UNIVERSITÀ DEGLI STUDI DI PARMA

Dottorato di Ricerca in Tecnologie dell'Informazione

XXVII Ciclo

Connect Sensor Networks to the Internet of Things

Coordinatore:

Chiar.mo Prof. Marco Locatelli

Tutor:

Chiar.mo Prof. Gianluigi Ferrari

Dottorando: *Pietro Gonizzi*

Gennaio 2015

Contents

List of Figures

List of Tables

Preface

The Internet of Things (IoT) provides unparalleled means to connect the physical world with the digital world, enabling important applications such as Smart Infrastructures, Smart Parking, and Smart Toys. But existing systems are typically proprietary and tailored to one specific application and sacrifice interoperability for low power consumption. This hinders widespread adoption.

This PhD thesis presents the scientific outcomes of a three-year research activity on low power IP-based wireless sensor networks. Within the scope of the FP7 CALIPSO ("Connect All IP-based Smart Objects!") project, this thesis studies limitations and enhancements of the standard IETF/IPv6 framework, which includes the recent RPL ("Routing Protocol for Low Power and Lossy Networks"), 6LowPAN ("IPv6 over Low power Wireless Personal Area Networks"), and CoAP ("Constrained Application Protocol") protocols, with a deep focus on radio duty cycling. In particular, the thesis contributes to three layers of the low power IP stack: MAC, network, and application. It also investigates cross-layer research challenges related to security in IoT.

Each contribution has been implemented as open source software for Contiki OS, Europe's leading smart object operating system, and evaluated through simulations as well as experiments on target IoT-based applications and testbeds.

Chapter 1

Context and Motivations

1.1 The Internet of Things

The Internet of Things (IoT) is a novel paradigm which comprises end-to-end IP connectivity between smart objects. A smart object is a small micro-electronic device that normally consists of: (i) a sensor or actuator which interacts with the physical environment, for example by measuring its temperature; (ii) a small microprocessor which collects and processes information, and (iii) a communication device, typically a low-power radio, that provides connectivity.

The range of smart objects varies significantly within the application domain in which they are applied. For instance, in a smart city, smart objects may correspond to traffic lights or lampposts providing information on their statuses. In a smart home, a smart object could be any electrical appliance, e.g., a refrigerator, that needs to be controlled and accessed remotely. Cisco predicts that about 50 billion devices will be connected by 2020.

The true innovative power of smart objects comes from their communication and networking capabilities. So far, however, smart objects' networks have largely been isolated islands whose interconnection has been made difficult because of a number of proprietary solutions, usually optimized for one specific application, that have not been amenable to integration. In addition, the proliferation of different communica-

tion protocols and radio technologies has delayed such integration.

IoT is changing approach by pushing IP all the way into smart objects, thus providing an open and standard-based technology for an endless number of applications to come. The main reasons for using IP are scalability, universality, and interoperability with existing solutions. The IoT groups different technology areas and scientific disciplines, such as embedded systems, wireless sensor networks (WSNs), mobile computing, and computer networks. The concept of WSNs is similar to that of smart objects, and much of the development in smart objects has occurred in the community around WSNs. WSNs are composed of small nodes, equipped with a wireless communication device, that autonomously configure themselves into networks through which sensor readings can be transported. The focus of this thesis is mainly on WSNs.

The vision of interconnected IP-based WSNs is seen by international organizations and standardization bodies such as IPSO Alliance, IETF, and IEEE. In particular, the IETF is the international open standardization body in charge of specifying the IP protocol suite.

1.2 Research Challenges

The interconnection of WSNs to IoT faces several research challenges, both at the node-level internals of each smart object, such as power consumption and physical size, as well as at the network-level mechanisms deployed by the smart objects. These two groups are likely intertwined, since networking mechanisms impact the power consumption of smart objects.

The node-level challenges of smart objects primarily have to do with power consumption, physical size, and cost. Power consumption is the most critical factor with smart objects because they are often either battery-powered or use an external low-power energy source such as physical vibrations or low-power energy harvesting solutions. Physical size is important because the size and form factor determines the potential applications for a given smart object system (we show later in this thesis that certain applications, i.e., Smart Parking, require smart objects to be small).

The severe power consumption constraints have design implications for the hard-

ware, software, the network protocols, and an IoT-based network architecture. Low power consumption implies the deployment of tiny and low-cost hardware components, with limited computational capabilities, low processing speed, and small memory. Nevertheless, the software which runs on top of it must support Internet-like protocols; therefore, it must be efficient enough to be able to run within a severely resource-scarce environment. However, such software limitations, which are driven by the small-scale factor of smart objects, are in contrast with the very large-scale factor of smart objects' networks, both in terms of (i) the number of nodes potentially involved in the system and (ii) the number of data items generated by each node.

The network size impacts the design of routing protocols. Routing is the process by which the network determines what paths messages should take through the network. Since communication requires energy, the routing protocol must make well-informed choices when planning how messages are transported through the network. For a node to make a well-informed routing choice, it typically requires information about both the network as a whole and about the node's nearest neighbors. This information requires memory, but, as already discussed, each node has a limited amount of memory.

Smart objects' networks often run over unreliable communication media, which causes messages to be lost or corrupted during transmission. The lossy nature of smart objects' networks is an additional challenge for routing protocols. It also requires the network to efficiently distribute and store its data in multiple positions, in order to increase reliability and system robustness.

Given the large scale of smart object networks, network management becomes a critical challenge. Instead of manual fine-tuning of the network infrastructure by a system administrator, the network must be prepared to manage itself, with self-configuration mechanisms at the nodes.

Finally, the adoption of ultra low-power radio duty cycling techniques on smart objects is in contrast with the severe requirements of applications and also impacts the routing protocols.

1.3 The CALIPSO Project

IoT has the potential to enhance Europe's competitiveness and is an important driver for the development of an information based economy and society. Within this context, the IoT European Research Cluster (IERC) has promoted a wide range of research projects, related to IoT, in different application fields. The FP7 CALIPSO project (grant agreement 288879) has been one of them.

"Connect All IP-based Smart Objects!" (CALIPSO) builds Internet Protocol (IP) connected smart object networks, but with novel methods to attain very low power consumption, thereby providing both interoperability and long lifetimes. CALIPSO leaned on the significant body of work on sensor networks to integrate radio duty cycling and data-centric mechanisms into the IPv6 stack. The project worked at three layers: the network, the routing, and the application layer.

CALIPSO worked within the IETF/IPv6 framework, which includes the recent IETF RPL, 6LowPAN and CoAP protocols. The project adopted the Contiki operating system, Europe's leading open source smart object OS, as the target development environment for prototyping and experimental evaluation. It envisioned three applications to drive the work: Smart Infrastructures, Smart Cities, and Smart Toys, all of which need both standardized interfaces and extremely low power operation. Experimental validation and evaluation were key aspects of the project.

As for the application scenarios, CALIPSO targeted two use cases: Smart Parking and Smart Toys. Smart Toys are radio-enabled devices whose primary function is entertainment/gaming, but they may also provide additional functionalities, such as sensing/monitoring, education, and multimedia communication. They may appear in various forms depending on the target user group (kids, teenagers, adults) and may be equipped with a variety of sensors (microphone, camera, accelerometer), user interfaces (display/touch screen, buttons/keyboard, or none), and radios (IEEE 802.11 and its Low Power variant, Bluetooth, IEEE 802.15.4/Zigbee). They can be accessed, monitored, and acted on remotely. Smart Toys can be used in theme parks, homes, schools, hospitals for entertainment, learning, and therapeutic purposes.

The Smart Parking application relies on sensors installed in parking places and

an intelligent system guiding drivers to vacant parking spots in an efficient way. Not to open the road's tarmac too often, the wireless nodes need to exhibit longevity; this, in turn, requires duty cycling of the radios that causes major problems to routing protocols.

The project investigated the requirements of its use cases, which are indeed heterogeneous, and led to the design of a standard unified IoT architecture with optimized routing, MAC and application layers, thus enabling secure and scalable end-to-end applications. Each solution was released as open source software and evaluated on real testbeds and field trials. The research presented in this thesis has been funded by the CALIPSO project.

1.4 Objectives of the Thesis

This thesis studies some of the aforementioned research challenges related to IP connection of WSNs to the IoT, taking into account related IETF standards and with a key focus on radio duty cycling techniques. As stated above, critical factors such as power consumption, small physical size, and losses, impact all functional layers of the IoT-based architecture. Therefore, the thesis studies cross-layer solutions at all layers within a unified approach. The thesis contributes in the following areas:

- study of an IoT architecture for WSNs at all layers;
- optimization and enhancements of IETF standards for constrained environments;
- design of novel techniques for energy efficiency;
- evaluation and experimentation on real testbeds of the proposed solutions.

At the application layer, the thesis first investigates distribution data storage techniques and data retrieval mechanisms to increase storage capacity and data resilience of the system. With the information coming from the routing plane, where the standard IETF RPL is used, data is spread across the network in an efficient manner.

At the network layer, the thesis studies the RPL routing protocol in depth. An innovative routing metric for the minimization of the delay to the sink node of the RPL tree-like topology is proposed. The proposed metric computes the shortest delay to the sink using information about radio duty cycling from the MAC layer.

At the MAC layer, a novel MAC protocol, namely RAWMAC ("Routing Aware Wave-based MAC"), is proposed. RAWMAC operates in ultra low power sensor networks, with duty cycling below 1%. RAWMAC uses routing information to dynamically align the wake-up phases of the nodes to speed up data collection.

Finally, the energy consumption problem is investigated at the security layer. The focus is on authorization to access the resources of smart objects, which is a crucial aspect related to security in IoT. Two solutions for handling authorization are investigated; the first implements authorization schemes on smart objects; the second one delegates the authorization process to an external service. A comparison of the two solutions in terms of energy consumption and memory footprint is performed.

The structure of the thesis is the following. Chapter 2 investigates research challenges at the application layer, by presenting a novel mechanism for distributed data storage and retrieval. Chapter 3 studies the RPL routing protocol and proposes a delay-sensistive routing metric. Chapter 4 investigates research challenges at the MAC layer and proposes RAWMAC, a routing-aware MAC protocol for IoT. Chapter 5 investigates research challenges related to security.

Chapter 2

Application Layer: Distributed Data Storage and Retrieval

In the emerging field of the IoT, WSNs have a key role to play in sensing and collecting measures on the surrounding environment. In the deployment of large scale observation systems in remote areas, when there is not a permanent connection with the Internet, WSNs are calling for replication and distributed storage techniques that increase the amount of data stored within the WSN and reduce the probability of data loss.

This chapter discusses the data storage problem at the application layer. Data storage is affected by (i) the lossy nature of WSNs, which impacts system reliability, and (ii) the lack of memory of the sensors, which impacts the amount of data that can be stored in the WSN. In particular, the chapter proposes two low-complexity distributed data replication mechanisms to increase the resilience and the storage capacity of WSN-based distributed storage systems at large scale. The first is a greedy mechanism which leverages on the residual available memory space of neighbors to distribute and replicate sensing data in a hop-by-hop fashion. By proposing a simple, yet accurate, analytical modeling framework and an extensive simulation campaign, we complement the analysis through experimental results on the SensLab testbed. The second mechanism extends its predecessor by proposing a cross-layer solution

which uses routing information for an efficient data placement. In particular, we use the RPL routing protocol to prioritize storage at nodes closer to the sink. Finally, we treat the data retrieval problem in the last part of the chapter.

2.1 The Data Storage Problem in WSNs

In contrast to conventional network data storage, storing data in WSNs represents a challenge because of the limited power, memory, and communication bandwidth of WSNs. Nowadays, sensors have reached higher capabilities, in terms of processing speed and local storage space, than in the past years [1]. This makes them more attractive for in-network storage applications.

WSNs for IoT observation systems are typically composed by unattended nodes, that sense the surrounding environment, and a sink node, which acts as data collector and gateway towards the Internet. Communications between sensor nodes and the sink are typically not instantaneous, especially in isolated WSNs where the sink node is not always present. Moreover, when applications do not require real-time data collection, storing data units and sending aggregate data bursts can contribute to reduce the amount of radio transmissions, thus increasing the lifetime operation of the WSN. Efficient data retrieval can be carried out periodically through the use of a mobile node which wanders across the WSN and collects the data stored at the nodes. However, rare data retrieval can cause local memory overflow and, therefore, data loss. In order to avoid this, nodes can cooperate by storing the sensed data in a distributed way.

As WSNs tend to be left unattended for long periods, distributed data storage has to be robust also against node failures. In order to achieve this goal, an attractive approach consists in combining distributed storage with data replication, i.e., by distributing and storing multiple copies of the same data across the WSN. This redundancy exacerbates the communication overhead required to find proper “donor nodes” (i.e., nodes available to store data of other nodes), as well as the storage capacity of the system. Therefore, the design of efficient distributed storage algorithms with data replication requires to deal with trade-offs between storage capacity, system

robustness, energy consumption (network lifetime), communication efficiency.

2.2 Related Work

In the past years, various schemes to efficiently distribute and replicate data in WSNs have been proposed [2].

In WSN distributed storage schemes, nodes cooperate to efficiently distribute data across the WSN. There are two main approaches: *data-centric storage* and *fully distributed data storage*. The data-centric storage approach is described in [3, 4, 5]. Here, some distinguished storage nodes, e.g., determined by a hash function, are responsible for collecting a certain type of data. A load-balanced distributed storage approach is proposed in [6], according to which data are preferably stored in densely populated areas of the sensing field to minimize data loss. In [7], data are stored according to their spatial and temporal similarity, in order to reduce the overhead as well as the latency of a query request. Even if data-centric storage approaches are based on node cooperation, they are not fully distributed since specific nodes store all the contents generated by the others. A detailed survey on data-centric storage schemes is presented in [8].

In a fully distributed data storage approach, all nodes contribute equally to sensing and storing. All nodes try, first, to store the sensor readings locally and, then, delegate other nodes in the WSN to store newly collected data as soon as their local memories are full. A first significant contribution in this direction is Data Farms [9]. The authors propose a fully data distributed storage mechanism with periodic data retrieval. They derive a cost model to measure energy consumption and show how a careful selection of nodes offering storage, denoted as “donor nodes," optimizes the system capacity at the price of slightly higher transmission costs. They assume a network tree topology, where each sensor node knows the return path to a sink node, which periodically retrieves data. The energy consumption problem is studied also in [10], where data preservation in an isolated WSN is considered. In this context, an energy-efficient data distribution scheme is proposed, based on the dissemination of data from low energy nodes to high energy ones. However, only low energy nodes

generate contents.

An interesting approach for load balancing is proposed in EnviroStore [11]. The authors focus on in-network data redistribution when the remaining storage space of a sensor node exceeds a given threshold. They use a proactive mechanism, where each node maintains a local memory table containing the statuses of the memories of its neighbors. Furthermore, mobile nodes (called *mules*) are used to carry data from an overloaded area to an offloaded one, as well as to send the collected data to a sink node. The deployment of mobile mules is also addressed in [12], where an efficient distributed data storage mechanism for an isolated WSN with limited storage space is proposed. Data is opportunistically offloaded to mobile mules when the latter are in proximity. Moreover, data are assigned different priorities: high prioritized data are stored closer to areas where the mules will pass more frequently. The main limitation of the above studies consists of a lack of a comprehensive performance evaluation framework, which encompasses analysis, simulations, and experiments. This holistic approach is a key contribution of our work.

Data replication strategies have been proposed in the literature, mainly to overcome the problem of node failures. The goal of replication is to copy data at other nodes within the WSN to increase resilience. Authors in [13] propose ProFlex, a distributed data storage protocol for replicating data measurements from constrained nodes to more powerful nodes. The protocol benefits from the higher communication range of such nodes and uses the long link to improve data distribution and replication against the risk of node failures. In [14], a replicator node is selected according to some critical parameters such as connectivity, available storage and remaining energy of the node. However, a model for such selection is not given. In TinyDSM [15], a reactive replication approach is presented. Replicas are randomly distributed within a predefined replication range influenced by the specific replica number and density.

2.3 Replication-based Distributed Storage

In this section, we describe a low complexity (“greedy") distributed data replication mechanism to increase the resilience and storage capacity of a WSN against node

failure and local memory shortage. The mechanism extends the preliminar approach presented in [16] and complements it with a simple, yet accurate, analytical performance evaluation framework, and with an extensive simulation campaign performed with the Cooja network simulator [17]. In particular, the proposed framework allows to evaluate: the network storage capacity; the time required to reach this capacity and, consequently, to start dropping data; the system robustness, in terms of retrievable data in case of a failure of several neighboring nodes("bomb-like scenario"). The validity of the proposed framework is confirmed by simulations and realistic experimental data obtained with SensLab, studying the impact of several key parameters. To the best of our knowledge, this is the first contribution that integrates experimental results of SensLab with analytical and simulation-based counterparts.

2.3.1 Design Principles

We assume that the nodes of the WSN keep on collecting data (acquired with a given sensing rate). Periodically, data is retrieved from a sink and canceled from their memories. This periodic retrieval is instrumental to allow the use of limited onboard memories. Data retrieval consists in forwarding the collected sensed data of the WSN to a central base station for further processing. In this section, we do not focus on data retrieval, which is the subject of the last part of the chapter.

In order to prevent data losses due to nodes' failures or memory shortages, nodes cooperate in the following way. A data acquired by a node is stored in several nodes (possibly including the generating node). This consists in copying and distributing replicas of the same data to other nodes with some available memory.

Information about memory availability is periodically broadcasted, by each node, to all its neighbors. Conversely, each node keeps on updating a local memory table relative to the memory statuses of all detected neighbors. Upon reception of a memory status from a neighbor, a node updates the corresponding entry of its local memory table with the new information received.

The proposed replication-based distributed data storage is *greedy*: in order to create a replica of a stored data, a node selects, according to its neighbors' memory table, the "best" neighbor—the selection criterion will be specified in the following.

The selected neighbor becomes a *donor node*. If no donor node can be chosen and there is no available space in the local memory, then the acquired data is dropped. In the remainder of this subsection, we formalize this greedy algorithm.

Symbol	Description	Unit
N	Number of nodes	scalar
A	Surface of deployment area	m^2
d	Node's transmission range	m
$V_1^{(i)}$	Number of 1-hop neighbors	scalar
B_i	Node i's buffer size, $i \in \{1,\ldots,N\}$	scalar
$T_{\text{sens},i}$	Node i's sensing interval, $i \in \{1,\ldots,N\}$	s
$r_{\text{sens},i}$	Node i's sensing rate, $i \in \{1,\ldots,N\}$	s^{-1}
P_t	Common node transmit power	mW
T_{adv}	Period of memory advertisement (from each node)	s
R	Maximum number of replicas per sensing data unit	scalar
T	Period of data retrieval (from the sink)	s

Table 2.1: Main system parameters.

Table 2.2 lists the main parameters of the system. A WSN with N fixed nodes is deployed over a region with area A (dimension: [m^2]). The radio transmission range of the nodes is denoted as d (dimension: [m]). The number of direct (1-hop) neighbors of node i ($i \in \{1,\ldots,N\}$), i.e., nodes within the transmission range, is denoted as $V_1^{(i)}$. The i-th node has a finite local buffer of size B_i (dimension: [data units]) and sensing interval $T_{\text{sens},i}$ (dimension: [s]), whose corresponding sensing rate is $r_{\text{sens},i} = 1/T_{\text{sens},i}$ (dimension: [data units/s]). Each node broadcasts, without acknowledgement and every T_{adv} (dimension: [s]), its memory status to all 1-hop neighbors. Each memory status message contains the following values, relative to the sending node: (i) node ID; (ii) current available memory space; (iii) sensing rate; and (iv) a sequence number identifying the message. Each node maintains a local memory table which records the latest memory status received from neighbor nodes. The local memory table contains one entry per neighbor, with indication of its most recent available memory space

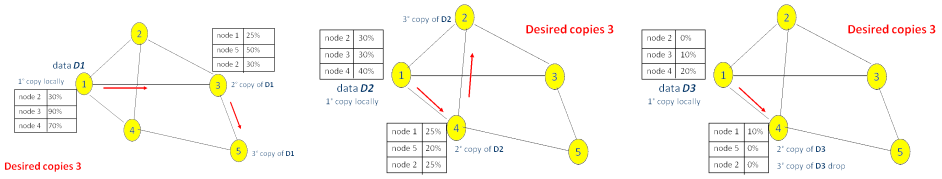
and the corresponding notification time. Upon reception of a memory advertisement from a neighbor, a node updates its memory table, using the sequence number field to discard multiple receptions or out-of-date advertisements. The memory table has a fixed size. In case of complete depletion of the memory table, as it may occur in dense networks (with a large number of 1-hop neighbors), a node stores only the "best" neighbors.

The greedy distributed storage mechanism consists in creating *at most R* copies of each data unit generated by a node and distribute them across the network, storing at most one copy per node. Each copy is referred to as *replica*. Let us focus on node $i \in \{1,\dots,N\}$. At time t, node i generates (upon sensing) a data unit. If node i has some available space in its memory, it stores a copy of the data unit locally, setting the number of remaining copies to $R-1$. Otherwise, if the local memory of node i is full, or multiple copies are to be stored, node i selects, from the memory table, a neighbor node to store a copy of the data unit. In particular, node i selects the neighbor node, called *donor*, with the largest available memory space and the most recent information. Denoting the neighbors of node i as $\{1,\dots,V_1^{(i)}\}$, the donor at time t, indicated as $D^{(i)}(t)$, is chosen according to the following heuristic rule:

$$D^{(i)}(t) = \underset{j\in\{1,\dots,V_1^{(i)}\}}{\operatorname{argmax}} \frac{B_j(t_j)}{t-t_j} \tag{2.1}$$

where $t_j < t$ denotes the time at which the available memory space $B_j(t_j)$ of node j was received by node i, with $B_j(t_j) \leq B_j$. If there is no suitable neighbor in the memory table (i.e., $B_j(t_j) = 0, \forall j \in \{1,\dots,V_1^{(i)}\}$), there is no possibility to distribute replicas of the data unit across the network. In this case, only the original data unit can be stored in the local memory of node i, provided that there is some available space at node i.

Upon reception of the copy, the donor node D: (*i*) stores the copy in its memory; and (*ii*) selects the next donor node among its neighbors, according to a modified version of (2.1). In particular, in order to avoid loops, previously selected donors (which already have a copy of the received data) are not enlisted among the candidate nodes. Therefore, the selection criterion for the choice of a donor for the r-th replica



(a) 2-hop path. (b) 1-hop path. (c) No suitable donor

Figure 2.1: Hop-by-hop replication in the case of $R = 3$ desired replicas, for several scenarios: (a) replicas propagate up to 2 hops from the source node; (b) replicas are stored at 1-hop; (c) the replication process stops at an intermediate donor node and the last (R-th) copy is dropped.

($r \in \{1,\dots,R\}$) is

$$D_r^{(i)}(t) = \underset{j \in \{1,\dots,V_1^{(i)}\} \setminus S^{(r-1)}}{\operatorname{argmax}} \frac{B_j(t_j)}{t - t_j} \tag{2.2}$$

where $S^{(r-1)}$ is the set of donor nodes for the previous $r-1$ copies. Upon selection of the donor, the r-th node thus sends it a copy, decrementing the number of required copies by 1. The replication process continues recursively until either the last (R-th) copy is stored or an intermediate (r^*-th, $r^* < R$) donor node cannot find any suitable next donor node. In the latter case, the final number of copies actually stored in the WSN is smaller than R.

In Figure 2.1, we show three illustrative scenarios with $R = 3$. In Figure 2.1(a), 3 copies of the data unit $D1$, generated by node 1, are stored (respectively in nodes 1, 3, and 5). In Figure 2.1(b), copies of the data unit $D2$ are stored (respectively in nodes 1, 4, and 2) — note that in this case replicas do not propagate beyond 1 hop from the generator node. In Figure 2.1(c), the replication process stops at node 4 and no suitable donor node can further be found. In this case, the last replica is not stored.

2.3.2 Analytical Performance Evaluation

Implementing distributed data storage with replication raises several trade-offs. Indeed, storing multiple copies of a single data reduces the amount of unique data that

can be stored in the WSN, whereas it improves the reliability against the risk of node failure and complete data loss. Also, the latter reliability goes against the lifetime operation of the WSN, as it entails more costly communication for the distribution of replicas across the WSN. In order to characterize these trade-offs, in the following we derive analytical bounds for key performance metrics of interest. The portfolio of metrics includes: the *network storage capacity*, interpreted as the amount of unique data that can be stored in the WSN; the *time to reach the storage capacity*, given by the time required by the WSN to fill the memories of all nodes; the *dropped data*, i.e., the amount of data lost because of local memory shortage; and the *system robustness*, given by the percentage of recoverable distinct data in the presence of a "bomb-like"failure event involving all nodes within a certain spatial range. An accurate analytical evaluation (also through the derivation of upper/lower bounds) of the system robustness, as it will be defined, is an open problem.

Network Storage Capacity

We derive analytical expressions for upper and lower bounds of the network storage capacity, considering the cases with and without replication.

We first consider the case *without* replication, when only the original copy is kept ($R = 1$). Given the number of nodes N, the buffer sizes $\{B_i\}$, and the sensing rates $\{r_{\text{sens},i}\}$, the network storage capacity C (dimension: [data units]) is simply given by

$$C = \sum_{i=1}^{N} B_i. \tag{2.3}$$

With replication (i.e. with $R > 1$), the system has a resulting storage capacity, denoted by C_{r}, which is upper-bounded by C. In this case, the capacity is reduced by the maximum number R of replicas and can be lower-bounded as follows:

$$C_{\text{r}} \geq \frac{C}{R}. \tag{2.4}$$

Obviously, $C_{\text{r}} = C$ when $R = 1$, i.e., only the original copy is kept and no replication is performed at all. On the other hand, the lower bound (2.4) can be actually reached only if R replicas of each generated data unit can be effectively stored across the

WSN. This can happen only if the storage spaces $\{B_i\}$ at the nodes are very large and/or the sensing rates $\{r_{\text{sens},i}\}$ are much lower than the retrieval rate $1/T$ of the sink.

Time to Reach Storage Capacity and Data Drop

Another key performance metric is the time required to reach the network storage capacity. This time is of interest for an operator when parameterising the period at which the sink has to retrieve the data from the WSN. In order to provide an analytical expression for this metric, we consider the three following cases.

In the case with *local* storage and *without* replication, the time required by the i-th node to fill its local buffer autonomously is $t_i = B_i/r_{\text{sens},i}$. Therefore, the time interval to reach the network storage capacity corresponds to the longest storage filling time across all nodes, i.e.,

$$t_{\text{cap}-1} = \max_{i\in\{1,\ldots,N\}} t_i. \tag{2.5}$$

Obviously, nodes which fill their buffers faster will drop newly sensed data because of local buffer overflow. Assuming that the sink retrieves the stored data when there is no available storage space left at any node in the network,[1] the total amount of dropped data can be expressed as

$$D_{\text{drop}-1} = \sum_{i=1,i\neq j}^{N} \left(t_{\text{cap}-1} - t_i\right) \cdot 1/T_{\text{sens,i}}. \tag{2.6}$$

Note that the more heterogeneous the times $\{t_i\}$, the larger the amount of dropped data. On the other hand, should all filling times be equal, i.e., $t_i = t_{\text{cap}-1}$, $\forall i$, it would follow that $D_{\text{drop}-1} = 0$, i.e., all nodes fill up their storage memories simultaneously.

In the case with *distributed* storage *without* replication, a performance benchmark can be obtained considering an *ideal* WSN where nodes can communicate with any other node, considering instantaneous transmissions. In this case, the WSN is equivalent to a single super-node with a storage capacity C equal to $\sum_{i=1}^{N} B_i$ and sensing

[1] This is a pessimistic assumption, as the sink might not wait till all nodes fill their buffers.

rate equal to $\sum_{i=1}^{N} r_{\text{sens},i}$. Thus, the time required to reach the storage capacity can be given the following expression:

$$t_{\text{cap-ideal}} = \frac{C}{\sum_{i=1}^{N} r_{\text{sens},i}} = \frac{C}{\sum_{i=1}^{N} 1/T_{\text{sens},i}}. \tag{2.7}$$

The amount of dropped data can then be expressed as follows:

$$D_{\text{drop-ideal}} = \begin{cases} 0 & t < t_{\text{cap-ideal}} \\ \left(t - t_{\text{cap-ideal}}\right) \cdot \sum_{i=1}^{N} r_i & t > t_{\text{cap-ideal}}. \end{cases} \tag{2.8}$$

In the case with *distributed* storage *with* replication (according to the proposed greedy mechanism), the time to reach the storage capacity can be lower-bounded using (2.7) after replacing C with C_{r}. Taking into account the lower bound (2.4), one thus obtains

$$t_{\text{cap-d}} = \frac{C_{\text{r}}}{\sum_{i=1}^{N} r_{\text{sens},i}} \geq \frac{C}{R\sum_{i=1}^{N} r_{\text{sens},i}} = \frac{\text{t}_{\text{cap-ideal}}}{R}. \tag{2.9}$$

In Figure 2.2, we present analytical results (using Matlab) relative to the network storage capacity. They refer to a scenario with $N = 10$ nodes, each with a buffer of dimension $B = 250$ data units and sensing rate $\{r_{\text{sens},i}\}$ uniformly distributed in the interval $[1, 10]$ data units/s. Regarding the *data stored local* curve (local storage without replication), the slope decreases each time a node fills its local buffer and the time to reach the storage capacity ($C = 2500$ data units) is $t_{\text{cap-1}} = 120$ s. The first dropped data occurs when the first node saturates the buffer (around 25 s). On the other hand, with ideal distribution and no replication (*data stored ideal* $R = 1$ curve), the WSN is equivalent to a single super-node: therefore, the amount of stored data increases linearly up to the storage capacity and no dropped data occurs until this point (around 40 s). Once the capacity is reached, the curve flattens and data starts to be dropped. By adding replication ($R = 3$ and $R = 5$), the total sensing rate is multiplied by R (see the denominator of the lower-bound in (2.9)): the capacity C is reached faster and data dropping starts earlier. In order to avoid dropped data, in practical scenarios data retrieval from the sink should be more frequent, preventing the local memories from saturating.

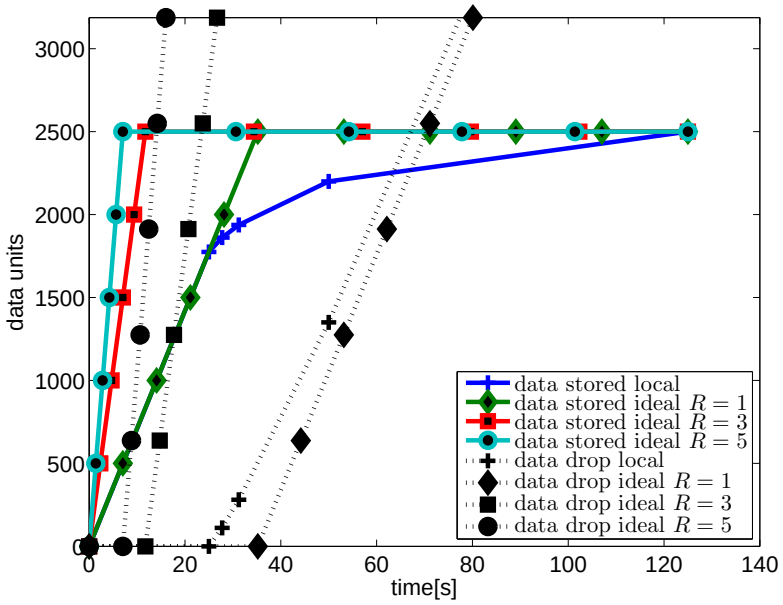


Figure 2.2: Stored and dropped data, as functions of time, with local and ideal distributed storage. For both cases, curves with and without replication are depicted.

2.3.3 Performance Results

In this subsection, we apply the analytical framework proposed in Section 2.3.2 to study the performance of our distributed storage mechanism. We complement this analysis with a simulation campaign, performed in the Cooja network simulator [17], and a large-scale experimental validation, carried out with the SensLab testbed [18]. Even if the SensLab platform is suitable for testing WSN-based applications [19], to the best of our knowledge, no studies with experimental results collected in SensLab have been published so far, but for preliminary experimental results which we presented in [16].

In the remainder of this subsection, we focus on a direct comparison between analytical, simulation, and experimental data, in order to clearly highlight the accuracy of the proposed framework. The interested reader is referred to [16] for further experimental results. In particular, we study how the system performance is affected by (*i*)

the (common) transmit power P_t; (*ii*) the sensing interval of the nodes T_{sens}; and (*iii*) the number of replicas R. We also investigate the system robustness against bomb-like events. Before presenting performance results, we summarize the experimental (SensLab) and simulation (Cooja) setups.

Setup

The SensLab platform offers 1024 sensors, equally distributed at 4 sites in France (Grenoble, Strasbourg, Lille, Rennes), where researchers can deploy their codes and run experiments. Each node's platform embeds a TI MSP430 micro-controller and operates in various frequency bands depending on the radio chip (either CC1100 or CC2420).

All the experimental results presented in the following are obtained from the Lille site of SensLab. The deployed WSN platform is the wsn430v14, which adopts the CC2420 radio chip, conformed to the IEEE 802.15.4 standard. An overview of the Lille site is shown in Figure 2.3. Nodes are installed in a regular grid, placed on verti-

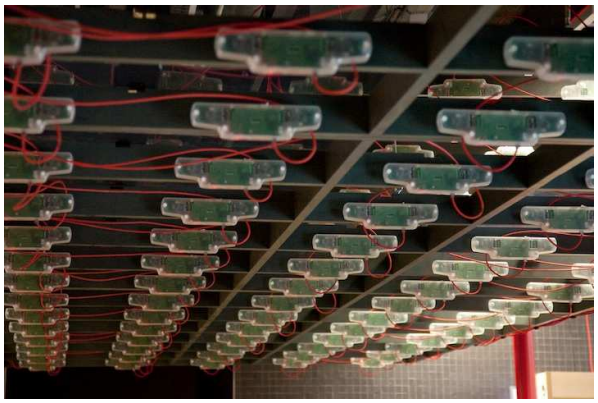


Figure 2.3: The Lille site of SensLab.

cal and horizontal trays. We have chosen the Lille site since the deployed wsn430v14 platform embeds the same MSP430 microcontroller and the same CC2420 radio transceiver of the *Tmote sky* platform, which can be easily emulated in the Cooja simulator, thus allowing a direct comparison between simulations and experiments.

