Réseaux de capteurs sans fils multi-sauts à récupération d'énergie : routage et couche liaison de bas rapport cyclique

Liviu - Octavian Varga

Communauté
UNIVERSITÉ Grenoble Alpes

THÈSE

Pour obtenir le grade de

DOCTEUR DE L'UNIVERSITÉ DE GRENOBLE

Spécialité : **Mathématiques & Informatique**

Arrêté ministérial : 7 août 2006

Présentée par

Liviu-Octavian VARGA

Thèse dirigée par **Martin Heusse**
et codirigée par **Roberto Guizzetti**

préparée au sein du **Laboratoire d'Informatique de Grenoble (LIG), STMicroelectronics Crolles (France)**
dans l'**École Doctorale Mathématiques, Sciences et Technologies de l'Information, Informatique (EDMSTII)**

Multi-hop Energy Harvesting Wireless Sensor Networks:Routing and Low Duty-cycle Link Layer

Réseaux de capteurs sans fils multi-saut à récupération d'énergie: routage et couche liaison de bas rapport cyclique

Thèse soutenue publiquement le **16 Décembre 2015**,
devant le jury composé de :

Mme. Isabelle Guérin Lassous
Professeur, Université Lyon 1, Présidente
M. Thomas Noel
Professeur, Université de Strasbourg, Rapporteur
M. Alexandre Guitton
Maître de Conférences, Université Blaise Pascal, Rapporteur
M. Martin Heusse
Professeur, Grenoble INP, Directeur de thèse
M. Roberto Guizzetti
Ingénieur, STMicroelectronics, Co-Encadrant de thèse

Multi-hop Energy Harvesting Wireless Sensor Networks: routing and low duty-cycle link layer

Abstract:

The goal of the thesis is to enable IPv6 harvested and autonomous wireless sensor networks with very low duty-cycle. It is part of an industrial project, GreenNet, hosted by STMicroelectronics with the goal of being a pioneer in the Internet of Things. The new platform differentiates from its existing competitors by a small size, which implies small battery capacity. However, a photovoltaic cell is capable of recharging the battery even under low light conditions. On top of this, we aim at nodes that sleep for very long periods. Hence, the existing solutions were not completely suited for our needs.

The thesis proposes to analyze the possible challenges that one can meet while developing a harvested low-duty cycle platform. The most important contribution of this work is that we implement and evaluate the performance of our solutions on real hardware platforms in conditions very close to real-life.

In this dissertation, we first of all develop and implement a basic solution based on the IEEE 802.15.4 beacon-enabled standard. We choose the synchronized mode because it allows nodes to reach duty-cycles as low as 0.01%. A more difficult step was to bring multi-hop: we design a new routing scheme inside our network, and a time based access for routers and devices to eliminate interferences as much as possible. The routing scheme is meant to be simple and efficient.

We go even further to optimize the total time the nodes are on: we proposed to shut down coordinators before their standardized end of slot when there is no communication. Devices that do not need to send data can skip beacons and only need to wake up to synchronize their clock or to send data. In the same time we solve the problem of multicast for long sleeping nodes by converting these packets into unicast traffic. We also improved the duty-cycle of routers that do no have associated devices by forcing them to beacon at a slower rate, as long as they do not have any associated devices.

To improve the network performance we also propose a backward compatible multi-channel solution. Such a scheme is useful when a link between two nodes achieves very bad performance on a certain channel but better results on a different frequency.

All the solutions presented above and discussed in the dissertation were implemented and tested on the GreenNet platform. We also realized measurements of the nodes efficiency while in harvested conditions and showed that it is possible to handle harvested routers, when there is enough available light.

Keywords: IEEE 802.15.4, Harvesting Networks, Network Configuration, Wireless Sensor Networks, Energy Efficiency, Low duty-cycle

Réseaux de capteurs sans fils multi-sauts à récupération d'énergie: routage et couche liaison de bas rapport cyclique

Résumé:

L'objectif de cette thèse est de développer un réseau IPv6 constitué de capteurs sans fils autonomes grâce à la récupération d'énergie, fonctionnant à faible rapport cyclique. Cette thèse s'inscrit dans un projet industriel, GreenNet, lancé par STMicroelectronics afin de se positionner sur le marché de l'Internet des Objets. La nouvelle plate-forme utilisée dans ce projet se différencie de ses concurrents par sa petite taille, ce qui implique une faible capacité de batterie. Une cellule photovoltaïque permet en revanche de recharger la batterie, y compris dans des conditions de luminosité faible. Pour atteindre l'autonomie, nous avons besoin que les nœuds dorment pour de très longues périodes. Par conséquent, les solutions existantes, bien que peu consommantes, ne sont pas complètement adaptées à nos besoins spécifiques.

Dans cette thèse, nous proposons d'analyser les difficultés possiblement rencontrées pendant le développement d'une plate-forme à récupération d'énergie et de bas rapport cyclique. La contribution la plus importante de ce travail est de mettre en œuvre et d'évaluer le rendement de nos solutions sur des plates-formes matérielles dans des conditions très proches de la vie réelle.

Une première étape du travail réalisée est la conception et l'implémentation de la norme IEEE 802.15.4 utilisant les balises pour maintenir la synchronisation. Nous choisissons le mode synchronisé car il permet aux nœuds d'atteindre des rapports cycliques aussi bas que 0,01%. La seconde étape est d'apporter le multi saut : nous proposons une optimisation du protocole de routage, ainsi qu'un contrôle d'accès par multiplexage temporel pour les routeurs et les dispositifs afin d'éliminer les interférences.

Nous allons même plus loin dans l'optimisation du temps où les nœuds sont allumés: nous proposons d'éteindre les coordinateurs avant la fin de leur période d'activité définie par le standard, lorsqu'il n'y a pas de communications. Les nœuds qui ne nécessitent pas d'envoyer des données peuvent sauter des balises et se réveiller seulement lorsqu'il est nécessaire de synchroniser les horloges, ou d'envoyer des données. Dans le même temps, nous résolvons le problème de multicast pour les nœuds qui dorment durant de longues périodes, en convertissant ces paquets en paquets unicast. Nous améliorons également le rapport cyclique de routeurs qui n'ont pas de nœuds associés en les forçant envoyer la balise moins souvent, tant qu'ils n'ont pas des nœuds associés.

Pour améliorer la performance du réseau, nous proposons aussi une solution rétrocompatible qui utilise plusieurs canaux. Un tel système est utile quand un lien entre deux nœuds subit de très mauvaise performance sur un certain canal fréquentiel, mais obtient de meilleurs résultats sur une fréquence différente.

Toutes les solutions présentées ci-dessus, et discutées dans la dissertation ont été mises en œuvre et testées sur la plate-forme GreenNet. Nous avons également réalisé des mesures sur des nœuds pour vérifier leurs efficacité.

Mots clés: IEEE 802.15.4, Récupération d'Énergie, Configuration de Réseau, Réseau de Capteurs, Efficacité Énergétique, Bas Rapport Cyclique

Acknowledgements

First of all, my research would not have been possible without the partnership between ST Microelectronics and Laboratoire d'Informatique de Grenoble (LIG), therefore I have a special appreciation for all those who made it possible.

I would like to express my special appreciation and thanks to my advisors, Martin Heusse and Roberto Guizzetti. You have been a tremendous mentor for me. I would like to thank you for encouraging my research and for allowing me to grow as a research scientist. Your advice on both research as well as on engineering aspects have been priceless.

I would like to give a special thank to Michel Favre for his extremely valuable advice and encouragements.

The members of GreenNet *team have contributed immensely to my personal and professional development. The team has been a source of friendship as well as good advice and collaboration.*

I would also like to thank my committee members for agreeing to judge my work, for their time and for their comments and suggestions.

Furthermore I would like to thank all of my friends who supported me in writing, and incented me to strive towards my goal.

Last but not least, a special thanks goes to my family. Words cannot express how grateful I am to my mother, and father for the support they provided me throughout my studies.

Contents

List of Figures

List of Tables

Abbreviations and Acronyms

6LoWPAN	IPv6 over Low power Wireless Personal Area Networks
ACK	Acknowledgement
ADC	Analog to Digital Converter
AES	Advanced Encryption Standard
AODV	Ad hoc On-Demand Distance Vector
BI	Beacon Interval
BO	Beacon Order
CAP	Contention Access Period
CCA	Clear Channel Assessment
CoAP	Constrained Application Protocol
CSMA/CA	Channel Sensing Medium Access/Collision Avoidance
CTS	Clear to Send
DAC	Digital to Analog Converter
DC	Duty Cycle
EUI-64	64-bit Extended Unique Identifier
FFD	Full Function Device
GW	Gateway
HTTP	Hypertext Transfer Protocol
HWSN	Harvesting Wireless Sensor Network
IEEE	Institute of Electrical and Electronics Engineers
IETF	Internet Engineering Task Force
IoT	Internet of Things
IP	Internet Protocol
IPv6	Internet Protocol version 6
Li	Lithium
LLN	Low-power Lossy Network
LQI	Link Quality Indicator
LR-WPAN	Low Rate Wireless Personal Area Network
MAC	Medium Access Control
MTU	Maximum Transmission Unit
OS	Operating System
PAN	Personal Area Networks
PHY	Physical Layer
PMU	Power Management Unit
PV	Photovoltaic
RDC	Radio Duty Cycle
RFC	Request for Comments
RFD	Reduced Function Device
RPL	Routing Protocol for Low-Power and Lossy Networks
RSSI	Received Signal Strength Indicator
RTS	Request to Send
SD	Superframe Duration
SO	Superframe Order

STM	STMicroelectronics
TCP	Transmission Control Protocol
UDP	User Datagram Protocol
USB	Universal Serial Bus
WSN	Wireless Sensor Network
ZDAA	ZigBee Distributed Address Allocation

Chapter 1

Introduction

"Simplicity is the ultimate form of sophistication."

William Gaddis

Contents

1.1 Motivation

Internet of Things is one of the most popular topics nowadays with more and more objects becoming "intelligent" thanks to their capability to connect with other devices. Most of the big players in Internet market are predicting that in a few years time, there will be billions of such connected objects, so the demand of research and work in this field is in continuously growth.

Looking at how smartphones have developed these years, we could imagine the same evolution with respect to the Internet of Things. One of the main advantages of smartphones was to give to developers the opportunity of creating and innovating on a wide range of devices, all of them running on top of at most two or three different operating systems. Thanks to the fact that a developer does not have to reprogram his/her application for tens or hundreds of devices but for only a couple of platforms, we observed a high expansion of third party softwares running on mobile phones.

The same kind of "revolution" could happen with all smart objects connected to Internet. The biggest challenge is then the following : how to offer a unified platform which manages to abstract as much as possible these networks and devices in a way that encourages their usage on a large scale.

One good example of unified solution for Internet of Things is "If This Then That" platform [4], a web-based service that allows users to create chains of simple conditional statements, called "recipes". It allows any Internet user (not necessarily programmers) to build connections between different services. We can apply the same reasoning for the Wireless Sensors Networks where we have different nodes sensing and acting. The interaction between them could be realised through intuitive user interfaces : any computer user could define the behaviour of his network. Final users would have a large freedom in personalizing their networks. However, to reach this high level of

abstraction, we first need to design an efficient networking stack working at the lower layers (radio and communication protocols): this is the main focus of this thesis.

The very first projects of Wireless Sensor Networks already start to hit the market, especially the so called *smart buildings* and *smart homes* solutions. These networks can include temperature monitoring, inside or outside, windows position, movement detection or switches (lights on/off, opening/closing a garage door, etc).

A self sustaining network would be composed of sensors and nodes that have minimum constraints for a laymen and are easy to configure. The thesis focuses on providing solutions to efficiently enable connectivity of such sensors while keeping energy consumption at minimum. We are mostly interested in energy consumption because we target autonomous nodes from the energy point of view. The typical operation mode we are targeting is duty-cycle : for a short period of time the node is active and communicating. The rest of the time, it sleeps and might charge a small battery, if the node has harvesting capabilities (Fig. 1.1).

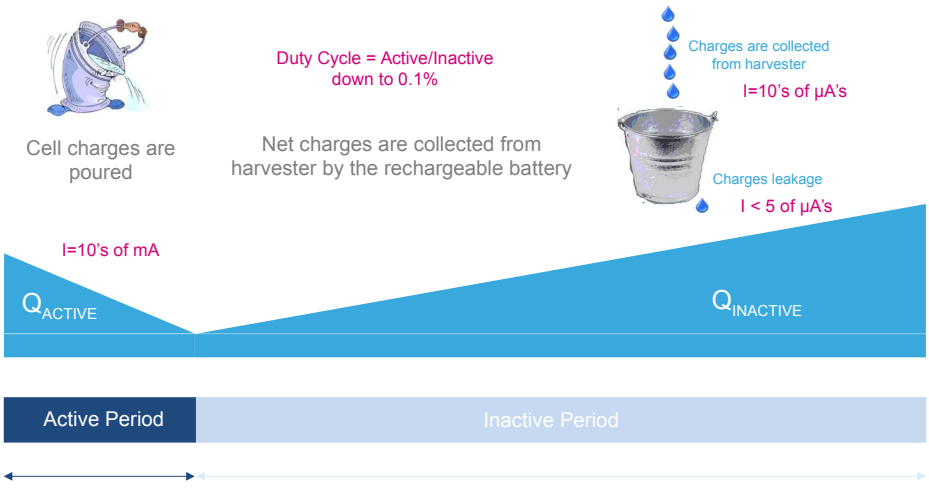


Figure 1.1: Typical duty-cycling operation

Early work in sensor networks suggested that the constrained and application-specific nature of sensor networks would require networking to be based on non-IP concepts [5, 6]. This resulted in a body of work on clean-slate communication architectures for wireless sensor networks [7–10]. Although such systems could achieve very low power consumption and reasonable performance [11–13], they incur a significant complexity due to cross-layer interactions (Fig. 1.2) and non-modular designs. Recently, IP-based sensor networking platforms became of interest [14–19], understanding that layered IP-based sensor network systems could be as efficient as those based on monolithic designs.

Sensor networks traditionally consist in isolated and homogeneous networks of sensors that periodically report their data to a sink [20–22]. A decade of research in the sensor networks community resulted in algorithms, mechanisms and protocols that address this domain's unique challenges.

Isolated sensor networks worked well in the past, but the new set of emerging applications in IP-based smart objects increases the demands for integration within

existing network infrastructures, to cope with heterogeneity in hardware, software and communication technologies [23]. IP-based sensor networks [16,18,24,25] have recently become popular for this emerging area. The idea of using IP in sensor networks is not new, but IP-based sensor networks have only recently become widespread. The Contiki operating system explored IPv4 communication for sensor networks [15, 24] and later provided the first fully certified IPv6 stack for IP-based smart objects [25].

Hui and Culler developed an IPv6 architecture for low-power sensor networks based on IEEE 802.15.4 [16, 26]. Although significant progress has been made in routing protocols for low-power IP, as demonstrated by the emerging RPL IETF standard [27, 28], the integration of IP routing with standard low & ultra-low duty cycling MAC layers is still a challenge that we have addressed within our project, called GreenNet.

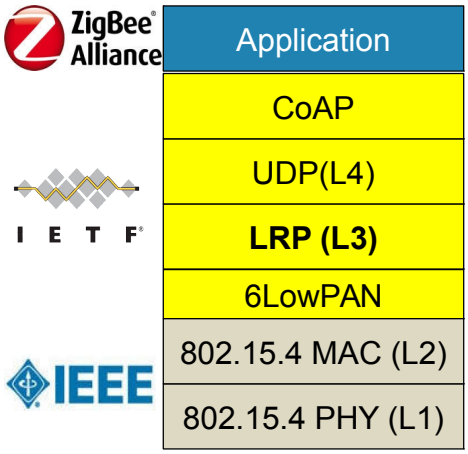


Figure 1.2: Networking layers

The GreenNet network is composed of sensor nodes with a small photo-voltaic cell capable of energy harvesting from ambient light that recharges a small coin-cell battery.

The first steps of GreenNet project was to design and develop a hardware platform that uses efficient and low power components. The main controller and the radio, at the time of development, were among the most efficient ones. Another important component, the Power Management Unit, was completely developed by GreenNet team [29]. Once we had a hardware platform that we could use, the next milestone was to design, implement and optimize an efficient communication stack that would be able to use at best the capabilities of the hardware. The objective was to have a node running properly on harvested energy only, under very low light (less than 200 Lux). In order to obtain this we needed a very low duty cycle stack (0.1% - 1%). This lead us to choose the IEEE 802.15.4 [30] beacon-enabled mode but it was not completely tailored to our needs: most of the work was to adjust this stack with our objectives.

Other contributions to GreenNet project is related to security. In [31] the authors propose an architecture (OSCAR) that leverages the security concepts both from content-centric and traditional connection-oriented approaches. It relies on secure channels established by means of (D)TLS for key exchange, but gets rid of the notion of the "state" among communicating entities. It provides a mechanism to protect from

replay attacks by coupling the scheme with the CoAP application protocol. The object-based security architecture intrinsically supports caching and multicast, and does not affect the radio duty-cycling operation of constrained objects.

Figure 1.3: Graphical User Interface of GreenNet network

1.2 Contributions and organization of this thesis

The main contributions of the GreenNet project and in particular this thesis is to enable autonomous Wireless Sensors Networks running on an energy efficient platform using IP communication. We provide a complete solution starting from hardware and low level firmware all the way to high level software (Fig. 1.3) and communication protocols.

Most of the work focused on using the GreenNet platform for either evaluation, implementation or testing of different solutions. This gives a rather practical character to the work and contributions.

The thesis started after six months as intern in the same team and project. During this time I could already get familiar with the software tools, hardware platform and also long term objectives.

At the beginning of the thesis the project already had in place some of the software and protocol stack. This included the use of ContikiOS along with a partially implemented IEEE 802.15.4 beacon-enabled protocol. Accordingly, the thesis focuses on improving and proposing solutions of an existing but not well developed protocol stack.

Among the first steps is to enable routing and implement the association protocol to be able to create a star-topology network. This involved the proper use of the radio timers and embedding routing information within beacon frames (cf. Chapter 7). The next step is to have nodes acting not only as simple sensors that send data but also as relays (routers), in order to create a multi-hop network (cf. Chapter 6). At this point we needed to ask ourselves questions on how to schedule the exact moment to send beacons for multi-function devices. This pushed us to investigate what are the radio capability in terms of reception performance when several packets collide.

Once we had this rather stable stack that could do multi-hop, the next logical step was to improve the performance at different layers of the code (firmware, communication protocol). On top of this we aimed at a more robust solution: we needed a version that could communicate on various channels while remaining compatible with the single channel mode of operating. In the end, we measured different performance aspects of the entire stack (current consumption, packet delivery ratio etc) to confirm our improvements.

The thesis is organized in two major parts. The first part presents the state of the art from hardware and software/protocols point of view. Chapter 2 presents the most appropriate radio technologies for IoT, followed by Chapter 3, that introduces the hardware platforms used by the community for development. Chapter 4 takes a look at four of the most popular operating systems for WSN and finally Chapter 5 describes the current state of the art of communication protocols.

The second part focuses on the contributions of the thesis. Chapter 6 presents several propositions to improve lifetime of a network running with a IEEE 802.15.4 beacon-enabled MAC protocol. Chapter 7 presents the improved routing scheme in GreenNet. Chapter 8 focuses on the complete stack evaluation and energy consumption. Chapter 9 describes a solution that improves the service quality by using multiple channels while remaining backward compatible with single channel usage. Chapter 10 presents the industrial side of the thesis.

Part I

State of the Art

Chapter 2

Radio Technologies

Contents

This chapter focuses on presenting radio technologies that aim at enabling the development of IoT. The choice of which radio technology to use is in close connection to the profile of the targeted use cases. According to this, one technology could be more advantageous than another (latency, range, robustness, etc.). It is important to keep in mind that a particular radio hardware has also a big impact on the protocol communication layers. Besides, a single hop network is simpler and faster to deploy than a multi-hop one. However, it requires a centralized node or tower to allow direct communication with other nodes.

The following chapter will review the existing MAC layer protocols for WSN, in order to help readers understand the IoT field, in terms of MAC technologies.

2.1 IEEE 802.15.4

The first technology reviewed here is also the one that the thesis focuses on: the IEEE 802.15.4 standard [30]. It is intended to enable communications in harsh environments while remaining energy efficient.

ZigBee Alliance [32] is the entity that has mostly pushed the usage of the IEEE 802.15.4 physical layer and tried to become the leader of the Internet of Things. The alliance's latest standard ZigBee 3.0 is a unification of their previously released standards while attempting to simplify development. The used networking layer is ZigBee Pro, that does not support IP protocol (Fig. 2.1).

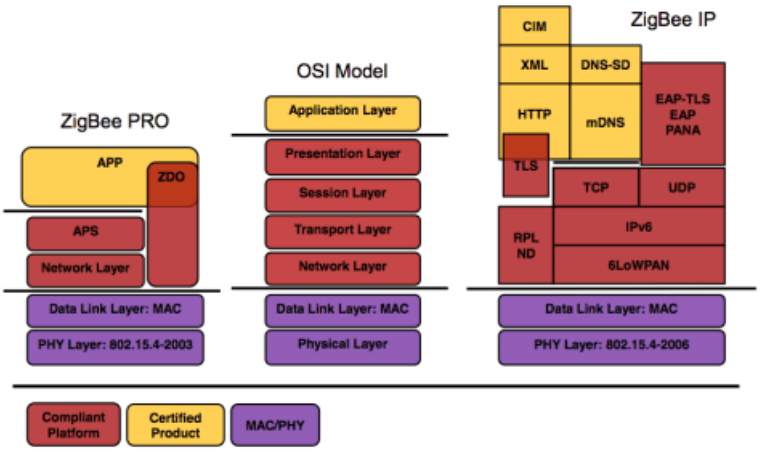


Figure 2.1: ZigBee protocol layers

IEEE 802.15.4 devices are expected to transmit from 10 to 100 meters, depending on the environment, and operate on one of three possible unlicensed frequency bands (868 MHz in Europe, 915 MHz in North America and 2.4 GHz worldwide). The data rate for the most used frequency of 2.4 GHz is 250kbps. There are 16 available channels (numbered from 11 to 26) with a 5 MHz difference between two consecutive channels. Each channel is 2 MHz wide. The modulation used is O-QPSK which is a 16-ary quasi-orthogonal modulation technique. This modulation offers a good reliability in noisy environments. The O-QPSK shall be capable of transmitting at a power level of at least -3dBm. The data rate is 4 bits/symbol, 62.5 kBauds and with a chip rate of 2.0 Mchips/s.

The spreading gain helps to successfully receive frames even in presence of interference, such as from 802.11 transmitters (*i.e.* Fig. 2.2), which are harmless as long as they are further away than the 802.15.4 source. 802.11 signal power is spread over 22 MHz, of which only 10% are caught by a 802.15.4 receiver in an overlapping band. So the initial power difference of $\approx$ 20dB is reduced to 10dB, which is of the same order of magnitude as the spreading gain.

One of the important limitations of IEEE 802.15.4 standard is the maximum frame length of 127 bytes. As a consequence, a lot of research has been dedicated to reduce various fields of the frame, like addresses, to have a bigger payload size. This gave rise to the 6LoWPAN [33] working group. Expecting a very large number of devices connected to Internet, the only solution for 802.15.4 standard was to use the IPv6 protocol. Compared to IPv4, that uses 32 bits for addresses, IPV6 defines addresses on 128 bits, allowing up to $3.4 \cdot 10^{38}$ unique addresses. The goal of the 6LoWPAN group is to provide encapsulation, header compression mechanism and most importantly fragmentation.

The general MAC frame format is presented in Fig. 2.3, where the MAC header (MHR) contains the specific 802.15.4 fields followed by the useful MAC payload and finally by the MAC footer (MFR).

A frame can be transmitted with the *acknowledge request* bit set. In that case,

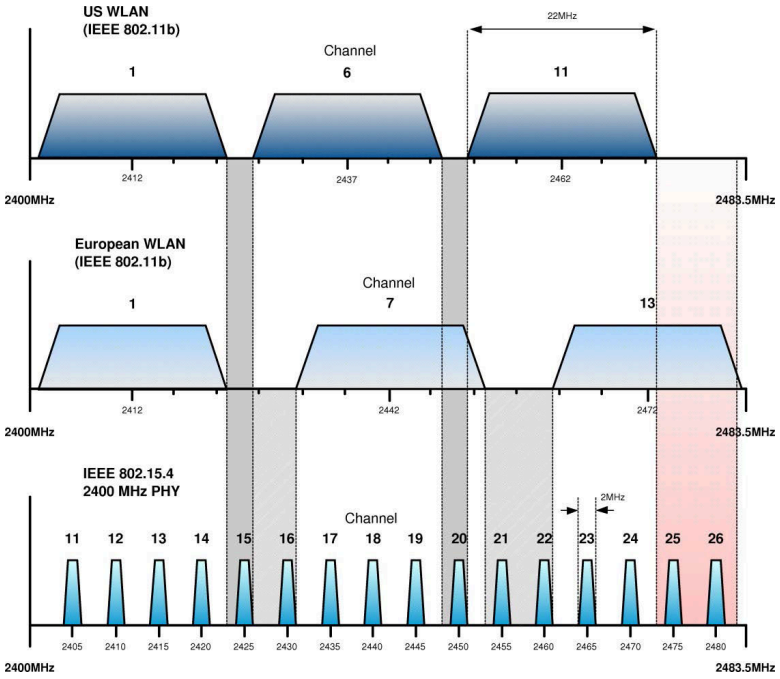


Figure 2.2: 802.15.4 channel spectrum vs. Wi-Fi frequency channels [1]

Octets: 1/2	0/1	0/2	0/2/8	0/2	0/2/8	variable	variable	variable	2/4
Frame Control	Sequence Number	Destination PAN Identifier	Destination Address	Source PAN Identifier	Source Address	Auxiliary Security Header	Information Elements (IE)	Frame Payload	FCS
		Addressing fields					Header IEs / Payload IEs		
MHR							MAC Payload		MFR

Figure 2.3: Frame format for 802.15.4

if the recipient does not acknowledge the frame, the sender will retry to send it. The choice of the number of retries is left to the developer.

In the IEEE 802.15.4 [30] standard there are two different types of devices that can participate in a network: a full-function device (FFD) or a reduced-function device (RFD). An FFD device can serve as a Personal Area Network (PAN) coordinator or a coordinator, whereas a RFD is intended for applications that are extremely simple, such as a light switch or a passive sensor. At the same time a RFD does not need to send large amounts of data and associates with a single FFD at a time.

Network Topologies. Depending on the application requirements an IEEE 802.15.4 network can be either a star topology or peer-to-peer topology (Fig. 2.4). An example of peer-to-peer topology is the cluster tree. In the cluster tree topology, the PAN coordinator is the root of the network. It serves as a data sink and represents the first coordinator in the cluster-tree.

The two main single channel access methods are the unslotted CSMA-CA used in

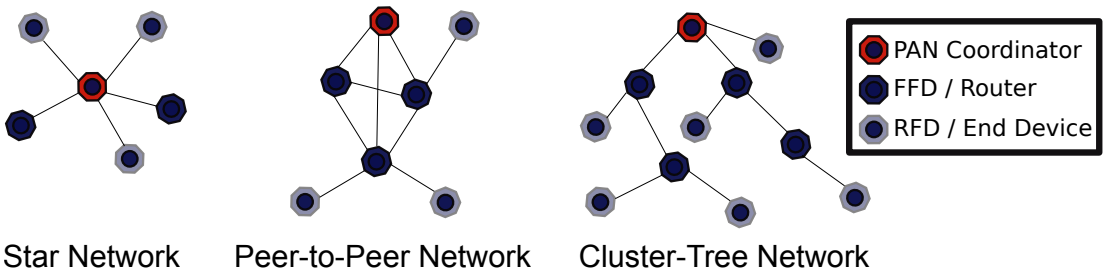


Figure 2.4: IEEE 802.15.4 topologies

nonbeacon-enabled PANs and the slotted CSMA-CA used in beacon-enabled PANs.

It is important to note that during the back-off of the CSMA-CA in the slotted mode, the radio is off, whereas in unslotted mode, the radio is on during the entire CSMA-CA protocol.

Using the unslotted access method, a node, typically the gateway (FFD), needs to stay always on. A device (RFD) can just wake up and send data only when it is necessary, in a push only mechanism. This reduces energy consumption to minimum on the device's side but with the consequence of a higher consumption on the router side.

2.1.1 IEEE 802.15.4 beacon-enabled mode

The main purpose of using the beacon-enabled mode is to save energy in multi-hop networks. This is because of the active/inactive periods that can vary from a few milliseconds up to 4 minutes. On top of this, the slotted CSMA-CA method keeps the radio off during the back-off. It is important to mention that beacons are sent without any CSMA-CA, like broadcast packets. In synchronized mode, FFD devices can also work with low duty-cycle. This contributes to a lot of energy saving not only for RFD but also for devices that act as routers. The standard also puts in place a signaling mechanism so that bidirectional communication is possible with RFD nodes. Nodes that want to join a network first need to associate and be synchronized with a coordinator.

