first phase a Network Spanning Tree (NST) containing all nodes is calculated, then it is pruned to find subsets called TTAMA trees that will route both Internal Group Traffic (IGT) and External Group Traffic (EGT) traffic. IGT includes nodes belonging to a group with the same configurations Internal traffic is aggregated by means of a simple mathematical function, it can be a Sum, Maximum, Minimum or an Average. Despite the number of input packets, the output is always one packet. EGT comprises all traffic from different groups. Payloads from different groups are merged into a single message. TTAMA has been developed to work with group-based communications and was tested with CoAP groups [24] .

AIDA [25] proposes an adaptive application independent data aggregation method. The goal is to maximise the utilisation of the communication channel with the benefit of lowering energy consumption. This is achieved by employing various degrees of data aggregation, in accordance with current traffic patterns. The goal is to reduce channel contention, packet header overhead and data padding for fixed size packets. Conceptually AIDA is implemented in the protocol stack as a intermediate layer between the MAC and network layers. It can be combined with other application specific data aggregation schemes. Aggregation is performed by concatenating payloads in one single packet. The number of payloads depends on a parameter called Degree Of Aggregation (DOA) which is tuned according to the network state. The AIDA layer intercepts communications between the MAC and Network layers to gather information to tune the DOA parameter. As the network traffic builds up and transmission is delayed, the feedback loop adjusts the DOA to allow for a greater degree of aggregation prior to sending.

3.3 Summary

In order to develop energy efficient protocols we first need to implement a reliable way to measure energy consumption. If we are able to assign energy usage to individual protocols rather than just to hardware states the optimisation we can achieve is much greater. Energy consumption increases with the number of bytes sent. In this section various methods of performing data aggregation were mentioned, their main objective is to reduce the amount of data flowing in the network in order to save power.

Chapter 4

Tools Validation

This Chapter describes the background experimental work done to prepare the testbed experiments. Section 4.1 describes the main objectives. Section 4.2 presents the hardware and software tools used. Section 4.3 describes all of the experimental scenarios and the Chapter ends with a summary.

4.1 Objectives

The main objective of this chapter was to validate the testbed. The purpose of the testbed experiments was to evaluate the energy consumption on real hardware. The energy consumption was obtained by using Energest [7]. This application has enough precision, as shown in section 4.3.2 and is the most adequate to measure energy usage in many devices simultaneously without added complexity. Validation and calibration tests have been performed to ensure the values reported by this tool are indeed correct.

4.2 Environment Description

This section contains all the relevant information about the components used in the experimental testbed. First the hardware devices are listed, followed by the software tools, finally the protocol stack is introduced.

4.2.1 Hardware

The board, which will act as a node, selected for this project is the Atmel Atmega256rfr2 Xplained Pro [26]. It has an embedded IEEE 802.15.4 radio chip, one ceramic antenna, and a SMA header for an external antenna. The flash memory has 256Kb. This board has embeded transcievers that avoid configurations, and an integrated bootloader that facilitates programing. An FTDI chip was also used to interact with the border router through a Serial Line Internet Protocol (SLIP) connection. The other Atmel boards have had the output redirected to the Universal Serial Bus (USB) port, to allow for a

faster development with less cables. A Texas Instruments P401R board was also used to measure energy consumption using the Energy Trace technology.

Figure 4.1 is the FTDI chip used to communicate with the Border Router (or gateway). Figure 4.2 corresponds to the board wich will be used as a node. Finally Figure 4.3 depicts the Texas Instruments P401R board used to measure the energy consumption.

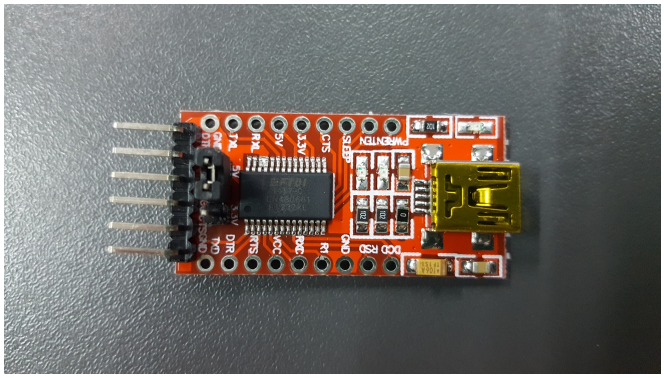


FIGURE 4.1: FTDI chip

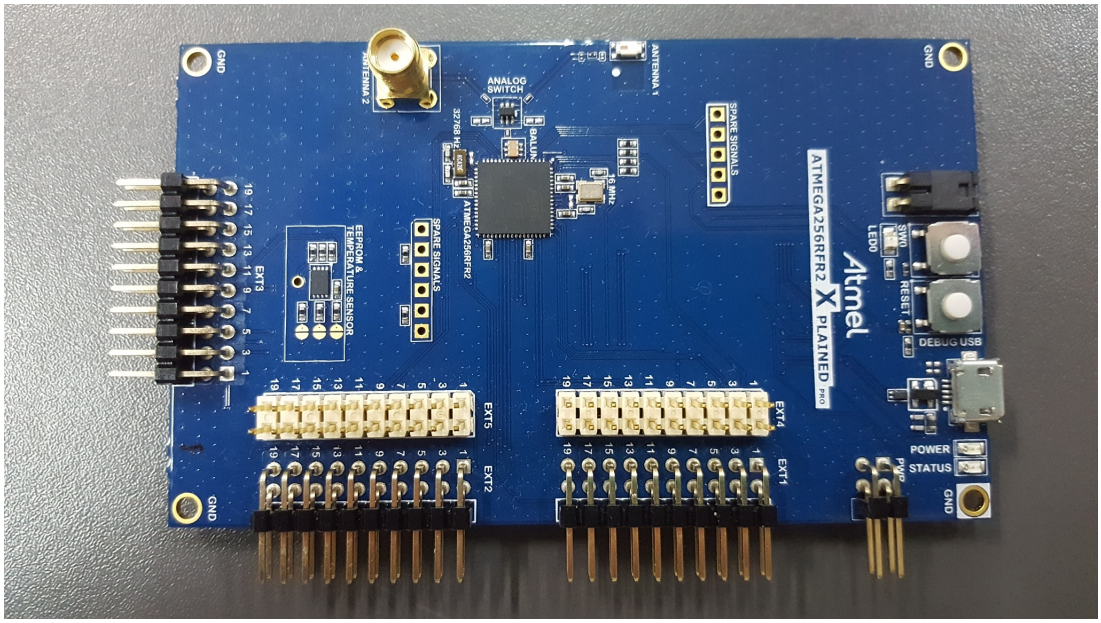


FIGURE 4.2: Atmega256rfr2 X-pro board

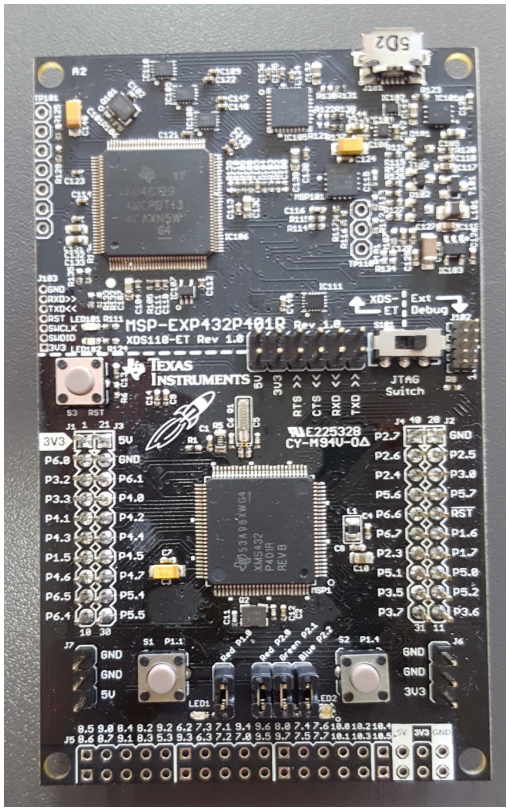


FIGURE 4.3: Texas Instruments MSP432 P401R

4.2.2 Software

The following list contains all the software used in the experimental testbed.

- Ubuntu 16.04 LTS, as the operating system
- Windows 7 with Atmel Studio and Code Composer Studio 7
- AVR 8 bit toolchain (avr-gcc, version 4.9.2), compiler for the board
- Avrdude version 6.2, used to program the board
- Screen version 4.03.01, terminal program to grab the board output
- Wireshark version 2.0.2, to check packet flow on the network

Ubuntu was used as the development system because its compatibility with the Avrdude, Screen and the AVR compiler. Windows 7 was used because both programs mentioned were not available in other platforms. Atmel studio was used to flash the boards with the bootloader provided by the current Atmel platform mantainer. Code Composer was used because it contains the Energy Trace tool. This software alied to the P401R board can measure the energy consumption of a target board with a resolution of 2kHz. This tool had two purposes, it was used to calibrate the values of the energy consumption for both the CPU and the transceiver in the Atmel board.

4.2.3 Protocol Stack

The chosen protocol stack, shown in Table 4.1, contains the standard protocols used by low power sensor networks [27] [28] [29] [30].

CoAP is an application layer protocol designed for resource-constrained devices. It translates easily to Hypertext Transfer Protocol (HTTP) to enable integration with the web. The User Datagram Protocol (UDP) protocol is a transport layer protocol, well know in the TCP/IP suite. IPv6 Routing Protocol for Low-Power and Lossy Networks (RPL) is a protocol designed specifically to address constraints related with routing within LLN such as processing power, energy or memory. 6LoWPAN is the technology developed to deliver IPv6 to LLN. The IEEE 802.15.4 is a standard that specifies the physical layer for low-rate wireless networks.

These protocols were chosen because the IOT community is rapidly converging to these standards [31].

TABLE 4.1: IOT Protocol stack

CoAP
UDP
RPL
6LowPAN
IEEE 802.15.4

4.3 Tool Validation

This section contains information about the experimental work produced. Its purpose is to validate the tools used in the testbed. This includes the internal clock and Energest validation. To ensure the Energest module was functioning properly two things were necessary. First the timer keeping track of the time spent on each state had to be working properly. Then the energy spent on each state (transmission/reception and active CPU) had to be calibrated.

The first experiment in section 4.3.1 was done to test the clock. The second experiment in section 4.3.2 was done to calibrate the Energest module and measure its accuracy.

4.3.1 Clock Validation

The Contiki operating system has different sets of timer libraries that are used both by applications and system alike. The Rtimer library [32] provides scheduling and execution of real time tasks. This library has a very fine grained resolution and is used by the Energest module to count the time

spent on each state. In early tests some drifts appeared between what was the expected on time of the CPU module and the actual value.

In an attempt to correct the drift some more recent bootloaders from Atmel were flashed and different fuses used. Fuses are configuration parameters set in special registers that can control, amongst others, what oscillator to use and the frequency the CPU runs at. This proved ineffective, but the different bootloaders/fuses had an impact on the actual time passing, just not the expected one. Upon reading the support material about CPU frequency on ContikiOS, it was found out that there is a variable that can set the CPU frequency. The CPU frequency was set to 8MHz and some tests were performed to ensure the reported value was consistent to what was expected. These tests were done to evaluate the drift between the expected time and the real value.