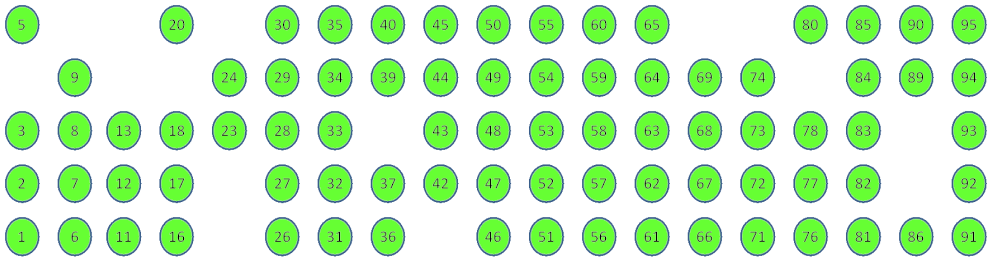
The proposed distributed storage mechanism has been implemented in Contiki, which is an open source operating system for the IoT. It allows tiny, battery-operated low-power systems to communicate with the Internet. Contiki provides two wireless networking stacks: (*i*) a full IP network stack (with standard IP protocols such as UDP, TCP, and HTTP), denoted as uIP; and (*ii*) the Rime stack, a lightweight protocol stack that supports simple primitives, such as sending a message to all neighbors or to a specified neighbor. In the implementation considered for this work, we adopt the latter.

At the link layer, nodes run the Carrier Sense Multiple Access with Collision Avoidance (CSMA/CA) mechanism, coupled with ContikiMAC [20], a low-power asynchronous radio duty cycling protocol. All nodes have the same buffer size, equal to $B = 100$ data units. The memory advertisement period T_{adv} is set to 25 s. The scenario consists of 80 nodes,[2] placed on the same horizontal tray, i.e., at the same height. A graphical representation of the experimental setup is shown in Figure 2.4(a).

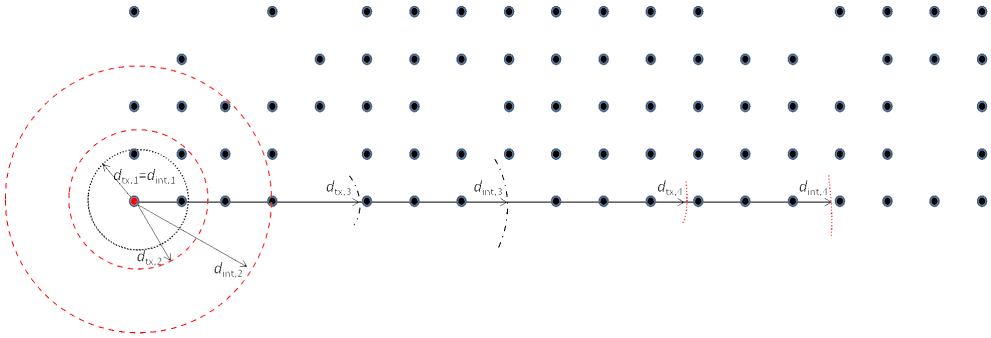
Impact of Transmit Power

We argue that our greedy approach is significantly influenced by the network topology. For instance, in a dense network, where nodes have several neighbors, our data distribution mechanism could work better than in a sparse network with only a few direct neighbors. The network topology depends on the transmit power of the nodes. The higher the transmit power P_{t}, the higher the number of neighbors a node can communicate with. However, it is hard to compute the exact number of detected neighbors in SensLab. This is due to the highly variable propagation conditions of the environment, because of: reflections and shadowing effects; radio interference with experiments run by other users; presence of people in the room; instabilities of the nodes. For this reason, experiments are executed with different power settings, namely, -20 dBm and -25 dBm, according to the CC2420 datasheet. Higher values

[2]It has been found that some SensLab nodes do not work properly or are not available for testing. This prevents us to deploy all the 256 nodes in Lille.



(a) SensLab Lille



(b) Cooja simulator

Figure 2.4: Network topology: (a) scenario tested in SensLab Lille, (b) equivalent topology simulated in Cooja, for various transmission ranges.

of the transmit power cause high interference, which leads to many collisions and degrades the overall communication performance.

On the simulation side, Cooja offers a unit disk communication model, composed by an inner transmission circle, with radius d_{tx}, and an outer interference circle, with radius $d_{\text{int}} \geq d_{\text{tx}}$. A node communicates with nodes within the circle with radius d_{tx} and interferes with nodes located within the outer circle with radius d_{int}. No interactions occur with nodes located outside the interference circle. Four combinations of d_{tx} and d_{int}, numbered from 1 to 4, have been considered in the simulations, as shown in Figure 2.4(b).

At this point, it is of interest to evaluate the time required by the system to reach the network storage capacity. In Figure 2.5, the amount of stored data is shown, as

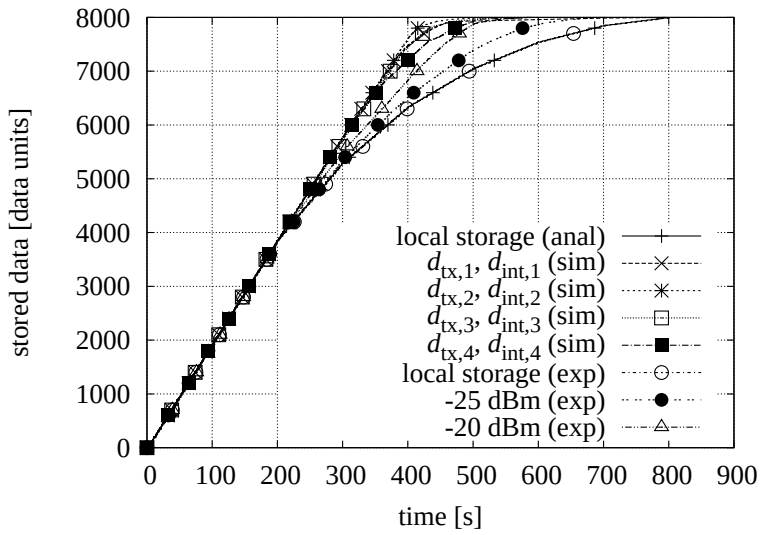


Figure 2.5: Data stored in the system, for various values of the transmit power. In all cases, we consider: no replication ($R = 1$), buffer size equal to 100 data units, $N = 80$ nodes, and $T_{\text{adv}} = 25$ s.

a function of time, by comparing the analytical prediction with experimental and simulation results.

- Considering the experimental results, it can be observed that the capacity, equal to 8000 data units, is reached later when a lower transmit power is used — for instance, -25 dBm — since fewer neighbors are detected. Consequently, data cannot be distributed efficiently through the network. On the other hand, with a higher transmit power — namely, -20 dBm — each node has a “larger" neighborhood and the capacity is reached earlier.

- The analytical framework is applied considering the case with local storage (“local storage (anal)" curve), i.e., where nodes fill their own local buffers autonomously, according to (2.5) in Subsection 2.3.2. In this case, the time to reach the storage capacity corresponds to the longest storage filling time across all nodes. As expected, the analytical curve relative to local storage

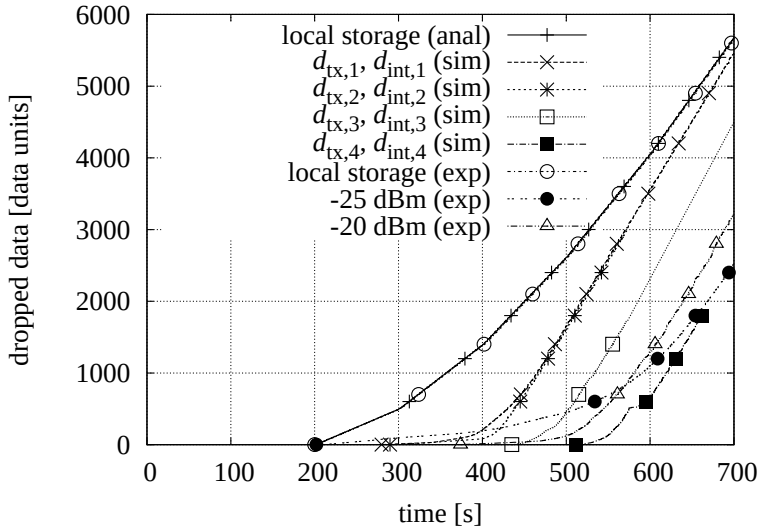


Figure 2.6: Data dropped in the system, for various values of the transmit power. In all cases, we consider: no replication ($R = 1$), buffer size equal to 100 data units, $N = 80$ nodes, and $T_{adv} = 25$ s.

lower bounds the experimental curves. Note how the “local storage (exp)" curve, obtained by setting the transmit power to 0 in SensLab, is equivalent to the analytical one.

- Considering the simulation results, it can be observed that increasing the communication range d_{tx} leads to a delayed data storage. In this case, the CSMA module detects the radio channel busy, being occupied by radio activities from other nodes, and postpones the transmission. Consequently, the time to reach the storage capacity increases.

Overall, an excellent agreement between analysis, simulations, and experiments can be observed.

We also evaluate the amount of dropped data due to local memory shortage. Results are shown, considering all configurations of Figure 2.5, in Figure 2.6. Nodes drop newly acquired data once their local memories are full and no neighbor is avail-

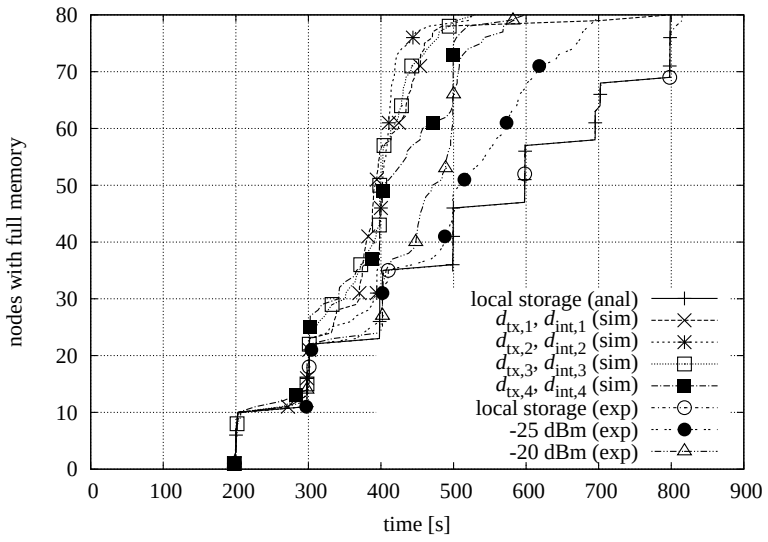


Figure 2.7: Number of nodes which fill the local memories, as a function of time, for various values of the transmit power. In all cases, we consider: no replication ($R = 1$), buffer size equal to 100 data units, $N = 80$ nodes, and $T_{\text{adv}} = 25$ s.

able for donating extra storage space. It can be observed that data dropping starts early for lower transmission ranges, e.g., with [$d_{\text{tx},1}$,$d_{\text{int},1}$] and [$d_{\text{tx},2}$,$d_{\text{int},2}$] configurations, respectively. In these cases, data is distributed more rapidly and the capacity is reached earlier. In the same figure, the performance predicted by our analytical framework with local storage, according to (2.6), and the experimental performance ("local storage (exp)" curve, -20 dBm and -25 dBm cases) are shown. As observed for Figure 2.5, in this case an excellent agreement between analysis, simulations, and experiments can be observed as well.

In order to monitor the dynamic filling of the memories, in Figure 2.7 the number of saturated nodes, i.e., nodes which have their local memories completely filled, is shown as a function of time. Again, it can be observed that the scenarios with lower transmit power are associated with a faster saturation of the nodes' memories. The step function is related to the case with local storage (both analytical and experimen-

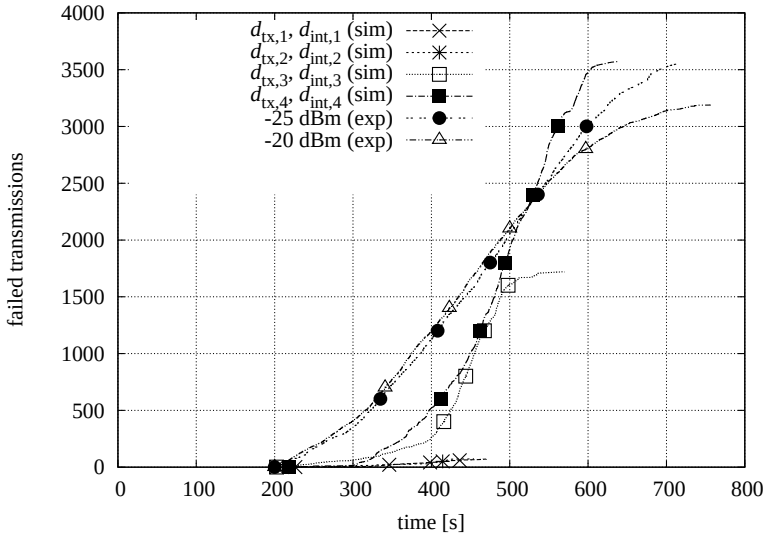


Figure 2.8: Number of failed transmissions due to no reception of an acknowledgement from the receiver or to saturation of the CSMA output queue, for various values of the transmit power. Epxerimental and simulation results are considered.In all cases, we consider: no replication ($R = 1$), buffer size equal to 100 data units, $N = 80$ nodes, and $T_{\text{adv}} = 25$ s.

tal results).

It has been anticipated that the CSMA module delays the transmission of radio packets when detecting a busy radio channel. In this case, packets are queued in the output MAC buffer up to successful delivering. The transmission fails if no acknowledgement is received back from the destination or if the queue is full.[3] In Figure 2.8, the number of failed transmissions is shown as a function of time. In this case, only experimental and simulation results are shown. As expected from the results previously shown, failed transmissions occur more frequently in the presence of a higher transmit power. However, higher congestion can be observed in the experiments, even

[3] In network theory, this situation is referred to as congestion, and it occurs when the arrival rate is greater than the departure rate.

if they are run at a very low transmit power. The same behaviour has been observed for higher values of the transmit power. We remark that a comprehensive performance evaluation of the SensLab testbed goes beyond the scope of this work and is currently under investigation.

Impact of the Sensing Interval

We now show the results obtained varying the sensing interval T_{sens} of the nodes. Recall from Table 2.2 that the sensing interval, defined as $T_{\text{sens}} = 1/r_{\text{sens}}$, is the time interval between the generation of two consecutive sensed data units by a node. We run various experiments and simulations setting the sensing interval T_{sens} to an integer value in the possible ranges: [1-4] s, [2-8] s and [4-13] s, respectively. We deploy the same scenario as in the previous case, with no replication ($R = 1$). The transmit power P_t of the SensLab nodes is set to -20 dBm. The unit disk model in Cooja is set to [$d_{\text{tx},3}$,$d_{\text{int},3}$]. In Figure 2.9, the amount of stored data is shown, as a function of time, in correspondence to the ranges of values of T_{sens} indicated above. It can be observed that the memories of the nodes are filled faster with a shorter sensing interval. In addition, the agreement between simulations and experiments is very good.

In Figure 2.10, the number of saturated nodes, i.e., nodes which have their local memories completely filled, is shown as a function of time. Nodes fill their local memories faster when shorter sensing intervals are used, e.g., when T_{sens} is selected in the range [1-4] s. In the simulated cases, the saturation of the memories occurs earlier, since the impact of the CSMA backoff delay is reduced as compared to experiments.

In order to track the energy consumption of the nodes in SensLab, we evaluate the average percentage of time the radio of the nodes is turned on. For this purpose, we use Powertrace, a system for network-level power profiling for low-power wireless networks [21]. The obtained results are shown in Figure 2.11. As expected, in the case with a short sensing interval, i.e., $T_{\text{sens}} \in [1-4]$ s, the power consumption increases, since nodes tend to distribute data more frequently. In the other two cases, i.e., $T_{\text{sens}} \in [2-8]$ s and $T_{\text{sens}} \in [4-13]$ s, respectively, the energy consumption is lower.

Overall, it can be concluded that results obtained through Cooja simulations and through real experiments in SensLab are in agreement with each other. As mentioned

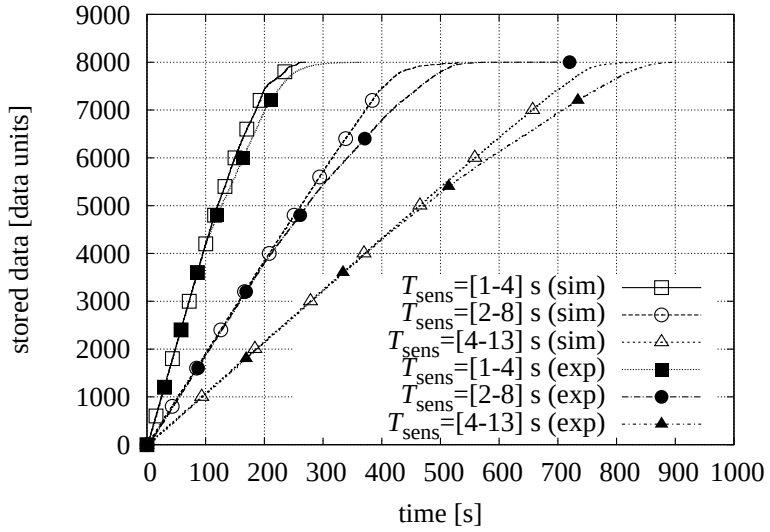


Figure 2.9: Data stored in the system, for various values of the sensing interval T_{sens}. In all cases, we consider: no replication ($R = 1$), buffer size equal to 100 data units, $N = 80$ nodes.

earlier, a thorough characterization of the SensLab platform is an interesting research topic.

System Robustness

There is often strong spatial correlation among failed nodes. The events that destroy one node may very likely influence a nearby node and destroy it as well. Examples could be a natural disaster (earthquake, fire, etc.) or system crashes (application failure, network errors, external attack).

First of all, robustness against nodes' failure is directly related to how well copies of a given data can spread across the network. As already discussed, redundancy is introduced by setting the number of copies (referred to as replicas) to a value $R > 1$. Replicas of a sensed data unit follow a hop-by-hop replication from the generator node to subsequent donor nodes, avoiding loops. For the following tests, the transmit

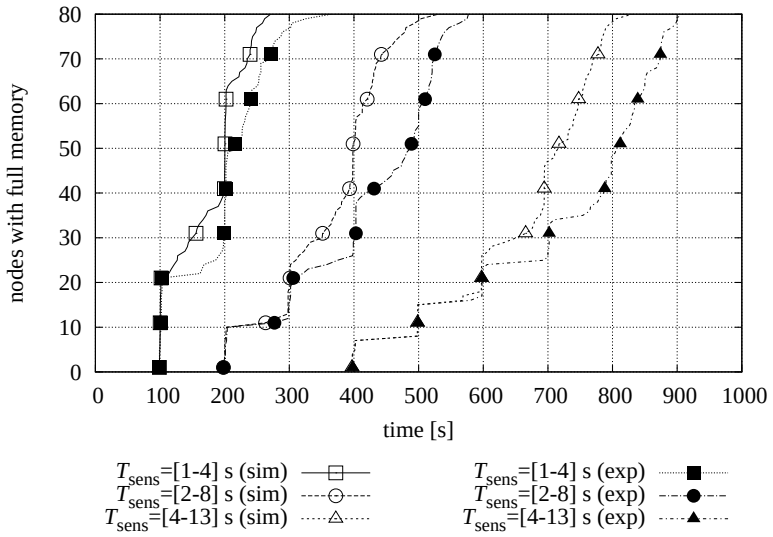


Figure 2.10: Number of nodes which fill the local memories, as a function of time, for various values of the sensing interval T_{sens}. For each value of T_{sens}, both experimental and simulated results are shown. In all cases, we consider: no replication ($R = 1$), buffer size equal to 100 data units, $N = 80$ nodes.

power P_{t} in SensLab has been increased to -10 dBm, and the unit disk in Cooja has been set to $[d_{\text{tx},3}, d_{\text{int},3}]$.

In Figure 2.12, the average hop distance (from the generator node) reached by replicas is shown as a function of the replica number — the 0-th replica refers to the original data. Various values of R are considered: for each value, the replica number varies between 0 and $R-1$. It can be observed that, on average, consecutive replicas tend to spread reasonably through the network. Simulated and experimental curves almost coincide for $R = 3$ and $R = 5$. This is in agreement with the proposed mechanism, since donor nodes are selected on the basis of their available memory space and sensing rate, which are the same for simulations and experiments. In the case with $R = 7$, there is a discrepancy between experimental and simulation results for high values of the replica number. This is due to the fact that, according to the results

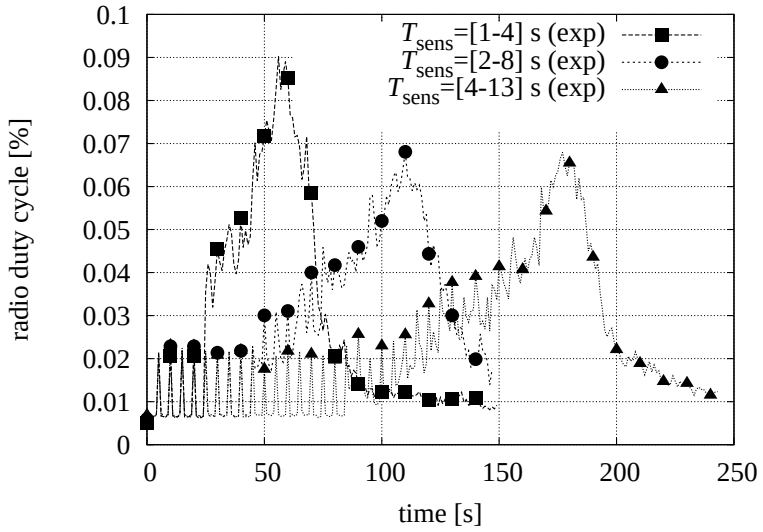


Figure 2.11: Average radio duty cycle (percent) over time, for various values of the sensing interval T_{sens}. In all cases, we consider: no replication ($R = 1$), buffer size equal to 100 data units, $N = 80$ nodes, and $T_{\text{adv}} = 25$ s.

in Figure 2.10, in the realistic SensLab scenario nodes saturate their memories more slowly. Therefore, copies are likely to be stored closer to the originator than in the simulation scenario. As already observed, however, the agreement between simulations and experiments is very good. In order to make the agreement excellent, the Cooja simulator needs to be extended to capture the phenomena (mostly at physical layer) that affects the communication performance in SensLab.

At this point, in order to investigate further the robustness of the proposed distributed storage mechanism, we assume a "bomb-like" failure event, involving a node and all its direct neighbors within a certain spatial range. We assume a squared failure range, with edge c, as depicted in Figure 2.13. Intuitively, higher values of c correspond to a more violent bomb-like event. The system robustness is defined as the percentage of distinct data units (i.e., not counting replicas) which can still be retrieved after a bomb-like event. According to this definition, assuming that the bomb-like

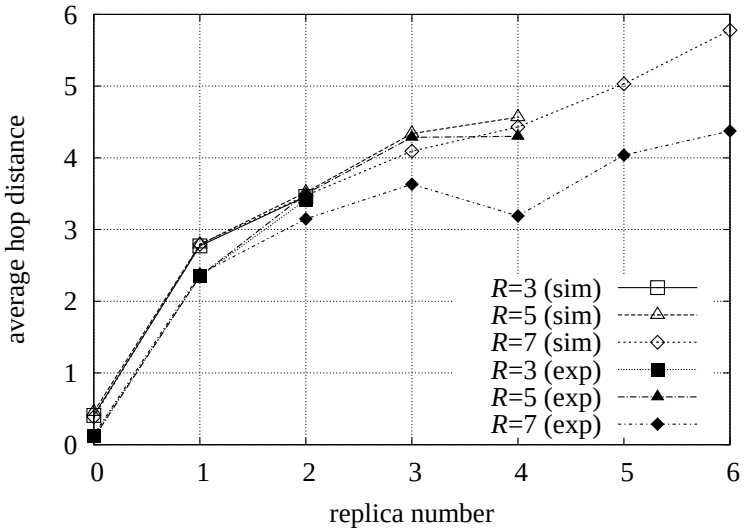


Figure 2.12: Average hop distance reached by the k-th replica versus k. k varies between 0 and 2 ($R = 3$), 0 and 4 ($R = 5$), 0 and 6 ($R = 7$), respectively. The 0-th replica refers to the original data.

event happens at time t, the system robustness $Rob_{\mathrm{sys},c}(t)$ can be expressed as

$$Rob_{\mathrm{sys},c}(t) = \frac{\sum_{l=1}^{X(t)} Rob_{\ell,c}(t)}{X(t)} \tag{2.10}$$

where: $X(t) \leq C_{\mathrm{r}}$ is the amount of distinct data stored in the system up to time t; and $Rob_{\ell,c}(t)$ equals 1 if at least a replica of the l-th data unit has been stored, at time t, outside the c-th square (otherwise, $Rob_{\ell,c}(t) = 0$). Note that the time instant t at which a bomb-like event happens has an impact on the system robustness — obviously, our definition of system robustness depends implicitly on t. At the beginning of the collection period (i.e., with limited storage), it is more likely that at least one of the R replicas is stored in a surviving node (in a node beyond the direct neighbors). On the other hand, at the end of the collection period the network saturates and nodes' buffers are almost full. Thus, it is more likely that no replica can be stored and, therefore, a bomb-like event may destroy a significant amount of information.

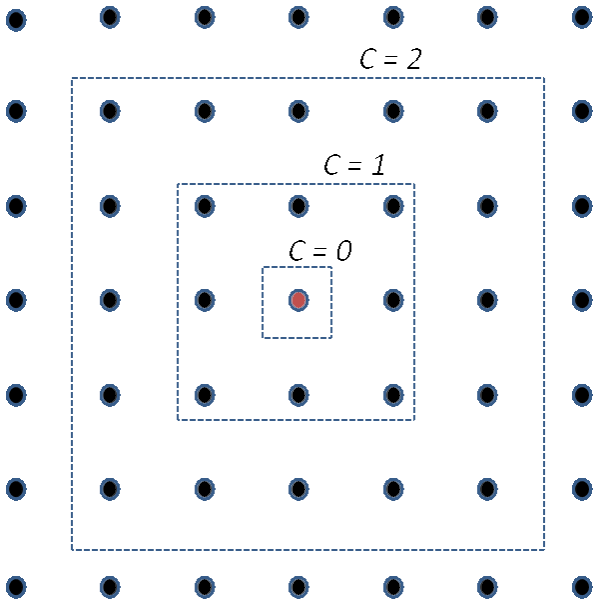


Figure 2.13: Node failure in the case of a "bomb-like" failure event, involving the node and several neighboring nodes within a certain spatial range. The c parameter controls the impact of the bomb.

In Figure 2.14, the system robustness is shown as a function of time, for various configuration of R and c. For each considered combination, both simulated and experimental results are shown. Two values of c are considered, namely, $c = 1$ and $c = 3$. The robustness increases for lower values of c, since the geometrical shape of the failure event is reduced. As for the number of replicas R, the case with $R = 3$ surprisingly has the highest robustness. In fact, storing fewer replicas slows down the saturation of the memories, and it allows more data to be replicated. Both simulations and experiments show this trend.

Simulated results show a higher robustness than experimental ones. In particular, the robustness is around 0.7 in the simulated case with $R = 3$ and $c = 1$, while in the corresponding experimental case the robustness reaches 0.5—similar conclusions hold for the remaining configurations. This is caused by (i) the slightly higher spread

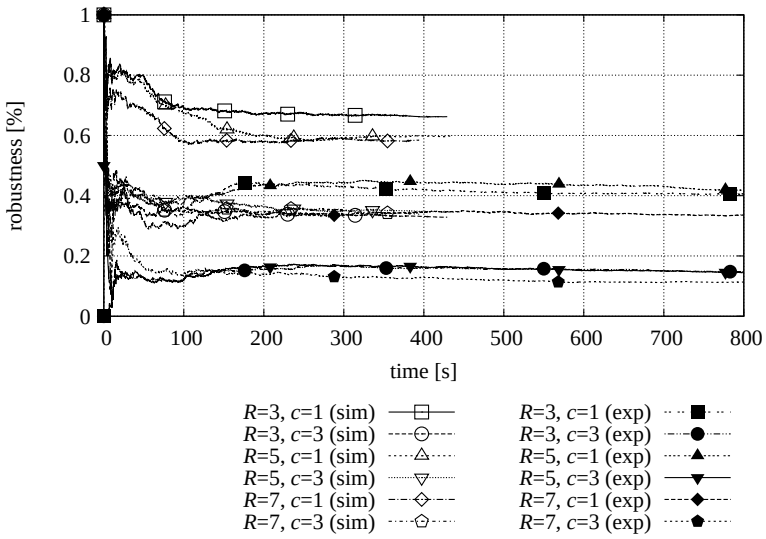


Figure 2.14: System robustness, as a function of time, considering various values of R and c. Both simulated and experimental results are shown.

of the replicas in simulations, as observed in Figure 2.12 for $R = 7$, and by (*ii*) the amount of stored data for each replica number. The latter parameter is investigated in Figure 2.15, where the Probability Mass Function (PMF) of the total stored data, as a function of the replica number, is shown, considering various values of R. It can be observed that, in the experimental case, the PMF concentrate at the origin, i.e., a considerable amount of data, about 60%, has only the original copy. On the opposite, the PMFs predicted by the simulations, regardless of the value of R, have the same trend of the experimental ones, but for a higher average value—indeed, less than 50% of the data has only the original copy. The discrepancy between experimental and simulation results is caused, as already observed, by the collisions in the SensLab testbed, which limit effective data spreading.

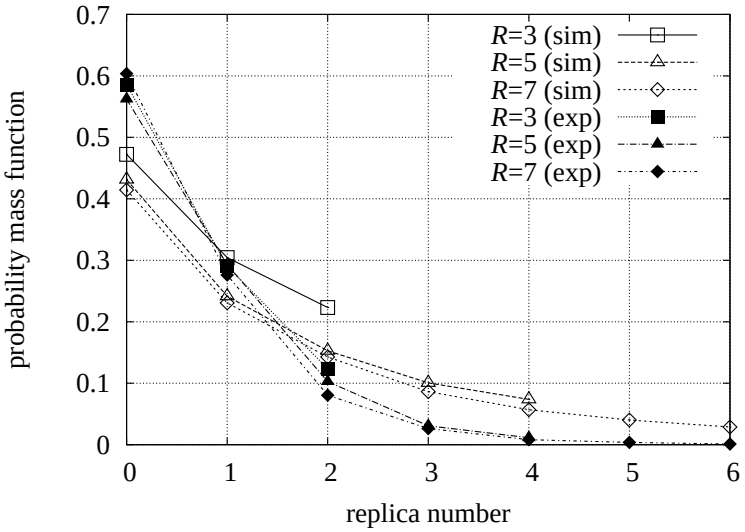


Figure 2.15: Distribution function of the stored data in the system, as a function of the copy number, for various values of R. Both simulated and experimental results are shown.

2.3.4 Discussion

The impact of several parameters on the performance of the proposed distributed data storage mechanism has been investigated, through analysis, simulations, and experiments. On the basis of the obtained results, the following observations can be carried out.

Experimental results, which significantly extend the preliminary results in [16], have been validated through an analytical framework and a comprehensive simulation campaign. In particular, the Cooja network simulator has been configured to reproduce the SensLab Lille experimental scenario as accurately as possible. We have shown that a careful calibration of the communication range in the simulator allows to obtain simulation results which have a limited discrepancy with respect to the experimental results. This discrepancy is mainly related to the higher amount of collisions in the SensLab Lille testbed, which cause the CSMA module to delay the

transmissions or even to drop packets. Accurate modeling of the propagation conditions, the interference, and the received radio signal strength of the SensLab nodes is an interesting research direction and is currently under investigation.

As for the proposed greedy distributed storage mechanism, it can be concluded that the lightweight hop-by-hop replication scheme guarantees a reasonable spread of the replicas. Therefore, this mechanism is robust in the case of a failure involving several nodes within a limited area. A shortcoming of the proposed approach, however, is that it tries to make exactly R replicas of all data. This may be unconvenient when the memories of the nodes are almost full, and may prevent new data to be stored. A solution to this problem could rely on a dynamic self-organization of the replicas, which autonomously decide to replicate or not. For example, the replication process may stop in correspondence of an intermediate r-th step (with $r < R$) if a prior replica has already been stored far away. In order to prevent dropping of locally generated data, thus increasing robustness, the storage space at each node could also be divided into two blocks: one for local measurements and the other for data coming from other nodes.

Another appealing extension of the proposed mechanism is in the direction of including load balancing, e.g., by equalizing the levels of occupancy of the nodes' memories over time. Replicas may move from a saturated network region to an offloaded one. However, the complexity of this solution could be quite high and may require the use of a centralized approach.

Finally, the proposed mechanism could be extended with a cross-layer solution between application layer and routing layer. By using information from the routing layer could lead to (i) a more efficient donor node selection algorithm, and (ii) a broad propagation of redundant copies through the network. The next section presents an enhancement to the current solution that makes use of RPL, the IPv6 routing protocol for Low power and Lossy Networks (LLNs) standardized by IETF ROLL [22], which is one of the building blocks of the IoT.

2.4 Routing-based Distributed Storage with RPL

In the previous section we have presented a low complexity greedy mechanism for distributed data storage. A drawback of the proposed solution is that replicas of a given data item do not spread throughout the WSN. In this section, an alternative approach, which makes use of the RPL routing protocol, is described. Basically, we want to include information on the position of the node within the network for a more efficient data placement as well as helping data retrieval. Information about the position of a node is suggested by the RPL rank, a value provided by the RPL routing protocol that indicates the position of a node within a routing tree, according to a routing function.

This section is organized as follows. First, an overview of RPL is given in Subsection 2.4.1. A detailed description of the new storage algorithm, highlighting the main differences with the previous greedy one, is provided in Subsection 2.4.2. In Subsection 2.4.3, we evaluate the protocol through extensive simulations conducted in Cooja, the wireless sensor network simulator in Contiki. At last, Subsection 2.4.4 concludes the discussion about data storage and introduces the data retrieval problem, which is the topic of the last part of the chapter.

2.4.1 Overview of RPL

RPL is a distance-vector protocol which creates a routing tree, referred to as *Destination Oriented Acyclic Directed Graph* (DODAG), where the cost of each path is evaluated according the metrics defined in an Objective Function (OF). The goal of this protocol is the creation of a collection tree protocol, and a point-to-multipoint network from the root of the network to the devices inside the LLN, as well as a point-to-point network between any pair of devices.

