LA6
Local Aggregation in the Internet of Things

Bruno Graça Coelho

Thesis to obtain the Master of Science Degree in
Communication Networks Engineering

Examination Committee
Chairperson: Prof. Paulo Jorge Pires Ferreira
Supervisor: Prof. Rui Manuel Rodrigues Rocha
Members of the Committee: Prof. Paulo Rogério Barreiros d’Almeida Pereira

May 2013
Acknowledgments

First and foremost, I would like to express my deep gratitude to my supervisor, Professor Rui Rocha, for introducing me to the world of wireless sensor networks. Working with him has been a great honour, I have learned a lot with him, he is one of a kind, his rigour contribute for making me a better person, a better professional.

I also extend my thanks to the whole GEMS team (Group of Embedded networked Systems and Heterogeneous Networks). Their helpful input contributed directly for the brainstorming of many of the ideas that were bounced around before being put into practice.

I am grateful to Fábio Constantino, the best reviewer I could wish for. His patience and hard work greatly increased the quality of this document. I would also like to thank to Eduardo Bettencourt for being a partner in this journey from its beginning, for your words and constant support, always willing to help and give the best suggestions.

Amongst my fellow colleagues and friends the support given by Duarte Mechas and João Rosa in the final steps of this work will stand as an example of friendship that I will never forget.

I am eternally grateful to my friends Hugo Mineiro and Tiago Silva that were a constant support, in this adventure in Instituto Superior Técnico, always there to help me in every circumstances.

I dedicate this work to my family, that sacrificed themselves to support my studies my dream of making a difference. I owe them everything, I owe them the person I became and the person I will be.

Thank You all.
Resumo

Vivemos num mundo em constante mudança, condicionado pela constante evolução tecnológica, que tornará real a Internet of Things, onde tudo estará ligado. Contudo, a definição do 6LoWPAN existente proposta pelo IETF, tem alguns problemas de eficiência, sendo quase inviável a sua utilização numa rede de sensores sem fios. As implementações actuais de 6LoWPANs herdam o paradigma de comunicação da rede cablada baseada nos endereços, Internet.

Este trabalho propõe uma comunicação inteligente baseada no conteúdo das mensagens (data-centric) para redes de sensores sem fios que supera os problemas do 6LoWPAN através da agregação de mensagens.

Este trabalho contribui directamente para a viabilidade das redes 6LoWPAN no contexto das redes de sensores sem fios, sendo por esse motivo um contributo directo a revolução que será a Internet Of Things.

Palavras-chave: agregação local de informação, 6LoWPAN, redes de sensores sem fios, Internet of Things, CTP
Abstract

We live in a changing world, extremely dynamic, resulting in a constant technological evolution which will lead us to an Internet of Things reality, where everything is connected. However, the current 6LoWPAN standard proposed by IETF has some efficiency problems which can make its application on wireless sensor networks not viable. Even though the current literature and implementations already contemplate the use of 6LoWPAN in this type of networks, it does so in a non data-centric way.

This work transforms the existent 6LoWPAN implementation enabling a data-centric solution designed for wireless sensor networks reality, that will overcome the current viability issues of 6LoWPAN in these networks through the integration of an in-network data processing aggregation mechanism.

In resume, LA6 Data Aggregation Mechanism increases dramatically the sustainability and network lifetime of 6LoWPAN based wireless sensor networks, contributing directly to the Internet of Things revolution.

Keywords: local data aggregation, 6LoWPAN, wireless sensor networks, Internet Of Things, CTP
# Contents

Acknowledgments ................................................................. v
Resumo .................................................................................. vii
Abstract ................................................................................ ix
List of Tables ........................................................................ xiii
List of Figures .......................................................................... xvi
List of Acronyms ..................................................................... xix

1 Introduction 
   1.1 Motivation and Objectives ........................................... 2
   1.2 Document Organization .............................................. 3

2 State Of The Art 
   2.1 Introduction .............................................................. 5
   2.2 Communication Technologies ....................................... 5
      2.2.1 IEEE 802.15.4 ...................................................... 5
      2.2.2 Bluetooth Low Energy .......................................... 6
      2.2.3 Summary .......................................................... 7
   2.3 In-Network Data Aggregation ........................................ 7
      2.3.1 Data Aggregation Mechanisms Overview .................. 7
      2.3.2 In-Network Data Aggregation Key Features .............. 9
      2.3.3 Summary .......................................................... 15
   2.4 6LoWPAN ................................................................. 15
      2.4.1 Addressing .......................................................... 16
      2.4.2 Stateless Address Autoconfiguration and 6LoWPAN Neighbour Discovery ........... 16
      2.4.3 Header Compression .............................................. 17
      2.4.4 Routing .............................................................. 17
      2.4.5 TinyOS and Blip .................................................... 19
      2.4.6 Discussion from an In-Network Data Processing Perspective ......................... 20

3 6LoWPAN In-Network Data Aggregation Mechanism Architecture 23
   3.1 Traditional 6LoWPAN Sensor Network Architecture ........ 23
   3.2 6LoWPAN In-Network Processing Architecture Design .......... 24
## List of Tables

2.1 Comparison of the main characteristics of the IEEE 802.15.4 and BLE. .................. 7  
2.2 Classes of Aggregates according to [38]. .......................................................... 12  
2.3 Data Aggregation Mechanisms comparison summary. .......................................... 15  
4.1 MICAz device main characteristics. ................................................................. 34  
6.1 CC2420 Radio transceiver consumption. ........................................................... 49
## List of Figures

2.1 Data Aggregation Taxonomy. .......................... 8
2.2 Routing implementation approaches: Link Layer Mesh-Under, 6LoWPAN Mesh-Under, and Router-Over respectively. .......................... 17
2.3 ROLL-WG 6LoWPAN typical architecture, and deployment scenario. .................. 18
2.4 Blip Protocol Stack. ................................ 20
3.1 Traditional 6LoWPAN sensor network architecture. .................. 24
3.2 BLIP code architecture. ................................ 25
3.3 Architecture design and the relations established between each component .......... 26
3.4 Data packets reception scenario. .......................... 27
3.5 Data packets send scenario. ............................ 28
3.6 In-Network Data Aggregation invocations process. .................. 31
4.1 CTP Routing Mechanism operation flow chart. ............... 35
4.2 Beacon Message format. ................................ 36
4.3 LA6 In-Network Data Aggregation interfaces implementation. .............. 37
4.4 LA6 data aggregation message format. ........................ 38
5.1 6LoWPAN deployment infrastructure. ........................ 45
5.2 LA6 Web Service Interface. .............................. 47
5.3 Database Entity Model. .................................. 48
6.1 6LoWPAN node deployment in the testbed. .................. 50
6.2 6LoWPAN node deployment in tree-based node organization scenario. ........... 51
6.3 Overhead number of messages comparison between LA6 In-Network Processing Mechanism and traditional 6LoWPAN. .................. 52
6.4 Overhead number of bytes comparison between LA6 In-Network Processing Mechanism and traditional 6LoWPAN. .................. 53
6.5 Number of messages transmitted comparison between LA6 In-Network Data Aggregation and traditional 6LoWPAN. .................. 53
6.6 Number of bytes transmitted comparison between LA6 In-Network Data Aggregation and traditional 6LoWPAN. .................. 54
6.7 Energy consumption comparison between LA6 In-Network Data Aggregation and traditional 6LoWPAN. .......................................................... 55
6.8 Number of messages transmitted comparison between LA6 In-Network Data Aggregation and traditional 6LoWPAN. ......................................................... 55
6.9 Energy consumption comparison between LA6 In-Network Processing Mechanism and traditional 6LoWPAN. .......................................................... 56
6.10 Number of beacons transmitted when detected a new node appears. ................. 56
List of Acronyms

IoT  Internet of Things
WSN  Wireless Sensor Network
WSNs Wireless Sensor Networks
IP   Internet Protocol
IPv6 Internet Protocol Version 6
IETF Internet Engineering Task Force
6LoWPAN IPv6 over Low power Wireless Personal Area Networks
LEACH Low-Energy Adaptive Clustering Hierarchy
SPIN Sensor Protocol for Information Negotiation
SPIN-EC SPIN-PP with a low-energy threshold
OPAG Opportunistic Data Aggregation
CTP  Collection Tree Protocol
PEGASIS Power-Efficient Gathering in Sensor Information Systems
TEEN Threshold sensitive Energy Efficient sensor Network Protocol
APTEEN Adaptive Threshold sensitive Energy Efficient sensor Network Protocol
DRIP Dissemination Routing Protocol
LE  Link Estimation Exchange Protocol
LQI  Link Quality Indication
BLIP Berkeley Low-power IP stack
ROLL-WG Routing Over Low power and Lossy networks Working Group
MANET-WG Mobile Ad-Hoc Network Working Group
RPL IPv6 Routing Protocol for Low power and lossy networks
LOAD 6LoWPAN Ad-hoc Routing Protocol
AODV Ad-Hoc On-Demand Distance Vector
BLE Bluetooth Low Energy
IEEE Institute of Electrical and Electronics Engineers
TDMA Time Division Multiple Access
FDMA Frequency Division Multiple Access
DSSS Direct Sequence Spread Spectrum
FHSS Frequency Hopping Spread Spectrum
<table>
<thead>
<tr>
<th>Abbreviation</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>GFSK</td>
<td>Gaussian Frequency Shift Keying</td>
</tr>
<tr>
<td>ED</td>
<td>Energy Detection</td>
</tr>
<tr>
<td>TCP</td>
<td>Transmission Control Protocol</td>
</tr>
<tr>
<td>UDP</td>
<td>User Datagram Protocol</td>
</tr>
<tr>
<td>ICMPv6</td>
<td>Internet Control Message Protocol Version 6</td>
</tr>
<tr>
<td>SAA</td>
<td>Stateless Address Autoconfiguration</td>
</tr>
<tr>
<td>6LoWPAN-ND</td>
<td>6LoWPAN Neighbour Discovery</td>
</tr>
<tr>
<td>NUD</td>
<td>Neighbour Unreachability Detection</td>
</tr>
<tr>
<td>SNMP</td>
<td>Simple Network Management Protocol</td>
</tr>
<tr>
<td>6GLAD</td>
<td>IPv6 Global to Link-layer ADdress Translation for 6LoWPAN Overhead Reducing</td>
</tr>
<tr>
<td>RA</td>
<td>Router Advertisement</td>
</tr>
<tr>
<td>DAD</td>
<td>Duplicate Address Detection</td>
</tr>
<tr>
<td>OSI</td>
<td>Open Systems Inteconnection</td>
</tr>
<tr>
<td>LoWPAN</td>
<td>Low Power Wireless Personal Area Networks</td>
</tr>
<tr>
<td>DLL</td>
<td>Data Link Layer</td>
</tr>
<tr>
<td>PHY</td>
<td>Physical Layer</td>
</tr>
<tr>
<td>CSMA-CA</td>
<td>carrier sense multiple access with collision avoidance</td>
</tr>
<tr>
<td>GTS</td>
<td>Guaranteed Time Slot</td>
</tr>
<tr>
<td>ISM</td>
<td>Industrial Scientific-Medical</td>
</tr>
<tr>
<td>BPSK</td>
<td>Binary Phase Shift Keying</td>
</tr>
<tr>
<td>ASK</td>
<td>Amplitude Shift Keying</td>
</tr>
<tr>
<td>O-QPSK</td>
<td>Offset Quadrature Phase Shift Keying</td>
</tr>
<tr>
<td>SIG</td>
<td>Special Interest Group</td>
</tr>
<tr>
<td>C-H</td>
<td>Cluster-Head node</td>
</tr>
<tr>
<td>N-CH</td>
<td>Non-Cluster-Head node</td>
</tr>
<tr>
<td>BS</td>
<td>Base Station</td>
</tr>
<tr>
<td>ETX</td>
<td>Expected Transmission</td>
</tr>
<tr>
<td>DAN</td>
<td>Data-Aggregation Nodes</td>
</tr>
<tr>
<td>WEI</td>
<td>Wireless Embedded Internet</td>
</tr>
<tr>
<td>PAN</td>
<td>Personal Area Network</td>
</tr>
<tr>
<td>IID</td>
<td>Interface Identifiers</td>
</tr>
<tr>
<td>MANET</td>
<td>Mobile Ad-Hoc Network</td>
</tr>
<tr>
<td>OLSR</td>
<td>Optimized Link State Routing Protocol</td>
</tr>
<tr>
<td>OS</td>
<td>Operating System</td>
</tr>
<tr>
<td>ROLL</td>
<td>Routing Over Low power and Lossy networks</td>
</tr>
<tr>
<td>DHCPv6</td>
<td>Dynamic Host Configuration Protocol Version 6</td>
</tr>
<tr>
<td>JSON</td>
<td>JavaScript Object Notation</td>
</tr>
<tr>
<td>GWT</td>
<td>Google Web Toolkit</td>
</tr>
<tr>
<td>LLN</td>
<td>Low-Power Lossy Network</td>
</tr>
</tbody>
</table>
Chapter 1

Introduction

We live in a changing world, extremely dynamic, resulting in a constant technological development. It started with the World Wide Web, in the 90’s, nowadays we live in a Mobile Internet paradigm and looking ahead is expected the third, and potentially most “disruptive” phase of the Internet revolution, the Internet of Things (IoT) [13]. Today there are roughly 1.5 billion Internet-Enabled personal computers and over 1 billion Internet-Enabled cell phones. With this revolution 50 to 100 billion devices connected to the Internet by 2020 are expected.

The IoT will connect the objects of the world in both a sensory and intelligent manner combining technological developments in item identification (“tagging things”), sensors and Wireless Sensor Networks (WSNs), (“feeling things”), embedded systems (“thinking things”) and nanotechnology (“shrinking things”) [57]. This will lead us to a new model of human-computer interaction in which information processing is thoroughly integrated into everyday objects and activities. This paradigm can also be described as pervasive computing [43] where “the things think”. Such pervasive computing creates the need of convergence between the actual WSNs [5, 27] and Internet. Wireless embedded devices have processor, memory and energy consumption constraints, the short communication range and dynamic environment makes WSNs highly unstable. Hence, WSNs have very different needs, in contrast to wired networks, which could justify a redefinition of the overall structure of Internet Protocol (IP) applications and services.

On the other hand, there have been significant advances in WSNs, with smart phones and some improved embedded devices capable of providing seamless interaction experience with IoT. There are still great challenges in the development of pervasive work support systems, WSNs are data-centric, i.e., a sensor node does not need an identity (address) because the communication is performed according to the information conveyed. Traditional networks demand the use of IP addresses, address-centric, and their operations are based on a point-to-point architecture versus the many-to-one Wireless Sensor Network (WSN)’s paradigm.

To deploy the IoT concept requires an enormous address space. In the actual context the Internet Engineering Task Force (IETF) proposed Internet Protocol Version 6 (IPv6) [28] which has addresses with 128 bits against the 32 bits of IPv4. From the junction of IPv6 and WSNs, IPv6 over Low power
Wireless Personal Area Networks (6LoWPAN)[29, 50] was created in order to be the main standard behind IoT.

