TU Clausthal

A SURVEY ON CONSTRUCTIVE INTERFERENCE PROTOCOLS IN WIRELESS SENSOR NETWORKS

ITIS RESEARCH PROJECT REPORT

presented by

LUIS ALBERTO MEZA RUIZ

Embedded Systems group
Department of Informatics
Technische Universität Clausthal

ES-P004

Luis Alberto Meza Ruiz: *A survey on constructive interference protocols in wireless sensor networks*

STUDENT ID
450801

ASSESSORS
First assessor: Dr.-Ing. Andreas Reinhardt
Second assessor: Prof. Dr. Sven Hartmann

SUBMISSION DATE
31 December 2016

STATUTORY DECLARATION

Ich erkläre hiermit, dass ich die vorliegende Arbeit selbständig verfasst und keine anderen als die angegebenen Quellen und Hilfsmittel benutzt habe. Alle Stellen der Arbeit, die wörtlich oder sinngemäß aus anderen Quellen übernommen wurden, wurden als solche kenntlich gemacht. Die Arbeit wurde in gleicher oder ähnlicher Form noch keiner anderen Prüfungsstelle vorgelegt.

Ich erkläre mich zudem mit der öffentlichen Bereitstellung meines ITIS Research Project Report in der Instituts- und/oder Universitätsbibliothek einverstanden.

Clausthal-Zellerfeld, 31 December 2016

Luis Alberto Meza Ruiz

ABSTRACT

Nowadays, Wireless Sensor Networks (WSN) have multiple applications in a wide variety of scenarios: health, industrial, measure environmental conditions, etc. These networks are composed of multiple small sensors (or sometimes called nodes).

It is common to deploy such nodes in remote locations or in areas hard to reach, therefore these networks are designed to work with batteries as power supply.

A common duty in WSN is to update code or commands to reconfigure, spread data into the nodes or adjust parameters for the sensors after these have been deployed. We can summarize these tasks as data dissemination, an important topic in WSN because a manual reprogramming is many times not feasible or possible.

For these reasons, energy consumption is considered a critical issue in WSN and there have been some strategies that try to minimize this topic; one of those is the technique known as Constructive Interference (CI), which in practice is about multiple nodes transmitting the same packet in parallel. There have been some research work trying to implement CI with a bunch of protocols or architectures, all of them with different results and advantages or disadvantages regarding to some defined metrics.

One of the first implementations is "Glossy"[6], which in fact can be considered as the state-of-the-art architecture for CI.

Based on Glossy, there have been a couple of implementations (or improvements) of this technique. However, a reference of such implementations is missing and would be helpful in order to choose the best option if we want to get benefit from this effect in a WSN.

CONTENTS

ACRONYMS

IEEE	Institute of Electrical and Electronics Engineers
RF	Radio Frequency
MAC	Medium Access Control
WSN	Wireless Sensor Networks
CI	Constructive Interference
CE	Capture Effect
PHY	Physical Layer
TDMA	Time Division Multiple Access
GPIO	General Purpose Input Output
SFD	Start of Frame Delimiter
MCU	Microcontroller Unit
ACK	Acknowledgement
EACIF	Energy Adaptive CI-based Flooding protocol
LWB	Low-power wireless bus
PIP	Packets in Pipe

1 INTRODUCTION

The Institute of Electrical and Electronics Engineers (IEEE) 802.15.4 standard [10] specifies the Radio Frequency (RF), Physical Layer (PHY) and Medium Access Control (MAC) layers for WSN. Nowadays, WSN have multiple applications in a wide variety of scenarios: health, industrial, measure environmental conditions, etc. These networks are composed of multiple small sensors (or sometimes called nodes). Usually, WSN are deployed and unattended for a long time, because of their location. But sometimes when the nodes are already deployed, it is necessary to reconfigure them, forward data into the network, create a data collection tree, synchronize the nodes with the sink, and so forth. All these common duties in WSN depend on the service of data dissemination, which propagates a packet through the whole network. A dissemination protocol for WSN needs to overcome several challenges, such as high packet reliability and a short delay.

We have identified many dissemination protocols in the literature [2, 6, 9, 12, 13, 20, 22]. These protocols can be separated in two categories, contention resolution approaches and constructive interference based protocols. Contention resolution approaches employ a MAC protocol like CSMA/CA or Time Division Multiple Access (TDMA) to guarantee access to the shared wireless channel, and typically their dissemination times are in the order of minutes for disseminating a full image in the network. In order to eliminate the need for contention resolution, a new method of disseminating data is proposed which uses constructive interference, i.e. allowing multiple senders to transmit an identical packet at the same time, and still guarantees receivers to decode the packet correctly.

Before we can analyse the protocols and the constructive interference effect, it is necessary to understand what happens in a concurrent transmission. During a concurrent wireless transmission, some effects can ocurr depending on the transmission timing and signal strength: Constructive Interference (CI) and Capture Effect (CE), respectively. By concurrent packets, we mean that these are overlapping, i.e. the last packet starts before the first packet ends. Actually, whether a packet is received or not, it is the consequence of the combined results of the following three effects:

1. Capture effect. This is basically the ability of certain radios to correctly receive a strong signal from one transmitter despite significant interference from other transmitters.

2. Constructive interference. If a packet has the same content as the synchronization packet and the time displacement between these two at the receiver is less than or equal to the half of chip duration *Tc*, then they add up constructively.

3. SINR/SER model. The symbol error is determined by the SINR. The packets that form CI with the synchronization packet, add to the signal by a factor depending on the temporal displacement while those that do not, add to the interference, which reduces symbol error rate.

These effects and their consequences in concurrent transmissions are studied and analyzed in [25].