In order to build the tree and to keep the status of the network updated, the root of the RPL tree periodically sends *DODAG Information Object* (DIO) messages. The receiving nodes may relay these messages or just consume them, if configured as leaves of the tree. The mechanism of RPL is quite simple: each node has a rank which places it in the hierarchy of the RPL tree and lets it define which nodes are

its parents. When a DIO message is received for the first time, a node, before setting its rank, listens through the network discovery mechanism which are the possible parents to whom it can join. At the end of this discovery phase, according to the outcomes of the OF, a node sets its rank to be highest among the ranks of its possible fathers. To avoid loops or network misconfiguration, two nodes in the same network are not allowed to have the same rank so, as soon as such a situation is detected, one of the conflicting node updates its status. According to the outcome of the OF, the node selects also the best path it can use to transfer the data to the root of the tree. On the other side, if a DIO message has already been received at least once, the node evaluates the incoming DIO message to check whether its position in the DODAG tree can remain the same or must be updated. In the former case, the node discards the packet, whereas in the latter it computes and its new rank and it discards the list of parents, in order to avoid creating loops due to its new position in the DODAG tree. In Figure 2.16, we show the operations that a node carries out to establish its role in the DODAG tree when it receives a DIO packet.

Since devices in LLNs are typically resource constrained, the RPL protocol also introduces a *trickle* mechanism to reduce the transmission frequency of DIO messages according to the stability of the network. If the network status is stable, the frequency of transmission of DIO messages decreases. As soon as an anomaly or an inconsistency within the network is detected, the frequency is kept back to the default value and a procedure of recovery is started. If the chosen procedure is a local repair, the node simply selects a new best parent. Otherwise, in the case of a global repair, the root sends a new DIO message to restart the construction of the tree.

Since information may flow also in the other direction, that is from the root to the leaves, and since the communication links are asymmetric, RPL has defined a strategy also for the construction of downwards routes. In this case, the remote nodes send *Destination Advertisement Object* (DAO) messages to the root. While traveling back to the root, the intermediate nodes addresses are stored in the packet, so that the complete route from the root to the node is created. There are two ways of operating while creating a downward route: (i) *storing* and (ii) *non-storing* mode. In the former case, a message is sent from the remote node to the root and each intermediate parent

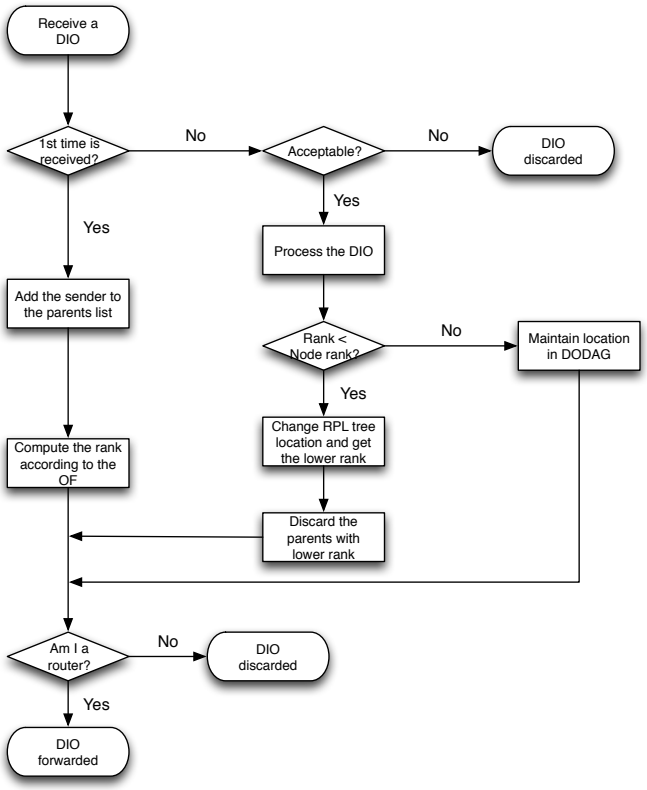


Figure 2.16: Diagram of the creation of a RPL tree in a node belonging to a DODAG tree.

node stores the addresses of the children from which it has received a DAO. In the latter, instead, intermediate nodes do not store the addresses of their children, but they only limit to insert their address in the DAO, transferring it to the root, which will be the only node with the complete topology to transfer information downward.

The control messages exchanged by RPL can also be used to convey additional information, depending on the specific application requirements. This technique, referred to as *piggybacking*, allows to significantly reduce the amount of exchanged messages. For instance, it can be used to transfer information about the status of a

node.

2.4.2 Design Principles

The RPL-based distributed storage mechanism is based on the same principles of previous greedy one: nodes of the WSN, upon joining a RPL DODAG, keep on collecting data (acquired with a given sensing rate). Data retrieval is performed by an external agent that periodically connects to the DODAG root and gathers all the data from the WSN.[4]

In order to prevent data losses, data is stored in several nodes (possibly including the generating node). This consists in copying and distributing replicas of the same data to other nodes with some available memory. As in the greedy mechanism, information about memory availability is periodically broadcasted, by each node, to all direct neighbors.

The main parameters are listed in Table 2.2. Without loss of generality, we consider a WSN with N fixed RPL nodes deployed over an area whose surface is A (dimension: [m^2]). Nodes only interact with 1-hop neighbors, i.e., with nodes within the radio transmission range d (dimension: [m]). Note that an additional node acts as RPL DODAG root, even if it does not participate in sensing and storage. Therefore, the final number of nodes is $N+1$. The i-th node has a finite local buffer of size B_i (dimension: [data units]) and sensing rate r_i (dimension: [data units/s]).

Each node broadcasts without acknowledgement and every T_{adv} (dimension: [s]), its memory status to all nodes within direct transmission range (i.e., 1-hop neighbors). Each memory advertisement consists of 6 fields relative to the sending node: the RPL rank of the node; value of sensing rate; up-to-date available memory space; an aggregate that indicates the status of node memories in the down direction in the DODAG, an analogous value for the up direction, and a sequence number. Each node maintains a table which records the latest memory status received from neighbor nodes. Upon reception of a memory advertisement from a neighbor, a node updates its memory table, using the sequence number field to discard multiple receptions or

[4] RPL-based data retrieval is the focus of the next section.

Symbol	Description	Unit
N	Number of RPL nodes	scalar
A	Surface of deployment area	m^2
d	Node transmission range	m
B_i	Node i's buffer size, $i \in \{1,\ldots,N\}$	scalar
r_i	Node i's sensing rate, $i \in \{1,\ldots,N\}$	s^{-1}
MaxHopDown	Maximum distance (in the down direction), that can be announced in a memory advertisement, at which some available space is present	scalar
MaxHopUp	Maximum distance (in the up direction), that can be announced in a memory advertisement, at which some available space is present	scalar
T_{adv}	Period of memory advertisement (from each node)	s
R	Maximum number of replicas per sensing data unit	scalar
T	Period of data retrieval (from the sink)	s

Table 2.2: Main system parameters.

out-of-date advertisements. The aggregate of the status of node memories in the down direction in the DODAG is given the minimum hop distance at which a node with some available space can be found. This distance is computed as follows: if a node detects that at least one of its children (i.e., neighbors with higher RPL rank) has some space locally, it sets this distance to 1. Otherwise, a parent increments by 1 the value of the minimum distance given by its children. Once the distance reaches a maximum value, a node assumes that there is no available memory in the down direction of the DODAG. The maximum value of the flag is given by the *MaxHopDown* parameter, listed in Table 2.2. Similarly, the status of the node memories in the up direction is

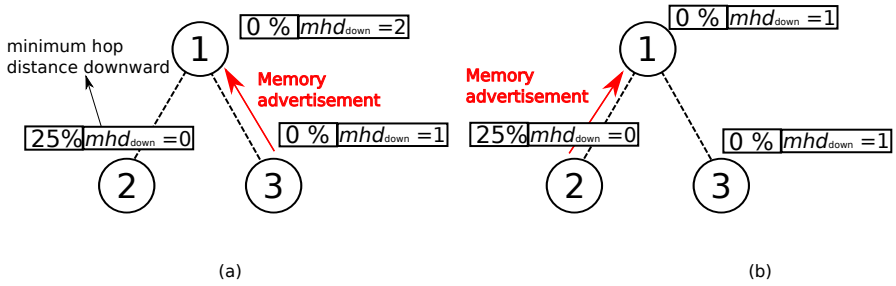


Figure 2.17: Messages exchanged for memory advertisements.

computed in the same way, but in the inverse direction of the DODAG; in this case, the maximum hop distance that can be announced in a memory advertisement is given by the *MaxHopUp* parameter.

The first message exchange, depicted in Figure 2.17 (a), is between node 3 and node 1. Node 3 transfers a memory advertisement saying that it has no available space but there is one of its 1-hop children with available space, as stated by the minimum hop distance downward (mhd_{down}) parameter. Node 1, which has no available memory, then sets mhd_{down} to 2, because it has received the information that the closest node with available memory is at 2-hop distance. Then, as shown in Figure 2.17 (b), node 2 sends its memory advertisement saying that it has available space. Consequently, node 1 becomes aware that there is a closer node with available space, so it updates its mhd_{down} to 1.

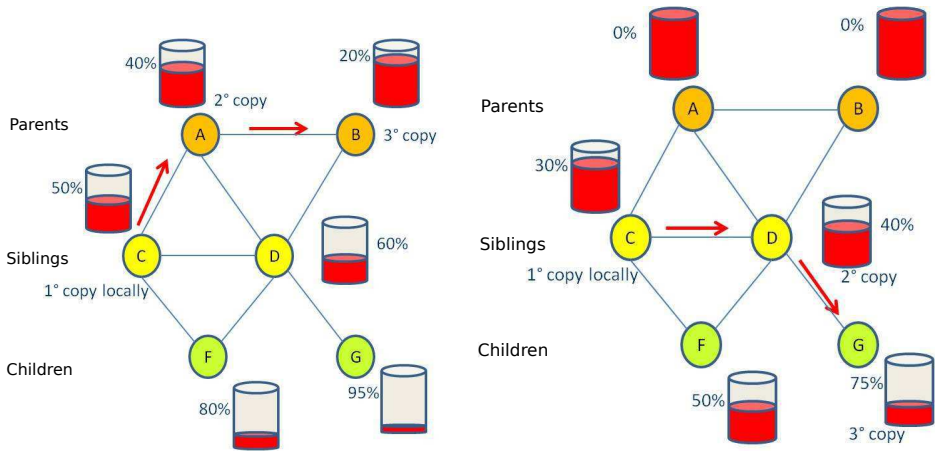
An illustrative scenario is shown in Figure 2.17.

Like the greedy mechanism, the RPL-based mechanism is fully decentralized, in the sense that all nodes play the same role. It consists in creating at most R copies of each data unit generated by a node and distribute them across the network, storing at most one copy per node, possibly closer to the DODAG root to reduce later the energy consumption of the retrieval phase. Each copy is referred to as *replica*. Let us focus on node $i \in \{1,\ldots,N\}$. At time t, the node generates (upon sensing) a data unit. The memory table of node i contains one entry per direct neighbor. Node i selects from its memory table the neighbor node, called *donor*, with the largest available memory space and the most recent information. Moreover, priority is given to those donors

which are parents of node i in the tree, i.e., nodes with lower rank. If no parents can be selected, node i looks for a sibling in the tree, i.e., a node with the same parent, providing that such node has some available space. If no sibling can be chosen, node i searches for a child. In case that all neighbors have no space locally, then node i checks if a neighbor at least knows about some available space beyond 1 hop, i.e., in the up direction and/or in the down direction of the DODAG. In this case, again, priority is given to nodes in the up direction. If there is no suitable neighbor in the memory table, there is no possibility to distribute replicas of the data unit across the network. In this case, only one copy can be stored in the local memory of node i, if i has some space locally.

If a donor node can be selected, node i sends to it a copy of the data unit, specifying how many other copies are still to be distributed in the WSN. As in the greedy mechanism, the number of required copies is set to either $R-1$ (if node i can store the original data locally) or R (if node i's local memory is full). Upon reception of the copy, the donor node stores the copy in its memory, if it has some space locally, and selects the next donor node among its neighbors, discarding the sending node and the source node from the candidate nodes. The next donor is chosen such that its RPL rank diverges from the one of the generator node. This causes replicas to spread well throughout the RPL tree. At this point, the donor sends the copy to the next chosen donor node, decrementing the number of required copies by 1. The replication process continues recursively until either the last (R-th) copy is stored or stops when one donor node cannot find any suitable next donor node. In the latter case, the final number of copies actually stored in the WSN is smaller than R. Note that, if a donor cannot find a next neighbor with a suitable RPL rank, the replica may follow a different reversed path along the DODAG, and retake the original direction later.

In Figure 2.18 we show an illustrative example with $R = 3$ desired replicas. In Figure 2.18(a), nodes always try to distribute replicas to parents in the up direction of the RPL tree, e.g., nodes with a lower rank. As the network saturates, data is distributed towards nodes with the same rank or with a higher rank, i.e., in the down direction of the DODAG, as shown in Figure 2.18(b).



(a) Distribution to parents in the RPL tree.

(b) Distribution to siblings and children.

Figure 2.18: Hop-by-hop replication in the case of $R = 3$ desired replicas, for several scenarios: (a) replicas propagate to nodes closer to the RPL root first; (b) data is distributed to nodes in the down direction of the tree as the memories of the parents are full.

2.4.3 Performance Evaluation

The RPL-based mechanism has been implemented in Contiki 2.5 and evaluated in Cooja. The scenario is depicted in Figure 2.19. A rectangular grid is composed of 60 storing nodes, shown in green. Each node inside the grid communicates with 4 direct neighbors. Moreover, to simulate real conditions, the node interferes with some extra nodes: collision occurs if a node and at least one amongst its direct neighbors or its interfering nodes transmit a packet at the same time. For example, referring to Figure 2.19, the neighbors of node 36 are nodes: 50, 51, 55, and 43, respectively. The interfering nodes, shown between the two circles, are nodes: 29, 54, 56 and 57, respectively. Collisions may be caused also by the hidden terminal problem [23]. A 100% Packet Delivery Ratio (PDR) is assumed, i.e., packets are always delivered, providing that no collision has occurred. Note that nodes along the borders have less neighbors than the others. Finally, node 1, in the upper left part of the grid, acts

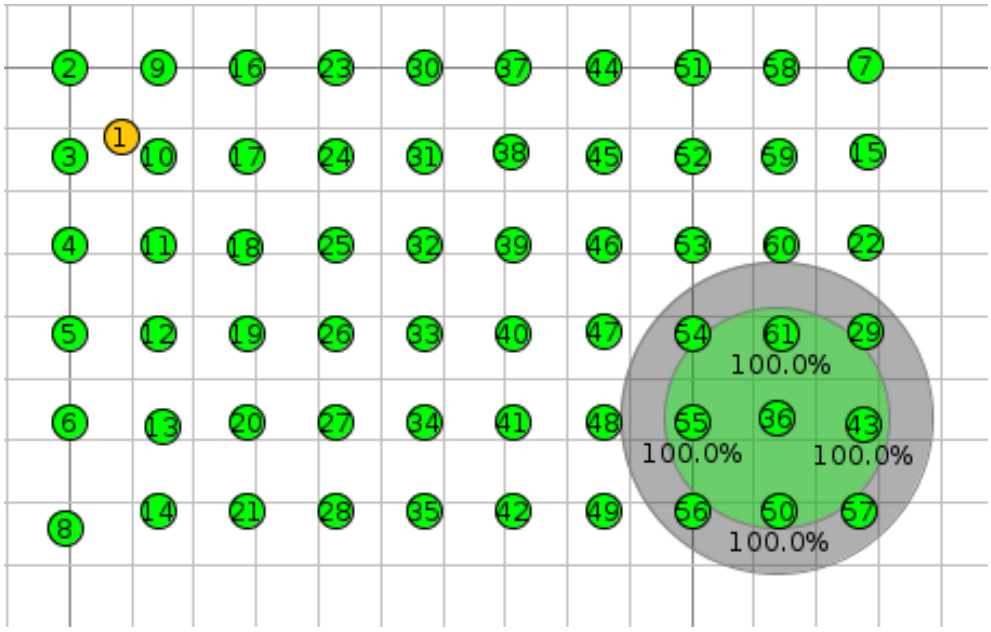


Figure 2.19: Scenario evaluated in Cooja. 60 storage nodes (shown in green) are deployed in a regular grid. Each node has 4 direct neighbors. The RPL root is node 1 of the figure (shown in yellow).

as RPL root, creates the routing topology, and performs periodical data retrieval. In Figure 2.20, a detailed view of the scenario is depicted. Nodes at the same hop distance from the RPL root are drawn with the same color. The maximum number of hops equals 13. It has been observed that RPL sometimes modifies the structure of the tree, therefore, a node may change its RPL rank.

The interval between two consecutive data retrieval is $T = 10$ min. The sensing period of the nodes is an integer number chosen randomly and independently in the range [1,9] s. All nodes have the same buffer size, which equals $B = 100$ data units. The memory advertisement period T_{adv} is set to 30 s. We adopt the Expected Transmission Count (ETX) as RPL metric. According to this metric, the routing function tends to minimize the number of expected transmission towards the RPL root. In the presence of perfect reliable links, it is equivalent to minimize the number of hops.

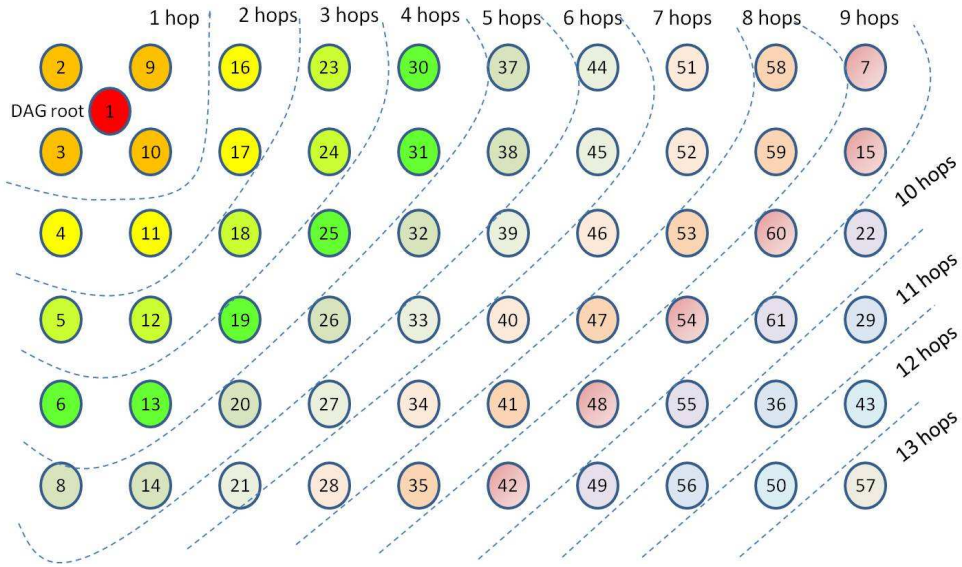


Figure 2.20: Hop distance of each node from the RPL root. The network has a total size equal to 13 hops.

Impact of the Number of Replicas

We run 4 simulations, setting the number of replicas R to 1, 3, 5, and 7, respectively. Note that the case with $R = 1$ is without replication, i.e., only the original copy is stored.

The time required by the system to reach the network storage capacity is computed in the different cases. With the setting described above, the network storage capacity C, which denotes the maximum amount of data that can be stored in the WSN, equals $C = N \times B = 60 \times 100 = 6000$ data units. In Figure 2.21(a), the amount of total stored data is shown, as a function of time, for the four considered simulated cases. It can be observed that the capacity is reached later when fewer replicas are used — for instance when $R = 1$ — since less data is generated. On the other hand, for higher values of R, the capacity is reached earlier. However, the slope of the curve, in the case with $R = 3$, tends to approximate the ones obtained with $R = 5$ and $R = 7$.

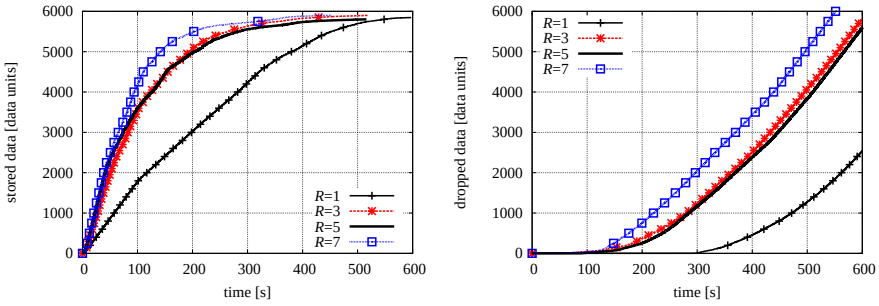
This is due to the fact that a higher storing rate quickly saturates the memories of the nodes at the beginning of the simulation, leading to a less efficient data distribution later.

In Figure 2.21(b), the amount of dropped data is shown, as a function of time. Nodes drop newly acquired data once their local memory has filled up and no neighbors are available for donating extra storage space. Remember that data is marked as dropped if no replicas at all can be stored. This explains why related curves with $R = 3$ and $R = 5$ are close to each other. This suggests that it is inconvenient to store exactly R replicas for each sensed data unit, since the first replicas that are stored would quickly congest the memories, and impede the next ones to be stored.

In Figure 2.21(c), the amount of *unique*' stored data is shown instead, as a function of time. In fact, in the presence of redundancy ($R > 1$) several copies of the same data unit are distributed in the WSN, so that the number of original node data is smaller than the total number of actually stored data. The analysis of the unique stored data is expedient to evaluate the efficiency of data retrieval, as will be shown in Section 2.5.

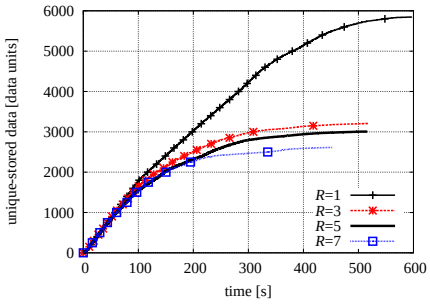
At this point, it is of interest to evaluate the data placement throughout the WSN over time. As discussed previously, the mechanism distributes replicas prioritizing donor nodes closer to the root. Figure 2.22 shows the average saturation level of the memories of the nodes at different hop distances from the root, for several observation instants. It can be noticed that the portion of the RPL tree closer to the root fills the memories faster. This can be of help in the data retrieval phase, since data follows a shorter path to reach the sink.

As for the distribution of the replicas, Figure 2.23 shows the average hop distance reached by the redundant copies, from the owner of the original one, for the different cases with $R > 1$. Results show that replicas keep on moving away from each other, passing through nodes with a higher RPL rank in the tree. By comparing the results in Figure 2.23 with those, relative to the LG mechanism, shown in Figure 2.12, it can be observed that, with the help of RPL, replicas tend to spread farther from the originating node than with the previous hop-by-hop greedy mechanism. Therefore, the current cross-layer solution leads to a more resilient data preservation in the presence



(a) Stored data.

(b) Dropped data.



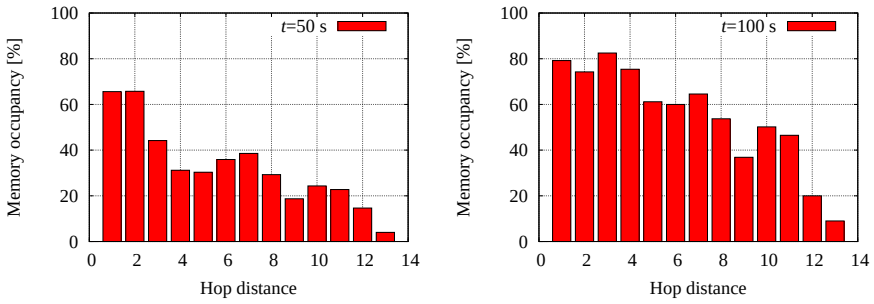
(c) Unique stored data.

Figure 2.21: Stored and dropped data in the system for various values of replicas R. In all cases, we consider: memory size equal to 100 data units, $N = 60$ storage nodes, and a memory advertisement period of 30 s.

of a failure involving a source node and several of its neighboring nodes.

2.4.4 How to Retrieve the Stored Data?

In Section 2.3 and in the current one, two redundant data distribution mechanisms have been presented: the greedy hop-by-hop mechanism increases the network storage capacity with a slight signaling overhead; the RPL-based mechanism, owing to a more accurate view of the network status (thanks to routing information), allows replicas to spread better across the WSN. While the two proposed mechanisms deal



(a) $t = 50$ s (b) $t = 100$ s

Figure 2.22: Percentage of memory occupancy varying the hop distance from the RPL root, at different time instants. Memories of nodes closer to the RPL root saturate faster. The number of replicas R is set to 7.

with distributed storage, data retrieval is fundamental in order to collect (to a sink) the stored data.

Data retrieval consists of a periodic collection of the whole sensed data, stored in the WSN, at a single node in the network, usually denoted as sink. The sink then passes all the collected data to a monitoring center for further processing. During the retrieval phase, the major challenges are the following.

- Sending lot of data depletes the battery of the nodes, thus the lifetime of the WSN.
- Multihop communication is required for those nodes which are far away from the sink. Therefore, battery depletion is even more critical for intermediate nodes which have to forward the data of the others.
- The amount of stored data in the WSN can be extremely high, if the retrieval is not timely performed. Sending all the data instantaneously to the sink can cause many collisions, thus losing a lot of information. A solution could be letting the nodes to wait some time between two consecutive data transmissions. However, this may excessively prolong the duration of the retrieval.

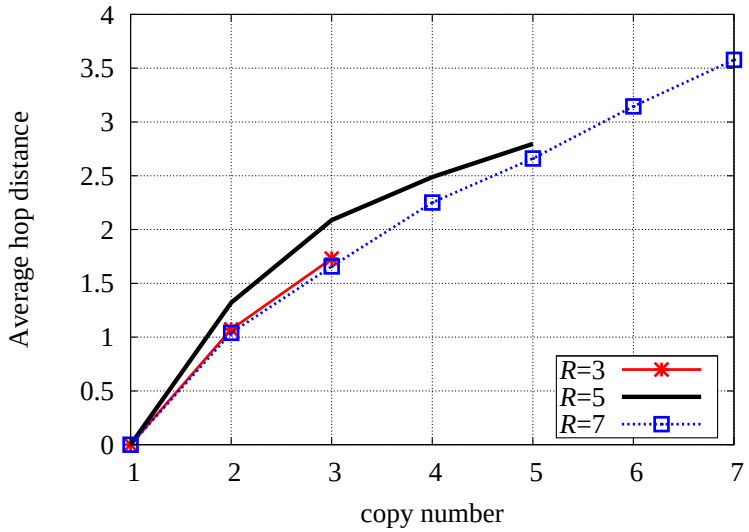


Figure 2.23: Average hop distance reached by the k-th replica versus k. k varies between 1 and 3 ($R = 3$), 1 and 5 ($R = 5$), 1 and 7 ($R = 7$), respectively. The distance is calculated from the position of the first replica.

The RPL-based mechanism leads directly to a data retrieval phase, as it builds upon the RPL routing algorithm. Therefore, the next section is dedicated to data retrieval with RPL and all the aspects outlined above will be covered in depth.

2.5 Data Retrieval with RPL

Data retrieval consists in forwarding the collected sensing data of the WSN to a central base station for further processing. In literature it often appears with various terms, e.g.,, *data collection* and *data gathering*. This section briefly reviews some existing works in this field and presents the data retrieval mechanism that has been implemented and evaluated on top of the RPL-based distributed storage mechanism.

2.5.1 Related Work on Data Retrieval

As stated in [24], data retrieval in WSNs is still in its early stage. Two main approaches can be identified: *mobile data retrieval* and *fixed data retrieval*.

In the context of *mobile data retrieval*, the idea is to have a mobile entity which travels across the WSN and gathers data from every sensor node within the communication range. This should save battery at the sensors and increase the lifetime of the network. Usually, no specific routing topology is required, as the communication between the mobile sink and the sensor node is single-hop. In [25] authors investigate the optimal path selection for a mobile sink in a path-constrained scenario with delay requirements. In [26], a rich analytical framework to measure data retrieval rate, latency and consumed power is given. Several protocols to optimize data transfer at the minimum energy cost have been proposed [27]. In [28], the problem of data collection scheduling to avoid node's buffer overflow is investigated. For a detailed survey on mobile data retrieval please refer to [29]. In our work, we consider periodical data retrieval performed by a fixed sink node.

As for *fixed data retrieval*, a first approach, discussed in [30], consists of a sink node who periodically stops at a fixed point in the network. Once reached the point, the sink advertises its presence by sending queries. Each sensor node builds its own return path to the sink and consequently sends the collected data. In such a way a routing tree towards the sink is formed. The Collection Tree Protocol (CTP) is probably the routing mechanism most frequently used for multi-hop fixed data retrieval in sensor networks [31]. The strengths of CTP are its ability to quickly discover and repair path inconsistencies and its adaptive beaconing, which reduces protocol overhead and allows for low radio duty cycles. Extended versions of CTP have been proposed to deal with nodes' mobility [32]. Dozer [33] is a data retrieval protocol aiming at achieving extremely low energy consumption. It builds a tree structure used to convey data to the sink, enriched with a Time Division Multiple Access (TDMA) scheme at the MAC layer to synchronize the nodes. In Koala [34], a data gathering system is proposed. In contrast with Dozer, routes are built at the sink and no persistent routing state is maintained on the motes. Coupled with Low Power Probing (LPP) technique at the MAC layer, Koala can achieve very low duty cycles.

With respect to related works, we consider periodical data retrieval in an isolated RPL-based network. Instead of a cross-design among MAC layer, topology control, routing and scheduling, we let the RPL protocol to handle the routing structure autonomously. Moreover, we focus more on data availability at the sink and on latency of the retrieval than on energy efficiency.

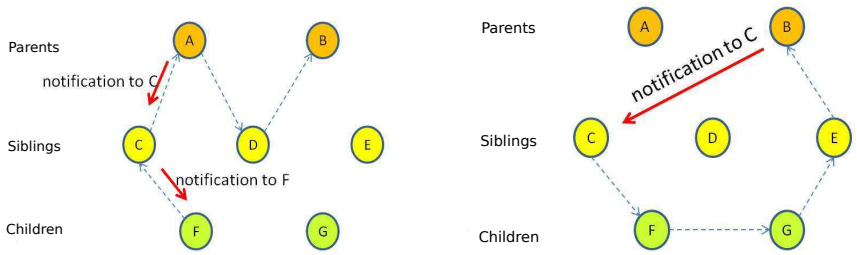
2.5.2 Data Retrieval Mechanism

We describe here the data retrieval mechanism that has been designed and implemented to work together with the RPL-based distributed data storage mechanism proposed in Section 2.4.

Recall from Section 2.4.2 that the mechanism replicates and distributes copies of sensed data units towards nodes closer to the sink. The sink is supposed to periodically send retrieval requests through the RPL root. In particular, the retrieval request is broadcasted by each node of the tree only once. Subsequent receptions of the same retrieval request are ignored. Moreover, a sequence number is used by the RPL root to identify each periodical retrieval request.

The data collection takes place from each sensor node towards the RPL root. Such *many-to-one* traffic pattern, if not carefully handled, can cause (i) many collisions and (ii) high unbalanced and inefficient energy consumption in the whole network. To reduce these risks, the storage mechanism has been enriched with the following: a replica of a given data item is sent to the sink only if it is the closest to the RPL root, amongst all the stored replicas of that data item. To let a node be informed that it holds a replica closer to the root, donor nodes include their IPv6 address and RPL rank within a data packet, during the distribution phase. Since the mechanism follows a hop-by-hop greedy replication scheme, a next donor node along the chain checks the rank of the closest donor node who has stored the replica before, and compares it with its own rank. In case, its own rank is lower than the value announced in the data packet, i.e., the node is closer to the RPL root than its ancestor donor, and providing that the node stores the replica in its local memory, then the node informs the ancestor donor about the newly closer position of the replica. In the retrieval phase, the ancestor donor will not send such replica to the sink, as it knows that it is not

the closest one. An illustrative example is depicted in Figure 2.24. In Figure 2.24(a), replicas of a data item generated at node F are stored, progressively, at nodes C, A, D, and B, respectively. Node C informs node F of its better position in the tree (C has rank k, while F has rank $k+1$, with $k > 0$). Similarly, node A has to inform node C for the same reason. Note that D and B do not send notification to A, as they are not closer to the RPL root. In Figure 2.24(b), notification is delivered only from B to C.



(a) Case with two notifications.

(b) Case with only one notification.

Figure 2.24: Only the closest replica is sent to the sink. A donor node with lower rank informs the prior *best* donor about the new position of the replica.

2.5.3 Performance Evaluation

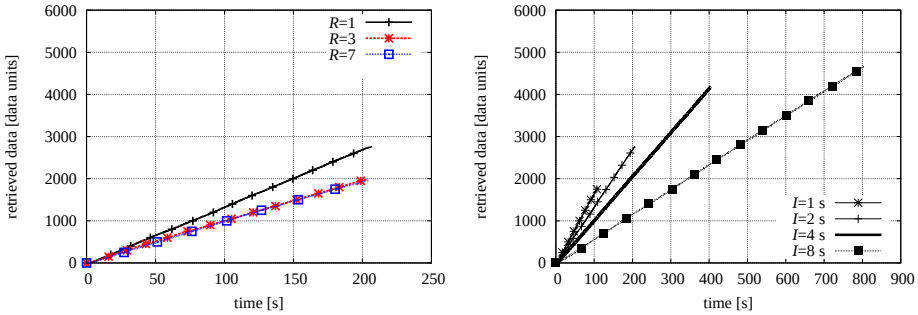
The proposed data retrieval scheme, has been evaluated with the same settings of Section 2.4.3. The periodical retrieval occurs every $T = 600$ s. At this time instant, as shown in Figure 2.21(a), the system has already reached the network storage capacity C, equal to 6000 data units. Therefore, the amount of data stored in the WSN to be delivered to the RPL root is considerable; if all nodes would send 100 data units contemporaneously, collisions would impact the amount of data successfully delivered to the sink. To investigate the data availability at the sink, i.e., the amount of data effectively retrieved by the sink, data units have been injected in the network, from each node towards the RPL root, one at a time. The interval between the injection of two consecutive data units is denoted as I. Intuitively, higher injection rates may speed

up the retrieval process but could also increase the collisions' probability, thus the percentage of data loss at the sink. On the other hand, a longer transmission interval I may increase data availability, penalizing the latency. This is critical in scenarios where the sink cannot prolong the duration of the retrieval.