Creating a cluster tree topology means that a device will act, at some point, as RFD and at a different moment as FFD. If it acts as router it will have to beacon but when it will send the beacon frame exactly is not specified in the standard, leaving the implementation open. ZigBee [34] defines a protocol for cluster-tree construction based on three constraints: a maximum number of devices, a maximum number of coordinators and a maximum depth.

In the cluster tree topology, nodes are unassociated at the beginning and they wait for beacons (passive scanning) from coordinators to join the network. The channel is potentially unknown. Passive scanning is the only available discovery mechanism in the beacon-enabled mode.

When a node receives a beacon from a neighbor, it may associate with it by exchanging control frames. A coordinator maintains a list of devices and responds with an association response if it has not reached the maximum limit of associated devices.

After association, the node may become a coordinator itself: it periodically sends its beacons to invite other nodes to associate.

Beacons also indicate the starting point of the Active Period and contain the list of destination addresses for frames stored at the coordinator. During the Active Period, a device either retrieves frames by transmitting a data-request frame if its address was present in the pending destination list or transmits its data frames to the coordinator. Note that the coordinator never initiates a transmission (except broadcast frames, that are anyway signaled with a flag in the beacon header), but only replies to solicitations from its devices. Devices have to explicitly request their frames from a coordinator, which enables switching off their radio and saving energy without deafness. To avoid collisions, all devices use the slotted CSMA-CA method to access the medium.

Coordinators act as devices with respect to their coordinators when they forward packets to the root of the cluster tree. To support forwarding over multiple hops, IEEE 802.15.4 defines the Outgoing (used to send beacons and communication with associated nodes) and the Incoming superframes (used to receive beacons and send data to the coordinator) interspaced with *StartTime* (cf. Figure 2.5).

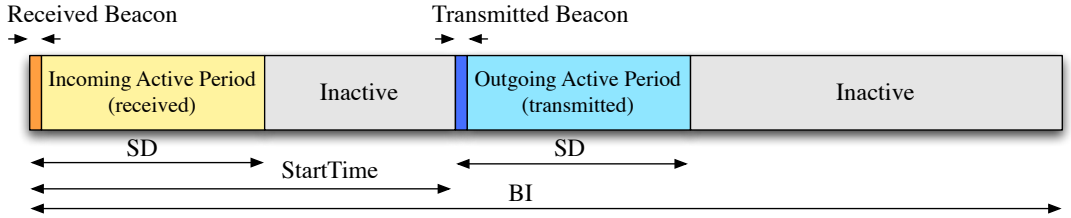


Figure 2.5: Incoming and Outgoing superframe structure in IEEE 802.15.4

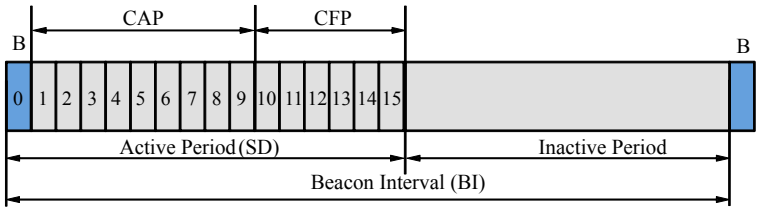


Figure 2.6: 802.15.4 superframe structure. GreenNet nodes only use the Contention Access Period (they do not use the Contention Free Period).

Superframe structure. Figure 2.6 presents the structure used in beacon-enabled mode. Coordinators transmit beacons every Beacon Interval (BI) while a superframe lasts for a Superframe Duration (SD). The intervals depend on the corresponding Beacon Order (BO) and Superframe Order (SO) parameters:

$$\mathrm{BI} = aBaseSuperFrameDuration * 2^{\mathrm{BO}}$$

$$\mathrm{SD} = aBaseSuperFrameDuration * 2^{\mathrm{SO}}$$

$$(0 \leq \mathrm{SO} \leq \mathrm{BO} \leq 14, \textit{ i.e. } 15.36ms \leq \mathrm{SD},\ \mathrm{BI} \leq 4.2\ min).$$

SO (BO)	SD (BI)	Max #Bytes/slot	Approx time
0	15.36 ms	30	~15 ms
1	30.72 ms	60	~30 ms
2	61.44 ms	120	~60 ms
3	122.88 ms	240	~120 ms
4	245.76 ms	480	~250 ms
5	491.54 ms	960	~ ½ s
6	983.04 ms	1920	~ 1 s
7	1.97s	3840	~ 2 s
8	3.93s	7680	~ 4 s
9	7.86s	15360	~ 8 s
10	15.73s	30720	~ 16 s
11	31.46s	61440	~ ½ min
12	62.91s	122880	~ 1 min
13	125.83s	245760	~ 2 min
14	251.66s	491520	~ 4 min
15	No beacons		

Figure 2.7: IEEE 802.15.4 possible values and durations for SO(BO)/SD(BI)

All the possible durations for SD/BI are presented in Fig. 2.7.

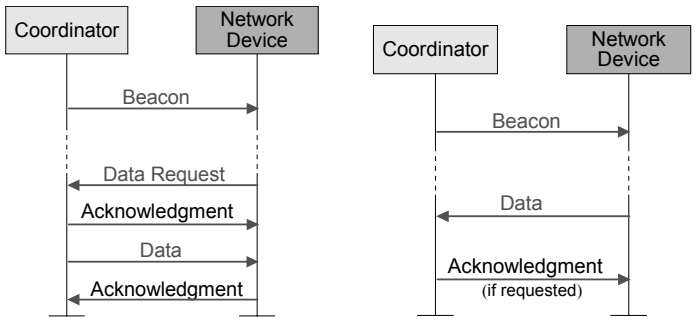
Data Communication. When a node needs to send a packet to its coordinator it will do it after receiving the beacon by using the CSMA/CA access method (Fig. 2.8b).

A characteristic of this operating mode is the mechanism to send data packets from a coordinator to another node. A device is notified in the beacon if there is a pending frame with its address. If applicable, the device will send a data request packet (Fig. 2.8a). The coordinator replies with the data packet and can also give information whether there are still pending frames by setting the frame pending bit in the frame control field.

This mechanism allows long sleeping periods on RFD devices without missing packets. The solution allows working with duty-cycles as low as 0.01%, like waking up every 4 minutes for less than 15 ms which opens the possibility to harvested and autonomous nodes.

2.1.2 ContikiMAC

ContikiMAC [35] which operates like X-MAC [36] is a radio duty cycling protocol that uses periodical wake-ups to listen for packet transmissions from neighbors. If there is a packet transmission detected during the wake-up, the receiver stays on to actually receive the packet. A link layer acknowledgement is sent if the packet is successfully received. A packet that needs to be transmitted is repeatedly sent until the link layer acknowledgement is received.



(a) Downward traffic **(b)** Upward traffic

Figure 2.8: IEEE 802.15.4 communication in beacon-enabled mode

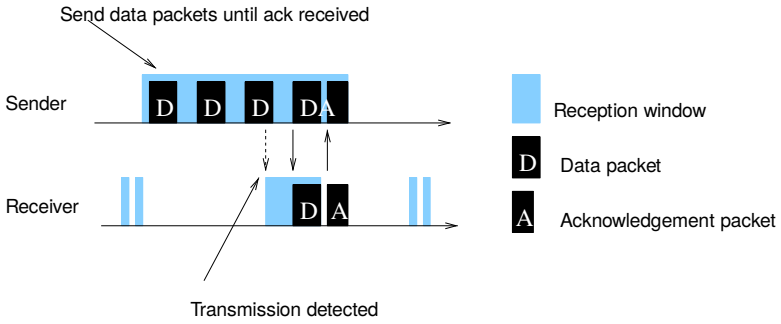


Figure 2.9: ContikiMAC: nodes wake up periodically to check for radio activity

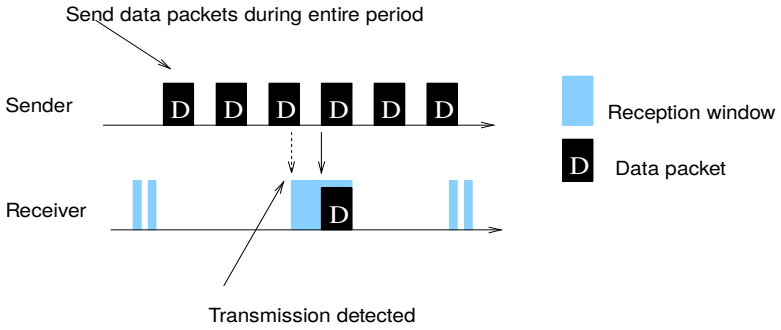


Figure 2.10: ContikiMAC: broadcast transmission during the entire wake-up interval

Broadcasting is costly in terms of energy because the frames are not acknowledged, which means the sender repeatedly sends the packet during the full wake-up interval to ensure that all neighbors receive it (Fig. 2.9 and 2.10). The main issue is that a node does not know when all its neighbors will wake up. An optimization proposed by the Contiki developers is the so called "phase-lock" mechanism where the sender learns the wake-up of its neighbors based on the time the acknowledgement is received (Fig. 2.11). However, this synchronization is kept only if there is an active traffic between two nodes.

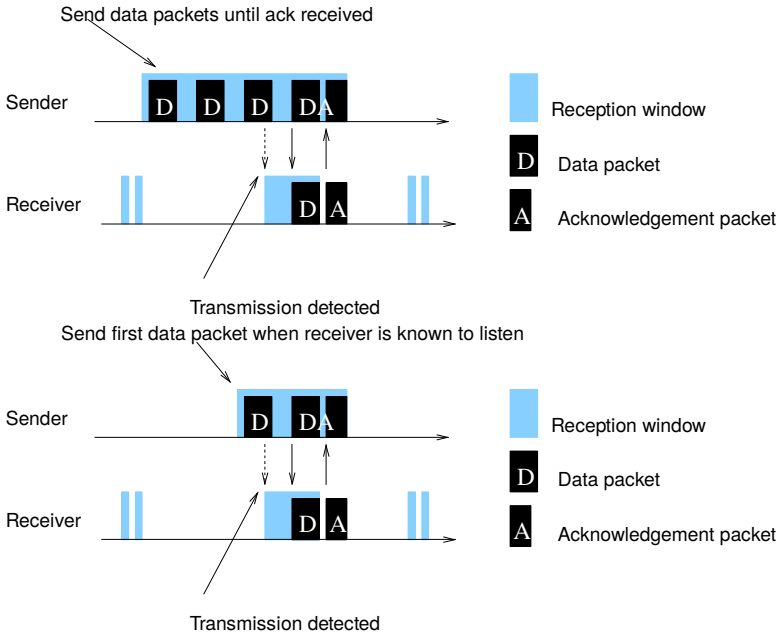


Figure 2.11: ContikiMAC: transmission phase-lock: after a successful transmission, the sender has learned the wake-up phase of the receiver, and subsequently needs to send fewer transmissions.

2.2 Channel Hopping protocols for IEEE 802.15.4

2.2.1 Deterministic Synchronous Multichannel Extension

Deterministic Synchronous Multichannel Extension (DSME) is an extension of 802.15.4 [37] beacon-enabled mode with similar features like beacons for synchronization and the basic superframe structure. It also uses the two defined types of devices, the Full-Function Device (FFD) and Reduced-Function Device (RFD). DSME extends the superframe structure with the so-called multi-superframe (cf. Fig. 2.12).

$$\mathrm{SD} = aBaseSuperFrameDuration * 2^{\mathrm{SO}}$$

$2^{(MO-SO)}$ superframes in a multi superframe
$2^{(BO-MO)}$ multi-superframes in a beacon interval
$2^{(BO-SO)}$ superframes in a beacon interval

In DSME, 7 slots are reserved for Guaranteed Time Slots (GTS) and 9 slots for Contention Access Period (CAP). Enhanced Beacons are sent at the beginning of each super-frame containing Information Elements to describe various properties of the network. CAP Reduction is used in DSME to only use the first superframe in a multi-superframe as a CAP period. The remaining of superframes are available for GTS allocation, allowing greater energy savings.

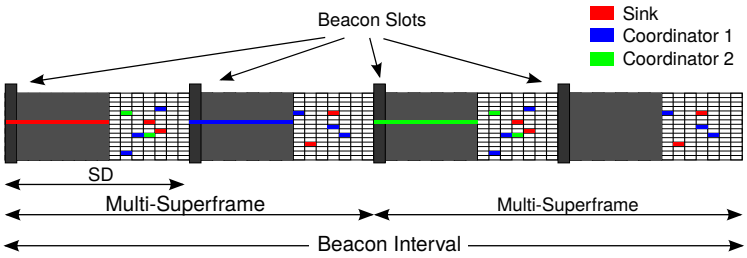


Figure 2.12: DSME: multi-superframes example

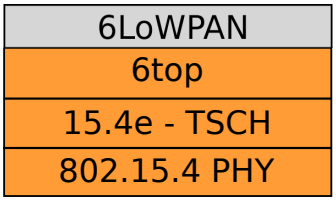


Figure 2.13: TSCH IPv6 Stack

2.2.2 Time Slot Channel Hopping

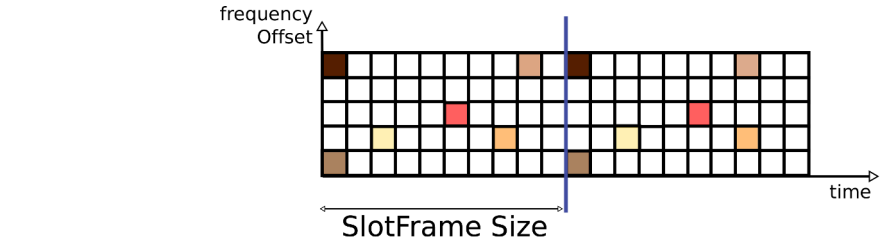
Time Slot Channel Hopping (TSCH) Mode is part of the 802.15.4-2012 amendment [37] that includes the well-tested TSMP technology [38] inside 802.15.4 networks. Not only brings it channel hopping to 15.4 networks, but also it makes network fully deterministic thanks to a shared-schedule among nodes. Between the MAC and 6LoWPAN layer (Fig. 2.13), IETF 6tisch draft proposes a control layer for TSCH, called 6top [39] that offers both management and data interfaces. The main functionalities are feedback metrics for routing decisions, TSCH configuration and control procedures, and support for slots scheduling policies.

TSCH PHY Layer. 802.15.4-2006 PHY layer does not change with the new amendment, except that nodes can use frequency hopping to be more robust against interference. The frequency hopping algorithm is time based. At the beginning of any slot, a node that needs to transmit computes the frequency it should use, based on the formula:

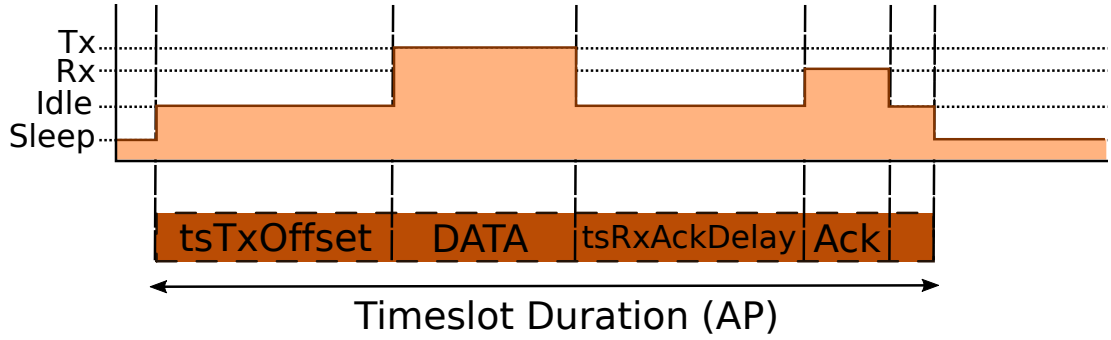
$$f = ((ChannelOffset + ASN) \% Nb\ of\ Channels)$$

where *ChannelOffset* can be considered as a logical channel and *Absolute Slot Number* (ASN) is the total number of timeslots since the network bootstrap.

TSCH MAC Layer. The changes brought by 2012 version [37] concern the MAC layer: data transmission scheme is replaced by a deterministic schedule. Fig. 2.14a shows an example of schedule, represented by a slotframe: a time-frequency representation of the different frequency channels at different timeslots. Hence, TSCH is a hybrid FDMA/TDMA mechanism.



(a) TSCH scheduling example



(b) TSCH timeslot organisation

Figure 2.14: TSCH principles

A colored square represents a link, a *(time, frequency)* couple completed by addresses of the communicating nodes during that slot. Three types of link can be identified in a TSCH network: *advertising link*, used to transmit advertising data, *shared link* which can be used by any node that needs CSMA/CA access scheme, and *dedicated links*, that are allocated resources. Fig. 2.14b shows the operation of a node within a slot. The way the scheduling is built in a TSCH network is not defined in the new revision of the standard, but some proposal of centralized and decentralized scheduling can be found in [40], [41] and [42].

Joining phase. The TSCH joining phase is similar to 802.15.4 beacon-enabled association mechanism. The main difference lies in different frame formats: Enhanced Beacons (EB) contain classical beacon information plus several necessary Information Elements (IE), such as slotframe IE. Contrary to classical 802.15.4, EBs are only used during the joining phase. Hence the transmission happens less often than classical beacons which leads to a longer joining interval.

Synchronization. TSCH networks do not run under a specific frame to keep synchronized. Among TSCH network, synchronization can be done in 3 different ways:

- Enhanced beacon listening: TSCH amendment no longer uses a superframe structure. However, it defines Enhanced beacon frames that are used within *advertising* slots to distribute the scheduling and the absolute slotframe number (ASN) among the network. This type of synchronization is only used while trying to join the network.

- Frame-based: TSCH ensures that a data frame leaves the transmitter **exactly** $tsTxOffset$ (fixed-value) after the beginning of the timeslot, from the sender point of view. Since this offset is common to all nodes, while decoding a data frame, the receiver knows $tsTxOffset$ date from the sender point-of-view. It can then compute the difference Δ between its own expected $tsTxOffset$ date. If the sender is known as a time-source neighbor by the receiver, the receiver then adjusts its own time by Δ to be synchronized to the network. The MAC standard does not define yet how and when the time-source neighbors are chosen.

- ACK-based: the principle is identical to the one in frame-based synchronization, except that there is a timestamp in the ACK packet: when the ACK sender starts writing a Start of Frame over the air, it captures the time and adds that timestamp to the ACK payload. Then, the same mechanism is used to keep synchronized.

Since TSCH does not use any specific synchronization frame but exploits data packet transmissions, it is hard to keep synchronization when the network is idle, *i.e.* when no data packets are transmitted. In order to solve that issue, the standard uses some keep-alive messages (no data packet). These packets are sent following an adaptive scheme that takes into account the amount of traffic between nodes, hence, no keep-alives are sent when data packets are flowing through the network.

2.3 Other Wireless IoT Technologies

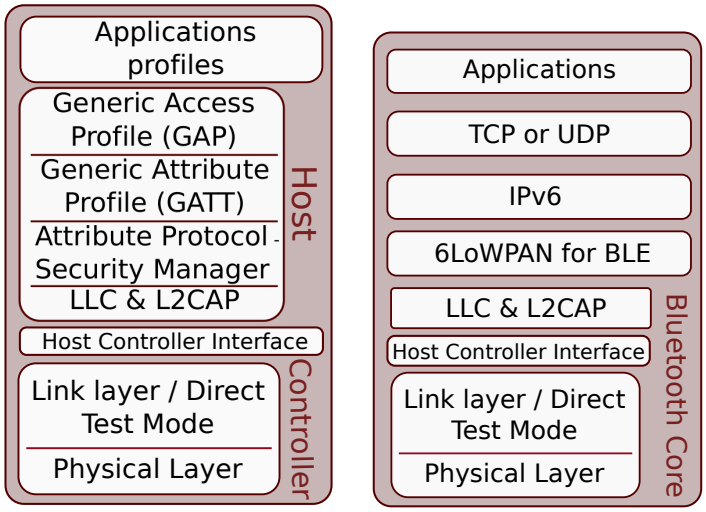
2.3.1 Bluetooth Low Energy

Another technology that is becoming more and more available especially due to its high availability of its previous versions is Bluetooth Low Energy [43].

BLE is a low energy version of Bluetooth standard that aims at reducing energy consumption and thus allowing users to build IoT with Bluetooth devices. The classical Bluetooth stack is presented on the left part of Fig. 2.15, whereas a hybrid IPv6/BLE stack is presented on the right. This future IP-stack will be enable by the *6lo* IETF working group [44]. In this context, BLE is a WSN-specific Bluetooth *controller*. The first version of this controller has been presented in [45], and further improved in [46] in December 2014.

BLE PHY Layer. BLE controllers use 2.4 GHz ISM frequency band, split in 40 channels of 2 MHz wide, 2 Mhz apart from each other. Three channels are dedicated to be advertisement channels, and the remaining 37 channels are for data transfer only. The modulation scheme for BLE is GFSK, which allows to reach a 1 Mb/s over-the-air rate. In order to be robust against interference and fading channels, Bluetooth uses frequency hopping spread spectrum. BLE is a master/slave-based network, the topology brought by BLE is a star, or a *piconet*, as it is named in Bluetooth terminology.

Frequency hopping happens either periodically or at some logical time, depending on the frequency hopping mode. The mechanism is based on a frequency hopping pattern that may be automatically adapted to fit the environment.



(a) Classical **(b)** IP-based

Figure 2.15: BLE Stacks

BLE MAC Layer. BLE motes can operate in four different modes: advertising, scanning, slave and master. The two first forms are used in the joining phase whereas the latter ones are reached once a link is established between two nodes, i.e. one node can communicate through connection events. In these events, the master and the slave wake up simultaneously to exchange frames. Connections intervals repeat periodically over time and are maintained until one of the node wants to close the link, or the maximum number of *Connection Interval* without traffic (*Slave Latency*) is reached. Hence, these parameters are defining the duty cycle of a node.

Joining phase. BLE joining mechanism is initiated by the incoming node, which is in advertising mode: it advertises on one or several of the three dedicated advertising channels. Nodes in scanning mode should be scanning these channels in order to discover soon-to-be slaves. Once a master hears an advertising packet, it answers with a connection request, containing the information for the connection events that will occur between our nodes: *Connection Interval*, *Slave Latency*, *Transmit Windows offset*, and the frequency data channel they will use for the first connection event, as it is shown on Fig. 2.16. Thus, both nodes switch to the data channel (dark red) and change their modes: the scanning node becomes a master while the advertising node becomes a slave.

Data exchange. The mechanism of data transmission can be easily understood by taking a look at Fig. 2.16: at the beginning of each Connection Interval, the slave wakes up and waits for a poll from its master before sending its packet in order to avoid collision with another slave for the same master. The wake up mechanism is not compulsory if the slave does not need to transmit any data, except if the maximum number of Connection Interval without data traffic (Slave Latency) has already been

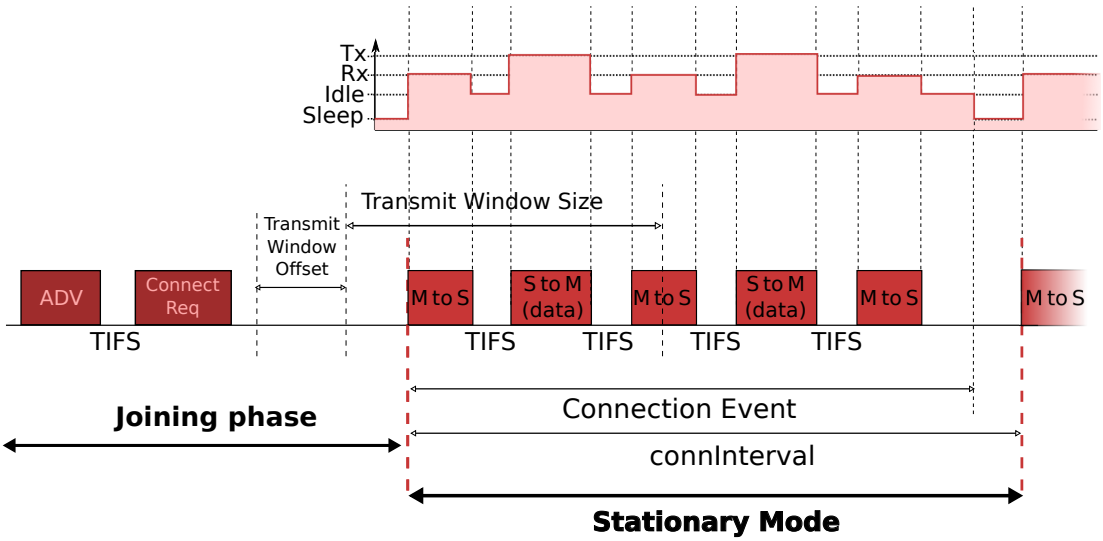


Figure 2.16: BLE maximum efficiency

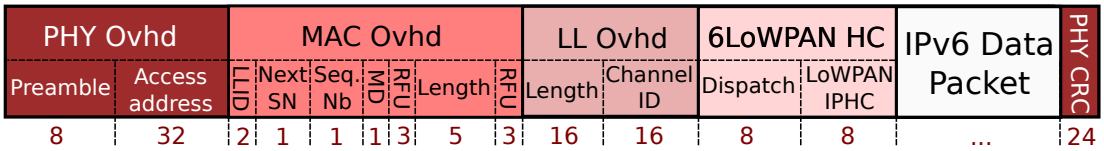


Figure 2.17: BLE Minimal Data Packet Format (size in bits)

reached by the slave.

The overhead in a BLE data packet can be found in Fig. 2.17. We assumed a Link Local 6LoWPAN fragment, since BLE is master-slave directed.

Fig. 2.16 gives the overview of the maximum efficiency we can get when using Bluetooth Low Energy: when no one else needs to dialog with the master of the connection, then the slave can transmit data packet whenever it needs. If the slave needs to send more than one data packet, the maximum throughput will be reached if the slave can schedule all these packets in a row.

Differences between 4.2 and 4.0 BLE core: Bluetooth's 4.0 main drawbacks are mainly that it is restricted to a star topology and the small maximum packet size: 21 bytes once removed all the overhead presented in Fig. 2.17. Hence, version 4.2 of the specification offers solution to these issues. It specifies that, "*slaves are permitted to have physical links to more than one master at a time and a device is permitted to be master and slave at the same time*" [46], which allows layer 2 mesh topology. The limits that still prevent BLE to be full-mesh (layer 3) are:

- no possibility of direct link between slave devices.
- no mesh-allowing routing layer yet.

Besides, the new specification changes the new maximum data packet size to 251 bytes and brought some new security features.

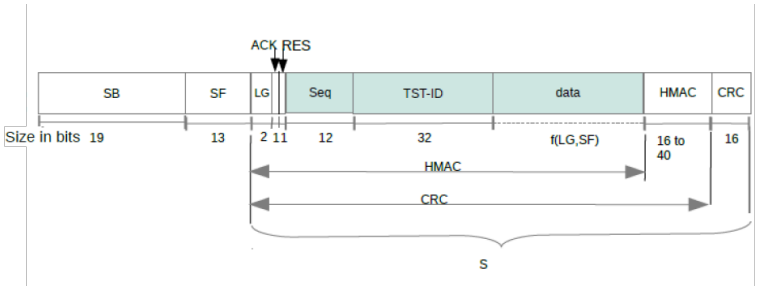


Figure 2.18: Sigfox uplink datagram format

2.3.2 Long Range Solutions

3.2.1 SIGFOX

Global overview. Sigfox [47] presents itself as a "Global cellular connectivity for the Internet of Things". Sigfox's principle indeed lies in the cellular aspect of the technology, which makes it a connectivity provider in the same way as a mobile network provider: Sigfox provides customers with an infrastructure to link their devices to.

Sigfox PHY Layer transmits data within 868 Mhz frequency band in Europe and 903 MhZ in US, and is an Ultra Narrow Band technology. It is based on DBPSK and GFSK modulations, which allow to reach PHY rates of 100 bps in Europe and 600 bps in US. Whereas the available rates seem ridiculously small compared to other IoT technologies, the receiver sensibility fixed to -140 dBm allow Sigfox to achieve extraordinary ranges: it can actually reach up to 40 km in open space area. Hence, the application context of Sigfox differs from classical IoT solutions.

Regarding the security aspect, Sigfox networks do not understand or parse user messages, which allow users to cypher their data if needed. Moreover, MAC layer allows device authentication through the HMAC field, that contains device ID.

Up-link traffic. Sigfox technology focuses on extremely low throughput devices: Sigfox devices can only transmit up to 140 messages per day to their base station (up-link). The useful data payload can be encapsulated in 5 different container sizes: 1 byte, 4 bytes, 8 bytes and 12 bytes. This leads to a maximum data of 1680 bytes a day, which should cover most of the needs for devices that transmit data such as location of a device, energy consumption index, alarm, or any other type of basic sensing information.

Sigfox MAC layer is simple since it is not a synchronized network: a device that wants to transmit data encapsulates it within a packet as shown on Figure 2.18; this up-link packet is then transmitted three times on different random frequencies with channel encoding, in order to have good chances the base station receives it properly. This three-times transmission mechanism was originally a real need since no downlink traffic was possible, so the transmission could not be acknowledged. Hence, to make sure the base station receives a packet, end-devices guard themselves from bad channel conditions by transmitting 3 copies of the same packet, with different encoding each time.