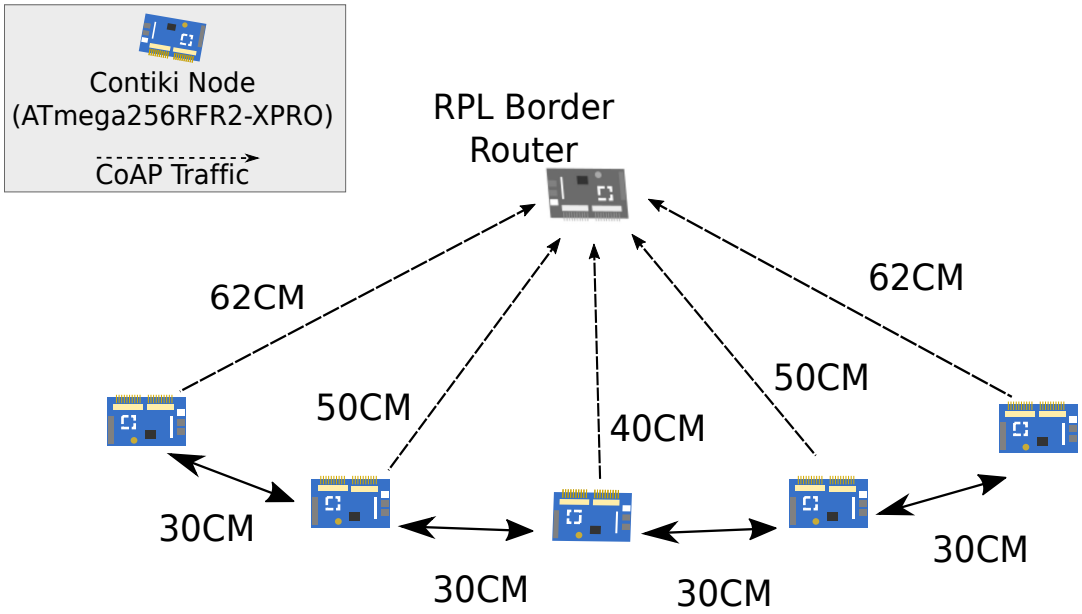


FIGURE 4.4: Clock drift evaluation scenario

Figure 4.4 represents the topology chosen to measure the clock drift. In this test five boards were placed on a table communicating simultaneously with each other and with the border router. Five separate executions were done and then the results were averaged. Each execution took 63 minutes and the first three minutes were discarded as warm-up time. Each node sent a CoAP message every 30 seconds. The Energest CPU variable was used to count the total time. The Rtimer library has a variable that contains the value of timer ticks that correspond to a second, which is set to 7812 ticks.

So at the end of the 60 minutes of test the CPU total value is expected to be 28 123 200 ticks. The averaged value obtained across all boards was 29 184 312 ticks. This means the average drift is 3.7% or about 135 seconds every hour. The standard deviation was 1 087 794 ticks or about 139 seconds.

These values were odd because the standard deviation was very close to the average drift value. Upon an inspection of the Energest values it was found out one function in the Energest module had an issue. The function was being called to update the Energest without taking into account the Rtimer register only has 16 bits, so, about every 8 seconds it is necessary to take into account the register is reset. That issue was corrected and the test was redone.

Upon the new execution the CPU total value is expected to be 28 123 200 ticks, same as before. The averaged value obtained across all boards was 28 377 768 ticks. This means the average drift is 0.9% or about 32 seconds every hour. The standard deviation was 54 915 ticks or about 7 seconds. These values are much better than before and the standard deviation is much lower.

The values for the drift are acceptable considering there is no protocol in place correcting the clock for each board.

4.3.2 Energest Validation

The Energest module was already explained in the Section 2.4. In order to use this tool it was necessary to measure the energy consumption in different states (the states being active CPU and radio Transmit/Receive). Then it was also necessary to perform a testbed experiment and measure the energy consumption. The energy consumption was measured by the Texas Instruments board and by the Energest. The obtained values were then compared to measure the difference between.

The energy consumption of each state is indicated in the datasheet. However those values only take into account the microcontroller and the board has other chips, LED'S, resistors and other components that also have an energy draw.

The first tests were performed in order to measure the power consumption of the different states. The methodology of those tests was the following: 10 boards were selected to be measured. This number represents a significant sample from the total number. All of the tests had a duration of 10 minutes and the first two were discarded as a warmup.

For the CPU test all radio functions were turned off and each node ran a simple program that printed "Hello World!" every 10 seconds. Two executions were done for each board and then all the results were averaged. The final value was then set as the CPU consumption. That value was 9.32 mA, the standard deviation was 0.18 mA. The CPU measurements were performed running Contiki without entering in low power mode, since this version of Contiki is not stable when ATmega256RFR2 is in low power mode.

For the radio transmit value, three separate executions were done for each of the 10 boards and then all the results were averaged. In each of those executions three transmissions were picked and averaged. This was done by

exporting the Energy Trace CSV file containing the values of each execution. Then a Python script was created. It took as input the start and end time of the transmission and outputted the average transmission power of that execution. The final value was 21.44 mA, and the standard deviation was 0.42 mA. The radio receive methodology was similar to the radio transmit with the difference being the Python script was fed the start and end time of a packet reception. The final value was 15.49 mA and the standard deviation was 0.15 mA. At this point one of the 10 boards, the board number 2, consistently reported higher values than the average, both for active CPU and radio Transmit/Receive. Its values were discarded because the behaviour was classified as an outlier. The final values measured for the Energest are in Table 4.2.

TABLE 4.2: Current Measurement of the States

State	**CPU**	**Transmission**	**Reception**
Electrical Current	9.32 mA*	21.44 mA	15.49 mA

*This measurement includes idle and active.

The values for the different states were applied to the Energest to measure the Energy consumption. It was then necessary to check if the energy consumption reported by the Energest was consisted with the values measured by the Texas P401R board. For these tests the methodology was the following. Each of the 9 remaining boards was subject to 63 minute tests communicating with the border router with a CoAP message every 30 seconds. The first three minutes were discarded as warmup time. Then three separate executions were done for each board and then the results were averaged.

The results for this test are shown in Table 4.3, the average error was 3.16% and the standard deviation is 1.57%, these values are consistent with the error interval measured in [8]. The error rate is acceptable, so the Energest was used to measure the energy consumption in subsequent tests.

TABLE 4.3: Total Energy Consumption, in mJ

Board	**Real**	**Energest**	**Error (%)**	**Board**	**Real**	**Energest**	**Error (%)**
1	112707.3	115272.3	2.28	6	109282.3	113216.6	3.60
2*	n/a	n/a	n/a	7	113827.7	113929.6	0.20
3	108492.0	114006.3	5.09	8	111921.7	114646.3	2.44
4	108710.0	114345.0	5.19	9	109843.7	114412.0	4.16
5	111113.0	114455.0	3.01	10	111296.3	114086.0	2.50
Avg	110799.3	114263.2	3.16				

*Outlier.

4.4 Summary

In this Chapter the testbed environment was introduced. The hardware and the software tools were depicted and their purpose was described. Then the requirements to use the Energest tool were presented. The Rtimer clock had some issues that had to be solved and then tested. Then the Energest required that the correct energy consumption value was set for the different states. Tests were performed to measure those values. The Chapter ended with the comparison between the hardware and software measurements.

Chapter 5

Testbed Experiments

This chapter contains information about the development of the aggregation mechanism and subsequent tests.

Section 5.1 starts by introducing the two data aggregation techniques that will be implemented. Chapter 5.2 contains the overview of the data aggregation architecture. Section 5.3 contains the results and discussion about the first experiments conducted. The Chapter ends with a summary.

5.1 Data aggregation techniques

There are two types of aggregation that were implemented, namely payload aggregation [24] and payload concatenation [24]. **Payload aggregation**: occurs when one node receives messages from child nodes and combines that data into a single value using a mathematical function (e.g. average value).

As per Table 5.1, node 1 receives one packet from node 2 and one packet from node 3. When the node 1 has to send data it combines its data with data it received. In this case node 1 uses the average function, and the average value between those three values is 12. Now instead of transmitting three separate packets, only one packet is sent, but its data is the average value between those three nodes.

TABLE 5.1: Payload Aggregation in Node 1

Node Id	Packet Header	Payload(integer)
Node 2 message	yes-discard	20
Node 3 message	yes-discard	10
Aggregator node (1)	no	6
Message to be transmitted	yes	12

TABLE 5.2: Payload Concatenation

Node Id	Packet header	Payload(integer)		
Node 2 message	yes-discard	20		
Node 3 message	yes-discard	10		
Aggregator node (1)	no	6		
Message to be transmitted	yes	6	20	10

Payload concatenation: uses the same principles mentioned above, the data originates in other nodes and then arrives at the node performing aggregation. However instead of using a mathematical function to reduce the size of the data the payloads are concatenated and then sent as a single payload without loss in precision. The payload aggregation is a classical method of aggregating data, already presented in Section 3.2. In Table 5.2 the messages received from other nodes are concatenated in the payload, this means the data produced by nodes 2 and 3 will be kept intact.

5.2 Data aggregation mechanism

The objective of the data aggregation mechanism was to perform aggregation at the application level. As described in 4.2.3 CoAP was selected as the application protocol. When nodes sent CoAP messages in the network it was first necessary to capture packets that were being forwarded. Then it was necessary to use one of the aggregation techniques described above to merge the data.

In order to capture packets a cross-layer approach was necessary. This layer was located between the Internet Protocol (IP) and the User Datagram Protocol (UDP) layers. It was created between these layers in order to verify the destination IP address of every message received from the neighbours.

After packets had been captured one of the aggregation techniques would be used to merge the data. The first data aggregation technique that was implemented was the payload concatenation, as it seemed more complex and could need more testing.

Algorithm 1 Concatenation algorithm

```
1: Start
2:     function NEW_PACKET_ARRIVES(msg)
3:         typeCode ← parseCode(msg)
4:         address ← parseAddrs(msg)
5:         if isCoAP(typeCode) and notMyAdd(address) then
6:             buffer(msg)
7:             if timer_not_set() then
8:                 set_timer()
9:             end if
10:        end if
11:    end function
12:
13:    function TIMEOUT_CALLBACK( )
14:        msg_num ← count_msg(buffer)
15:        payloads_data ← extractPayloads(buffer)
16:        new_coap ← assembly_new_coap(payload_data)
17:        forward_next_layer(new_coap)
18:    end function
19: End
```

Algorithm 1 depicts the decision mechanism of the referred layer. At line 2 of Algorithm 1 a verification is performed when a packet arrives. This verification checks if the received packet is a CoAP message and is not destined for the node itself. If the message meets the criteria it is stored in a buffer and a timer is set. After the timer is set all messages will be stored until that timer is triggered. When the timer is triggered the node assembles a single message containing all of the payloads from the packets. Other information contained in those packets is discarded.

5.3 Testbed experiment and results

This section contains results of the testbed experiments with the aggregation. First a threshold study was conducted to verify the minimum number of messages that had to be aggregated in order for the aggregation to be cost efficient. This cost is related to the cost of extracting payloads *versus* the energy again achieved by transmitting less data. Then a larger testbed was created to verify if the values obtained in the smaller experiment were similar when a greater number of nodes was used.

5.3.1 Threshold experiment

After the mechanism for data concatenation had been implemented in Contiki, it was necessary to find in which conditions would aggregation achieve energy savings.

To do this a threshold test was conducted. The objective was to find which is the minimum number of payloads on a message able to be cost-efficient in terms of energy consumption. A small test scenario was created with 4 nodes. Two nodes were producing packets at different time intervals and sending them to one node that communicated with the Border router (or gateway). For comparison purposes, the standard CoAP has been evaluated in the same scenario. Standard CoAP produces messages containing only a single payload and forwards all the received messages from the 2 injector nodes.

A communication round takes 60 seconds. At each communication round the number of packets created by the injector varies between 6, 12, 18 and 24. When aggregating data the node only produces a message every 60 seconds. This is done in order to test how the aggregation behaves if it aggregates more payloads. That message will contain all the payloads received in that 60 second round. The tests consisted of 15 individual experiments for each number of packets per round. The tests were performed for simple data forwarding and data aggregation. The tests lasted 8 minutes and the first 3 minutes were discarded.