6LoWPAN stems from the idea that IP should be applied even to the smallest embedded devices, so the IETF developed an adaptation layer to provide encapsulation and header compression mechanisms that allow IPv6 packets to be sent and received over IEEE 802.15.4 based networks. 6LoWPAN can guarantee interoperability between WSNs and Internet, due to IPv6. Nevertheless, this interoperability can hardly be absolute, because Internet is based on the traditional IP protocol which is application-independent, address-centric, with one-to-one communications, high data-rates, and with almost no resource constraints. On the other hand, WSNs are application-specific, data and location centric, with typical data flows one-to-many and many-to-one, supporting low data-rates, and are resources constrained[61].

In this context, to be viable, 6LoWPAN needs to be improved with some features, such as aggregation functions which will favour robustness, scalability and energy sustainability. Data aggregation is the process of aggregating data from multiple sensors in order to avoid redundant transmissions and assure compressed information to the data repository. Aggregation functions are mechanisms that optimize wired or wireless sensor communications according to information gathered by the sensor nodes. Such mechanisms have been studied since the beginning of WSNs research and they can improve the WSN objectivity, reduce the energy consumption, and deepens the data-centric paradigm in order to better satisfy the user’s application[34]. Currently there are several solutions within traditional WSN, such as Low-Energy Adaptive Clustering Hierarchy (LEACH), Collection Tree Protocol (CTP), Power-Efficient Gathering in Sensor Information Systems (PEGASIS), Directed Diffusion etc. However for 6LoWPAN there are no solutions developed, deployed, and properly tested.

1.1 Motivation and Objectives

This work aims to contribute to the IoT reality, trying to ensure the convergence between WSNs and Internet. In order to guarantee the adaptation of 6LoWPAN to a WSN paradigm, aggregation functions for the 6LoWPAN adaptation layer will be developed. This aggregation mechanisms will make 6LoWPAN much more data-centric, contributing to improve the energy sustainability of 6LoWPAN. It is also necessary because 6LoWPAN network has some characteristics that can be exploited such as heterogeneity, scalability of devices, and others that should be avoided as the use of large addresses or point-to-point communications.

So, to better adapt 6LoWPAN to a WSN paradigm aggregation mechanisms will be applied within, and outside of the 6LoWPAN. Concluding, all the processed data obtained will be stored in a repository where it can suffer additional process to create correlations with information from different networks, in order to create services with additional utility.
1.2 Document Organization

The remaining contents of this document are organized as follows: chapter 2 presents the state of the art which is organized by the key features presented from the related work; chapter 3 proposes an architecture for the In-Network Data Aggregation in order to be implemented; chapter 4 characterizes the implementation of the In-Network Data Aggregation mechanism; chapter 5 reports the architecture and implementation of the Off-Network Data Aggregation mechanism; Chapter 6 describes the tests and the evaluation done to this project; Finally the section 7 gives a complete conclusion and presents the future work chapter.
Chapter 2

State Of The Art

2.1 Introduction

In this chapter the current literature of in-network data aggregation will be briefly reviewed. Communication Technologies and 6LoWPAN will also be analysed in order to provide a comprehensive overview of the existing aggregation solutions.

Thereby, after presenting the most widely used communication technologies (layer 1 and 2 of OSI model respectively). The most representative existing in-network data processing will be also presented, explained and categorized according to an appropriate set of key characteristics.

Finally, 6LoWPAN will be addressed in its different aspects. Addressing, auxiliary mechanisms, routing, and Berkeley Low-power IP stack (BLIP) are some of the 6LoWPAN main features considered in order to identify eventual opportunities and obstacles for the integration of aggregation mechanisms.

2.2 Communication Technologies

2.2.1 IEEE 802.15.4

The IEEE 802.15.4 standard defines low-power wireless radio techniques better known as Low Power Wireless Personal Area Networks (LoWPAN). Nowadays IEEE 802.15.4 is a widely popular radio communication standard aiming at providing cheap low-power, with relaxed throughput requirements, short-range ubiquitous communications for embedded devices. Several other standards or stack specifications use IEEE 802.15.4 as their Physical Layer (PHY) and Data Link Layer (DLL), such as 6LoWPAN or Zigbee[6], etc.

The latter version of the standard is IEEE 802.15.4 - 2006[54] assures over-the-air data rates of 250 kb/s, 100kb/s, 40 kb/s, and 20 kb/s in different frequency bands using a carrier sense multiple access with collision avoidance (CSMA-CA) channel access mechanism. It supports short or extended addresses of 16 bit and 64 bits, respectively, with unicast and broadcast capabilities, and optionally a Guaranteed Time Slot (GTS) allocation mechanism. It operates over three different Industrial Scientific-
Medical (ISM) bands, 16 channels in the 2450 MHz band (worldwide use), 30 channels in the 915 MHz band (only in the United States), and lastly it is able to use 3 channels in the 868 MHz band (only in Europe).

In the band of 2.4 GHz, the PHY implements Direct Sequence Spread Spectrum (DSSS) with different phase-shift keying modulations. In the lower frequency bands, 868 and 915 MHz, the radio transceiver uses Binary Phase Shift Keying (BPSK) and Amplitude Shift Keying (ASK), whereas the upper band uses Offset Quadrature Phase Shift Keying (O-QPSK). Due to constraints imposed by the modulation schemes only the latter can provide 250 kb/s over-the-air with radio coverage between 10 and 70 meters.

Beyond the low power consumption IEEE 802.15.4 assures some other mechanisms as Energy Detection (ED), Link Quality Indication (LQI) and a fully acknowledged protocol for transfer reliability. An important characteristic, is the payload of the physical frame that can be up to 127 bytes in size, with 72 to 116 bytes of payload available after link-layer framing, depending on a number of addressing and security options.

### 2.2.2 Bluetooth Low Energy

Bluetooth Low Energy (BLE) is an integral part of Bluetooth 4.0[52] wireless radio technology for low-power and low-latency applications. It was defined by the Bluetooth Special Interest Group (SIG) with the purpose of transferring small amounts of data, payload maximum size is 27 bytes, with a very low cost per bit.

BLE operates in the unlicensed 2.4 GHz ISM band. It employs a frequency hopping transceiver to avoid interference and fading. BLE uses two multiple access schemes: Time Division Multiple Access (TDMA) and Frequency Division Multiple Access (FDMA) with 40 physical channels, separated by 2 MHz. Three of those channels are used as advertising channels, and 37 as data channels.

In association with FDMA a Gaussian Frequency Shift Keying (GFSK) modulation ensures a symbol rate of 1 Megasymbol per second (Ms/s) supporting the bit rate of 1 Megabit per second (Mb/s).

The physical channel is sub-divided into time units known as events. There are two different kinds of events, the Advertising and Connection events. Devices that transmit advertising packets on the advertising PHY channels are referred to as advertisers. Devices that receive advertising, on the advertising channels, without the intention to connect to the advertising device are referred to as scanners. So, the devices that need to form a connection to another device are the initiators. Once a connection is established, the initiator becomes the master device and the advertising device becomes the slave device, this communication architecture composes a Piconet. The devices in a Piconet use a Frequency Hopping Spread Spectrum (FHSS) to implement a specific frequency hopping pattern (pseudo-random ordering of the 37 frequencies), defined by the initiator device.

BLE can also exclude some of the frequencies that are used by interfering devices, this adaptive hopping technique can improve Bluetooth co-existence with static (non-hopping) ISM systems when co-located.
This new network organization and *modus operandi* obliged the definition of a new IETF draft for transmission of IPv6 Packets over BLE [46].

### 2.2.3 Summary

A summary of the main features of each communication technology can be seen in the following Table 1.

<table>
<thead>
<tr>
<th></th>
<th>IEEE 802.15.4</th>
<th>BLE</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Data-Rate</strong></td>
<td>20, 40, 100, 250 (kb/s)</td>
<td>1 (Mb/s)</td>
</tr>
<tr>
<td><strong>Output Power (mW)</strong></td>
<td>1 - 10</td>
<td>0.01 - 10</td>
</tr>
<tr>
<td><strong>Network Topology</strong></td>
<td>Star, Mesh</td>
<td>Star</td>
</tr>
<tr>
<td><strong>Modulation Schemes</strong></td>
<td>O-PSK, BPSK (with DSSS)</td>
<td>GFSK (with FHSS)</td>
</tr>
<tr>
<td><strong>Network Contention Protocol</strong></td>
<td>CSMA-CA</td>
<td>TDMA, FDMA</td>
</tr>
<tr>
<td><strong>Maximum Range (m)</strong></td>
<td>70</td>
<td>150</td>
</tr>
<tr>
<td><strong>Addressing Support</strong></td>
<td>16-bit Short</td>
<td>16-bit Short</td>
</tr>
<tr>
<td><strong>Operation</strong></td>
<td>GTS, ACKs</td>
<td>Master-Slave</td>
</tr>
<tr>
<td><strong>Other Mechanisms</strong></td>
<td>LQI, ED</td>
<td>Adaptive Frequency Hopping</td>
</tr>
</tbody>
</table>

Table 2.1: Comparison of the main characteristics of the IEEE 802.15.4 and BLE.

### 2.3 In-Network Data Aggregation

#### 2.3.1 Data Aggregation Mechanisms Overview

In-Network data aggregation is a complex problem that involves numerous aspects. So far, a wide range of routing protocols, aggregation mechanisms have been developed for traditional WSNs. In order to clarify the issues, understand the constraints and identify hypothetical implementation opportunities, in the following section will be categorized the existent solutions upon a taxonomy based on WSN nodes organization. To deploy a WSN there are some aspects that must be well defined in order to guarantee the appropriate solution. When you are deploying a WSN you must understand where are you establishing your network, what information do you want to transmit and with which purpose.

Despite of those aspects, there is a WSN feature that underlies and influences all the data aggregation mechanisms[20, 58]. That feature is the Network Architecture[4, 19] and is upon that characteristic that i will categorize the In-Network processing mechanisms analyzed.

It is also unequivocal that a WSN deployment is impossible without some type of In-Network data processing mechanism, such as routing protocols or data aggregation mechanisms.

To satisfy its purpose aggregation mechanisms are dependent and intrinsically linked to routing protocols because data aggregation is performed across the routing process, and therefore is natural to refer, as well, routing protocols from the traditional WSN. Hereafter, keeping this mindset i will review,
analyse the most diverse and representative WSN routing protocols according to the aggregation approaches implemented by them.

In this context I will review the following WSN routing protocols:

- **LEACH** [25] is a self-organizing and adaptive clustering protocol which explores clustered structures to perform data aggregation. In this protocol the aggregation is performed within each cluster, where the nodes gather data and send it to a Cluster-Head node (C-H) which performs the data aggregation retrieving the aggregated value to the sink node.

- Continuing, there is **CTP** [21, 23] that routes a message up the tree until the root (Sink), performing aggregation at every hop of the routing tree structure.

- With a chain organization there is **PEGASIS** [49] that routes a message up the chain towards the sink. These protocols perform data aggregation in every hop of the protocol communication, improving its functioning.

- **Threshold sensitive Energy Efficient sensor Network Protocol (TEEN)** [39] uses the same clustering organization of LEACH, but imposes thresholds (hard and soft threshold, whose functionality will be explained ahead) to limit transmissions according to user requirements.

- On the other hand, there is **Opportunistic Data Aggregation (OPAG)** [11] in WSNs that applies a relaxed timing approach for data retrieving, which is a consequence of the multi-path routing that underlies a tree-based aggregation process.

- Even though all solutions are data-driven, **Directed Diffusion** [18] and **Sensor Protocol for Information Negotiation (SPIN)** are the ones where this feature is more evident. Within Directed Diffusion the data gathered is identified by attribute-value pairs. In case a node desires some specific information it sends an Interest message over the WSN. If any node has information that matches the interest, then it sends it back to the node (Sink) that issued the interest. In Directed Diffusion the data aggregation is performed over the routing of the response to the interest issued.

- On the other hand, **SPIN** [35] is a three stage protocol because a node advertises its information gathered and if neighbours need that information they request it.

- Finally, there is an hybrid solution called **Tributaries and Deltas** [41] which combines the advantages of performing aggregation over a tree or multi-path topologies by running them simultaneously in different regions of the network.

![Figure 2.1: Data Aggregation Taxonomy.](image)
2.3.2 In-Network Data Aggregation Key Features

In order to characterize and identify the best approaches used in each one of the previous in-network data processing solutions, some key system features were defined to be used as a comparison criteria. The key features are:

- Energy Efficiency and Network Lifetime
- Timing strategies and Data Report
- Data accuracy
- Data representation and used bandwidth
- Implosion, overlap and resource blindness problems
- Latency
- Aggregation Scalability

The aforementioned characteristics were presented by order of relevance. Afterwards, all of them will be contextualised, and used for analysing the aggregation mechanisms previously stated, in order to understand their importance within WSNs.

Energy Efficiency and Network Lifetime [7, 9] evaluates the efficiency of a data aggregation scheme. For instance in an ideal data aggregation scheme each sensor should have expended the same amount of energy in each gathering round. Aggregation by itself is a process that takes advantage of redundant data, avoiding unnecessary transmissions, and thus enhancing network lifetime due to power savings. More precisely network lifetime is defined as the number of data aggregation rounds until x% of nodes die.

Within cluster-based aggregation as the case of LEACH, TEEN, and Adaptive Threshold sensitive Energy Efficient sensor Network Protocol (APTEEN) the following techniques are used to save energy: a periodically election of the C-Hs that will receive and aggregate the gathered data from its clusters nodes. The clustering aggregation organization promotes the local communication, with the exception of inter-cluster transmissions. The C-H establishes a TDMA transmission scheme, which reduces the message collision and allows the nodes to remain in sleep state, with transceivers powered-down for a long time. However the responsibility of being a C-H is much more energy intensive than being a Non-Cluster-Head node (N-CH), thus LEACH incorporates a randomized rotation of the high-energy C-H position among the sensors to avoid draining the battery of a specific group of nodes in the network. The sensor nodes pick its C-H based on the signal strength received from C-H reducing the energy expenditure to the minimum possible. Nevertheless the cluster-based aggregation has drawbacks, the lack of flexibility of the organization can cause a situation where sensors that are quite far from Base Station (BS) are elected leaders.
This situation obliges the C-H to an additional transmission energy waste. TEEN is an ad aggre-
gation scheme deployed over a hierarchical LEACH but with the addition of a hard threshold that
avoid unnecessary transmissions.

Another architecture dependent approach is CTP, a tree-based data aggregation mechanisms.
CTP is the most representative solution (it is the data aggregation solution implemented in TinyOS
section 2.4.5). When a query is sent by the sink node, the nodes composing the tree will
answer the query, aggregating the information along the tree structure, from its leaves to its root.
CTP has some auxiliary mechanisms such as Expected Transmission (ETX) to reinforce the best
routing paths (can cause message loops), CTP nodes have cache and queues to avoid duplicate
message transmission. It also defines epoch periods to transmit, allowing nodes to sleep, saving
energy when not participating in aggregation.

The aggregation can also be realized over a chain-structure, as PEGASIS does. PEGASIS also
use epoch time division, over a chain based multi-hop local communications, performing aggre-
gation at every level of chain, it normally has more levels than CTP, and has a rounding leader
responsibility as LEACH does, but with the improvement of avoiding the election of nodes that are
far away from the final sink.