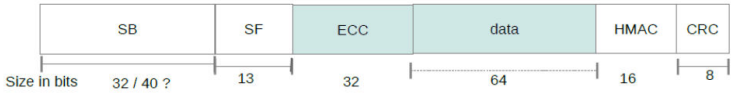


Figure 2.19: Sigfox downlink datagram format

Downlink traffic. Recently, Sigfox introduced the possibility of downlink transmissions so one can also transmit a maximum of 4 messages of 8 bytes payload to each device per day. These 8 bytes messages allow to send configuration data to end-devices if needed, but one can optimize battery life by only being one-way if there is no need for two-way communications. Since Sigfox's devices do not synchronize with the network, a device has to poll the base station for data in order to receive downlink messages. Base station then transmits the downlink packet on the same shift of frequency as the request packet in the downlink band: assuming the poll packet at a frequency of 10kHz below the central uplink frequency, the downlink packet will then be sent on the frequency of 10kHz below the central downlink frequency. Hence, no prior communication is needed to select downlink channels. Downlink packet format is displayed in Figure 2.19.

3.2.2 LoRaWAN

LoRaWAN [48] is a Low Power Wide Area Network (LPWAN) specification intended for wireless battery operated objects.

LoRaWAN can either use LoRa (Long Range) modulation technique or the GFSK modulation. With LoRa modulation, the data rate can range between 250 bytes/s to 11kbps, while in GFSK mode it can be 50kbps. The choice of data rate for each particular device is decided by the LoRaWAN network server, based on the link quality. This means that a device that is close to a gateway can have a higher data rate while reducing the necessary emission power. On the other hand, for devices that are far away from the towers, higher emission power and lower data rates will be used to ensure a reliable transmission. This mechanism is called Adaptive Data Rate and its goal is to maximize battery life and overall network capacity.

The architecture of a typical LoRaWAN network is a star-of-stars topology where gateways are connected with end devices using LoRa modulation. All the gateways are then connected either through standard Ethernet network or GSM with a central server (Fig.2.20).

Devices communicate with gateways on different frequency channels and data rates. A LoRaWAN device must at least implement the Class A, bi-directional end-device. These devices are mostly designed for upward traffic and the only downward traffic that is done is shortly after the scheduled uplink. In case there is a packet pending for a Class A device it will have to wait the next scheduled uplink. The other two classes (B and C) have more receiving slots to enable downward traffic. Of course they consume more energy as they need to synchronize with the gateway.

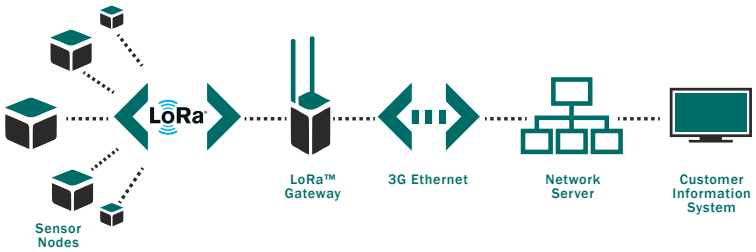


Figure 2.20: LoRa topology [2]

2.4 Energy per bit comparison between the different technologies

Table 2.1: Energy per bit consumption

	15.4	**BLE**	**LoRa**	**Sigfox**	**802.11ah**	**802.11b**
Rate (kbps)	250	1024	0.25	0.1	150	$11 * 10^3$
TX (dBm)	0	0	20	14	14	18
Eb (nj)	4	1	400000	25119	167	5.7

Table 2.1 presents the theoretical energy per bit (Eb) of the above presented technologies and Wi-Fi. The metric is directly correlated to the emission power (dBm) and data rate (time on air needed to transmit a certain amount of data). This does not take into account the consumption of radio chips or MCUs. For LoRa we considered the minimum data rate and highest transmission power, that offers also the longest communication distance so the highest energy consumption.

Fig. 2.21 shows the above presented technologies from the point of view of range and data rate. We should keep in mind that consumption of each technology is also closely related to communication range.

A network or a project can have its own particularities or characteristics so the choice of the technology to use should be done having these aspects in mind.

2.5 Conclusions

This chapter presented the important radio technologies (close range and long range) that are meant for IoT. It is important to keep in mind that each of them have their advantages and disadvantages. If some application designers needs to cover distances of tens of kilometers, they will most probably choose a long range solution whereas for indoor usage or office buildings, an IEEE 802.15.4 deployment is suitable.

IEEE 802.15.4 protocol is versatile: it can work very well with alternative duty-cycling (like ContikiMAC), allows functioning of different type of devices (RFD, FFD) depending on the amount of energy they have and it also offers solution for very low duty-cycling when using the beacon-enabled mode. Because of all these reasons and especially of the energy saving potential,GreenNet project uses the IEEE 802.15.4 beacon-enabled mode [30].

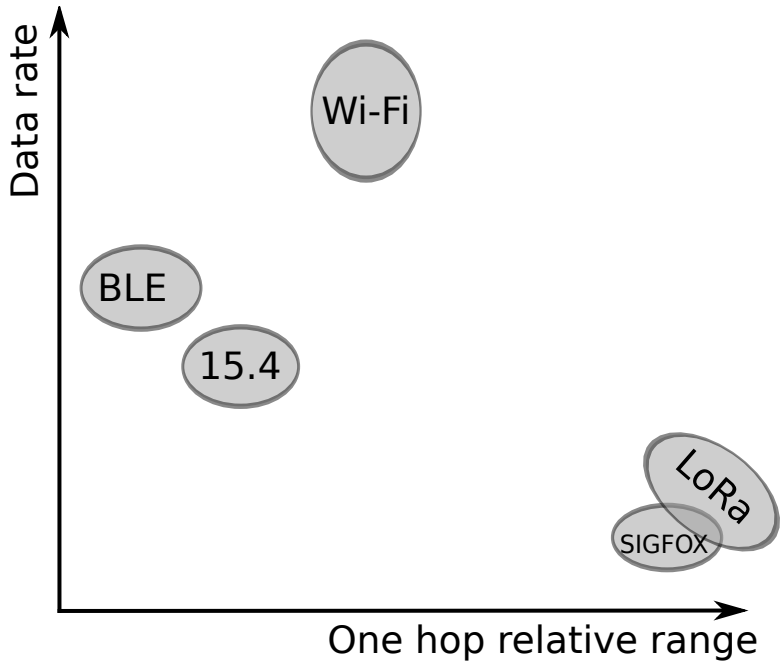


Figure 2.21: IoT radio technologies

Three recent initiatives show an increased industrial interest in protocol stacks for the Internet of Things: Thread, AllSeen Alliance and Open Interconnect Consortium. The Thread stack builds up on the IEEE 802.15.4 MAC/PHY and standard 6LoWPAN adaptation [49]. It provides a streamlined IP routing protocol and supports security through an authentication scheme and AES encryption. AllJoyn by the AllSeen Alliance is an open source framework and set of services that enable interoperability among connected products and software applications to create dynamic proximal networks [50]. Open Interconnect Consortium defines an open specification for interoperability across connected devices [51]. It considers that secure and reliable device discovery and connectivity are a foundational capability to enable IoT.

In the next chapter we present some of the hardware platforms used in industry and research for wireless sensor networks.

Chapter 3

Hardware

Contents

This chapter presents the most relevant and most efficient 802.15.4-compliant radio chips and CPUs available on the market as well as a few platforms that integrate them. The choice of hardware components can make a big difference regarding the autonomy of a mote, especially if it is designed to run on harvested energy.

3.1 Various Radio Chips and CPUs for Sensor Networks

One of the best placed chips with respect to energy consumption is the RF200W chip from STMicroelectronics that features a 32-bit Cortex-M3 ARM microcontroller and an IEEE 802.15.4-compliant radio with a current draw at 3.6V of 4.5mA in RX mode and 4.9 mA in TX mode (0dBm).

Smart Mesh IP is the latest protocol stack from Linear Technology Dust Networks group [52] that runs on the LTC5800 chip. It features a 32-bit Cortex-M3 ARM microcontroller and an IEEE 802.15.4-compliant radio with a current draw at 3.6V of 4.5mA in RX mode and 5.4 mA in TX mode (0dBm). The reported average current consumption varies between 17.7μA for a 4-hops deep leaf mote and 34.2μA for a 1-hop mote when all motes send 80 bytes of application payload every 30 seconds. The stack builds on the IEEE 802.15.4e TSCH standard and offers IPv6 connectivity over 6LoWPAN. A centralized manager builds a TSCH schedule that supports IP forwarding.

ZigBee PRO GreenPower [53] offers an asynchronous solution in which devices consume very little, but require always-on routers. Moreover, they are not IP-enabled and the sink cannot initiate a data exchange with a Green Power Device, because it does not know the instant at which the device will be on.

Table 3.1 compares the GreenNet platform with currently available commercial solutions. The values are taken from the official data-sheets of each product. All of them are IP-enabled and offer the possibility to attach different sensors in addition to the integrated ones. What singles out GreenNet is the fact that it can autonomously

Table 3.1: Commercially available motes

Mote	#bits	RAM [kB]	CPU ON [mA/ MHz]	CPU sleep [μA]	TX 0dBm [mA]	RX [mA]	Har- vested	Batt. size/type [mAh]
GreenNet	32	32	0.185	0.44	4.9	4.5	Y	25
Hikob [54]	32	16	0.180	0.6	13.8	11.8	Y	2000
SmartMeshIP [52]	32	72	0.176	0.8	5.4	4.5	OPT	2AA
M3OpenNode [55]	32	64	1.138	25	11.6	10.3	N	650
OpenMote [56]	32	32	0.438	0.4	24	20	N	2AAA
WisMote [57]	16	16	0.312	1.69	25.8	18.5	N	2AAA
TelosB [58]	16	10	1.8	5.1	19.5	21.8	N	2AA
Waspmote15.4 [59]	8	8	1.07	7.5	45	50	OPT	N/A
MICAz [60]	8	4	1.0	<15	17.4	19.7	N	2AA

run on the energy from an integrated solar panel and offers a protocol stack optimized for reactive operation allowing for very long sleep periods.

We can see that taking into account only the radio and CPU can add up from 10mA to 50mA. On one hand, platforms that are not energy efficient are more dependent on external power supplies (like larger batteries). On the other hand we can consider that very low energy solutions can work with a small rechargeable battery and a form of energy harvesting.

3.2 Harvesting

To reach autonomous solution, a radio node can either be connected to a power supply or be a harvested node. Among the different and most common possibilities to harvest energy the most used ones are the light, mechanical movements and temperature difference.

3.2.1 Energy harvested from ambient light

One of the most present forms of energy is light. This can be either in the form of solar light or artificial light (indoor). Harvesting this source of energy is done through photo-voltaic cells. A panel of 5cmx4cm can harvest sufficient energy to send a packet every few minutes with a light intensity of 100lux (the light intensity that is usually present at desk level in a typical office). For home applications this is one of the most accessible form of energy that can be harvested.

Several authors suggest algorithms for duty-cycling in harvesting environment based on the prediction of light availability and battery level.

Kansal et al. [61] proposed an energy prediction model based on an Exponentially Weighted Moving-Average filter. They considered that the energy available at a given

time of a day is close to that available during the previous days, which is only partially true for outdoor conditions. GreenNet aims at a different environment, namely indoor under artificial light, where the conditions can change more often, like switching on/off different sources of light. Vigorito et al. [62] proposed a model-free approach to adapt the duty cycle. Their solution uses an objective function based on the battery level. It is difficult to implement such a solution, because the exact value of the battery level is hard to obtain on the GreenNet platform and also on many other platforms. Nevertheless, the battery level, even if it is an approximate value, can serve as complementary information [63].

In [64] the author proposes a sustainable algorithm to adapt the node activity according to the available energy and traffic conditions (STADA), that takes into account the energy present in the battery, the energy harvesting rate and network traffic. They also propose a metric that takes into account the light variations and other parameters to facilitate the path choice in multi-hop harvesting wireless sensor networks.

3.2.2 Mechanical

Grasping energy from the environment can moreover be realised through mechanical ways: harvesting the wind's force, or energy coming from vibrations.

For networks that are deployed outdoor, taking advantage of wind can be an alternative source to complement light, but it is not at all predictable, and the hardware complexity is higher. Moreover, the size of the platform is affected, as a wind turbine occupies more space than a solar panel. The efficiency can be high if enough wind is present but in a completely unpredictable way.

Other solutions to harvest energy from mechanical movements can be to harvest a small amount of energy when a user pushes a button to send an event. This can make a node work without batteries and be turned on only when the button is pushed and in the rest of the time be asleep, that is suitable for RFD devices.

3.2.3 Thermal

Besides, the differences in temperature between two surfaces can be used to generate electricity. This is especially found for body area networks, where the body temperature can be a source of energy. Another use case that can take advantage of the temperature difference are windows, where usually there is always at least a few Celsius degrees of difference.

Conclusion. Having reviewed the three most important forms of harvesting, the one that can offer enough energy, in a predictable manner as well as a reduced dimension of the system, is the photovoltaic cell. As a consequence, the GreenNet project is built using a small photovoltaic cell as a source for recharging the battery.

3.3 GreenNet hardware platform

The main objective of GreenNet is to obtain nodes that consume very little energy in order to be able to work for very long periods without any maintenance,

Figure 3.1: GreenNet node

such as changing the battery. In order to achieve this goal, *i.e.* autonomous harvested nodes, duty-cycles as low as 0.01% are often required.

A GreenNet node (Fig. 3.1) is based on the energy efficient STM32L1 microcontroller along with a new generation radio that supports the IEEE 802.15.4 beacon-enabled mode (Fig. 3.2). This radio requires precise timers to wake up at the right instant and remain active only as much as necessary. A Power Management Unit (PMU) [65] handles the charging of a rechargeable coin battery (nominal voltage of 3V) with the energy harvested by the photovoltaic cell or with the current from the USB connection (when it is connected to a power source). The microcontroller also embeds a Low Drop-Out (LDO) that regulates the internal voltage to 1.2V.

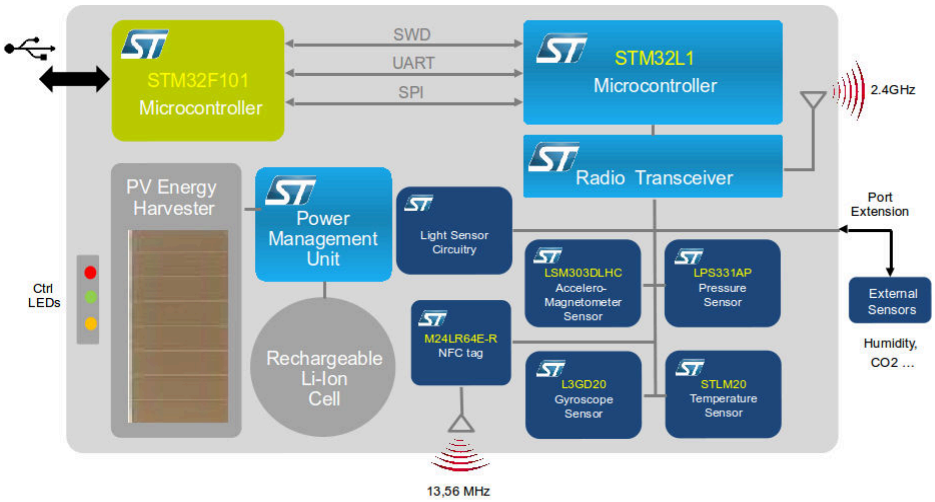


Figure 3.2: GreenNet node schematics

As stated in previous sections, the radio consumes 4.5mA in RX mode and 4.9mA in TX mode (0dBm). The MCU with an activated sensor consumes less than 5mA. The PMU consumes less than 5% of the harvested energy. In the Low Power Mode (sleep state), the entire board (radio + MCU + PMU) consumes less than 2.3μA [66].

The photovoltaic cell provides 15μA at 100lux with a quasi linear gain when light is provided by a fluorescent lamp.

The design choices of the system regarding the size of the photovoltaic cell, battery capacity and power management is aimed at the use case of an autonomous operation while sending a packet every 4 minutes with 100 lux of light during 8 hours/day. To satisfy these constraints a battery capacity of 20mAh (17 mAh of useful capacity) is sufficient.

3.4 Conclusions

This chapter presented the latest hardware platforms and their performances in terms of energy consumption. It must be said that another important aspect of a mote is the software that runs on it, the operating system and all the communication protocols.

Through this section, we mentioned what possibilities we have in terms of energy harvesting and we ended with a more detailed presentation of GreenNet platform.

As mentionned before, a hardware platform is a good starting point towards IoT. However, reaching energy autonomy needs to minimize energy consumption at every layer. Hence, most of the contributions of the thesis will focus on this aspect. The following chapter presents operating systems specially designed for the Internet of Things motes.

Chapter 4

Operating Systems

Contents

A good integrated solution needs to consider wisely taken decision for both the hardware platform and the OS (which can also include a protocol stack).

The mostly used OS among IoT community is Contiki but other systems are becoming interesting, like RIOT and OpenWSN project.

The following sections present the operating systems that are the most active in the community of IoT.

4.1 Contiki

Contiki [67] is an open source operating system, written in C, designed especially for the Internet of Things. It allows connection of tiny low-cost, low-power microcontrollers to Internet. Contiki first explored IPv4 communications for sensor networks [15] and currently fully supports IPv6 [68].

Contiki also supports the standards 6LoWPAN and RPL [69]. Duty-cycled or sleepy routers can be deployed by using the integrated ContikiMAC protocol.

The development of applications is done in C. With the Cooja simulator, networks can be emulated before actually flashing the hardware (as long as the hardware is supported by the emulator). Contiki can run on systems that only have a few kilobytes of memory available. A full IP network stack is already implemented including UDP, TCP and HTTP.

A characteristic of Contiki is the mechanism called *protothreads*. A protothread is a mixture of event-driven and multi-threaded programming mechanism. The implementation of protothreads is based on macros that use switch/case functions. Consequently, event-handlers block until the required event is triggered. The main advantage of protothreads is that they are light in terms of memory consumption and they do not require their own execution stack. Contiki also offers real time clocks that are based on hardware timers. These are used especially for synchronization purposes and for protocol timings.

Contiki also has the ability to load and unload individual applications or services at run-time. A running Contiki system (Fig. 4.1) consists of a kernel, libraries, the program loader and a set of processes. A service implements functionality used by more than one application process. Communication between processes goes through the kernel. A Contiki system is partitioned into two parts: the core and the loaded programs. Typically, the core consists of the Contiki kernel, the program loader and a communication stack with device drivers for the communication hardware. Programs are loaded into the system by the program loader.

The Contiki kernel supports two kinds of events: asynchronous and synchronous events. In addition to events, the kernel provides a polling mechanism. A poll request is a special type of event that causes a process to be scheduled as quickly as possible. Polling is the way to make a process run from an interrupt. The kernel consists in a lightweight event scheduler that dispatches events to running processes and periodically calls the processes' polling handlers. An event handler is not preempted, therefore it must run to completion. Contiki uses a single shared stack for all process execution.

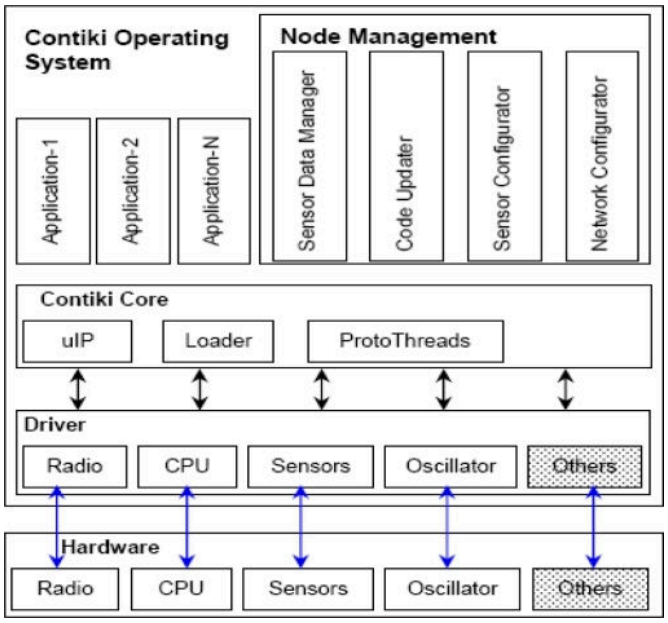


Figure 4.1: Contiki architecture

4.2 RIOT

RIOT OS [70] aims at bridging the gap between OS for WSNs and traditional full-fledged OS, currently running on Internet hosts. RIOT implements a microkernel inherited from FireKernel, thus supporting multi-threading with standard API. RIOT adds support for C++ and provides a TCP/IP network stack (Fig. 4.2). RIOT allows developers to create as many threads as needed, the only limitation is the available memory.

To be a real-time OS, RIOT has constant periods for kernel tasks (e.g. scheduler

run, inter-process communication, timer operations). It exclusively uses static memory allocation in the kernel. To allow long sleep modes and energy efficiency, RIOT introduces a scheduler that works without any periodic events. In the idle thread RIOT will determine the deepest possible sleep mode depending on peripheral devices in use and will only wake up from the idle state on an interrupt (external or kernel generated).

Available open source RIOT code requires less than 5 kBytes of ROM and less than 1.5 kBytes of RAM for a basic application on MSP430.

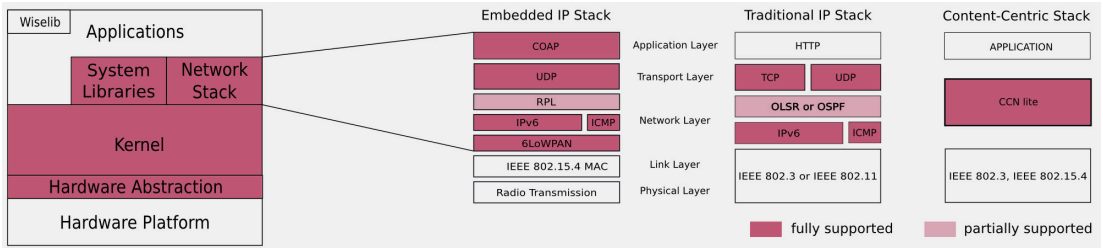


Figure 4.2: RIOT architecture

4.3 OpenOS

The OpenWSN [71] project is part of an ecosystem of commercial products and open-source projects closely related to the Internet of Things. OpenWSN is an open-source implementation of a fully standard-based protocol stack based on the new IEEE 802.15.4e TSCH. OpenWSN stack uses two different abstractions (Fig. 4.3). The first one is the Berkeley Socket Abstraction, as part of the Berkeley Software Distribution operating system development, that considers that applications communicate through a socket which is uniquely identified by the IP addresses of the hosts. The second one is the Hardware Abstraction, and consists in grouping all functions accessing the hardware into a group of files called the 'board support package' (BSP). The scheduler of the OpenWSN is called OpenOS and is based on hardware and timer interrupts to push tasks in a list, based on their priority. OpenOS is non-preemptive, that is, tasks do no interrupt one another.

OpenWSN integrates IoT standards such as 6LoWPAN, RPL and CoAP that can enable ultra-low-power and highly reliable mesh networks, fully integrated into the Internet. Like Contiki, OpenWSN offers a simulation environment, called OpenVisualizer, which is a Python-based debugging and visualization program that runs on PC and interacts with the OpenWSN motes connected to it.

4.4 TinyOS

TinyOS [72] is a free and open source software component-based operating system and platform targeting wireless sensor networks (WSNs). TinyOS is an embedded operating system written in the nesC programming language as a set of cooperating tasks and processes. It is intended to be incorporated into smartdust [73]. TinyOS started as a collaboration between the University of California - Berkeley, Intel Research

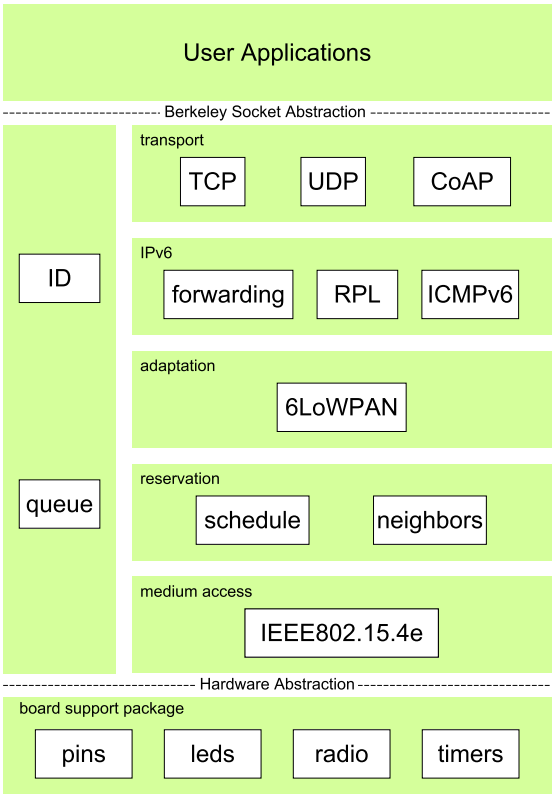


Figure 4.3: OpenWSN architecture

and Crossbow Technology, and has since grown to be an international consortium, the TinyOS Alliance.

TinyOS programming language, nesC, is a dialect of the C language optimized for the memory limits of sensor networks. Its supplementary tools are mainly in the form of Java and shell script front-ends. Associated libraries and tools, such as the nesC compiler and Atmel AVR binutils toolchains, are mostly written in C.

TinyOS programs are built out of software components, some of which present hardware abstractions. Components are connected to each other using interfaces. TinyOS provides interfaces and components for common abstractions such as packet communication, routing, sensing, actuation and storage.

TinyOS is completely non-blocking: it has one execution stack. Therefore, all I/O operations that last longer than a few hundred microseconds are asynchronous and have a callback. To enable the native compiler to better optimize across call boundaries, TinyOS uses nesC's features to link these callbacks, called events, statically. While being non-blocking, TinyOS enables to maintain high concurrency with one stack, it forces programmers to write complex logic by stitching together many small event handlers. A TinyOS component can post a task, which the OS will schedule to run later. Tasks are non-preemptive and run in FIFO order. This simple concurrency model is typically sufficient for I/O centric applications, but its difficulty with CPU-heavy applications has led to the development of a thread library for the OS, named TOSThreads.

TinyOS code is statically linked with program code and is compiled into a small binary, using a custom GNU toolchain. Associated utilities are provided to complete a development platform to work with TinyOS.

4.5 Conclusions

From this list of operating systems for IoT, it is to the developer to choose the most suitable one, given the requirements of a particular project. RIOT and OpenWSN are promising solutions but with a rather small community for now, whereas ContikiOS has a much larger group of users. Nevertheless, one would choose an OS based on what it can already provide out of the box (eg. duty-cycling protocols, IP features etc.). These last considerations are far more important than a specific characteristic of the operating system as it can drastically reduce the development time needed to put in place a working and efficient network.

In conclusion, we used ContikiOS on GreenNet platform and improved it to our needs. At the application layer, in GreenNet, we use Californium [74], a low-power CoAP [75] implementation for Contiki that supports RESTful Web services on nodes.

During the project we also ported OpenWSN on our platform so we have demonstrated that we can use the same hardware with different software stacks.

Finally, having a suitable OS, the next step of a developer choice would be the communication protocol, as it has a big impact on the performance of the network and quality of service.

Chapter 5

Routing protocols

Contents

For multi-hop networks, a very important layer is the routing. In the context of sensor applications it needs to support specific traffic patterns: first of all, the most important aspect is to forward the gathered data to one or several sinks (MP2P - multipoint-to-point or convergecast). Secondly, the network has to support downward traffic from a sink to all or some sensor nodes. This is needed for the traffic pattern resulting from the CoAP queries or sensor nodes configuration. In the following sections we present RPL and LOADng, two of the most used forms of routing protocols in sensor networks and at the end, RPL-LR which is an improvement of RPL.

5.1 Routing Protocol for Low-power and Lossy Networks

RPL [27] consists in constructing and maintaining a Destination-Oriented Directed Acyclic Graph (DODAG) for upward routes using DODAG Information Object (DIO) messages. An Objective Function (OF) is used to select a node and optimize routes within a DODAG.

For downward routes, RPL uses Destination Advertisement Object (DAO) messages. DAO messages are an optional feature for applications requiring point-to-multipoint (P2MP) or point-to-point (P2P) traffic. RPL supports two modes of Downward traffic: Storing (fully stateful) or Non-Storing (fully source routed). In both cases, P2P packets travel up toward a DODAG root then down to the final destination (unless the destination is on the upward route). In the Non-Storing case, the packet will travel all the way to a DODAG root before traveling down. In Storing mode the DAO message is unicast by the child to the selected parent(s) whereas in the Non-Storing mode it is unicast to the DODAG root. There is an overhead in each packet for non-storing and in the routing table for storing mode.

There is no possibility to request/rebuild a route in RPL except global repair or when the Destination Advertisement Trigger Sequence Number (DTSN) increases, in which case all below nodes resend DAOs.