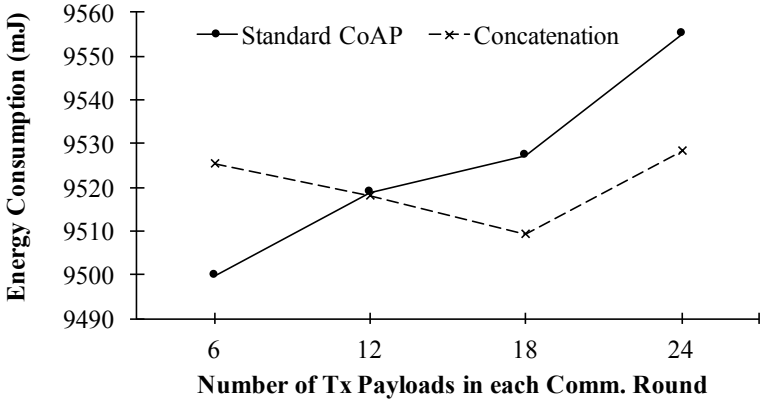


FIGURE 5.1: The fixed unit fields

Figure 5.1 shows the obtained results. The results show that when comparing standard data forwarding to packet concatenation energy gains are only achievable if more than 12 packets are concatenated.

5.3.2 Testbed experiment results

A larger testbed was created to check if the threshold values were similar when a larger number of nodes was used. In this experiment only packet concatenation was tested due to time constraints.

In the testbed scenario the 8 Contiki nodes were deployed to create a balanced tree topology in an indoor environment. The distance between each board is depicted in Figure 5.2, the distance between nodes may have varied between experiments by a few centimetres.

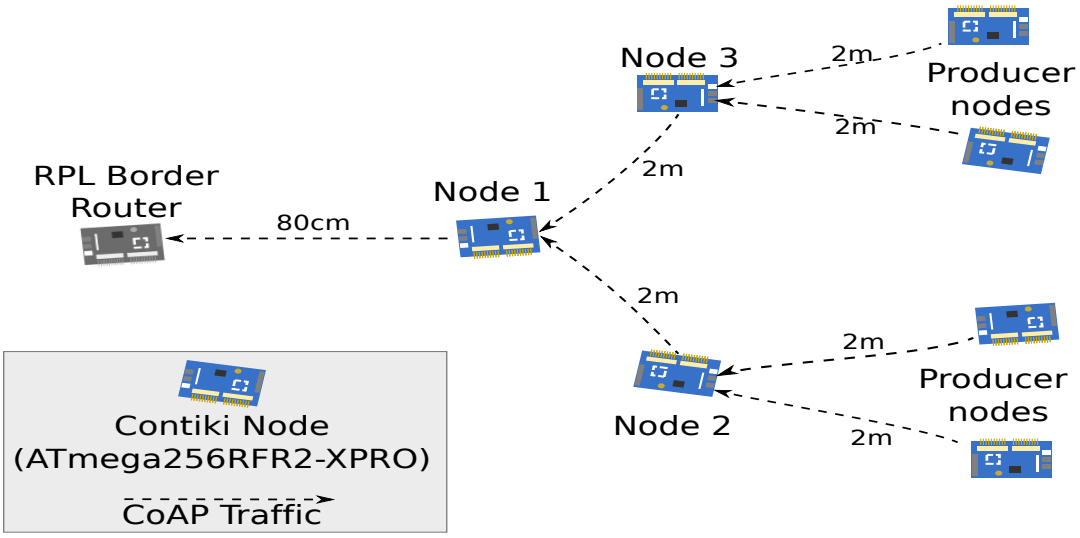
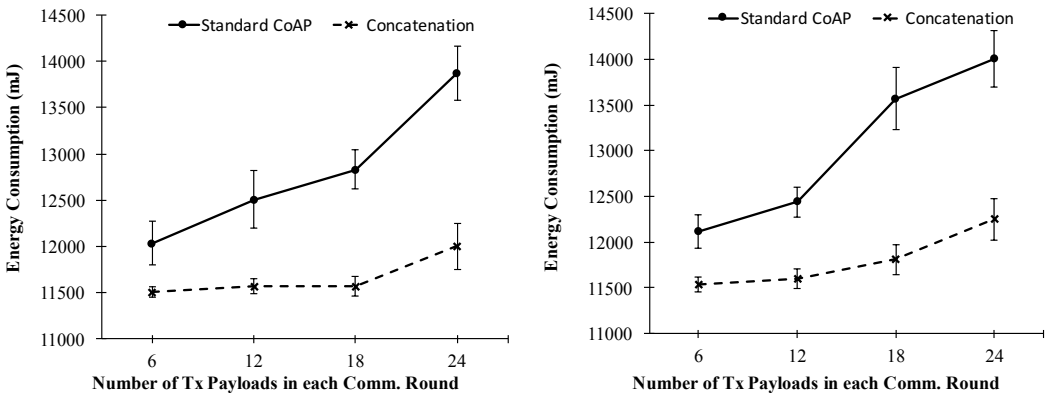


FIGURE 5.2: Evaluation Scenario topology

The duration of each execution was 10 minutes, the first 4 minutes were discarded as a warmup time. In Figure 5.2 the 4 boards that connect to both nodes 2 and 3 were producing data at different time intervals. The number of packets per round was the same of the threshold experiment but the communication round time was 120 seconds. Each packet sent had a payload of 2 bytes. All obtained results were computed based on 15 independent experiments. The energy measurements were obtained by the Energest.



(A) Node 2 energy

(B) Node 3 energy

FIGURE 5.3: Energy Consumption

Figure 5.3 depicts the difference in energy consumption between packet forwarding and data concatenation in nodes 2 and 3. Using the topology represented in Figure 5.2. Table 5.3 contains the difference between forwarding and packet concatenation. The values show a decrease in energy consumption up to 14.96% in the case of 24 payloads in a single message.

TABLE 5.3: Energy difference between Forwarding and Packet concatenation

	Difference Node 2	Difference Node 3	Average difference
6	4.54%	4.99%	4.77%
12	8.06%	7.23%	7.65%
18	10.91%	14.86%	12.88%
24	15.62%	14.30%	14.96%

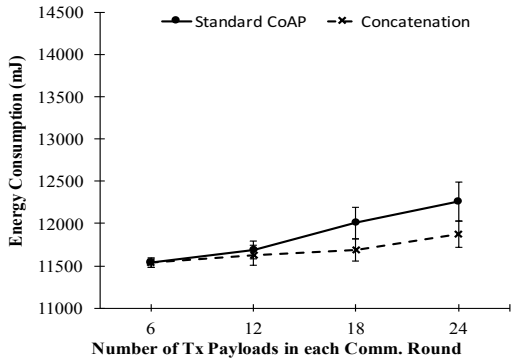


FIGURE 5.4: Energy consumption in node 1

Figure 5.4 depicts the difference in energy consumption in node 1. The difference in energy consumption is up to 3.2% when 24 packets are sent in each communication round. This value is low, particularly because node 1 is a bottleneck in this topology. So the energy savings should be higher because of the difference in the number of sent packets. The cause of this values was only discovered after performing the experiments in Chapter 6 and will be discussed later.

5.4 Summary

This Chapter started by presenting the data aggregation techniques that were implemented. Then a small scale test was conducted to find how many packet a node had to aggregate to reduce the energy consumption. This test was redone and the results were different. Those results showed that even when just 6 payloads were concatenated the energy consumption could be reduced.

Chapter 6

Cross layer group-based data aggregation

This Chapter contains the information regarding the implementation of the group-based in-network data aggregation solution. Section 6.1 introduces the developed solution and Section 6.2 details the implementation. Section contains an evaluation and discussion of the results. The chapter ends with a summary.

6.1 Introduction and Goals

Leveraging previous work presented in Chapter 3, this chapter introduces the developed in-network data aggregation solution for multi-hop networks.

Typically in an IOT network deployment there are several nodes measuring the same target [33] (e.g. Temperature or humidity). In a multi-hop network each of those nodes will be producing its own data and forwarding data received from its neighbours. If several nodes are producing data of the same type groups of nodes can be formed. Those groups are comprised of nodes with similar configurations. Figure 6.1 represents two different groups measuring different targets inside a multi-hop network.

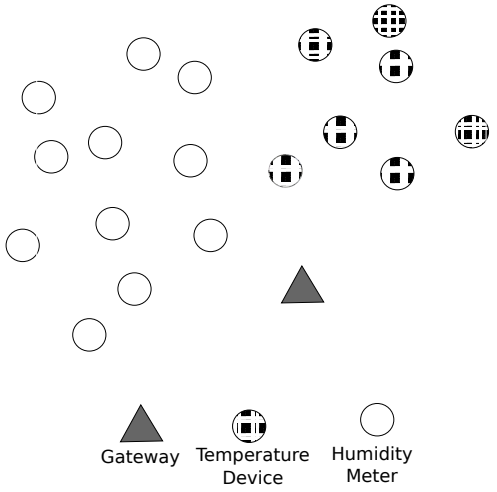


FIGURE 6.1: Diferrent groups inside the same network

Since the nature of the data exchanged by the elements of the same group is similar, that data can be subject to aggregation. When one node measuring temperature receives a packet from other node, also measuring temperature, that data can be merged together using a mathematical function (e.g. average value). Now instead of sending 2 packets with temperature values only one packet is sent. That packet will contain the average temperature recorded by those two nodes. Several groups can be created inside a network and the data from those different groups can also be combined to reduce further the number of messages being sent.

6.2 Solution Description

This section contains the details of the group based aggregation solution.

6.2.1 Objectives

As discussed in the previous Chapter, transmitting and receiving data is a costly operation in terms of energy. If the number of messages exchanged inside a network is reduced the radio will be used less often saving power. The main objective of this data aggregation solution is to save energy by performing in-network data aggregation. That means every node forwarding packets has the possibility of merging its data with the data it is forwarding.

6.2.2 Requirements

The proposed data aggregation technique requires two things to work. First some extra information needs to be added to the packet to differentiate

between different groups. Second, it is necessary to implement an analysis and decision layer between the IP and UDP layers.

In order to support multiple group aggregation fixed sized units had to be created. Each of those group units contains enough information about the data produced by its members. Another fixed unit was also necessary. Its purpose was to give information about how many group units follow and if the data produced by that group could be concatenated with other group units.

6.2.3 Data units and underlying layer

The data produced by members of the same group is to be subject to internal aggregation with a mathematical function (e.g. average value). There is also the possibility of merging together packets from different groups by concatenating their group units. This reduces even further the number of packets flowing in the network.

As depicted in Figure 6.2, the data units consists of a 1 byte unit and one or more 4 byte group payload units. The fixed unit contains information about whether data from different groups can be concatenated to that packet and how many group units follow. Each group unit belongs to a different group.

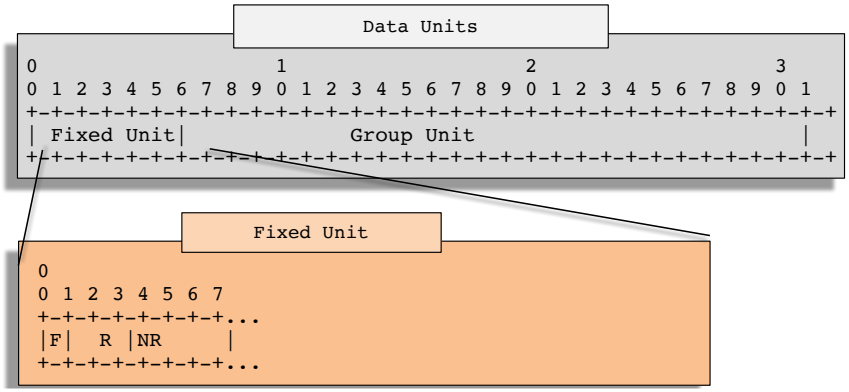


FIGURE 6.2: The fixed unit fields

The fields in the fixed unit, shown in Figure 6.2, are defined as follows:

External concatenation (F): 1 bit unsigned integer.A value of 1 indicates that data from a different group can be appended to that packet. A value of 0 will mean that packet will be forwarded by members of a different group

Reserved (R): 3 bit unsigned integer. Reserved for future use.