In other perspective data aggregation can also in a non-structured way, using flat organization, as
happens in Directed Diffusion or SPIN. In terms of energy consumption SPIN performs aggrega-
tion only locally, and only when necessary. SPIN also has a version. SPIN-PP with a low-energy
threshold (SPIN-EC), which has a low-energy threshold to adapt its behaviour according to en-
ergy levels. Directed Diffusion saves energy through a short-range hop-by-hop communication,
the data is identified using attribute-value pairs composing an Interest message which is dissem-
inated across the sensor network setting up Gradients for that specific data Interest. The best
aggregation paths are reinforced by the sink node using those gradients, saving energy in poste-
rior aggregations. The Interest message has a range and lifetime values which avoid unnecessary
transmissions, saving energy. It also has location coordinates which avoids the network interest
flooding, improving network lifetime.

Concerning energy, OPAG opportunistically uses multi-path routing to compensate communication
losses, and achieve better energy efficiency than other schemes using retransmission. So, OPAG
presents itself as a good solution to support a tree-based aggregation scheme in an environment
with communication problems.

To finalize the energy analysis, Tributaries and Deltas is an hybrid solution that inherits the energy
advantages of the tree-based aggregation schemes, and the burden of a multi-path multi-hop
aggregation mechanisms. The multi-path multi-hop aggregation mechanisms are typically less
energy efficient because it sends the same message over multiple-paths expending much more
energy than other solutions mentioned before. Due to the two different approaches, Tributaries
and Deltas can adapt its behaviour according to energy needs.

**Timing strategies and Data Report**[55] define a time model for data transmissions. In the literature
there are three types of timing strategies, Periodic simple aggregation, Periodic per-hop aggregation, and Periodic per-hop adjusted[55]. The cluster-based aggregation approaches use a TDMA scheme defined by the C-H that defines the periods for sensing, transmission, and sleep of nodes. TEEN adds the Hard threshold that avoids transmission while values are not in the user’s interest range, but immediately starts transmission when the threshold is achieved. So LEACH performs a periodic per-hop aggregation while the TEEN can be a solution for critical time events.

Directed Diffusion floods the sensor network with the Interest messages which has a lifetime period for the aggregation process, and the rhythm of the aggregation. On the other hand SPIN has no timing strategy, when a node wants some information it requests it locally to the nodes who have published it.

Concerning Time, CTP and PEGASIS are periodic per-hop adjusted aggregation solutions theirs tree-based, and chain-based aggregation approaches use timing synchronization to define epochs in which the messages in each level of the tree or chain structure must be received, aggregated, and transmitted.

A different approach is realized by OPAG that despite having a tree-based aggregation layer it does not implement any time constraint or schedule for the transmissions on the multi-path routing layer, creating an opportunistic aggregation scheme.

Finally, Tributaries and Deltas has the same time schedule of tree-based aggregation scheme, being however limited by the multi-path communication performed at the tributary region.

**Data accuracy** is a key feature that evaluates the ratio of total number of readings received at the sink and total number of readings generated, although an exact definition of this feature depends on the specific application for which the WSN is designed. During the data aggregation process data accuracy can be affected by computation (aggregation functions used, or similar operations), communication errors, and data report rhythm.

Starting data accuracy comparison with the cluster-based aggregation schemes, LEACH offers a good data accuracy due to the reduced message losses caused by one-hop local communications and the TDMA applied which avoids message collision. TEEN can be used as complementary tool in all data aggregation mechanism because it applies a soft threshold that quantifies the variation of the value sensed in order to assure a more accurate sensing. However its configuration must be extremely accurate because an inappropriate threshold setting can inadvertently stop data collection.

CTP is a tree-based aggregation scheme, very sensitive to packet loss, because if an aggregated message is lost, the aggregated values of the sub-tree will be lost. To overcome this lack of robustness CTP uses queues to verify duplicate messages, loop transmissions, incoherent aggregation situations, and a best-effort policy of retransmission to assure message delivery to the upper levels of the tree, assuring the aggregation process. However, the best-effort retransmission policy can create duplicate messages which can affect data accuracy. From all solutions PEGASIS is the
most sensitive because it has no compensation mechanisms to overcome its chain-structure.

In Directed Diffusion the aggregation accuracy is achieved by the definition of *Interest* messages which specifies objectively the rhythm, timing, and precision of the values to aggregate. The interest caching, and path reinforcements made in Directed Diffusion can also avoid message losses and consequently improve accuracy. SPIN uses meta-data to succinctly describe the data desired when advertised, and only performs aggregation locally, so from all solutions SPIN is less affected by transmission errors.

OPAG overcomes the accuracy problems of the tree-based aggregation through the underlying multi-path routing which guarantees almost no packet loss, improving the aggregation layer performance.

Finally, Tributaries and Deltas inherits the robustness of multi-path communication in the tributary region to compensate the less accurate aggregation assured in delta region. Tributaries and Deltas also have an auxiliary mechanism to avoid duplicate messages, and a mechanism to find the most frequent items, which can improve aggregation process.

The way data is represented is directly correlated to data content, and to operations applied over it. These data aggregation operators can be lossy or lossless, and affect directly data accuracy, and the following key feature under analysis, the data representation and used bandwidth. So, in Table 2.2 are characterized the principal aggregation operators, the resulting aggregates, and their properties.

In the context of Data accuracy there are three properties of aggregates, from Table 2.2, that must be considered. The aggregates can be *Duplicate Sensitive, Exemplary or Summary*, and *Monotonic*. These aggregates properties are important, and should be contemplated during the design of data aggregation mechanism in order to assure aggregation data accuracy. For a more precise explanation consult [38].

**Data representation and used bandwidth** is an important feature because it reflects the effectiveness of the aggregation mechanism. It is also directly related with data aggregation functions realized over data, and its results have direct impact in used bandwidth. Some of the most used aggregation functions are shown in Table 2.2.

<table>
<thead>
<tr>
<th>Duplicate Sensitive</th>
<th>Max, Min</th>
<th>Count, Sum</th>
<th>Average</th>
<th>Median</th>
<th>Count Distinct</th>
<th>Histogram</th>
</tr>
</thead>
<tbody>
<tr>
<td>Yes</td>
<td>No</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
<td>No</td>
<td>Yes</td>
</tr>
<tr>
<td>Exemplary, Summary</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Yes</td>
<td>Yes</td>
<td></td>
<td></td>
<td>Yes</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Monotonic</td>
<td>Yes</td>
<td>Yes</td>
<td>No</td>
<td>No</td>
<td>Yes</td>
<td>No</td>
</tr>
<tr>
<td>Partial State</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Yes</td>
<td>Yes</td>
<td>Distributive</td>
<td>Algebraic</td>
<td>Holistic</td>
<td>Unique Sensitive</td>
<td>Content</td>
</tr>
</tbody>
</table>

Table 2.2: Classes of Aggregates according to [38].
clusters such as LEACH, implies a number of messages equal to the number of nodes within a cluster plus a final aggregate message. Nevertheless, due to the thresholds, TEEN can reduce the used bandwidth significantly.

Tree-based approaches as CTP can perform aggregation in every level of the tree, and so reduce the size of the aggregated message. In good transmission conditions CTP achieves a reduced bandwidth usage because a message from some leaves is converted into one aggregated message. The same happens in PEGASIS that performs aggregation in every level of the chain, reducing the message transmission to one message per node on each round.

Continuing the analysis of data representation and used bandwidth, Directed Diffusion transforms a human query into attribute-value pairs composing an Interest message that can be stored by every node in an interest cache. SPIN realizes data negotiation process using two messages, the ADV (new data advertisement, advertises if it has data to share, contains only meta-data), REQ (request for data, if a node has interest in that sensed data). Both solutions use a significant bandwidth, Directed Diffusion uses a multi-path routing to spread the Interest message, and SPIN floods the sensor network with Advertise messages.

In terms of the aggregation, OPAG uses the same aggregation structure than CTP, but has an additional communication and storage overhead. The communication overhead is a consequence of Data-Aggregation Nodes (DAN) announcements and the extra bits transmitted to compensate communication losses.

Finally, the hybrid solution as any other solution, is dependent on the data gathered, and on the operations applied over it. Tributaries and Deltas inherits the bandwidth usage from the tree-based aggregation, but if the tributary region (multi-path communication) becomes very large, it can use more bandwidth than what is necessary.

**Implosion, overlap and resource blindness problems**[35] are three problems that try to evaluate the inadequacy of a protocol to WSN. Implosion is the resources waste of sending multiple copies of the same data to the same node. Overlap identifies a situation where sensor nodes cover overlapping geographic areas. Finally the resource blindness problem verifies if the WSN is “resource-aware”, and adapts its behaviour according to the resources available.

The cluster-aggregation approaches, LEACH and TEEN, solve the Implosion the cluster-scheme where each node only sends one message, the resource blindness is solved by the randomized C-H election process, however the overlapping of cluster-areas can not be resolved.

Within flat topologies, Directed diffusion takes advantage of its interest cache, and informations within each Interest message to avoid implosion and overlap. The resource blindness is solved due to the path reinforcement mechanism. An other approach is used by SPIN which solves all problems with the negotiation process realized during the aggregation.

CTP, accomplishes Resource Blindness using reinforcement mechanisms that pick the best paths among all, in order to keep the aggregation burden evenly distributed. However, CTP does not solve the implosion problem despite of using queues, and caches, because it still has the loop
problem. In CTP, as happens in PEGASIS, the Overlap problem is solved with the nodes organization.

PEGASIS is better than LEACH in Resource blindness concern due to the condition which avoids the election of some nodes to become leaders in some harmful situations, and SPIN-EC version adapts protocol behaviour according to battery levels. The implosion in PEGASIS is avoided by sending only one aggregated message on every round.

OPAG, and Tributaries and Deltas inherit some mechanisms used in the tree-based approaches, and some problems as the implosion caused by multi-path communication, which can cause implosion.

**Aggregation latency** is the time interval between the aggregate packets being received at the sink and the data gathered by source nodes.

About aggregation latency the cluster-based aggregation mechanisms as LEACH or TEEN have a reduced delay, because the aggregation is made within a few communication hops.

Within the flat topologies, SPIN has a reduced latency, because the aggregation is performed locally between neighbours, but in Directed diffusion the aggregation latency can be high. In Directed Diffusion the aggregation is performed over a multi-hop communication, and the latency depends on the number of hops covered by the aggregation process, however the number of hops or the lifetime of an *Interest* message can control aggregation latency.

The same happens in the tree-based, CTP, where the aggregation is dependent on the number of levels of the tree plus the time for data processing at each node. The latency is even more problematic in PEGASIS where the chain tends to have a higher number of levels, causing very high delay.

Finally, OPAG due to the opportunistic aggregation can cause high delays, and the aggregation latency in Tributaries and Deltas behaves according to the size of the tree-based or multi-path areas.

**Aggregation scalability** evaluates if it is possible to apply an aggregation mechanism in a large, and scalable WSN.

Within cluster-based aggregation scheme LEACH is scalability constrained, because C-H must transmit the aggregated result to the BS, this limitation does not happen in TEEN that use a Hierarchical LEACH which allows the communication between C-Hs to enable the expansion of the aggregation.

The tree-based aggregation mechanisms, such as CTP are well suited to scale, due to the possibility of creating numerous levels for the tree, enabling aggregation over a large area.

PEGASIS is a good solution, for a scalable aggregation problem, being however limited by the rapid growth of latency.

In a flat organization, SPIN scalability is not possible because the aggregation is performed locally, but in Directed Diffusion the possibility of flooding the network with an *Interest* message allows the
aggregation process to be realized over a large area.

Tributaries and Deltas is scalable due to the ability to expand the delta-region, and finally OPAG inherits the scalable characteristic from the tree-based aggregation schemes improved by the robustness offered by the underlying multi-path routing layer.

2.3.3 Summary

In the present section a brief summary of the key features comparison realized is shown the next table.

Table 2.3: Data Aggregation Mechanisms comparison summary.

<table>
<thead>
<tr>
<th></th>
<th>LEACH</th>
<th>PEGASIS</th>
<th>CTP</th>
<th>Tributaries and Deltas</th>
<th>SPIN</th>
<th>Directed Diffusion</th>
<th>TEEN</th>
<th>OPAG</th>
</tr>
</thead>
<tbody>
<tr>
<td>Energy Efficiency</td>
<td>Moderate</td>
<td>Very Good</td>
<td>Good</td>
<td>Moderate</td>
<td>Good</td>
<td>Good</td>
<td>Good</td>
<td>Moderate</td>
</tr>
<tr>
<td>Timing Strategies</td>
<td>Periodic Per-Hop</td>
<td>Periodic Simple</td>
<td>Periodic Per-Hop</td>
<td>Hybrid</td>
<td>Not Periodic</td>
<td>User Dependent</td>
<td>User Dependent</td>
<td>Not Periodic</td>
</tr>
<tr>
<td>Data Representation</td>
<td>Low</td>
<td>Low</td>
<td>Low</td>
<td>High</td>
<td>Low</td>
<td>Low</td>
<td>Variable</td>
<td>High</td>
</tr>
<tr>
<td>Used Bandwidth</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Data Accuracy</td>
<td>Good</td>
<td>Moderate</td>
<td>Moderate</td>
<td>Variable</td>
<td>Good</td>
<td>Good</td>
<td>Good</td>
<td>Good</td>
</tr>
<tr>
<td>Implosion Overlap</td>
<td>Yes</td>
<td>No</td>
<td>Yes</td>
<td>No</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>Resource Blindness</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>Latency</td>
<td>Good</td>
<td>High</td>
<td>Good</td>
<td>Variable</td>
<td>Low</td>
<td>Variable</td>
<td>Variable</td>
<td>High</td>
</tr>
<tr>
<td>Aggregation Scalability</td>
<td>Limited</td>
<td>Limited</td>
<td>Good</td>
<td>Very Good</td>
<td>Very Good</td>
<td>Very Limited</td>
<td>Very Good</td>
<td>Very Good</td>
</tr>
</tbody>
</table>

2.4 6LoWPAN

Presently there is a huge range of applications which would benefit from the Wireless Embedded Internet (WEI) approach. There are already some proprietary technologies[8] that in some contexts can be useful, but interoperability is required. IETF 6LoWPAN working group, officially created in 2007, has written a problem statement document "6LoWPAN: Overview, Assumptions, Problem Statement and goals"[36]. The main purpose was the creation of 6LoWPAN standards to enable the efficient use of IPv6 over low-power, low-rate wireless networks on a simple embedded device through an adaptation layer and the optimization of related protocols. Although other complementary solutions were proposed, such as IPv6 Global to Link-layer ADdress Translation for 6LoWPAN Overhead Reducing (6GLAD)[62] and IPv6 Label Switching[47] are two good examples.
2.4.1 Addressing

An IP adaptation layer normally involves two kinds of addresses, the link-layer (Layer 2 Open Systems Interconnection (OSI) Model) and IP (Layer 3 OSI Model) addresses. Nevertheless, due to addressing modes granted by Institute of Electrical and Electronics Engineers (IEEE) 802.15.4, 6LoWPAN has also IEEE 64-bit extended addresses, and 16-bit short addresses[50], dynamically unique assigned within the PAN during the Stateless Address Autoconfiguration (SAA) (explained in the following section 2.4.2).

The 64-bit IEEE EUI-64 (Extended Unique Identifier) is a globally unique address, typically assigned by the manufacturer of the device, whereas the short addresses are transient in nature, since they are doled out by the Personal Area Network (PAN) coordinator during the sensor network bootstrap, the uniqueness and validation is therefore limited by the lifetime of that association. Additional attention must be given to the short 16-bit addresses according to the purpose of the address (Unicast, Multicast).