5.2 LOADng

LOADng [76] is a successor of the Ad hoc On-demand Distance-Vector routing protocol (AODV). AODV [77] is a method of routing messages between mobile computers. Nodes that can be reached directly are considered Neighbors. When one node needs to send a message to another node that is not its neighbor, it broadcasts a Route Request (RREQ) message. If a node that receives a RREQ message and has the destination in the routing table, it replies with a Route Reply (RREP) otherwise it forwards the RREQ.

LOADng presents simplifications and additional features with respect to AODV. Part of the extensions of the protocol are the modularity (possibility to add arbitrary attributes), short address support and various metrics support. LOADng uses the basic protocol operations from AODV, including Route Discovery and Route Maintenance but in a simplified form: during Route Discovery, RREQ messages are flooded through the network and only the node with the address in the RREQ will respond with an unicast RREP. Route maintenance is performed when an actively used route fails. If a packet cannot be delivered a RERR message is generated, sent as unicast along the route to the source of data packet.

5.3 RPL-LR

RPL-LR [78] is an improvement proposal of RPL to improve its performances. The proposed mechanism includes proactive DODAG construction with link reversal for fast route repair of link failures as well as proactive and reactive point-to-multipoint and point-to-point routing. The core function of RPL-LR is to construct and maintain the Collection Tree (CT) structure of the network. Each node selects a preferred parent to form a CT. One of the advantages of using the scheme proposed by RPL-LR is that only the sink can generate RREQ. If a node has a packet to transmit to another one, it will follow the default route all the way to the sink. When the sink needs to forward a packet for which there is no entry point for the destination in the routing table, it will generate a RREQ and flood the network (Fig. 5.1).

The link reversal mechanism is an adapted version of TORA [79] which is a multi-path routing protocol designed for mobile ad hoc networks (MANETs). Each node has a temporal order that is used for fast local repair of links towards the sink. The goal is to locally repair as fast as possible while minimizing the routing table size at intermediate nodes. The link reversal technique is only used for routes towards the sink.

To trigger a link reversal update, a DODAG update packet is generated when a node cannot find its preferred parent anymore. RPL-LR main drawback is that, in the case of a lost update packet, a loop between the node that lost the connection with its parent and its future node towards a sink can appear (Fig. 5.2). The node that should also change its successor will just ignore the fact that its hop towards the sink has reversed its link.

Another aspect is that if, for some reason, an intermediate node does not have a successor for a packet coming from ”above“ among its connected devices, the packet will be forwarded back towards the sink, creating a loop. This can be avoided by dropping

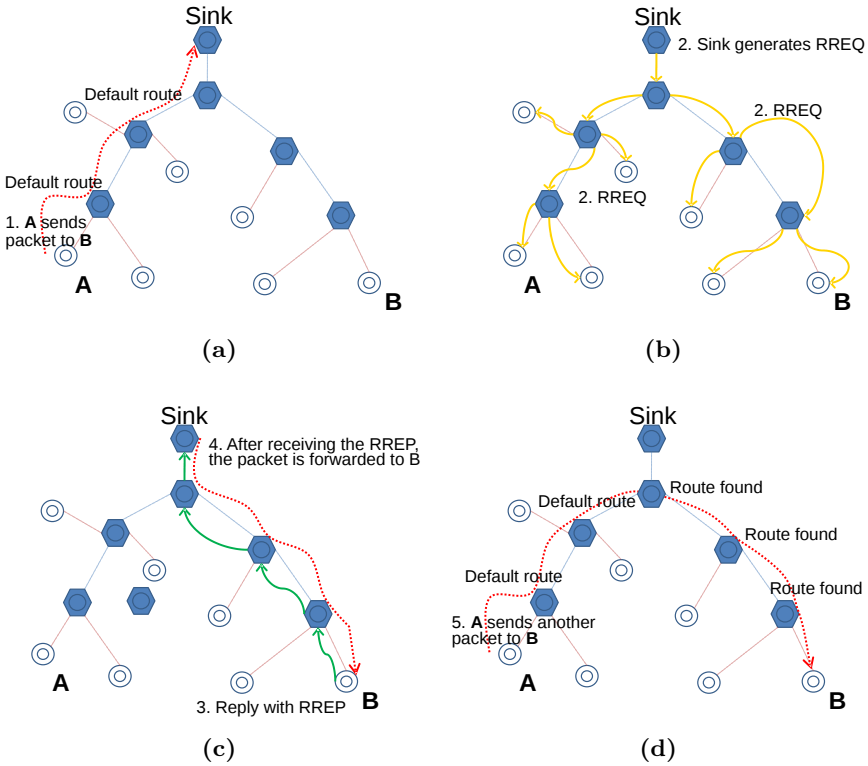


Figure 5.1: Routing scheme of RPL-LR and eventually of LRP for GreenNet: (a) Packet sent to node B takes the default route up to the Sink; (b) Sink generates a RREQ packet flooded throughout the network; (c) Node B replies with RREP and the packet goes over the created route; (d) Any other packet follows the default route and the created route.

every packet that is coming from a lowest rank node and send a RERR to force the sink to send a RREQ. More details on the improvements of routing and implementation in GreenNet are described in Chapter 7.

5.4 Conclusion

This chapter briefly presented the most interesting routing protocols for IEEE 802.15.4 beacon-enabled mode. In next part of the thesis we present in further details the entire GreenNet stack along with our improvements: proposals in terms of routing and energy consumption optimization.

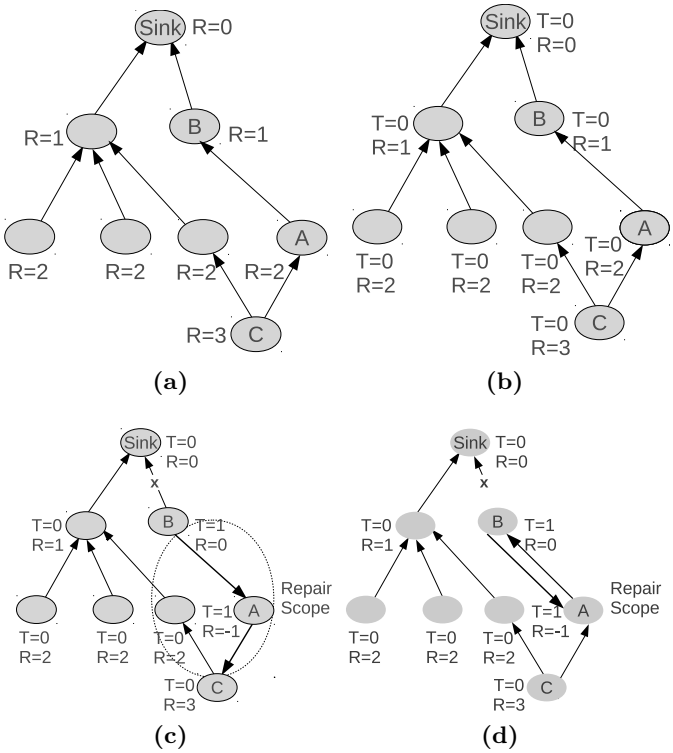


Figure 5.2: a) DODAG with ranks, b) DODAG with ranks and temporal orders, c) localized route repair with link reversal: nodes A and B change their temporal order to 1 and become "higher" in the DODAG than node C, d) If update packet is lost, a loop between A and B can appear

Part II

Contributions

Chapter 6

Enhanced Beacon-Enabled IEEE 802.15.4 MAC

Contents

The main reasons of using the beacon-enabled mode is its ability to achieve ultra low duty cycles. Devices could sleep for several minutes while still keeping synchronization with their coordinator. In other words, if an information is required every 2 minutes (like temperature) a device will only have to stay awake for not more than 10ms, the time to receive the synchronization beacon from the coordinator and send the useful information. This results in extremely low energy consumption which ensures long life of the sensor.

In this chapter, we start by comparing the Beacon-Enabled Mode to ContikiMAC. This comparison goes beyond the mere measurements of Idle or packet transmission power draw in both cases: in fact, the choice of one mechanism or the other has consequences on the entire stack. For instance, ContikiMAC allows a node to communicate with all of its neighbor at any time, but this comes at such a cost that an algorithm to limit broadcast transmission throughout the network is then mandatory. This is one of the objectives of RPL with Trickle [80] or MPL [81] for distributing multicast messages

throughout the network. Conversely, in the IEEE 802.15.4 beacon enabled mode, beacon transmissions need to be organized (Section 6.2) but we still need to reduce the number of relays in the network. This is the subject of section 6.4 while section 6.3 focuses on reducing the power consumption of the coordinators or the sensors beyond what the standard can achieve.

6.1 Beacon-Enabled Mode vs. ContikiMAC

The main advantage of the IEEE 802.15.4 beacon-enabled mode is low energy consumption: most of nodes (leaf devices) just wake up when they need to communicate and the rest of time, they sleep. On the other hand, coordinator nodes may consume some additional energy for sending beacons, but this activity is necessary to organize the network and allow other nodes to join the network or change topology. Duty cycle adaptation may increase the activity of nodes that benefit from better performance when nodes have enough energy.

ContikiMAC provides the abstraction of an always-on link layer and converges to a low energy consumption state because of the phase-lock mechanism that synchronizes the wake up instants of two nodes. ContikiMAC suffers from two main drawbacks. First, broadcast is energy expensive, because the sender stays active during the whole check interval and retransmits a broadcast frame so that all neighbors can receive it. Second, the wake up every check interval allows for only one transmission so that if the first attempt fails, the node retransmits the frame after another check interval, which increases the delay. Compared to ContikiMAC, the beacon-enabled mode supports energy efficient broadcast from a coordinator to devices and provides long periods of sleep (up to 4.2 minutes long for leaf nodes) during which a coordinator holds a broadcast frame until a sleeping device wakes up.

Figure 6.1 and 6.2 present a theoretical comparison of energy consumption by ContikiMAC and the beacon-enabled mode based on an analytical model under simple assumptions: we consider a node sending packets at various intervals and the current drain in the receiving/transmitting mode (at 0dBm) of 4.5mA/4.9mA at 3V. The clock drifts are set to 20ppm and the microcontroller current consumption to 4mA. Note that the $0.125s$ check interval of ContikiMAC and the beacon interval of $0.5s$ provide the same throughput for transferring packets, which leads to a fair comparison of two access methods. We also plot the power drain of ContikiMAC for various broadcast transmission periods corresponding to RPL routing, the main source of broadcasts. As trickle governs the generation of broadcast DIO messages, we have chosen the minimal Contiki RPL trickle interval (4s), a medium one of 1min and the largest one of 17min.

We can see the low consumption of devices that only listen to beacons compared to ContikiMAC. Indeed, a coordinator node may consume more energy, but it can provide synchronization to several nodes that in turn will consume very little energy. Note that the consumption of ContikiMAC strongly depends on the intensity of broadcasts. The irregularities on the ContikiMAC curve appear because a node needs to listen to an additional frame due to the increasing clock drift as packet transmissions are further and further apart.

In conclusion, beacon-enabled mode is good for ultra-low power devices but also for

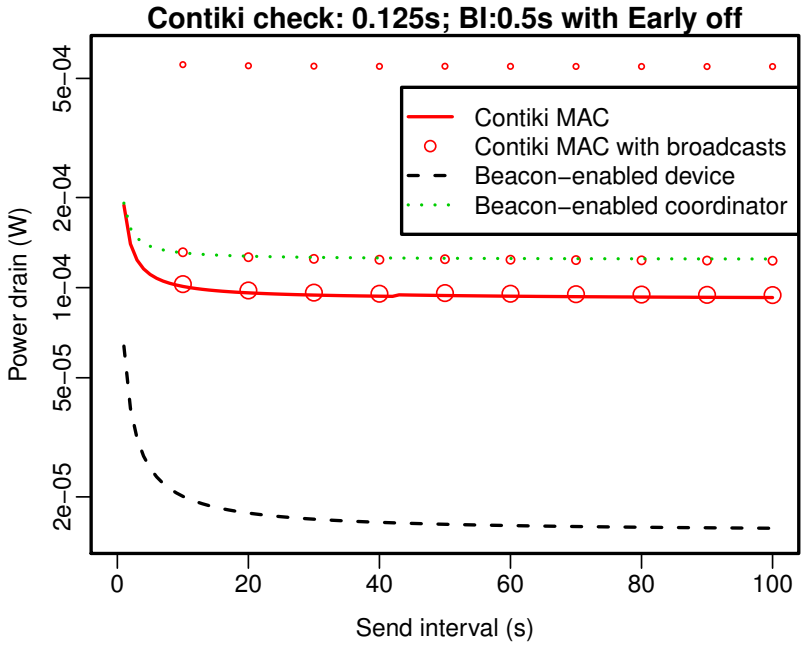


Figure 6.1: Theoretical comparison of energy consumption between ContikiMAC with 0.125s check interval and the beacon-enabled mode with a beacon interval of 0.5s. For ContikiMAC, broadcasts consume significant energy (as unicasts when nodes are not synchronized), so we take them into account by considering three broadcast transmission periods: 4s (smaller dots), 1min, and 17min (larger dots). The vertical axis scale is logarithmic. For Early-off see Section 6.3.1

some low power FFDs.

The SO value has little influence on the results as anyway the coordinator node will go to sleep if there is no traffic. With the early-off mechanism (see Section 6.3.1), the SO value merely determines an upper limit to the time a coordinator stays active for receiving consecutive transmissions. So, SO bounds the consumption of nodes and the network capacity, but if there is little traffic, the duty cycle decreases compared to the nominal duty cycle defined by SO and BO.

6.2 Association Phase

This section focuses on the initialization of the network and in particular the aspect of choosing the *StartTime* parameter for nodes in the network that will become routers.

6.2.1 Scheduling the StartTime of Active Period

The parameter StartTime allows spreading beacons and active periods of coordinator nodes in time. Let us call a slot the duration of the active period starting with a beacon within a superframe. In the ideal case, slots of nodes within the radio range

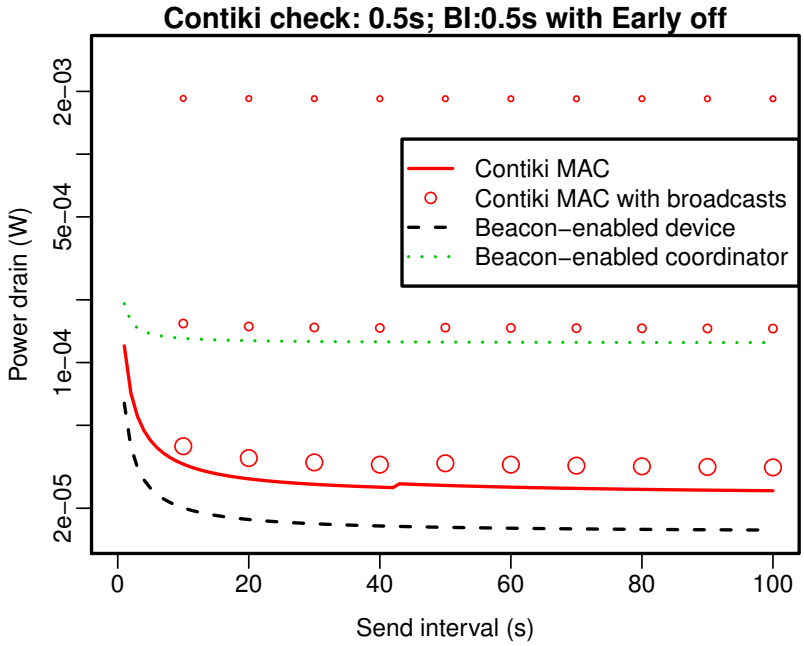


Figure 6.2: Theoretical comparison of energy consumption between ContikiMAC and the beacon-enabled mode with beacon and Contiki check intervals of 0.5s. We consider three broadcast transmission periods: 4s (smaller dots), 1min, and 17min (larger dots). The vertical axis scale is logarithmic.

are non-overlapping in time, because otherwise beacons may collide or overlapping active periods may lead to increased contention. Note that there is a limited number of slots for given parameters BO and SO: 2^{BO-SO}, which may be insufficient for a required number of nodes in the network. For example, Fig. 6.3 shows a network with a $BO - SO = 3$, so the number of possible disjoint slots are $2^3 = 8$.

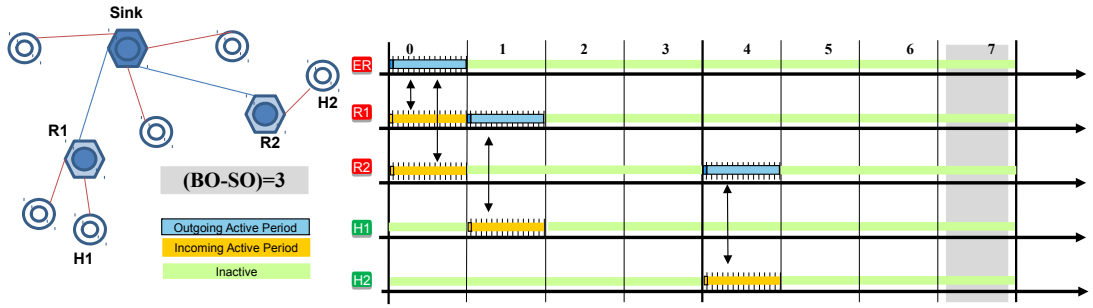


Figure 6.3: Quick view of the active period scheduling concept

There are two main types of solution for scheduling the active periods:

- Centralized, where the PAN Coordinator is managing the number of slots for each router. This solution is good as it ensures non-overlapping active periods but does not reuse slots along the network.

- Distributed, where each node takes a decision based on the information it has locally (or two hop distance).

6.2.2 Non-overlapping scheduling of active periods.

To guarantee a collision-free scheduling of active periods we use the ZigBee Distributed Address Allocation algorithm. The algorithm takes into consideration the network depth (L_m) and the maximum number of FFD of a router (R_m). Each router, to identify the first address of its block, uses the $Cskip$ function:

$$Cskip(d) = \begin{cases} (L_m - d), & \text{if } R_m = 1 \\ \frac{1 - R_m \cdot R_m^{L_m - d - 1}}{1 - R_m}, & otherwise \end{cases} \tag{6.1}$$

where d is the depth of the node calculating the $Cskip$. The n_{th} joining FFD of a coordinator will have the active period slot based on the time of received beacon at which we add the number of slots defined by the formula:

$$slots = (n-1) \cdot Cskip(d) + 1 \tag{6.2}$$

Fig. 6.4 shows an example of slot allocation where the numbers for each node represents the active period slot. It considers a maximum depth of the network of 3 and the maximum number of associated FFDs to a coordinator is also 3.

This solution ensures all the slots are disjoint. The problem is that the number of available slots is at maximum 2^{BO-SO}, so if the depth or the number of FFDs of each router is too large, it can happen that not all the routers will be able to receive a slot, thus the nodes will remain RFDs and will not be able to beacon to accept association from other nodes. The algorithm does not take into consideration that some slots can be reused in the situation where nodes are distributed sparsely in the network.

6.2.3 Scheduling of active periods with potential overlapping

Given the limitations imposed by a centralized solution, we started analyzing a decentralized solution where each node could choose a slot by itself so that no matter where in the network the router is, it would be able to beacon such that other nodes can join the network.

The main problem with this decentralized (fully localized) approach is the possibility to have different routers beaconing in the same time. Avoiding direct interference is straightforward as a node will only have to listen for the entire beacon interval and avoid scheduling its beacons when it hears other routers. The difficult part is to avoid interferences for a node that is between two routers that do no hear each other (known as Hidden Node Problem in literature).

To further investigate the impact of such interferences and check whether it can really make nodes to be deaf due to radio collisions we have experimented with three nodes that send beacons at the same time and a receiver that is moving.

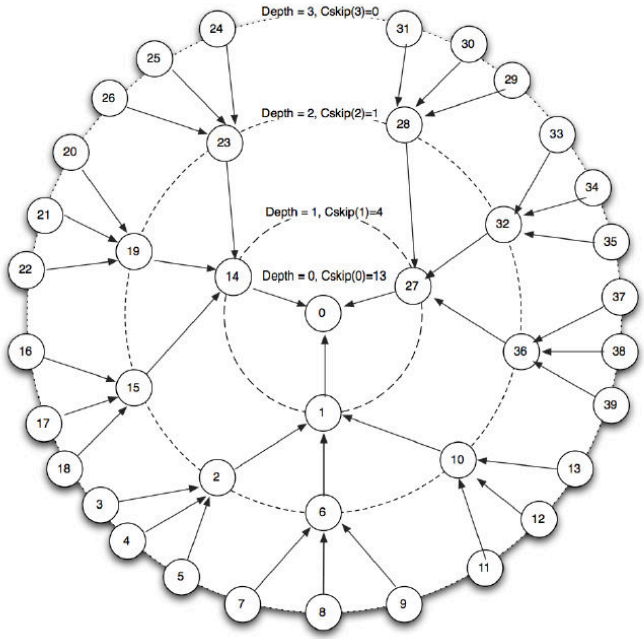


Figure 6.4: Example of centralized slot allocation where max depth is 3 and max FFDs for a router is 3

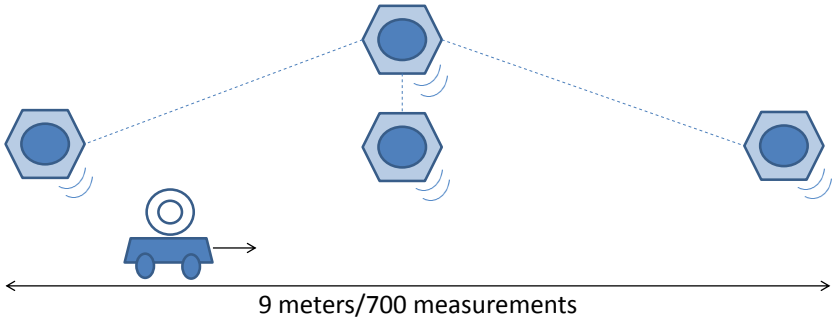
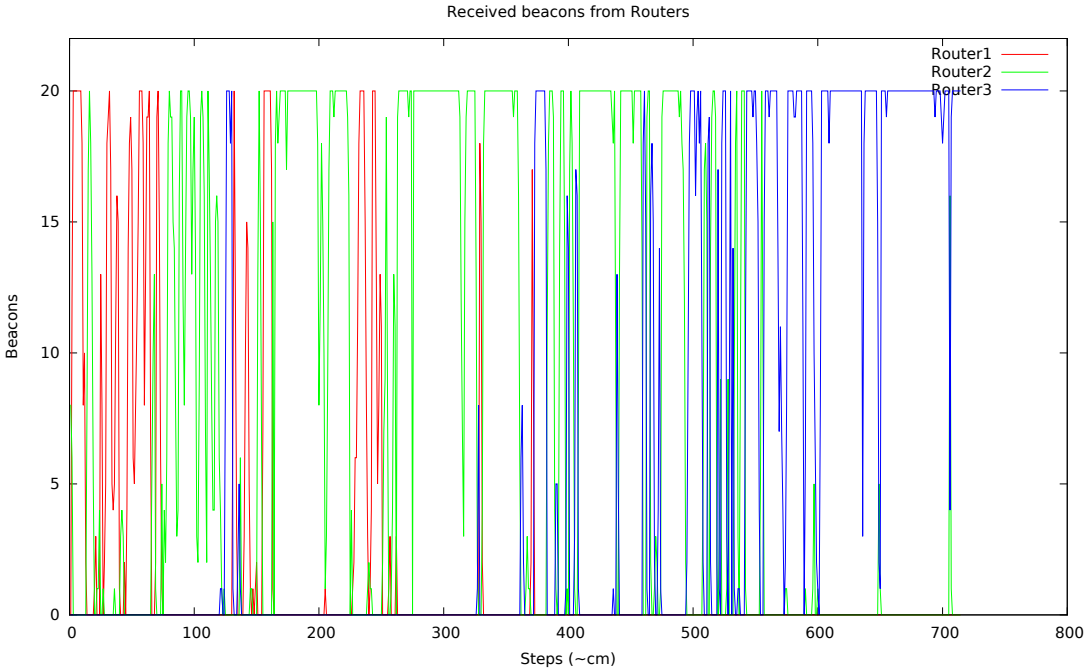


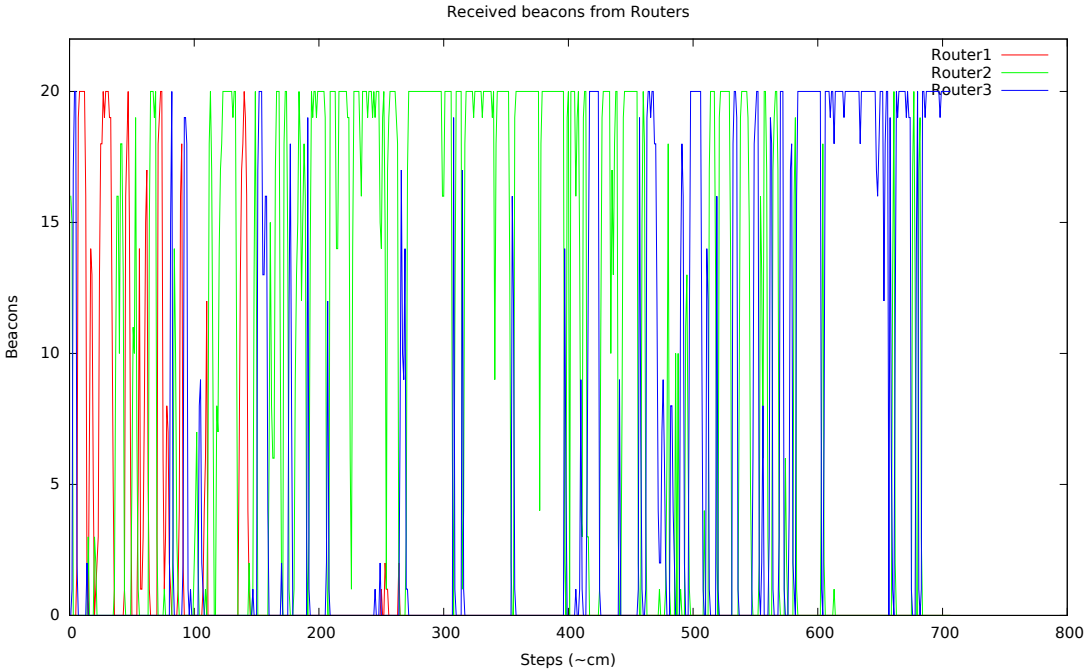
Figure 6.5: Measurement set up

2.3.1 Evaluating Beacon Collisions

The goal of this experiment is to analyze the impact of beacon collisions in the case where three nodes beacon at the same time in close range. The experiment involved a sink and three coordinators associated with it. We place a moving node on a robot near coordinators and the node scans for beacons. It stays at a given position for a short time to receive beacons and moves to the next position: it covers 9 meters with 700 intermediate positions (cf. Figure 6.5). The sink synchronizes the coordinators that send beacons at the same instants—they have the same $StartTime$ for sending their own beacons. At each position, the node should receive 20 beacons if there is no beacon loss.



(a)



(b)

Figure 6.6: Beacons received from 3 synchronized coordinators. Each color represents a coordinator. In most positions, a large fraction of the beacons are successfully received, even though they all collide with each other at similar power.

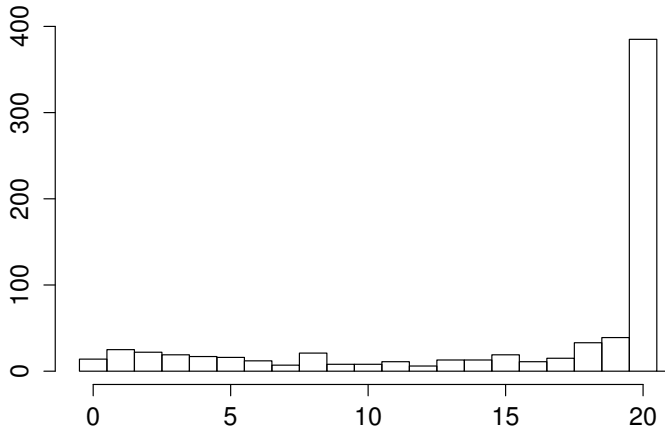


Figure 6.7: Histogram of the received beacons from all 3 coordinators.

Figure 6.7 presents the histogram of the number of received beacons at each position. It shows that the reception of 20 beacons at all positions is highly frequent and the percentage of completely missing the beacons is very low (2% of the total).

Figure 6.6 shows more in detail, for each position, the number of beacons received from each coordinator. Each color represents a different coordinator.

This result confirms the capture effect—the radio receiver can decode a frame even in the presence of other signals. Moreover, at one position, all received beacons come from a single transmitter. The result shows that even when two or more coordinators transmit beacons at the same time, a device can receive at least one of them so that it can join the network and use it effectively, flawlessly.

2.3.2 Network Wide Scheduling of Active Periods

We designed a distributed scheme for choosing non-overlapping slots: each coordinator node transmits in the beacon payload the relative instant of beacon transmission by its parent coordinator. The instant is relative to its own beacon transmission. The child node thus learns about the slots of its parent coordinator as well as the parent of the parent.

The node also listens for all other available beacons. It is doing this so as to avoid interferences not only with the direct parent but also with any other routers in the communication range. This is enough to avoid most direct interference.