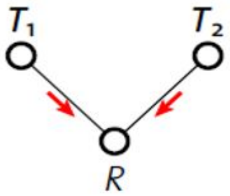

Figure 1.1: Basic constructive interference example.

Constructive Interference is a physical layer phenomenon and was first experimentally discovered by Dutta et al. [5] CI is basically a transmission technique for multiple nodes transmitting the same packet simultaneously through the network. This allows a fast network flooding in order to reduce the scheduling overhead of MAC protocols, and to achieve accurate time synchronization.

A basic example of how CI works is shown in Figure 1.1. Nodes *T1* and *T2* transmit each one a packet with the same data to *R*. Then, *R* can decode the packets as one packet if the maximum temporal displacement (Δ) between them is no more than one chip period *Tc*=0.5µs. This period corresponds to half of the DSSS chip duration in the IEEE 802.15.4 standard [25]. Since CI requires that multiple senders simultaneously transmit the same packet, this is useful and consistent with the characteristics of a network flooding. However, to get CI a tight synchronization among concurrent transmissions is essential. In a real-world implementation this is hard to obtain. But, even when this tight implementation is not met, the capture effect probably will occur. Capture effect is the ability to receive a signal despite interference from other concurrent transmissions. So, when the time synchrony condition of CI cannot be met, it is still very probable that capture effect happens since it has not a time synchrony requirement. In practice (talking about the Sky mote), the CE is caused by the continuous Start of Frame Delimiter (SFD) search implemented in the CC2420 radio [11]. This effect is studied in more detail in [24].

The idea behind this project is to test the protocols in a realistic environment, thus, is desirable to use real motes or testbeds. Due to its features, we will use the FlockLab testbed. Among the available services provided by FlockLab, we will specially use the General Purpose Input Output (GPIO) tracing and the power profiling features. We will talk in more detail about this testbed in Chapter 4. [1]

This work is relevant to those who would like to use or implement these protocols in a WSN application.

1.1 ORGANIZATION OF THE WORK

The organization of this work is structured as follows:

Chapter 2 looks over some of the related research done in the field. First we discuss some research done about concurrent transmissions focusing on constructive interference, and then we briefly review some related projects which employ this attribute.

We talk about the protocols to be implemented in Chapter 3, describing them more in detail and mentioning some of their features, advantages and possible drawbacks.

Chapter 4 details the implementation of the protocols and the steps needed, as well as the scenario where they are being tested. In the same section we also explain the metodology as well as the services provided by FlockLab.

The results are presented in Chapter 5 and they indicate/confirm . . .

General conclusions and findings about the project will be discussed in Chapter 6.

[1] To do the same with real motes, we need logic analyzers and these have limited number of channels.

2 RELATED WORK

Most of WSN related research is focused on energy-efficient network operations as energy is one of the most valuated resources. During the last years, many work regarding how constructive interference can help with this issue has been conducted. We will present in this chapter some of this work.

2.1 CONCURRENT TRANSMISSIONS

The first work identified in the literature of concurrent transmissions was done by Dutta et al [5]. In this work, they discovered that when two Acknowledgement (ACK) frames collide at a initiator node, with fixed conditions, the collision will be non-destructive. They indicate that concurrent ACK frames do not cancel each other at the physical layer if they are identical, or if one ACK frame has a higher power than the remaining ACK frames.

Going one step beyond, Yuan et al. [25] propose an accurate model-algorithm that helps to understand the reception of concurrent transmissions and also provides a complete view on the nature of concurrent transmissions. This model is also validated by experimenting with a serie of tests. In the same work, they also give theoretical explanations on the nature of constructive interference and the capture effect, and validate these experimentally.

The crucial condition to obtain constructive interference is about the time displacement between concurrent packets. Some studies [6] have shown that if the interval among concurrent transmissions of the same packet is less than 0.5 µs (half of the DSSS chip duration in IEEE 802.15.4), there is a high probability that these packets will result in constructive interference.

In their work, Wang et al. [22] showed that the reliability of constructive interference decreases significantly as the number of concurrent transmitters increases. With a mathematical analysis, they disclosed that if the number of concurrent transmitters grows as well as the number of hops in the network, then the maximum time displacement among concurrent transmissions to a common receiver is probable to exceed the threshold period *Tc*; thus increasing the risk of collisions.

2.2 PROTOCOLS AND ARCHITECTURES

Glossy [6] can be considered as the seminal work for CI, because it is the first protocol that puts into practice this effect. Glossy exploits concurrent transmissions for fast network flooding. Thus, it considers interference an advantage rather than a problem. It also allows the receivers to decode the packets even if there is no CE. Furthermore, it implicitly synchronizes the nodes with the sink as the flooding packet is propagated through the network. We will take a closer look at Glossy in Chapter 3.

Based on Glossy, Yuan et al. proposed Sparkle [26], a periodic multi-loop control network where each control loop is mapped into one or more communication flows. The novelty of Sparkle is that the user controls each end-to-end flow based on runtime feedbacks, with the goal that the QoS metrics of the flow satisfy given requirements or are optimized.

To tackle the problem of the scalability in constructive interference [22], Doddavenkatappa et al. proposed Splash [2], a possible solution based on collection tree protocols, which create parallel pipelines effectively by using constructive interference. Splash is built upon 2 works: Glossy [6] and Packets in Pipe (PIP) [17]. Glossy exploits CI, while PIP employs channel diversity and avoids self interference for a single flow over multiple hops.

Splash forms a parallel pipeline by scheduling channel switch between nodes on adjacent layers.

Splash is used for fast data dissemination. Instead of a topology-unaware flooding, it uses a collection protocol to build a tree structure for flooding. To get a high reliability in the data dissemination, Splash uses some techniques such as XOR coding, local recovery, channel cycling among others.