Two such addresses are needed for each 6LoWPAN interface: a link-local address, constructed from the prefix FE80::/10, and a globally routable address, constructed from the globally routable prefix of the LoWPAN.

2.4.2 Stateless Address Autoconfiguration and 6LoWPAN Neighbour Discovery

Using SAA each host generates a link-local IPv6 unicast address from its IEEE EUI-64 address, 16-bit short address or both. In the case of using a IEEE EUI-64 address, the acquisition of IPv6 address is determined combining a 64-bit network prefix and 64-bit host address, to yield a 128-bit value.

In the case of 16-bit short addresses, a "pseudo 48-bit address" is built concatenating 16 zero bits and 16-bit PAN ID producing the 32 left-most significant. These 32-bit will consequently be aggregated with the 16-bit short address.

\[16\text{bit}_PAN\_ID : 16\text{zero}_bits : 16\text{bit}_short\_address\]

However, in the resultant interface, the "Universal/Local" (U/L) bit shall be set to zero, because this is not a globally unique value. In 6LoWPAN this derivation of IPv6 addresses from link-layer addresses is mandatory, because it enables 6LoWPAN Neighbour Discovery (6LoWPAN-ND) mechanism. The IPv6 interface identifier is formed from this 48-bit pseudo address as per the IPv6 over Ethernet specification. This specifies that the first 3 bytes of the 48-bit address are followed by FFFE.

\[16\text{bit}_PAN\_ID : 0x00 : 0xFF : 0xFE : 0x00 : 16\text{bit}_short\_address\]

The 6LoWPAN-ND allows neighbour discovery, maintain reachability information, configure default routes, and propagate configuration parameters. 6LoWPAN-ND performs address resolution, and Neighbour Unreachability Detection (NUD) through unicast queries. Each 6LoWPAN host listens for Router Advertisement (RA) from 6LoWPAN routers (consult Routing Over Low power and Lossy networks (ROLL) Architecture in Figure 2.3) to receive important network information, such as network prefixes, default routing hop limit, for configuration of default routes, etc.
6LoWPAN-ND also has the responsibility of performing Duplicate Address Detection (DAD). To execute this additional burden the 6LoWPAN routers use link-local multicast to send the RAs and DAD solicitations.

### 2.4.3 Header Compression

As stated before, the IEEE 802.15.4 frame has a maximum payload of 127 bytes. IPv6 has a typical header size of 40 bytes, and a maximum transmission unit (MTU) of 1280 bytes. To enable IPv6 over WSN, 6LoWPAN adaptation layer (RFC4944) defined the format for fragmenting the IPv6 datagrams and compressing IPv6 headers.

The 6LoWPAN header compression is stateless and applies two techniques to compress data. The first approach is making assumptions about common values for IPv6 header fields in WSNs (Version is 6, Class and Flow Label are 0, Next Header is User Datagram Protocol (UDP), Transmission Control Protocol (TCP), or Internet Control Message Protocol Version 6 (ICMPv6), and IPv6 prefixes are link-local), and the second is removing redundant information across layers (Payload Length and Interface Identifiers (IID), are derived from the link header). It is also defined a similar methodology for UDP header compression.

6LoWPAN does not compress efficiently headers when communicating over multi-hops (due to global addresses), using multicast or with nodes outside of the link-local scope. For a more detailed explanation of the header compression implementation consult [26, 50].

### 2.4.4 Routing

The routing in 6LoWPAN is a sensitive issue due to WSN resource constraints[32].

![Figure 2.2: Routing implementation approaches: Link Layer Mesh-Under, 6LoWPAN Mesh-Under, and Router-Over respectively.](image)

According to the layer where the routing decisions are made are defined three types of routing, the Route-Over (routing performed at IPv6 network layer), the 6LoWPAN Mesh-Under[12](routing realized within 6LoWPAN adaptation layer), and finally the Link-Layer Mesh-Under[44] (routing is implemented at link layer, under 6LoWPAN adaptation layer). These three solutions are illustrated in Figure 2.2.
About the previous routing approaches a detailed discussion in aggregation perspective is made in section 2.4.6.

However, 6LoWPAN is not the Internet, and as long as we assume the premises of the actual world wide web, it will not be. Even so, IETF established two working groups from IETF, Mobile Ad-Hoc Network Working Group (MANET-WG) and Routing Over Low power and Lossy networks Working Group (ROLL-WG)[17, 30]. The protocols developed in MANET-WG were mainly meant for routing in ad-hoc mobile networks. IETF has proposed some protocols from Mobile Ad-Hoc Network (MANET), disregarding the WSN paradigm.

![Figure 2.3: ROLL-WG 6LoWPAN typical architecture, and deployment scenario.](image)

Taking a different approach, ROLL-WG started with a WSN requirements analysis followed by the study of the existing routing proposals. ROLL-WG analysed the 6LoWPAN deployment requirements and defined a typical 6LoWPAN network organization, shown in Figure 2.3.

ROLL-WG concluded that without improvements, the existing solutions such as Ad-Hoc On-Demand Distance Vector (AODV)[15], Optimized Link State Routing Protocol (OLSR)[14], etc. could never be sustainable in WSN. So, other solutions were developed, LOAD[31] and HYDRO[16] are some examples. LOAD is a routing protocol that arises from some enhancements made in AODV, and Hydro was an important contribution for the final routing proposal made by ROLL-WG, IPv6 Routing Protocol for Low power and lossy networks (RPL)[59], and is the routing protocol implemented in BLIP 1.0. In 2010, the working group introduced RPL. This solution intends to provide efficient routing paths for the following traffic patterns multipoint-to-point, point-to-multipoint and point-to-point traffic in Low-Power Lossy Network (LLN). When RPL node obtains a proper global IPv6 address, it tries to form a Destination Oriented Directed Acyclic Graph (DODAG) by exchanging ICMPv6-based DODAG information Solicitation (DIS) or DODAG Information Object (DIO) messages. This routing solution supports a multipointtopoint (data collection) traffic with little routing state, whereby each node stores the next hop leading to the single
destination, the root of the DODAG. The design of RPL inherits from the a decade of WSN research. It can offer a solution equivalent to CTP but to 6LoWPANs reality. Nowadays, RPL implementations in TinyOS affected two layers of the software stack BLIP and TinyOS 6LoWPAN stack. TinyRPL is the actual implementation of RPL Standard.

2.4.5 TinyOS and Blip

Currently there are numerous solutions, under development, whether proprietary or open-source. Among the open-source solutions there are some worth mentioning, such as uIPv6 (currently SICSlowpan)[53] build over Contiki\(^1\), and BLIP\(^2\) developed over TinyOS-2.x[22]. Within the proprietary solutions there are NanoStack (Sensinode)[50] and Jennic 6LoWPAN (Jennic Companies).

In the present section TinyOS and BLIP will be reviewed in order to better understand the operating system where the future work will be developed.

TinyOS began as a collaboration between the University of Berkeley and Intel Research, currently it is being developed by an international consortium, the TinyOS Alliance. TinyOS is an event-driven embedded operating system in its third release, TinyOS 2.1.1. It was thought and designed for extremely restricted resource devices such as the WSN motes. The TinyOS implementation has a very small footprint, with the core Operating System (OS) requiring only 400 bytes of code and data memory. In order to offer a flexible deployment, it features a component based architecture of reusable elements which can be combined together. The components implement hardware abstractions and the TinyOS concurrency model enables a fine grained power management due its event-driven execution model.

It is an open-source operating system written in nesC\(^3\) programming language. NesC is a dialect of the C language with no dynamic memory allocation and no dynamic linking. NesC applications are based on interfaces and components. The TinyOS core distribution is based on the following network protocols: CTP, Dissemination Routing Protocol (DRIP), Link Estimation Exchange Protocol (LE), and MultiHopLQI.

As previously stated, CTP is the de-facto standard data collection protocol for TinyOS-2.x, and BLIP is the de-facto IPv6/6LoWPAN stack for TinyOS 2.x which do not use CTP as its routing protocol. Nowadays, in BLIP 2.0\(^3\) an implementation of the ROLL proposal is being developed, the TinyRPL[33] that is intended to replace CTP within the 6LoWPAN context.

The protocol stack of BLIP, shown in Figure 2.4, is composed by three layers. The Transport layer which enables two standard Internet protocols UDP and TCP, the Network layer where the routing is realized according to Hydro, and finally the 6LoWPAN implementation that provides a 2.5 abstraction layer on top of the link layer. The latter is responsible for header manipulation, fragmenting packets, etc.

\(^1\)http://www.sics.se/~adam/contiki/docs/
\(^2\)http://docs.tinyos.net/tinywiki/index.php/BLIP_Tutorial
\(^3\)http://docs.tinyos.net/tinywiki/index.php/BLIP_2.0
2.4.6 Discussion from an In-Network Data Processing Perspective

The main communication technologies capable of supporting 6LoWPAN are IEEE 802.15.4, and BLE which is still under development.

The actual aggregation mechanism literature is vast and only considers solutions for the traditional WSN, and none for 6LoWPAN. After the research realized it was concluded that does not exist any theoretical aggregation solution or implementation.

As can be seen in Figure 2.3, the underlying 6LoWPAN typical architecture requires the existence of sensor nodes with different functions, called Full Function Devices (FFD) that normally are Border Routers responsible for routing or 6LoWPAN gateway operations and support other auxiliary mechanisms of 6LoWPAN, and also require Reduced Function devices (RFD) which normally perform are called Hosts which only communicate and perform sensing. This 6LoWPAN issue imposes a hierarchy to the future aggregation mechanisms to be embedded.

The actual 6LoWPAN standard, and BLIP’s implementation supports Mesh-Under and Route-Over. So it is possible to incorporate aggregation mechanisms in both layers, at network and within 6LoWPAN adaptation layer. The 6LoWPAN standard also offers IPv6 SAA, 6LoWPAN-ND, Dynamic Host Configuration Protocol Version 6 (DHCPv6), a common network prefix, and a “pay-as-you-go” adaptation paradigm, in order to offer the maximum interoperability. However the previous mechanisms are still under development in BLIP 2.0 release.

The classical view of IP routing (Route-Over) presupposes a permanent connectivity between nodes which demands a network graph (topology formation), an address-based and table-driven routing algorithm. The previous assumptions are against the WSN paradigm, where there is no network graph a priori, no addressing or connection guarantees, because IEEE 802.15.4 networks are limited in bandwidth, power transmission, coverage range, memory, CPU processing, and battery.

The Route-Over approach is address based consequently it requires 6LoWPAN fragmentation and reassembly at every radio hop. As a consequence the transmission of fragments of a given datagram to its final destination through multiple paths can not be done (fragmentation and reassembly is performed at 6LoWPAN adaptation layer, and therefore, only the first fragment carries the IP header). The previous
characteristic limits the appliance, of any aggregation mechanisms which use multi-path routing. IP routing in LoWPAN is also limited by some characteristics, such as the forwarding on a single interface, the common IPv6 network prefix shared by all sensor nodes, which are strict resource constrained, and the constant routing changes. Additionally, the connection of 6LoWPAN networks (typically stub networks) to the Internet involves the development of two different routing protocols, intra and inter 6LoWPAN routing. The need of two routing protocols is an additional obstacle for the integration of an aggregation mechanism.

In other hand, in the Mesh-Under approach PAN is a single IP link. This paradigm derail the use of network diagnostic tools, such as traceroute or Simple Network Management Protocol (SNMP), because every node is one hop away. In this context if the data aggregation solution will be a Mesh-Under solution it must coexist with the IPv6 reality of 6LoWPAN to avoid the development of new tools. An another obstacle is the fact that broadcast is no longer a basic communication primitive (only receive if listening, link local broadcast requires either more wake up or more coordination than link local unicast), which creates additional complexity to the development of aggregation mechanism that uses that functionality. Despite of that the Mesh-Under approaches are more flexible fragments can be delivered over multiple hops without requiring fragmentation and reassembly at each hop. Mesh-under also provides the use of multiple paths to deliver fragments of a given datagram favouring the aggregation development.

The real world scenarios must be considered during the solution design because real environment that will evaluate and test if the designed solution is successful.

The data aggregation solution can also be implemented at application level. But this solution is not practical because it would require that every sensor node should have that application installed causing interoperability problems with other applications. This approach is also inadequate because it would require that every packet should pass through all decapsulation steps in order to evaluate if the data conveyed should be aggregated or not. This reality would require unnecessary processing and complexity.

According to the research performed and my perspective the adequate aggregate solution should be developed at 6LoWPAN Mesh-Under routing level because it is more reliable to deliver fragments, and outperforms Route-Over in terms of delay, processing, flexibility, fits well in a stub network paradigm, despite having additional complexity and not offering a transparent global connectivity.

After the analysis realized to the literature, the well suited existent aggregation mechanism to fit into a 6LoWPAN is CTP, despite being address-free, its network topology fits well in 6LoWPAN network topology. Despite having some issues identified in section 2.3.2, CTP has the advantage of being a well tested routing protocol with embedded aggregation, and was the chosen protocol to be incorporated in TinyOS 2.x implementation. The fact of being address-free can be useful, because at each level the sensor nodes know to which node they should send the information to, and so can avoid the use of addresses when they are not necessary, in the aggregation process.

Resuming we intend to develop a data aggregation mechanism to 6LoWPANs based on CTP without affecting 6LoWPANs thus contributing to make 6LoWPANs more adequate to WSN reality.
Chapter 3

6LoWPAN In-Network Data Aggregation Mechanism Architecture

Here, will be discussed the design approach to the LA6 In-Network Data Aggregation Mechanism’s architecture, starting off by describing the traditional 6LoWPAN deployment scenarios. To implement in-network data aggregation BLIP 6LoWPAN existent implementation were analysed in detail. To achieve data aggregation the LA6 implementation shall identify a single point where it is able of intercepting all packets receipted in every sensor node. After intercepting these packets of interest, they will be processed according to the configurations. Otherwise, if the messages do not interest, they will be transmitted over the air.

3.1 Traditional 6LoWPAN Sensor Network Architecture

There are uncountable different WSN usage scenarios. A sink node and small sensor devices, energy constrained, low-cost embedded devices normally compose a typical WSNs. Due to these limitations, academy researchers developed numerous data aggregation mechanisms, ad-hoc routing, and distributed signal processing for low-power multi-hop wireless networks. These approaches exploit the local communications, application specific protocols and data content (data-centric). The utilization of these in-network data aggregation mechanisms can drastically reduce the amount of information transmitted, reducing costs and increasing WSN nodes lifetime.

WSN consists of spatially distributed autonomous sensors to monitor physical or environmental conditions, such as temperature, light, etc. Considering the WSN deployment context 6LoWPAN were developed. The typical 6LoWPAN sensor network architecture, visible in Figure 3.1, is composed by sensors that cooperate using multi-hop communication to transmit the gathered information from the WSN to a central repository. 6LoWPAN implements IPv6, an address-based approach using addresses of 128 bits over the IEEE 802.15.4 frames of 128 bytes maximum. In 6LoWPAN deployment architecture each sensor node has an IPv6 address, enabling point-to-point communication.