Number of units (NR): 4 bit unsigned integer. This value indicates how many group units follow, each group unit contains data belonging to one CoAP group. If one specific group does not allow external concatenation this value will always be 1.

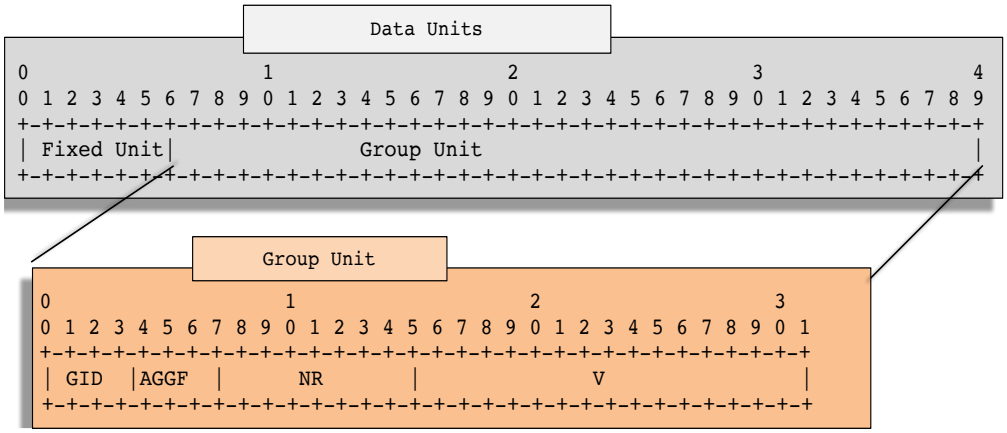


FIGURE 6.3: The Group Unit

The group unit, shown in Figure 6.3, contains the data of a single CoAP group. One message has one or group units. It is defined as follows: **GroupID (GID)**: 4 bit unsigned integer. Indicates the ID number of the group this unit belongs to.

Aggregation function (AGGF): 4 bit unsigned integer. This field indicates the number of the corresponding mathematical function used. (0000) value corresponds to the MIN function, (0001) corresponds to the MEAN function and (0010) corresponds to the MAX function, no other values are supported yet.

Number of values (NR): 8 bit unsigned integer. This value indicates how many different values were used to produce the value V (e.g. an average of 10 numbers).

Value (V): 16 bit unsigned integer. This field indicates the value of the resource being measured (e.g. Temperature or humidity).

The size, in bits, of each of the fields in both the Fixed and the Group Unit were developed with the basis of measuring targets with a small range of values. These targets include temperature, humidity, co2 levels and others that values that fit into 16 bits.

6.2.4 Decision Mechanism

As referenced in Section 6.2.3 this data aggregation solution relies on both a light-weight layer and a standard packet.

The light-weight layer is responsible for analysing incoming packets and deciding if the data can be stored in the buffer to be aggregated. That decision process relies on the analysis of the fields of the header mentioned above.

Algorithm 2 Decision algorithm

```
1: Start
2:     function NEW_PACKET_ARRIVES(msg)
3:         typeCode ← parseCode(msg)
4:         address ← parseAddrs(msg)
5:         if isCoAP(typeCode) and notMyAdd(address) then
6:             if Inspect_packet(msg) then
7:                 cancel_forwarding and aggregate
8:             else
9:                 forward_next_layer(buffer)
10:            end if
11:        end if
12:    end function
13:
14:    function INSPECT_PACKET( )
15:        if packet has my groupID then
16:            Aggregate_group (return 1)
17:        else
18:            if Can I concatenate externaly(F) then
19:                if Does the packet allow external concatenation then
20:                    Aggregate_other_group (return 1)
21:                else
22:                    forward_next_layer (return 0)
23:                end if
24:            else
25:                forward_next_layer (return 0)
26:            end if
27:        end if
28:    end function
29: End
```

Algorithm 2 shows the decision process behind the group aggregation. First, in line 2 when the packet arrives it is necessary to check if it is a CoAP message. If it is and it was not self-produced it is necessary to check the header to verify if that packet is suitable for data aggregation. In the Inspect_Packet() function the Fixed Header is checked, the *NR* field is accessed to check how many different Group Units have to be inspected. If any of those Group Units has a GroupID value equal to the nodes, the data is stored and will be aggregated only to the correct group. The forwarding of the packet will be cancelled.

If the packet does not have any Group Unit with the same GroupID of the node, it is necessary to verify if that packet can carry the data from more than one group and if the data from the node itself can carry it. In the Algorithm

2, at lines 18 and 19 that verification is made. First the node checks if it can concatenate its own data. If it can and if that packet allows it the data will be stored. In the negative case the packet is forwarded.

As an example, one node receives packets from two different groups, both suitable for external aggregation, as well as the node. When a message is to be sent the Fixed Header will have an *F* value of 1 and an *NR* value of 2 meaning two 4 byte Group Units will follow. Those Group units will be fulfilled with the data stored in their respective section of the buffer.

6.3 Evaluation

The discussed aggregation method was implemented on a testbed to obtain real-world measurements. In a simulator there are several behaviours that cannot be accurately represented. Some of those behaviours include the interference between nodes, the energy drain of the node or packet loss.

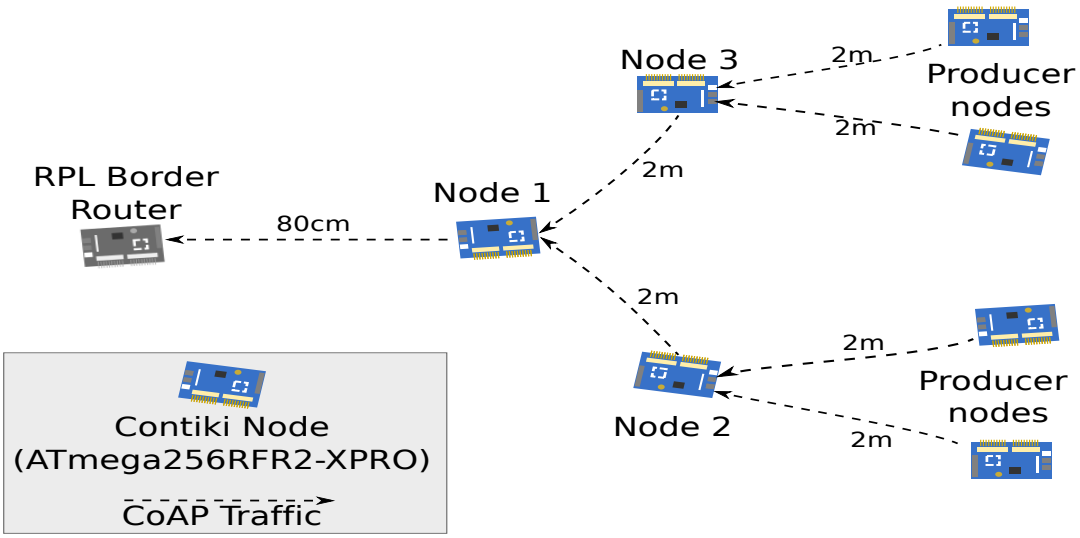


FIGURE 6.4: Evaluation Scenario topology

The testbed scenario is similar to the one used in Chapter 5. In the testbed scenario the 8 Contiki nodes were deployed to create a balanced tree topology in an indoor environment. The distance between each board is depicted in Figure 6.4, the distance between nodes may have varied between experiments by a few centimetres.

The duration of each execution was 10 minutes, the first 4 minutes were discarded as a warmup time. In Figure 6.4 the 4 boards that connect to both nodes 2 and 3 were producing data at different time intervals. The time intervals were 10, 13.2, 20 and 40 seconds. Nodes 2 and 3 were either forwarding data or aggregating the packets. The communication round time is 120 seconds. When the nodes 2 and 3 produce an aggregated packet the size of the payload is 5 bytes.

All obtained results were computed based on 10 independent experiments.The energy measurements were obtained by the Energest.

6.3.1 Results and Discussion

This section contains the results of the experimental scenario described in 6.3. The metrics gathered were: energy consumption and packet loss.

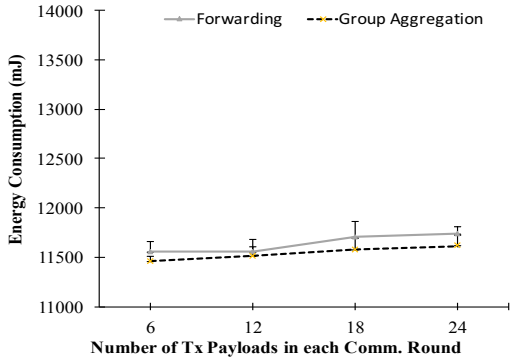


FIGURE 6.5: Node 1 energy

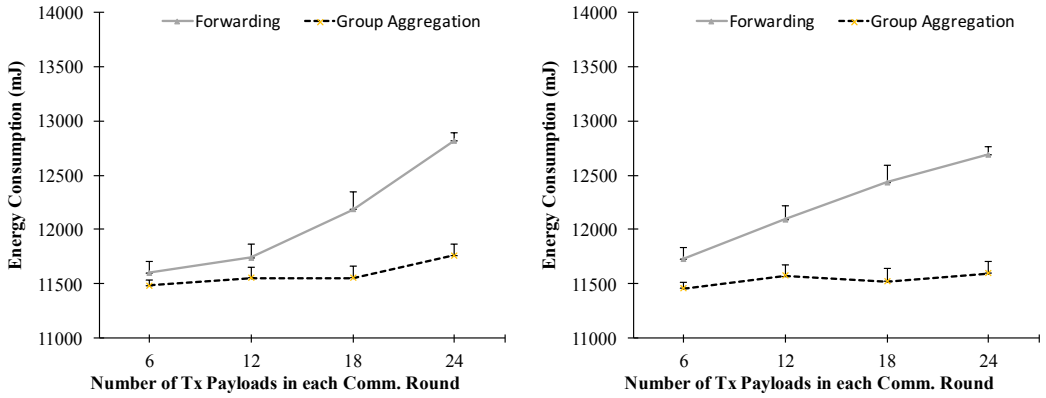
Figure 6.5 contains the energy comparison between simply forwarding or aggregating data. The difference is small, lower than 1,14%. This happens because although the Border router was configured to have a Channel Check Rate (CCR) rate of 8Hz, its code turns the Radio Duty Cycling (RDC) protocol off to ensure a high packet reception. Having no RDC protocol means the border router is always listening and the packet loss between itself and node 1 is zero.

TABLE 6.1: Packet loss in node 1

	Forwarding	Group Aggregation
6	6,50%	0,69%
12	7,11%	5,45%
18	5,35%	3,61%
24	11,09	8,14%

Table 6.1 contains the packet loss in node 1. This packet loss is related to the number of packets sent by nodes 2 and 3. When performing data aggregation the number of packets is smaller, 6 packets in total, although the number of values they hold is different. So in order to be able to compare packet loss, a function that counts the number of values in each packet was created. This function just accesses the *NR* field in the group unit and ads its value to variable. Losing one packet carrying the aggregated data of twenty four transmissions is different from losing one packet in twenty four. The

number of lost packets in the last case is high because during one experiment only 4 packets are received instead of 6.



(A) Node 2 energy (B) Node 3 energy

FIGURE 6.6: Energy Consumption

Figures 6.6a and 6.6b contain the energy measurement values for each number of packets. When performing data aggregation nodes 2 and 3 send packets at every 120 seconds. This time interval is considered a communication round.

Table 6.2 contains the differences in energy consumption for each node. As expected, the difference increases when a greater number of packets is sent trough the network. The average difference in energy consumption is up to 9.17%.