Cheng et. al proposed with their work [1] an Energy Adaptive CI-based Flooding protocol (EACIF). EACIF improves the CI-based flooding through energy scheduling and topology control, via distributed management of node residual energy. Compared with Glossy, EACIF can reduce redundant transmissions and thus achieves lower energy consumption, lower latency and the same reliability. We also can find in their work a comprehensive description of a CI-based flooding.

Also bassed on Glossy but unlike the previous works, which are more oriented to flood or disseminate data, Suzuki et al. designed Choco [21], an energy-efficient collection protocol for WSN.

These are some protocols which employ CI. In the next chapter we will review the protocols chosen for our implementation.

3 PROTOCOLS OVERVIEW

In this section, we describe the protocols chosen for this project.

The reason for choosing these protocols is that their code is publicly available. Thus, it is possible to modify the parameters of each protocol and evaluate its behavior.

3.1 GLOSSY

Glossy [6] was developed

Glossy achieves the synchronization condition ($\Delta \leqslant 0.5\mu s.$) of CI. An accurate synchronization guarantees high reliability of CI-based flooding.

In Glossy, the sink node sends a periodic sync packet. Afterwards, every other node after receiving the sync packet, becomes synchronized with the sink node. Thus, Glossy can quickly resynchronize the whole network faster than traditional clock synchronization protocols. This synchronization packet contains a counter initiated to 1 by the sink. When the nodes receive the packet, they increase the counter before forwarding it. So, based on the counter value, a node can determine its distance from the sink.

One important feature of Glossy, is that it does not require knowledge of the network topology, because all nodes relay the packet as soon as they receive it, independent of their current position and neighboring nodes.

There are also a pair of disadvantages identified. One of them, is that Glossy produces many redundant packets during the flooding process, because it makes all nodes to get involved in the data forwarding. All these unnecesary data transmissions waste energy and possibly lead to a network life reduction. The most straightforward solution would be to reduce these unnecesary transmissions.

Glossy also has a scalability problem. If the hops of independent paths increase, the packet reception rate decreases, due to independent paths increase cumulative synchronization errors. This drawback was discovered by Wang et al.[22]

3.2 LOW-POWER WIRELESS BUS (LWB)

The Low-power wireless bus (LWB) has been proposed by Ferrari et al [7]. LWB is built upon Glossy and uses a synchronous and slot-based communication scheme. LWB is a protocol that supports one-to-many, many-to-one and many-to-many communications. The main features of the protocol is that communication schedule is flooded to all nodes by a centralized sink and the communication requests are flooded to the hosts through contention. Also, it does not require information about the network topology. LWB uses a slotted communication scheme, where a slot length is the amount of time required to flood the whole network. Every node requests a data slot from the sink (also called scheduler) and it assigns a slot for every node. Once the slot assignment is done, the data transmission is accomplished using a Glossy-based flooding, where every node initiates a flood in its assigned slot. LWB offers high data yield and uses less power.

Since the original LWB implementation was done for the CC430 radio, we will use the implementation provided in the work done by Chayan Sarkar [18]. Thus, we will use the same hardware platform for all the protocols.

3.3 CHAOS

Many application approaches typically split data interactions into sequential collection, processing and dissemination phases. This leads into an inefficiently use of the resources. Chaos [15] tries to tackle this concern implementing an all-to-all fashion data sharing.

Chaos is built upon Glossy and in this way, it takes advantage of synchronous transmissions. Chaos is scalable to networks with hundreds of nodes, thus outperforming protocols in the field such as LWB [7] and CTP (a data collection protocol). Reliable dissemination with Chaos works by using the data to be distributed as the payload, and the flags as an implicit confirmation whether all nodes received it. Based on the flags, it is possible for the nodes to know which nodes missed an update and so initiate a loss recovery without awaiting an explicit request. The Chaos implementation has an 8-byte header.

The IEEE 802.15.4 and Chaos headers together are 8 bytes and the flags occupy dN/8e bytes in an N-node network. But, due to the limited packet size in IEEE 802.15.4, multiple Chaos rounds might be necessary to share data among the nodes, depending on the amount of data and number of nodes in the network. For the same reason, Chaos needs some form of data compression if we want to implement it in networks with more than 1000 nodes.

4 METODOLOGY

As specified in the Introduction, the main goal of this project is to provide an overview of the most known protocols based on constructive interference, as well as test them with some defined guidelines. In this section, we will describe the implementation of the protocols.

To accomplish this, some performance metrics will have to be measured and compared among the protocols. Specifically will be studied energy consumption, packet reliability and end-to-end latency, which are key factors in a wireless sensor network. In order to fulfill these expectations, it is desired to evaluate the protocols in a real-world environment and for this reason, we will be testing the protocols using a sensor network testbed. Altough simulators help in the development of new protocols or applications, testbeds play a key role in developing real-world wireless embedded systems by providing the facilities to debug and evaluate protocols and applications in a controlled and realistic distributed environment. The FlockLab testbed [16] was chosen due to its features and because it allows us to test the protocols in a less intrusive way.

Usually, the practice of testbed-assisted development is based on LED status and `printf` debugging. However, it is well known that `printf` statements are intrusive and alter the timing behavior, therefore they are unsuitable for analyzing timing sensitive code such as MAC protocols or synchronization protocols, like in this case. These means are unsuitable for detailed investigation of interactions among multiple devices or real-time issues. Different from `printf` statements, setting digital GPIO pins on a node introduces a known delay of just a few cycles, which makes GPIO tracing a powerful technique for debugging time sensitive code.

4.1 FLOCKLAB OVERVIEW

FlockLab is a WSN testbed, developed and run by the Computer Engineering and Networks laboratory at the Swiss Federal Institute of Technology Zurich in Switzerland (ETH Zürich) [2].