6LoWPAN is evolving and adapting itself to the WSN reality, through the development of some header
3.2 6LoWPAN In-Network Processing Architecture Design

3.2.1 6LoWPAN In-Network Data Aggregation Mechanism Architecture Overview

One possible implementation of 6LoWPAN adaptation layer is BLIP. It implements 6LoWPAN stack over TinyOS through the instantiation of the architecture visible in Figure 3.2. Thus, IPDispatch, IPRouting, IPAddress and IPExtensions components compose BLIP. Its architecture organization implements a central component, IPDispatch, supported by the remaining modules, being each one responsible for different operations, complementing IPDispatch in its functionalities.

IPDispatch performs IEEE154 frames decapsulation, IPv6 message handling, packet fragmentation, packet assembling and packet forwarding. IPDispatch plays a central role, through it flow all messages received, sent or forwarded.

To better understand the LA6 solution architecture the BLIP normal operation will be briefly explained. The BLIP central operation is performed in IPDispatch. When an IEEE154 data frame is received, IPDispatch decapsulates and, if necessary, reassembles the IPv6 packet. Having the IPv6 packet, it invokes IPRouting to analyse the IPv6 header in order to determine if it is a data or a ICMP packet. If it is an ICMP packet it is routed directly to ICMP implementation component. Otherwise, if it is a data packet, IPRouting shall verify if the packet is for itself. If so, delivers it to UDP layer, if not forwards it according to its IPv6 address. To perform the IPv6 header manipulation, IPDispatch and IPRouting invoke some procedures of IPAddress. Regarding routing, IPRouting applies a Route-Over approach based on IPv6 addresses.

In this context, to be viable, the solution architecture can not be defined without IPDispatch central component. To perform data aggregation, LA6 In-Network Data Aggregation must intercept every data message to decide whether to aggregate or to forward the data message. To do so, we have created a procedure, in the IPDispatch component, responsible for intercepting and decide whether to perform data aggregation.
Data aggregation mechanism should be based on a tree-based routing protocol in order to create, and take advantage of the data aggregation opportunities. In this subject CTP is definitely the de facto data collection protocol of traditional WSNs. Therefore, in order to meet the solution requirements, some functionalities of CTP were incorporated in the data aggregation solution.

Analysing CTP it was concluded that to ensure a tree-based routing protocol, the following three components: LinkEstimator, CtpRoutingEngine, CtpForwardingEngine are necessary. In Figure 3.3 is shown how these components are integrated in the proposed in-network data aggregation solution. Despite taking advantage of CTP, this solution is designed to avoid being dependent of CTP to perform data aggregation. To ensure this, an additional component, AggregationEngine was developed. This component is responsible for performing data aggregation regardless of the routing protocol used in the aggregation process. In other words, this solution takes advantage of CTP tree-based routing to create data aggregation opportunities that will be used by the AggregationEngine to realize data aggregation. For this reason, this design approach is extremely flexible and adaptive because it ensures data aggregation independent by the routing protocol or the routing approaches used, namely Mesh-Under and Router-over. Since, the AggregationEngine was developed to perform data aggregation regardless of the packet routing implementation, it will handle and manage data aggregation configurations using the traditional default routing mechanism implemented in IPRouting.

In Figure 3.3 the solution architecture and the relations between each component are shown. Using the same approach as in BLIP design where IPDispatch interacts with IPRouting, IPAddress and IPExtensions, an additional Intercept interface is implemented, being responsible for intercepting messages. The interactions with AggregationEngine component are only realized from the Intercept interface. This
interface implements the same functionalities that it had in the old CtpRoutingEngine. Since every data message traffic flows through IPDispatch, all data messages will be intercepted and analysed by the Intercept and according to the AggregationEngine configurations is realized, or not data aggregation according to message data content and the information pigged-back in the additional CTP header decides whether to aggregate or to forward data messages. Therefore, the possibility of analysing the packets is given by the additional header inherited from CTP. The encapsulation of BLIP message by the CTP header is necessary, because is based on the aggregation identifier conveyed by this header that the aggregation process decision is made. Additionally, the AggregationEngine will also be responsible for performing statistics of the execution process.

The present data aggregation mechanism maintains the tree-based architecture and performs aggregation based on the CTP data header conveyed in data messages. After the developments realized at 6LoWPAN adaptation layer, the insertion of this auxiliary header in aggregated data messages achieved. This solution is developed over IEEE154 stack, but to preserve CTP functionality intact, the solution relies on the AM stack as well. This additional stack is necessary for maintaining CTP tree-based routing architecture through beacon probing.

Regarding the In-Network Data Aggregation mechanism architecture, the invocations of procedures of AggregationEngine and CTP are restricted to the Intercept interface, in order to bound the changes realized in IPDispatch, ensuring that all IPv6 functionalities provided by BLIP remain intact. In accordance with Figure 3.5, 3.4 and in order to clarify its operation, the flow diagrams of packet reception and sending scenarios are depicted. IEEE154 frames are received and decapsulated in IPDispatch. After that, the resulting packet is intercepted by the Intercept interface that will evaluate if it conveys an addi-
tional CTP header making that packet susceptible of aggregation. In case of having that header it will be redirected to the AggregationEngine component that will decide whether to aggregate (highlighted in red) or to forward (highlighted in green). If this mote is not aggregating such type of information carried by this data packet, it will be passed to the CtpForwardingEngine that will forward the packet through the IEEE154 interface (highlighted in green). The frames received that do not have the additional CTP header will follow the normal process within BLIP stack (highlighted in blue).

As for the sending scenario in Figure 3.5, the information produced locally in WSN applications will flow from the application layer, through the Service layer implementation, namely UDP, arriving at IPDispatch. At IPDispatch, the message is intercepted by the Intercept Interface, and according to the type of information produced it will be redirected, or not, to the AggregationEngine. If the data produced is not susceptible of being aggregated it will not be redirected to the AggregationEngine and will follow its normal course (highlighted in blue). Otherwise, if the AggregationEngine is configured to aggregate
that type of information, it will be aggregated locally (highlighted in red), being forwarded at the end of the aggregation period. In any other case, it can simply be encapsulated and forwarded (highlighted in green), via CtpForwardingEngine.

### 3.2.2 LinkEstimator Module

The data aggregation solution incorporates the Link Estimator module because it is responsible for determining the inbound and outbound quality of the one hop communication links (ETX). It is upon to this metric that the next-hop is picked and so the tree-based routing is built and sustained. LinkEstimator calculates ETX metric by collecting statistics from the number of beacons received and from the successfully transmitted data packets. From these statistics, LinkEstimator computes the inbound metric as the ratio between the total number of beacons sent by the neighbour over the fraction of received beacons. To gather the necessary statistics and compute ETX, LinkEstimator needs 2 bytes in the frame’s
header and a variable length field in the frame's tail for outgoing routing beacons. Upon reception of a beacon, the LinkEstimator extracts the information from both the header and tail and includes it in its table.

This table holds a list of identifiers of its neighbours, along with related information as the ETX or the amount of elapsed time since the reception of the last beacon. It is through these link estimations, produced by the LinkEstimator that the CtpRoutingEngine will choose the next-hop for transmission.

3.2.3 CtpRoutingEngine Module

CtpRoutingEngine was also incorporated in our data aggregation solution because it is responsible for sending and receiving beacons, as well as creating and updating the routing table that governs CTP tree-based routing operation. Every sensor node holds a list of neighbours, the routing table, from which is selected a parent (node used as next-hop). This table is filled using the information extracted from beacons transmitted by the LinkEstimator. Along with the identifier of neighbouring nodes, the routing table holds further information, as metrics indicating the “quality” of a node as a potential parent.

In the case of CTP, this metric is the ETX, which is communicated by a node to its neighbors through the beacon exchange. The ETX of a node is defined as the ETX of its parent plus the ETX of its link to its parent. For each of its neighbors the node sums one hop ETX with the ETX that the corresponding neighbours had declared in their routing beacons. The result of this sum is the metric which we call the multi-hop ETX. The multi-hop ETX of a neighbour quantifies the expected number of transmissions required to deliver a packet to a sink using that neighbour as a relay node. Upon this information, the CtpRoutingEngine selects the neighbour corresponding to the lowest multi-hop ETX as its parent. For instance, the multi-hop ETX of a sink node is zero.

CtpRoutingEngine uses a Trickle algorithm[45] to determine the frequency of the beacon transmission. The Trickle algorithm controls each node to progressively reduce the send rate of the beacons to save energy and bandwidth. The occurrence of specific events such as route discovery requests may however trigger a reset of the sending rate. This latter behaviour is called adaptive beaconing. It achieves both fast recovery and low cost, giving to CTP the possibility to adapt its topology according to environmental changes.

In addition, CtpRoutingEngine also uses data packets to maintain, probe, and improve link estimates and routing state. When the routing topology is working well and routing cost estimates are accurate, CTP slows its beaconing rate controlling the traffic timings. CtpRoutingEngine can also reset its beacon interval if it receives a packet with the P bit set. The P bit is is set when a node does not have a valid route. For instance, when a node boots, its routing table is empty, so it beacons with the P bit set. As can be seen in 4.2. Setting the P bit allows a node to “pull” advertisements from its neighbours, in order to quickly discover its local neighbourhood.

In other words, CtpRoutingEngine is important to the proposed solution because with the support of LinkEstimator it is responsible for maintaining the tree-based network topology, controlling the beacon transmission and defining the next-hop for packet transmission.
3.2.4 CtpForwardingEngine Module

The CtpForwardingEngine is the last component being reviewed but is by no means of less importance for the proposed architecture. CtpForwardingEngine will be responsible for forwarding aggregated data packets produced locally or received from neighbouring nodes. To do so, CtpForwardingEngine maintains two types of queues, a queue for each WSN application where each client can have a single outstanding packet. These client queues provide isolation, as a single client can not fill the send queue and starve others. CtpForwardingEngine hybrid queue contains both route through and locally generated traffic managed by a FIFO policy. This hybrid send queue possesses a size such that the send queue never rejects a packet. Additionally CtpForwardingEngine is also responsible for detecting and repairing routing loops as well as suppressing duplicated packets using a cache for the sent packets.

Within the developed solution, this component will be used to transmit, manage and control all aggregated information. The CtpForwardingEngine will send aggregated data messages according to the next-hop indicated by the CtpRoutingEngine, so the routing will be done upon link-layer addresses, Mesh-Under.

3.2.5 IPDispatch Module

This component is crucial for BLIP functionality. Through it flows all data packets received and sent by every node running BLIP, as is highlighted in Figure 3.6. BLIP implements a route-over approach because it uses IPv6 addresses to send/forward data packets. The routing of data messages is implemented at the IPRouting component while IPDispatch will provide message header information, which IPRouting processes according to the routing mechanism being used. BLIP can enable routing through the addition of routing headers, enabling multi-hop, or using IPv6 header address information.

To integrate the modules comprising this solution, some significant changes were made in this component. Namely, the creation of the interface Intercept to intercept all aggregated data messages received or produced locally. For that, an additional CTP header is added to every aggregation message. As can be seen in Figure 3.6, the Intercept interface will evaluate all received packets interacting with the AggregationEngine and with the CtpForwardingEngine accordingly. The Intercept interface will analyse every frame verifying if it possesses the additional header. If so, it will redirect the CTP header to CtpForwardingEngine to preserve and maintain the CTP tree-based routing and it will redirect the packet content to the AggregationEngine. AggregationEngine will perform data aggregation according to the aggregation identifier conveyed in the header.

All possible cases are shown in Figure 3.6. The LA6 data aggregation solution will have six possible cases. If the AggregationEngine is performing any kind of data aggregation, at some point the AggregationEngine will deliver the aggregated information to the WSN applications (highlighted by the blue arrow). That case is visible in Figure 3.6, and is the first case enumerated in the figure caption. On the contrary, the opposite situation can also happen, when the WSN application is producing some type of information that is redirected to IPDispatch for transmission but is intercepted and routed to AggregationEngine where is aggregated (highlighted by the green arrow). The previous two situations
were invocations where AggregationEngine and WSN Applications changed aggregated information. However it can also happen the other case where the information sent by the WSN Applications is susceptible of being aggregated, but according to AggregationEngine configurations, it does not interest to this sensor being immediately forwarded by the CtpForwardingEngine (highlighted by the orange arrow).

The other case of information not being forwarded is the case where a mote receives an aggregated message, or a message that is susceptible of being aggregated in the present sensor. In this case, the message is redirected (highlighted by the purple arrow) to the AggregationEngine where it is processed. Finally, when performing data aggregation the AggregationEngine send a message at the end of every aggregation interval. These aggregated messages are produced at the AggregationEngine and forwarded by the CtpForwardingEngine (highlighted by the light green arrow). As a matter of fact, the implementation of the Intercept interface, in IPDispatch is a critical strategic element within LA6 Data Aggregation Mechanism.

IPDispatch is also responsible for interacting with 6LoWPAN adaptation layer where the data packets are processed (IPv6 header compressing). Some significant changes were made to this module in order to enable the addition of the CTP data header containing the aggregation identifier upon which the AggregationEngine will decide whether or not to perform aggregation.

This modular design approach was developed to preserve the existent BLIP functionalities reducing to a minimum the modifications implemented in the IPDispatch core component of BLIP architecture.
3.2.6 AggregationEngine Module

This component will be invoked in IPDispatch by the Intercept interface. It is also a critical component because it manages the aggregation configuration and according to it decides whether to aggregate or to forward the messages intercepted. The data packets susceptible of being aggregated are different from the others because they possess an additional CTP Header that encapsulates the data packets susceptible of being aggregated. The resulting encapsulation can be seen in Figure 4.4. Thus the CTP header will encapsulate the message content, the compressed IPv6 header (OSI layer 3) and the UDP / TCP (OSI layer 4) header. LA6 architecture takes advantage of the code organization to perform data aggregation based on the information conveyed at the link layer level header. This approach will theoretically avoid unnecessary IPv6 decapsulation and data processing.

In fact, the AggregationEngine intends to be an independent module that can be used through a predefined interface, rather Intercept, independently of the circumstances of its invocation. This characteristic enables its invocation regardless of the routing approach. This modular approach allows the use of this component to perform aggregation even if CTP is not being used because it only needs an aggregation identifier to perform it.

Due to the AggregationEngine independence it will also be responsible for managing the aggregation configuration options, as well as additional statistics functionality. These functionalities shall be borne by AggregationEngine because these operations should be independent of the routing mechanism being used.
Chapter 4

6LoWPAN In-Network Data Aggregation Mechanism Implementation

4.1 6LoWPAN In-Network Data Aggregation Implementation

In order to achieve an implementation similar to BLIP solution some interfaces were developed linking BLIP components with the new modules being incorporated. Additionally, was used conditional compilation.

The implemented solution uses compilation flags to separate the existent BLIP functionalities, from the new functionalities added by the LA6 In-Network Data Aggregation solution. This conditional deployment allows the use of LA6 In-Network Data Aggregation as an add-on. Therefore, the implemented solution gives to the user the option to use or not the LA6 In-Network Data Aggregation mechanism.

The conditional compilation provides three different usage scenarios. The component based implementation approach together with the conditional compilation ensures that the existent BLIP functionalities remain operational despite of the modifications realized. Thus, the conditional compilation is used to include, to adapt and confine the invocations to CTP components and AggregationEngine.

These deployment scenarios provided by the LA6 In-Network Data Aggregation solution give the flexibility and helps the identification of new development opportunities.