TABLE 6.2: Energy difference between Forwarding and Group Aggregation

	Difference Node 2	Difference Node 3	Average difference	Stdev
6	1.02%	2.39%	1.7%	0.96%
12	1.61%	4.48%	3.04%	2.03%
18	5.47%	7.91%	6.69%	1.72%
24	8.95%	9.40%	9.17%	0.32%

TABLE 6.3: Energy Consumption by state in node 2

	Tx Payloads	**General CPU**	**Tx**	**Rx**	**CPU for Group Aggregation**
Groups	6	96.08%	0.06%	2.89%	0.05%
	12	95.53%	1.21%	3.16%	0.09%
	18	95.52%	0.65%	3.68%	0.13%
	24	93.69%	1.72%	5.06%	0.17%
Forward	6	95.19%	1.54%	3.25%	n/a
	12	94.10%	2.99%	2.90%	n/a
	18	90.68%	4.06%	5.24%	n/a
	24	86.18%	6.94%	6.86%	n/a

Table 6.3 contains a comparison between forwarding and group aggregation in node 2. This comparison is based on percentage of energy consumed by state. So when the number of packets per round is 24 there is notable difference particularly in CPU and TX percentages. When forwarding the number of packets forwarded by round is much greater and naturally the TX state consumes more energy. The energy spent in the RX state is similar in both cases. Table 6.4 contains the packet loss at node 2. The difference between both sets of values is not very high.

TABLE 6.4: Packet loss in node 2

	Forwarding	Group Aggregation
6	3.33%	1.66%
12	1.11%	2.77%
18	0.18%	0.55%
24	5%	2.91%

Table 6.5 contains a comparison between forwarding and group aggregation in node 3. The values are similar to the ones of node 2. The difference in the values is not very significant.

TABLE 6.5: Energy Consumption by state in node 3

	Tx Payloads	**General CPU**	**Tx**	**Rx**	**CPU for Group Aggregation**
Groups	6	96.02%	0.99%	2.93%	0.05%
	12	95.04%	0.95%	2.93%	0.09%
	18	95.43%	1.03%	3.40%	0.13%
	24	94.64%	1.74%	3.44%	0.17%
Forward	6	93.81%	2.48%	3.70%	n/a
	12	91.07%	4.17%	4.74%	n/a
	18	88.60%	5.84%	5.53%	n/a
	24	86.71%	7.71%	5.68%	n/a

Table 6.6 shows the packet loss of node 3, again these values are similar to the ones of node 2.

TABLE 6.6: Packet loss in node 3

	Forwarding	Group Aggregation
6	3,33%	1,66%
12	1,11%	2,77%
18	0,18%	0,55%
24	5%	2,91%

The obtained results clearly show improvement in energy consumption. As expected the improvement increases with higher traffic rates. The results of nodes 2 and 3 are similar both in terms of energy consumption and packet loss.

The results from node 1 show that being closest to the border router does not always mean higher energy consumption. This is mainly due to the router keeping the radio permanently in the receive state. If the experiment had a higher number of hops the energy savings should be even higher because of the reduction in the traffic rate.

6.4 Summary

In this Chapter the Cross layer data aggregation technique was presented. This technique consisted in creating a light-weight layer to parse incoming packets and extra information to the packet itself to distinguish between different groups. A test scenario similar to the one used before was deployed. The results also showed that this aggregation reduced the energy usage despite not being the optimal test scenario. This group-based mechanism is expected to benefit from a scenario where a greater number of nodes is involved.

Chapter 7

Project Management

This chapter describes the activities that took place in the first and second semester. Each subsection contains the expected tasks to be performed and the actual tasks performed.

7.1 First Semester

The work during the first semester consisted mainly of research for the state of the art and preliminary experimental work. The next two Gantt charts present the proposed schedule and the effective schedule.

- Task 1: Research state-of-the-art in IOT communications
- Task 2: Research state-of-the-art in energy efficiency/data management
- Task 3: Characterization of dense IoT environments in terms of energy consumption, reliability and scalability
- Task 4: Specification of the mechanisms for energy-efficient, reliable and scalable IOT communications in dense scenarios
- Task 5: Writing of first semester report

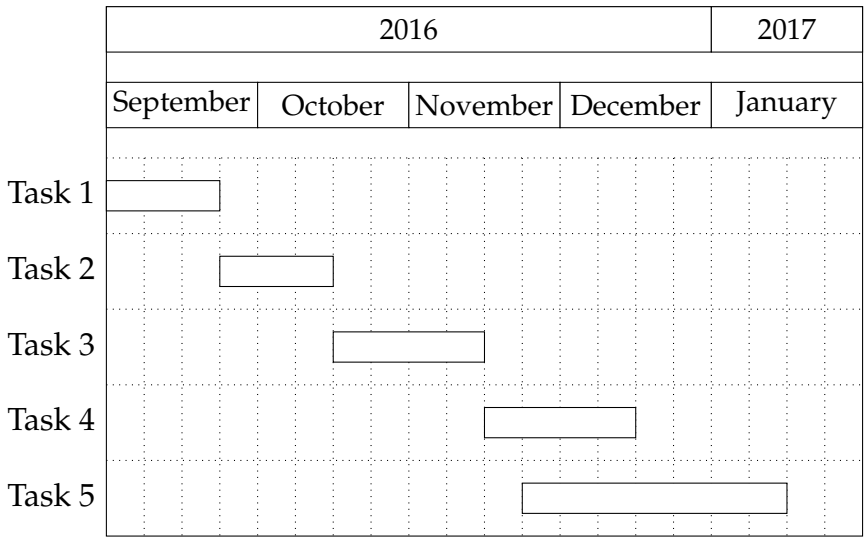


FIGURE 7.1: Proposed first semester plan

The actual tasks carried out during the first semester were different from those initially proposed. They are listed bellow:

- Task 1: Research about IOT operating systems
- Task 2: Research about IOT deployments
- Task 3: Research about energy efficiency in IOT networks
- Task 4: Research about data aggregation techniques
- Task 5: Installing Contiki on the hardware
- Task 6: Experiments with radio and overall tests with the hardware
- Task 7: Data aggregation implementation and testing
- Task 8: Writing of the first semester report

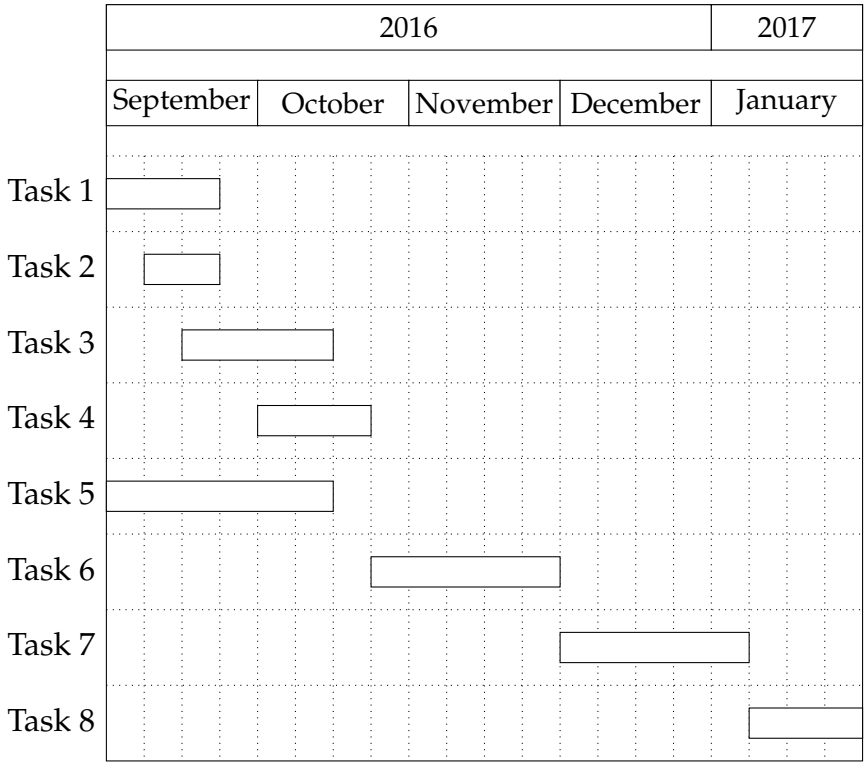


FIGURE 7.2: Effective first semester plan

7.2 Second Semester

This subsection also contains the planned work schedule and the effective that took place during the second semester. There were some changes to the proposed schedule as well as some delays.

The proposed schedule was the following:

- **Task 1- Implementation of the first version of the group communications (06/02/2017 - 24/03/2017):** This task starts by finishing the issues from the previous semester. This includes correcting the internal clock deviation and implementing the payload aggregation. Then the work will begin on the practical scenario. An aggregation module will be developed and include the necessary functions to perform the different types of aggregation. A separate module with energy statistics will also be included and its task will be to measure the energy spent on each state (Transmission, Listening, Sleep and Active) by using the Energest [7].
- **Task 2- Evaluation of the mechanisms created in the previous task (27/03/2017 - 7/04/2017):** In this task the previously created mechanisms will be tested on the testbed and the results evaluated. The test scenarios include one where no aggregation is performed and one with varying traffic rates and different network topologies. The aggregation parameters will be tuned across several runs of the experiment to measure their impact across the selected metrics mentioned in the previous chapter. The results from the testbed will then be analysed against assumption, such as, is the number of packets received at the border router much smaller when the nodes are performing aggregation. The conclusions withdrawn from those experiments will then be used to improve and modify the testbed.
- **Task 3- Specification of enhancements to the mechanisms developed in the first phase (10/04/2017 - 19/05/2017):** This task will be largely dependant on the results of the last. Firstly the mistakes found when evaluating the testbed will be fixed, this task can include changes to the code, the addition of new parameters or changes in topology.
- **Task 4- Evaluation of the enhancements to the basic mechanisms (22/05/2017 - 9/06/2017):** This evaluation will take into account all the results from previous tasks. First the topologies will be specified, then the number of groups, the duration of the experiments, the traffic interval, the type and size of the messages, the type of internal aggregation function, the maximum length of the aggregated payloads. Other parameters will be added according to previous runs of the experiments. Then the results will be evaluated accordingly.
- **Task 5- Scientific paper writing (7/04/2017 - 9/06/2017):** In order to contribute to the ongoing work with data aggregation, this work will be compiled into a scientific paper to present to the community.

- **Task 6- Writing of the final report (7/04/2017 - 30/06/2017):** This report will include a fully detailed version of every step taken to complete this work. It will encompass all the materials consulted and produced along this thesis.