It consists of several distributed target-observer pairs and some servers. *Observers* are basically platforms that host the devices un-

[2] https://www.flocklab.ethz.ch/

der test, i.e. the nodes. The distribution of the nodes in FlockLab is shown in Figure 4.1

In order to schedule a test in FlockLab, we need to upload the test configuration, basically a XML file that has specified the services we want to run in the node and even one or more compiled binaries. It is possible to get the results via e-mail, or if we get large and heavy results (like the power tracing ones), through a WebDAV client. The current FlockLab deployment consists of 30 observers organized in a mixed indoor/outdoor topology, each of them hosting 4 different kind of motes.

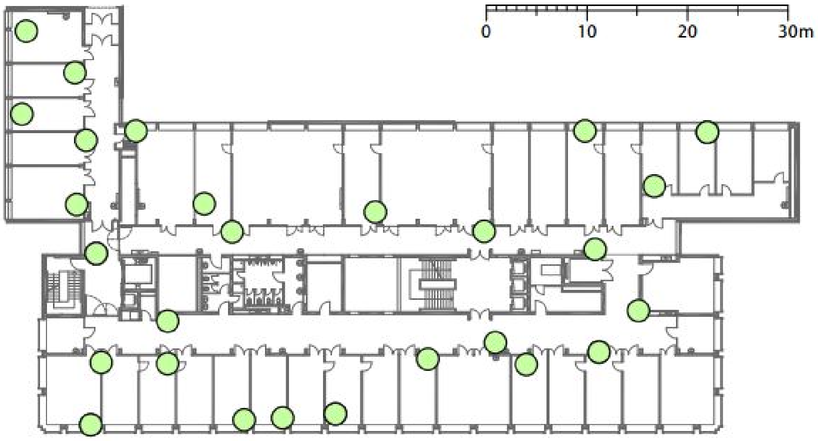

Figure 4.1: Position of the indoor nodes in FlockLab.

FlockLab has distributed target-observer pairs that are capable of capturing event and power traces of all targets locally, simultaneously, synchronously, and at high rates without sacrificing on timing accuracy or incurring data rate limitations of traditional testbeds [8, 23]. In this way, FlockLab combines the capability of a logic analyzer, power analyzer, serial data logger and programmable power supply with network synchronization and deep local storage adjacent to each target-node. FlockLab allows multiple services to run simultaneously and synchronously against all nodes under test in addition to the traditional port service. This is achieved by tracing GPIO pins to record logical events occurring on a node [16].

FlockLab's accurate timing information in the low microsecond range enables logical events to be correlated with power samples. FlockLab accurately timestamps data acquired during a test across all nodes. (cambiar)

4.2 metrics

1. Energy Consumption.

   Normally it is possible to estimate the energy consumption by software means. The feature Energest [4], available in Contiki, provides accurate energy estimations in an intrusive way, because generally incites the use of the radio duty cycle. But with FlockLab, this can be done in a non-intrusive fashion. The method used in FlockLab is slightly more accurate than Energest or even a power analyzer, as described in [16]. This feature is called *power profiling*, and it allows us to measure the energy consumption of a node in a non-intrusive fashion, instead of the common way normally made by software means.

   This procedure is as follows: When a SFD interrupt indicates the start of a packet reception, a receiver sets a GPIO pin and this operation will increase the current draw, which is timestamped and recorded. Afterwards, when the next SFD interrupt signals the end of the reception, i.e the packet reception has finished, the receiver clears the GPIO pin and the current decreases accordingly. This event is also timestamped and recorded.

2. Packet Reliability.

   Like every application code, each protocol makes use of several functions and variables. One of them is regarding the times a node has received a packet. Then, the reliability of the protocols can be tested checking this variable of the code . . .

3. Latency.

   We define data latency as the time difference between the end of packet transmission and the end of packet reception, i.e. the difference between the time the SFD pins of both nodes falling. The GPIO tracing service provided by FlockLab can be used to measure the latency: when the sink generates a packet, a GPIO pin is toggled and a source does the same when it receives a packet. In this non-intrusive way, we can take the interval between both events in order to obtain the measurement of the latency of received packets. We just have to insert the GPIO tracing statements in the application code and configure the appropriate service in the FlockLab's XML file. The latency of each protocol can be measured checking the time between the transmission at a defined initiator node and the successful reception at the receiver. This is possible by recording the timestamp of the SFD pin on the CC2420 radio.

4.3 EXPERIMENT SETUP

In this section we describe the experiment setup for studying constructive interference protocols. A flooding based on CI is described as follows: In the first round, the initiator (node 1) broadcasts a packet to its one-hop neighbors (nodes 2 and 4). Then, the one-hop neighbors will forward the packets to the initiator and the two-hop neighbors (nodes 5 and 3) in the second round. Every other round, nodes on each layer forward the flooding packet once. Each node stops forwarding the packet if its transmission count reaches the transmission threshold.

The experiment topology is shown in Figure 4.2. Node 1 is the initiator in all cases.