One deployment scenario is the installation in a mote of a solution composed by the BLIP implementation and the additional AggregationEngine add-on. In this case BLIP system will run and interact with AggregationEngine, implementing in-network data processing through the interception of messages received in IPDispatch. This scenario, contributes for the possibility of using other routing protocols with the AggregationEngine module. Nevertheless, the LA6 In-Network Data Aggregation solution is composed by the AggregationEngine and by the CTP components CtpRoutingEngine, CtpForwardingEngine and LinkEstimator. This scenario is also implemented using conditional compilation. Therefore, the
implemented solution is dependent of CTP components. With CTP components active each deployed 6LoWPAN node is capable of forwarding CTP beacons according to the Trickle algorithm determined by CtpRoutingEngine. Additionally, there is CtpForwardingEngine that forwards data packets. In fact, the LA6 In-Network Data Aggregation solution can be deployed in two modes, however despite of that, in both implementations the BLIP functionalities remain intact.

After ensuring that CTP key elements are loaded, these must be linked with the existent BLIP core components, namely IPDispatch. To do so and to ensure total interoperability, the interfaces CtpRoutingEngine and CtpForwardingExtension were re-implemented along with the adaptation of the RootControl, CtpInfo, CtpCongestion, UnicastNameFreeRouting and Intercept interfaces. All these interfaces will only be invoked inside of its conditional environment assuring independently of the deployment, the existent BLIP solution would operate flawlessly.

The LA6 In-Network Data Aggregation implementation approach can, however, introduce some drawbacks or limitations. Firstly, it needs much more memory. Secondly, to operate properly, the LA6 solution must use a dual stack. This is a consequence of using CTP as our tree-based routing mechanism to perform Data Aggregation.

The other deployment scenario would be the development of a data aggregation mechanism without CTP routing mechanism, using only the AggregationEngine and Intercept Interface to achieve that. This scenario is already implemented and is used, as proof of concept, by the data aggregation configuration process.

4.1.1 Development Platform

During the In-Network Data Aggregation Mechanism reference implementation, Crossbow’s MICAz nodes [3] were used as the development platform, a well-known and popular mote in a major variety of applications.

Alongside, Crossbow’s Iris motes [2] were also used because they possess 8kbytes of RAM (twice the memory of MICAz’s), which enables the use of some debug mechanisms and allows a more flexible and stable development process. Despite also using Iris, the objective was the deployment of the LA6 In-Network Data Aggregation Mechanism using MICAz. This requirement caused numerous obstacles to the development of a final solution, namely due to MICAz RAM limitations.

<table>
<thead>
<tr>
<th>Radio</th>
<th>Memory</th>
<th>Interfaces</th>
<th>Power Consumption</th>
</tr>
</thead>
<tbody>
<tr>
<td>Band Mod.</td>
<td>Maximum Data Rate</td>
<td>Maximum Range</td>
<td>Program RAM</td>
</tr>
<tr>
<td>2.4 GHz</td>
<td>220 kbps</td>
<td>30 to 100 m</td>
<td>128 kbyte</td>
</tr>
</tbody>
</table>

Table 4.1: MICAz device main characteristics.

MICAz have very small dimensions, being battery powered nodes with an ATmega128L processor, a CC2420 radio [1], and a 512 kbyte flash memory on board. The radio operates in the 2.4 GHz bandwidth and uses DSSS modulation, having this way an overall high resilience to ambient noise and
radio interference.

The main processor provides several interfaces for sensor sampling and communication with other devices at very low power expense. For a detailed review of the device characteristics see Table 4.1. Additionally, three essential support devices were used: MIBs 520 USB - Gateway. These devices enabled the Programming of Crossbow’s MICAz and Iris hardware platforms.

4.1.2 RoutingEngine

The routing mechanism implemented is based in CTP tree-based routing. The developed solution incorporated CtpRoutingEngine and LinkEstimator in BLIP. This approach separates the IPv6 traffic from the data aggregated traffic because the data aggregated traffic is sent according to a tree-based routing implemented by CtpRoutingEngine while the traditional IPv6 traffic will flow according to the existent routing implementation of IPRouting.

To link the CtpRoutingEngine with IPDispatch, the CtpRoutingEngine interface was developed. This interface is composed by the following procedures:

- **command error t start()**
  
  The procedure start() is invoked in IPDispatch when the sensor mote boots. This procedure was implemented to start the execution of the CtpRoutingEngine when the mote running BLIP system starts. Its invocation starts the timer RouteTimer, implemented in CtpRoutingEngine. This timer is responsible for the periodic execution of the CTP task that will update the CTP routing table maintained by CtpRoutingEngine. This task evaluates the neighbour link quality and according to it, starts a BeaconTimer that implements a Trickle algorithm for sending CTP beacons. For a detailed analysis of CTP beacon see Figure 4.2. Upon the beacon reception, the LinkEstimator is invoked to calculate the ETX. Through the execution of the start procedure, a sequence of events is triggered from which results the definition of the next best hop for transmission. The sequence of invocations resulting from the execution of start procedure is visible in figure 4.1.

![Figure 4.1: CTP Routing Mechanism operation flow chart.](image)

- **command error t stop()**
  
  On the other hand the procedure stop() was implemented to do the opposite.
• command error_t getNextHop(struct ip6_hdr *hdr, struct ip6_route *sh, ieee154_saddr_t prev_hop, send_policy_t *ret)

Despite the routing approach previously presented and used by the implemented In-Network Data Aggregation Mechanism, some other routing approaches were tested during the development process. In this context the getNextHop procedure was developed that was implemented in the IPRouting component. This routing approach follows the existent route-over solutions, such as HYDRO. However, during the implementation process, the decision of using a mesh-under routing solution was made, and as a consequence, this procedure is left unused. Even so, it is mentioned as a proof of concept.

After creating the previous interfaces, some modifications were demanded to maintain the interfaces UnicastNameFreeRouting, CtpInfo and CtpCongestion operational. Regarding the UnicastNameFreeRouting interface, in the traditional CTP it is responsible for passing the best next-hop (parent) address from CtpRoutingEngine to CtpForwardingEngine using the procedure nextHop(). Therefore, in the present implementation it will keep its functionality, but it was adapted to convert the AM addresses given by CtpRoutingEngine to short IEEE154 identifiers. Those addresses will be used by the ForwardingEngine implementing a mesh-under routing approach. CtpInfo and CtpCongestion interfaces are important in the CTP header manipulation. Since the CTP data packet was modified to be incorporated in IEEE154 frames, the procedures of this interface were reimplemented in order to keep its functionality.

![Figure 4.2: Beacon Message format.](image)

### 4.1.3 ForwardingEngine

Considering that the routing approach used in the LA6 solution shall implement a forwarding tool for sending and forwarding messages using link layer IEEE154 addresses given by the RoutingEngine, the ForwardingEngine implementation is based in CtpForwardingEngine from CTP. However, since the transmission stack used is the IEEE154, some modifications were made.

The implemented procedures use a data structure inherited from IPDispatch. The CTP implementation of CtpForwardingEngine used queues containing structures of this type fe_queue_entry. During the implementation process those structures were replaced by the structures used in IPDispatch, send_entry_t. This modification gives more flexibility, transparency, and ensures less memory usage. This modification is essential because send_entry_t structure contains some additional attributes oriented to IPv6 paradigm, such as, the fragmentation flag that indicates if the packet is fragmented or not.
To incorporate CtpForwardingEngine in BLIP, the interface CtpForwardingExtension was developed and, according to the necessity, were created the following procedures:

- **command error_t send(send_entry_t * agg_msg, uint8_t aggID, uint8_t port)**
  
  This procedure is used for sending aggregated data messages produced locally. It is invoked in the IPDispatch component to send aggregated data messages according to the best next hop determined by the RoutingEngine. Thus, the sending process is done according to the IEEE154 link layer address. To do so, the transmission AM interface was replaced by the IEEE154Send. In this case, the implemented RoutingEngine will only use the IEEE154Send interface because the IEEE154Frames will be received in IPDispatch. This decision results from the necessity of having a unique local where all data frames converge, in order to intercept them.

  The send procedure will be invoked from the Intercept interface and will pass the aggregated message to the ForwardingEngine through a pointer to a send_entry_t structure, as well as the aggregation identifier and the WSN application port. The aggregation identifier and the WSN application port will be used in the CTP header construction and in the queue system. So, with this information the send procedure will queue the data structure in the queue allocated to that WSN application, according to its port. This is a new feature of the implementation that will be reviewed in the next chapter.

- **command error_t forward(send_entry_t * ctp_entry, struct ctp_data_header * ctp_hdr, uint8_t * fw_pos)**
  
  This procedure is used for forwarding aggregated messages that are not produced locally. Instead, they are intercepted in the IPDispatch, but will not be processed locally. Therefore, despite having the additional CTP header, these messages will be forwarded to the best next hop according to the RoutingEngine. In this case, the only operation made is the CTP header process and message forwarding.

![Diagram](image)

Figure 4.3: LA6 In-Network Data Aggregation interfaces implementation.

Regarding Figure 4.3 is becomes evident the crucial importance of the Intercept Interface. This original Intercept interface existed in the original solution of CTP. It was implemented in CtpForwardingEngine. The reimplementation of the Intercept interface is more complex due to the numerous options...
for redirecting the packet. It is patent in the previous figure. This interface links in IPDispatch the AggregationEngine, the ForwardingEngine and the IP interface, and according to an aggregation identifier each message is routed, to the appropriate LA6 In-Network Data Aggregation component. For instance, the 6LoWPAN ND messages and the normal IPv6 data traffic received at IPDispatch by the interface IEEE154Receive are unequivocally identified by an aggregation identifier equals to zero. In these cases, the Intercept interface only route this messages to the original BLIP functionalities implemented. This aggregation identifier can be defined at the AggregationEngine, after some aggregation configuration have been made, or it can be decasulated from the packet received at IPDispatch via IEEE154Receive interface. All the aggregation identifiers non-zero are associated to an aggregation function. Thus, aggregation identifiers are unique, and will be used to perform data aggregation accordingly.

Additionally, it shall be highlighted that in LA6 implementation the role of CtpForwardingEngine is different from the role and the context of CTP. In CTP, this component was linked directly with WSN applications passing the received messages to each application. Therefore, in CTP this component had a key role. However, in this implementation CtpForwardingEngine was incorporated as an alternative add-on used to forward the data aggregation messages. Therefore, in the present implementation IPDispatch is the key element. Because of that, some functionalities of CtpForwardingEngine became inadequate and useless. This situation and the memory constraints lead to the removal of the interface CtpPacket, because it was no longer useful. In this context, it was also decided to move the implementation of the Intercept and RootControl interfaces to the IPDispatch component. This modification is crucial for the LA6 implementation. The Intercept interface links the AggregationEngine component that stores the aggregation configurations, based upon the aggregation decisions and the ForwardingEngine that sends the aggregated messages.

![Figure 4.4: LA6 data aggregation message format.](image)

So far, the BLIP implementation and the incorporation of CTP components have been reviewed. However, as a result of the adaptation of the AggregationEngine, inherited from CTP to a IEEE154 context, the CTP data packet suffered some modifications. The new CTP header is equal to the original, but is smaller, because it does not have the origin AM address of the data packet, as is visible in Figure 4.4. Furthermore, the information that is not in the CTP header is the origin address of the data packet, although in IEEE154 frames this information is already in the conveyed message. As such, this removal was advantageous, reducing the overhead in 2 bytes.
4.1.4 Queueing, Congestion Control and Duplicate Message Detection

CTP has its own queues, a congestion control and duplicate message detection mechanism. The LA6 Forwarding component keeps all those functionalities despite the modifications realized.

The CTP queuing system is composed by a per-client (WSN applications running on each mote) queue, and by an hybrid send queue. Regarding the implementation process, and despite the memory limitations, it was possible to maintain both queues. However, some conceptual over CTP could not be avoided. The CtpForwardingEngine Queue system used client identifiers to identify each WSN application running on the mote. In the LA6 solution, the ForwardingEngine lost its relation with the WSN applications, and as such, can not use a client identifier to each WSN application. To overcome this issue the LA6 Queueing system reimplemented the client queues using the application port as the identifier of the WSN application. Therefore, This adaptation will unequivocally identify the WSN application sending each data message.

As well as CtpForwardingEngine, the new AggregationEngine uses an hybrid send queue. In CTP implementation it has length C (number of WSN Application clients) plus F (size of forwarding buffer pool), but in LA6 implementation it was not possible to implement that for MICAz motes due to memory constraints. So theoretically, the implemented solution for MICAz does not guarantee that the aggregated packets are never rejected. For your acknowledgement this limitation only exists in MICAz motes, in Iris the queueing system is perfectly integrated and does not cause any problem.

Additionally, it shall be mentioned that the Queue (FIFO, First In Firt Out) and Pool structures used in CTP, IPDispatch and ForwardingEngine are the same and operate over two different stack implementations: AM and IEEE154. However, both stacks process messages to send in a round-robin fashion, keeping the fairness of the FIFO implementation of the up layers. Therefore, despite changing the stack below the CtpForwardingEngine, this modification does not affect the fairness of the new AggregationEngine, maintaining CtpForwardingEngine intact for this subject.

Regarding Congestion Control Mechanism, it is important to state that after the modifications realized in the CTPCongestion interface this functionality was preserved, because it was adapted to the new format of the CTP data packet, see Figure 4.4. Thus, the ForwardingEngine can detect traffic congestion, viewing the congested bit, C, or analysing the pull bit, P. Since, the analysis of these header fields is possible then the ForwardingEngine will be capable of adapting its transmit timers according to the congestion state or according to the link quality information piggyed-back in the new aggregated message.

Finally, the issue of duplicate messages transmission is properly addressed by the implementation made in ForwardingEngine. The proposed solution is based upon the CTP approach. It uses a cache to keep the last messages sent, and so, to verify if any of the aggregated messages received is a duplicate. If so, it is dropped. As stated previously, the aggregated messages convey an additional CTP header that contains a sequence number and THL. The ForwardingEngine implementation detects duplicate messages verifying these two values and the IEEE154 source address carried in the IEEE154 frame. It examines those values using the fw_pos pointer received in forward procedure of the CtpForwardingEngine. The new format of the CTP header leads to the total reimplementation of the existent interface...
4.1.5 Aggregation Engine

The AggregationEngine will only be accessible from the IPDispatch. Its invocation is done at the Inter-cept interface. This ensures that all received packets will, at some point, be analysed in order to verify if it is eligible of being aggregated or not. This evaluation is performed using the Aggregation interface called Aggregator. This interface is implemented by AggregationEngine and invoked in IPDispatch. It is composed by the following procedures:

- **command TSNMsg\_t\* aggregate(uint8\_t aggID, TSNMsg\_t \* data)**

  This procedure is critical. It is invoked by the IPDispatch Intercept interface and returning the data aggregation result. It is this procedure that decides if the intercepted information is aggregated locally or not. Its implementation received the aggregation identifier and the packet content. The aggregation identifier was carried in the additional header and identifies what type of data is conveyed and what aggregation function shall be applied to it, in case of aggregation. This procedure receives this information and verifies if the mote is configured for aggregating this aggregation identifier. If so, it processes the information locally, otherwise it will return a pre-determined value and forward the message, redirecting it to AggregationEngine.