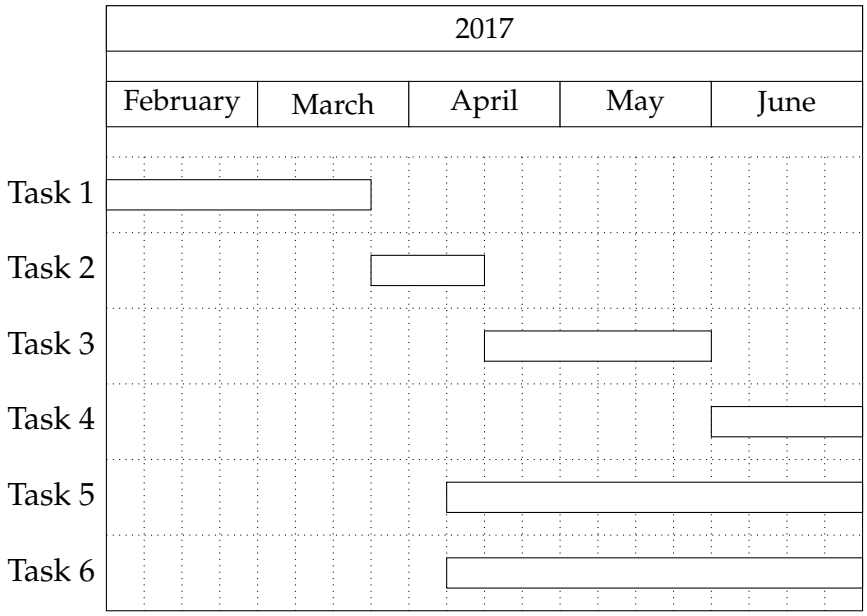


FIGURE 7.3: Proposed second semester plan

The effective plan fulfilled during the second semester was the following:

- **Task 1- Clock validation (13/02/2017 - 17/03/2017):** This task was necessary in order to be able to use the Energest as it relies on tracking the time spent on different Radio and CPU states. This validation included finding out the issues that caused the clock drift, fixing them and then running some tests to measure the clock drift.
- **Task 2- Energest validation (20/03/2017 - 03/04/2017):** With the clock measuring the correct time, to use the Energest it was necessary to measure energy consumption on each state. Then tests needed to be run to check if the values reported by the Energest were close to those measured with the P401r board.
- **Task 3- Testbed preparation (10/04/2017 - 5/05/2017):** This task involved defining parameters and adjusting variables for the testbed. This included adjusting CoAP parameters, defining topologies, traffic rate ,payload size and warmup time and test duration.
- **Task 4- Testing first aggregation scenario (8/05/2017 - 19/05/2017):** This task consisted of testing the aggregation scenario with the parameters defined in the previous. A paper was also written with the results.
- **Task 5- Defining, implementing and testing the Group based aggregation layer (22/05/2017 - 16/06/2017):** This task consisted in developing the CoAP header extension and modifying the underlying layer to support the CoAP groups.
- **Task 6- Writing final report (22/05/2017 - 16/06/2017):** This task represents the writing of the final report. All the work performed, the results and the final conclusions were compiled in this document.

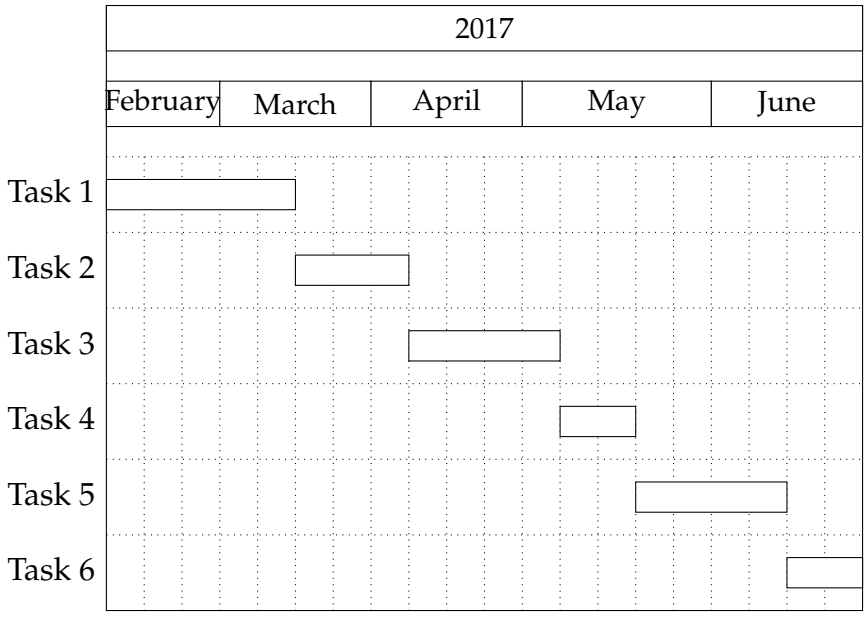


FIGURE 7.4: Effective second semester plan

Chapter 8

Conclusions and Future work

This work contains several contributions worth mentioning. First the testbed was prepared. This involved finding a method to measure the energy consumption. The Energest was a candidate but measurements had to be performed to ensure it was accurate. Test were performed that showed the error was within acceptable parameters.

Then a testbed experiment was set up. This experiment compared two scenarios, one where data was simply being forwarded and other where data was aggregated. This showed application level aggregation can have significant energy savings. The energy savings in this experiment were up to 14.9%. The results of this experiment were submited to *The 20th ACM International Conference on Modeling, Analysis and Simulation of Wireless and Mobile Systems* (MSWiM 2017).

Then a group-based data aggregation was developed and tested. This method of performing data aggregation relied on the creation of groups with similar properties. This method of data aggregation achieved up to 9.4% in energy savings. In this experiment node 1 showed little improvement, mainly because it was sending its data to a router that was permanently listening. If the topology contained a higher number of hops the savings could be even higher.

For future work larger scenarios should be tested with different topologies. As the network density increases the ability to reduce, even by a few percent, the number of messages can generate significant energy savings. Taking into account that sometimes nodes are deployed for several weeks those savings can add up to a significant time increase.

Bibliography

[1] Dave Evans. *The Internet of Things How the Next Evolution of the Internet Is Changing Everything*. URL: http://www.cisco.com/c/dam/en_us/about/ac79/docs/innov/IoT_IBSG_0411FINAL.pdf.

[2] Contiki Developers. *Contiki: The Open Source OS for the Internet of Things*. URL: http://contiki-os.org/#why.

[3] Adam Dunkels, Bjorn Gronvall, and Thiemo Voigt. "Contiki-a lightweight and flexible operating system for tiny networked sensors". In: *Local Computer Networks, 2004. 29th Annual IEEE International Conference on*. IEEE. 2004, pp. 455–462.

[4] Adam Dunkels. "uIP-A free small TCP/IP stack". In: *The uIP* 1 (2002).

[5] Mathilde Durvy et al. "Making sensor networks IPv6 ready". In: *Proceedings of the 6th ACM conference on Embedded network sensor systems*. ACM. 2008, pp. 421–422.

[6] Adam Dunkels et al. "Protothreads: simplifying event-driven programming of memory-constrained embedded systems". In: *Proceedings of the 4th international conference on Embedded networked sensor systems*. Acm. 2006, pp. 29–42.

[7] Adam Dunkels et al. "Software-based on-line energy estimation for sensor nodes". In: *Proceedings of the 4th workshop on Embedded networked sensors*. ACM. 2007, pp. 28–32.

[8] Philipp Hurni et al. "On the accuracy of software-based energy estimation techniques". In: *European Conference on Wireless Sensor Networks*. Springer. 2011, pp. 49–64.

[9] Philip Levis et al. "Tinyos: An operating system for sensor networks". In: *Ambient intelligence*. Springer, 2005, pp. 115–148.

[10] *What is IoT-LAB?* https://www.iot-lab.info/what-is-iot-lab/. Accessed: 12-01-2017.

[11] Niclas Finne et al. "Experiences from Two Sensor Network Deployments—Self-Monitoring and Self-Configuration Keys to Success". In: *International Conference on Wired/Wireless Internet Communications*. Springer. 2008, pp. 189–200.

[12] *Publications*. http://www.ict-calipso.eu/category/publications/. Accessed: 12-01-2017.

[13] Anton Hergenröder, Joachim Wilke, and Detlev Meier. "Distributed energy measurements in WSN testbeds with a sensor node management device (SNMD)". In: *ARCS 2010* (2010).

[14] Geoffrey Werner-Allen, Patrick Swieskowski, and Matt Welsh. "Motelab: A wireless sensor network testbed". In: *Proceedings of the 4th international symposium on Information processing in sensor networks*. IEEE Press. 2005, p. 68.

[15] Jang-Ping Sheu, Chia-Chi Chang, and Wei-Sheng Yang. "A distributed wireless sensor network testbed with energy consumption estimation". In: *International Journal of Ad Hoc and Ubiquitous Computing* 6.2 (2010), pp. 63–74.

[16] Adam Dunkels et al. "Powertrace: Network-level power profiling for low-power wireless networks". In: (2011).

[17] Adam Dunkels. "The contikimac radio duty cycling protocol". In: (2011).

[18] Michael Buettner et al. "X-MAC: a short preamble MAC protocol for duty-cycled wireless sensor networks". In: *Proceedings of the 4th international conference on Embedded networked sensor systems*. ACM. 2006, pp. 307–320.

[19] Tim Winter. "RPL: IPv6 routing protocol for low-power and lossy networks". In: (2012).

[20] Thiemo Voigt, Hartmut Ritter, and Jochen Schiller. "Solar-aware routing in wireless sensor networks". In: *IFIP International Conference on Personal Wireless Communications*. Springer. 2003, pp. 847–852.

[21] Giuseppe Anastasi et al. "Energy conservation in wireless sensor networks: A survey". In: *Ad hoc networks* 7.3 (2009), pp. 537–568.

[22] Jamal N Al-Karaki, Raza Ul-Mustafa, and Ahmed E Kamal. "Data aggregation and routing in wireless sensor networks: Optimal and heuristic algorithms". In: *Computer networks* 53.7 (2009), pp. 945–960.

[23] Zi Li et al. "Lifetime balanced data aggregation for the internet of things". In: *Computers & Electrical Engineering* (2016).

[24] André Riker et al. "A Two-Tier Adaptive Data Aggregation Approach for M2M Group-Communication". In: *IEEE Sensors Journal* 16.3 (2016), pp. 823–835.

[25] Tian He et al. "AIDA: Adaptive application-independent data aggregation in wireless sensor networks". In: *ACM Transactions on Embedded Computing Systems (TECS)* 3.2 (2004), pp. 426–457.

[26] *Atmega256rfr2 datasheet*. 8393C-MCU Wireless-09/14. Atmel Corporation. 2014.

[27] Z. Shelby, K. Hartke, and C. Bormann. *The Constrained Application Protocol (CoAP)*. RFC 7252 (Proposed Standard). Updated by RFC 7959. Internet Engineering Task Force, June 2014. URL: `http://www.ietf.org/rfc/rfc7252.txt`.

[28] C. Holmberg, I. Sedlacek, and G. Salgueiro. *UDP Transport Layer (UDPTL) over Datagram Transport Layer Security (DTLS)*. RFC 7345 (Proposed Standard). Internet Engineering Task Force, Aug. 2014. URL: `http://www.ietf.org/rfc/rfc7345.txt`.

[29] T. Winter et al. *RPL: IPv6 Routing Protocol for Low-Power and Lossy Networks*. RFC 6550 (Proposed Standard). Internet Engineering Task Force, Mar. 2012. URL: http://www.ietf.org/rfc/rfc6550.txt.

[30] N. Kushalnagar, G. Montenegro, and C. Schumacher. *IPv6 over Low-Power Wireless Personal Area Networks (6LoWPANs): Overview, Assumptions, Problem Statement, and Goals*. RFC 4919 (Informational). Internet Engineering Task Force, Aug. 2007. URL: http://www.ietf.org/rfc/rfc4919.txt.

[31] Isam Ishaq et al. "IETF standardization in the field of the internet of things (IoT): a survey". In: *Journal of Sensor and Actuator Networks* 2.2 (2013), pp. 235–287.

[32] *MS Windows NT Kernel Description*. https://github.com/contiki-os/contiki/wiki/Timers. Accessed: 2017-03-03.

[33] Robert Szewczyk et al. "Lessons from a sensor network expedition". In: *European Workshop on Wireless Sensor Networks*. Springer. 2004, pp. 307–322.

Appendix A

Appendix A

This Appendix contains the paper submitted during the development of this thesis.

A New Perspective for Energy Efficiency in the Internet of Things

André Riker, João Subtil, Marilia Curado, Edmundo Monteiro
Centre for Informatics and Systems
Department of Informatics Engineering
University of Coimbra
Coimbra, Portugal
{ariker, jsubtil, marilia, edmundo}@dei.uc.pt

ABSTRACT

Internet of Things (IoT) protocols provide the fundamental mechanisms to collect data from low power devices and lossy networks. The IoT protocols collect data blocks from the devices in messages that have one header and a single payload, regardless the size of the payload. This paper presents a solution to collect small size data blocks from low power devices in an efficient way, carrying these data blocks in the payload of a single message. The proposed solution is a light-weight layer designed to operate with the standard IoT protocol stack aiming to reduce the energy consumption of the energy constrained devices without lowering the data accuracy. The proposed solution was developed in Contiki devices and the measurements conducted on a testbed showed improvements of up to 14% in the energy consumption.