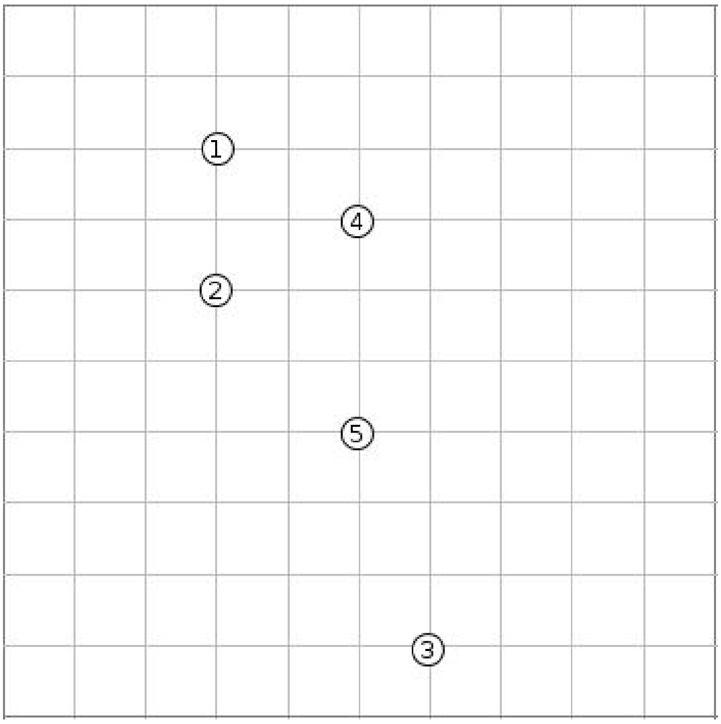

Figure 4.2: Topology used for the experiments.

We implemented the protocols in the Contiki embedded operating system [3] and the experiments are based on the widely-used Tmote Sky [19], a mote that features the Texas Instruments CC2420, a radio compliant with the IEEE 802.15.4 STD. [11]. One of the main contributions of Contiki is the introduction of protothreads. Protothreads introduce a very low memory overhead because they share the same stack and the thread switching only needs a little rewind of the stack positions. Programs in Contiki can be disseminated and executed dynamically [14].

The low power operation of the Tmote Sky module is due to the ultra low power Texas Instruments MSP430 F1611 microcontroller featuring 10kB of RAM, 48kB of flash, and 128B of information storage. We will evaluate the protocols with the maximum transmit power output power of a Sky mote, i.e 0 dBm, as specified in the mote datasheet [19]. Moreover, the experiments will be carried out on the Channel 26 (2,480GHz), a channel that does not overlap with the common WiFi channels. These features provide the needed protocol and hardware support.

The code of all the protocols is publicly available.

According to the CC2420 datasheet [11], the SFD pin rises once the SFD byte has been completely received and falls only after the last byte of MPDU has been received. In transmit mode, the SFD pin rises when the SFD byte has been completely transmitted and falls when the complete MPDU has been transmitted. In general, at the beggining and end of a packet transmission/reception, the SFD pin will rise or fall accordingly. This means that we can take the SFD pin rise and fall as the start and end of a packet transmission and reception.

In the Tmote, the SFD pin of the CC2420 radio chip is directly connected to the timer B of the MSP430F1611 Microcontroller Unit (MCU). Then, it is possible to detect each rise or fall of this pin and actuate accordingly using the pins P6.0 and P6.1, pins attached to the observers in FlockLab.

Experiments have been carried out with 5 nodes. We define the packet parameters as follows: we will have 3 different payload lengths, 10 bytes, 50 bytes and 100 bytes.

In the next chapter, we present the results obtained.

5 EXPERIMENT RESULTS

In this section, the results obtained will be presented.

For each protocol, we show a graphic of the current consumption so we can observe the behavior of the protocols[3], as well as a table indicating the minimum and maximum current measured during the execution of the protocols. The reliability was calculated with a total of 200 packets in each case.

The implementation of the protocols was done using the Contiki OS[3], an operating system for WSN with the ability to load and unload individual applications or services once the nodes are deployed, i.e at run-time.

We evaluated the protocols on a real world environment and to this end, we used the FlockLab testbed [16].

5.1 GLOSSY

1. Energy Consumption.

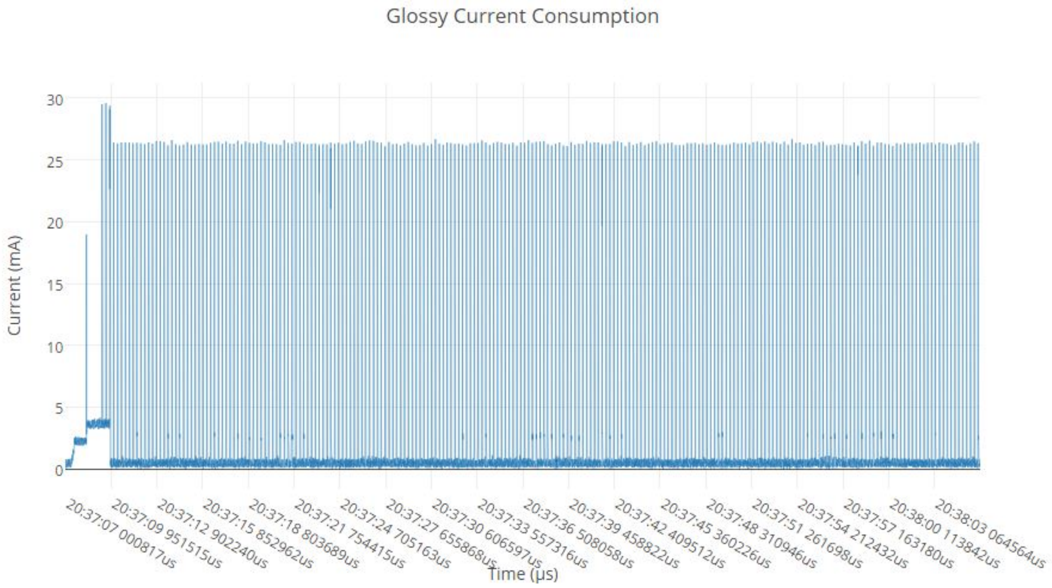

Figure 5.1: Current consumption during a Glossy round.

[3] Although in the graphs we can observe peaks of up to 70 mA, we do not consider these in the results, as they are part of the initialization process of the mote and not of the protocol itself.