Collisions are influenced also by the amount of data that is to be retrieved. With redundancy ($R > 1$), less data is to be delivered to the sink, since the proposed schema only sends one replica of a data unit, e.g., only the closest replica is sent. It has already been shown in Figure 2.21(c) that the amount of unique stored data in the system with $R > 1$ is much less than in the case with no redundancy, i.e., $R = 1$. However, a comparable volume of unique data is present in the cases with $R = 3$, $R = 5$, and $R = 7$, respectively. In Figure 2.25(a), the amount of retrieved data is shown, as a function of time, for the various values of replicas R. The interval I is set to 2 s in this case. It can be observed that the amount of retrieved data decreases with R, since there is more unique data in the system when no redundancy is required. With $R = 1$, all 6000 data units are sent to the sink, as there is no redundancy, but only a small fraction, i.e., about 50% of the capacity, is successfully retrieved; the rest is lost because of collisions. For higher values of R, there are more replicas of the same data unit in the WSN, therefore, according to the mechanism, only the closest replica is collected. However, since the amount of unique stored data is similar, the volume of retrieved data is the same in the cases with $R = 3$ and $R = 7$, respectively. The about 2000 retrieved data units correponds approximately to 80% of the capacity.

To reduce the impact of collisions, several simulations have been run with different values of the transmission interval I. Results, depicted in Figure 2.25(b) for $R = 1$, show that the amount of retrieved data significantly increases for higher values of I. To summarize, the percentage of retrieved data amongst the total unique stored data in the system is shown in Table 2.3 for various combinations of R and I.

Finally, Figure 2.26 shows the average percentage of data sent to the sink from several hop distances from the RPL root. In this case, the data retrieval period is set to $T = 100$ s, thus at the beginning of the simulation. Figure 2.26(a) corresponds to the case with $R = 1$: all nodes of the WSN send all their data to the sink. In Figure 2.26(b), $R = 7$ replicas are stored in the system. In this case, nodes closer to the RPL root



(a) Retrieved data for several values of R. Each node injects a data unit, towards the sink, every $I = 2$ s.

(b) Retrieved data for several values of I.

Figure 2.25: Retrieved data considering memory size equal to 100 data units, $N = 60$ storage nodes, and a memory advertisement period of 30 s.

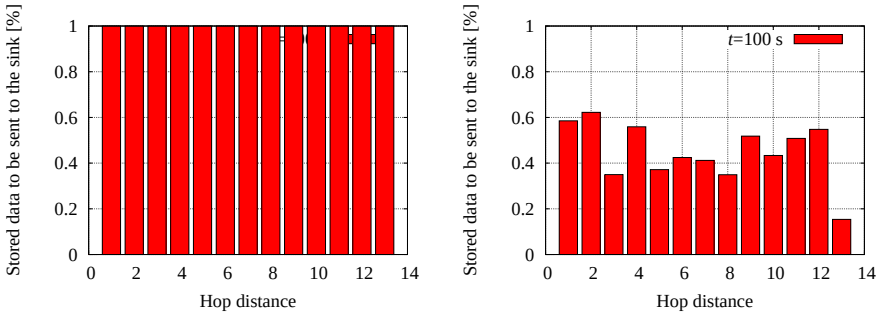
	$I = 1$ **s**	$I = 2$ **s**	$I = 4$ **s**	$I = 8$ **s**
$R = 1$	33%	50%	68%	82%
$R = 3$	49%	74%	95%	100%
$R = 5$	50%	74%	95%	100%
$R = 7$	55%	80%	100%	100%

Table 2.3: Percentage of retrieved data.

deliver more data than the others.

2.6 Conclusion

This chapter has studied the effects of common problems of WSNs at the application layer. In particular, it has been shown how the lossy nature of WSNs, the lack of memory space on the sensors, and the constraints on energy consumption impact distributed storage systems in IoT. Two redundant distributed data storage mechanisms have been proposed in order to increase the resilience and storage capacity of a WSN against node failure and local memory shortage. The first mechanism is a



(a) Data to be sent to sink $R = 1$.

(b) Data to be sent to sink $R = 7$.

Figure 2.26: Data that is sent to the sink in the retrieval phase, for various values of R: (a) $R = 1$ (no redundancy) all stored data is to be sent. (b) $R = 7$ nodes closer to the sink are going to send more data than the others. Data retrieval period is set to 100 s.

greedy hop-by-hop replication scheme which distributes data to the direct neighbor with most available space. The second mechanism adds knowledge from the underlying routing topology for an efficient data placement, by storing data at nodes closer to the root. The performance of both mechanisms has been evaluated extensively through simulations and real experiments. The routing-based mechanism, being built on top of RPL, leads directly to the implementation of a complimentary data retrieval mechanism, whose performance has been evaluated as well. Our results show clearly a trade-off between storage redundancy (which depletes the total available storage memory) and robustness against possible node(s) failure.

The work presented in this chapter also suggests that the constrained nature of WSN impacts all layers of the protocol stack simultaneously. A deep understanding of the network topology is required to take proper decisions at higher layers. Therefore, next chapter is devoted to a comprehensive analysis of the routing layer and, in particular, to the RPL protocol, since it is the target routing protocol for IoT.

Chapter 3

Routing Layer: Optimizing RPL Routing Metric

The Routing Protocol for Low power and Lossy Networks (RPL) is the IETF standard for IPv6 routing in low-power wireless sensor networks. It is a distance vector routing protocol that builds a Destination Oriented Directed Acyclic Graph (DODAG) rooted towards one sink (the root of the DODAG), using an objective function and a set of metrics/constraints to compute the best path.

In the previous chapter, we briefly presented RPL and its application in distributed storage systems. The main focus has been on the lossy nature of WSNs and on the lack of memory space on the sensors. In this chapter, we analyze the protocol in more detail. In particular, we study how routing paths are built. We show that stable and efficient routing topology can be obtained by using information from the underlying MAC and PHY layer, for example the received signal strength (RSSI) and the radio link quality (LQI) indicators, respectively.

The focus of this chapter is on the trade-off between energy consumption and latency, from the routing perspective. The main contribution is the design of a routing metric which minimizes the delay towards the root of the DODAG, assuming that nodes run with very low duty cycles (e.g., under 1%) at the MAC layer. We evaluate the proposed routing metric with the Contiki operating system and compare its per-

formance with that of the Expected Transmission Count (ETX) metric. Moreover, we propose some extensions to the ContikiMAC radio duty cycling protocol to support different sleeping periods of the nodes.

3.1 Related Work

Energy-efficiency and latency are relevant concerns in most WSN-based applications. The radio transceiver is one of the components with the highest power consumption on a low-power wireless sensor node. For this reason, nodes of a WSN often keep their radio transceivers off as much as possible in order to prolong their batteries' lifetimes. Since a node cannot receive any data when the transceiver is turned off, a duty cycling mechanism must be used at the MAC layer to periodically turn the radio on.

Many radio duty cycling mechanisms have been proposed in the literature. They can be divided into two main categories: synchronous and asynchronous. Synchronous mechanisms require neighboring nodes to be synchronized with each other, whereas asynchronous mechanisms do not depend on any a priori synchronization. Asynchronous mechanisms can be further subdivided into sender-initiated and receiver-initiated mechanisms. In X-MAC [35], the sender uses short preambles to wake up the receiver. WiseMAC [36] uses phase-lock optimization to allow nodes to learn the wake-up phase of each other. ContikiMAC [20] achieves extremely low energy consumption by inheriting multiple ideas from existing duty cycling protocols. For a good survey on asynchronous duty cycling MAC protocols, the interested reader is referred to [37]. It is worth observing that energy-efficient duty cycling mechanisms have an impact on the latency of the system, as they cause the communicating nodes to spend more time waiting for the active periods of each other, which inevitably influences the one-hop delay and, in the case of multihop communication, the cumulative end-to-end delay.

The *Routing Protocol for Low power and Lossy Networks* (RPL) [22] is the IETF standard for IPv6 routing for low-power WSNs. It is a distance vector routing protocol that builds a Destination Oriented Directed Acyclic Graph (DODAG) rooted

towards one node of the WSN, denoted as DAG root. RPL calculates the best path between the nodes according to an objective function and a set of metrics/constraints. The current RPL implementation for the Contiki operating system adopts, as default, the Expected Transmission Count (ETX) metric. According to ETX [38], the objective function to be minimized is the expected total number of packet transmissions required to successfully deliver a packet to the ultimate destination. However, it does not take the end-to-end delay into account. In [39], authors propose Opportunistic Routing in Wireless sensor networks (ORW), an opportunistic routing mechanism that forwards packets to the first awoken neighbor offering routing progress towards the destination.

In this chapter, we propose a RPL routing metric aimed at the minimization of the average delay towards the DAG root, assuming that nodes run with very low duty cycles (e.g., under 1%) at the MAC layer. We refer to the new RPL metric as Averaged Delay (AVG_DEL) metric. We evaluate the proposed AVG_DEL metric with Cooja [17], the WSN simulator for Contiki, and we compare it with the performance of ETX. Moreover, we enhance ContikiMAC to support different sleeping periods of the nodes. We investigate the performance with the proposed AVG_DEL metric in terms of delay and throughput, comparing it directly with that of ETX and taking into account also the number of exchanged control messages. The performance is analyzed considering various communication channel quality levels.

The rest of the chapter is organized as follows. An overview of RPL is presented in Section 3.2, focusing on the objective function and on the metric of the protocol. Section 3.3 is devoted to the design of the proposed AVG_DEL routing metric. Section 3.4 presents the performance results. Finally, Section 5.7 concludes the chapter.

3.2 RPL Overview

RPL has recently emerged as the IETF standard for routing in low-power IPv6 wireless sensor networks. It is based on a DODAG anchored at one or more nodes, denoted as DAG root(s). This kind of structure with one anchor node is particularly suitable for multipoint-to-point traffic, where a DAG root is the destination of all data

packets.

Each node maintains its RPL rank towards the DAG root, which describes the depth of the node in the DODAG. In order to build and maintain the topology, RPL nodes periodically exchange routing information: in the down direction, through DODAG Information Object (DIO) messages; and, in the up direction, through Destination Advertisement Object (DAO) messages. DAO messages are used to populate the routing tables of ancestor nodes in the DAG (i.e., nodes with lower ranks), in support of point-to-multipoint and point-to-point traffic.

Routes are built according to an Objective Function (OF) and a set of metrics and constraints. For instance, a popular choice of OF aims at finding the path that provides the smallest number of links with poor quality or at minimizing the number of hops. Up to now, ROLL has specified the OF 0 (OF0) [40], where the hop count is the only routing metric adopted, and the Minimum Rank with Hysteresis Objective Function (MRHOF) [41]. MRHOF selects routes that minimize a metric, while using hysteresis to reduce churn in response to small metric changes. MRHOF works with metrics that are additive along a route, and the metric it uses is determined by the metrics RPL DIO messages advertise. For a detailed survey on the set of metrics and constraints defined by ROLL, the interested reader is referred to [42].

3.3 Design

We target a RPL network composed of nodes with different duty cycles. The DAG root is assumed to keep the radio always on. Our goal is to build a DODAG such that the average delay from each node towards the DAG root is minimized. This choice is reasonable, as most traffic in WSNs occurs from leaf nodes towards a common sink. Intuitively, nodes with longer sleeping intervals would cause longer delay than nodes with shorter ones.

3.3.1 Duty Cycle with ContikiMAC

After recalling the basics of ContikiMAC, we then describe possible enhancements to support different sleeping periods of the nodes.

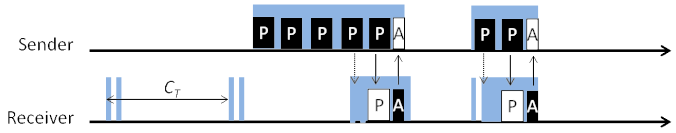


Figure 3.1: ContikiMAC: nodes periodically wake up to listen for incoming transmissions. The sender learns the wake-up phase of the receiver after reception of an ACK. The cycle time is C_T.

ContikiMAC is an asynchronous, sender-initiated radio duty cycling protocol [20]. In order to transmit a packet, a sender repeatedly sends its packet until it receives a link layer acknowledgment (ACK) from the receiver. On the other side, the receiver periodically wakes up to listen for packet transmissions from neighbors. If a packet transmission is detected during a wake-up, the receiver keeps its radio transceiver on to be able to receive the packet. After complete reception, the receiver sends a link layer acknowledgment to the sender. In terms of delay, the sender has to wait for the receiver to wake up. Additionally, after a successful transmission, ContikiMAC learns the wake-up phase of the receiver and subsequently needs to use fewer transmissions. An illustrative scenario is depicted in Fig. 3.1.

In its current implementation for Contiki 2.6, ContikiMAC assumes that all nodes have the same wake-up interval, in the following simply denoted as *cycle time* (C_T, dimension: [s]). Therefore, the number of wake-ups per second is $1/C_T$ (dimension: [Hz]). However, nodes with different cycle times ($C_{T,i} \neq C_{T,j}$, with $i \neq j$) cannot communicate. In particular:

1. Broadcast packets are sent repeatedly during a full cycle time C_T to ensure that all neighbors receive it: nodes with a longer cycle time may not receive the packet.

2. If the wake-up phase of the destination is unknown, the sender repeatedly transmits its packet until it receives an ACK. If no ACK is received within a cycle time, the destination is assumed unreachable.

3. A node cannot estimate the wake-up phase of neighbors with different cycle

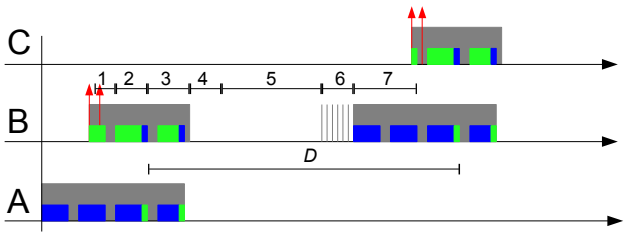


Figure 3.2: Time spent, by node B, to forward a packet from node A to node C. Of all components of the delay, only the waiting time (5) can be minimized.

time, since the cycle time is unknown.

We solve the first two problems by sending a packet repeatedly during the maximum cycle time amongst all nodes of the network, in order to ensure that all nodes receive it. This solution slightly increases the power consumption of broadcast transmissions. A more efficient approach would consist in making a node select the longest cycle time amongst its neighbors and adjust it dynamically as soon as a new neighbor with a longer cycle time is detected.

As for the third issue, a node should inform its neighbors about its cycle time. Therefore, we include the cycle time of a node within a DIO, which is used by RPL to broadcast control information. In such a way, the routing layer can be aware of the underlying MAC.

3.3.2 Forwarding Delay

In Subsection 3.3.1, it has been shown that the delay on a direct link is caused by the time spent by the sender waiting for its receiver to wake up. In a multi-hop RPL network, intermediate nodes have to forward packets from leaf nodes to the DAG root. In Fig. 3.2, the components of the time spent to forward a packet are shown in detail. Node B forwards a packet from node A to node C. The forwarding delay is indicated with D. It corresponds to the time interval from the instant at which node B receives the packet from node A to the instant in correspondence to which the packet

is successfully delivered to C. Such interval, on average, equals the interval between the wake-up times of B and C, respectively, and is thus the sum of 7 contributions: (1) time interval of a partial reception of the packet; (2) time interval of a complete reception; (3) time interval for the reception of any additional packets; (4) time spent for internal processing; (5) waiting time till node C's wake-up; (6) backoff time to check channel availability; and (7) time interval to repeat the transmission of the packet until the receiver wakes up. Of all seven components, only the 5-th one can be minimized. Therefore, a lower bound for D is given by the sum of the intervals 1, 2, 3, 4, 6, and 7, respectively. We denote such lower bound as *Minimum Forwarding Time* (MFT). Note that if the wake-up time of nodes B and C are too close to each other, node B has to wait the next C's wake-up time to send the packet, e.g., a cycle time later. To compute the forwarding delay D between B and C, two cases can be distinguished:

1. B and C have the same cycle time. In this case, $C_{T,B} = C_{T,C}$ and the difference between their wake-up phases is constant. Therefore, D is given by

$$D = \begin{cases} \Phi_C - \Phi_B & \Phi_C - \Phi_B \geq MFT \\ \Phi_C - \Phi_B + C_{T,C} & \Phi_C - \Phi_B < MFT \end{cases} \tag{3.1}$$

where, Φ_B is B's wake-up phase, and Φ_C is C's wake-up phase, estimated by B.

2. B and C have different cycle time. In this case, $C_{T,B} \neq C_{T,C}$ and the difference $\Phi_C - \Phi_B$ is not constant—under the implicit assumption that the cycle times are not multiple of each other. Therefore, only the average delay can be calculated. As shown in Fig. 3.3, the wake-up time of node B is a uniform random variable distributed in the red area. In this case, the average forwarding delay D can be expressed as

$$D = ((C_{T,C} + MFT) + MFT)/2 = C_{T,C}/2 + MFT. \tag{3.2}$$

3.3.3 The Averaged Delay Metric

In Subsection 3.3.2, we have computed the forwarding delay D on a single link. In this subsection, we introduce the Averaged Delay (AVG_DEL) routing metric. The

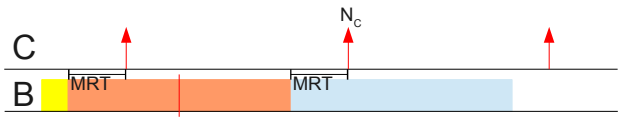


Figure 3.3: Node B's wake-up time with respect to node C's wake-up. The wake-up time of node B is a uniform random variable distributed in the red area.

proposed metric is additive and corresponds to the cumulative sum of the hop-by-hop delays along the path from a node towards the DAG root. In particular, it can be expressed as

$$AVG\_DEL = \begin{cases} 0 & \text{for the DAG root} \\ AVG\_DEL_p + D_p & \\ \qquad \text{if } AVG\_DEL_p + D_p < D_{\max} & \\ D_{\max} & \\ \qquad \text{if } AVG\_DEL_p + D_p > D_{\max} & \end{cases} \tag{3.3}$$

where: AVG_DEL_p is the average delay announced by a candidate parent p; D_p is the forwarding delay between the node and its candidate parent p; and $D_{\max}$ is a maximum threshold. In order to compute D_p, the node uses: equation 3.1, if it has the same cycle time C_T of its parent; or equation 3.2, if its cycle time is different from that of its parent. In order to join an existing DODAG, a node selects the parent with the shortest delay.

3.3.4 Wake-up Phase Discovery

Neighboring nodes with same cycle time C_T have to learn the wake-up phase of each other to be able to compute the metric in equation 3.3. More generally, a node should learn the wake-up phases of all its neighbors during the construction of the DAG and chooses the best parent among all candidates. A node could send a unicast probe packet to each of its neighbors to learn its wake-up phase. However, no IP

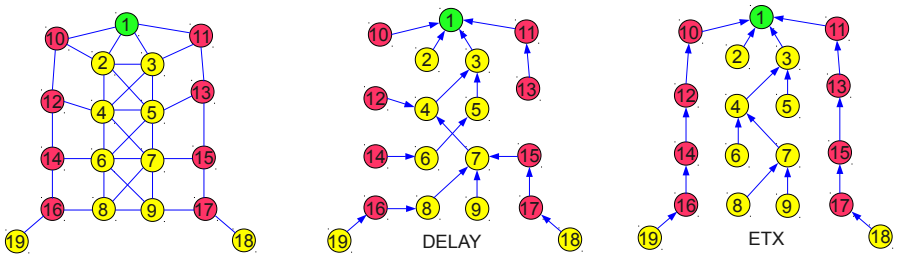


Figure 3.4: Network topology simulated in Cooja (left), with the corresponding configurations of the RPL DODAG, using different routing metrics, namely the AVG_DEL metric (center) and ETX (right).

communication can occur until the DAG is formed. To solve this issue, we let a node send a unicast MAC-layer packet to a candidate parent to discover its wake-up phase. The node can learn the wake-up phase of the candidate parent upon reception of the acknowledgment. During the discovery process, RPL assumes the worst delay for such candidate parent, i.e., $D_p = C_T + MFT$, and updates it with the correctly estimated value later, using equation 3.1.

3.4 Performance Evaluation

The proposed AVG_DEL routing metric has been implemented in Contiki 2.6 operating system and evaluated within the Cooja simulator. The performance has been compared with ETX. The tested scenario is composed of 19 *TMote Sky* nodes, shown in the left of Fig. 3.4. A central backbone is composed of nodes with cycle time C_T equal to 0.125 s, shown in yellow. Nodes with longer cycle times are placed along the borders and colored in red. The cycle time C_T of the peripheral nodes is set to 0.2 s. The DAG root is node 1 of the figure, which runs with a 100% duty cycle, i.e., its radio is always on.

3.4.1 Configuration of the DODAG

We first study the routing topology considering AVG_DEL and ETX metrics, respectively. The obtained results, in terms of DODAG formation, are shown in the center and right of Fig. 3.4. It can be observed that, in the case with ETX, routes are built in order to minimize the number of expected transmissions to reach the sink. For this reason, nodes tend to route packets through peripheral nodes, where the network density is reduced. Therefore, less interference occurs along the borders and the radio links are more reliable. This explains, for instance, why node 4 selects node 12 as preferred parent.

On the other hand, in the case with AVG_DEL metric, nodes are encouraged to route through the fast central backbone, which is composed by nodes with short sleeping periods. In particular, it can be observed that:

- nodes 2, 3, 10, and 11, connect directly to the DAG root as expected;
- peripheral nodes located far from the DAG root, e.g., nodes 15, 16 and 17, respectively, connect to the central backbone as it is faster;
- an exception to the previous fact is given by node 14, who selects node 12 as best parent—it has been found that this node has a very advantageous wake-up phase for node 14.

3.4.2 Average Delay with Lossless Links

To compute the average delay from each node towards the DAG root, in the case with perfect reliable links and *Packet Delivery Ratio* (PDR) equal to 100%, packets are generated by one node and injected in the network every 1.5 s, one packet at a time, in order to minimize the collisions. For each node, 1000 packets have been generated and collected at the DAG root, using ETX and AVG_DEL metrics. The obtained results are directly compared in Fig. 3.5. It can be observed that, in the case with AVG_DEL metric, almost all nodes improve their end-to-end delays to the DAG root. As expected, the delay does not decrease for those nodes located at one hop from the DAG root, i.e., nodes 2, 3, 10 and 11, respectively. On the other hand,

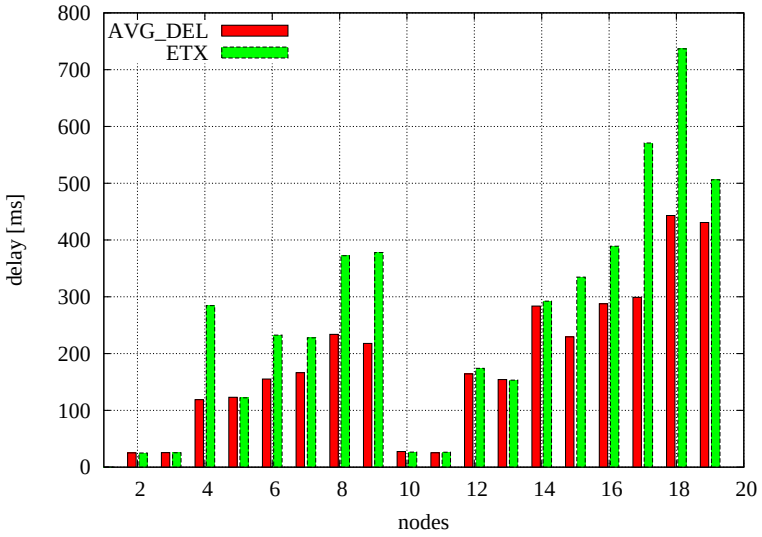


Figure 3.5: Average delay from each node towards the DAG root. A direct comparison between ETX and AVG_DEL metrics is shown.

it can be observed that the delay decreases considerably for nodes of the periphery, in particular for those which are located far away from the DAG root. For instance, nodes 17 and 18 improve their delay of about 40%. Such nodes follow a different path than in the case with ETX.

3.4.3 Impact of the Radio Link Quality

In Subsection 3.4.2, it has been shown that the use of the AVG_DEL metric significantly reduces, in the case of lossless links, the average transmission delay to the DAG root with respect to ETX. In order to simulate more realistic conditions, we now investigate the impact of the radio link quality. The Cooja simulator allows to set the probability of successful transmission, denoted as P_{Tx}, and the probability of successful reception, denoted as P_{Rx}. If a transmission fails, which happens with probability $1 - P_{\mathrm{Tx}}$, no radios receive the packet. If a reception fails, which happens with probability $1 - P_{\mathrm{Rx}}$, the receiver with this reception probability does not receive

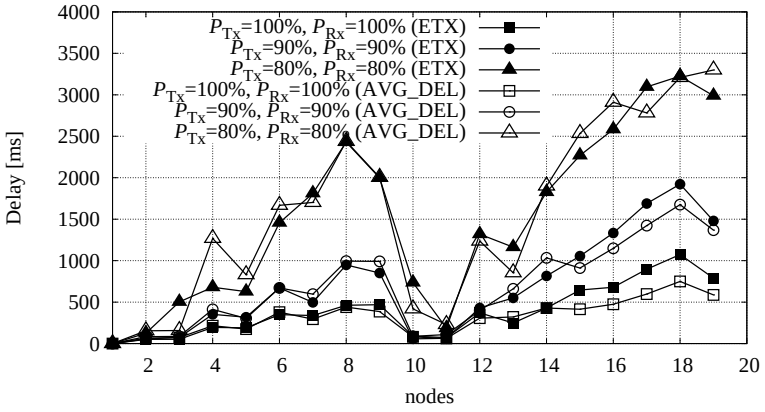


Figure 3.6: Average per-node delay (with reference to the node numbering in Fig. 3.4) towards the DAG root, for various combinations of P_{Tx} and P_{Rx}. A direct comparison between ETX and AVG_DEL metrics is shown.

the transmitted packet. We assume, under the implicit assumption of homogeneous communication conditions, that $P_{\mathrm{Tx}} = P_{\mathrm{Rx}}$. An interesting extension, which goes beyond the scope of this paper, consists in considering a non-homogeneous scenario, where the communication link quality varies across the network.

In Fig. 3.6, the average delay to the sink is shown, for each node (numbered according to the scenario in Fig. 3.4), for various combinations of P_{Tx} and P_{Rx}. The delay is averaged over 100 packet transmissions from each node to the DAG root. It can be observed that the delay increases for lower link quality. In the case with "good" links, i.e., with success probability equal to 100% and 90%, the delay with the AVG_DEL metric is still lower, as expected from the results in Fig. 3.5 (corresponding to the case with $P_{\mathrm{Rx}} = P_{\mathrm{Rx}} = 100\%$). For lower values of P_{Tx} and P_{Rx}, e.g., $P_{\mathrm{Tx}} = P_{\mathrm{Rx}} = 80\%$, the delay in the case with ETX is slightly better than that with the AVG_DEL metric. This is expected, as with ETX the link quality is taken into account, whereas the use of the AVG_DEL metric might lead to transmissions over bad quality links, with consequent retransmissions.

In Fig. 3.7, the overall network throughput, defined as the percentage of trans-

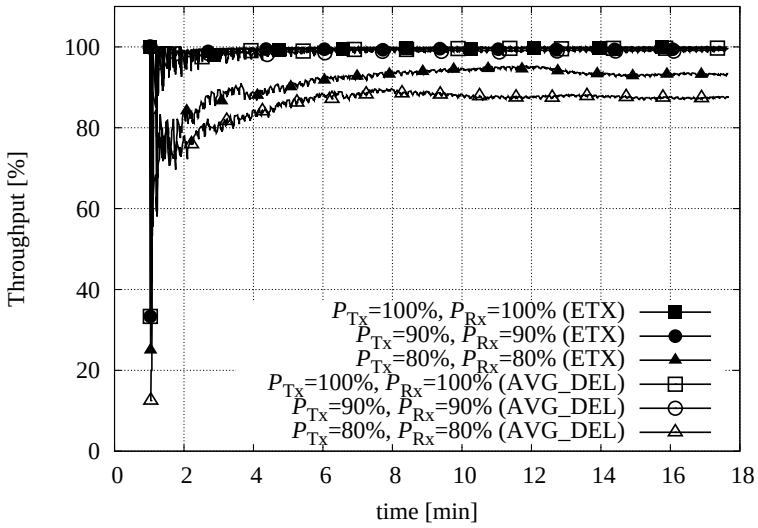


Figure 3.7: Network throughput, as a function of time, for various combinations of P_{Tx} and P_{Rx}. A direct comparison between ETX and AVG_DEL metrics is shown.

mitted packets which reaches the DAG root, is shown as a function of time, comparing directly the performance with ETX and AVG_DEL metrics. As expected, the throughput is around 100% for values of P_{Tx} and P_{Rx} greater than 90%. However, as soon as the link quality decreases ($P_{\mathrm{Tx}} = P_{\mathrm{Rx}} = 80\%$), the system with ETX outperforms that with the AVG_DEL metric. This can be explained as follows. ETX constantly monitors the radio link quality, and it switches to a more reliable link if this choice would reduce the number of expected transmissions to the DAG root. Therefore, the throughput remains stable and the delay is still acceptable, since a smaller number of retransmissions are needed on the new link. On the other hand, the AVG_DEL metric prioritizes the forwarding delay to the DAG root, which mainly depends on the relative wake-up phases of the nodes along the path, but it is less reactive than ETX in the presence of bad links.

At this point, it is of interest to evaluate the overhead of the two metrics in terms of RPL control messages. The amounts of RPL DIOs and DAOs are shown, as functions

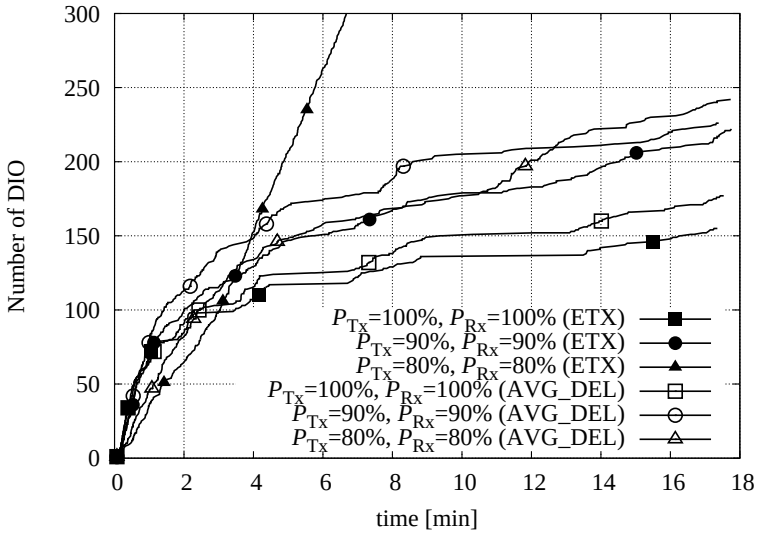


Figure 3.8: Amount of RPL DIO control messages exchanged in the network, using AVG_DEL and ETX metrics, respectively.

of time, in Fig. 3.8. In the case of poor link quality ($P_{\mathrm{Tx}} = P_{\mathrm{Rx}} = 80\%$), ETX injects a much larger number of DIOs in the network, in order to maintain the stability of the DODAG. This behavior also impacts the number of preferred parent changes performed by the nodes, which is shown, as a function of time, in Fig. 3.9. ETX quickly reacts to changes of the radio link quality, thus maintaining an acceptable throughput and latency. However, the DODAG reconfiguration has a cost in terms of overhead and, therefore, in terms of energy consumption. On the other hand, the AVG_DEL metric is less influenced by changes to the link quality level; therefore, it has a lower throughput.

3.5 Conclusions

RPL is the IETF standard for IPv6 routing in low-power WSNs. In this chapter, we have proposed a new routing metric for RPL that minimizes the average delay towards

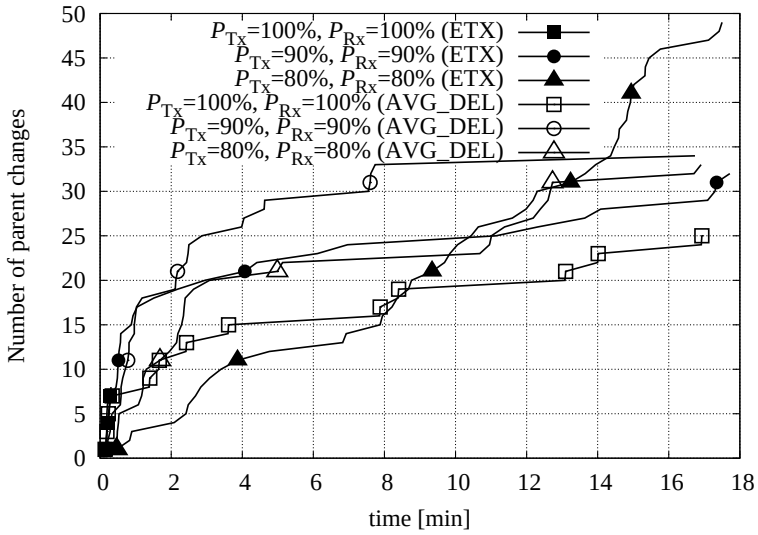


Figure 3.9: Amount of RPL preferred parent changes, using AVG_DEL and ETX metrics, respectively.

the DAG root, assuming that nodes run with very low duty cycles (e.g., under 1%). We have evaluated the proposed routing metric with the Contiki operating system and compared its performance with that of ETX. Moreover, we have proposed an enhancement of the ContikiMAC protocol to support different sleeping periods of the nodes. Our results show that, in the case of good quality communication links ($P_{\mathrm{Tx}} = P_{\mathrm{Rx}} \geq 90\%$), our solution significantly reduces the end-to-end delay of almost all nodes, especially of those located far away from the DAG root. In the case of worse channel link quality (with $P_{\mathrm{Tx}} = P_{\mathrm{Rx}} = 80\%$), the throughput with ETX remains higher, at the cost of a larger number of exchanged control messages (DIOs).

Next chapter focuses on the trade-off between energy consumption and latency at the MAC layer.

Chapter 4

MAC Layer: RAWMAC

In the previous chapter, a detailed analysis of the RPL routing protocol has been presented. It has been shown how the routing layer can leverage on information from the underlying MAC and PHY layers to optimally build routing paths.

In this chapter, we continue the discussion on energy consumption and delay in WSNs from the MAC layer point of view, with a key focus on radio duty cycling. Radio duty cycling techniques mainly consist in turning on and off periodically the radio interfaces to reduce energy consumption [43].