CCS CONCEPTS

• **Networks** → **Network experimentation**; *Traffic engineering algorithms*; *In-network processing*;

KEYWORDS

Energy Efficiency; Overhead; IoT.

ACM Reference format:
André Riker, João Subtil, Marilia Curado, Edmundo Monteiro. 2017. A New Perspective for Energy Efficiency in the Internet of Things. In *Proceedings of ACM International Conference on Modeling, Analysis and Simulation of Wireless and Mobile Systems, Miami, USA, Nov 2017 (MSWiM),* 5 pages.
https://doi.org/10.1145/nnnnnnn.nnnnnnn

1 INTRODUCTION

Energy efficiency is a major concern within the context of the Internet of Things (IoT). Data aggregation has been applied to reduce energy consumption, with special focus on periodic many-to-one traffic, which is common in applications such as smart metering and building monitoring. With such approach it is possible to reduce the amount of messages generated by these applications. However, traditional data aggregation reduces the accuracy of the collected information. In most solutions, the data-receiver entity has only generic information about the network, since data aggregation resorts to statistical functions on the data (e.g. Maximum, Minimum, Sum, and Average). Thus, it is still a challenge to reduce the energy consumption of IoT communications while keeping data accuracy.

This paper presents an approach that is able to reduce the energy consumption and also maintain the accuracy of the collected data. The proposed solution, named Messaging mULTIple Payloads LayEr (MULTIPLEx), allows the low power nodes to produce messages with multiple payloads when the node is producing its own messages or forwarding messages. MULTIPLEx assembles messages with multiple payloads to exploit the fact that most of the periodic many-to-one traffic in IoT environments is composed by payloads that correspond to less than 10% of the whole message. Besides, MULTIPLEx is able to reduce the energy consumption of the nodes without reducing the data accuracy, since the content of all payloads is preserved. The paper is structured as follows. Section 2 and 3 introduce the related work and MULTIPLEx. Section 4 presents the real implementation and the performance evaluation experiments. Section 5 shows the conclusions and future works.

2 RELATED WORKS

The idea of inserting many payloads inside the same message is also proposed by Stasi et. al [4] and Tsitsipis et. al [3]. Although these solutions were not implemented for IoT protocols, they show that a network can significantly improve the communication efficiency by transmitting messages with several payloads combined. Stasi et. al [4] propose a layer that extracts all payloads at every communication hop and decides whether the extracted payloads should be inserted again in a single message or not. This solution allows the network to improve throughtput, but does not improve the energy consumption. Tsitsipis et. al [3] propose an approach in which a node inserts all the payloads received from the neighbors inside a single message, but it also inserts self-produced data on the payload until the new message reaches the maximum payload size. Thus, the solution proposed by Tsitsipis et. al improves the energy spent on message headers,

MSWiM, Nov 2017, Miami, USA

ACM ISBN 978-x-xxxx-xxxx-x/YY/MM. . . $15.00
https://doi.org/10.1145/nnnnnnn.nnnnnnn

but it does not reduce energy consumption because every created message has the same payload size. On contrary of the works presented by Stasi et. al and Tsitsipis et. al, MULTIPLEx is proposed for IoT protocols with the objective of reducing the overall energy consumption of the nodes.

In the literature, controlling the number of payloads is related to data aggregation approaches. In most cases, as presented by Becchetti et al [2] and Li et al [8], data aggregation solutions are designed to reduce to one all the payloads received during a time interval. To compute the single payload, these solutions use simple mathematical functions, such as maximum, minimum, average, or sum. In these solutions, the data accuracy is largely reduced, since the final destination receives one payload for the entire network.

Riker et. al [9] present a solution considering multiple groups on the network, named Two-Tier Aggregation for Multi-target Application (TTAMA). This solution aggregates the payloads using a simple mathematical function while the payloads are transiting inside a group. Outside groups, all the payloads are preserved, creating messages with multiple payloads. However, the cost to create the multiple payload messages is not considered. In addition, TTAMA is not evaluated on real devices.

Harb et. al [5] present an approach implemented and evaluated on real devices that performs two types of procedures: one is executed by the nodes and the other is performed by the cluter heads. On the nodes, the sensed data is summarized using statistical functions to avoid payload transmissions with raw data. On the cluster-heads, this solution applies algorithms to reduce the similarities found on a large set of payloads. One important drawback of this solution is that in some cases it reduces the data accuracy in 33.8%.

Ishaq et. al [7] present a real implementation of a messaging approach designed for periodic communication of multiple Constrained Application Protocol (CoAP) nodes. In this solution, the gateway aggregates all the CoAP messages received from the nodes, sending a single CoAP message to the final destination. However, in this solution, only the gateway is able to send messages with multiple payloads on a single message, so it does not improve the energy consumption of the low power devices.

In summary, MULTIPLEx advances the related works in the following aspects: (i) it is designed, implemented, and evaluated considering a stack of IoT protocols; (ii) the proposed solution is able to reduce the overall energy consumption of the low power devices without reducing the data accuracy; (iii) the implementation of the proposed solution is cost effective on a real testbed and considers the real costs to execute the proposed approach.

3 MESSAGING MULTIPLE PAYLOADS LAYER

This section describes the generic ideas of the proposed solution, named Messaging mULTIple Payloads LayEr (MULTIPLEx). As Figure 1 shows, MULTIPLEx stores in a buffer the application messages received from the neighbors and also the self produced messages. The application messages are only stored if the node is not the final destination of the message. A timer is set periodically to verify if the MULTIPLEx criterion has been satisfied for issuing a multiple payload message. In case the criterion is positively verified, the payloads are extracted and information can be added to each individual payload, such as node id, data-type or timestap. Then, the extracted payloads are inserted as a bulked payload into a new application message. In case the criterion has not been satisfied, MULTIPLEx does not change any information in the messages. In both cases, the resulting messages are forwarded to the appropriate layer.

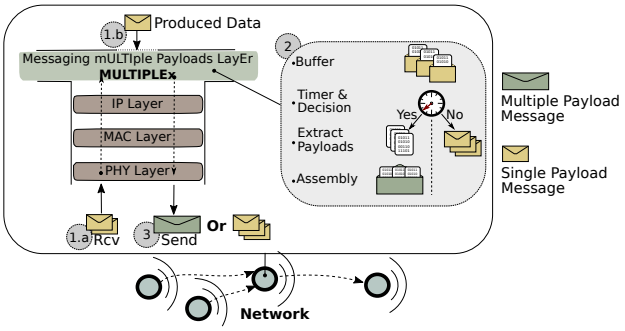


Figure 1: MULTIPLEx overview

MULTIPLEx can be designed with different criteria to decide if a new message will be created with multiple payloads or not. In this paper, MULTIPLEx applies a simple, but effective criterion for this decision. The decision is based on a threshold, called δ, which is related to the number of stored messages. When the timer expires, MULTIPLEx counts the number of messages in the buffer. If this number is equal or greater than δ, then a new multiple payload message will be produced. Otherwise, all messages are preserved.

The decision based on the number of received messages is justified by the fact that the extraction and assembly operations demand a considerable amount of processing and energy resources. Therefore, MULTIPLEx is designed to perform these operations when the energy savings due to the message header suppression payoff the costs.

4 EXPERIMENTAL CASE STUDY

This section presents a case study where MULTIPLEx is implemented on real Contiki devices. Section 4.1 presents the system and the hardware. Section 4.2 shows the details of the MULTIPLEx implementation in Contiki. Section 4.3 describes the real measurements used to set the parameters of the energy consumption software. Section 4.4 and 4.5 present the tuning study for threshold δ and the obtained results in the testbed, respectively.

4.1 System and Harware Specifications

MULTIPLEx can be applied to different IoT application protocols, for instance CoAP [10] and Message Queuing Telemetry Transport (MQTT) [1]. For this case study, CoAP has been selected as the application protocol because it supports natively many-to-one traffic and has the "*observe*" option that enables a client to receive data periodically.

Contiki has been selected as the operating system. Some reasons that motivated this choice are: (i) Contiki provides stable open-source version of the IEEE standard protocols; (ii) it has a large community of developers; and, (iii) it is designed for several low-energy hardware platforms.

Regarding the hardware, the Contiki nodes run on the ATmega256RFR2-XPRO board[1]. It has a 16 MHz Micro-Controller integrated with a 2.4Ghz transceiver, 256 KBytes of flash, and one temperature sensor.

4.2 Implementation of MULTIPLEx

MULTIPLEx is implemented on Contiki as a light-weight layer, located between the Internet Protocol (IP) and the User Datagram Protocol (UDP) layers. MULTIPLEx was implemented between these layers in order to verify the destination IP address of every message received from the neighbors. This verification is necessary because payloads are extracted for the creation of new CoAP messages if the message has not reached the final destination. For the case of a CoAP message arriving from the UDP layer, MULTIPLEx assumes that this message has been produced by the node itself, so it does not extract the payload from this message.

Algorithm 1 shows the main logic executed by MULTIPLEx when a new packet arrives from the IP layer and also presents the algorithm executed when the timer triggers the decision of creating a new multiple payload message.

Algorithm 1 MULTIPLEx algorithm

```
1: Initialize: threshold
2: Start
3:     function NEW_PACKET_ARRIVES(msg)
4:         typeCode ← parseCode(msg)
5:         address ← parseAddrs(msg)
6:         if isCoAP(typeCode) and notMyAdd(address) then
7:             buffer(msg)
8:             if timer_not_set() then
9:                 set_timer()
10:            end if
11:        end if
12:    end function
13:
14:    function TIMEOUT_CALLBACK( )
15:        msg_num ← count_msg(buffer)
16:        if isGreater(msg_num, threshold) then
17:            payloads_data ← extractPayloads(buffer)
18:            new_coap ← assembly_new_coap(payload_data)
19:            forward_next_layer(new_coap)
20:        else
21:            forward_next_layer(buffer)
22:        end if
23:    end function
24: End
```

In Algorithm 1, from line 3 to 12, MULTIPLEx implements a simple buffer for messages arriving from the IP layer and sets a timer. In lines 14 to 23, MULTIPLEx decides if it will create a new CoAP message having multiple payloads. Depending on the number of stored packets on the buffer, the creation of a multiple payload message is cost-efficient or not. For this reason, it is necessary to set the threshold δ with an adequate value.

4.3 Calibrating the Contiki's Energy Tool

Contiki contains an Energy Consumption Tool, called Energest, that is able to estimate how much energy a node has consumed. Energest uses a Contiki service that maintains a table with the total time the CPU and transceiver have been active. This service produces time stamps when the component is turned on and off. Having these time stamps, it is possible to estimate the energy consumption of the node, as presented in Equation 1.

$$\frac{E}{V} = I_m t_m + I_t t_t + I_r t_r \quad (1)$$

In Equation 1, t_m, t_t, and t_r are the time the microcontroller is in the following states: active, transmitting, and receiving, respectively. Besides, the set of constants $\{I_m, I_t, I_r\}$ represents the electrical current necessary to run each of these states. To achieve an accurate estimation of the total energy consumption via Energest it is necessary to calibrate the values of the set $\{I_m, I_l, I_t, I_r\}$. This calibration was performed using a two step methodology. First, real measurements were performed on the Contiki devices to find the electrical current in each state. Second, the total energy consumption estimation provided by Energest was compared to the real measurements.

The MSP432 P401R[2] board was used to measure all the real current energy consumption of the ATmega256RFR2-XPRO boards. The P401R board can measure the amount of energy consumed and the electrical current of a target board with a resolution of 2kHz.

4.3.1 Current Measurements of the States. The energy states have been measured in 10 different ATmega256RFR2-XPRO boards and executed twice for each board. Each test lasted 10 minutes and the first two minutes were considered as a warm-up period. Table 1 shows the obtained values of the current for each state.