After creating a virtual map of occupied slots, the node randomly chooses from the available slots a starting time to beacon. The overhead of this scheme is only the extra information present in the beacon payload that can be as low as 1 byte. The overhead depends on the number of bytes used to represent the number of slots for given values of BO and SO.

6.3 Improving Network lifetime

This section presents several optimizations that can be done such that with a limited amount of energy, the overall lifetime of the network is improved.

6.3.1 Early-off Coordinators

Because of the typical tree topology imposed by the beacon-enabled protocol the coordinator that is closer to the sink will have to relay more data compared to those that are further away (in terms of hops) from the sink. For example, a coordinator that has only one device associated with it has to stay on for up to 10ms after the sending of the beacon in order to receive the potential packet from its device. In contrast, a coordinator that has several devices and/or coordinators has to stay on much longer in order to be able to receive large amount of traffic coming from its associated devices.

Further more, this traffic can fluctuate based on whether the devices are sending all after the same beacon (all devices wake up for the same beacon and try to send data and after other beacons there is no traffic at all). Ideally is to have the packets distributed in time. It is quite difficult to estimate or schedule this kind of traffic so in turn it is easier to have a variable time in which to stay active based on the traffic. Another important aspect is the energy consumption. Because we want to be energy neutral, the nodes should also not stay on more than a certain time. This is easy to impose because when setting up the Superframe Duration with the SO parameter, in fact we know the maximum amount of time that the node will stay awake.

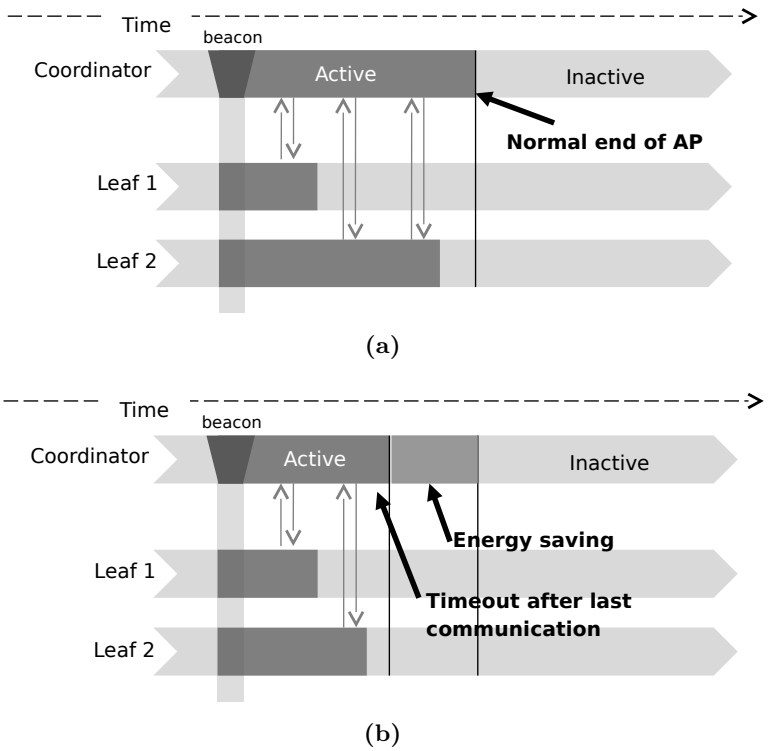


Figure 6.8: Early-off mechanism for coordinators

Having this in mind we propose, in order to avoid bottlenecks in the network and at the same time not to waste energy while staying awake with no reason, to keep the coordinator awake for the time needed to perform the communications or the maximum SD. This is possible because, in practice, the associated nodes of a coordinator will

send their frames just after the beacon. (Fig. 6.8). The maximum time is in fact the time corresponding to Clear Channel Assessment and CSMA-CA backoff. We have implemented this optimization in which the coordinator stays awake another 10 ms after the last packet was received/sent. So the node restarts the timer after each frame transmission or reception so that the node stays awake as long as there is some traffic.

The drawback of this approach is the lack of support for GTS slots, but nodes can save energy by going to sleep early. Even in case of traffic peaks, a node can stay active longer to forward pending frames. With this mechanism a packet does not stay longer than one beacon interval in the buffer of a coordinator node before the node forwards it, if traffic does not exceed the transmission capacity that depends on the active period duration.

6.3.2 Long Sleeping Periods

Beacon-enabled mode supports devices that may sleep during the inactive periods. We implemented a mechanism that allows to skip beacons thus saving energy (Fig. 6.9). This optimization results in some tradeoff in latency, because a device will receive a packet only after waking up, but for sending a sensed data, the node can do it as soon as possible by waking up at the next beacon from the coordinator. In other words, the tradeoff is only for downward traffic, for which the latency grows linearly with the number of skipped beacons, but only at the last hop to the leaf.

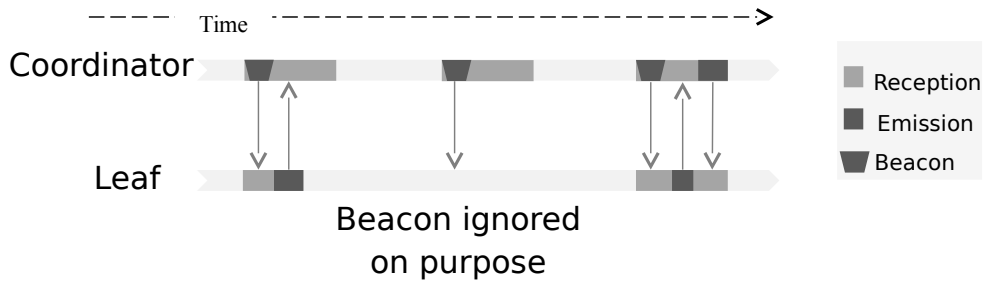


Figure 6.9: Skipping beacons

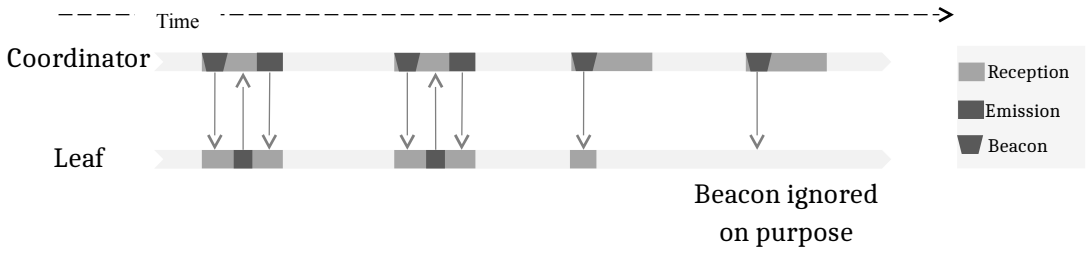


Figure 6.10: Avoid skipping beacons if data is present at coordinator.

To reduce the latency that can arise from skipping beacons, we have implemented a scheme in which if there is data needed to be retrieved from the coordinator, the leaf will avoid skipping beacons and will wake up for the consecutive synchronization

beacons (Fig. 6.10). This is especially useful when for a short period of time a node needs to be awake, like for different types of handshakes (security, CoAP etc.).

Any user should bear in mind that when trying to send packets to devices that wake up at intervals of a couple of minutes, the capacity towards these nodes is very limited. Due to memory constraints, in our implementation, a node cannot buffer more than 20 packets for all their associated devices so flooding the network with downward packets should be avoided as much as possible. The gateway could easily implement some rules that control the traffic that enters the network on a per leaf basis. Nevertheless, it would be either the gateway or a coordinator that would have to drop the extra packets.

6.3.3 Multicasting

It is worth mentioning that multicast packets are not supported by IEEE 802.15.4 standard and that broadcast packets are not supported by IPv6. So we use the terms broadcast and multicast with the same meaning, that is to send a frame to all the nodes.

Multicasting is important, because routing and more specifically, route discovery by means of RREQ messages, builds upon propagating multicast packets to all nodes in the network. Sending a multicast packet to always-on neighbor nodes is easy, because it can be delivered as a broadcast frame. In the case of duty cycled nodes, a similar operation is possible under the IEEE 802.15.4 beacon-enabled mode—a coordinator can send a broadcast frame immediately after beacon. Note that a device associated with a coordinator cannot send broadcasts, so multicasting is limited to downstream traffic, *i.e.* from the sink to leaf devices.

Coordinators cannot apply the same approach for the nodes that skip beacons, which is possible under the IEEE 802.15.4 beacon-enabled mode to save energy. To still support multicasting, coordinator nodes need to deliver multicast packets in unicast frames to nodes that skip beacons. In our implementation, coordinator nodes keep the packets for a maximum sleeping time of devices (max. 4.2 minutes) until dropping them.

Its worth mentioning that in GreenNet network the only multicast/broadcast messages are RREQ that are only generated by the sink.

6.4 The Beacon Forwarding Tree - Building a sparse relaying tree

In some cases it is desirable to limit the number of relays in a network, so that if some nodes already perform this task, there is no need for other nodes in the proximity to relay messages.

We aim at designing a mechanism to construct a forwarding tree that will also serve as a Collection Tree for convergecast. In BFT, each node may become a coordinator in the following distributed way. A node uses two beacon intervals BI_{min} and BI_{max} for sending beacons (Fig. 6.11). As soon as a node associates with a coordinator, it randomly selects instant t ($StartTime$) at which it starts to send a beacon with the interval BI_{max}. If it receives an association requests from a node, it becomes a

coordinator (and a relay for broadcasts) and accelerates the beacon interval to BI_{min}. This change makes it more likely that other nodes wanting to join the network receive its beacon and send association requests. The change increases the duty-cycle of the node. With this method, a node with a larger number of neighbors is more likely to receive first an association request and become a relay. A node that transmits with BI_{max} does not participate in forwarding broadcast packets. Moreover, such a node has a much lower duty-cycle, so it consumes less energy.

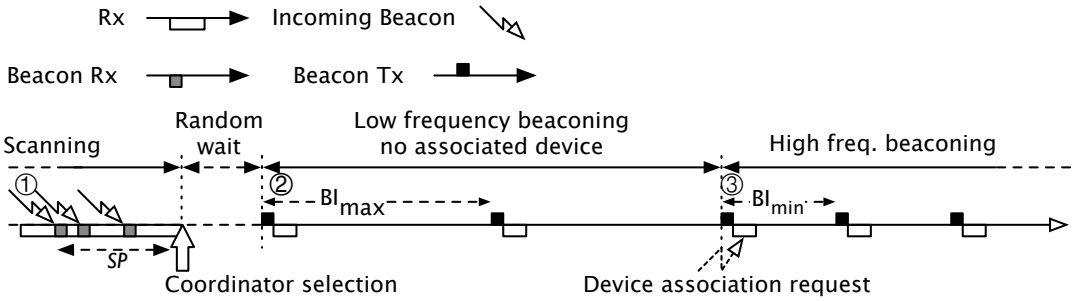


Figure 6.11: Principle of BFT: a node first scans for beacons from neighbors (1) and then chooses a coordinator. It then sends beacons to enable other nodes to join the network: at first, it sends beacons at a small rate (2) and switches to a higher rate if it becomes a coordinator for at least one node (3).

The aim of this scheme is to bias coordinator (and thus relay) selection towards selecting the nodes that are already relays (they send beacons at a fast rate). The probability that a newly arriving node receives first a beacon transmitted by an already active relay in presence of D nodes sending beacons with a slow rate is:

$$P[\text{Choose already active}] = \frac{BI_{\max}}{BI_{\max} + D \times BI_{\min}}.$$

The probability is greater for greater ratio $\frac{BI_{\max}}{BI_{\min}}$, which results in more nodes joining an already active relay so the number of relays does not increase. Nevertheless, there is a trade-off: a large value of $BI_{\max}$ gives us a lower number of relays throughout the network, but results in a long latency of association for a node that would be covered by a limited number of coordinators.

As a result of this scheme of operation, BFT creates a forwarding tree with a small number of relays. It guarantees the coverage of all nodes in the network, since they all associate with a coordinator as long as the network is not partitioned. To find minimal hop paths, nodes may include the distance from the broadcast source in beacons (typically, this information can be included in the payload of 802.15.4 beacon frames). To optimize the distance to the broadcast source, nodes should overhear all nearby coordinators and collect the distance information in beacons for scanning duration SP. SP can vary from 0 (no path optimization) to $BI_{\max}$ (the variant that we call an Optimized Beacon-based Forwarding Tree—Opt-BFT). The scanning procedure can be repeated later on to allow nodes to reselect the optimal paths in case of a topology

change.

6.4.1 Multicast Protocol for Low Power and Lossy Networks (MPL)

In a WSN composed of resource-constrained devices, MPR-based (Multipoint Relay) [82] mechanisms require the two-hop topology information and the cost of maintaining such information grows polynomially with network density. MPL [81] avoids the need for constructing or maintaining a multicast or broadcast forwarding overlay. It uses the Trickle algorithm [80]: on receiving packet m, node i selects a random interval $I \in [I_{\min}, I_{\max}]$ and computes random waiting time $t \in [\frac{I}{2}, I]$ during which it counts the number of times k it receives forwarded packet m. After time t, node i suppresses forwarding of m if $k > K$ or forwards m otherwise. K controls the transmission redundancy throughout the network. MPL is simple to implement and requires only a limited memory to keep track of on-going broadcast dissemination processes. Basically, it needs to store the on-going broadcast packets and the associated counters k that count redundant retransmissions. The forwarder set in MPL is created on-demand and may change randomly with every packet, hence it is more dynamic compared to MPR-based mechanisms.

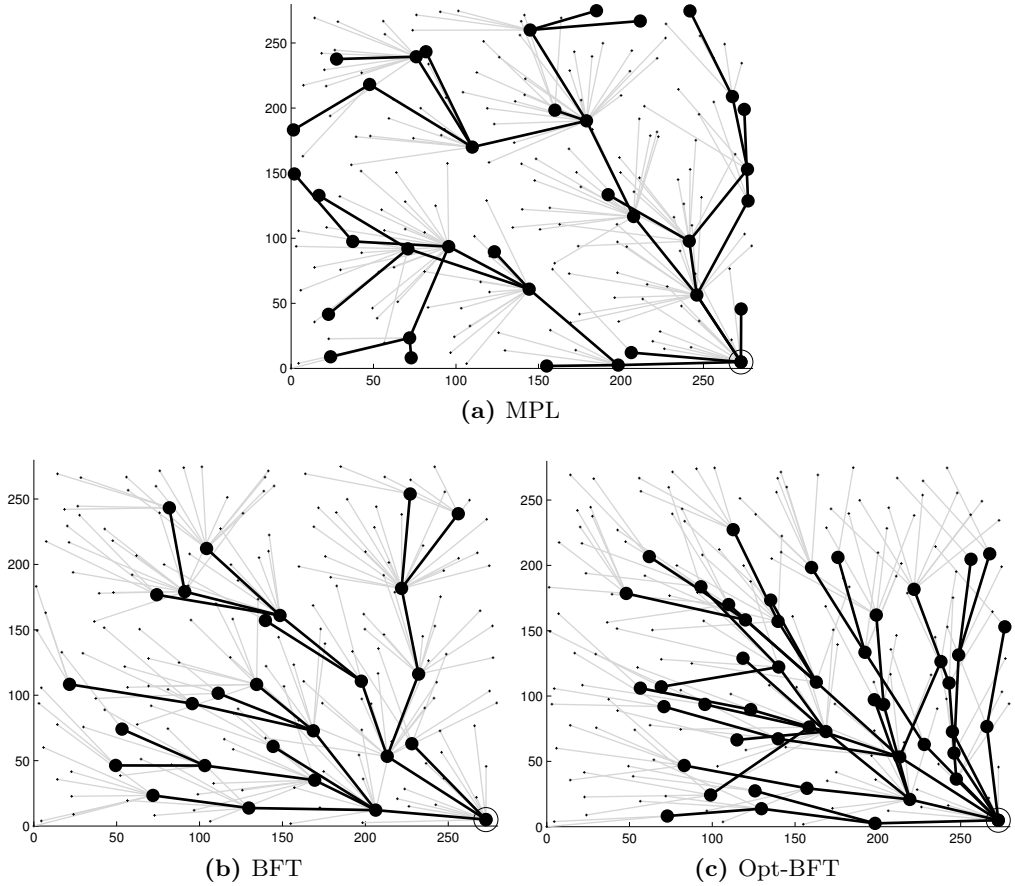
Nevertheless, MPL suffers from several drawbacks. MPL does not guarantee minimal hop count paths. In a low duty cycle WSN with a MAC layer based on preamble sampling such as ContikiMAC, sending a local broadcast to all neighbors requires repeating the transmission during the whole duty cycle that may last for a long time compared to the packet duration, which consumes a large amount of energy. Packet repetitions increase the load on the channel and the latency whereas collisions increase the overhead and make it challenging to count the actual number of forwarded broadcast packets. Furthermore, the choice of a low redundancy limit K in MPL may impact the broadcast delivery coverage, which is not guaranteed to include all destinations anyway. In fact, even when there is no packet loss, broadcast reachability can be affected by canceled retransmissions when K is reached even though some nodes have not received the packet.

6.4.2 Theoretical analysis of MPL and BFT

We have implemented the MPL algorithm and proposed BFT scheme in a simulator, a Matlab program that allows modeling the relay selection schemes with Unit Disk graph assumptions. For MPL, we set $I_{\min} = 1$, $I_{\max} = 20$, and $K = 3$, the time unit being the simulator time tick in this case, as only relative values matter in the simulation. To build BFT without path optimization, we use $BI_{\min} = 1$, $BI_{\max} = 20$, and $SP = 0$. For Opt-BFT, the scheme with path optimization, we select scanning duration $SP = BI_{\max}$ to obtain the distance to the broadcast source of all nearby coordinators before selecting the relay.

We have run simulations for a varying number of nodes uniformly placed on variable size square areas. We set the radio range r to 80m (unless otherwise specified) resulting in an average node degree of 40 on the areas of side lengths from 280m to 640m with 200 to 1000 nodes. The broadcast source is randomly selected in each simulation run and we average the results from at least 5 simulations runs. To make the results comparable,

we use the same set of broadcast sources for all evaluated schemes.



(a) MPL

(b) BFT

(c) Opt-BFT

Figure 6.12: Forwarding relays for 200 nodes with average node degree of 40. Big black dots represent the relays. The broadcast source is at the right bottom corner. [3]

Figure 6.12 shows the set of relays selected by different schemes in a 200 node topology (the figure illustrates the cases in which the broadcast source is at the bottom right corner).

Relay selection under MPL is interestingly biased in a way such that the nodes at the edge are more likely to become relays as they naturally have less opportunity to receive broadcast packets multiple times. On the other hand, BFT excludes the nodes at the edges, because they do not give access to any other node. BFT builds a sparse tree, because existing relays have more opportunity to propose association to other nodes.

6.4.3 Implementation of MPL and BFT

We have implemented MPL and BFT under Contiki OS [67]. BFT uses our implementation of the IEEE 802.15.4 beacon-enabled mode. To include the information about the distance to the broadcast source in BFT, we have developed a routing layer

on top of the 802.15.4 MAC (See Chapter 7). It allows a joining node to pick its neighbor closest to the sink as a coordinator. We set the beacon-enabled 802.15.4 parameters to BO=5, SO=1, so the beacon interval is 0.5s and we use the intervals of 8s between beacons (BO=9) to achieve the slow beacon sending rate.

We compare BFT with MPL running on top of ContikiMAC configured with a channel check interval of 62.5 ms, whereas the broadcast transmission duration is 125ms. The parameters allow sending approximately the same amount of unicast traffic: 7 frames every 500ms in beacon-enabled 802.15.4 *vs* 1 frame every 62.5ms with ContikiMAC.

We consider a simple topology formed by a chain of 3 nodes (cf. Figure 6.13) with the distance between Nodes 1-2 and 2-3 of 2 meters. We set the transmission power to -23 dBm, which creates a two-hop network (even though from time to time, with ContikiMAC, some packets are directly received from Node 1 by Node 3). Although this configuration may include hidden nodes, we have observed that it is not harmful, because the modulations of the 802.15.4 radio standard are extremely robust[1].

Node 1 is the broadcast traffic source, Node 2 relays traffic, and Node 3 has no associated nor downstream devices. With one coordinator, one intermediate coordinator associated with Node 1 and a simple device, the difference between the behaviors of nodes is much more marked in BFT/802.15.4 compared to MPL/ContikiMAC in which all three nodes will relay packets as the redundancy limit cannot be reached. The setup is basic, but it already allows us to capture many properties of the two considered approaches. Besides, since we have already seen in Section 6.4.2 that the number of relays is similar under BFT and MPL, we focus here on assessing the energy consumption by BFT and MPL along with their associated MAC layers.

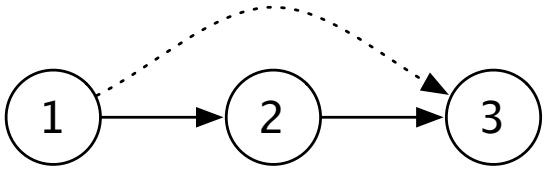


Figure 6.13: Simple topology—a chain of 3 sensors. Under ContikiMAC, some frames are directly received by Node 3 from Node 1. Under 802.15.4, Node 2 is associated with Node 1 and Node 3 with Node 2.

We first look at the time passed in transmission by the nodes in Figure 6.14. The general trends match the expected behavior: ContikiMAC is energy expensive under notable broadcast traffic. Conversely, beacon-enabled transmissions are strongly dominated by the energy consumption due to beacon transmissions, so the presence of traffic has much less impact. One important feature of BFT is apparent here: Node 3 that uses a slow beacon rate (it does not have any associated node), spends very little time in transmission mode.

As the energy consumption during idle listening is also significant, we evaluate the fraction of the time the radio is on for reception or transmission (cf. Figure 6.15).

[1]Nodes can differentiate two signals that arrive at the same time as long as they have at least a couple of dB of difference in their received power [83, 84].

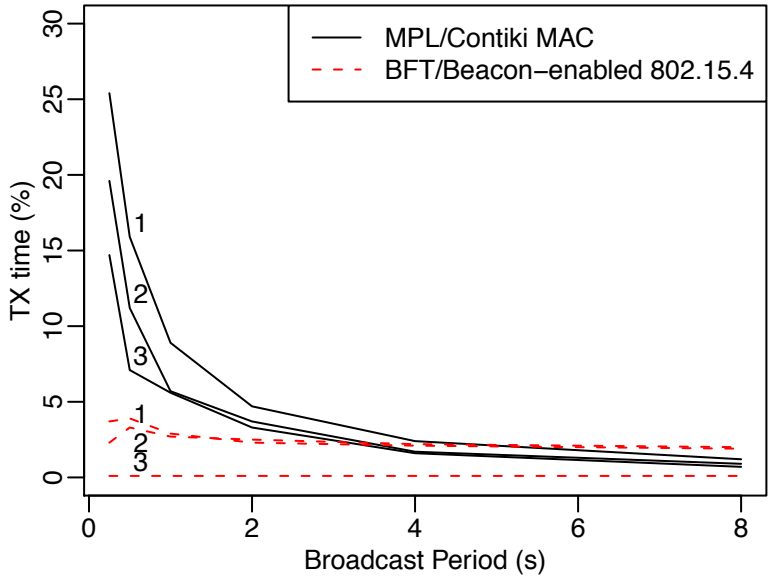


Figure 6.14: Proportion of time the radio is transmitting for a chain of 3 sensors.

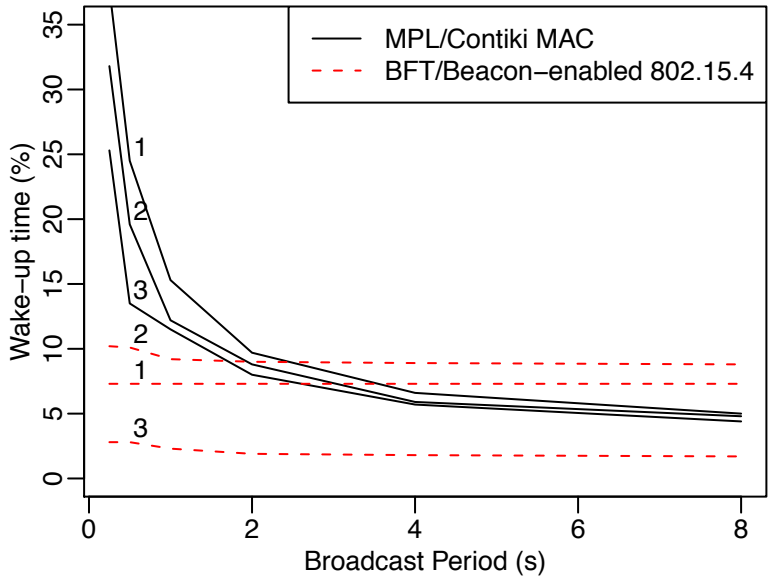


Figure 6.15: Proportion of time the radio is up for a chain of 3 sensors.

For ContikiMAC, it is the sum of packet transmission, reception, and periodic channel checks. For beacon-enabled 802.15.4, it includes the beacon transmission and the following CAP period for a coordinator (we do not define any GTS slots) as well as only the beacon reception for an idle leaf node. The performance difference between the relaying node and Node 3 under BFT is even more noticeable than for transmissions. We can observe that Node 2 is more active than Node 1, because it needs to receive the beacon from Node 1 and also acts as a coordinator on behalf of Node 3.

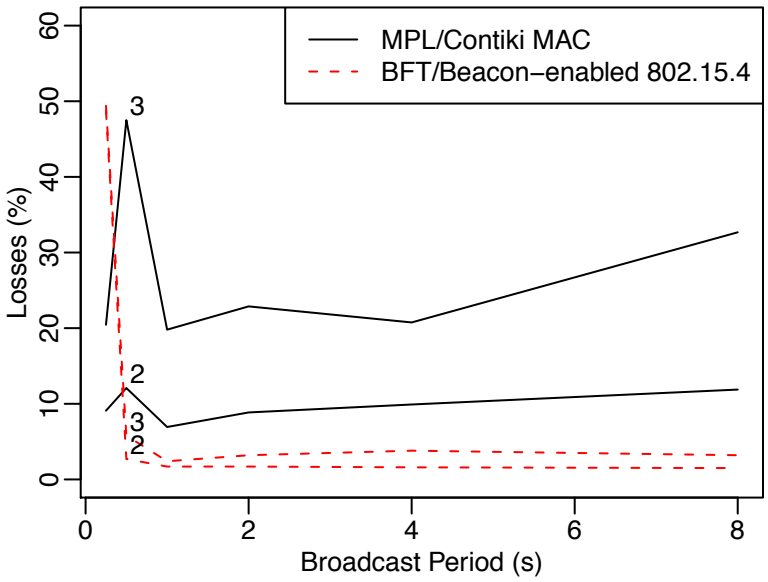


Figure 6.16: Percentage of lost frames at Node 2 and 3 with respect to the number of frames sent by Node 1.

Loss rate that we can observe in Figure 6.16 may be in part due to our implementation of ContikiMAC that fails to receive ≈10% of the packets, so that losses accumulate at Node 3, which explains the difference between the transmissions times observed between the 3 nodes in the previous figures. For a broadcast transmission period of 0.5s, many packets are lost at Node 2 due to a failed Clear Channel Assessment (CCA) before the transmission starts. Note that broadcast periods of less than 1s are well below the typical traffic MPL/ContikiMAC was engineered for and we have tuned $I_{\min}$ and $I_{\max}$ accordingly: $I_{\min} = 1$ for periods of less than 1s or 2, otherwise; $I_{\max}$ is equal to 2, 4, and 7, respectively, for periods of 0.25s, 0.5s, and 1s or more.

In BFT, losses for the 0.25s period are also due to the limitation of our implementation as nodes store only one packet for later transmission and they can only send one packet after each beacon.

6.4.4 BFT on a 16 Node Testbed

We have also run BFT on a testbed of 16 nodes. One node is the edge router, 4 nodes are Full Function Devices (FFD) that can act as relays if needed, whereas the remaining 11 nodes are Reduced Function Devices (RFD) that can only associate as leaf nodes. All nodes are in the range of communication of any other node.

Without BFT, when 4 FFDs send beacons at a nominal rate (BO=4), all 4 nodes have associated nodes and thus become relays 9 out of 10 times. With BFT, only one FFD sends beacons at a nominal rate (BO=4) and other FFDs use a larger BO=9 (32-fold increase of BI), which results in a single FFD becoming the relay.

6.5 Conclusions

This chapter presents a set of optimizations and improvements that we propose on top of IEEE 802.15.4 beacon-enabled mode in order to achieve very low duty-cycles in multi-hop networks. We have also evaluated how the radio behaves in case of multiple beacons sent at the same time. This allowed us to develop a much simpler solution for deciding the $StartTime$ of beacons in multi-hop networks.

After the bootstrap of the network we focused on improving the energy consumption while the network is running. This led us to implement different mechanism that help to reduce the energy consumption, like early-off for coordinators and skipping beacons for devices. The coordinators that do not have any associated device can reduce their duty-cycle by sending less frequently the beacons that invite other devices to join the network. With these improvements, the coordinators consumption is reduced to a point that is compatible with energy harvesting.