Without loss of generality, MAC protocols can be schematically divided into synchronous and asynchronous. Synchronous protocols align all nodes on one common schedule. While these protocols are suitable for delay-sensitive applications, they require an underlying synchronization mechanism, which consumes more energy, computation resources, and bandwidth [44]. On the other hand, asynchronous protocols do not require any control overhead for the synchronization; therefore, these protocols improve energy efficiency and are more robust to clock drifts, but they may exhibit a delay degradation due to the decoupled wake-up periods of the nodes.

In this chapter, we propose RAWMAC [45], an adaptation layer which exploits the routing layer, where the RPL protocol is used, into a management layer for asynchronous duty-cycled MAC protocols (such as ContikiMAC). The idea is to exploit the Directed Acyclic Graph (DAG) built by RPL to align each node wake-up phase

with that of its preferred parent, creating a data propagation "wave" from the leaves of the DAG to the root. This allows to significantly reduce the latency, as it depends only on small propagation delays and on the internal processing at each device. By properly configuring the phase lock mechanism of ContikiMAC, the transmitting node wakes up only when the receiving node is ready to receive the packet, so that the energy consumption is kept as low as possible. A similar approach to align duty cycles using routing information has been proposed in DMAC [46]. However, DMAC considers a synchronous MAC layer with static routing, while we aggregate an asynchronous duty cycling mechanism with a dynamic routing plane, such as RPL.

The main contributions of RAWMAC are: (i) it leverages on existing standards for unslotted constrained WSNs, such as RPL; (ii) it configures the asynchronous MAC layer wake-ups using information from a dynamic IoT-oriented routing plane; and (iii) it provides an implementation for a real deployment, based on Contiki OS.

The obtained results show that RAWMAC outperforms ContikiMAC in terms of delay for data collection, while keeping the same performance in terms of throughput and energy consumption. In addition, the creation of propagation "waves" does not impact the packet delivery ratio of the WSN, since no overhead is introduced by RAWMAC. Conversely, if we fix a delay threshold for data collection, RAWMAC allows to satisfy the requirement with a larger duty cycle, resulting in a consistent energy saving.

The structure of the chapter is the following. Section 4.1 presents the related work. Section 4.2 describes our proposed solution in detail; analytical bounds for the delay are also derived. Section 4.3 presents the performance evaluation of RAWMAC. Finally, Section 4.5 concludes the chapter.

4.1 Related Work

In order to overcome intrinsic limitations of traditional single MAC layer protocols, a number of cross-layer approaches have been proposed. The cross-layer optimization involving physical, MAC and routing layers has been considered in [47]: by jointly optimizing MAC and routing layers, this work adapts the wake up phases in

order to minimize the delays. This approach echoes the one proposed in [48], where a "wave" of propagation is created to collect alarms generated by the nodes deployed in a given monitored area. However, this approach requires an external configuration phase, since the collection path must be a priori determined. In [49], authors present an interesting solution, based on IEEE 802.15.4 and RPL, to reduce the delay for data collection. However, the proposed approach is for slotted cluster-tree networks with very low duty-cycles (several minutes), which does not fit the requirements of a surveillance system, where required delays are in the order of milliseconds. The idea of considering propagation waves has also been presented in [50], where authors focus on alarms collection and redistribution via an energy efficient broadcast. However, network topology is statically defined at network startup and never updated during time. Finally, a similar mechanism is proposed in DMAC [46]. DMAC is based on the staggering of duty cycles using information coming from the routing plane. However, this solution does not use a dynamic routing protocol, such as RPL, and it relies on a synchronous approach rather than ContikiMAC. In addition, to the best of our knowledge, no implementation of DMAC is available in Contiki, therefore no comparison tests are possible.

Current work on 6TiSCH [51] at IETF aims at providing a management layer for the synchronous and multichannel 802.15.e MAC layer. The idea is that an external Path Computation Element (PCE) solves the joint MAC and routing optimization to minimize the collection times while minimizing the global energy consumption. However, in order to provide fault-tolerance to clock drifts, nodes failures and propagation issues, we believe that managed asynchronous MAC protocols are very powerful as they can still operate in asynchronous mode in the case of failure.

4.1.1 ContikiMAC

ContikiMAC is an asynchronous, sender-initiated radio duty cycling protocol [20]. In order to transmit a packet, a sender repeatedly transmits its packet until it receives a link layer ACKnowledgment (ACK) from a receiver. The packet is repeated, in the worst case, for an entire sleeping interval, to ensure that the receiver awakes at least once during this period. The sleeping interval is denoted as *cycle time* (C_{T}, dimen-

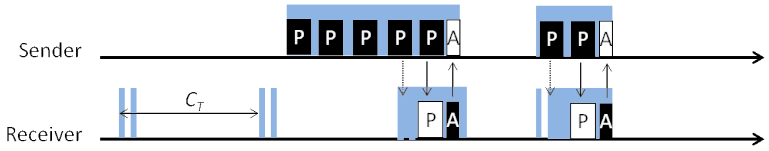


Figure 4.1: Phase lock mechanism in ContikiMAC: the sender learns the wake-up phase of the receiver after reception of an ACK. At the second transmission, the sender can decrease the number of probes. The cycle time is C_T.

sion: [s]). On the other side, the receiver periodically wakes up, with period C_T, to check for possible incoming packet transmissions. If a packet transmission is detected during a wake-up, the receiver turns its radio transceiver on to be able to receive the entire packet. After complete reception, the receiver sends the ACK to the sender. Before transmitting, the node must sense if the channel is available. In order to reduce energy consumption, ContikiMAC introduces a phase lock mechanism to learn the wake-up phase of the receiver. By recording the time of the reception of the ACK, and assuming that the receiver will wake-up at the constant interval C_T, the phase lock mechanism is able to estimate the wake-up time of the receiver. This allows to decrease the number of probes needed in the following transmissions, thus reducing the energy consumption. A graphical representation of the phase lock mechanism is shown in Fig. 4.1.

4.2 Design Principles of RAWMAC

We assume that a WSN is deployed in a surveillance system with bounded delay requirements. In such a scenario, the traffic transmitted upward, which consists of alerts, is far more critical than the one directed downward, which may include configuration requests or software updates. Nodes are organized in a tree-like structure, e.g., a RPL DAG, rooted at a sink node. In order to save energy, nodes periodically switch their radios on and off.

The goal of RAWMAC is to minimize the data collection process (i.e., the delay of upward traffic), while keeping radio duty-cycling for energy saving purposes. We

assume that the sleeping period is the same for all nodes and has been set to meet the requirements of external applications. To achieve the described goal, nodes progressively align their wake-up phases to those of their preferred parents in the RPL DAG, configuring the asynchronous phase lock mechanism of ContikiMAC. This phase shifting mechanism reduces the delay for alert collection, without requiring any overhead for the alignment of the phases.

4.2.1 Wake-up Phase Alignment

An illustrative representation of the wake-up phase alignment in RAWMAC is shown in Fig. 4.2. As long as the routing structure is established, a node shifts its wake-up phase in order to be aligned with that of its parent. More precisely, it sets the wake-up phase to the time at which it saw the last link layer ACK from its RPL preferred parent. Since the preferred parent must have been awake to receive the packet, the reception of the ACK means that it has successfully transmitted a packet within the preferred parent's wake-up window and, thus, that it has found the preferred parent's wake-up phase. We define the phase offset P_{o} (dimension: [s]) as the offset between the node's wake-up phase and the wake-up phase of its parent. Given the node's sleeping interval C_{T}, it holds that $0 \leq P_{\mathrm{o}} \leq C_{\mathrm{T}}$. The parameter P_{o} has indeed to be chosen carefully, since it has an impact on the system delay performance. If P_{o} is too short, a node relaying a packet may not be able to catch its parent's wake-up because the reception of the same packet from its child has not completed yet. If this is the case, then the child should wait the next cycle time C_{T} to be able to forward the packet. If P_{o} is too long, instead, the delay increases significantly, as the sender has to wait for the receiver to wake up to be able to transmit the packet.

RAWMAC adjusts the phase alignment every time the node receives an ACK from its preferred parent: this happens for the transmission of application data, and when RPL DAO messages are sent. In addition, RAWMAC leverages the phase lock mechanism of ContikiMAC which allows the transmitting node to send the packet only when the destination node is ready to receive it, allowing a considerable energy saving since useless packet strobing is suppressed.

With RAWMAC, nodes which are located at the same hop distance from the sink

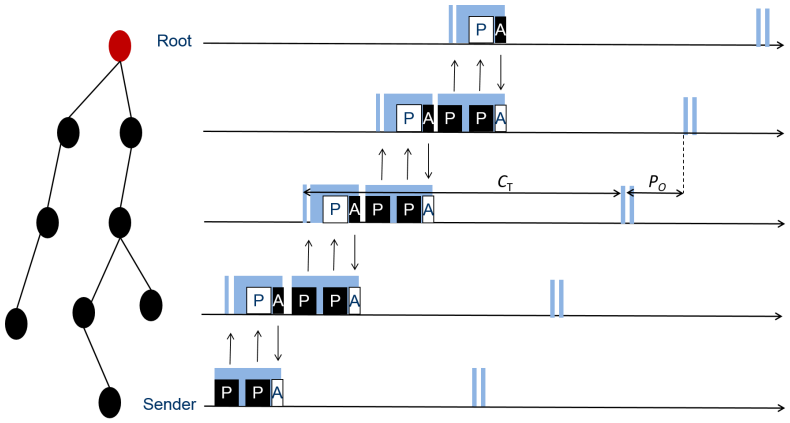
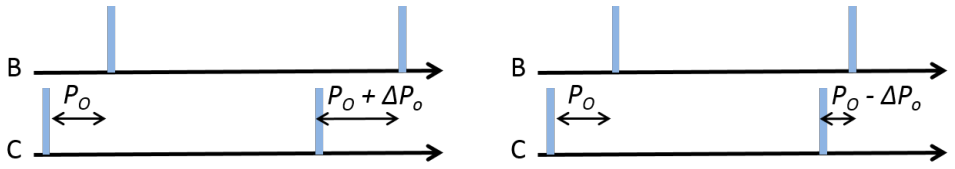


Figure 4.2: Design principle of RAWMAC: wake-up phase alignment. According to the network topology built by RPL, each node aligns its wake-up phase to that of its preferred parent in the DAG. The top node in red is the root of the DAG.

in the routing structure calibrate the same wake-up phase. In case a parent has several children, packet collisions may arise. RAWMAC handles this problem exactly as ContikiMAC, relying on a CSMA strategy. In Section 4.3, we evaluate the impact of traffic load on the delay and energy consumption performance. In particular, we highlight how the collisions affect those two performance indicators, trying to identify the maximum offered traffic that can be supported by the network with negligible performance degradation.

4.2.2 Clock Drift

The phase lock mechanism estimates the wake-up phase of the receiver with some error. The reception of the ACK does not correspond to the time at which the receiver has switched the radio on, but it comes from several contributions, such as: (i) the time spent by the receiver for a partial reception of the packet; (ii) the time for a complete reception of a packet; (iii) the time to process the packet and to generate the ACK; and (iv) the time to deliver the ACK back to the sender. In short, the reception of the ACK depends on the length of the transmitted packet. In addition, the internal



(a) (b)

Figure 4.3: Examples of clock drift which may occur after the wake-up phase alignment. In (a), the wake-up phases diverge. In (b), the wake-up phases get very close with each other. ΔP_o is the clock drift.

clocks of the nodes are not synchronized with each other, and different oscillators may cause small clock drifts, which would affect the phase synchronization as well.

RAWMAC introduces the following mechanism to prevent a node from shifting its wake-up phase if the offset with its preferred parent has not changed significantly. We define a phase offset threshold ΔP_o (dimension: [s]), with $\Delta P_o \leq P_o$, to check if the phase should be aligned or not. In particular, denoting with ϕ_C and $\phi_{B,C}$ (dimension: [s]) the wake-up phases of node C and of node C's preferred parent B (estimated by C), with $\phi_C \leq \phi_{B,C}$, respectively, node C aligns its wake-up phase to B's wake-up phase if

$$\phi_{B,C} - \phi_C \geq P_o + \Delta P_o \vee \phi_{B,C} - \phi_C \leq P_o - \Delta P_o. \tag{4.1}$$

Note that Eq. 4.1 underlies the assumption that the drift may cause a convergence as well as a divergence of the phases with equal probability, as shown in Fig. 4.3.

Similarly to P_o, the parameter ΔP_o has to be chosen carefully, since it has an impact on the system delay performance. If ΔP_o is too short, a continuous reshift of the phase would occur at any new received ACK, which may cause instabilities downward and may make children lose the synchronization with their preferred parents. If ΔP_o is too long, reshift would occur more occasionally, which may cause a node to lose the synchronization with its preferred parent.

4.2.3 Topology Reconfiguration

Changes in the network topology may occur. In particular, a node may change preferred parent whenever a neighbor has a lower path cost to the DAG root. If a node changes its preferred parent in the DAG, RAWMAC aligns its wake-up phase to that of the new preferred parent. This may cause children to lose the synchronization, and the effect would propagate to children nodes, since each child should reconfigure its wake-up phase as well.

In order to be more reactive against changes of the routing structure, we modify the phase lock mechanism of ContikiMAC. If a node does not receive an ACK after 4 consecutive transmissions—16 in ContikiMAC— it starts a new phase discovery process.

4.2.4 Delay Evaluation

Under the assumption of low generated traffic and equal duty cycles at the nodes, the transmission delay can be analytically evaluated as follows.

The phase offset used by RAWMAC to align the wake-up phase of a node with that of its parent node is denoted as P_{o}. As demonstrated in [52], the forwarding delay on a single hop d_{sh}, with uncorrelated phases, can be expressed as

$$d_{\mathrm{sh}} = C_{\mathrm{T}}/2 + P_{\mathrm{min}} \tag{4.2}$$

where P_{min} (dimension: [s]) is the minimum forwarding time needed to forward a packet to a neighbor. In Section 4.3 we will derive this parameter by simulations. In the case of RAWMAC, each node shifts the wake-up phase to match that of its preferred parent (which is, indeed, a neighbor). The shift is dictated by the phase-offset P_{o}. In this case, the upward delay $D_{\mathrm{up,R}}$ for a node which is h hops away from the sink can be expressed in two ways, depending on the value of P_{o}:

$$D_{\mathrm{up,R}} = \begin{cases} (h-1)\cdot P_{\mathrm{o}} + d_{\mathrm{sh}} & P_{\mathrm{o}} > P_{\mathrm{min}} \\ (h-1)\cdot (P_{\mathrm{o}} + C_{\mathrm{T}}) + d_{\mathrm{sh}} & 0 \leq P_{\mathrm{o}} \leq P_{\mathrm{min}} \end{cases} \tag{4.3}$$

In the second expression in Eq. 4.3, P_{o} is shorter than the minimum forwarding time P_{min}: therefore, each intermediate node has to wait an additional cycle time C_{T} to

forward the packet. For the first hop transmission, the delay is given by d_{sh}, since there is no correlation between the packet generation instant and the status of the duty-cycle of the node. For the remaining $h-1$ hops, the delay is given only by the P_o, since the propagation wave has been created.

Similar considerations hold for the delay in the downward direction, denoted as $D_{\text{down,R}}$. In particular, two cases can be distinguished in this case as well:

$$D_{\text{down,R}} = \begin{cases} (h-1)*(C_{\text{T}}-P_{\text{o}})+d_{\text{sh}} & C_{\text{T}}-P_{\text{o}} > P_{\text{min}} \\ (h-1)*(C_{\text{T}}-P_{\text{o}}+C_{\text{T}})+d_{\text{sh}} & 0 \leq C_{\text{T}}-P_{\text{o}} \leq P_{\text{min}} \end{cases} \tag{4.4}$$

As before, the delay for the first hop transmission is given by d_{sh}.

In the case of ContikiMAC, the average downward delay $D_{\text{down,C}}$, from the sink to a node placed at h hops, is equivalent to the delay in the up direction $D_{\text{up,C}}$, since the wake-up phases are uncorrelated. Thus, one obtains

$$D_{\text{down,C}} = D_{\text{up,C}} = h \cdot (C_{\text{T}}/2 + P_{\text{min}}) . \tag{4.5}$$

4.3 Performance Evaluation

We have implemented RAWMAC in Contiki 3.x and we have tested it via Cooja, a Java-based simulator for Contiki-based WSNs [17]. The simulated scenario, shown in Fig. 4.4, is composed of $N = 50$ randomly-deployed nodes. Two types of node are deployed: (*i*) a sink node, responsible for configuring the RPL DAG and collecting the data, and (*ii*) sending nodes, in charge of transmitting packets periodically to the sink and, eventually, relaying incoming packets. As soon as the network is started, the sink creates an RPL DAG which can be joined by the other nodes. In Fig. 4.4, the sink node is node 1. Each sending node generates an 8-byte length alert packet at a random instant within consecutive time slots of duration T (dimension: [s]). For instance: one packet at a random instant in $[0,T]$; one packet at a random instant in $[T,2T]$; etc. Therefore, the average packet generation rate r per node (dimension: [pck/s]) is equal to $1/T$. Packets are sent to the sink using the UDP transport protocol.

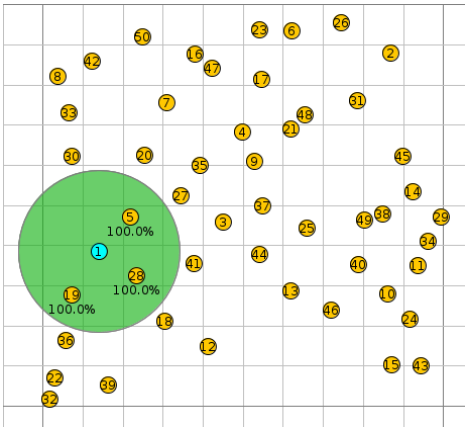


Figure 4.4: Evaluation scenario with $N = 50$ nodes. We use the unit disk radio model with distance loss. The transmission range is set to 20 m.

The duration of the simulations T_{sim} (dimension: [s]) is 5 hours. In order to bound statistical fluctuations, each simulation result is obtained by averaging over three runs (with different random seeds).

Table 4.1 summarizes the main parameters of RAWMAC, with the associated values. In the remainder of this section, when not explicitly stated, we will consider $T = 120$ s.

In order to estimate energy consumption, we rely on Powertrace, a Cooja plugin that measures and logs energy consumption for each node in the network. Our simulations have been carried out with Tmote Sky nodes, whose datasheet current consumptions for transmission and reception phases, namely, I_{Tx} and I_{Rx}, respectively, are indicated in Table 4.1. We study how the system performance is affected by (*i*) the phase offset P_{o}, (*ii*) the phase offset threshold ΔP_{o}, and (*iii*) the average packet generation period T. The performance of RAWMAC is directly compared with that of ContikiMAC in terms of delay to and from the sink, energy consumption, packet

Fixed parameters		
Duration of the simulation	T_{sim}	5 hours
Number of nodes	N	50
Radio duty cycle	C_{T}	250 ms
Transmission current consumption	I_{Tx}	20 mA
Reception current consumption	I_{Rx}	20 mA
Voltage tension	V_{DD}	3 V
Packet length	L	8 bit
Variable parameters		
Phase offset	P_{o}	$25 \div 55$ ms
Phase offset threshold	ΔP_{o}	$1 \div 9$ ms
Packet generation period	T	$25 \div 200$ s

Table 4.1: Fixed and variable parameters considered in simulations.

loss rate, number of parent changes, and number of wake-up phase realignments.

4.3.1 RPL Topology

In this subsection, we first evaluate the depth of the RPL DAG created according to the considered topology. Routing information is collected from the nodes periodically, during the simulations, in order to track, over time, changes in the routing structure. This analysis is expedient to understand, in the following subsections, the impact of traffic load and of the DAG depth on RAWMAC. In Fig. 4.5 (a), the DAG depth is shown for each node involved in the network. Since topology variations are present, we also show the standard deviation associated to the average number of hops at each node. Node 1 is not shown since it is the DAG root. It can be observed that the resulting DAG is quite stable and nodes rarely change their positions. However, nodes at a higher depth in the DAG experience a higher variability than those closer to the root. This result is confirmed in Fig. 4.5 (b), where the Probability Mass Function (PMF) of the nodes distribution, as a function of the number of hops to the

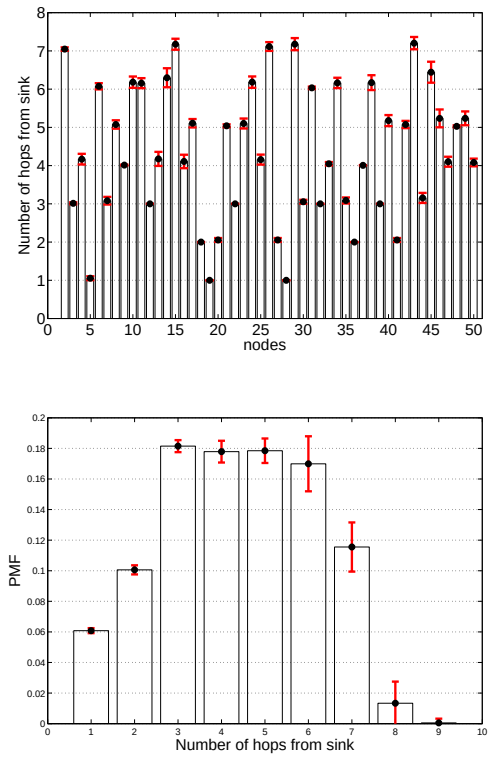


Figure 4.5: Configuration of the RPL routing topology. In (a), the average hop distance and standard deviation are shown for each node. In (b), the PMF of the number of hops (to the DAG root) is shown.

DAG root, is shown. Each mass is associated with the corresponding standard deviation. There are only a few nodes which are 8 or 9-hops away from the sink. It can also be observed that deep nodes (at least 6 hops away) are not very stable. This result is reasonable since the computation of the ETX objective function is more reliable at shorter depths. If there is an error in the measurements of the metric, this error propagates hop-by-hop, causing instability in the DAG at higher depths. For this reason, in the following we consider only a maximum DAG depth of 7 hops.

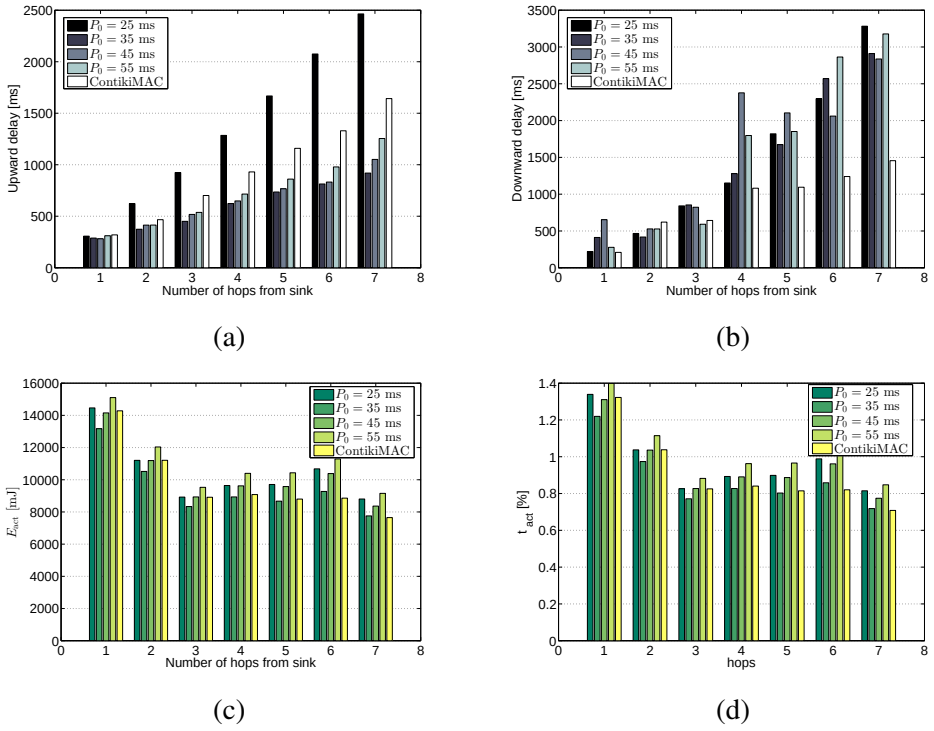


Figure 4.6: Performance analysis of RAWMAC: (a) average delay upward; (b) average delay downward; (c) energy consumption; and (d) percentage of activity of radio interface as functions of the number of hops to the DAG root, for different values of the phase offset P_o. A direct comparison with ContikiMAC is considered.

4.3.2 Impact of the Phase Offset P_o

As stated in Section 4.2.1, the phase shift impacts the delay and energy consumption performance. In Fig. 4.6 (a), we show the impact of the phase offset on the upward delay performance. In order to validate our adaptation layer, we compare RAWMAC with ContikiMAC using the phase lock mechanism. The performance results with RAWMAC have been generated considering $\Delta P_o = 6$ ms. According to Eq. (4.2), the delay at the first hop (i.e., at depth 1 in the DAG) is the same for RAWMAC and ContikiMAC, since it depends only on the packet generation instant. At higher

depths, two different behaviors can be observed. First, as expected, for proper values of P_o RAWMAC outperforms ContikiMAC, especially for increasing number of hops to the sink. In particular, for packets generated 6- or 7-hops away from the sink, the gain of RAWMAC, with respect to ContikiMAC, is over 30%. Second, we can remark that when P_o is too small (i.e., $P_o = 25$ ms), a packet cannot "ride" the same wave till the sink: at some depth, it has to wait for the subsequent wave to reach the next parent. This means that, with reference to Eq. (4.2), we can approximate $P_{min} \approx 35$ ms.

RAWMAC has been conceived for fast and delay-efficient data collection. The cost of this adaptation mechanism is a larger delay for downward traffic. In Fig. 4.6 (b), the downward delay is shown as a function of the DAG depth of the destination node, considering various values of P_o. More precisely, the sink generates a packet, on average, every $T = 120$ s and this packet is directed to one randomly-chosen node in the WSN. Besides statistical fluctuations, ContikiMAC performs like RAWMAC for the very first hops, while it outperforms RAWMAC when the destination node is deeper in the DAG, i.e., farther from the sink.

In Fig. 4.6 (c), the energy consumption of the radio interface in the active mode, denoted with E_{act} (dimension: [mJ]), is shown as a function of the number of hops to the sink, for various values of P_o. As expected, there is roughly no difference between RAWMAC and ContikiMAC, since they are both based on the phase lock mechanism that allows to turn radio interfaces on only when the receiver is expected to be available. Even for $P_o = 25$ ms, when RAWMAC is outperformed by ContikiMAC in terms of upward delay (see Fig. 4.6 (a)), the energy performance is still comparable to that of ContikiMAC, since relaying nodes postpone their transmissions without staying awake till the intended receivers wake up.

From the energy consumption curves it is possible to derive the time each node has spent in transmission and reception during the simulation. In particular, the power consumption of the radio in the active mode P_{act}, that is when the node is either transmitting or receiving, can be expressed as

$$P_{act} = V_{DD} I_{act} = V_{DD}(I_{Tx} + I_{Rx}) = \frac{E_{act}}{t_{act}} \tag{4.6}$$

where I_{act} is the current consumption due to the active phase, that is for transmitting

(I_{Tx}) and receiving (I_{Rx}) a packet, E_{act} is the energy consumption due to the active phase, and t_{act} is the time spent in the active phase, over the duration of the simulation T_{sim}. The values for V_{DD}, I_{Tx}, I_{Rx}, and T_{sim} are shown in Table 4.1. From (4.6), one can compute t_{act} and, consequently, the percentage of time spent for radio activity by a node over T_{sim}, which is investigated in Fig. 4.6 (d), for various values of P_{o}. For nodes closer to the sink, the amount of time spent with the radio interface on is approximately 1.4% with both RAWMAC and ContikiMAC. Intuitively, the deeper the position of a node in the DAG, the higher should t_{act} be. This trend is confirmed up to the third hop. However, since the network is rather dense, at 4-hops from the sink there are more contentions to transmit to the parents, resulting in a much higher radio activity. The same trend can be highlighted for ContikiMAC, confirming that the higher radio activity at 4 hops is due to topology configuration.

4.3.3 Impact of the Phase Offset Threshold ΔP_{o}

Another parameter that has a relevant impact on RAWMAC is the tolerance to the phase shift offset. Internal clocks imperfections and internal processing delays may make a node unprepared to receive a packet at a scheduled instant. If no tolerance is introduced, the sender would always need to track the changes of its parent's wake up scheduling. Clearly, this would impact on the node's children activity, since they should track as well the changed wake-up instant. As the misalignment between nodes may cause retransmissions, the phase offset threshold ΔP_{o} has an impact on the ETX metric used by RPL and may lead to network instability. In Fig. 4.7 (a), the cumulative number of parent changes experienced by the nodes in the network is shown as a function of simulation time, considering several values of ΔP_{o}. Simulations have been carried out with $P_{\mathrm{o}} = 40$ ms. Focusing on the values of ΔP_{o} which guarantee the best performance, namely, $\Delta P_{\mathrm{o}} = 5 \div 9$ ms, it can be observed that the number of RPL parent changes with RAWMAC is the same as with ContikiMAC. Therefore, it can be concluded that these changes are caused by the instability of the DAG rather than by the impact of phase shifts on the RPL metric.

In Fig. 4.7 (b), we evaluate the number of phase shifts that are experienced by the nodes as a function of the simulation time. The number of shifts for $\Delta P_{\mathrm{o}} = 5 \div 9$ ms

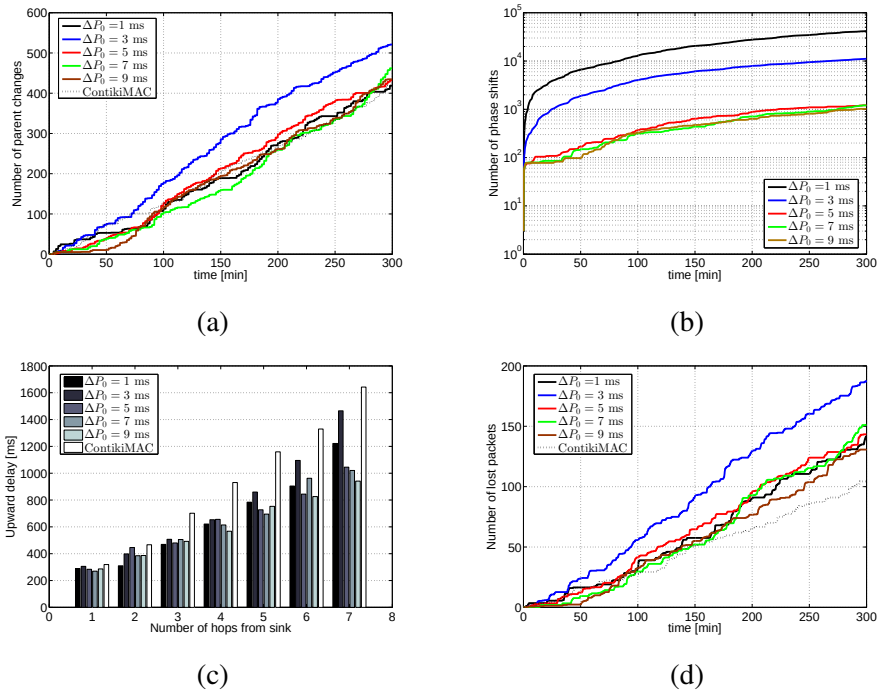
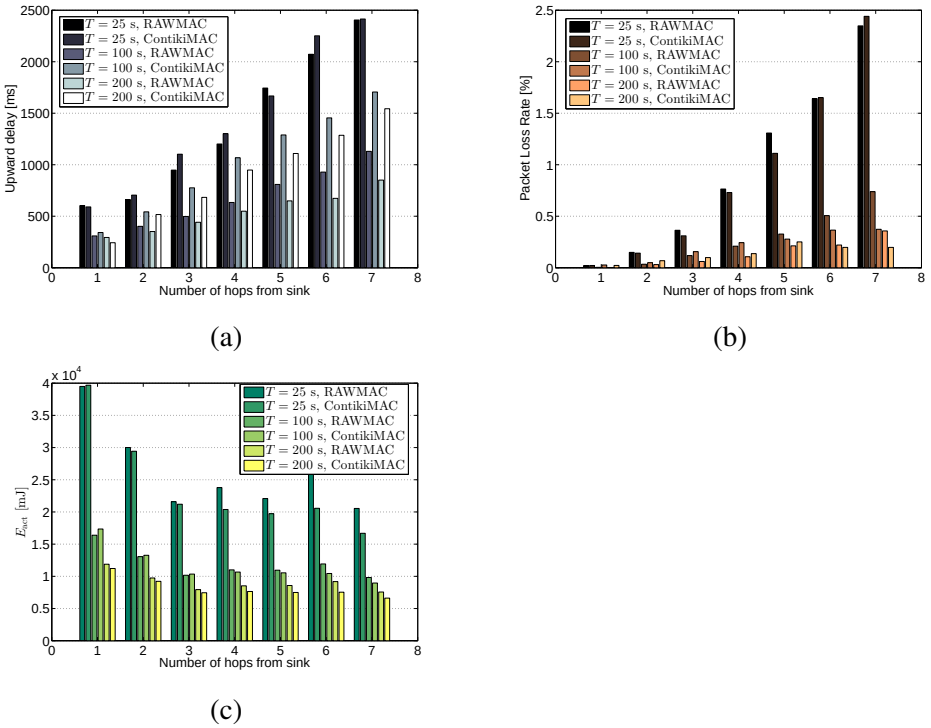


Figure 4.7: Performance analysis of RAWMAC: (a) parent changes as a function of the simulation time; (b) phase shifts as a function of the simulation time; (c) upward delay as a function of the RPL tree depth; and (d) packet losses as a function of the simulation time, for different values of the delta phase offset ΔP_o. A direct comparison with ContikiMAC is considered.

is basically the same, whereas for $\Delta P_o = 1 \div 3$ ms the number of phase shifts is one order of magnitude larger than for the other values of ΔP_o. This means that the nodes continuously try to track their parents, resulting in worse delay performance, as shown in Fig. 4.7 (c), where the upward delay is shown as a function of the depth of the RPL DAG. The effect of the phase offset threshold becomes more evident the longer is the number of hops, resulting in a higher delay. In this case, the delay is due to the misalignments introduced by the nodes trying to track the changes of their parents' phase offsets.



(a) (b)

(c)

Figure 4.8: Performance analysis of RAWMAC: (a) average delay upward; (b) packet losses; and (c) energy consumption as functions of the hop distance, for various values of the packet generation period T. A direct comparison with ContikiMAC is considered.