Payload	10 bytes		50 bytes		100 bytes	
Current (mA)	Min.	Max.	Min.	Max.	Min.	Max.
Node 1	0.012785	29.60109	0.02512	29.805985	0.016721	30.796218
Node 2	0.070941	29.874704	0.054635	29.876711	0.04544	29.797209
Node 3	0.02845	29.926413	0.023337	29.814899	0.028333	30.050486
Node 4	0.038564	30.177732	0.038564	30.017092	0.045913	30.097111
Node 5	0.043407	30.198857	0.023213	30.132438	0.045547	30.12061

Table 5.1: Glossy current consumption per node.

2. Packet Reliability.

   In Table 5.2, we show the reliability results on average, for all nodes:

Payload	10 bytes	50 bytes	100 bytes
	Reliability	**Reliability**	**Reliability**
Node 1	100%	100%	83.189%
Node 2	100%	100%	84.888%
Node 3	8.69%	100%	82.327%
Node 4	100%	100%	83.189%
Node 5	100%	100%	81.896%

Table 5.2: Glossy reliability.

3. Latency.

Payload	10 bytes	50 bytes	100 bytes
	Latency (mS)	Latency (mS)	Latency (mS)
Node 1	1.586	14.999	16.752
Node 2	0.714	14.999	16.709
Node 3	1.998	14.973	16.709
Node 4	0.718	14.987	16.733
Node 5	0.718	15.063	16.757

Table 5.3: Average latency during a Glossy round.

5.2 low-power wireless bus (lwb)

1. Energy Consumption.

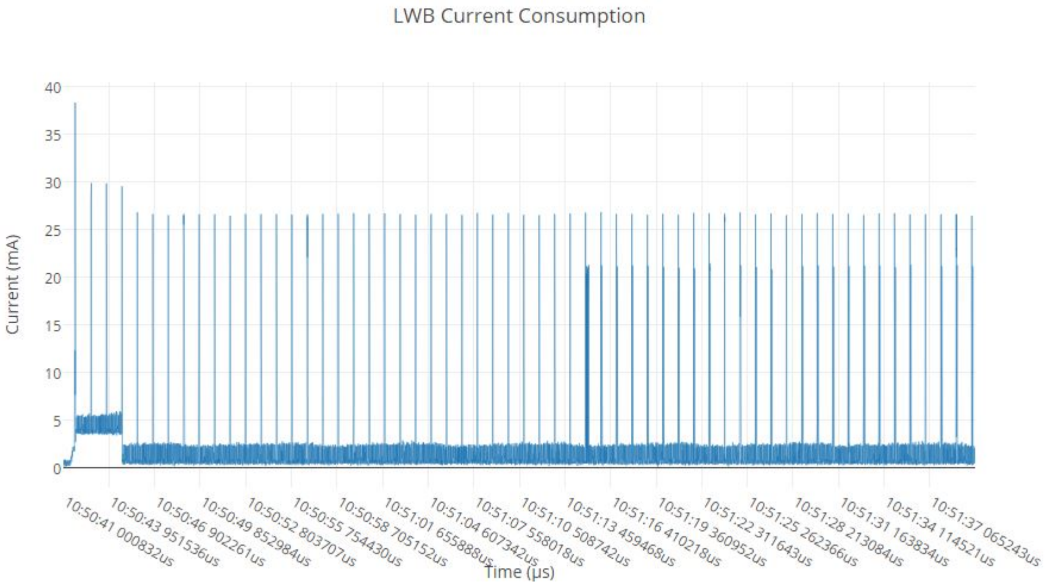

Figure 5.2: Current consumption during a LWB round.

Payload	10 bytes		50 bytes		100 bytes	
Current (mA)	Min.	Max.	Min.	Max.	Min.	Max.
Node 1	0.002282	30.128145	0.016721	29.989893	0.016721	29.824633
Node 2	0.054635	29.83997	0.061395	29.758617	0.054635	29.630276
Node 3	0.018263	29.973869	0.024931	29.953398	0.02845	29.793106
Node 4	0.090452	30.171356	0.054778	30.054477	0.038564	30.013962
Node 5	0.045547	30.262687	0.052356	30.027675	0.055566	30.08351

Table 5.4: LWB current consumption per node.

2. Packet Reliability.

   In Table 5.5, we show the reliability results on average, for all nodes:

Payload	10 bytes	50 bytes	100 bytes
	Reliability	**Reliability**	**Reliability**
Node 1	-%	-%	0%
Node 2	-%	-%	0%
Node 3	-%	-%	0%
Node 4	-%	-%	0%
Node 5	-%	-%	0%

Table 5.5: LWB reliability.

3. Latency.

Payload	10 bytes	50 bytes	100 bytes
	Latency (mS)	Latency (mS)	Latency (mS)
Node 1	-	-	-
Node 2	-	-	-
Node 3	-	-	-
Node 4	-	-	-
Node 5	-	-	-

Table 5.6: Average latency during a LWB round.

5.3 CHAOS

1. Energy Consumption.

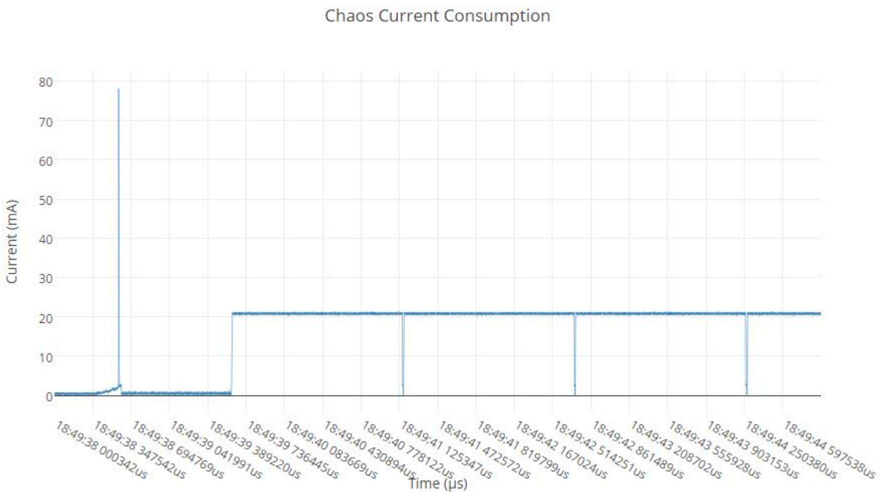

Figure 5.3: Current consumption during a Chaos round.