Chapter 7

Routing in GreenNet

Contents

The choice of duty cycling mechanism drives the design and the choice of configuration parameters for the routing protocol that runs on top of it. Just like ContikiMAC and other preamble sampling justify to some extent the design of RPL, this chapter details a routing protocol that suits the needs of the GreenNet project.

7.1 LRP for GreenNet

Routing needs to support specific traffic patterns related to sensor applications [85]. This includes in particular convergecast traffic. LRP (Lightweight Routing Protocol) [86] builds a Collection Tree (CT) [87] in a proactive way to form the spine of the sensor network. The CT structure is also the core principle of RPL (Routing Protocol for Low-power and lossy networks) [27] in which each node selects a preferred parent to form a CT[1]. As LRP operates on top of beacon-enabled 802.15.4 that builds a tree topology, its L3 topology can only be a tree. Nevertheless, if the underlying MAC layer supports any topology, LRP can also create a DODAG instead of the CT by maintaining several parent nodes.

The CT backbone enables efficient multicast/broadcast, point-to-multipoint (P2MP) communications, sending packets from a sink to all nodes. Multicasting over a CT can dramatically reduce the cost of reaching all nodes in the network, the function required for instance for route establishment or application layer queries (Chapter 6).

Maintaining downward routes to all nodes in a proactive way as in RPL may be too expensive in terms of the overhead and memory consumption [88]. Besides supporting a proactive scheme like in RPL, we have also chosen to provide an on-demand scheme to build point-to-multipoint routes as in LOADng [76]: whenever only a subset of nodes

[1] RPL nodes keep the information on several parents, so the actual structure built by RPL is more general—a DODAG, Destination Oriented Directed Acyclic Graph.

needs to be accessible at any time (*i.e.*, the network only needs to support low point-to-multipoint downward traffic), reactive routing is an interesting solution even though it may introduce latency as nodes may search for a route with a network lookup that may result in significant traffic. Constructing routes on demand is also interesting for sporadic P2P traffic between any node in the network. The choice between a proactive scheme like RPL or an on-demand one depends on traffic and specific network usage.

To search for a route in the on-demand scheme, LRP uses a Route Request (RREQ) message. A node floods RREQ to reach a destination node. When the destination node receives RREQ, it generates a unicast Route Reply (RREP) message. If a node detects a broken route (when it cannot forward a packet to a destination), it sends a Route Error (RERR) message to the originator of the packet that failed to reach the destination.

Besides the messages for reactive routing, LRP also uses: DIO (DODAG Information Object) that are used by the nodes to advertise their distance to the sink. The sink regularly increases the DIO sequence number ("DODAG version") to allow for quick re-attachment even further down the CT.

7.2 Building a Collection Tree with LRP

To form a CT rooted at the sink, each node needs to select a preferred coordinator. The selection takes into account a rank that monotonically increases from the sink. A node that wants to join the network, scans a particular channel for periodic beacons that provide DIO routing messages with the rank information in their payload (which is impossible with RPL as DIOs are too big to be included). When a node receives the rank from all its neighbors, it can associate with the node closest to the sink. If a coordinator refuses the association request, the node will avoid any further association with it during a period of time, while trying to associate with other potential coordinators. In the case of a link failure, the node will rescan the channel to find other nodes and select the best one. This way of constructing the Layer 2 topology takes into account the metrics from Layer 3 so that the topology has all the desired characteristics for low duty cycle operation at Layer 2 and efficient packet forwarding at Layer 3.

7.3 Downward routes

Nodes can construct downward routes either in a reactive or proactive way. A node joining the network can use the proactive scheme: it may send an unsolicited RREP to the sink to create a route toward itself. The sink address is known as this information is contained in the DIO message sent in the beacon payload. The sink can also use the reactive scheme to rebuild a route towards a specific node, for instance if a routing table overflowed along the way. A node that associates with a coordinator, inserts a default route towards the sink with the coordinator as the next hop. The node will use the default route for any destination address not present in the routing table. All nodes on the route to the sink apply the same algorithm until the packet reaches the sink.

When the destination address is not in the routing table, for instance when the packet reaches the sink through the default routes and it does not have an entry for the destination, the sink will broadcast RREQ and nodes forward the message down

Table 7.1: Features of LRP vs. RPL

DIO	Similar to RPL DIO. It is included in the beacon payload. Contains the rank information of the node—the distance to the sink and some information for choosing the initial instants of superframes (*cf.* Section 6.2.1).
RREP	Equivalent to DAO. Sent proactively when a node joins the network or on-demand as a reply to RREQ.
No Trickle	In the case of beacon-enabled 802.15.4, Trickle becomes irrelevant, because the wake up periods of nodes are governed by the beacon interval and not by the Trickle adaptation.
Collection Tree/DODAG	Tree formed with the preferred parent, potentially DODAG with possible several parent nodes. Nodes create upward routes after association. On-demand downward routes.
Objective function	OF0 (hop distance), but several other metrics can be used.
L3 routing	Default route with the coordinator as the next hop except for the sink that generates RREQ for an unknown address.
NUD (IPv6 Neighbor Unreachability Detection)	Not used. The lack of beacon reception for upward routes can indicate a link failure.
ND (IPv6 Neighbor Discovery)	Not used. Neighbor discovery by scanning channels for beacons.

the CT. When the packet reaches the destination node, it replies with a unicast RREP message. The intermediate nodes add a routing entry according to the information present in the message. Table 7.1 presents a short comparison of LRP features with those of RPL.

7.4 Managing Routing Tables

Due to strict memory limitation and a very high number of nodes that can potentially be present in a network we have implemented a rather simple way of managing the routing tables. As long as there is free space available we add the new entry and in case the table is full we replace the oldest entry with the new one.

Depending on the size of the network some nodes might need to replace entries in the routing tables and others not. So a route could be broken somewhere in the middle between the sender (or the sink) and destination. To avoid loops where a node receives frames from its coordinator but doesn't have a route entry for the destination, the node should drop the packet and send a RERR for the specific destination. So we need to keep track of where the packet came from, at least the fact that it is coming from the coordinator or elsewhere. After the sink receives the RERR, it generates a RREQ to

create the complete route to the destination.

7.5 Fast route creation for sleepy nodes

In GreenNet network coordinators can work with higher duty-cycles, depending on the available energy, and leaf nodes could just wake up when they need to send a data or to synchronize their clock. The problem for long sleeping nodes is that in case they need to reply to a RREQ message it might take minutes until they wake up. This can create a long wait only for creating the route. To speed-up this process we implemented a proxy mechanism where the coordinator that has such nodes associated to it, can reply immediately in the name of the leaf. The big advantage is that during the time the leaf is still sleeping the route is created and next the packets can be forwarded and stored by the coordinator. When the leaf wakes up it can already retrieve the packets with the leaf destination.

This solution should also keep track if the leaf is still attached to the coordinator. The simplest way to do this and remain standard compliant is for the coordinator to send a dummy packet to the leaf. In this way when the leaf first wakes up, it sends a Data Request to retrieve the packet. If after a time out, that can last up to 4 minutes, the leaf does not demand the packet, the coordinator can consider it is no longer associated and send a RERR to sink in order to remove the route towards this particular leaf. A solution that could be implemented to reduce the number of exchanged packets (Data Request + Ack, Frame + Ack) to a single packet, is for the leaf to send a spontaneously "I am here" frame to coordinator. It not only reduces the number of packets exchanges but it allows the leaf to go to sleep earlier, thus saving energy.

7.6 Mobility

When it comes to mobility, it can happen that at some point a node is changing coordinators (not necessarily its physical position) and the proactive RREP that is sent after the new association should replace the old routing entry for nodes that have it (especially in the case of sink). One simple way to implement this is to have a monotonically increasing sequence number for route replies. But as a node can also reset and lose all its RAM memory the safest way is to store the last used sequence number in the flash memory and at every reset retrieve it. This imposes a writing to flash memory after every RREP packet.

A very important aspect to mention is the lost of synchronization for a FFD that has other nodes associated with it. In the present implementation of the stack a node that does not receive any more beacons from its coordinator will just reboot and try to join the network by performing the listening and all the protocol steps. This means that a lost link for a coordinator will propagate in the entire subtree of the node.

In future, in case of link failure, we should keep the beacons that are meant for the associated nodes and during the inactive period perform a scanning and association procedure. One would have to take care of timings and timers, as at least on GreenNet platform, all events are relative to the reception of the synchronizing beacon. After association with another coordinator all the events should be modified in such a

way that it will be invisible for the associated nodes, meaning that downwards beacons are sent at the same time relative to the associated nodes.

Having fixed the timers, the next problem is the routing tables that are not up to date. Most probably if a node should be contacted, a RERR will be first generated followed by a RREQ and eventually the packet can reach its destination after the RREP of the node.

7.7 Conclusions

In this chapter we present the details regarding the routing protocol we implemented in GreenNet. Even though most of the concepts used by LRP are present in either LOADng or RPL, we still had to improve the protocol on top of the low duty-cycle link layer. This involved putting the routing information inside the beacons payload that helps at creating the Collection Tree. Further more, we sent instantaneous RREP messages for every devices that is joining the network. We keep the RREP sequence number stored in flash so that in case a node changes parents the new routes are correctly updated.

To avoid loops in case some paths are missing, the rule of thumb is to not send a packet that is coming from the coordinator back to it. Instead, a RERR needs to be sent to inform the sink that the route is broken so it will trigger a RREQ for that specific destination.

We solved the mobility problem by using always increasing sequence number for route replies but we still have to solve the case when a coordinator looses its synchronization and generates a domino effect for its associated nodes.

Last but not least, we proposed a solution for very low duty-cycle nodes to have their coordinators reply to a RREQ, to enable a faster route creation. This puts more effort on routers as they need to track if the devices are still associated and alive.

Chapter 8

Complete Stack Evaluation

Contents

This chapter reports on the evaluation of the energy consumption and performance offered by GreenNet. First, it analyzes the energy consumption of nodes and gives measures of the available power from energy harvesting. This allows to determine the duty cycling values that permit to reach a self-sufficient sustainable operation. The efficiency and impact of the early-off operation of coordinators is evaluated as well as the delay for communicating with leaf devices.

Another performance indicator that appears below is the time for the network to converge to a stationary state in which all nodes are associated, routes are created and application level registering is completed.

Nodes run the full GreenNet protocol stack with CoAP at the application layer and a subset of the Smart Energy Profile (SEP 2) [89] specified by the Zigbee Alliance for IP-based control of energy management in Home Area Networks. At bootstrap, the nodes perform the SEP 2 registration in addition to the 802.15.4 association process. Once registered, they send sensed data at a given time interval in one CoAP message. An example packet is 108 byte long with the payload containing the temperature data over 11 bytes, the rest accounting for headers of all the involved layers.

8.1 Network Initialization

When we switch on nodes, they begin to scan for beacons and associate with coordinators, which results in the creation of the CT. After association, a node terminates this initialization phase with SEP 2 registration. To measure the initialization time, we have used a testbed with 16 nodes (the sink, two coordinators and 13 devices) with a 2-hop depth. Table 8.1 presents the total initialization time that includes the association and the SEP 2 registration (12 CoAP packets exchanged). To reduce interference, we switch nodes on at random intervals in the range of 0-50 BI. We run 10 experiments for each couple of parameters (BO, SO) and average the results.

Table 8.1: Average initialization time.

(BO, SO)	(BI, SD)	Initialization Time (min)
4, 1	246ms, 31ms	∼1min 30s
5, 1	492ms, 31ms	∼2min 30s
6, 1	983ms, 31ms	∼5min 30s

8.2 Energy Efficiency of 802.15.4 Enhancement

To evaluate the energy efficiency of our enhancements, we evaluated the real energy consumption of the nodes. The mean energy consumption of a device is 3μA when it skips beacons and sends packets every 4 minutes. The mean energy consumption of a coordinator is higher with 50μA under the parameters (BO, SO)=(10, 1) - (BI, SD)=(15.73s, 31ms).

Figure 8.1a shows the consumption profiles of a coordinator and a device for different cases: wake-up with no traffic and sending sensed data. In Figure 8.1a, a coordinator wakes up for a period of 31ms corresponding to SO=1. Figure 8.1b shows a device waking up to receive a beacon from the coordinator and going to sleep as there is no traffic to send nor to receive. Figures 8.1c and 8.1d present the consumption of the device when sending two sensed pieces of data: temperature and light. We can observe that for long active periods, the early-off mechanism can save energy, because the coordinator for which there is no traffic, can go to sleep 10ms after the last packet sent or received.

Figure 8.2 shows the current consumption of a router for different (BO, SO) combinations when its radio is on during the incoming and outgoing superframe periods. The LUX lines correspond to the harvested current for a given light intensity (24h a day). The router may operate autonomously for the (BO, SO) parameters corresponding to a point below a LUX line.

To evaluate the energy efficiency of the early-off mechanism and validate the overall operation of the integrated stack, we have set up a testbed with 11 nodes including 1 sink as presented in Figure 8.3a. Nodes are within the transmission range of each other. We force the creation of the CT with the rule of maximum 2 nodes associated with the same coordinator, which results in a tree with a depth of 3. We test the network with parameters (BO, SO) = (6, 3), that results in 8 FFDs and 2 RFD nodes. All nodes send a packet every beacon interval with temperature data (there is a packet every 983ms).

We use the ENERGEST function of Contiki that measures the time between two events (on/off in our case) to evaluate the wake-up duration. We can then find the duty cycle from the wake-up duration.

Table 8.2 presents the wake-up duration and the duty cycle for different types of nodes with and without the early-off mechanism (Section 6.3.1) as well as the number of forwarded packets and loss rate. We can observe a significant gain in the duty cycle for coordinators that are further away from the sink—they have less traffic to forward so they can go to sleep earlier than defined in the standard. The maximum gain is up to a 4 times less active period duration for the coordinator, while the number of

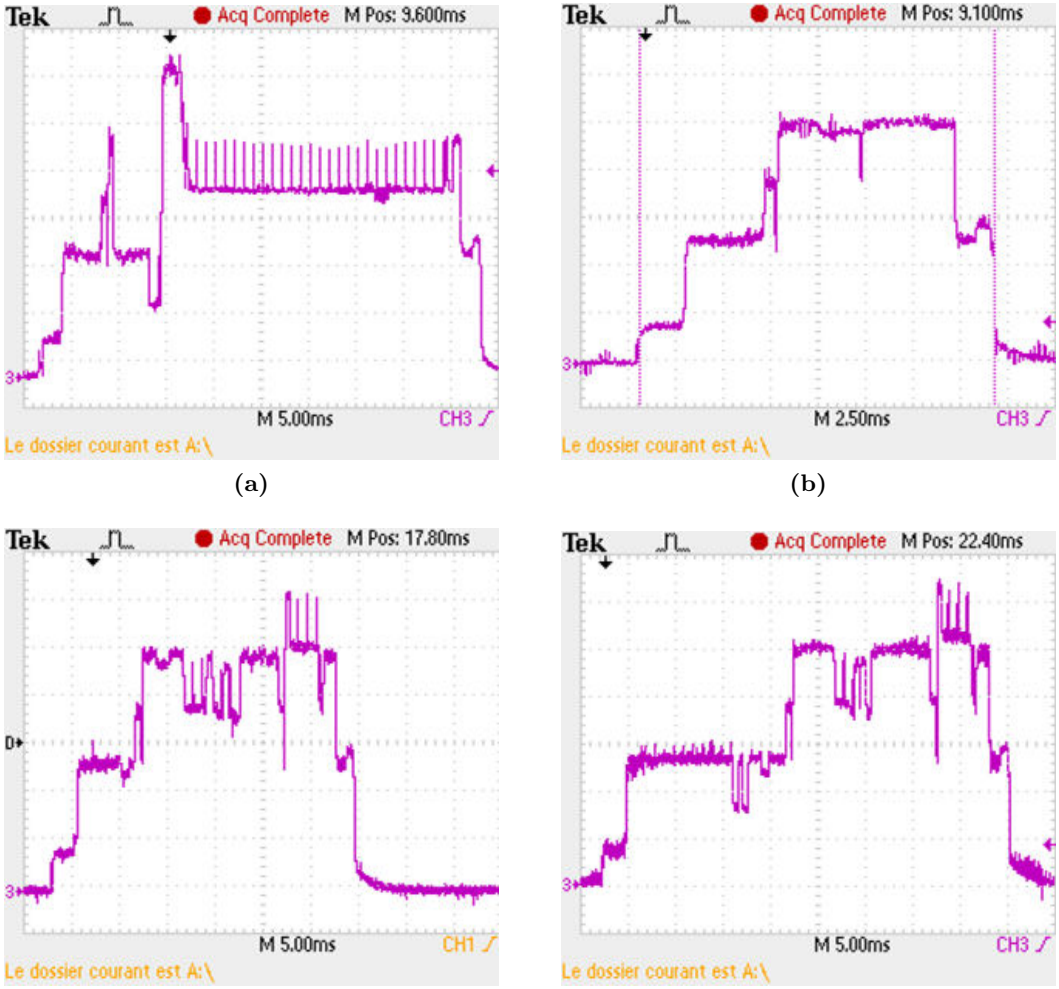


Figure 8.1: Consumption profiles (one unit on Y axis is 2mA): (a) Coordinator wake-up: send beacon and stay active for communication with devices (active period of 31ms, SO=1). 13mA current peak, 7.3mA average. (b) Device wake-up: receive a beacon from the coordinator and go to sleep as there is no traffic. (c) Device wake-up: sense the temperature, receive the beacon and send the data. (d) Device wake-up: sense the light, which takes more time than the temperature, receive the beacon and send the data.

forwarded packets stays almost the same.

8.2.1 Delay

We have used `ping` on the gateway of the GreenNet to evaluate the packet RTT (Round Trip Time) and packet loss in a simple set up network (*cf.* Figure 8.3b) configured with parameters (BO, SO) = (5, 2) (BI around 500ms). The `ping` command

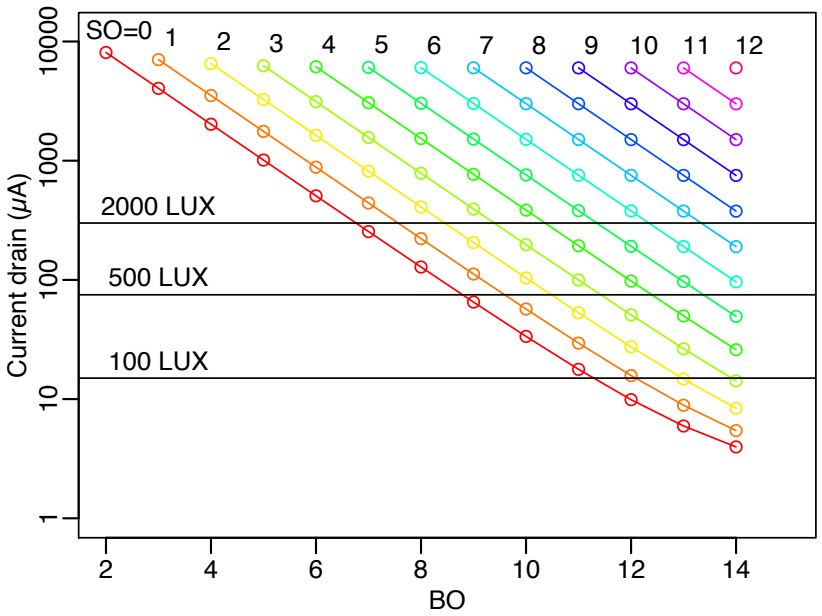


Figure 8.2: Current consumption of a router for different (BO, SO) combinations. The router and its radio are active during the incoming and outgoing superframe periods. The LUX lines correspond to the harvested current for a given light intensity (24h a day). The router may operate autonomously for the (BO, SO) parameters corresponding to a point below a LUX line.

used the interval of 1s, 2s, 3s and 5s to vary the traffic load. Table 8.3 presents the statistics returned by ping that show good delay performance of the network when the network can support the traffic load. For the increased load, we can observe higher packet loss rates, which comes from attaining network capacity. Our implementation of the sink imposes a limitation of the traffic sent to sensor nodes: to avoid overflowing the network, the sink gets only one packet from the gateway after its active period.

8.3 Harvested GreenNet Testbed

Figure 8.4 shows the battery voltage level of devices operating in the dark while sending temperature data every 4 minutes (0.01% duty-cycle). The figure provides data of the first 53 days.

When a node benefits from periodic light, given the duty-cycle, it can be energy balanced. In the following experiment, we measure the battery voltage of three devices that send temperature data every 4 minutes during 78 days (*cf.* Figure 8.5). A light intensity of 90 Lux was available from 8AM to 9PM except during several weekends when there was no light at all. We can observe that during the first 3 days, the battery voltage level becomes stable around its working level that depends on the battery internal resistance and then, it enters a stable phase. We can note a fast recovery during day 18, after the dark period, that demonstrates the capability of the system to

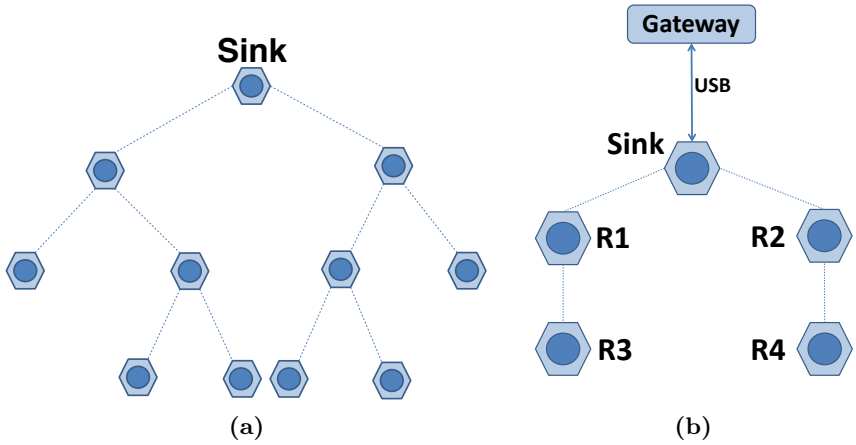


Figure 8.3: Testbed with 11 nodes and delay measurements topology.

Table 8.2: Wake-up duration (min/max) and duty cycle

	Coord. + device (FFD)		**packets**
	Min	**Max**	(loss)
E-OFF disabled	1018,6s (13.8%)	1345,5s (18.3%)	74500 (< 0.1%)
E-OFF enabled	249,3s (3.4%)	968s (13.1%)	74611 (< 0.1%)

support dark periods and the level of the intake energy greater than the energy needs of devices for their operation. If the operation had required more energy, the system would have needed several days to recover.

Energy balance depends on the available light intensity and suitable duty cycle. Table 8.4 provides the estimated values of BI (and thus BO) required to achieve energy balance based on the measured energy consumption of nodes under a given light intensity (we assume a light source active 6 hours a day and SO=0).

Figure 8.6 presents the energy balanced operation of nodes (1 device and 2 coordinators) during a long run of 14 days with a light profile shown in Figure 8.7: 9-10 hours of light per day at 2500-3000 lux. The set up of this experiment included two coordinators associated with the sink and a device associated with Coordinator 1. Nodes use (BO, SO)=(11, 0) as parameters, which results in a duty cycle of 0.1%. All 3 nodes generate a measurement of light every 4 minutes. The results demonstrate the capability of GreenNet nodes (devices as well as coordinators) to operate in a sustained way when only powered by the energy harvested from the photovoltaic cell. Even if the coordinators are energy balanced, their batteries discharge to levels lower than the recommended 90%, which shows that they would require a battery with a larger capacity to maximize the number of rechargeable cycles as mentioned previously.

Table 8.3: Network set up and the results of ping. Parameters: (BO, SO)=(5, 2), 600 ping packets per node.

Int Dev	Node	RTT Min (ms)	Avg	Max	Dev	Loss(%)
5s	R1	529	1002	2360	297	0.3%
	R2	528	969	2283	316	0.2%
	R3	1065	1469	3134	291	0%
	R4	1018	1514	3393	349	0%
3s	R1	532	1238	2414	347	1.2%
	R2	530	1151	3346	355	0.7%
	R3	1169	1918	3440	352	0.5%
	R4	1021	1559	3179	355	1.2%
2s	R1	1112	4613	7204	1133	1.6%
	R2	579	5004	7550	1167	6.3%
	R3	1446	5314	7853	1111	4.1%
	R4	1081	5507	8349	1175	5.1%
1s	R1	2287	5723	7334	518	62%
	R2	919	5719	7130	610	49%
	R3	2099	6187	7409	541	45%
	R4	1675	6278	8174	681	47%

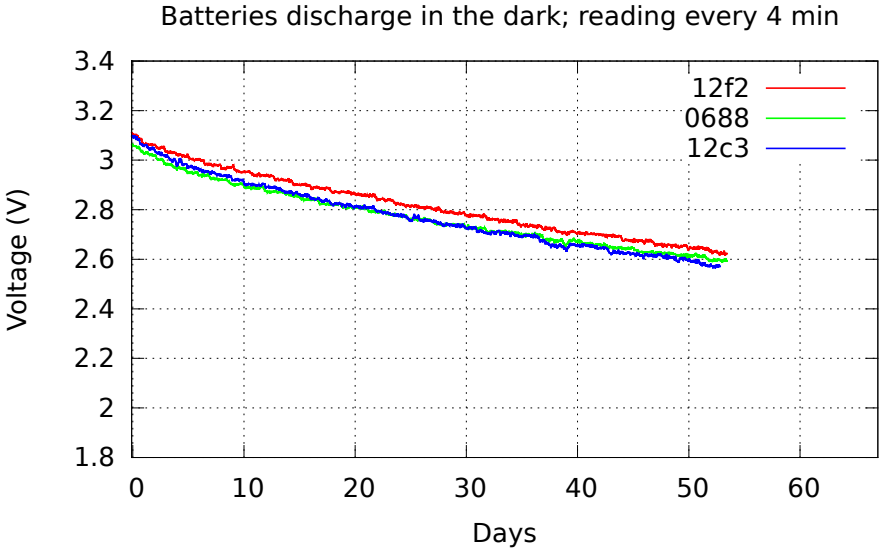


Figure 8.4: Battery discharge process in darkness for devices sending temperature data every 4 minutes (duty-cycle of 0.01%).

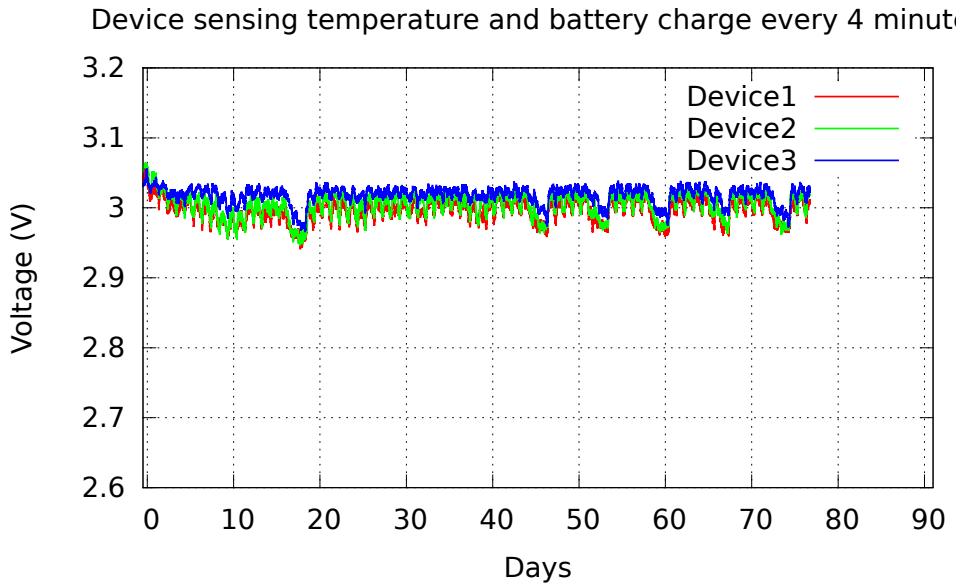


Figure 8.5: Harvested leaves sending temperature every 4 minutes during 78 days under a light of 90 lux for 13 hours/day; battery is stable around 3V.

Table 8.4: Minimum BI in ms (BO) for a given light intensity (6h/day) required for energy balance, SO=0.

lux	2500	1300	700	400	200	100
Device	7864(9)	15729(10)	31457(11)	62915(12)	125829(13)	251658(14)
Coord.	31457(11)	62915(12)	125829(13)	251658(14)	N/A	N/A

8.4 Conclusions

This chapter detailed an evaluation of the GreenNet platform running the stack we developed and implemented. The performance is measured in terms of latency and protocol overhead, but also in the perspective of global energy efficiency. There is a tradeoff between, on the one hand, how much current draw there is during the active period and, on the other hand, how much the platform gathers from the photovoltaic cell. The tests include nodes that are set up as simple leaf devices but also coordinators, thus with a higher duty-cycle. Whereas these latter nodes require more light than the leaf nodes, they are capable of continuous sustainable operation for an unlimited time since energy balance is met. At a greater time scale, a battery with a slightly larger capacity would allow them to run with either a greater duty cycle or greater durability.