In Fig. 4.7 (d), the packet loss rate is shown as a function of simulation time. The losses introduced by the presence of "waves" are really limited. Even for small values of ΔP_{o} the packets only experience higher delay but no supplementary losses with respect to those experienced with ContikiMAC.

4.3.4 Impact of the Packet Generation Period T

In order to measure the impact of traffic load on RAWMAC, we vary the packet generation period T from 25 s up to 200 s. According to previous results, P_{o} and ΔP_{o}

are set to 40 ms and 6 ms, respectively. In Fig. 4.8 (a) the upward delay is shown as a function of the distance of the nodes from the sink. For small values of T, the performance of RAWMAC and ContikiMAC is the same since the delay is mainly due to retransmissions introduced by the access contention mechanism. When T increases, i.e., in low traffic conditions, RAWMAC outperforms ContikiMAC due to the propagation waves that allow to quickly deliver packets to the sink.

This behavior is confirmed by the packet loss rate shown in Fig. 4.8 (b). For small values of T, the traffic offered to the WSN is too high, so nodes that are far from the sink (i.e., deeper) experience higher losses, and packets are dropped since nodes fail to retransmit. On the other hand, when T is large, the packet loss rate for the two protocols is comparable.

From an energy point of view, as shown in Fig. 4.8 (c), the nodes closer to the sink have a larger energy consumption since they have to relay information coming from their children. When T is small, the high energy consumption is not only due to the large number of packets to be forwarded, but also to the large number of packets' retransmissions which are due to collisions.

4.4 A Practical Application: the CALIPSO Smart Parking Demonstrator

RAWMAC was also tested within the CALIPSO project, in the Smart Parking application deployed in Barcelona. In this section, we show the main collected experimental results.

4.4.1 Demonstrator Overview

The test facility was situated in a corner street in Barcelona. The corner is a Parking Load Zone that is restricted for loading and unloading activities, with a maximum stop time of 30 minutes. The system consisted of: (i) 6 sensor nodes, which were buried into tarmac and placed about 2 m a part of each other (6 nodes were sufficient to cover the entire corner); and (ii) a base station, installed on a lamppost above

Figure 4.9: Picture of the Smart Parking test facility.

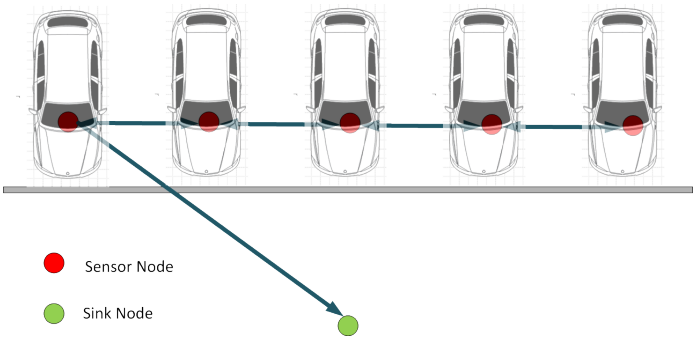


Figure 4.10: Test deployment topology, corresponding to the real scenario in Figure 4.9.

the corner, which acted as sink node. Two illustrative representations can be seen in Figures 4.9 and 4.10, respectively.

Sensor Node

Each sensor node was composed by two interconnected motes: (i) a Parking Detection Sensor Board, provided by Worldsensing, and (ii) a standard Tmote Sky mote. The communication between the Parking Detection Sensor Board and the TmoteSky was done through a 2-way serial port, as shown in Figure 4.11.

The Parking Detection Sensor Board carried out sensing: it was composed by

a digital magnetic sensor and a microcontroller able to detect the presence of a car parked above it. It also contained a sub-GHz radio for point-to-point communication purposes. Every time a change in the state of the parking place was detected, the sensor sent a predefined message through the serial port to the communication system, i.e., to the Tmote Sky.

The Tmote Sky ran the CALIPSO protocol stack and provided connectivity. It received the message from the serial port and forwarded it to the base station. The CALIPSO protocol stack included all software modules developed within the project on top of Contiki OS. In particular, the following modules were evaluated and compared:

- RAWMAC;
- ContikiMAC;
- RPL;
- Reactive RPL [53], which enhances RPL with an efficient mechanism for on-demand route search — this mechanism works better in networks with variable link quality or event with disappearing motes;
- ORPL [54], which is an extension of RPL designed to work on heavy duty-cycled networks.

Base Station

The base station had two different components: (i) a TmoteSky, which acted as RPL border router, and (ii) a standard embedded Linux PC, which acted as server. The RPL border router created the RPL DODAG that sensor nodes should connect to. It also implemented the same protocol stack deployed in the sensor nodes, namely, RAWMAC, ContikiMAC, RPL, Reactive RPL, and ORPL.

The Linux PC was responsible of the collection, analysis, storage and visualization of the data gathered from the sensor nodes. In particular, its software consisted of the following main modules:

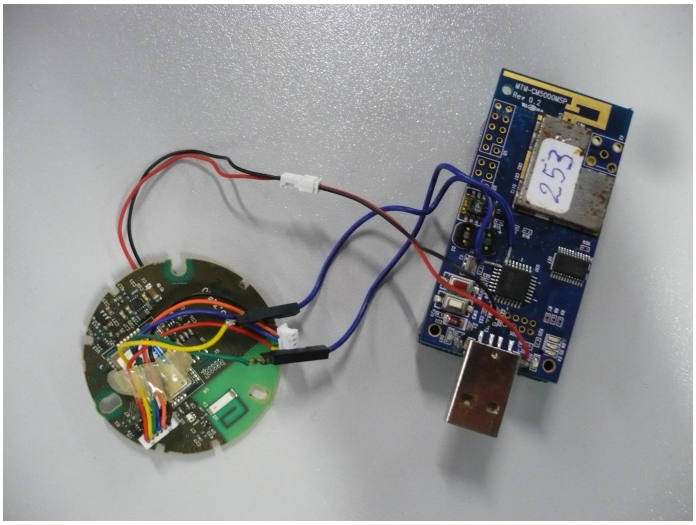


Figure 4.11: Sensor board (left) and TMote Sky (right) connected. The sensor board periodically senses the electromagnetic field to detect the presence or absence of a car in the parking place. This information is sent to the TMote Sky (using a standard serial port) in order to be forwarded to the base station (and the cloud servers) by using the CALIPSO communication stack.

- CoAP Server, which received CoAP messages from the sensor nodes, extracted the payload and passed it to the Resource Manager module and to the HTTP Server;

- Resource Manager, which received CoAP payload from the CoAP Server and stored data in a local database;

- HTTP Server, which was used for the visualization of the statistics.

The whole system architecture is shown in Figure 4.12.

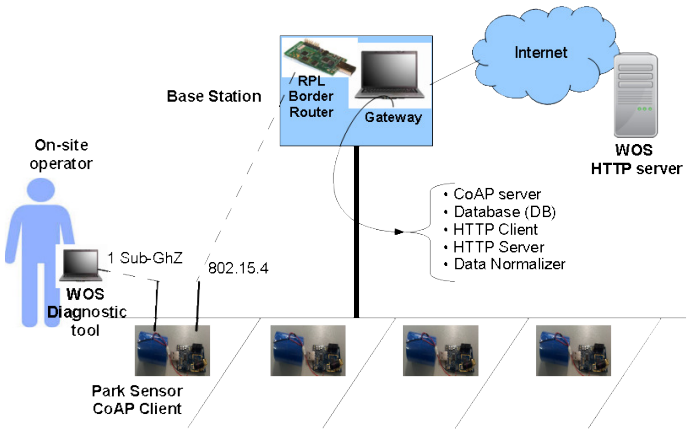


Figure 4.12: System architecture of the Smart Parking demo.

4.4.2 Performance Evaluation

The main system indicators chosen for the performance evaluation of the demo were the following:

- energy consumed by the node;
- RPL information (parent id, ETX metric);
- packet delivery ratio (PDR);
- routing protocol transmission overhead (number of non-application packets transmitted by the routing layer to maintain the routing tree);
- routing tree convergence time;
- node link latency;
- node hop count.

The above indicators were sent as CoAP resources by all sensor nodes to the CoAP server running in the base station.

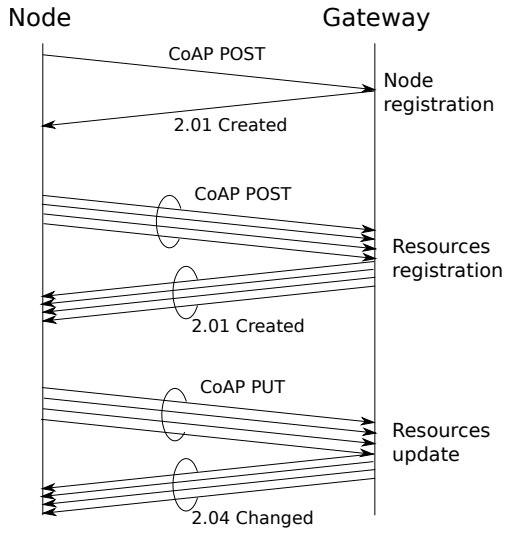


Figure 4.13: Communication scheme between the nodes and the gateway (base station).

Experimental setup

In the performed tests, sensor nodes formed a RPL tree with a maximum depth of three hops to the sink. In some cases, the RPL topology changed because of several cars parking and leaving quickly.

In Figure 4.13, the registration procedure followed by the nodes, before starting the collection of the statistics, is depicted. Once a node enters the network, it sends a CoAP POST request to a specific resource of the gateway which, in turns, replies with a 2.01 CREATED message with an indication of the ID assigned to the node and the URI that the node must use to collect statistics. At this point, the node can start registering each resource. Exploiting the URI received at the previous step, the node builds a new URI for each resource and issues a new CoAP POST request to the gateway to notify it of the starting data collection. The gateway replies with a 2.01 CREATED message for each POST. Once this phase is terminated, the node can update the status of the resources via a CoAP PUT message, to which the gateway replies with a 2.04 CHANGED message. These updates were sent periodically by a

node. Clearly, this frequency has an impact on energy consumption and network traffic. In order to have sufficient granularity for the tests, the data collection frequency was set to 30 s.

The CoAP resources collected by a sensor node were:

1. presence of a node;
2. number of transmitted packets;
3. overhead messages;
4. consumed energy;
5. preferred parent.

In addition, the RPL border router notified the base station about the convergence time of each node and the number of hops traversed by a packet issued by a node.

From all collected information, the base station extrapolated statistical characteristics of: the energy consumption, the tree depth of a node, the total per-node overhead, the packet delivery ratio, and the convergence time. In addition, to measure the round-trip delay, the base station cyclically sent ping messages to the sensor nodes. Once the experiment was concluded, the base station stored all the statistics in a log file for further processing.

Simulations

In order to validate the code and the collected results, the system was first simulated in Cooja before the final deployment. In Cooja, the CALIPSO stack ran as well all radio communications and devices. A model for the Parking Detection Sensor Board was written from scratch in order to have the complete system running in the simulation.

For this evaluation phase, the following five scenarios were compared:

1. RPL + NullRDC (i.e., radio always on);
2. RPL + ContikiMAC;

Protocol	Delay [ms]
RPL + NullRDC	156.0
RPL + ContikiMAC	576.7
RPL + RAWMAC	612.0
ORPL + ContikiMAC	459.9
RRPL + ContikiMAC	1780.8

Table 4.2: Average delay (dimension: [ms]).

3. RPL + RAWMAC;
4. ORPL + ContikiMAC;
5. RRPL + ContikiMAC.

Since in ORPL a node opportunistically transmits to the first available parent, it follows that the concept of tree depth is meaningless for ORPL. In order to provide a fair comparison, in the ORPL case we averaged the results over all the nodes in the network.

In Table 4.2, the delay performance is investigated. It can be observed that the lowest delay is achieved with NullRDC: this is obvious, since there is no duty cycling and the radio interfaces are always on to receive packets. Since with real nodes there is no way to accurately measure the time it takes to send a message from a node to the gateway, the delay measurements have been carried out via ping messages issued by the gateway, collecting statistics on two-ways delay. For such a reason, RAWMAC has better performance than ContikiMAC for upward traffic (since RAWMAC aligns wake-up phases of nodes), but this is achieved at the price of a longer delay for downward data transmission. The delay of ORPL, instead, is lower since a node transmits, upward, to the first available parent, without waiting for a predefined parent that relays data to the destination. Finally, RRPL is the one with longest delay since nodes need to find a route before sending data.

In Figure 4.14, a comparison between these protocols in terms of (a) overhead, (b) energy consumption, and (c) packet delivery ratio, averaging the results over all

the nodes in the network, is provided. In terms of overhead (Figure 4.14(a)), ORPL broadcasts more packets than RPL-based protocols since it relies on DIO messages (as RPL) together with some protocol-specific ORPL broadcast messages. RRPL instead is the one transmitting more control packets since it requires to create a route to transfer data. In terms of energy consumption (shown, in logarithmic scale, in Figure 4.14(b)), the always-on interface leads to significant energy consumption. In addition, due to the larger number of exchanged packets, ORPL and RRPL have a slightly higher energy consumption than RPL-based protocols. Finally, all protocols have the same performance in terms of packet delivery ratio (Figure 4.14(c)) which remains close to 1 in all cases.

Field Trials

In order to reproduce a multihop scenario in the installation described above, we tuned the receiver sensitivity of the border router to -65 dBm, so that only the two nodes close to the pole can communicate directly with it. The other nodes, instead, could only reach the sink via a multi-hop path. The target setup is also shown in the Figure 4.10 where the arrows represent the ideally preferred next hop device for the given node. We remark that this by no means implies that the other nodes cannot be reached by the given node: in fact, nodes close to the preferred parent are reachable and the given node may use them, instead of the preferred parent, for communication. The circumstances in which this would happen depend on changes in the environment, such as: cars parking on the monitored parking spots; cars passing or stationed nearby; interferences from a nearby public WiFi hotspot; etc.

Since tests have been performed in a public street, the interference of ongoing traffic, radio interference from personal devices, and weather conditions are common problems and cannot be controlled. For such a reason, unexpected behaviors may arise, such as: time-varying weaker connectivity; jitters; extra communication overhead; extra energy consumption.

Since nodes run in a real environment, we carried out a performance comparison, in terms of delay, overhead, energy consumption, and packet delivery ratio, between the protocols developed in CALIPSO and the standard protocols available in

Protocol	Delay [ms]	Standard deviation [ms]
RPL + NullRDC	99.3	66.7
RPL + ContikiMAC	441.4	286.0
RPL + RAWMAC	321.3	141.8
ORPL + ContikiMAC	542.3	886.2
RRPL + ContikiMAC	840.3	1025.1

Table 4.3: Field evaluation of average delay (dimension: [ms]) and standard deviation (dimension: [ms]).

Contiki. The evaluated multi-layer protocol combinations are those outlined in Subsection 4.4.2: (i) RPL+NullRDC; (ii) RPL+ContikiMAC; (iii) RPL+RAWMAC; (iv) ORPL+ContikiMAC; and (v) RRPL + ContikiMAC. In order to stress the network behavior, especially during the startup phase, we present results over the first 40 minutes of the experimental trial, since at that point the network has already converged and its behavior is stable.

In Table 4.3, we compare the delay performance of the protocols. For each protocol, we show the average delay and the standard deviation. As in Table 4.2, we measured the round-trip delay, since it is not possible to have clock synchronization between nodes and to know the exact packet transmission instant. The best performance is obtained with NullRDC, since radio interfaces are always on. RAWMAC, basic ContikiMAC, and ORPL, have approximately the same delay performance. In fact, the benefit of phase alignment in RAWMAC is partially lost since ping messages accumulate the whole round-trip delay to reach the node and return to the sink. In ORPL, instead, the low spatial density of nodes limits the impact of opportunistic transmissions. The differences between the considered multi-layer protocols are mainly due to the number of cars that arrived or left during the experience. In fact, each car arrival perturbs the link conditions and requires some additional time to readapt the routing topology. Finally, RRPL, because of its reactive nature, experiences the highest delay.

We want to point out that some experiments, namely those with RRPL + Contiki-

MAC, ORPL + ContikiMAC, and RPL + ContikiMAC, have been strongly affected by the presence and the disappearance of cars. These three protocols, in fact, have been validated during a normal working day with several cars and vans parking in the demo site. We noticed that some delay peaks correspond to the arrival of shipping-service vans that strongly interfered with the transmission of packets. On the contrary, RPL + NullRDC and RPL + RAWMAC have been carried out during a local holiday, so the number of car movements were considerably smaller and the experimental conditions were quite stable. Finally, the experience with ORPL + ContikiMAC has been carried out during the local holiday as well: the higher standard deviation can be attributed, owing to the opportunistic nature of the protocol, to the fact that during the experience some cars parked above the nodes closer to the gateway.

In Figure 4.15, we compare the five protocols in terms of overhead, energy consumption, and packet delivery ratio. Referring to Figure 4.15(a), the lowest overhead among the protocols running RPL is achieved with NullRDC, since nodes are always on and the effects of duty cycles on the RPL metric is absent. A similar overhead performance is achieved with ORPL, where only a few broadcast messages are transmitted to diffuse information about each node phase. As shown in Figure 4.15(b), the fact that ORPL uses many broadcast messages leads to an energy consumption similar to that obtained with RPL+ContikiMAC and RPL+RAWMAC, where both unicast and broadcast messages are sent, thus balancing the lower overhead. RRPL, instead, incurs the price of route creation. Finally, RPL+NullRDC has an energy consumption ten times higher since nodes have interfaces always on. Concerning the packet delivery ratio, as shown in Figure 4.15(c), even if we experienced some losses for ORPL + ContikiMAC, RRPL + ContikiMAC, and RPL + ContikiMAC due to the above-mentioned experience conditions, all the solutions have a very good packet delivery ratio performance, which makes the protocols developed in CALIPSO suitable for the use in realistic applications.

Simulation and experimental results allow us to draw some conclusions about the protocols we considered and their applicability in smart parking applications.

RPL + NullRDC: This protocol has an attractive performance in terms of delay and PDR since nodes, being always on, can receive any incoming message.

However, the extremely elevated energy consumption makes it definitively unsuitable for smart parking applications that target node lifetime maximization.

RPL + ContikiMAC: This solution is attractive for smart parking applications. However, it is not suited to converging (towards a collector) traffic, since it has been designed for a generic traffic pattern, without a preferred traffic direction.

RPL + RAWMAC: RAWMAC specifically targets upwards traffic, since it aligns the wake-up phase of a node with that of its preferred parent. The obtained performance results show that this protocol is well suited to smart parking applications since it has reasonable energy consumption, small delay, and high PDR.

ORPL + ContikiMAC: ORPL is another potentially good candidate for smart parking applications, as it conceived to guarantee low delay and it proved to have a reduced overhead and very good energy consumption. However, ORPL performs best in the presence of high node spatial density. In our testbed, we only have 6 nodes communicating, so the benefits of ORPL are not completely evident.

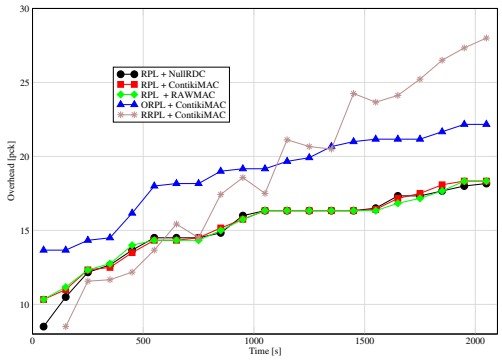
RRPL + ContikiMAC: Since RRPL is a reactive protocol, its higher signalization is not suited for pure data collection in smart parking applications. However, this protocol can be exploited for any application where rapid topology reconfiguration is required, e.g., in the presence of node mobility.

4.5 Conclusions and Perspectives

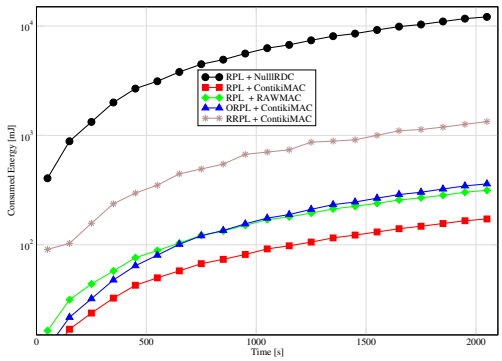
In this chapter we have presented RAWMAC, an adaptation layer for minimizing the alert collection delay in monitoring systems. RAWMAC uses routing as a management module for asynchronous MAC layers, which are more robust to clock drifts, in order to create a wave of propagation from the leaves to the root of a routing tree. We have exemplified this adaptation layer on RPL and ContikiMAC, as these routing protocols are emerging as standards de facto for the Internet of Things. Performance

results have shown that RAWMAC largely outperforms the standalone ContikiMAC in terms of delay for upward traffic. From energy and packet losses points of view, instead, RAWMAC under the same configuration parameters has the same performance of pure ContikiMAC. This means that, given a target maximum delay, RAWMAC allows to largely reduce the duty-cycle of the nodes compared to ContikiMAC, resulting in a remarkable energy saving. In addition, we have studied the impact of the parent changes due to RPL and of the misalignment due to clock drifts. RAWMAC proves to be sufficiently reactive to track all these changes without affecting the packet delivery ratio, that remains equivalent to that of ContikiMAC.

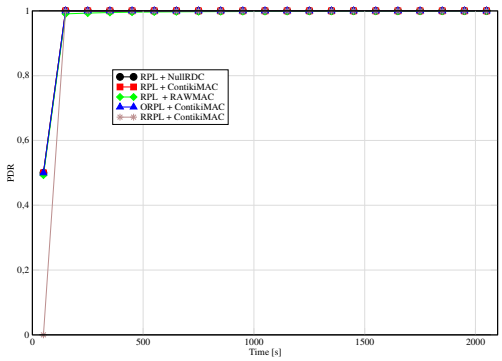
As future work, we plan to extend RAWMAC to take into account efficient broadcasting mechanisms, based on the local neighborhood, as well as quicker ways to discover the wake-up phases of neighbors. Another relevant research direction is the design and performance evaluation in the presence of heterogeneous radio duty-cycling across the nodes.



(a) Overhead

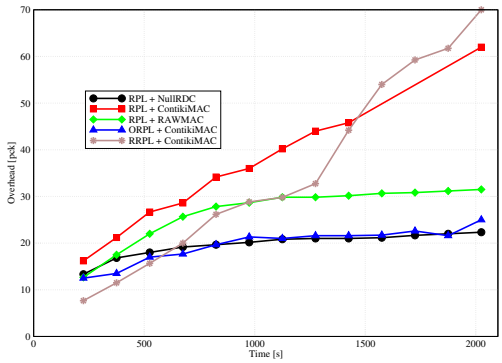


(b) Energy consumption

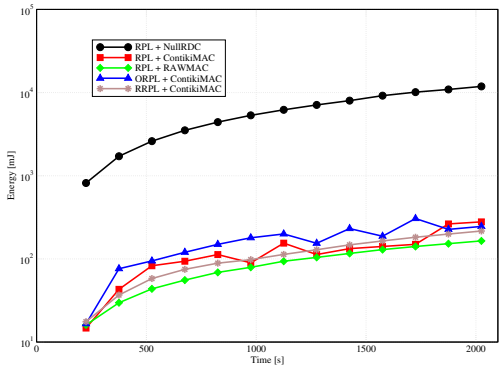


(c) Packet delivery ratio

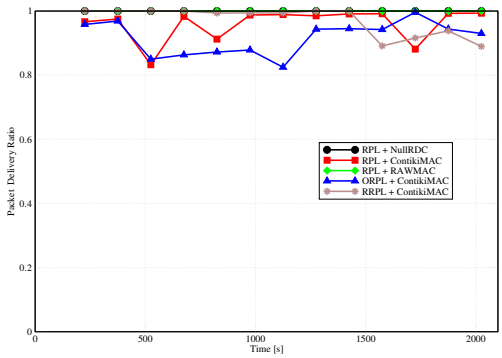
Figure 4.14: Performance comparison of the five combinations of protocols in terms of (a) overhead, (b) energy consumption, and (c) packet delivery ratio averaged over the nodes in the network.



(a) Overhead



(b) Energy consumption



(c) Packet delivery ratio

Figure 4.15: Performance comparison through on-the-field experiments of the five combinations of protocols in terms of (a) overhead, (b) energy consumption, and (c) packet delivery ratio averaged over the nodes in the network.

Chapter 5

Security: Authorization Service with OAuth

So far we have addressed common issues of WSNs at several layers of the IoT-based architecture, namely, the application, the routing, the MAC layer. In this final chapter we show how the aforementioned issues have a significant impact on security, which is a relevant topic in IoT-based sensor networks. Security does not fall into a specific layer of the architecture, but it can be applied to all layers in a vertical way. In particular, this chapter deals with authorization and privacy for WSNs.

A sensor node (or smart object) may be equipped with several sensors, keeping on collecting information from the environment. Access to the data gathered from the sensors may be permitted to some users (e.g., an application that monitors vibrations of a building should be allowed to get data from the magnetic sensors deployed around), but it may be forbidden to some others (e.g., a person entering the building). Some important questions arise: who is authorized to access sensor data? How can the node check if a user is allowed to access its own resources? The node may check on its own the credentials of potentially all the incoming requests to decide whether to serve them or not. In a few words, this requires the node to implement an authorization logic itself, which may however waste memory footprint, computational capabilities as well as energy consumption. Alternatively, the node could delegate the

job to someone else.

We propose a flexible service-based architecture for WSNs, which delegates the authorization to an external service. We adopt the standard Open Authorization (OAuth) protocol [55] to provide secure authorization. We compare our approach with the one which uses OAuth-based authorization directly on the sensor nodes. Performances are evaluated through Cooja-based simulations emulating real Zolertia Z1 hardware platforms.

5.1 The Authorization Problem in IoT-based WSNs

The evolution of online services, such as those enabled by social networks, has had a relevant impact on the amount of data and personal information disseminated on the Internet. Furthermore, it has determined the birth of applications that rely on the disseminated information in order to offer new services, such as aggregators. The information owned by online services is made available to third-party applications in the form of public Application Programming Interfaces (APIs), typically using HTTP [56] as communication protocol and relying on the REpresentational State Transfer (REST) architectural style. The possibility that someone else, besides the entity which generates the information and the service that is hosting it, can access this information has brought up concerns about the privacy of personal information, since the trust is no longer a pairwise relationship but possibly involves other parties, which may be unknown at the time of service subscription.

Open Authorization (OAuth) is an open protocol to allow secure authorization from third-party applications in a simple and standardized way [55]. The OAuth protocol provides an authorization layer for HTTP-based service APIs, typically on top of a secure transport layer, such as HTTP-over-TLS (i.e., HTTPS) [57]. OAuth defines three main roles in the above scenario:

- the *User* (*U*) is the entity which generates some sort of information;
- the *Service Provider* (*SP*) hosts the information generated by the users and makes it available through APIs;

- the *Service Consumer* (*SC*), also referred to as "client application," accesses the information stored by the *SP* for its own utilization.

In order to comply with the security and privacy requirements, *U* must issue an explicit agreement that some client application can access information on its/his/her behalf. This is achieved by granting the client an access token, containing *U*'s and *SC*'s identities, which must be exhibited in every request as an authorization proof. The OAuth 2.0 protocol is the evolution of the original OAuth protocol and aims at improving the client development simplicity by defining scenarios for authorizing web, mobile, and desktop applications [58]. While connecting to existing online services is a simple task for client application developers, implementing an OAuth-based authorization mechanism on the *SP*'s side is a more complicated, time-consuming, and, potentially, computationally intensive task. Moreover, it involves the registration of both users and client applications, and the permissions that *U*s grant to *SC* applications and integrating with authentication services.

Within the IoT world, the IETF CoRE Working Group has defined the Constrained Application Protocol (CoAP) [59], a generic web protocol for RESTful constrained environments, targeted to Machine-to-Machine (M2M) applications, which maps to HTTP for integration with the existing web.

While the use of the OAuth protocol has little impact, in terms of processing and scalability, on conventional Internet-based services, its adoption in IoT has to deal with the limitations and challenges of constrained devices. The limited computational power of Smart Objects may not be sufficient to perform the cryptographic primitives required for message authentication, integrity checks, and digital signatures, which may have a negative impact on energy consumption. Moreover, if the access permissions for the services provided by the Smart Object reside on the Smart Object itself, it could be extremely hard, if not impossible, to dynamically update them once they are deployed on thousands of nodes (e.g., the Fastprk[1] system shown in the previous chapter).

In this chapter, we present a novel architecture targeted to IoT scenarios for an external authorization service based on OAuth, denoted as *IoT-OAS*. The delegation

[1] http://www.fastprk.com/

of the authorization to an external service, which may be invoked by any subscribed host or thing, affects:

1. the time required to build new OAuth-protected online services, thus letting developers focus on service logic rather than on security and authorization issues;

2. the simplicity of the Smart Object, which does not need to implement any authorization logic but must only invoke securely the authorization service in order to decide whether to serve an incoming request or not;

3. the possibility to dynamically and remotely configure the access control policies that the SP is willing to enforce, especially in those scenarios where it is hardly possible to intervene directly on the Smart Object.

Experimental results are presented highlighting the existing trade-off between communication and processing costs.

The rest of this chapter is organized as follows. Section 5.2 discusses related work in the field. In Section 5.3, the *IoT-OAS* architecture is presented. In Section 5.4, a few *IoT-OAS* application scenarios are presented. In Section 5.5, an extensive experimental performance evaluation of the proposed solution is carried out. In Section 5.6, we discuss about open issues related to the proposed architecture and IoT scenarios. Finally, in Section 5.7 we draw our conclusions.

5.2 Related Work

In the rapidly evolving IoT scenario, security is an extremely timely issue. The heterogeneous and dynamic nature of the IoT brings up several questions related to security and privacy, which must be addressed properly by taking into account the specific characteristics of Smart Objects and the environments they operate in. Classical security algorithms and protocols, used by traditional Internet hosts, cannot simply be adopted by Smart Objects, due to their processing and communication constraints. An extensive overview of state-of-the-art security mechanisms in the IoT (including

symmetric/asymmetric cryptographic algorithms, hashing functions, security protocols at network/transport/application layers), aiming at providing features such as confidentiality, integrity, and authentication, is provided in [60]. An architecture for solving the problem of securing IoT cyberentities (which include Smart Objects, traditional hosts, and mobile devices), denoted as "U2IoT," has been proposed in [61], with the goal of addressing the issues of expanding domains, dynamic activity cycles, and heterogeneous interactions. U2IoT takes into account security in interactions that occur in three different phases: preactive, active, and postactive. In particular, the active phase provides authentication and access control functionalities. Authorization is therefore being considered a major issue, since it is becoming increasingly evident that access to resources in a global-scale network, such as the IoT, must be controlled and restricted in order to avoid severe security breaches in deployed applications.

Several works have also addressed very specific issues in IoT. A lightweight multicast authentication scheme for small-scale IoT applications is proposed in [62]. In [63], the authors take into account user mobility (i.e., roaming) and propose CPAL, an authentication mechanism designed to provide a "linking function" that can be used to enable authorized parties to link user access information, while preserving user anonymity and privacy. The secure integration of Wireless Sensor Networks (WSNs) into IoT is discusses in [64]. The authors propose a security scheme, which allows secure communication with Internet hosts by providing end-to-end confidentiality, integrity, and authentication, based on a Public-Key Infrastructure (PKI). The proposed scheme also introduces a two-step (offline/online) signcryption mechanism, in order to minimize processing time.

Several authentication mechanisms have also been defined for other issues, such as network access, which are also relevant for IoT scenarios. The Protocol for Carrying Authentication for Network Access (PANA) [65] is an IETF standard defining a network-layer transport for network access authentication methods, which are typically provided by the Extensible Authentication Protocol (EAP) [66]. In particular, PANA carries EAP, which can carry various authentication methods. OpenPANA [67] is an open-source implementation of PANA.

The problem of service authorization has been extensively treated in literature.

Several works have focused on how to implement different access control strategies. *Discretionary Access Control* (DAC) restricts the access to objects based on the identity of subjects and/or groups to which they belong. The controls are discretionary in the sense that a subject with certain access permissions can transfer that permission on to any other subject [68]. *Role-Based Access Control* (RBAC) relies on a policy that restricts access to resources to those entities which have been assigned a specific role [69, 70, 71]: RBAC requires that the roles are defined and assigned to users, and access permissions are set for resources. *Attribute-Based Access Control* (ABAC) restricts resource access to those entities which feature one or more specific attributes (e.g., age, geographic location, etc.) [72].

RBAC and ABAC are the most widespread approaches to restricting system access to authorized users. RBAC maps permissions to *roles* that a user has been assigned. On the other hand, ABAC maps permissions to *attributes* of the user. Typically, authorization mechanisms strongly depend on an authentication step that must have been previously taken, in order to identify users so that either their roles or their attributes can be verified and matched against the policies set for resource access.

In [73], the authors present a mechanism for fine-grained sub-delegation of access permissions for consumers of web applications, denoted as "DAuth." Applying access control mechanisms in constrained scenarios, such as wireless sensor networks, is a challenging task. A complex, context-aware access control system designed for a medical sensor networks scenario, which presents critical privacy and confidentiality issues, is described in [74].

The IETF ACE WG has also proposed "Delegated CoAP Authentication and Authorization Framework" (DCAF) [75]. The DCAF architecture introduces authorization servers, which are used to perform authentication and authorization, in order to unburden Smart Objects from storing a large amount of information by delegating such task to an external entity. While this solution is very similar to the one presented in this work, it focuses mainly on constrained environments, while the proposed one is intended to be generic and transparently integrated into IoT and Internet scenarios.

Although much work has been done with the goal of defining and integrating authorization mechanisms in several scenarios, our work, unlike others, focuses on

the definition of a generic authorization service which can be integrated into both Internet and IoT scenarios. In particular, the proposed mechanism explicitly takes into account the hybrid nature of the extended Internet that will be deployed in the next years. Moreover, the proposed architecture aims at minimizing the effort required by service developers to secure their services by providing a standard, configurable, and highly interoperable authorization framework.