Payload	10 bytes		50 bytes		100 bytes	
Current (mA)	Min.	Max.	Min.	Max.	Min.	Max.
Node 1	0.121243	21.81497	0.124517	21.858014	0.016721	21.771049
Node 2	0.054635	22.83065	0.054635	21.146864	0.054635	20.992126
Node 3	0.02845	21.292307	0.02845	21.327631	0.032241	21.22403
Node 4	0.038564	21.279862	0.038564	21.308849	0.038564	21.29386
Node 5	0.045547	22.270579	0.045547	21.473053	0.045547	21.393075

Table 5.7: Chaos current consumption per node.

2. Packet Reliability.

   In Table 5.8, we show the reliability results on average, for all nodes:

Payload	10 bytes	50 bytes	100 bytes
	Reliability	**Reliability**	**Reliability**
Node 1	100%	100%	100%
Node 2	100%	100%	100%
Node 3	100%	100%	100%
Node 4	100%	100%	100%
Node 5	100%	100%	100%

Table 5.8: Chaos reliability.

3. Latency.

Payload	10 bytes	50 bytes	100 bytes
	Latency (mS)	Latency (mS)	Latency (mS)
Node 1	90,452	12,555	28,465
Node 2	101,635	12,695	42,717
Node 3	101,635	12,663	14,252
Node 4	90,445	12,632	14,253
Node 5	90,445	12,614	297,355

Table 5.9: Average latency during a Chaos round.

5.4 GENERAL FINDINGS

Beyond the particularities that the protocols exhibit, we can point to some general findings.

LWB and Glossy have more energy consumption, reaching peaks up to 28-30 mA. Whereas LWB and Glossy have a consumption over 25 mA during a transmission, Chaos has a consumption of roughly 21-22 mA per node, as we can observe in Table 5.7. Then, Chaos is the protocol that waste a lower amount of energy than the others but its rounds are larger. It is also visible that the longer the packet size is, the longer the latency in Chaos, although this metric seems to be some unstable, as is visible in Table 5.9.

In Figure 5.3 we can observe the behavior of the current consumption during a Chaos round; this goes from 0 mA to around 20 mA, which corresponds to the typical operating conditions indicated in the Tmote datasheet [19]. It is also easily visible the duration of a Chaos round, this was measured from the rising of the edge to the falling edge, averaging 1,54 seconds in all cases.

Furthermore, if we take a closer look to both the GPIO tracing and the power profiling files, we can estimate that a logical 1 is measured from around 3 mA.

Moreover, the advantage of Chaos is more clear in its reliability. Whereas Glossy and LWB are based just in CI, and even tough Chaos is also built upon Glossy, it also takes advantage of the capture effect. In practice this means that even if the timing requirement of CI is not satisfied, it is possible that a packet is being received due to the capture effect and consequently is possible to have a 100 % reliability even with large packet sizes. For these reasons, Chaos has the best reliability, with 100 % in all cases. In the case of Glossy, we can observe that with larger packets, the reliability drops by 18%. Node 3 has the worst reliability because it is located far away from the other nodes (around 10m from Node 5). Thus, we confirm that the realiability decreases if we increase the packet size. This outcome is even more visible with LWB.

In the case of LWB, even though the protocol rounds are working normally, as we can observe in the current consumption, something happens: with a 10 and 50-bytes payload, no packets are neither being transmitted nor received (therefore we cannot present reliable results), and with a 100-bytes payload, all the packets are being dropped. Then, we have to modify the time slots during a LWB round to support larger payloads.

6 CONCLUSIONS

In this project, a survey work was done on some of the existing protocols exploiting constructive interference for wireless sensor networks and their performances were tested, using the same hardware platform and the same set of metrics: latency, reliability and power consumption, crucial metrics in every WSN.

The key requirement for CI is that nodes have to transmit the same packet at the same time, or at least with a time displacement lower than 0.5 µs. Beyond this limit, the effect is not guaranteed and we can have packet loss.

We also provide a brief description of concurrent transmissions as well as various protocols and implementations that use CI as part of their architechture.

With a serie of tests and using different payloads (10, 50 and 100-bytes), we have validated the performance of each protocol.

Based on the results, we can conclude that Chaos proved to have the best performance in all cases, even with large packets. This can be explained because besides CI, the capture effect also takes place, thus helping ensure the correct reception of the packets.

Moreover, it can also be concluded that CI is completely suitable for small and medium packets, i.e. around maximum 50-60 bytes. For larger packets, CI is not guaranteed due to its timing requirement, but it could be still possible to receive the packet thanks to the CE.

The results gained are valuable for simulation and application design in WSN.

BIBLIOGRAPHY

[1] Dapeng Cheng, Yanyan Mao, Yin Wang, and Xiangrong Wang. "Improving energy adaptivity of constructive interference-based flooding for WSN-AF." In: *International Journal of Distributed Sensor Networks* 2015 (2015), p. 6.

[2] Manjunath Doddavenkatappa, Mun Choon Chan, and Ben Leong. "Splash: Fast data dissemination with constructive interference in wireless sensor networks." In: *Presented as part of the 10th USENIX Symposium on Networked Systems Design and Implementation (NSDI 13)*. 2013, pp. 269–282.