  The aggregation identifier is a crucial element and identifies unequivocally the information conveyed, and the operation requested. However, this procedure is dependent of the configure procedure, because it is upon the aggregation configurations that the data aggregation is performed.

- **command error\_t receive(void * TSNMsg, struct ip6\_hdr * ip, uint8\_t aggID)**

  This procedure is used to deliver the aggregated information to the WSN applications, or to the sink node that receives all aggregated data. This procedure was moved from CtpForwardingEngine to the AggregationEngine because the LA6 In-Network Data Aggregation mechanism could not be dependent of the CTP component to deliver the aggregation process result to the WSN applications. In the implementation context CtpForwardingEngine functionalities were incorporated in BLIP as an add-on, losing the direct relation with the WSN applications.

  The receive command will only be invoked when delivering the aggregated content to the respective WSN application, or when the sensor is the tree root. The interface RootControl reimplemented in IPDispatch informs if the sensor is the root node. If so, all aggregated messages will be delivered to the sink node through this procedure.

- **command error\_t configure(unpack\_info\_t u\_info, struct ip6\_hdr * ip, uint8\_t sensorMsg\_type, uint16\_t sensorMsg\_data)**

  Aiming to implement control and configuration functionality for the LA6 In-Network Data Aggregation features the present procedure was developed. It was implemented in AggregationEngine because the aggregation status and configurations are kept by this component. With it, it's possible to configure a sensor to start an aggregation process for some aggregation identifier. It is achieved
transmitting a configuration message from the LA6 Web Service. These configuration messages will contain the aggregation interval and the aggregation identifier. The configuration message is always sent to a specific sensor. These configuration messages can also be used to configure the statistics. In these cases, some special aggregation identifiers are used. Whenever it is intended to stop an aggregation process, configuration message is sent with the aggregation identifier and the aggregation interval equals to zero.

All these configuration messages will be received and processed by the configure procedure. Upon the reception of these messages, an aggregation timer will be triggered for that specific aggregation identifier with the aggregation delay defined according to the aggregation interval conveyed in the configuration message.

The configuration functionality is independent of the routing mechanism being used, because the configuration messages are routed according to BLIP default routing mechanism.

- command void updateStats(uint8_t OPERATOR)

This latter procedure was implemented to create statistics for the number of messages sent, and as such, calculate the burden of the overhead. This functionality is also not dependent of the routing mechanism deployed because it uses the default BLIP routing mechanism.

This procedure was implemented to give statistics for all data traffic, generated locally or forwarded. The statistics include the normal 6LoWPAN traffic, the control and management traffic and the aggregated messages. It ensures monitoring in real time and it can also be configured remotely using the LA6 Web Service.

The modular implementation created proved that the AggregationEngine could operate independently of the routing mechanism being used. This reality is shown when the AggregationEngine implements configuration and statistics functionality and does not need the CTP tree-based routing to achieve that.

### 4.1.6 IPDispatch

Concerning the existent BLIP implementation shall be stated that the changes realized in BLIP implementation are minimal. In order to integrate the data aggregation mechanism was re-implemented in IPDispatch the Intercept interface, the RootControl interface. These interfaces were migrated to IPDispatch component due to its central role in BLIP and consequently in LA6 Data Aggregation Mechanism.

The Intercept interface implementation possesses one single procedure that is invoked every-time a message is intercepted. That procedure is forward. This procedure was implemented to forward all messages that were not aggregated or not delivered to the sink node. The messages not susceptible of being aggregated are the messages that do not convey the additional CTP header. Those messages will pass and will not suffer any processing. This condition implemented in Intercept interface will permit the normal operation of the existent BLIP implementation.
• default event bool Intercept.forward(uint8_t agglID)(send_entry_t * ctp_entry, unpack_info_t u_info, struct ip6_hdr * ip, struct ctp_data_header * ctp_hdr, uint8_t * fw_pos)

On the other hand the data messages conveying the additional CTP header will be intercepted and redirected to the appropriate component to be aggregated of forwarded, AggregationEngine and ForwardingEngine respectively.

Regarding the forward procedure implementation there are visible the arguments used to execute that decisions. The argument struct ctp_data_header and unpack_info are obtained after IEEE154 frame being processed by the 6LoWPAN adaptation layer that underlies BLIP implementation, namely IPDispatch.

At IEEE154 frame reception the aggregation identifier is analysed using fw_pos pointer. Fw_pos points directly for the aggregation identifier conveyed in the frame. If the aggregation identifier is susceptible of being aggregate the decapsulation process is performed and the resulting ctp_data_header and unpack_info structures will be redirected to the AggregationEngine. The ctp_data_header will be analysed and if the aggregation identifier transported by the ctp_data_header indicates that this message must be aggregated, the unpack_info will be delivered for data aggregation and IPv6 (struct ip6_hdr) and UDP header information will be stored to be used later.

On the contrary if ctp_data_header is not conveyed, the resulting unpack_info will continue its normal processing path enabling the normal BLIP operation. In case of not susceptible for aggregation the data message, message_t, will be allocated in send_entry_t without IEEE154 Frame decapsulation process and forwarded.

In every case the ctp_data_header information will be obtained and processed by the Forwarding and RoutingEngine. This information is important to maintain the tree-based routing protocol.

IPDispatch component is linked with the 6LoWPAN adaptation layer implementation through the procedures getNextFrag and unpackHeaders. These procedures are used for encapsulation and decapsulation respectively. Through these procedures BLIP enable the 6LoWPAN adaptation layer and its functionalities, such as, header compression.

• getNextFrag(msg, &progress, call Packet.getPayload(outgoing, call Packet.maxPayloadLength()), call Packet.maxPayloadLength(), ctp_hdr)

• unpackHeaders(lowmsg, &u_info, unpack_buf, ctp_hdr, LIB6LOWPAN_MAX_LEN)

With the LA6 Data Aggregation solution this procedures will keep its function, but with the additional functionality. They will encapsulate and decapsulate the ctp_data_header as well. The implementation of this functionality does not interfere with the normal IPv6 traffic because if ctp_data_header is not conveyed it will return an empty structure with the aggregation identifier with value zero.

Concerning IPDispatch, despite of using the aggregation identifier to separate IPv6 traffic from the aggregated messages it does not in any circumstance changed the essential interfaces used in BLIP implementations. For instance UDP and IP interfaces. These interfaces are responsible for connecting layer 3, layer 4 and the application level implementation; they represent the OSI model in BLIP and were preserved intact in the resulting implementation.
4.2 Implementation Considerations

The effort of implementation was not only realized for the purpose of demonstrating that data aggregation is conceivable in a 6LoWPAN paradigm, but also to reaffirm that it is essential for 6LoWPAN. This reference implementation was developed for the TinyOS platform and, thus, benefits from its modular component based architecture. As was stated previously the LA6 Data Aggregation solution inherit the modular component based architecture in order to offer a flexible approach. Under these circumstances, each of LA6 modules was cleanly mapped into an individual TinyOS component that could be easily included or excluded from the compiled binary through the simple manipulation of the MakeFile. Using this approach, it easy to measure how much program and data memory is used by each feature. This implementation also gives to the user the possibility of decide what routing mechanism is used enabling different data aggregation solutions, and new development possibilities.

The presented implementation resulted from numerous attempts and essays. Although the final results is fairly aligned with the conclusions from section 2.4.6.

The LA6 Data Aggregation solution uses CTP as its routing protocol. It takes advantage of the de facto data collection protocol of the traditional WSN using its converge-cast routing paradigm to perform data aggregation in every hop of the tree. The LA6 Data Aggregation solution keeps all CTP functionalities and performs routing upon layer two, IEEE154 addresses. This is an tremendous advantage because it separates the normal IPv6 traffic from the aggregated traffic. Therefore, the LA6 implementation maintains the point-to-point communication paradigm of IPv6 and enables the use of an additional aggregation framework as an add-on whenever it’s necessary according to user configuration.

The implementation of LA6 Data Aggregation solution was extremely difficult due to hardware limitations of memory. The debug mechanisms available were not reliable and in many circumstances could not be used.

However, LA6 data aggregation solution was implemented and positions itself as a possible major contribution the 6LoWPAN viability, as is perceptible in section 6.
Chapter 5

6LoWPAN Off-Network Data Processing Mechanism

5.1 LA6 Off-Network Data Processing Mechanism Architecture

Despite the focus of this thesis being the development of an in-network data processing mechanism, in order to enable data aggregation within a 6LoWPAN, a deployment infrastructure was designed to accommodate and gather the results obtained. Nevertheless, the work being developed also creates conditions to analyse and confront the results obtained with the in-network data processing mechanism and the data processing performed outside of 6LoWPAN, i.e. off-network data aggregation.

![Figure 5.1: 6LoWPAN deployment infrastructure.](image)

Looking at Figure 5.1, it can be seen that the infrastructure architecture is composed by a Database Server running SQL Server and a WebServer upon which the developed Web Application providing the 6LoWPAN status is installed. Using the LA6 Web Application, the researcher can analyse the results obtained from the two data processing approaches and may also configure each 6LoWPAN node. Additionally a Bifferboard equipped with an application that processes the messages exchanged between each 6LoWPAN node and the LA6 Web Application installed in the server composes the infrastructure architecture. This device was chosen since it will offer more flexibility in case of migration to real environment scenarios. Finally, for the front-end of the 6LoWPAN, an IPBaseStation (MIB 520 USB Gateway) was deployed where all the messages received from the 6LoWPAN nodes will be gathered.
5.2 LA6 Off-Network Data Processing Mechanism Implementation

The LA6 Off-Network Data processing Mechanism was implemented using a typical 6LoWPAN, later deployed in Tagus-SensorNet, see Figure ref:testbed. In every 6LoWPAN node a data gather application was installed, which senses the environment, collecting temperature and light values.

Unlike the approach used in the LA6 In-Network Data Processing solution analysed in sections 3 and 4, this approach does not process or perform any operation over the data sensed in the 6LoWPAN inside the network. In other words, all sensed information by each sensor is sent hop by hop to the LA6 Application Server where it will be stored and processed. All the data processing will be done in the LA6 Application Server.

Between the 6LoWPAN and the application server, BifferBoard and 6LoWPAN sink nodes were deployed. Together these two devices enable the conversion of the IPv6 messages received from the 6LoWPAN to IPv4 allowing its transmission to the LA6 Application Server.

For this intermediate component, the BifferBoard was chosen due to its small size, offering good flexibility in the deployment process. It has a 486 SX processor with 8 MB ROM and 32 MB DRAM, offering a USB 2.0 and Ethernet ports that are used to connect the MIB 520 CA (where the sink node is installed) and the application server via Ethernet cable.

The network at IST Taguspark Campus uses IPv4 addresses. Therefore, to transmit the information obtained from the 6LoWPAN to the LA6 Application Server, an application already developed in previous projects had to be adapted. Some modifications were made in order to accommodate the transmission of configuration and statistic messages.

The LA6 Application Server was developed in order to provide a user interface for configuration, viewing, and data analysis. The LA6 Application Server was developed in JAVA using Google Web Toolkit (GWT). GWT is a development tool-kit for the development of complex browser-based applications such as the one provided by the LA6 Web Application. Due to the flexibility and ease of access these technologies were used in the development of this work.

The LA6 Application is divided in two modules, the one responsible for handling and managing database storage of the information received from the 6LoWPAN, and the module responsible for providing the web browser user interface and the services offered in it.

Reviewing the implementation of the web browser based LA6 Web Application, see Figure X, it is important to mention the three main services provided by it.

The LA6 Web Application shows, in real time, the 6LoWPAN topology according to the information gathered from the 6LoWPAN and stored in the database. This information is retrieved from the database using JavaScript Object Notation (JSON) and presented in real time. The graphics created give the user network topology information, including the IPv6 address of the node, the information of what in-network data processing operation each node is performing, and what information each node is sensing at that moment, along with the respective status of the each node.
The other implemented functionality offers the user the interface for performing 6LoWPAN node configuration. This functionality was developed independently of the LA6 In-Network Data Processing Mechanism since 6LoWPAN node configuration independent of the existent routing mechanisms deployed is desired. Therefore, the user through the web interface controls the configuration process.

There, the user can configure each node to perform data aggregation, data processing or simply to disable all In-Network Data Processing Mechanisms deployed. This was implemented creating two comboBox in the web interface where the user can pick which node he intends to configure, choosing the aggregation function and the aggregation interval. When introduced, this information is encapsulated in a TSNMsg and sent to the bifferboard through socket communication. At the Bifferboard, the LA6 Application installed will translate the message and send it to the respective sensor node according to the 6LoWPAN address introduced in the LA6 Web Service interface.

Finally, the last main functionality is the possibility of generating line charts with the information obtained and already processed by the developed LA6 In-Network Data Processing Mechanism or using the non-processed information gathered with the other solution.

Analysing the Off-Network Data processing Mechanism in more detail, the implemented solution performs data processing off-network. In other words, the 6LoWPAN nodes will sense the environment and send the gathered data to the SQL Server database without performing any aggregation function. All information will be stored in the database and processed off-line using SQL queries for calculating the Average values, the maximum value, and the Min value. To perform those operations a SQL database data model was developed for storing and processing data.
Figure 5.3: Database Entity Model.
Chapter 6

Tests and Evaluation

To assess the LA6 In-Network Data Aggregation Mechanism effectiveness, a test application was developed to perform data gathering in a real environment. In this context a test-bed was built to reproduce real-life WSN conditions.

6LoWPAN deployments are, by nature, a complex operation because multiple aspects, shall be taking into account. Additionally, to determine the real impact of test-bed system was determined the real energy usage under normal operating conditions. The sensor operation is determined according to the following states: radio Active - Transmission, Active - Reception, Idle and Sleep. The power consumption values for each state are shown in table 6.1. These consumption values for radio transceiver CC2420 used by MICAz were obtained from [10]. The power consumption values achieved were calculated disregarding the MAC layer used and the route variations (static routing paths were established) because these conditions are similar to both scenarios under analysis. These decisions were made because, in real conditions, evaluating efficiency is extremely difficult.

<table>
<thead>
<tr>
<th>Radio</th>
<th>State</th>
<th>Power Consumption [mW]</th>
</tr>
</thead>
<tbody>
<tr>
<td>CC2420</td>
<td>Active - Transmission</td>
<td>52.2</td>
</tr>
<tr>
<td></td>
<td>Active - Reception</td>
<td>56.4</td>
</tr>
<tr>
<td></td>
<td>Idle</td>
<td>1.28</td>
</tr>
<tr>
<td></td>
<td>Sleep</td>
<td>6x10^-5</td>
</tr>
</tbody>
</table>

Table 6.1: CC2420 Radio transceiver consumption.

Under these conditions, a series of tests were realized in two different scenarios. Aiming to prove that LA6 In-Network Data Aggregation Mechanism can achieve significant energy savings if compared with the traditional 6LoWPAN deployment, achieved in the Off-Network Data Aggregation Mechanism scenario, the following test scenarios were built.

As stated in chapter 4.1.2 the LA6 routing is generally determined by the CtpRoutingEngine reimplementation. Thus the 6LoWPAN arrange itself according to a tree-based structure. To do so each sensor node picks the best next hop as parent node. On the contrary in both test scenarios the routes were set manually, in order to avoid network instability and unstable beacon transmission. This characteristic ensures a more reliable and stable aggregation results.
It shall also be mentioned that the data gathering application will sense the environment at every minute sending the messages through the 6LoWPAN hop-by-hop until the sink node. Every sensors will be configured for performing data aggregation with an aggregation interval equal to 5 minutes. The tests were realized during periods of 36 hours.