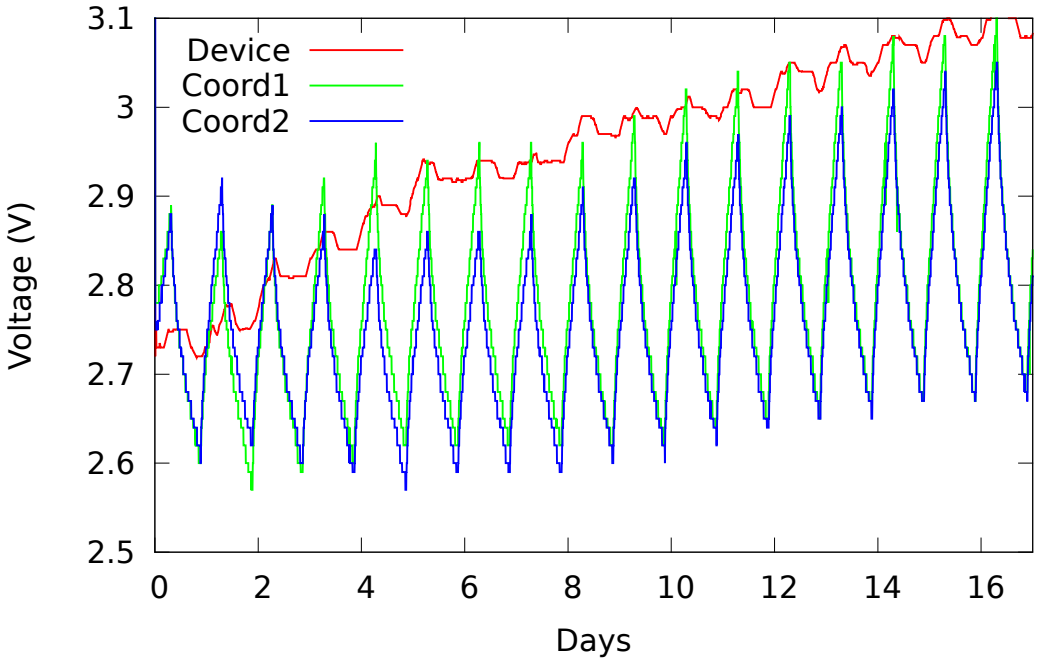


Figure 8.6: Battery level of harvesting nodes that use (BO, SO)=(11, 0) as parameters. All nodes send a measurement of light every 4 minutes.

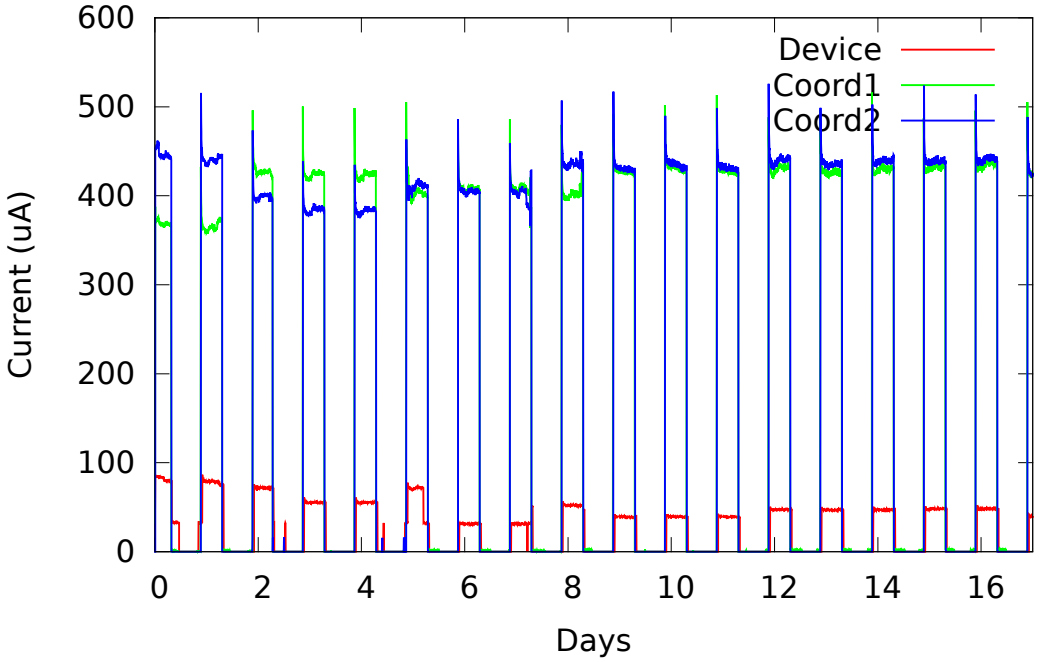


Figure 8.7: Harvested current from the photovoltaic cell

Chapter 9

Towards Channel Hopping

Contents

9.1 Introduction & Motivation

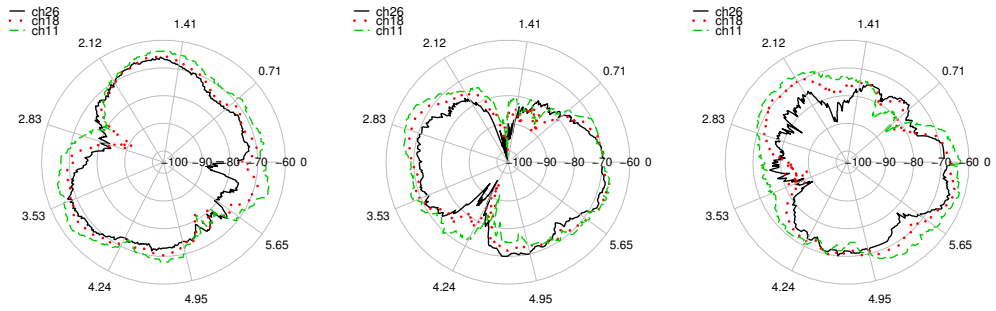
There has been a lot of research that points out the benefits of multichannel transmissions. We confirm this aspect with experimental measurements and point out a new reason to exploit channel diversity that is not usually put forward in the literature. We also present a method to alleviate multiple channel usage in GreenNet.

9.2 Experiments

9.2.1 Outdoor radiation pattern

The first outdoor experiment takes place on a large empty parking lot with the receiver node in the middle, one meter above the ground and the sender on a robot that circles around the receiver. The constant distance of 1 meter between the two nodes is maintained using a string between the center of the circle and the robot. Each experiment corresponds to a different position of the sender (flat, vertical pointing up and then, laterally).

Fig. 9.1 shows the average of RSSI (Received Signal Strength Indicator) for 20 packets at each robot position. We can observe that signal strength notably varies with the aspect angle. The pattern in Figure 9.1b is expected—it corresponds to a typical dipole radiation pattern and, in passing, validates our measurement method. What is less expected, is the difference of up to 10dB from one channel to another in many directions. Furthermore, channel 11 has a higher signal reception strength in many directions.

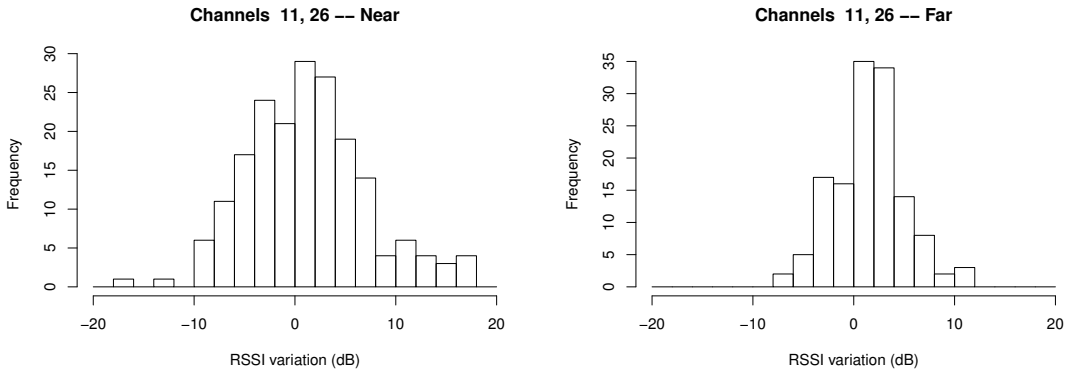


(a) Flat position **(b)** Vertical pointing up **(c)** Vertical lateral

Figure 9.1: Three RSSI measurements (dBm) on three different channels: the receiver is in the center at different positions and the sender goes around.

9.2.2 Indoor measurements

Indoor conditions are different from those observed outdoor with multiple paths even when a sender and a receiver are in the same room. Our experiment involves a fixed transmitting node and a mobile receiver on a robot that moves away from the sender by steps of 1.5 cm, which is well below the wavelength. In each position, the sender transmits a burst of 20 packets on three different channels, then the robot moves to the next position.



(a) **(b)**

Figure 9.2: RSSI variation when changing from channel 26 to channel 11. Near: transmitter in the same room as receiver, the distance between the sender and receiver is a few meters. Far: receiver outside in the corridor, no direct line of sight, over 3 meters distance.

We are interested in the so called variation or the absolute difference of RSSI when changing from one channel to another channel in the same position.

When analyzing the variation of RSSI we witness notably more variations of RSSI at a lower distance than further away (Fig. 9.2), whereas multipath fading alone would

lead to the opposite: in the same room, less multipath fading because of a greater coherence band. So there needs to be another explanation, the changing radiation patterns are indeed a good candidate reason.

Fig. 9.3 presents the number of received packets in each position along with the RSSI values. The sensibility of the receiver appears to be at -105 dBm: below this threshold, the Packet Reception Ratio (PRR) drops. The position of the receiver has a major impact on RSSI and consequently on PRR.

We have performed two runs to assess the stability of the results. The RSSI plots follow similar trends, but one would be hard-pressed to reach this conclusion just by looking at the number of received packets. We note that, in general, channel 11 gives better results than the other channels, which reflects the overall better gain observed for this channel outdoor. Nevertheless, there are positions, *e.g.* around step 700 at which the other channels give much better results, more pronounced in the case of the first run.

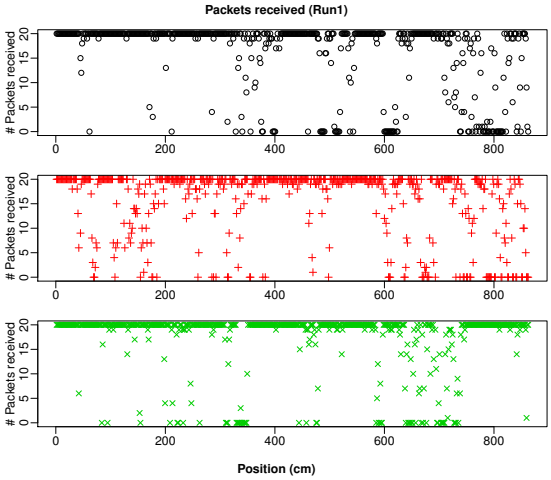
So the results for outdoor measurements provide an alternate explanation to the results obtained indoor reported in the literature that solely point to frequency selective multipath fading [90]. Since the antennas change transmission/reception patterns from one channel to another, it is not surprising to witness major differences in the recombination at the receiver in the indoor environment: the channel is most probably frequency selective, but multiple paths also get different radiation power shares.

To evaluate the advantage of using multiple channels simultaneously, we present the Packet Reception Ratio on all 3 channels (*cf.* Fig. 9.4). We can see that there is hardly any position where no packets are received.

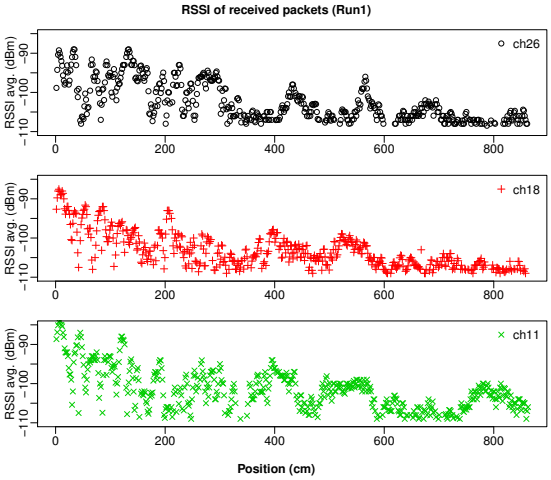
The results suggest that we can benefit from better transmission performance by taking advantage of channel diversity—using multiple channels at the same time. However, the problem is how to organize the operation of nodes that already have their place in the cluster-tree topology. Actually, we need to enhance the 802.15.4 beacon-enabled mode to notify other nodes with the channel to use, which is the principle of the MRR (Multi-channel Round-Robin) proposal.

9.3 Multi-Channel Round-Robin

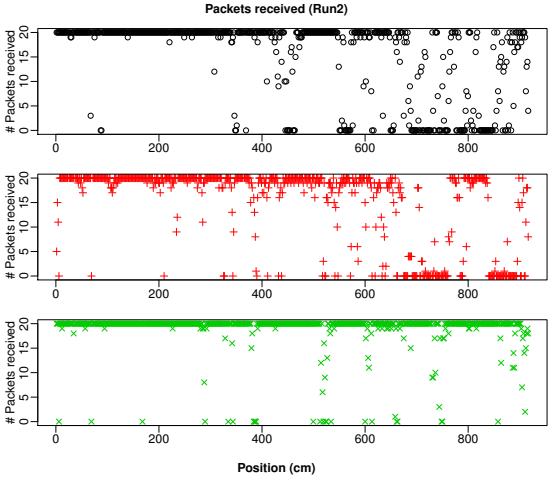
The MRR scheme consists of using available channels in a round-robin way by sending beacons to invite associated devices to send packets on the same channel on which they receive the beacons. We propose to insert additional active periods operating on different channels during the inactive period of the standard beacon interval as defined in 802.15.4. The standard beacon interval is delimited by beacons sent on the main channel and a node can send beacons on other channels to notify the beginning of additional active periods. We propose to choose the instants of sending additional beacons at random, but with caution to reduce interference: before sending a beacon, the node listens to a given channel to detect possible on-going transmissions. If it is the case, the node chooses another instant or changes the channel. If the channel is not used, the node sends a beacon. Fig. 9.5 illustrates the principles of MRR in the case of sending 3 additional beacons on 3 different channels (the Beacon Order BO of 6 on a single channel is equivalent to BO=4 on all 4 channels).



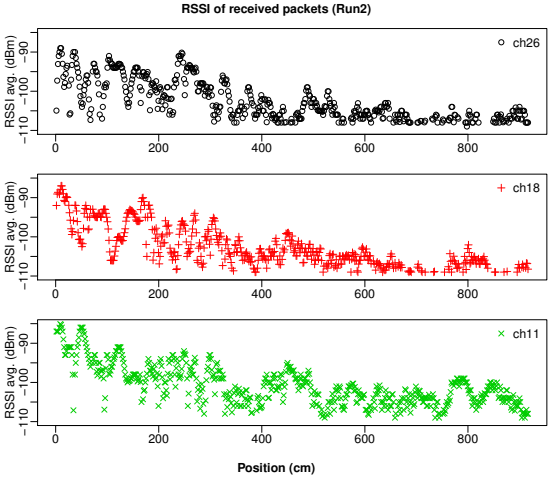
(a) Number of received packets for each channel at each position.



(b) Average RSSI for each position and channel

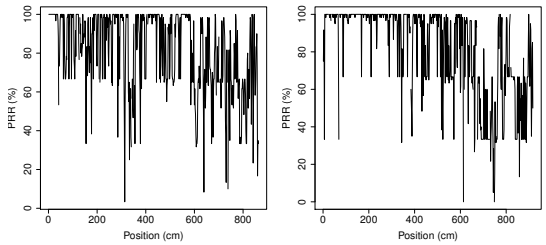


(c) Number of received packets for each channel at each position.



(d) Average RSSI for each channel and position.

Figure 9.3: Number of received packets on different channels in different positions as well as RSSI for each successfully received packet. RSSI varies a lot from one position to another, which can influence the transmission quality on a specific channel.



(a) Run 1 **(b)** Run 2

Figure 9.4: Packet Reception Ratio (PRR) on all channels. There is only a couple of positions at which no packet is received.

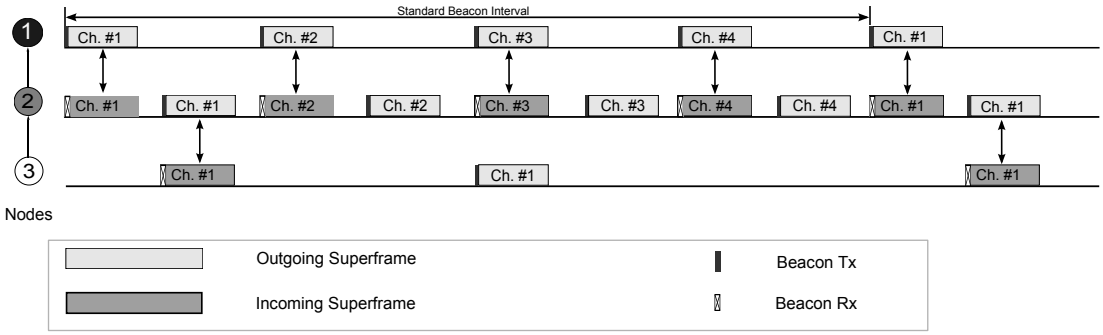


Figure 9.5: Multi-channel Round-Robin. During a standard beacon interval delimited by two beacons on the same channel (Channel 1), a node sends additionnal beacons on different channels (Channels 2, 3, 4) to take advantage of channel diversity. Node 1: Sink; Node 2: multi-channel coordinator; Node 3: single-channel coordinator.

Note that MRR is backward compatible with the standard—a device that does not implement MRR can still associate and communicate with other nodes on a single channel. In other words, we duplicate the superframe on several channels to improve packet reception ratio while keeping the standard compatibility on each individual channel.

9.4 Determining *StartTimes* on Multiple Channels

The aim of each node is to send additional beacons on the same set of channels, so we need to determine their instants as time offsets since the received beacon with the parameter *StartTime*. We define a *superframe slot* as an active period of SD duration placed at multiples of SD (SD is the superframe duration, *cf.* Fig. 2.6). We propose a random choice of slots for sending additional beacons. After an initial selection of slots, a joining node first listens during those slots to check if there are other nodes that send beacons at the same time. The goal is to avoid collisions. If some activity is detected, this particular slot is marked as not available and another

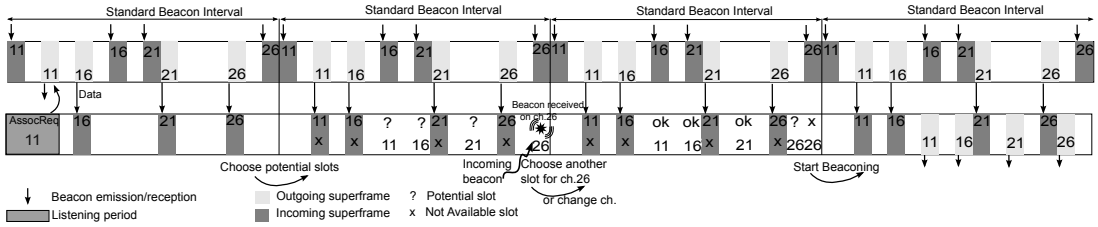


Figure 9.6: Distributed slot selection for nodes joining the network. Random selection of slots and avoidance of overlapping active periods.

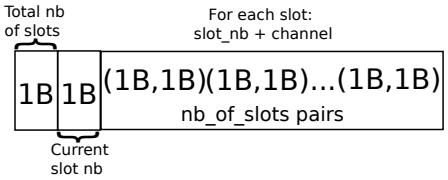


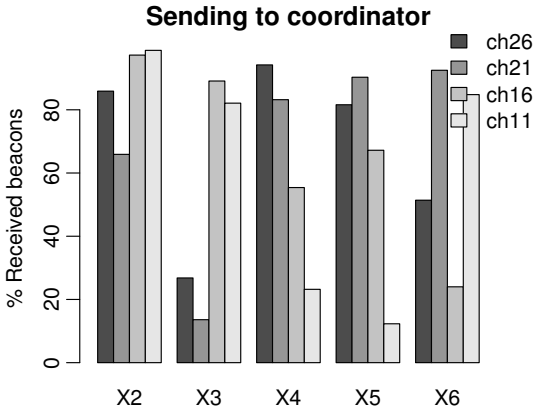
Figure 9.7: Extra information included in the beacon payload.

one is chosen (*cf.* Fig. 9.6). The slots of the coordinator of the node are automatically marked as not available. Such a distributed solution is suitable for networks with sub-cluster trees having different parameters of superframe durations and beacon intervals, giving a high degree of flexibility to schedule the instants of sending beacons.

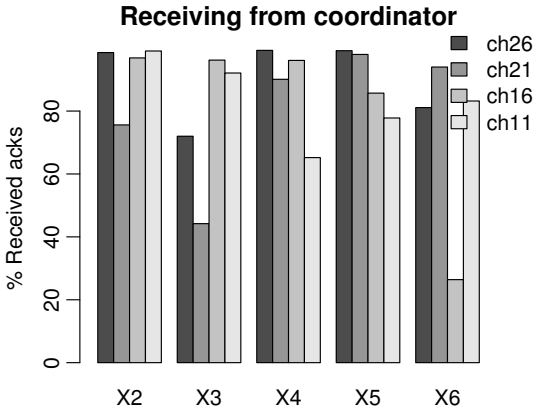
A node includes the information on the additional beacons sent, their slots, and the channel used in the beacon frames (*cf.* Fig. 9.7). In this way, a single beacon is enough for a node that tries to join the network to obtain the information for synchronizing with the other nodes that send beacons on different channels. This mechanism allows for great flexibility in choosing the beacon slots. The same slots can be used for different channels and also the same slot can be reused on the same channel if no other coordinator uses it for sending beacons.

9.5 Dealing with Multicast Frames

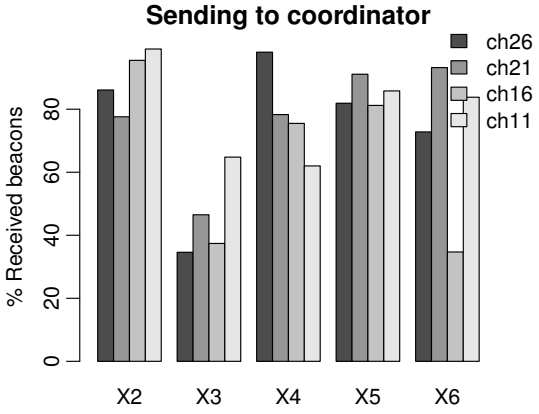
Multicast packets need special handling if legacy nodes are present as they only listen to the beacons on a single channel. In this case, if we send a multicast packet on a different channel than the one on which a single-channel node operates, it will be received by all other nodes, but not by the legacy one. To overcome this issue, we propose to change multicast frames into a series of unicast frames for the associated devices, similar to the solution presented in Section 6.3.3, which guarantees that the frame will be received by all the nodes, even when they wake up for a subset of the beacons. When the single-channel node wakes up, it retrieves the multicast frame using the usual unicast transmission procedure: it gets notified that it has a pending frame in the beacon and requests its transmission.



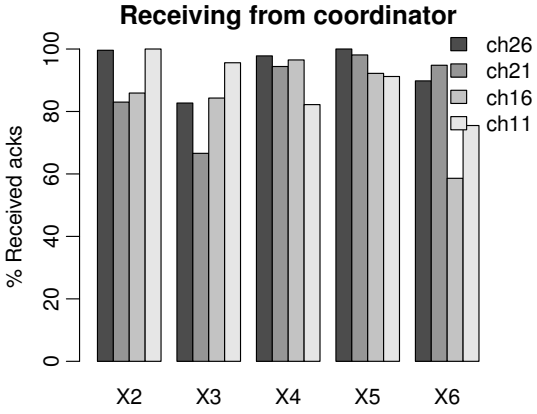
(a) Successfully received beacons (Run 1)



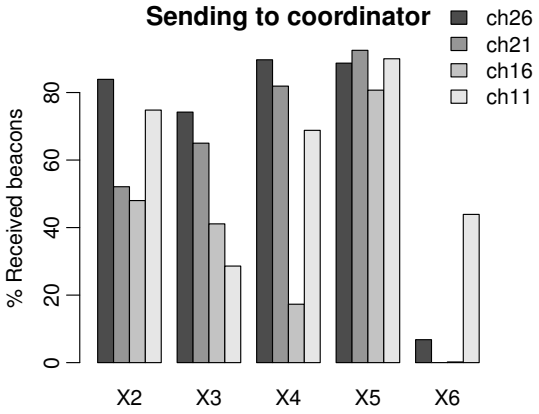
(b) Received acknowledgments for unicast packets (Run 1)



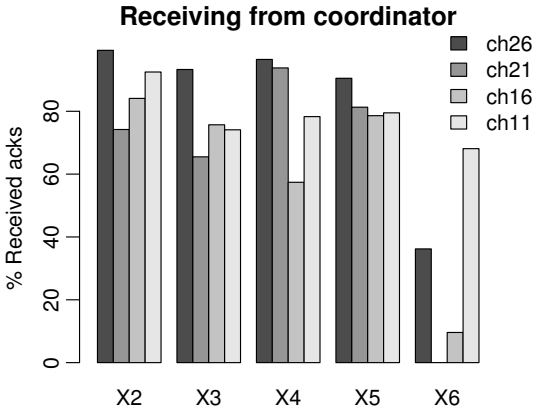
(c) Successfully received beacons (Run 2)



(d) Received acknowledgments for unicast packets (Run 2)



(e) Successfully received beacons (Run 3)



(f) Received acknowledgments for unicast packets (Run 3)

Figure 9.8: Multi hop experiment using 4 channels for sending beacons (downward traffic) and unicast packets (upward traffic). Measurements of six nodes in a chain topology.

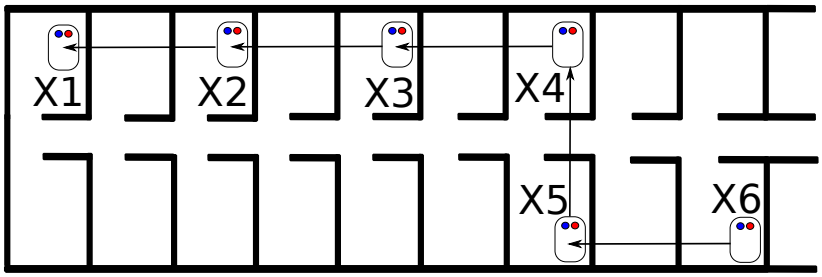


Figure 9.9: Node placement in the experiments.

9.6 Evaluation and Performance Results

We have implemented the proposed enhancement of beacon-enabled 802.15.4 in the GreenNet protocol stack [86]. Fig. 9.8 shows the results obtained from a 6 node testbed set up in our lab. Each node is in a different room at a distance of typically 2 offices (or 10m) between adjacent nodes (*cf.* Fig. 9.9). In this set up, even though we use the maximum transmission power (6 dBm), many packets are lost, but using 4 channels (11, 16, 21, 26) evenly spread along the whole ISM 2.4GHz band, there was always at least one of them that obtained PRR over 40%.

Recall that beacons are broadcast frames so they are not retransmitted in case of collision. Moreover, they are transmitted without any clear channel assessment. To ease the comparison with unicast frames, we have configured no retransmission for unicast packets when the acknowledgment is not received, so nodes only attempt to transmit each packet once.

Each run lasts long enough for the transmission of 15,000 beacons on each channel. As for unicast packets, depending on the node placement in the chain topology and the channel number, the number of acknowledged frames or missed ones varies from 200 to 15,000 (except for two cases in which there were only about 20 frames on a particular channel between two nodes). The unicast frame size is 22 bytes in Run 1 and 2, and 88 bytes in Run 3. The beacon frame size is 61 bytes.

First, we can observe in Figure 9.8 that depending on the channel, the link quality notably fluctuates. Moreover, none of the 4 channels we use gives satisfactory results (*e.g.* receive over 50% of beacons on the same channel on all the nodes). So, if we use only one single channel for all communications as defined in the standard, nodes obtain poor performance—even if we were able to select the best channel after carefully assessing the transmission conditions, it turns out that the channel does not result in a consistently good PRR over all hops. Moreover, from one day to the next (Run 1 to 3), the conditions notably change, so we would need to start the selection process of the best channel once again.

9.7 Conclusions

Our experimental results show that taking advantage of multiple channels is paramount to improve the quality of communications that use publicly available radio bands. Indoor environment often varies (people move around, doors open/close etc.) so the radio channel quality changes in a unpredictable way. The best way to fight fading and interference is to use multiple channels that span the whole ISM band.

In this chapter, we proposed MRR (Multi-channel Round-Robin), a scheme that uses multiple channels in a round-robin way. A node places additional active periods operating on different channels during the inactive period of the standard beacon interval. Active periods start with beacons sent on different channels. A joining node can transmit a data packet on the same channel as the received beacon. We have also proposed a random allocation scheme to choose the instants of the additional active periods to avoid beacon collisions. MRR integrates perfectly well with beacon-enabled IEEE 802.14.5 by using beacons on multiple channels to invite nodes to use a given channel for communication.