[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, Fredrik Osterlind, Nicolas Tsiftes, and Zhitao He. "Software-based on-line energy estimation for sensor nodes." In: *Proceedings of the 4th workshop on Embedded networked sensors*. ACM. 2007, pp. 28–32.

[5] Prabal Dutta, Razvan Musaloiu-e, Ion Stoica, and Andreas Terzis. "Wireless ACK collisions not considered harmful." In: *Proceedings of the 7th ACM Workshop on Hot Topics in Networks (HotNets-VII)*. 2008, pp. 1–6.

[6] Federico Ferrari, Marco Zimmerling, Lothar Thiele, and Olga Saukh. "Efficient network flooding and time synchronization with glossy." In: *Information Processing in Sensor Networks (IPSN), 2011 10th International Conference on*. IEEE. 2011, pp. 73–84.

[7] Federico Ferrari, Marco Zimmerling, Luca Mottola, and Lothar Thiele. "Low-power wireless bus." In: *Proceedings of the 10th ACM Conference on Embedded Network Sensor Systems*. ACM. 2012, pp. 1–14.

[8] Vlado Handziski, Andreas Köpke, Andreas Willig, and Adam Wolisz. "Twist: a scalable and reconfigurable testbed for wireless indoor experiments with sensor networks." In: *Proceedings of the 2nd international workshop on Multi-hop ad hoc networks: from theory to reality*. ACM. 2006, pp. 63–70.

[9] Jonathan W Hui and David Culler. "The dynamic behavior of a data dissemination protocol for network programming at scale." In: *Proceedings of the 2nd international conference on Embedded networked sensor systems*. ACM. 2004, pp. 81–94.

[10] "IEEE Standard for Low-Rate Wireless Networks." In: *IEEE Std 802.15.4-2015 (Revision of IEEE Std 802.15.4-2011)* (2016), pp. 1–709. DOI: 10.1109/IEEESTD.2016.7460875.

[11] Texas Instruments. "CC2420: 2.4 GHz IEEE 802.15. 4/ZigBee-ready RF Transceiver." In: *Available at Available at http://www. ti. com/lit/gpn/cc2420* 53 (2006).

[12] Jaein Jeong, Sukun Kim, and Alan Broad. "Network reprogramming." In: *University of California at Berkeley, Berkeley, CA, USA* (2003).

[13] Sandeep S Kulkarni and Limin Wang. "MNP: Multihop network reprogramming service for sensor networks." In: *25th IEEE International Conference on Distributed Computing Systems (ICDCS'05)*. IEEE. 2005, pp. 7–16.

[14] Rafael Lajara, José Pelegrí-Sebastiá, and Juan J Perez Solano. "Power consumption analysis of operating systems for wireless sensor networks." In: *Sensors* 10.6 (2010), pp. 5809–5826.

[15] Olaf Landsiedel, Federico Ferrari, and Marco Zimmerling. "Chaos: Versatile and efficient all-to-all data sharing and in-network processing at scale." In: *Proceedings of the 11th ACM Conference on Embedded Networked Sensor Systems*. ACM. 2013, p. 1.

[16] Roman Lim, Federico Ferrari, Marco Zimmerling, Christoph Walser, Philipp Sommer, and Jan Beutel. "Flocklab: A testbed for distributed, synchronized tracing and profiling of wireless embedded systems." In: *Information Processing in Sensor Networks (IPSN), 2013 ACM/IEEE International Conference on*. IEEE. 2013, pp. 153–165.

[17] Bhaskaran Raman, Kameswari Chebrolu, Sagar Bijwe, and Vijay Gabale. "PIP: A connection-oriented, multi-hop, multi-channel TDMA-based MAC for high throughput bulk transfer." In: *Proceedings of the 8th ACM Conference on Embedded Networked Sensor Systems*. ACM. 2010, pp. 15–28.

[18] Chayan Sarkar. "LWB and FS-LWB implementation for Sky platform using Contiki." In: *arXiv preprint arXiv:1607.06622* (2016).

[19] Tmote Sky Data Sheet. *Moteiv, San Francisco, CA, 2006*. 2004.

[20] Thanos Stathopoulos, John Heidemann, and Deborah Estrin. *A remote code update mechanism for wireless sensor networks*. Tech. rep. DTIC Document, 2003.

[21] Makoto Suzuki, Yasuta Yamashita, and Hiroyuki Morikawa. "Low-power, end-to-end reliable collection using glossy for wireless sensor networks." In: *Vehicular Technology Conference (VTC Spring), 2013 IEEE 77th*. IEEE. 2013, pp. 1–5.

[22] Yin Wang, Yuan He, Xufei Mao, Yunhao Liu, and Xiang-yang Li. "Exploiting constructive interference for scalable flooding in wireless networks." In: *IEEE/ACM Transactions on Networking* 21.6 (2013), pp. 1880–1889.

[23] 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.

[24] Kamin Whitehouse, Alec Woo, Fred Jiang, Joseph Polastre, and David Culler. "Exploiting the capture effect for collision detection and recovery." In: *Proc. of IEEE EmNetS-II* 5 (2005), pp. 45–52.

[25] Dingwen Yuan and Matthias Hollick. "Let's talk together: Understanding concurrent transmission in wireless sensor networks." In: *Local Computer Networks (LCN), 2013 IEEE 38th Conference on*. IEEE. 2013, pp. 219–227.

[26] Dingwen Yuan, Michael Riecker, and Matthias Hollick. "Making 'glossy'networks sparkle: Exploiting concurrent transmissions for energy efficient, reliable, ultra-low latency communication in wireless control networks." In: *European Conference on Wireless Sensor Networks*. Springer. 2014, pp. 133–149.