Table 1: Current Measurement of the States

State	CPU	Transmission	Reception
Electrical Current	9.32 mA*	21.44 mA	15.49 mA

*This measurement includes idle and active.

CPU measurements were performed running Contiki without entering in low power mode, since this version of Contiki is not stable when ATmega256RFR2 is in low power mode. The result for CPU is the average electrical current for the whole period of the experiment, in which the transceiver is turned off and the CPU can be active or idle.

[1] http://www.atmel.com/tools/atmega256rfr2-xpro.aspx

[2] http://www.ti.com/lit/ug/slau597c/slau597c.pdf

4.3.2 Real vs Estimated Energy Consumption. Table 2 shows the real and the estimated energy consumption reported by the P401R board and the Energest, respectively. For these tests, 10 boards were subject to three separated executions of 1 hour of operation, which comprises the transmission to a border router of one CoAP message every 30 seconds. The error column in Table 2 shows in percentage how much the estimated energy is different from the real measurements.

Table 2: Total Energy Consumption

Board	Real	Energest	Error (%)	Board	Real	Energest	Error (%)
1	112707.3	115272.3	2.28	6	109282.3	113216.6	3.60
2*	125105.1	127952,3	n/a	7	113827.7	113929.6	0.20
3	108492.0	114006.3	5.09	8	111921.7	114646.3	2.44
4	108710.0	114345.0	5.19	9	109843.7	114412.0	4.16
5	111113.0	114455.0	3.01	10	111296.3	114086.0	2.50
Avg	110799.3	114263.2	3.16				

*Outlier.

Hurni et al [6] present measurements related to the calibration of the Contiki's energy consumption tool. The obtained average error value of 3.16% is compatible with the error interval presented by Hurni et al.

4.4 Preliminar Study of Threshold δ

The objective of this study is to determine the threshold δ, which is the minimum number of payloads on a message able to be cost-efficient in terms of energy consumption. The study of the threshold δ was conducted using four nodes: 1 node running MULTIPLEx, 2 nodes that inject CoAP traffic, and 1 receiver node. For comparison purposes, the standard CoAP has been evaluated in the same scenario. Standard CoAP produces messages containing only a single payload and forwards all the received messages from the 2 injector nodes. Figure 2 shows the obtained results, considering a communication round of 120 seconds.

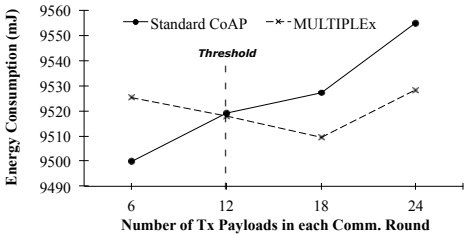


Figure 2: Threshold δ

As can be seen in Figure 2, MULTIPLEx has a lower energy consumption if the number of payloads in the message is equal or greater than 12. Otherwise, it is more efficient to use the standard CoAP, in which the messages have a single payload. The main reason MULTIPLEx does not payoff for less than 6 payloads is related to the cost of extracting and producing a new message.

4.5 Experimental Evaluation and Results

To evaluate the energy consumption of MULTIPLEx, a testbed composed of 8 Contiki nodes was used. As depicted in Figure 3, the 8 Contiki nodes were deployed to create a balanced tree topology in an indoor environment. The experiments were conducted to evaluate MULTIPLEx and the Standard CoAP solution under the same settings. The following set of protocols were used: CoAP, UDP, 6LowPAN, RPL, ContikiMac, and IEEE 802.15.4. All obtained results were computed based on 15 independent experiments. Each experiment lasted for 10 minutes, and the first 4 minutes were defined as the warm-up internal. The measurements were obtained by the Contiki's Energy Consumption Tool (see Section 4.3).

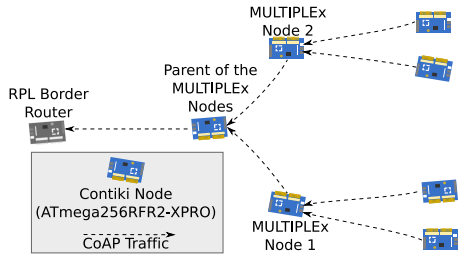


Figure 3: Testbed Topology

The final destination of all injected CoAP messages is an external computer, which is connected to the border router and runs a traffic capture software to collect the received messages. Table 3 presents additional seetings used in the experimental evaluation.

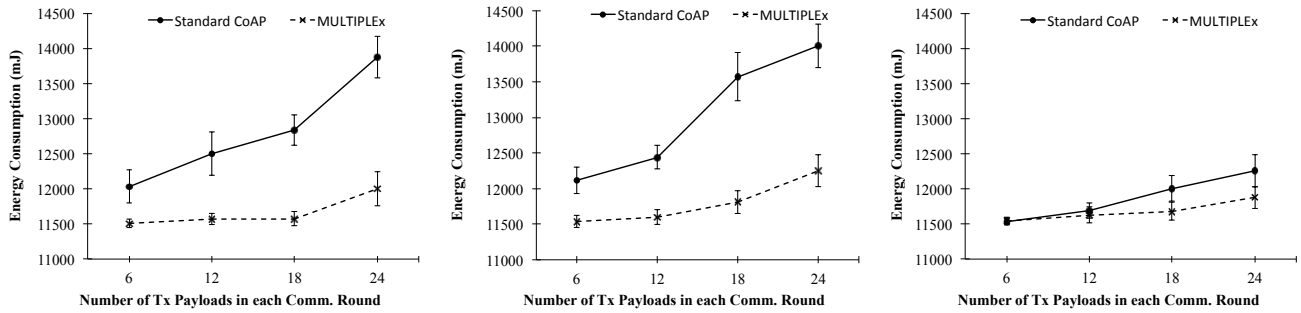
Table 3: Settings of the Real Experiments

Setting	Value
MULTIPLEx Threshold	12 Payloads
Application Protocol	CoAP
Single Payload Size	2 Bytes
MAC Protocol	ContikiMac
ContikiMac Channel Check Freq.	8hz
Wireless Technology	IEEE 802.15.4
Communication Round Interval	60 sec

Among the 8 nodes, 1 node is the border router, 1 node acts as parent for the rest of the network, 2 nodes execute the MULTIPLEx code, and 4 nodes inject CoAP messages in the network. The parent node is set to forward the CoAP messages received.

Figure 4 presents the average total energy consumption and the standard deviation for different numbers of CoAP payloads. In Figures 4a and 4b, the results show the energy consumption of the two MULTIPLEx nodes. Regarding these results and for all measured cases, MULTIPLEx has the lowest energy consumption. The improvement is 14.85% for the case of 24 payloads in a single message.

In Figure 4c, the obtained energy consumption corresponds to the node that acts as parent for the MULTIPLEx nodes.



(a) MULTIPLEx Node 1 **(b) MULTIPLEx Node 2** **(c) Parent Node**

Figure 4: Energy Consumption

In this set of results, MULTPLEx is more cost-effective in the case of 18 and 24 payloads. This figure shows that MULTIPLEx also improves energy consumption of the parent node, since this node forwards less CoAP headers.

Table 4 shows the average energy consumed in each state of the MULTIPLEx nodes. In these results it is possible to measure how MULTIPLEx changes the energy spent on Transmission, Reception, General CPU operation, and also it presents the CPU demanded by MULTIPLEx.

Table 4: Energy Consumption

	Tx Payloads	General CPU	Tx	Rx	CPU for MULTIPLEx
MULTIPLEx	6	95.50%	1.09%	3.36%	0.03%
	12	94.80%	0.97%	4.14%	0.07%
	18	93.30%	1.16%	5.41%	0.11%
	24	89.40%	1.41%	9.03%	0.14%
Standard	6	91.32%	2.97%	5.70%	n/a
	12	88.20%	6.06%	5.73%	n/a
	18	83.27%	6.21%	10.51%	n/a
	24	78.90%	9.46%	11.63%	n/a

The results in Table 4 show that MULTIPLEx compared to the standard CoAP solution increases the energy spent by the CPU. However, the MULTIPLEx CPU consumption is cost-effective, since the saved energy on Transmission and Reception is greater than the energy consumed to execute MULTPLEx. Another advantage of MULTIPLEx is that it preserves the content of the payloads, which means that the data accuracy is not reduced.

5 CONCLUSIONS AND FUTURE WORKS

This paper presents the Messaging mULTIple Payloads LayEr (MULTIPLEx) solution, which is able to assemble messages with multiple payloads. MULTIPLEx exploits the fact that most of the periodic many-to-one traffic in IoT applications has small size payloads. The experimental evaluation conducted in real Contiki devices shows that the energy gain of MULTIPLEx is up to 14%, without reducing the data accuracy, since the content of all payloads is preserved.

In future works, a new criterion will be implemented in MULTIPLEx in order to improve the decision of assembling messages with multiple payloads. The new criterion will consider delay and the number of children nodes. Besides, MULTIPLEx will be evaluated in a larger testbed.

REFERENCES

[1] Andrew Banks and Rahul Gupta. 2015. Message Queuing Telemetry Transport. (2015). http://docs.oasis-open.org/mqtt/mqtt/v3.1.1/mqtt-v3.1.1.html

[2] Luca Becchetti, Alberto Marchetti-Spaccamela, Andrea Vitaletti, Peter Korteweg, Martin Skutella, and Leen Stougie. 2009. Latency-constrained aggregation in sensor networks. *ACM Transactions on Algorithms (TALG)* 6, 1 (2009), 13.

[3] Tsitsipis D., Dima S. M., Kritikakou A., Panagiotou C., and Koubias S. 2011. Data merge: A data aggregation technique for wireless sensor networks. In *ETFA2011*. IEEE, 1–4. https://doi.org/10.1109/ETFA.2011.6059175

[4] Giovanni Di Stasi, Jonas Karlsson, Stefano Avallone, Roberto Canonico, Andreas Kassler, and Anna Brunstrom. 2014. Combining multi-path forwarding and packet aggregation for improved network performance in wireless mesh networks. *Computer Networks* 64 (2014), 26–37.

[5] H. Harb, A. Makhoul, S. Tawbi, and R. Couturier. 2017. Comparison of Different Data Aggregation Techniques in Distributed Sensor Networks. *IEEE Access* 5 (nov 2017), 4250–4263. https://doi.org/10.1109/ACCESS.2017.2681207

[6] Philipp Hurni, Benjamin Nyffenegger, Torsten Braun, and Anton Hergenroeder. 2011. On the Accuracy of Software-Based Energy Estimation Techniques. In *Wireless Sensor Networks: 8th European Conference, EWSN 2011, Bonn, Germany, February 23-25, 2011. Proceedings*, Pedro Jose Marron and Kamin Whitehouse (Eds.). Springer Berlin Heidelberg, Berlin, Heidelberg, 49–64. https://doi.org/10.1007/978-3-642-19186-24

[7] Isam Ishaq, Jeroen Hoebeke, Ingrid Moerman, and Piet Demeester. 2016. Observing CoAP groups efficiently. *Ad Hoc Networks* 37 (2016), 368–388.

[8] Zi Li, Wensheng Zhang, Daji Qiao, and Yang Peng. 2017. Lifetime balanced data aggregation for the internet of things. *Computers & Electrical Engineering* 58 (2017), 244–264.

[9] A. Riker, E. Cerqueira, M. Curado, and E. Monteiro. 2016. A Two-Tier Adaptive Data Aggregation Approach for M2M Group-Communication. *IEEE Sensors Journal* 16, 3 (Feb 2016), 823–835. https://doi.org/10.1109/JSEN.2015.2487445

[10] Z. Shelby, K. Hartke, and C. Bormann. 2014. The Constrained Application Protocol (CoAP). RFC 7252. (jun 2014). http://www.ietf.org/rfc/rfc7252.txt