The deployment infrastructure has an interface to provide information regarding the node’s capabilities to the user through which he has the possibility of configuring the gathering of environmental conditions measurements, which will be drawn in graph, also provided on the user interface via the LA6 web service. A more detailed description can be seen in section chapter 5.

After realizing each test and gathered the statistics of messages transmitted and received the following equations were used to calculate the energy consumption of the LA6 In-Network Data Aggregation solution and the traditional 6LoWPAN.

\[
E_{tx}(n, R_c, P_a) = T_s P_s + \frac{n}{R_c}(P_{txE} + P_a)
\]

\[
E_{rx} = T_s P_s + \frac{n}{R_c}(P_{rxE} + P_a) + nE_{dBit}
\]

The energy consumption calculation were made according to the equation 6.1 (where \(T_s P_s\) stands for startup energy and \(P_{txE} + P_a\) for the power consumed by the RF amplifiers and the baseband circuits) refers to the energy consumed in the transmission of \(n\) bits at the nominal rate \(R\) and at the coding rate of \(R_{code}\) and Equation 6.2 (where \(nE_{dBit}\) stands for the decoding overhead) refers to the energy spent to receive the same packet. A more detailed explanation of this equations can be seen in [10].

6.1 6LoWPAN Tagus-SensorNet Test-bed

![Figure 6.1: 6LoWPAN node deployment in the testbed.](image)

In order to assess both the LA6 In-Network Data Aggregation effectiveness and efficiency a test application and a 6LoWPAN test-bed infrastructure was planned and deployed. This test infrastructure, or test-bed, is composed by 8 MICAz motes and an Iris Crossbow mote as a data-sink. The data sink identifier is 0x64, being located near chemical lab, room 1.69. The 6LoWPAN nodes are located on the
first floor of the IST Taguspark Campus, and its physical locations are shown in Figure 6.1, highlighted in red.

The test-bed visible in Figure 6.1 establishes a chain 6LoWPAN communication architecture. This organization will favour the LA6 In-Network Data Aggregation because it will allow data aggregation in every hop of the chain.

### 6.2 6LoWPAN Tree-Based Network Architecture

In order to produce meaningful results from the test simulations, a test application was developed to establish connections and gather data using the LA6 In-Network Data Aggregation Mechanism under the above stated scenarios. In this context a statistical functionality was also developed, in order to measure certain specific metrics, such as the overhead messages and radio messages transmitted and received. This performance monitor module periodically dumps and resets its statistical values every 5 minutes, thus providing a clear state of the 6LoWPAN.

![Figure 6.2: 6LoWPAN node deployment in tree-based node organization scenario.](image)

Reviewing the test-bed scenario of Figure 6.2 a tree-base 6LoWPAN communication topology can be seen. In this situation the tree-base architecture is favoured enabling data-aggregation of the information received from multiple nodes. This converge-cast topology will contribute for more reliable information, with a lower aggregation delay.
6.3 Test Scenarios

6.3.1 Energetic gain of the LA6 In-Network Data Aggregation Mechanism

Overhead Analysis

Regarding the overhead introduced by both approaches, it shall be mentioned that LA6 In-Network Data Aggregation mechanism is substantially less efficient than BLIP default implementation. Although, the LA6 routing mechanism introduces significant overhead caused by the trickle timer for transmitting data packets. The RoutingEngine when boots bursts a significant amount of beacons in order to announce itself and to receive answers from the neighbour nodes. The beacons transmitted will be used by LinkEstimator component to calculate the best next hop according to the link estimation metric, ETX.

![Figure 6.3: Overhead number of messages comparison between LA6 In-Network Processing Mechanism and traditional 6LoWPAN.](image)

In Figure 6.3 is visible the number of beacons sent by every nodes during 36 hours of test. Regarding the results obtained is visible that the majority of beacons were transmitted in the initial hours of testing. This happens because initially the nodes are trading link estimation information between each other establishing the CTP routing topology.

The energy cost of this behaviour is compounded by the results obtained with the LinkEstimator footer that LinkEstimator adds to the beacon message. Without this footer the CTP beacon has a size of approximately eight bytes, however, due to this footer the CTP beacon’s size can vary according to the link quality rounds 25 bytes. The direct consequence of this issue is visible in Figure 6.4 where, in terms of control traffic overhead, CTP consumes more energy, because it transmits more information.

Resuming, the LA6 control traffic rate is high at network start-up as probes and discovers the topology, but decreases and stabilizes over time.

The 6LoWPAN control traffic is reduced because it is based on 6LoWPAN ND. 6LoWPAN ND that sends its control messages at the beginning of operation. This is made to establish a neighbour discovery, maintain reachability information, configure default routes, and propagate configuration parameters. After the communication links are established, under normal conditions, almost no traffic control is trans-
Concluding, in terms of overhead traffic the LA6 In-Network Data Aggregation is worse than the existent 6LoWPAN implementation. The LA6 In-Network Data Aggregation underlying tree-based routing protocol causes this. To enable a converge-cast topology more messages control are necessary if compared with the traditional 6LoWPAN implementation that does not implement any concrete routing strategy.

**6LoWPAN Tree-Based Network Architecture**

This test scenario aims to start presenting the benefits of LA6 In-network Data Aggregation. This 6LoWPAN architecture does not favour perfectly the data aggregation process, however it fits in the tree-based routing approach implemented by LA6 In-Network Data Aggregation Mechanism, inherited from CTP.

As is visible in Figure 6.5, the number of 6LoWPAN messages sent with the LA6 In-Network Data Aggregation Mechanism is relatively low if compared to the number of 6LoWPAN data messages sent.
by the traditional 6LoWPAN approach.

This was caused by the fact that all the information produced locally by the nodes are sent through the 6LoWPAN towards the sink using a multi-hop communication paradigm. This situation shows how inefficient the 6LoWPAN communication paradigm can be. The data message sent from each 6LoWPAN node will be transmitted and retransmitted in every hop until the sink. From this fact results the linear progression of the 6LoWPAN data messages sent over the time, as is visible in Figure 6.5.

![Figure 6.5: Linear progression of 6LoWPAN data messages sent over time.](image)

Regarding the 6LoWPAN data packet format and the information conveyed in each message it is clear that typically every 6LoWPAN data packet will contain a sensed value of temperature or light, that is normally one or two bytes.

The existent 6LoWPAN implementation sends significant more information of overhead than the useful two bytes sent a 6LoWPAN packet (address-based communication paradigm). This is visible in Figure 6.6.

On the other hand, with the new LA6 In-Network Data Aggregation mechanism implementation, will reduce the 6LoWPAN packets being transmitted at every aggregation interval to 1 packet sent from each node.

Thus, due to the aggregation interval, despite of producing sensing values every minute, each node will only send the aggregation value after 5 minutes. Due to the additional header the data aggregation message transmitted possesses an additional overhead of 6 bytes.

Even though, the mesh-under tree-based routing implementation of LA6 avoids the use of the additional 6LoWPAN multi-hop header used in the traditional 6LoWPANs. Thus, even with this additional aggregation header the LA6 In-Network Data Aggregation Mechanism can improve dramatically the energy savings of the 6LoWPAN, as is visible in Figure 6.7.

### 6LoWPAN Tagus-SensorNet Test-bed

The same experience were also realized in 6LoWPAN Tagus-SensorNet Test-bed. In this scenario the 6LoWPAN nodes are organized in chain. Considering that every node in the 6LoWPAN performs data
Figure 6.7: Energy consumption comparison between LA6 In-Network Data Aggregation and traditional 6LoWPAN.

aggregation this network architecture will represent an optimal scenario. However the chain architecture increases the aggregation delay. Due to the aggregation interval defined of 5 minutes, the transmission of an aggregated message will only be realized at every 5 minutes.

Figure 6.8: Number of messages transmitted comparison between LA6 In-Network Data Aggregation and traditional 6LoWPAN.

This is an important issue, because it leads us to the important topic of synchronization. To enable data aggregation along the chain, all nodes must be synchronized to avoid loosing any aggregated message. For an analysis of this topic regarding a WSN deployed in a scenario identical to the 6LoWPAN Test-bed please see [48]. The present 6LoWPAN deployment was configured according to it.

At every aggregation interval will be transmitted by all nodes 6LoWPAN 180 UDP data messages. The traditional 6LoWPAN data message size rounds 70 bytes without multi-hop header and 80 bytes with multi-hop additional header. On the other hand the aggregated data messages size has, in average, 72 bytes. In this context, the LA6 aggregation interval reduces to 8 the number of messages transmitted. The comparison between the traditional 6LoWPAN and the LA6 In-Network Data Aggregation is visible in Figures 6.8.
The energy consumption is a key aspect to the WSN. Since the devices are small battery powered, it is important to understand their lifetime. The figures 6.9 and 6.7 show that the energy consumption with aggregation is 20 times less expensive.

### 6.3.2 Test LA6 Adaptive Control Traffic

This test studies the behaviour of the network when a new node appears. In this situation, all nodes start transmitting beacons acknowledging its link quality to the new node. With the beacons transmitted every 6LoWPAN node will adapt itself picking the best next hop for transmission.

In this context, as is visible in Figure 6.10, there is a substantial increase of message transmission of the neighbour nodes. From this behaviour it is also possible to conclude that the integration of the adaptive beaconing transmission was achieved.

**Figure 6.9:** Energy consumption comparison between LA6 In-Network Processing Mechanism and traditional 6LoWPAN.

**Figure 6.10:** Number of beacons transmitted when detected a new node appears.
6.4 Discussion

The dramatic energy saving obtained with the introduction of LA6 In-Network Data Aggregation Mechanism shows the expectable results. The LA6 In-Network Data Aggregation Mechanism reduces the number of data messages transmitted over the 6LoWPAN. The developed solution brought to 6LoWPAN the multipoint-to-point transmission paradigm, using the tree-based routing mechanism inherited from CTP. However, this approach has its drawbacks, namely the overhead needed to maintain and preserve the tree-based routing upon is made the data aggregation.
Chapter 7

Conclusion and future work

7.1 Conclusion

The traditional WSNs are data-centric networks used mainly for sensing the environment and transmit collected information cooperatively using a multi-hop communication paradigm. The WSNs implement a multi-point to point network, where all nodes sense the environment and communicate the information until the sink node, that passes the information to a central repository.

The traditional WSN deployment scenarios use battery-powered low power embedded devices whose lifetime is limited and inversely proportional to the data transmissions realized. To overcome this limitation the radio transmission shall be reduced to maximum, in order to increase device lifetime. To achieve this, data aggregation mechanisms were developed that take advantage of the similarities of data content to aggregate all correlated information in one message, reducing the number of messages sent through the WSN. These data aggregation mechanisms are essential in WSNs.

Aiming to enable a global communication paradigm where everything has IP, and consequently is accessible from the world wide web, 6LoWPAN was developed. 6LoWPAN is an adaptation layer that intends to deploy the IPv6 address-based communication paradigm in WSNs. To do so, it will be installed on top of the IEEE 15.4 communication stack. Despite being an extremely difficult challenge, its implementation was not its main obstacle. The real obstacle to its deployment is the communication paradigm that underlies the IPv6 against its deployment scenario, the WSNs.

Interestingly enough, IPv6 implements an address-based point-to-point communication paradigm that collides with the point-to-multipoint data-centric WSN reality. This is the biggest challenge of 6LoWPAN, and from what was studied in the state of the art, nothing was made to overcome this obstacle.

The existent 6LoWPAN standard is evolving and adapting itself to some auxiliary tools, such as IPv6 header compression and 6LoWPAN Neighbour Discovery, aiming to avoid unnecessary point-to-point transmissions. However, none of the 6LoWPAN improvements were made for contributing to change its communication paradigm.

In this context, and after regarding the existent WSN implemented approaches for saving energy, it was concluded that the best solution for improving 6LoWPAN operation paradigm should be the devel-
opment of an In-Network Data Aggregation Mechanism, for 6LoWPANs. It is an obvious approach, but until now, not implemented.

Considering the existent state-of-the-art, until this work there was no in-network data processing mechanism solution designed or implemented for 6LoWPAN. Even though from the data aggregation mechanisms implemented for the traditional WSN, CTP was identified as the de facto data collection protocol. CTP is widely accepted by the research community.

As a result of the necessity of improving 6LoWPAN implementations this dissertation proposes the LA6 In-Network Data Aggregation Mechanism capable of creating opportunities to perform data processing within a 6LoWPAN. To do so, adaptations to the existent CTP tree-based routing mechanism were done to the 6LoWPAN BLIP implementation. Regarding the implementation process, the solution was implemented by phases, integrating the CTP components first. Through the integration of CTP components, its tree-based mesh-under routing protocol was incorporated. To take advantage of CTP an independent AggregationEngine component was implemented, responsible for performing data aggregation.

The modular implementation provides some flexibility that was used to implement data aggregation configuration and producing statistics. The LA6 solution not only implements data aggregation, but also creates the conditions to accommodate other in-network processing mechanisms.

Finally, considering the tests done over the implemented LA6 solution, it was concluded that CTP has much more overhead traffic than the default 6LoWPAN routing implemented in BLIP. Nevertheless, the savings obtained using the implemented LA6 data aggregation are significant and compensate the energy wasted transmitting the additional CTP header and CTP Beacons.

The advantages of using the LA6 In-Network Data Aggregation are substantial and will contribute directly for energetic 6LoWPAN node viability, and consequently to 6LoWPAN WSN lifetime.

7.2 Future Work

Since LA6 In-Network Data Processing Mechanism uses a modular architecture, it is, by nature, extensible. The LA6 solution was implemented in BLIP 1.0, however this 6LoWPAN implementation was already replaced by BLIP 2.0. BLIP 2.0 architecture is extremely similar to the BLIP 1.0 implementation. Thus, due to the modular architecture of the LA6 In-Network Data Aggregation it would not be an issue to incorporate the present solution into BLIP 2.0. This is clearly an goal to be achieved in a near future.

Despite the tests already realized, many more shall be performed over the implemented solution, in order to consolidate its functionality and assess CTP functionalities. The challenge is to perform equivalent tests to those made over the traditional CTP implementation. This is important because it will consolidate this solution as an important tool for 6LoWPAN.

The modular based architecture of LA6 In-Network Data Processing Mechanism, highlighted the possibility of using an AggregationEngine component in other possible 6LoWPAN In-Network data processing solutions. The implementation opportunities provided by LA6 architecture are an open window to create new developments, new experiments and tests. The idea is to take advantage of this com-
ponent to implement data aggregation using other routing protocol approaches, such as the route-over
solutions.

The actual implementation of the AggregationEngine only applies MAX, MIN, and AVG aggregation
functions. This was caused by the memory constraints imposed by the MICAz development platform.
Because of this, the goal is the implementation of only some aggregation functions.

Even though the LA6 reference implementation was developed solely for TinyOS, nothing prevents
alternative implementations from being developed for different platforms. Implementing this solution over
FREE RTOS or Contiki would be rather obvious possible new goals.

An open issue in the LA6 In-Network Data Processing Mechanism implementation is its dependency
of the stack AM. In the future work we intend to address this topic in order to avoid using stack AM.
Bibliography