5.3 IoT-OAS Architecture

The OAuth-based Authorization Service Architecture (*IoT-OAS*) can be invoked by any subscribed host or Smart Object. It can be ideally thought of as a remotely triggered switch that filters incoming requests and decides whether to serve them or not. The design goal of the *IoT-OAS* architecture is to relieve Smart Objects from the burden of handling a large amount of authorization-related information and processing all incoming requests, even if unauthorized. By outsourcing these functionalities, Smart Objects can keep their application logic as simple as possible, thus meeting the requirements for keeping the memory footprint as low as possible, which is extremely important for constrained devices. From a broader perspective, entire IoT large-scale deployments can greatly benefit from the presence of *IoT-OAS* in terms of configurability: a single constrained node (or a group of constrained nodes as a whole) can have their access policies updated remotely and dynamically, without requiring any direct intervention, which is especially convenient for Smart Objects placed in hardly-reachable and/or unattended locations. OAuth allows third-party applications to get access to user-related information hosted on an online service. All issued requests must certify that the *SC* application has been granted permission by the user to access its personal information on its behalf, namely by adding an "access token," which relates the user's identity and the client application. For ease of presentation, the used acronyms are summarized in Table 5.1.

Besides the three roles introduced in Section 5.1 (*U*, *SP*, and *SC*), OAuth adds an additional role: the Authentication Service (*AS*), which is invoked by the *SP* to verify the identity of a user in order to grant access tokens. The standard OAuth operation

U	*User* or Resource Owner
SP	*Service Provider*, which hosts users' resources
SC	*Service Consumer*, which accesses users' data stored by the *SP*
AS	*Authentication Service*, which is used by the *SP* to verify the identity of a user
RT	*Request Token*, a temporary ticket used by the *SC* to ask *U* to authorize access to its resources
AT	*Access Token*, used by the *SC* to perform authenticated requests
IoT-OAS	Delegated external authorization service, which can be invoked by Smart Objects to perform authorization checks to access protected resources

Table 5.1: Used main acronyms.

flow is shown in Fig.5.1. The procedure through which a *SC* can get a valid access token is the following:

1. *U* is willing to use the *SC*, either from a webpage, a mobile app, or a desktop application;

2. *SC* needs to access *U*'s personal information hosted on *SP*; *SC* asks the *SP* a *RT* carrying *SC*'s identity, which will be later exchanged for an *AT*;

3. *SP* verifies *SC*'s identity and returns a *RT*;

4. *SC* redirects *U* to the *SP*'s authentication service with the *RT*;

5. *U* contacts the *SP*'s *AS* presenting the *RT* and is asked to authenticate in order to prove its consent to grant access permissions to the *SC*;

6. the *RT* is exchanged for an *AT*, which relates *U* and *SC*;

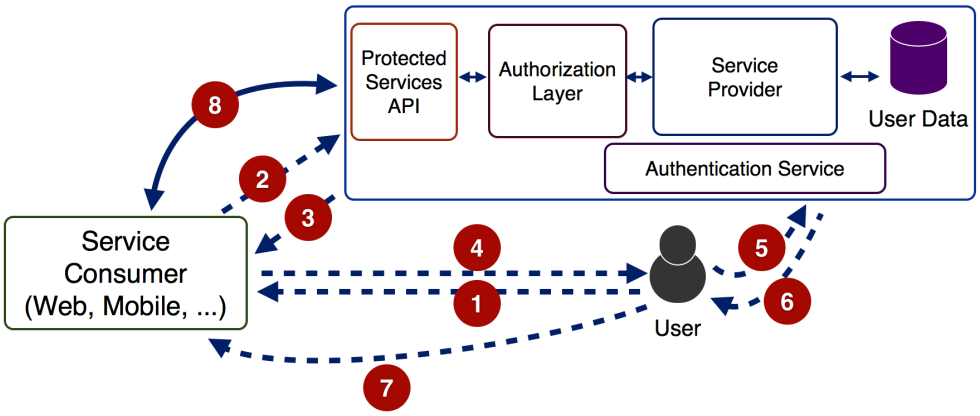


Figure 5.1: Standard OAuth roles and operation flow.

7. the *SC* receives the *AT* through a redirection to a callback URL (i.e., authentication callback);

8. the *SC* can issue requests to *SP* including the *AT*, for services that require *U*'s permission (protected APIs).

The design goal of the *IoT-OAS* architecture proposed in this work is to enable *SP*s, either based on HTTP or CoAP, to easily integrate an authorization layer without requiring any implementation overhead, other than invoking an external service. Delegating the authorization logic to an external service requires a strong trust relationship between the *SP* and the *IoT-OAS*. Fig. 5.2 shows the operation flows for a) *AT* grant procedure and b) *SC*-to-*SP* interaction in the *IoT-OAS* architecture. A detailed description of these operation flows in the proposed *IoT-OAS* architecture is presented in the remainder of this section.

5.3.1 Granting Access Tokens

The operation flow to grant an *AT* to a *SC* is shown in Fig. 5.2(a). The procedure resulting in the grant of an *AT* to a *SC* is similar to that of the standard OAuth operation flow, yet it has the following relevant differences:

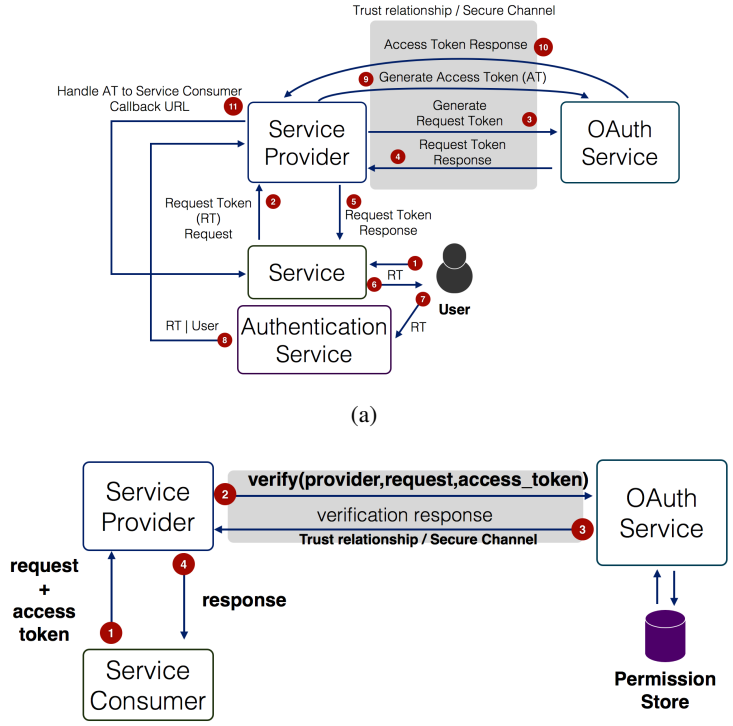


Figure 5.2: IoT-OAS main procedures: (a) AT grant procedure; (b) SP integration with IoT-OAS for request authorization.

1. as in the standard operation flow, the procedure is initiated by *U*;
2. the *SC* regularly contacts a *SP* to receive a *RT*;
3. the *SP*, which does not implement any OAuth logic, contacts the *IoT-OAS* asking to issue a *RT* for the *SC* by performing a *generate_request_token* RPC request;
4. the *IoT-OAS* verifies the identity of the *SC* and issues a *RT*, which is returned to the *SP*;
5. the *SP* handles the *RT* back to the *SC*;

6. the *SC* redirects *U* to the *AS* with the received *RT*;
7. *U* contacts the *SP*'s *AS* presenting the *RT* and authenticates in order to prove its consent to grant access permissions to the *SC*;
8. the *AS* notifies the *SP* that the authentication is successful and presents the *RT* with *U*'s identity;
9. the *SP* asks the *IoT-OAS* to exchange the *RT* with an *AT* for *U* by issuing a *generate_access_token* RPC request;
10. the *IoT-OAS* generates the *AT* and returns it to the *SP*;
11. the *SP* handles the *AT* to the *SC* with an authentication callback.

The use of an external *IoT-OAS* is totally transparent to the *SC*, which has no knowledge of how the *SP* is implementing the OAuth protocol. This leads to full backward compatibility with standard OAuth client applications. On the *SP*'s side, all the OAuth logic is delegated to the *IoT-OAS*, with the only exception of the *AS*. However, it is not mandatory that the *AS* resides within the *SP*'s realm, as it might interface with third-party authentication services, such as OpenID [76]. The only information the *SP* must hold is the reference to users' identities in order to make it possible to setup access permission policies on a per-user basis.

5.3.2 Authorizing Requests

The interaction between *SP* and *IoT-OAS* when serving incoming requests is shown in Fig. 5.2(b). Since the presence of the *IoT-OAS* is totally transparent to the *SC*, the communication between the *SC* and the *SP* is a regular OAuth communication. The difference is, again, on *SP*'s side, which needs to contact the *IoT-OAS* to verify that the incoming requests received from the *SC* are authorized in order to decide whether to serve them or not.

The operation flow is the following:

1. the *SC* requests *U*'s information to the *SP* using the *AT* received after *U*'s authentication (as in standard OAuth consumer-to-provider communication);

2. the *SP*, which does not implement any OAuth logic, refers to the *IoT-OAS* to verify if the incoming request is authorized (in order to do so, the *SP* issues a *verify* RPC request);
3. the *IoT-OAS* verifies the *SC*'s request and informs the *SP* about *SC*'s authorization for the request by performing a lookup in the permission store;
4. the *SP* serves the *SC*'s request according to the *IoT-OAS*'s response.

5.3.3 SP-to-IoT-OAS Communication: Protocol Details

The *SP* interacts with the *IoT-OAS* with a simple communication protocol. The protocol comprises three Remote Procedure Calls (RPCs), which are detailed below. It is important to remark that delegating the authorization decision to an external service requires an extremely high trust level between the *SP* and the *IoT-OAS*. Moreover, all communications between them must be secured and mutually authenticated, so that the *SP* security level is at least as high as if the authorization service were implemented internally. To ensure that an appropriate security level is met, communications between the *SP* and the *IoT-OAS* must occur with a secure transport such as HTTP-over-TLS (HTTPS), CoAP-over-DTLS (CoAPs) [77, 78], or HTTP/CoAP over a secure host-to-host channel setup with IPSec [78]. Mutual authentication ensures that i) the *IoT-OAS* is verified and ii) the requests come from a verified *SP*, whose identity is, therefore, implicit. The three RPCs of the *SP*-to-*IoT-OAS* communication protocol are the following:

1. *generate_request_token()*: this RPC is called by the *SP* to request the *IoT-OAS* to generate a request token for the given *SC*, to be later exchanged for an *AT*;
2. *generate_access_token(request_token,user_id)*: this RPC is called by the RPC to request the *IoT-OAS* to exchange the given *RT* for a new *AT* related to the given user;
3. *verify(request,access_token)*: this RPC is called by the *SP* to request the *IoT-OAS* to verify if the given *SC* request is authorized with the provided *AT* by performing a lookup into the permission store.

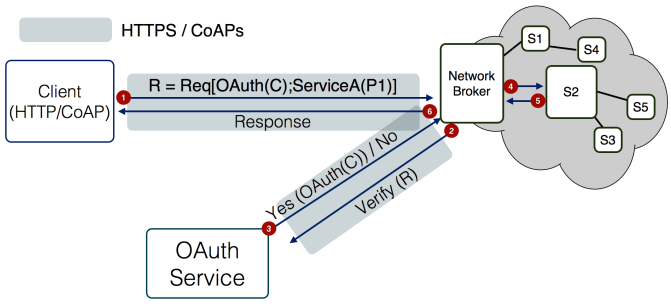
5.3.4 IoT-OAS Configuration

The *IoT-OAS* provides high customization to *SP*s by offering per-user and per-service access control policies. The *SP*s can remotely configure and manage the permissions that *SC*s are granted, which can be created, updated, revoked, and/or duplicated dynamically at any time. The *IoT-OAS* thus offers a dedicated *SP* access control configuration, denoted as "permission store." The permission store is a collection of the relations between *SC*s, users, and *SP* services and can ideally be seen as a lookup table.

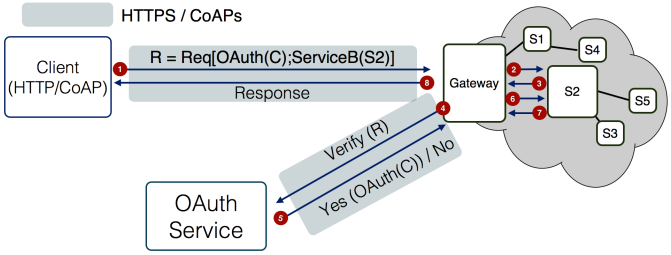
The configuration of the permission store can occur through web interfaces or API calls provided by the *IoT-OAS*. The possibility to dynamically manage the permissions, rather than having them co-located with the *SP*, is an extremely valuable feature, especially if *SP*s are Smart Objects with the need to be deployed in hardly accessible locations and may be automatically and/or remotely reconfigured.

5.4 Application Scenarios

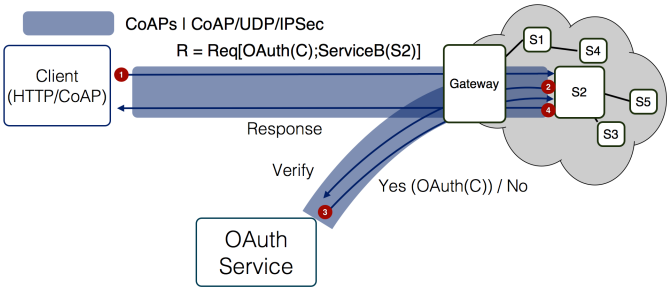
In this section, we present four significant IoT application scenarios to properly illustrate the functionalities of the proposed *IoT-OAS* service architecture. In the following scenarios, we consider an external client (based on HTTP or CoAP according to the context) that is interested to access a remote service provided by a Network Broker (*NB*), which is a border network element that exposes services on behalf of constrained nodes residing in the internal network, or directly by a generic Smart Object *S* directly available in the network behind a network Gateway. To clarify the following description we assume that the OAuth credentials owned by the involved external client have been obtained through a prior configuration phase based on the *IoT-OAS* service described in Section 5.3 and that service discovery is performed through an application-specific procedure, which is not directly related with the approach presented in this work.



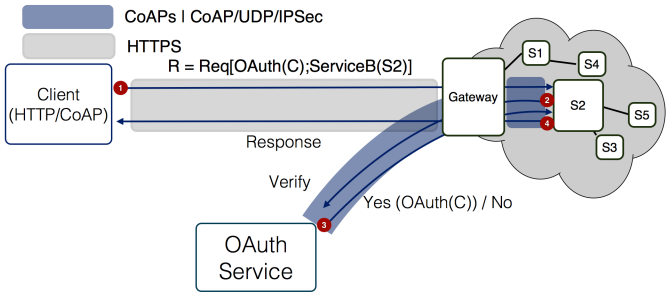
(a)



(b)



(c)



(d)

Figure 5.3: Application scenarios: (a) Client-to-Network broker communication; (b) Gateway-enabled communication; (c) End-to-End CoAP communication between the external client and the Smart Object; (d) Hybrid Gateway-based communication.

5.4.1 Network Broker Communication

In the first scenario, illustrated in Fig. 5.3(a), the client *C* (acting as a *SC*) discovers a $Service_A$ provided by the network broker *NB*. In order to serve external requests, *NB* can retrieve information from different Smart Objects in its network. In this case, in order to simplify the context, we assume that *NB* needs information only from the Smart Object S_2. The client, through a secure channel based on HTTPS or CoAPs, sends a request *R* for $Service_A$ including its OAuth credentials, denoted as *OAuth(C)*. The client's OAuth information is used by the service provider *NB* to properly validate the request and to verify the identity of *C* and that *C* has the right privileges to access the requested service. Since *NB* could be implemented using an embedded device or, generally, using a device with limited computational and storage capabilities (which, with high probability, does/should not implement a complex logic such as OAuth) delegates the verification of the incoming request to the *IoT-OAS*. Once *NB* receives *R* from *C*, it sends a verification request to *IoT-OAS* (always through a secure channel based on HTTPS or CoAPs, according to its implementation or capabilities) with the original incoming request and its credentials. The *IoT-OAS*, after verifying the validity of the submitted request (according to *NB*'s configuration) and *C*'s identity, replies communicating if *R* is authorized. If the feedback is positive and *C* is allowed to access requested service, *NB* internally contacts the Smart Object S_2 to retrieve the required information and sends back the response to *C*. If the response received from *IoT-OAS* is negative, *C* is not granted access and the *NB* responds immediately to *C* without any kind of interaction with the Smart Object.

5.4.2 Gateway-based Communication

In the second scenario, illustrated in Fig. 5.3(b), an external client *C*, based on HTTPS or CoAPs communications, is interested in accessing a service directly provided by the Smart Object S_2 which does not manage HTTP or CoAP (due to computational or implementation constraints) and is behind a Gateway *G*. In particular, *G* has the role to translate the incoming requests from the external networks to available Smart Objects inside its own network. In this scenario, *C* sends a request *R* (including the

client's OAuth credentials) to G for $Service_B$ provided by S_2. R is translated by G and forwarded to S_2 that, in order to validate the new request and the requestor's identity, sends through G a verification request to the *IoT-OAS* using a secure communication protocol such as HTTPS or CoAPs. The verification message and the response (positive or negative) generated by S_2 are properly managed and translated by G to allow the communication among *IoT-OAS*, the Smart Object, and the client.

5.4.3 End-to-End CoAP Communication

Fig. 5.3(c) shows a different scenario where a Smart Object S_2 (i.e., reachable at an IPv6 address) in a sensor network provides directly a remote CoAP service $Service_B$. Since all involved entities can use the same protocol, in this case the network gateway acts only as a router without the need to translate incoming and outgoing messages between the external world and the sensor network. The CoAP client *CC* sends securely and directly to the Smart Object a request R containing its OAuth credentials and the reference for $Service_B$ provided by S_2. Since the Smart Object is usually a sensor or an embedded device with limited computational and storage capabilities (which, as previously described, does not implement a complex logic like OAuth), it delegates the verification of the incoming request to the OAuth Service. S_2 sends a verification request to *IoT-OAS* over CoAPs to check R. The *IoT-OAS* validates the request based on *CC*'s credentials and the type of requested service; it then informs the Smart Object S_2 about whether R can be served or not. S_2, according to the response of *IoT-OAS*, replies to the requesting client with the service outcome or, if *CC* is not allowed to access $Service_B$, with an error message.

5.4.4 Hybrid Gateway-based Communication

The last scenario, shown in Fig. 5.3(d), is characterized by a hybrid approach where the external client uses an application protocol (such as HTTP) different from that used by Smart Objects (CoAP). Similarly to the case in Subsection 5.4.2, the gateway manages the communication between the external world and its network: in this case, it just translates incoming requests from HTTP to CoAP for S_2. Once a new request

(with OAuth credentials and service reference) arrives to the Smart Object, it securely uses *IoT-OAS* to verify the validity of *R*, through CoAPs. The response (positive or negative, according to the *IoT-OAS* feedback) is translated by the gateway from CoAP to HTTP and forwarded to the client through a secure channel.

5.5 Experimental results

In order to demonstrate the feasibility and performance of the proposed IoT-oriented authorization mechanism, we have conducted experimental tests to evaluate the energy consumption on Smart Objects. In fact, the security of the authorization process is guaranteed by the use of OAuth. We remark that the architecture scenarios in Fig. 5.3(a) and Fig. 5.3(b) are not critical from an energy consumption viewpoint, as they rely on communications between a gateway, which is a non-constrained node, and the external authorization service. For this reason, we have limited our experimental investigation to the scenarios in Fig. 5.3(c) and Fig. 5.3(d), which require that the Smart Object communicates with the authorization service: in this case, energy consumption is of concern.

5.5.1 Experimental Setup

Within the EU project CALIPSO [79], the *IoT-OAS* service has been implemented on a regular web server, based on open source technologies, equipped with HTTP/CoAP proxy capabilities in order to be easily integrated in all the application scenarios shown in Section 5.4. The *SP*-to-*IoT-OAS* communication has been implemented on a variety of devices, either regular hosts or Contiki OS-based constrained devices [80]. The choice of the Contiki OS has allowed to investigate the feasibility of the delegation approach to authorization in both simulated environments (using the Cooja simulator) and real testbeds, taking also into account the presence of duty-cycle. Moreover, the Contiki OS provides IPSec and DTLS implementations over 6LowPAN [81, 82] and CoAP [83].

The conducted tests aim at evaluating the energy consumption of Contiki-based Smart Objects. The simulations have been performed using the Cooja simulator, in

order to gather data for the activity of the CPU and radio interface. The used experimental platform is based on Zolertia Z1 nodes, with nominal 92 kB ROM (when compiling with 20-bit architecture support) and an 8 kB RAM. In practice, the compilation with the Z1 nodes has been performed with a 16-bit target architecture, which lowers the amount of available ROM to roughly 52 kB.

The experimental setup consists of a CoAP client node sending a request to a CoAP server. The CoAP server must then authorize the request and decide whether to serve it or not. This configuration is compatible with the application scenarios shown in Fig. 5.3(c) (denoted as *End-to-End CoAP Communication*) and Fig. 5.3(d) (denoted as *Hybrid Gateway-based Communication*). The tests involved four different scenarios, shown in Table 5.2, classified depending on the type of authorization strategy adopted by the CoAP server (columns) and security level adopted at the network-layer (rows). As for security at the network layer, IPSec has been configured to work with Encapsulated Security Payload (ESP) only. ESP has been selected since it is necessary to encrypt the payload of the packets, in order to protect the access token, rather than authenticating the endpoints of communication through Authentication Header (AH). Regarding the implementation of the OAuth protocol, the verification procedure has been performed using the HMAC-SHA1 signature scheme [84]. Although OAuth provides also other signature schemes (PLAINTEXT and RSA-SHA1), HMAC-SHA1 has been selected as it is does not require a secure transport and a PKI for the management of public keys. As for the use of SHA-1, we point out that, in principle, different hash functions, such as SHA-2 and SHA-3 (also known as KECCAK [85]), might be used instead. However, this would be a violation of the OAuth specification, which would require SPs and SCs to be aware of these different signature schemes.

The message flow, which depends solely on the delegation to *IoT-OAS*, is shown in Fig. 5.4 considering two cases: OAuth is implemented in the CoAP server (top); the CoAP server relies on the *IoT-OAS* server (bottom).

		Authorization strategy	
		Smart Object implements OAuth	**Smart Object delegates to *IoT-OAS***
Security	**IP** (none)	*oauth-ip*	*oas-ip*
	IPSec (ESP)	*oauth-ipsec*	*oas-ipsec*

Table 5.2: Experimental scenarios.

5.5.2 Energy Consumption Evaluation

The energy consumption of the CoAP server has been evaluated using Powertrace [21], a tool for network-level power profiling for low-power wireless networks. Powertrace estimates the energy consumption of each device component, such as the radio chip and the microcontroller. It computes the amount of time a component is turned on (active mode) and off (sleep mode). In order to determine the energy consumption, we refer to the current consumption of each component indicated in the Z1 datasheet. In particular, the MSP430f2617 microcontroller consumes 0.515 mA in active mode and 0.5 μA in low-power mode (lpm), respectively. Similarly, the CC2420 radio transceiver consumes 17.4 mA in TX mode and 18.8 mA in RX mode. In order to obtain the total consumed energy for the Smart Object, the following conversion formula has been used:

$$E = \sum_{j \in \mathscr{M}} i_j \cdot v \cdot \Delta t_j \tag{5.1}$$

where $\mathscr{M}$ is the set of all operation modes of the Smart Object (active, lpm, TX, and RX); v is the nominal voltage of the Smart Object; and Δt_j is the time the Smart Object was in the j-th operation mode j.

In Fig. 5.5, the aggregate energy consumption (dimension [mJ]) for different hardware components in the four scenarios in Table 5.2 is shown. The energy con-

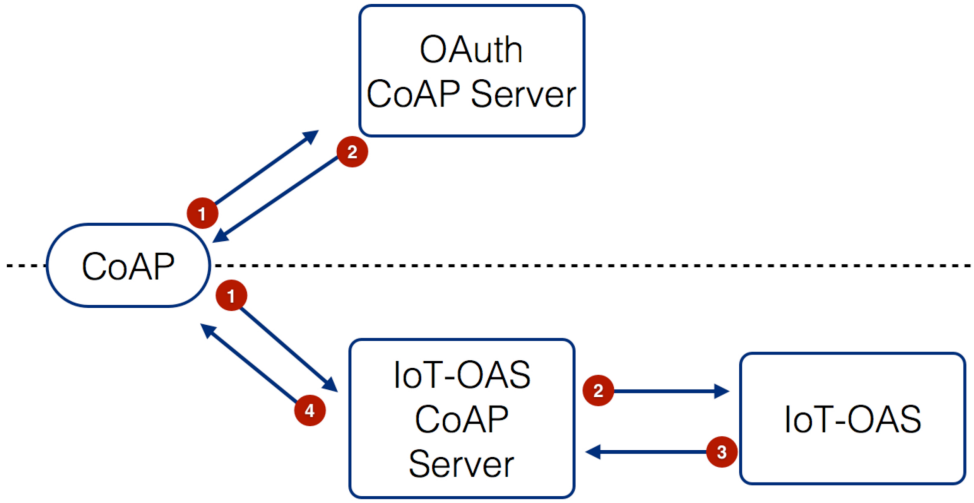


Figure 5.4: Message flow in the case OAuth is implemented in the CoAP server (top) and in case the CoAP server relies to the IoT-OAS server (bottom).

sumption is broken down into figures related to CPU, radio transmission, and lpm activity. The obtained results provide interesting insights on the performance of the proposed delegation approach. In particular, it can be observed that if *IoT-OAS* is being used on top of IP, the amount of energy consumption related to processing is lower than if the OAuth logic is implemented directly on the Smart Object. However, the overall consumption is higher when relying on *IoT-OAS*, due to the contribution of radio transmission. The reason for this behavior is that the large size of application-level packets requires fragmentation and, thus, multiplies the number of transmitted packets over IEEE 802.15.4 networks. If *IoT-OAS* is used, the number of transmitted packets doubles since the original packet must be relayed to the delegated service in order to perform the authorization checking procedure. When using IPSec, there is no gain, in terms of processing load, with respect to the local-OAuth approach. This happens because the number of decryption and encryption procedures also doubles and, since they rely on heavyweight asymmetric cryptographic primitives, also the processing load increases as well. However, the OAuth protocol specification states

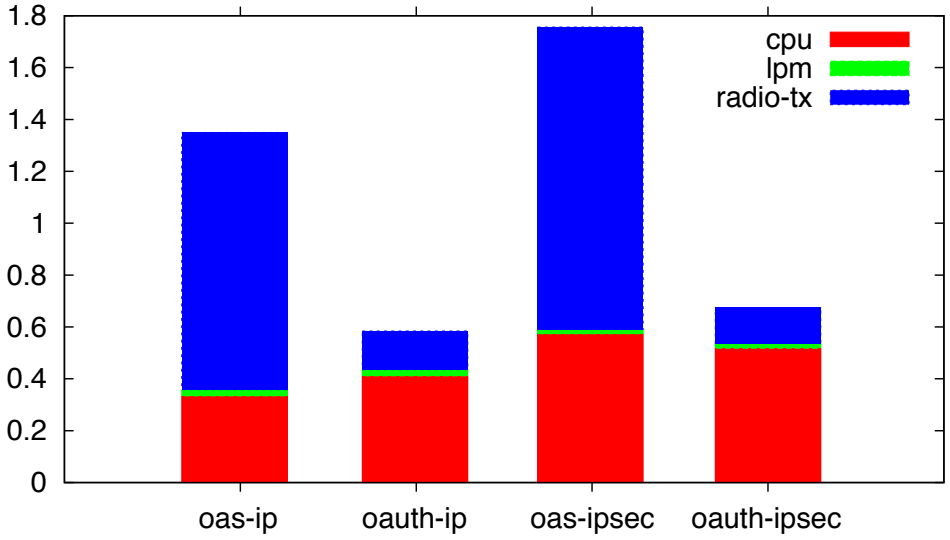


Figure 5.5: Aggregate energy consumption (dimension: [mJ]) for the four experimental scenarios.

that, if using the HMAC-SHA1 signature scheme, a secure underlying transport is not mandatory.

5.5.3 Practical Considerations

Even though the energy consumption results presented in Subsection 5.5.2 may seem to discourage the adoption of a delegation approach for authorization grant, there are other considerations that strongly motivate the use of *IoT-OAS*.

First of all, some considerations on memory footprint should be made. In order for the OAuth software module to fit inside the ROM of a Smart Object, we had to turn off many features of the Contiki OS: i) RPL could not be used (all communication acts were single-hop); ii) Contiki MAC was replaced by NULL MAC; and iii) no radio duty cycling was active. In Fig. 5.6, the contributions of all the software modules to the available ROM are shown. The shown numbers are best-case figures (e.g.,

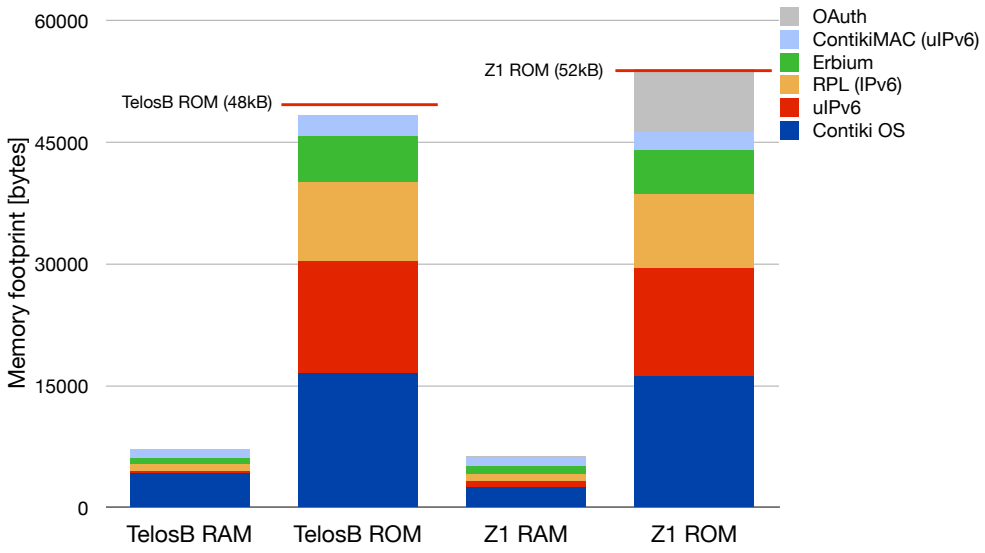


Figure 5.6: Memory footprint (dimension: [bytes]), with compiler optimization enabled, for different software modules in Contiki for TelosB (48 kB available) and Zolertia Z1 nodes (∼52 kB available with 16-bit target compilation).

no resources have been allocated for CoAP), but the Smart Objects are not operative in this case (any incoming request could not be served since no CoAP resource is hosted). Even in this best case, from a memory occupation viewpoint, from the results in Fig. 5.6 one can see that the amount of available ROM is not sufficient to host also the OAuth logic: therefore, some features must be excluded from the Smart Object to implement OAuth locally.

Moreover, in order to provide authorization, the Smart Object should also include a database of those clients that are authorized to perform requests for the resources managed by the constrained CoAP server, together with their access tokens. Of course, this is infeasible because of the memory shortage. On the other hand, an external authorization service might have a database large enough to fit all the information needed to manage any number of third-party clients. This is shown in Fig. 5.7, where one can see that the most lightweight configuration of software on a Zolertia

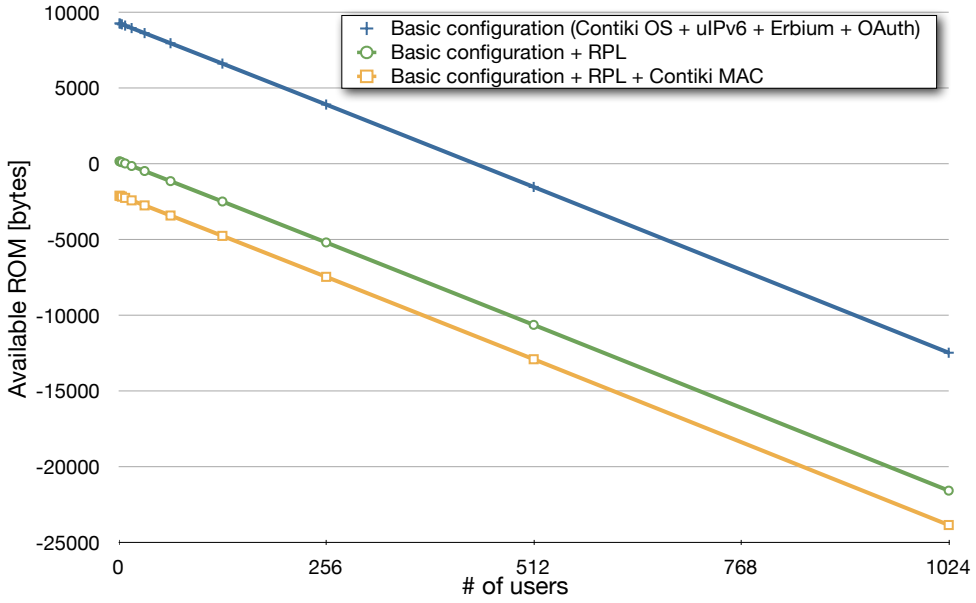


Figure 5.7: Available ROM memory (dimension: [bytes]) on a Zolertia Z1 (16-bit target compilation) when hosting a user database under different configurations of Contiki OS software modules.

Z1 node can at most fit a database of less than 500 users (assuming an average of roughly 20 bytes per user). From Fig. 5.7, it can also be observed that just integrating RPL decreases the maximum number of users stored in the database to 8.

Another important point that should be highlighted is that the delegation of the authorization procedures allows the owners of the Smart Objects to reconfigure, even with very fine-grained access policies, the authorization grants to external consumers without the need to re-program the Smart Objects, which is crucial when the Smart Objects are deployed in hardly reachable locations and their reconfiguration can be very complicated. As an external service can perform highly dynamic configuration, it could be very easy to grant or remove access permissions at any time with little, if any, need of human intervention, which can be a difficult and time-consuming operation. A mechanism that allows operators to change access policies remotely

and possibly acting on multiple nodes at the same time can, therefore, be far more convenient by cutting down management costs and minimizing the time required for the re-configuration of Smart Objects.

One final remark, related to energy consumption, is the fact that the analysis in Subsection 5.5.2 is performed using the HMAC-SHA1 signature scheme. The OAuth protocol specification considers also the use of the RSA-SHA1 signature scheme, which requires much higher processing load as it relies on asymmetric cryptographic primitives and introduces also the problem of public key management. The amount of processing load (and, thus energy consumption) considered in Fig. 5.5 is then underestimated.

In conclusion, the delegation approach can significantly improve and simplify the design and deployment of Smart Objects, allowing to focus on the fundamental functionalities that a constrained node should implement, rather than dealing with other aspects that could be easily outsourced with minimum impact on the development of the application. The use of *IoT-OAS* does not necessarily forbid or discourage an in-node implementation of authorization functionalities, which might be suited for smart objects with sufficient resources and capabilities. *IoT-OAS* can be applied effectively in several scenarios, with different network configurations and Smart Object capabilities. For instance, being aware of energy consumption issues, *IoT-OAS* can be used by network elements that do not present the same constraints of smart objects (e.g., LoWPAN border routers, LBRs, which can be connected to a fixed energy supply), as shown in Fig. 5.3(a) and Fig. 5.3(b). Such elements can apply access policies and filter incoming requests, in order to "shield" the constrained network, while allowing the Smart Objects to keep their implementation.

5.6 Discussion: Advantages, Limitations, and Outlook

In this work, we have presented a novel general architecture for service authorization to be used in Internet, IoT, and hybrid scenarios. According to the architectural nature of this work and the experimental results presented in Section 5.5, the following aspects need to be further discussed.

5.6.1 Security Issues

The use of Internet protocols and the very nature of Smart Objects raise security issues in the proposed *IoT-OAS* architecture. The interested reader is referred to [86] for an overview of security threats in IoT.

First, delegation to a third party service requires a strong trust on the third party. The trust relationship between a Smart Object and the *IoT-OAS* is actually a trust relationship between the owner of the Smart Object and the *IoT-OAS*. The establishment of such trust relationship falls beyond the scope of this work, which describes the service architecture to which the Smart Objects subscribe. A strongly secure communication channel, such as a VPN tunnel, could be a suitable solution to provide secure and authenticated communications with the authorization service.

The use of standard Internet security protocols can have a negative impact on the performance of Smart Objects, as the communication overhead and computational cost might dramatically affect the energy consumption. In order to tackle these issues, the following actions may be taken:

- define constrained versions and implementations of the security protocols, in order to make the handshaking phase lighter and to minimize packet fragmentation;
- define a constrained version of the OAuth protocol, in a similar fashion to what is being carried out in the IETF CoRE Working Group with the definition of the CoAP protocol (with respect to HTTP);
- define new cryptographic suites for the security protocols by integrating lightweight cryptographic algorithms (such as TEA [87], SEA [88], and PRESENT [89]) and lightweight hash functions (such as DM-PRESENT [90] and SHA-3 [91]), which may be more suitable for constrained devices.

Other security aspects might be related to the following possible attacks that smart objects might undergo.

- Denial-of-Service (DoS) might occur if Smart Objects receive a large number of requests to serve. In this case, the use of the *IoT-OAS* would decrease the

ability of the Smart Object to resist against this type of attacks. A solution could be to use the gateway to protect the Smart Object, possibly by offering caching capabilities.

- Man-in-the-Middle (MITM) attacks are possible if using a HTTP/CoAP proxy — which, by its nature, is a man-in-the-middle — thus destroying any form of end-to-end security. If the proxy is not trusted, it could access all communications among the endpoints and could possibly get access to the authorization information, thus gaining permission to spoof the requestor's identity.

- Physical threats are related to the impossibility to supervision constantly Smart Objects once they have been deployed in public/remote areas.

The considerations above are not strictly related to the *IoT-OAS* architecture, but are typically related to all IoT scenarios and are currently being investigated. However, even though they fall outside the scope of this work, it is important to cite them as open issues. It is also important to remark that the *IoT-OAS* architecture is not a solution for security aspects in IoT, but, rather, is meant to provide an authorization layer which is easy to integrate and manage for Internet and IoT services.

5.6.2 Computational and Storage Overhead

The OAuth protocol has been designed to handle computation and storage overhead while providing a standard and simple authorization mechanism for resource access by third-party applications. The same considerations can be applied to our approach, as there is no computation and storage on the node, but all the information resides on the central authorization service. Typical examples of the scalability of OAuth-based services are online social networks, such as Twitter and Facebook, which deal with hundreds of millions of users, billions of requests, and several thousands of third-party applications which access the hosted resources. The above considerations are confirmed by the performance evaluation of the *IoT-OAS* architecture in Section 5.5: i) the memory footprint of all software modules that should be run on the Smart Object leaves no space for the storage of user identities and access policies (as shown

in Fig.5.7); ii) from a processing perspective, the Smart Object does benefit from a delegation approach, as there is no computation to be performed besides sending a request to the *IoT-OAS*.

5.7 Conclusion

In this chapter, we have proposed a novel architecture to provide HTTP and CoAP service providers with an authorization layer to be able to disseminate their services without the need of implementing the OAuth logic, but, rather, by invoking an external OAuth-based authorization service, denoted as "*IoT-OAS*." The designed approach has been applied to significant IoT scenarios with multiple Smart Objects (or, more generally, constrained devices) characterized by limited computational power, operating in lossy and low-power networks, and usually battery-powered thus requiring extreme attention on energy consumption.

A performance evaluation has been performed by conducting simulations with Cooja targeting Contiki-based Zolertia Z1 nodes. The experimentation has shown that, from a purely energy consumption perspective, the delegation approach can increase the amount of energy consumed, due to the fragmentation of application-layer messages performed in order to fit in IEEE 802.15.4 packets. However, other issues, such as memory footprint and dynamic configuration capabilities, show that implementing the OAuth logic locally is infeasible with currently available Smart Objects, making the delegation approach provided by *IoT-OAS* preferable. Moreover, delegating the authorization logic to an external service leads to several additional benefits, such as: i) reducing the time required by service developers to implement OAuth-protected online services; ii) supporting legacy OAuth client applications seamlessly; iii) limiting the device complexity only to service logic, while still providing access control policies for its services. An extremely appealing advantage of externalizing the authorization logic is the possibility to dynamically and remotely configure fine-grained access control policies on a per-service and per-client basis, without the need of direct intervention on the deployed devices.

Security considerations have been also discussed, taking into account well-known

IoT-related issues. As a future research direction, the proposed architecture will be implemented and tested thoroughly in both simulation environments and real testbeds, in order to evaluate advanced performance metrics in constrained environments and in the presence of duty-cycle.

Conclusions and Outlook

This thesis summarizes the main outputs of a three-year research activity on low-power WSNs connected to the Internet of Things. Several critical issues of WSNs have been investigated at all protocol layers of an IoT-based architecture, including: small memory footprint on the sensor nodes; lossy links; energy consumption; and delay performance. Such problems are likely intertwined and impact all layers of the architecture. Therefore, all the presented contributions are not confined to a single layer, but rely on cross-layer solutions addressing the aforementioned problems simultaneously.

First, the problems related to lack of memory space and lossy links have been studied at the application layer. It has been shown how such problems impact the distributed storage of sensed data in WSNs. Two redundant distributed data storage mechanisms have been proposed in order to increase the resilience and storage capacity against node failure and local memory shortage. The first mechanism is a greedy hop-by-hop replication scheme, which recursively distributes data to the direct neighbor with most available space. The second mechanism adds knowledge from the underlying routing topology (created by RPL) for efficient data distribution, by storing data at nodes closer to the root of the network DAG.

The second chapter has shown that an accurate knowledge of the network topology is required to take proper decisions at higher layers. Therefore, the third chapter has been devoted to a comprehensive analysis of the routing layer and, in particular, to the RPL protocol, since it is the reference routing protocol in IoT scenarios. The focus of this chapter has been mainly on the trade-off between energy consumption

and latency in WSNs. A novel routing metric for RPL, which minimizes the delay towards the root of the RPL DODAG, has been designed, assuming that nodes run with very low duty cycles (e.g., under 1%) at the MAC layer. It has been shown how the routing layer can leverage on information from the underlying MAC layer to optimally build routing paths.

The previous results have suggested that, specularly to the previous design approach, a deep understanding of the MAC layer in the design of routing protocols is required. Therefore, the fourth chapter has studied the trade-off between energy consumption and delay from the MAC layer point of view, with a key focus on radio duty cycling. A cross-layer MAC protocol, denoted as RAWMAC, has been proposed. RAWMAC uses routing as a management module for asynchronous MAC layers, which are more robust to clock drifts, in order to create a wave of propagation from the leaves to the root of a routing tree.

Finally, limitations brought by energy consumption and small memory footprint have been studied from a security point of view. Security does not fall into a specific layer of the architecture, but it can be vertically applied to all layers. In particular, the fifth chapter has dealt with authorization and privacy for WSNs. The research activity has investigated how complex authorization mechanisms may waste the resources of a smart object. A flexible service-oriented architecture for WSNs, which delegates the authorization to an external service, based on the standard OAuth protocol, has been proposed.

All the proposed solutions have been tested both through simulations as well as experimental testbeds. In particular, the CALIPSO project, within which the research activity of the PhD program that has lead to this thesis was carried out, allowed to evaluate our work in real IoT-based application scenarios, such as Smart Parking.

In the future, IoT will continue to increase its pivotal role in our economy and society. The research community has been initially focusing on how to connect WSNs with IP. This has spurred a significant research activity trying to determine the “best” IoT-based technologies and protocols to be applied to WSNs. However, the fragmentation and proliferation of standards and technologies, designed within bodies such as IETF and IEEE, has moved the attention of the community towards Big Data and

other concepts. As for WSNs, now is really the time to test and evaluate things in real applications on a large scale. This will help new IoT-based unimagined applications to emerge, as well as novel business models to come.

Bibliography

[1] Gaurav Mathur, P. Desnoyers, Deepak Ganesan, and Prashant Shenoy. Ultra-low power data storage for sensor networks. In *The Fifth International Conference on Information Processing in Sensor Networks* (IPSN'06), pages 374–381, Nashville, TN, USA, 2006.

[2] Fang Hongping and Fang Kangling. Overview of data dissemination strategy in wireless sensor networks. In *International Conference on E-Health Networking, Digital Ecosystems and Technologies* (EDT), pages 260–263, Shenzhen, China, April 2010.

[3] Sylvia Ratnasamy, Brad Karp, Scott Shenker, Deborah Estrin, and Li Yin. Data-centric storage in sensornets with GHT, a geographic hash table. *ACM Mobile Networks and Applications*, pages 427–442, August 2003.

[4] Samuel R. Madden, Michael J. Franklin, Joseph M. Hellerstein, and Wei Hong. Tinydb: an acquisitional query processing system for sensor networks. *ACM Trans. Database Syst.*, pages 122–173, March 2005.

[5] A. Awad, R. Germany, and F. Dressler. Data-centric cooperative storage in wireless sensor network. In *2nd International Symposium on Applied Sciences in Biomedical and Communication Technologies* (ISABEL), pages 1–6, Bratislava, Slovak Republic, November 2009.

[6] M. Albano, S. Chessa, F. Nidito, and S. Pelagatti. Dealing with nonuniformity in data centric storage for wireless sensor networks. *IEEE Transactions on*

Parallel and Distributed Systems, 22(8):1398 –1406, aug. 2011. doi:10.1109/TPDS.2011.18.

[7] Haiying Shen, Lianyu Zhao, and Ze Li. A distributed spatial-temporal similarity data storage scheme in wireless sensor networks. *IEEE Transactions on Mobile Computing*, 10(7):982 –996, july 2011. doi:10.1109/TMC.2010.214.

[8] Khandakar Ahmed and Mark A. Gregory. Techniques and challenges of data centric storage scheme in wireless sensor network. *Journal of Sensor and Actuator Networks*, 1(1):59–85, 2012. doi:10.3390/jsan1010059.

[9] A. Omotayo, M.A. Hammad, and K. Barker. A cost model for storing and retrieving data in wireless sensor networks. In *IEEE 23rd International Conference on Data Engineering Workshop* (ICDE), pages 29–38, Istanbul, Turkey, April 2007.

[10] M. Takahashi, Bin Tang, and N. Jaggi. Energy-efficient data preservation in intermittently connected sensor networks. In *Computer Communications Workshops (INFOCOM WKSHPS), 2011 IEEE Conference on*, pages 590 –595, april 2011.

[11] Liqian Luo, Chengdu Huang, T. Abdelzaher, and J. Stankovic. Envirostore: A cooperative storage system for disconnected operation in sensor networks. In *26th IEEE International Conference on Computer Communications* (INFOCOM'07), pages 1802–1810, Anchorage, Alaska, USA, May 2007.

[12] Yu-Chee Tseng, Fang-Jing Wu, and Wan-Ting Lai. Opportunistic data collection for disconnected wireless sensor networks by mobile mules. *Ad Hoc Networks (2013)*, 2013. . To appear.

[13] Guilherme Maia, Daniel L. Guidoni, Aline C. Viana, Andre L.L. Aquino, Raquel A.F. Mini, and Antonio A.F. Loureiro. A distributed data storage protocol for heterogeneous wireless sensor networks with mobile sinks. *Ad Hoc Networks (2013)*, 2013. . To appear.

[14] J. Neumann, N. Hoeller, C. Reinke, and V. Linnemann. Redundancy infrastructure for service-oriented wireless sensor networks. In *NCA'10, 9th IEEE International Symposium on Network Computing and Applications*, Cambridge, MA, USA, July 2010.

[15] K. Piotrowski, P. Langendoerfer, and S. Peter. tinyDSM: A highly reliable cooperative data storage for wireless sensor networks. In *International Symposium on Collaborative Technologies and Systems* (CTS'09), pages 225–232, Baltimore, Maryland, USA, May 2009.

[16] P. Gonizzi, G. Ferrari, V. Gay, and J. Leguay. Redundant distributed data storage: experimentation with the SensLab testbed. In *Proc. Int. Conf. Sensor Networks* (SENSORNETS), Rome, Italy, February 2012.

[17] F. Osterlind, A. Dunkels, J. Eriksson, N. Finne, and T. Voigt. Cross-Level Sensor Network Simulation with COOJA. In *Proc. of IEEE LCN*, 2006.

[18] SensLAB. A very large scale open wireless sensor network testbed, 2008. Website: `http://www.senslab.info/`.

[19] T. Ducrocq, J. Vandaele, N. Mitton, and D. Simplot-Ryl. Large scale geolocalization and routing experimentation with the senslab testbed. In *IEEE 7th International Conference on Mobile Adhoc and Sensor Systems* (MASS'10), pages 751–753, San Francisco, CA, USA, Nov. 2010.

[20] Adam Dunkels. The ContikiMAC Radio Duty Cycling Protocol. Technical Report T2011:13, Swedish Institute of Computer Science, December 2011. URL: `http://www.sics.se/~adam/dunkels11contikimac.pdf`.

[21] A. Dunkels, J. Eriksson, N. Finne, and N. Tsiftes. Powertrace: Network-Level Power Profiling for Low-power Wireless Networks. Technical report, Swedish Institute of Computer Science, March 2011. URL: `http://soda.swedish-ict.se/4112/1/T2011_05.pdf`.

[22] T. Winter, P. Thubert, A. Brandt, J. Hui, R. Kelsey, P. Levis, K. Pister, R. Struik, J. Vasseur, and R. Alexander. RPL: IPv6 Routing Protocol for Low-Power and Lossy Networks. RFC 6550, March 2012. URL: http://www.ietf.org/rfc/rfc6550.txt.

[23] A. Tsertou and D.I. Laurenson. Revisiting the hidden terminal problem in a csma/ca wireless network. *Mobile Computing, IEEE Transactions on*, 7(7):817 –831, july 2008. doi:10.1109/TMC.2007.70757.

[24] Feng Wang and Jiangchuan Liu. Networked wireless sensor data collection: Issues, challenges, and approaches. *Communications Surveys Tutorials, IEEE*, 13(4):673 –687, quarter 2011. doi:10.1109/SURV.2011.060710.00066.

[25] Shuai Gao and Hongke Zhang. Energy efficient path-constrained sink navigation in delay-guaranteed wireless sensor networks. *JNW*.

[26] Sushant Jain, Rahul C. Shah, Waylon Brunette, Gaetano Borriello, and Sumit Roy. Exploiting mobility for energy efficient data collection in wireless sensor networks. *Mob. Netw. Appl.*

[27] Giuseppe Anastasi, Marco Conti, Emmanuele Monaldi, and Andrea Passarella. An adaptive data-transfer protocol for sensor networks with data mules. In *World of Wireless, Mobile and Multimedia Networks, 2007. WoWMoM 2007. IEEE International Symposium on a*, June 2007.

[28] A.A. Somasundara, A. Ramamoorthy, and M.B. Srivastava. Mobile element scheduling for efficient data collection in wireless sensor networks with dynamic deadlines. In *Real-Time Systems Symposium, 2004. Proceedings. 25th IEEE International*, December 2004.

[29] Mario Di Francesco, Sajal K. Das, and Giuseppe Anastasi. Data collection in wireless sensor networks with mobile elements: A survey. *ACM Trans. Sen. Netw.*, 8(1):7:1–7:31, August 2011. URL: http://doi.acm.org/10.1145/1993042.1993049, doi:10.1145/1993042.1993049.

[30] Yu-Chee Tseng, Wan-Ting Lai, Chi-Fu Huang, and Fang-Jing Wu. Using mobile mules for collecting data from an isolated wireless sensor network. In *ICPP'10, 39th International Conference on Parallel Processing*, September 2010.

[31] Omprakash Gnawali, Rodrigo Fonseca, Kyle Jamieson, David Moss, and Philip Levis. Collection tree protocol. In *Proceedings of the 7th ACM Conference on Embedded Networked Sensor Systems*, SenSys '09, pages 1–14, New York, NY, USA, 2009. ACM. URL: `http://doi.acm.org/10.1145/1644038.1644040`, `doi:10.1145/1644038.1644040`.

[32] N. Hassanzadeh, O. Landsiedel, F. Hermans, O. Rensfelt, and T. Voigt. Efficient mobile data collection with mobile collect. In *Distributed Computing in Sensor Systems (DCOSS), 2012 IEEE 8th International Conference on*, pages 25 –32, may 2012. `doi:10.1109/DCOSS.2012.12`.

[33] N. Burri, P. von Rickenbach, and R. Wattenhofer. Dozer: Ultra-low power data gathering in sensor networks. In *Information Processing in Sensor Networks, 2007. IPSN 2007. 6th International Symposium on*, pages 450 –459, april 2007. `doi:10.1109/IPSN.2007.4379705`.

[34] Razvan Musaloiu-E., Chieh-Jan Mike Liang, and Andreas Terzis. Koala: Ultra-low power data retrieval in wireless sensor networks. In *Proceedings of the 7th international conference on Information processing in sensor networks*, IPSN '08, pages 421–432, Washington, DC, USA, 2008. IEEE Computer Society. URL: `http://dx.doi.org/10.1109/IPSN.2008.10`, `doi:10.1109/IPSN.2008.10`.

[35] M. Buettner, G.V. Yee, E. Anderson, and R. Han. X-mac: a short preamble mac protocol for duty-cycled wireless sensor networks. In *Proc. ACM SenSys*, 2006.

[36] A. El-Hoiydi and J.-D. Decotignie. Wisemac: an ultra low power mac protocol for the downlink of infrastructure wireless sensor networks. In *Computers and Communications, 2004. Proceedings. ISCC 2004. Ninth Interna-*

tional Symposium on, volume 1, pages 244 – 251 Vol.1, june-1 july 2004. doi:10.1109/ISCC.2004.1358412.

[37] D. Messaoud, D. Djamel, and B. Nadjib. Survey on latency issues of asynchronous mac protocols in delay-sensitive wireless sensor networks. *Communications Surveys Tutorials, IEEE*, PP(99):1 –23, 2012. doi:10.1109/SURV.2012.040412.00075.

[38] Douglas S. J. De Couto, Daniel Aguayo, John Bicket, and Robert Morris. A high-throughput path metric for multi-hop wireless routing. In *Proceedings of the 9th ACM International Conference on Mobile Computing and Networking (MobiCom '03)*, San Diego, California, September 2003.

[39] Olaf Landsiedel, Euhanna Ghadimi, Simon Duquennoy, and Mikael Johansson. Low power, low delay: opportunistic routing meets duty cycling. In *Proc. of IPSN*, 2012.

[40] P. Thubert. Objective Function Zero for the Routing Protocol for Low-Power and Lossy Networks (RPL). RFC 6552 (Proposed Standard), March 2012. URL: http://www.ietf.org/rfc/rfc6552.txt.

[41] O. Gnawali and P. Levis. The Minimum Rank with Hysteresis Objective Function. RFC 6719 (Proposed Standard), September 2012. URL: http://www.ietf.org/rfc/rfc6719.txt.

[42] JP. Vasseur, M. Kim, K. Pister, N. Dejean, and D. Barthel. Routing Metrics Used for Path Calculation in Low-Power and Lossy Networks. RFC 6551 (Proposed Standard), March 2012. URL: http://www.ietf.org/rfc/rfc6551.txt.

[43] M. Doudou, D. Djenouri, and N. Badache. Survey on latency issues of asynchronous mac protocols in delay-sensitive wireless sensor networks. *Communications Surveys Tutorials, IEEE*, 15(2):528–550, Second Quarter 2013. doi:10.1109/SURV.2012.040412.00075.

[44] Messaoud Doudou, Djamel Djenouri, Nadjib Badache, and Abdelmadjid Bouabdallah. Synchronous contention-based {MAC} protocols for delay-sensitive wireless sensor networks: A review and taxonomy. *Journal of Network and Computer Applications*, 38(0):172 – 184, 2014. URL: http://www.sciencedirect.com/science/article/pii/S1084804513000994, doi:http://dx.doi.org/10.1016/j.jnca.2013.03.012.

[45] Pietro Gonizzi, Paolo Medagliani, Gianluigi Ferrari, and Jérémie Leguay. RAW-MAC: A Routing Aware Wave-based MAC protocol for WSNs. In *International Workshop on the GReen Optimized Wireless Networks (GROWN 2014), in conjunction with the 10th IEEE International Conference on Wireless and Mobile Computing, Networking and Communications (WiMob 2014)*, Larnaca, Cyprus, 8-10 October 2014.

[46] Gang Lu, Bhaskar Krishnamachari, and Cauligi S. Raghavendra. An adaptive energy-efficient and low-latency mac for tree-based data gathering in sensor networks: Research articles. *Wirel. Commun. Mob. Comput.*, 7(7):863–875, September 2007.

[47] A.C. Cabezas, R.G. Medina, N.M. Pea T, and M.A. Labrador. Low energy and low latency in wireless sensor networks. In *Proc. of ICC*, 2009.

[48] Paolo Medagliani, Gianluigi Ferrari, Vincent Gay, and JéRéMie Leguay. Cross-layer design and analysis of wsn-based mobile target detection systems. *Ad Hoc Networks*, (2), March 2013.

[49] B. Pavkovic, Won-Joo Hwang, and F. Theoleyre. Cluster-directed acyclic graph formation for ieee 802.15.4 in multihop topologies. In *Proc. of NTMS*, 2012.

[50] Peng Guo, Tao Jiang, Qian Zhang, and Kui Zhang. Sleep scheduling for critical event monitoring in wireless sensor networks. *Parallel and Distributed Systems, IEEE Transactions on*, 23(2):345–352, Feb 2012. doi:10.1109/TPDS.2011.165.

[51] Xavier Vilajosana, Pere Tuset-Peiro, Francisco Vazquez-Gallego, Jesus Alonso-Zarate, and Luis Alonso. Standardized low-power wireless communication technologies for distributed sensing applications. *Sensors*, 14(2):2663–2682, 2014. URL: http://www.mdpi.com/1424-8220/14/2/2663, doi:10.3390/s140202663.

[52] P. Gonizzi, R. Monica, and G. Ferrari. Design and Evaluation of a Delay-Efficient RPL Routing Metric. In *Proceedings of the 9th International Wireless Communications and Mobile Computing Conference* (IWCMC 2013), Cagliari, Italy, July 2013.

[53] Chi-Anh La, Martin Heusse, and Andrzej Duda. Link reversal and reactive routing in low power and lossy networks. In *Personal Indoor and Mobile Radio Communications (PIMRC), 2013 IEEE 24th International Symposium on*, pages 3386–3390, 2013. doi:10.1109/PIMRC.2013.6666733.

[54] Simon Duquennoy, Olaf Landsiedel, and Thiemo Voigt. Let the Tree Bloom: Scalable Opportunistic Routing with ORPL. In *Proceedings of the International Conference on Embedded Networked Sensor Systems (ACM SenSys 2013)*, Rome, Italy, November 2013.

[55] E. Hammer-Lahav. The OAuth 1.0 Protocol. RFC 5849 (Informational), April 2010. Obsoleted by RFC 6749. URL: http://www.ietf.org/rfc/rfc5849.txt.

[56] R. Fielding, J. Gettys, J. Mogul, H. Frystyk, L. Masinter, P. Leach, and T. Berners-Lee. Hypertext Transfer Protocol – HTTP/1.1. RFC 2616 (Draft Standard), June 1999. Obsoleted by RFCs 7230, 7231, 7232, 7233, 7234, 7235, updated by RFCs 2817, 5785, 6266, 6585. URL: http://www.ietf.org/rfc/rfc2616.txt.

[57] T. Dierks and E. Rescorla. The Transport Layer Security (TLS) Protocol Version 1.2. RFC 5246 (Proposed Standard), August 2008. Updated by RFCs 5746, 5878, 6176. URL: http://www.ietf.org/rfc/rfc5246.txt.

[58] D. Hardt. The OAuth 2.0 Authorization Framework. RFC 6749 (Proposed Standard), October 2012. URL: http://www.ietf.org/rfc/rfc6749.txt.

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

[60] Simone Cirani, Gianluigi Ferrari, and Luca Veltri. Enforcing Security Mechanisms in the IP-Based Internet of Things: An Algorithmic Overview. *Algorithms*, 6(2):197–226, 2013. URL: http://www.mdpi.com/1999-4893/6/2/197, doi:10.3390/a6020197.

[61] Huansheng Ning, Hong Liu, and L.T. Yang. Cyberentity Security in the Internet of Things. *Computer*, 46(4):46–53, April 2013. doi:10.1109/MC.2013.74.

[62] Xuanxia Yao, Xiaoguang Han, Xiaojiang Du, and Xianwei Zhou. A Lightweight Multicast Authentication Mechanism for Small Scale IoT Applications. *Sensors Journal, IEEE*, 13(10):3693–3701, Oct 2013. doi:10.1109/JSEN.2013.2266116.

[63] C. Lai, H. Li, X. Liang, R. Lu, K. Zhang, and X. Shen. CPAL: A Conditional Privacy-Preserving Authentication With Access Linkability for Roaming Service. *Internet of Things Journal, IEEE*, 1(1):46–57, Feb 2014. doi:10.1109/JIOT.2014.2306673.

[64] Fagen Li and Pan Xiong. Practical Secure Communication for Integrating Wireless Sensor Networks Into the Internet of Things. *Sensors Journal, IEEE*, 13(10):3677–3684, Oct 2013. doi:10.1109/JSEN.2013.2262271.

[65] D. Forsberg, Y. Ohba, B. Patil, H. Tschofenig, and A. Yegin. Protocol for Carrying Authentication for Network Access (PANA). RFC 5191 (Proposed Standard), May 2008. Updated by RFC 5872. URL: http://www.ietf.org/rfc/rfc5191.txt.

[66] B. Aboba, L. Blunk, J. Vollbrecht, J. Carlson, and H. Levkowetz. Extensible Authentication Protocol (EAP). RFC 3748 (Proposed Standard), June 2004. Updated by RFCs 5247, 7057. URL: http://www.ietf.org/rfc/rfc3748.txt.

[67] P. Moreno-Sanchez, R. Marin-Lopez, and F. Vidal-Meca. An open source implementation of the protocol for carrying authentication for network access: OpenPANA. *Network, IEEE*, 28(2):49–55, March 2014. doi:10.1109/MNET.2014.6786613.

[68] United States Department of Defense. Department of Defense Trusted Computer System Evaluation Criteria. Technical report, United States Department of Defense, December 1985. URL: http://csrc.nist.gov/publications/history/dod85.pdf.

[69] David Ferraiolo and Richard Kuhn. Role-Based Access Control. In *Proceedings of the 15th NIST-NCSC National Computer Security Conference*, pages 554–563, Baltimore, MD, USA, October 1992.

[70] David F. Ferraiolo, Ravi Sandhu, Serban Gavrila, D. Richard Kuhn, and Ramaswamy Chandramouli. Proposed NIST standard for role-based access control. *ACM Trans. Inf. Syst. Secur.*, 4(3):224–274, August 2001. URL: http://doi.acm.org/10.1145/501978.501980, doi:10.1145/501978.501980.

[71] R.S. Sandhu, E.J. Coyne, H.L. Feinstein, and C.E. Youman. Role-based access control models. *Computer*, 29(2):38 –47, February 1996. doi:10.1109/2.485845.

[72] E. Yuan and J. Tong. Attributed based access control (ABAC) for Web services. In *Proceedings of the 2005 IEEE International Conference on Web Services, 2005 (ICWS 2005).*, pages 2 vol. (xxxiii+856), July 2005. doi:10.1109/ICWS.2005.25.

[73] J. Schiffman, Xinwen Zhang, and S. Gibbs. DAuth: Fine-Grained Authorization Delegation for Distributed Web Application Consumers. In *Proceedings of the 2010 IEEE International Symposium on Policies for Distributed Systems and Networks (POLICY)*, pages 95–102, July 2010. doi:10.1109/POLICY.2010.12.

[74] Oscar Garcia-Morchon and Klaus Wehrle. Modular context-aware access control for medical sensor networks. In *Proceedings of the 15th ACM symposium on Access control models and technologies*, SACMAT '10, pages 129–138, New York, NY, USA, 2010. ACM. URL: http://doi.acm.org/10.1145/1809842.1809864, doi:10.1145/1809842.1809864.

[75] S. Gerdes, O. Bergmann, and C. Bormann. Delegated CoAP Authentication and Authorization Framework (DCAF). Technical report, IETF Internet Draft draft-gerdes-ace-dcaf-authorize-00, July 2014. URL: http://tools.ietf.org/html/draft-gerdes-ace-dcaf-authorize-00.

[76] OpenID Authentication 2.0 - Final. Technical report, OpenID Foundation, December 2007. URL: http://openid.net/specs/openid-authentication-2_0.html.

[77] E. Rescorla and N. Modadugu. Datagram Transport Layer Security Version 1.2. RFC 6347 (Proposed Standard), January 2012. URL: http://www.ietf.org/rfc/rfc6347.txt.

[78] S. Kent and R. Atkinson. Security Architecture for the Internet Protocol. RFC 2401 (Proposed Standard), November 1998. Obsoleted by RFC 4301, updated by RFC 3168. URL: http://www.ietf.org/rfc/rfc2401.txt.

[79] Connect All IP-based Smart Objects (CALIPSO) - FP7 EU Project. URL: http://www.ict-calipso.eu/.

[80] The Contiki Operating System. URL: http://www.contiki-os.org.

[81] Shahid Raza, Simon Duquennoy, Tony Chung, Dogan Yazar, Thiemo Voigt, and Utz Roedig. Securing Communication in 6LoWPAN with Compressed IPsec. In *Proceedings of the International Conference on Distributed Computing in Sensor Systems (IEEE DCOSS 2011)*, Barcelona, Spain, June 2011.

[82] Shahid Raza, Daniele Trabalza, and Thiemo Voigt. 6LoWPAN Compressed DTLS for CoAP. In *Proceedings of the 8th IEEE International Conference on Distributed Computing in Sensor Systems (IEEE DCOSS 2011)*, Hangzhou, China, May 2012.

[83] Matthias Kovatsch, Simon Duquennoy, and Adam Dunkels. A Low-Power CoAP for Contiki. In *Proceedings of the Workshop on Internet of Things Technology and Architectures (IEEE IoTech 2011)*, Valencia, Spain, October 2011.

[84] H. Krawczyk, M. Bellare, and R. Canetti. HMAC: Keyed-Hashing for Message Authentication. RFC 2104 (Informational), February 1997. Updated by RFC 6151. URL: `http://www.ietf.org/rfc/rfc2104.txt`.

[85] G. Bertoni, J. Daemen, M. Peeters, and G. Van Assche. The Keccak SHA-3 submission. Submission to NIST (Round 3), 2011. URL: `http://keccak.noekeon.org/Keccak-submission-3.pdf`.

[86] O. Garcia-Morchon, S. Keoh, S. Kumar, R. Hummen, and R. Struik. Security Considerations in the IP-based Internet of Things. Technical report, IETF Internet Draft draft-garcia-core-security-04, March 2012. URL: `http://tools.ietf.org/id/draft-garcia-core-security-04`.

[87] DavidJ. Wheeler and RogerM. Needham. TEA, a tiny encryption algorithm. In Bart Preneel, editor, *Fast Software Encryption*, volume 1008 of *Lecture Notes in Computer Science*, pages 363–366. Springer Berlin Heidelberg, 1995. URL: `http://dx.doi.org/10.1007/3-540-60590-8_29`, `doi:10.1007/3-540-60590-8_29`.

[88] Francois-Xavier Standaert, Gilles Piret, Neil Gershenfeld, and Jean-Jacques Quisquater. SEA: A Scalable Encryption Algorithm for Small Embedded Ap-

plications. In Josep Domingo-Ferrer, Joachim Posegga, and Daniel Schreckling, editors, *Smart Card Research and Advanced Applications*, volume 3928 of *Lecture Notes in Computer Science*, pages 222–236. Springer Berlin Heidelberg, 2006. URL: http://dx.doi.org/10.1007/11733447_16, doi:10.1007/11733447_16.

[89] A. Bogdanov, L.R. Knudsen, G. Leander, C. Paar, A. Poschmann, M.J.B. Robshaw, Y. Seurin, and C. Vikkelsoe. PRESENT: An Ultra-Lightweight Block Cipher. In Pascal Paillier and Ingrid Verbauwhede, editors, *Cryptographic Hardware and Embedded Systems - CHES 2007*, volume 4727 of *Lecture Notes in Computer Science*, pages 450–466. Springer Berlin Heidelberg, 2007. URL: http://dx.doi.org/10.1007/978-3-540-74735-2_31, doi:10.1007/978-3-540-74735-2_31.

[90] Andrey Bogdanov, Gregor Leander, Christof Paar, Axel Poschmann, Matt J. Robshaw, and Yannick Seurin. Hash Functions and RFID Tags: Mind the Gap. In *Proceedings of the 10th International Workshop on Cryptographic Hardware and Embedded Systems*, CHES '08, pages 283–299, Berlin, Heidelberg, 2008. Springer-Verlag. URL: http://dx.doi.org/10.1007/978-3-540-85053-3_18, doi:10.1007/978-3-540-85053-3_18.

[91] Guido Bertoni, Joan Daemen, Michael Peeters, and Gilles Van Assche. KECCAK specifications. Technical report, 2009. URL: http://keccak.noekeon.org/Keccak-specifications.pdf.