The evaluation of the proposal through measurements on a real-world multihop testbed deployed indoor shows that the scheme results in better PRR.

A simple and efficient extension to MRR would be to change the channel to avoid using the one on which beacon transmission results in poor performance. The change needs to be done with caution though, as not all associated devices perceive the same link conditions, so the change requires an explicit feedback from all nodes. A device can simply just skip beacons sent on a channel that exhibits poor performance. Along the similar lines, a node may avoid sending unicast packets on particular channels if the PRR is low.

Chapter 10

Engineering side of the thesis

Contents

Since my PhD was financed by ST Microelectronics, a good share of my work was to participate actively in GreenNet project: implementing, testing, debugging and validating different features. GreenNet aim is to reach autonomy with an IPv6 stack. Hence, all the improvements mentioned in the previous chapters were implemented and tested on GreenNet nodes. This gives a higher added value to the proposed ideas.

Being a new platform with its hardware related particularities, it was not always straightforward to implement and test a feature. The time needed to overcome a problem in the software running on the nodes is much higher compared to a simulator.

As any real implementation, developing a classical IPv6 stack that properly runs on our specific hardware requires good engineering skills and represents quite a lot of work. In this context, it is easily understood that implementing and validating new proposal brings not only coding work to actually run the proposed improvement, but also debugging work: in real environment, a slight change of conditions can results in really different outcomes. The following section will only review the situations demanding the most engineering work we met while either implementing the classical stack, or testing a new proposal.

First of all, a big limitation is the scale of the network. Most of the tests were done using up to 5 nodes on a desktop environment. Even in that simple case, the fact of flashing the nodes, waiting to boot and synchronize takes tens of seconds. Once the network runs, in order to understand the network's behaviour, we used a text log file: parsing this file to properly understand how nodes perform certain tasks also takes time.

Due to the fact that our stack is optimized, even printing messages can influence the network behavior, since it requires node's resources. During the time the debugging messages are sent on the serial line, the CPU is busy and cannot perform other tasks. Because 802.15.4 standard has strict timings, a routine of as low as 5 ms could already break the synchronization and the expected execution. To give an example, it often happened that just adding a single letter of debugging in certain regions of the code, just changed completely the behavior of the node. Identifying such a bug does not

always come to mind first. The solution that we later implemented was to buffer the log messages (with a time stamp) and send them on the serial line after the active period ends. In this way the scheduling inside the critical part is not affected.

A second issue we faced were timings handling. Since our nodes are highly constraint, timing margins are reduced, which unveils new issues. For instance, when we tested our nodes with SO=0 (meaning SD is 15 ms) we expected them to be able to associate with a coordinator, considering we have only one node trying to associate at a single moment. Association means that beacon is received by the node after which it sends an association request message (which is less than 40 bytes) and during the next active period the node will have to send a data request and the coordinator replies with a packet when it either accepts or refuses the association. The problem we encountered was that this exchange was not done in a sufficient time: at the end of the 15 ms the nodes would turn off their radio. What surprised us was that the on-the-air time of a packet is at maximum 4 ms (based on the size of the frame) which should give more than enough time to make an exchange of packets.

First of all, as it was a timing issue, using debug messages on the serial port was not a solution. To investigate this, I took an oscilloscope and tried to benchmark different sections of the code to see where was the extra time lost. My studies revealed that SPI configuration used in the frame exchange between CPU and radio was not proper. In fact, each byte was sent individually instead of sending a bulk of bytes. Hence, instead of taking less then 1 ms to send a frame, the SPI transfer took 5 ms. Having solved this bug, the improvement in performance appeared immediately: no problem in using the minimum active period for nodes to associate and send/receive data packets. Moreover, this had an impact on any SO value, meaning that the capacity had improved for any other active periods.

10.1 OpenWSN

The goal of the GreenNet project was not only to provide a complete solution (hardware + software) for the Internet of Things but also a generic hardware on which one could use different software implementations.

We tested the feasibility of this assertion by porting the OpenWSN project on the GreenNet node.

The porting consists in reaching the abstraction layer corresponding to what OpenWSN expects. This was done for CPU, radio and timers. Due to very tight schedule of events inside a slot (slots of 10 or 15 ms) it is impossible to observe the interrupts and the behavior of the node while using serial output messages. In order to analyze our board behaviour at different instants, we used logic analyzers, that are very small compared to an oscilloscope, which provide several pins to plug into any platform that has free GPIOs. The analyzer is plugged via an USB cable to the PC and through a very simple interface, one can see whether a particular GPIO is low or high. Associating up to 8 GPIOs to different events in the code we were able to implement and debug the required functionalities in a much faster way than using debugging messages (or oscilloscope).

We achieved to have a working version of the OpenWSN and participate to the first

PlugTEST on 6TiSCH where we were able to interconnect with several platforms that either implemented the same stack (OpenWSN) or different stack (like Contiki) but with the same TSCH protocol.

Through this implementation, I realized that logic analyzer not only speeds up development and debugging but it also gives a visual highly accurate timing execution.

One important detail to mention is that even though we can enter and debug the nodes step by step, this task becomes impossible when there is more than one node in the network since they become dependent of each other due to synchronization. Having a logic analyzer enables a programmer to see, for example, two nodes that try to synchronize or are already in sync, on a single screen with their different events in parallel. One can easily see the differences between different events, like starting the transmission of a packet and the end of receiving it on the other node.

However, this problem will still remain when larger networks are deployed and physical access to certain nodes is not always possible.

10.2 WALT

To be able to test networks that have more than a few nodes and are spread around a building, DRAKKAR team put in place a testbed composed of RaspberryPis and GreenNet nodes. One or two GreenNet nodes can be connected to a RaspberryPi that is used to control the node via USB (start/stop, reset) and collect debugging information. The access to the network is done via the server that is managing the RaspberryPis and it also collects the logs from each RaspberryPi with a time-stamp. Clocks are synchronized via a Network Time Protocol server. So even with tens of running nodes we get a global image of the network's behavior. We used this platform to evaluate the proposed scheme of Multi-Channel Round-Robin.

10.3 STLink

An important part of the GreenNet node is its USB interface. It allows flashing and debugging by using open-source software. This is possible thanks to the use of standard STLink interface. We have contributed to this interface by developing two serial interfaces that are presented to the operating system of a PC. Using these emulated serial interfaces we can send debugging and log information from the node to the PC. One serial is connected to a physical serial line between the microcontroller managing the USB and the low-power MCU, whereas the other interface is connected to a physical SPI connection between the two MCUs.

Because we needed special behavior for WALT testbed for the GreenNet nodes we were using one of the USB serial interface to control the low-power MCU via the USB controller. We wanted to enable/disable the serial line as well as resetting the low-power MCU.

Having worked on the customization of the USB firmware I also had the task to implement the different requirements for the WALT testbed.

10.4 Conclusions

Working on the thesis in collaboration with STMicroelectronics helped me a lot in doing the first steps in the industrial world and in professional environment. It not only helped me gaining experience and knowledge on technical side, but also the possibility to work on a big project while creating a commercially available product.

Part III

Conclusions and additional discussions

Chapter 11

Conclusions, reflections and future perspectives

Contents

11.1 Conclusions

The thesis is part of GreenNet project at STMicroelectronics. The project's objective is to have a new generation of harvesting platform and autonomous nodes able to contribute to the revolution called Internet of Things. The GreenNet project started with the development, internally, of the entire hardware platform and also of the whole protocol stack. Regarding the software part, the goal was to reuse as much as possible open-source projects and so we turned to the ContikiOS as operating system, as it already included a complete IPv6 protocol stack.

Most of my work involved in software R&D was to enable the IEEE 802.15.4 beacon-enabled mode on the GreenNet platform and to make it as efficient as possible. The first goal of the development was to have a star network topology and after, to have a multi-hop solution. Along these milestones, we made further energy optimizations, like the early-off mechanism for coordinators, skipping the beacons for very low duty-cycle devices, reducing the number of sent beacons if no devices are associated. On top of this we proposed a multi channel solution that is backwards compatible to single-channel mode. To validate our proposals we implemented them and tested them on GreenNet node. We moreover checked our proposals were not bringing new limitations to harvesting networks.

11.2 Cross layering

As much as we wanted to keep concepts separated one from each other, for example between applications and radio duty-cycling, it turns out this principle is hard to comply to in many cases.

Trying to separate HW and SW world reveals unoperational, since, most of the time, proper abstraction can no be perfect. In non tightly constrained environment,

the limitation of hardware abstraction is, in most cases, non visible. However, in our highly constrained environment, these side effects become preponderant.

The most pertinent example is the use of CO2 sensor. Due to the very long period to acquire data, depending on the schedule of the radio/superframes, it can happen that the sensing overlaps radio events. So implementing this sensor should be done using a mechanism of consumer/producer to not block the entire system while sensing.

Another example is the "freshness" of data sent by the node. If for example we have a node that is waking up every 4 minutes to send a temperature, the sent values could have differences that depend on the exact moment of sensing. As the user expects a relevant value it is not the same if the node does the sensing before receiving the beacon or after it sent a value. In the latter case, the measurement is used only 4 minutes later. Of course the best way is to sense and send then the data as soon as possible. The problem is that once in the radio routine, there might not be enough time to sense and send the data. Thus, the device should anticipate how much time before the moment it sends a packet it should wake up to sense. For a temperature the sensing time can be only 5ms before the beacon arrival but for a CO2 it can take up to several seconds. In order to optimize it, the layer that is doing the scheduling should know about radio timings (beacon, slotframe duration etc.) along with all the application demands (which sensors and how often should an information be sent) such that it creates an optimal time scheduling. Such a scheduling gives relevant sensing information and reduces the wake-up time of the node. This is all true for any radio duty-cycling, TSCH, beacon-enabled etc.

Regarding MAC and routing layer, once devices are synchronized with one node which is also the route towards the sink, it makes sense to simplify routing and consider the node as default route.

In beacon mode, we can imagine that one device is tracking two or more coordinators. In this case a device should elect a preferred parent based on different metrics (lowest delay towards the sink or link quality, stronger signal, etc.). The advantage of such a solution is that in case the link with the preferred coordinator is lost, the node is able to change immediately, without performing another scanning. We should note that, *a priori*, the device should have already performed the association protocol with the future coordinator. Whether this solution can really bring a better reliability or improvements should be analyzed case by case. If the nodes are stable, the extra energy spent to track more beacons than necessary might not compensate the sought benefits.

11.3 Impact of autonomous nodes

Having small devices that are autonomous, not only from auto configuration point of view, but also from energy side, can have a big impact on our everyday life: no need to worry about charging or changing the batteries.

We can imagine a smart home that reduces the energy bills all by itself and at the same time preserving the comfort we are expecting everyday. The endless number of devices and interconnections between them is just limited by our imagination. The control of the "things" we interact with should nevertheless be in the hands of any user.

A person that does not have the technical knowledge about how nodes work, should

be able to make his/her network act in a certain way. This is a must for the success and mass deployment of Internet of Things.

However, before this explosion of billions of devices, engineers need to render it not only ”user-friendly”, but also completely secure: privacy is indeed a must, dealing with private personal data must always be done with care.

11.4 Future perspectives

Targeting a completely autonomous solution in terms of energy and with very low duty-cycles should be done using a beacon enabled solution. Synchronization is done while receiving the beacon and data exchange takes place immediately afterwards. If we need to relay the information during multiple hops, the coordinators can also be energy harvesting based on the estimated traffic to relay. In order to reduce the delay of forwarding the packets all the way to the sink, after association we can schedule the sending of the beacon just before the wakeup for our coordinator. This means that even for several hops when nodes are waking up each 4 minutes we can still have a latency of only a few seconds upwards. Of course, the trade-off is that the latency for downward traffic is then of several minutes.

Moreover, all the problems that we encountered in developing our protocol stack for GreenNet are the same as for other solutions that are being proposed, like TSCH or BLE. Considering this, I think that the community should propose a global solution that merges the various implementations to finally reach a clear and explicit standard. In this way anyone who wants to provide a platform, should be able to deliver a product that inter-operates with any other device complying with the standard.

When it comes to long range solutions, these will most probably live along side with LR-WPAN because they have the advantage of covered area and/or distance. An interesting future research would be how to integrate the two solutions into a single one such that the requirements of the network are satisfied but with a minimum energy consumption and a high quality of service.

Publications

[c1] Energy-Efficient Multi-Hop Broadcasting in Low Power and Lossy Networks. Chi-Anh La, Liviu-Octavian Varga, Martin Heusse and Andrzej Duda. *Proceedings of the 17th ACM international conference on Modeling, analysis and simulation of wireless and mobile systems*, MSWiM. ACM, 2014.

[c2] GreenNet: an Energy Harvesting IP-enabled Wireless Sensor Network. Liviu-Octavian Varga, Gabriele Romaniello, Malisa Vucinic, Michel Favre, Andrei Banciu, Roberto Guizzetti, Christophe Planat, Pascal Urard, Martin Heusse, Franck Rousseau, Olivier Alphand, Étienne Dublé, and Andrzej Duda. *IEEE Internet of Things Journal*, IoT-J. IEEE, 2015

[c3] Improving Robustness of Beacon-Enabled IEEE 802.15.4 with Round-Robin Channel Diversity. Liviu-Octavian Varga, Martin Heusse, Roberto Guizzetti and Andrzej Duda. *IEEE International Conference on Sensing, Communication and Networking*, SECON IEEE, 2016 (submitted for publication)

[c4] Why is Frequency Channel Diversity so Beneficial in Wireless Sensor Networks? Liviu-Octavian Varga, Martin Heusse, Roberto Guizzetti and Andrzej Duda. *Wireless Days Conference*, WD 2016. (submitted for publication)

Bibliography

[1] Jennic. http://www.jennic.com/files/support_files/JN-AN-1079CoexistenceofIEEE0802.15.4InThe2.4GHzBand-1v0.pdf. 1, 19

[2] LoRa. https://www.semtech.com/images/mediacenter/collateral/ism_sg.pdf. 1, 32

[3] Chi-Anh La, Liviu-Octavian Varga, Martin Heusse, and Andrzej Duda. Energy-Efficient Multi-Hop Broadcasting in Low Power and Lossy Networks. In *Proceedings of the 17th ACM international conference on Modeling, analysis and simulation of wireless and mobile systems*, pages 41–50. ACM, 2014. 2, 66

[4] If This Then That. https://ifttt.com/. 9

[5] D. Estrin, R. Govindan, J. Heidemann, and S. Kumar. Next Century Challenges: Scalable Coordination in Sensor Networks. In *Proceedings of Mobicom*, pages 263–270, Seattle, WA USA, 1999. 10

[6] Hill, Jason and Szewczyk, Robert and Woo, Alec and Hollar, Seth and Culler, David and Pister, Kristofer. System Architecture Directions for Networked Sensors. *ACM SIGOPS Operating Systems Review*, 34(5):93–104, 2000. 10

[7] J. S. Heidemann, F. Silva, C. Intanagonwiwat, R. Govindan, D. Estrin, and D. Ganesan. Building Efficient Wireless Sensor Networks with Low-Level Naming. In *Proceedings of SOSP*, pages 146–159, 2001. 10

[8] C. Intanagonwiwat, R. Govindan, D. Estrin, J. Heidemann, and F. Silva. Directed Diffusion for Wireless Sensor Networking. *IEEE/ACM Trans. Netw.*, 11(1):2–16, 2003. 10

[9] J. Polastre, J. Hui, P. Levis, J. Zhao, D. Culler, S. Shenker, and I. Stoica. A Unifying Link Abstraction for Wireless Sensor Networks. In *Proceedings of ACM SenSys*, 2005. 10

[10] A. Dunkels, F. Osterlind, and Z. He. An Adaptive Communication Architecture for Wireless Sensor Networks. In *Proceedings of ACM SenSys*, Sydney, Australia, November 2007. 10

[11] N. Burri, P. von Rickenbach, and R. Wattenhofer. Dozer: Ultra-Low Power Data Gathering in Sensor Networks. In *Proceedings of ACM/IEEE IPSN*, Cambridge, Massachusetts, USA, 2007. 10

[12] R. Musaloiu-E., C-J. M. Liang, and A. Terzis. Koala: Ultra-Low Power Data Retrieval in Wireless Sensor Networks. In *Proceedings of ACM/IEEE IPSN*, St. Louis, Missouri, USA, 2008. 10

[13] O. Gnawali, K. Jang, J. Paek, M. Vieira, R. Govindan, B. Greenstein, A. Joki, D. Estrin, and E. Kohler. The Tenet Architecture for Tiered Sensor Networks. In *Proceedings of ACM SenSys*, Boulder, Colorado, USA, 2006. 10

[14] S. Dawson-Haggerty, X. Jiang, G. Tolle, J. Ortiz, and D. Culler. sMAP: a Simple Measurement and Actuation Profile for Physical Information. In *Proceedings of ACM SenSys*, pages 197–210, Zurich, Switzerland, 2010. 10

[15] A. Dunkels. Full TCP/IP for 8-Bit Architectures. In *Proceedings of ACM MobiSys*, San Francisco, CA, USA, May 2003. 10, 11, 41

[16] J. Hui and D. Culler. IP is Dead, Long Live IP for Wireless Sensor Networks. In *Proceedings of ACM SenSys*, Raleigh, North Carolina, USA, November 2008. 10, 11

[17] X. Jiang, S. Dawson-Haggerty, P. Dutta, and D. Culler. Design and Implementation of a High-Fidelity AC Metering Network. In *Proceedings of ACM/IEEE IPSN*, April 2009. 10

[18] B. Priyantha, A. Kansal, M. Goraczko, and F. Zhao. Tiny Web Services: Design and Implementation of Interoperable and Evolvable Sensor Networks. In *Proceedings of ACM SenSys*, pages 253–266, Raleigh, NC, USA, 2008. 10, 11

[19] D. Yazar and A. Dunkels. Efficient Application Integration in IP-Based Sensor Networks. In *Proceedings of the ACM BuildSys 2009 workshop, in conjuction with ACM SenSys 2009*, November 2009. 10

[20] M. Ceriotti, L. Mottola, G. P. Picco, A. Murphy, S. Guna, M. Corra, M. Pozzi, D. Zonta, and P. Zanon. Monitoring Heritage Buildings with Wireless Sensor Networks: the Torre Aquila Deployment. In *Proceedings of ACM/IEEE IPSN*, Washington, DC, USA, 2009. 10

[21] K. Langendoen, A. Baggio, and O. Visser. Murphy Loves Potatoes: Experiences from a Pilot Sensor Network Deployment in Precision Agriculture. In *Proceedings of IPDPS*, Rhodes Island, Greece, April 2006. IEEE. 10

[22] A. Mainwaring, J. Polastre, R. Szewczyk, D. Culler, and J. Anderson. Wireless Sensor Networks for Habitat Monitoring. In *Proc. of WSNA 2002*, Atlanta, GA, USA, September 2002. 10

[23] J-P. Vasseur and A. Dunkels. *Interconnecting Smart Objects with IP: The Next Internet*. Morgan Kaufmann, 2010. 11

[24] A. Dunkels, T. Voigt, and J. Alonso. Making TCP/IP Viable for Wireless Sensor Networks. In *Proceedings of EWSN*, Berlin, Germany, January 2004. 11

[25] M. Durvy, J. Abeillé, P. Wetterwald, C. O'Flynn, B. Leverett, E. Gnoske, M. Vidales, G. Mulligan, N. Tsiftes, N. Finne, and A. Dunkels. Making Sensor Networks IPv6 Ready. In *Proceedings of ACM SenSys*, Raleigh, North Carolina, USA, November 2008. 11

[26] J.W. Hui and D.E. Culler. IPv6 in Low-Power Wireless Networks. *Proceedings of the IEEE*, 98(11):1865 –1878, 2010. 11

[27] T. Winter, P. Thubert, A. Brandt, J. Hui, R. Kelsey, P. Levis, K. Pister, R. Struik, J-P. Vasseur, and R. Alexander. RPL: IPv6 Routing Protocol for Low-Power and Lossy Networks. RFC 6550 (Proposed Standard), March 2012. 11, 47, 71

[28] JeongGil Ko, Joakim Eriksson, Nicolas Tsiftes, Stephen Dawson-Haggerty, Mathilde Durvy, J-P. Vasseur, Andreas Terzis, Adam Dunkels, and David Culler. Beyond Interoperability: Pushing the Performance of Sensornet IP Stacks. In *Proceedings of ACM SenSys 2011*, Seattle, WA, USA, November 2011. 11

[29] Fabien Todeschini, Christophe Planat, Patrizia Milazzo, Salvatore Tricomi, Pascal Urard, and Philippe Benabes. A Nano Quiescent Current Power Management for Autonomous Wireless Sensor Network. In *Electronics, Circuits, and Systems (ICECS), 2013 IEEE 20th International Conference on*, pages 815–818. IEEE, 2013. 11

[30] IEEE Standard for Local and metropolitan area networks–Part 15.4: Low-Rate Wireless Personal Area Networks (LR-WPANs). IEEE Standard 802.15.4, 2011. 11, 17, 19, 32

[31] Mališa Vučinić, B. Tourancheau, F. Rousseau, A. Duda, L. Damon, and R. Guizzetti. OSCAR: Object Security Architecture for the Internet of Things. In *Proc. of IEEE WoWMoM, Sydney, Australia, 2014*. 11

[32] ZigBee Alliance, `http://www.zigbee.org`. 17

[33] Zach Shelby and Carsten Bormann. *6LoWPAN: The Wireless Embedded Internet*. Wiley Publishing, 2010. 18

[34] ZigBee Alliance. Zigbee. Web page. http://www.zigbee.org. 20

[35] Adam Dunkels. The ContikiMAC Radio Duty Cycling Protocol. Technical Report 5128, Swedish Institute of Computer Science, 2011. 22

[36] M. Buettner et al. X-MAC: A Short Preamble MAC Protocol For Duty-Cycled Wireless Networks. *In Proceedings of ACM SenSys*, Boulder, CO, November 2006. 22

[37] IEEE. Standard for Local and metropolitan area networks – Part 15.4e: Low-Rate Wireless Personal Area Networks (LR-WPANs) Amendment 1: MAC sublayer. Standard, IEEE SA, 2012. 24, 25

[38] K Pister and Lance Doherty. TSMP: Time synchronized mesh protocol. *IASTED Distributed Sensor Networks*, pages 391–398, 2008. 25

[39] Q. Wang, X. Vilajosana, and T. Watteyne. 6TSCH Operation Sublayer (6top). *IETF - draft-wang-6tsch-6top-00*, 2013. 25

[40] Maria Rita Palattella, Nicola Accettura, Mischa Dohler, Luigi Alfredo Grieco, and Gennaro Boggia. Traffic Aware Scheduling Algorithm for reliable low-power multi-hop IEEE 802.15. 4e networks. In *IEEE 23rd International Symposium on Personal Indoor and Mobile Radio Communications (PIMRC)*, pages 327–332. IEEE, 2012. 26

[41] Antoni Morell, Xavier Vilajosana, JosÃ© LÃ³pez Vicario, and Thomas Watteyne. Label switching over IEEE802. 15.4 e networks. *Transactions on Emerging Telecommunications Technologies*, 24(5):458–475, 2013. 26

[42] Nicola Accettura, Maria Rita Palattella, Gennaro Boggia, Luigi Alfredo Grieco, and Mischa Dohler. DeTAS: A decentralized traffic aware scheduling technique enabling IoT-compliant multi-hop low-power and lossy networks. In *Second IEEE WoWMoM Workshop on the Internet of Things: Smart Objects and Services, IoT-SoS*, 2013. 26

[43] Bluetooth SIG. Specification of the Bluetooth System v4.2. Standard, Bluetooth SIG, December 2014. 27

[44] 6lo Working Group Status Pages. https://tools.ietf.org/wg/6lo/. 27

[45] Specification of the Bluetooth System v4.0. Standard, Bluetooth SIG, June 2010. 27

[46] Specification of the Bluetooth System v4.2. Standard, Bluetooth SIG, December 2014. 27, 29

[47] SIGFOX, http://www.sigfox.com. 30

[48] LoRaWAN. https://www.lora-alliance.org/. 31

[49] Thread Group. Mesh Network for Connected Products in the Home, 2014. 33

[50] AllSeen Alliance. AllJoyn Enables the Internet of Things Near You, 2014. 33

[51] Open Interconnect Consortium. Connecting Billions of Devices, 2014. 33

[52] Thomas Watteyne, Lance Doherty, Jonathan Simon, and Kris Pister. Technical Overview of SmartMesh IP. In *Proceedings of IMIS '13*, Washington, DC, USA, 2013. 35, 36

[53] ZigBee PRO GreenPower. https://docs.zigbee.org/zigbee-docs/dcn/12/docs-12-0646-01-0mwg-new-zigbee-pro-feature-green-power.pdf. 35

[54] HIKOB. http://www.hikob.com/assets/uploads/2014/07/HIKOB_AZURE_LION_ProductSheet_EN.pdf. 36

[55] M3 Open Node motes. https://www.iot-lab.info/hardware/m3/. 36

[56] OpenMote. http://www.openmote.com/openmote-cc2538/. 36

[57] WisMote. http://wismote.org. 36

[58] TelosB motes. http://www4.ncsu.edu/~kkolla/CSC714/datasheet.pdf. 36

[59] Waspmote. http://www.libelium.com/products/waspmote/. 36

[60] MicaZ mote. http://www.openautomation.net/uploadsproductos/micaz_datasheet.pdf. 36

[61] Aman Kansal, Jason Hsu, Sadaf Zahedi, and Mani B Srivastava. Power management in energy harvesting sensor networks. *ACM Transactions on Embedded Computing Systems (TECS)*, 6(4):32, 2007. 36

[62] Christopher M Vigorito, Deepak Ganesan, and Andrew G Barto. Adaptive Control of Duty Cycling in Energy-Harvesting Wireless Sensor Networks. In *Proc. SECON*, pages 21–30. IEEE, 2007. 37

[63] Nicolò Michelusi, Leonardo Badia, Ruggero Carli, Kostas Stamatiou, and Michele Zorzi. Correlated Energy Generation and Imperfect State-of-Charge Knowledge in Energy Harvesting Devices. In *Proc. IWCMC*, pages 401–406. IEEE, 2012. 37

[64] Gabriele Romaniello. *Energy Efficient Protocols for Harvested Wireless Sensor Networks*. PhD thesis, Université de Grenoble, 2015. 37

[65] F. Todeschini, C. Planat, P. Milazzo, S. Tricomi, P. Urard, and P. Benabes. A Nano Quiescent Current Power Management for Autonomous Wireless Sensor Network. *Proc. of IEEE ICECS*, 2013. 38

[66] P. Urard, G. Romaniello, A. Banciu, J.C. Grasset, V. Heinrich, M. Boulemnakher, F. Todeschni, L. Damon, R. Guizzetti, L. Andre, and A. Cathelin. A Self-powered IPv6 Bidirectional Wireless Sensor & Actuator Network for Indoor Conditions. In *VLSI Circuits (VLSI Circuits), 2015 Symposium on*, pages C100–C101, June 2015. 38

[67] The Contiki OS. http://www.contiki-os.org/. 41, 66

[68] N. Tsiftes, J. Eriksson, and A. Dunkels. Low-Power Wireless IPv6 Routing with ContikiRPL. In *Proceedings of ACM/IEEE IPSN*, Stockholm, Sweden, April 2010. 41

[69] T Winter, P Thubert, A Brandt, J Hui, R Kelsey, P Levis, K Pister, R Struik, J.P. Vasseur, and R Alexander. RPL: IPv6 Routing Protocol for Low power and Lossy Networks. RFC 6550, IETF, March 2012. 41

[70] RIOT OS. http://www.riot-os.org/. 42

[71] OpenWSN project. https://openwsn.atlassian.net. 43

[72] J. Hill, R. Szewczyk, A. Woo, S. Hollar, D. Culler, and K. Pister. System architecture directions for networked sensors. In *Proceedings of the International Conference on Architectural Support for Programming Languages and Operating Systems (ASPLOS)*, November 2000. 43

[73] Joseph M Kahn, Randy H Katz, and Kristofer SJ Pister. Next Century Challenges: Mobile Networking for "Smart Dust". In *Proceedings of the 5th annual ACM/IEEE international conference on Mobile computing and networking*, pages 271–278. ACM, 1999. 43

[74] Matthias Kovatsch, Martin Lanter, and Zach Shelby. Californium: Scalable Cloud Services for the Internet of Things with CoAP. In *Proceedings of IoT 2014*, Cambridge, MA, USA, October 2014. 45

[75] Z Shelby, K. Hartke, and C. Bormann. Constrained Application Protocol (CoAP). IETF RFC 7252, 2014. 45

[76] T. Clausen. The Lightweight On-demand Ad hoc Distance-vector Routing Protocol - Next Generation (LOADng). Internet draft, 2013. 48, 71

[77] Charles Perkins, Elizabeth Belding-Royer, and Samir Das. Ad hoc on-demand distance vector (aodv) routing. Technical report, 2003. 48

[78] Chi-Anh La, Martin Heusse, and Andrzej Duda. Link reversal and reactive routing in low power and lossy networks. In *PIMRC*, pages 3386–3390. IEEE, 2013. 48

[79] V Park and M-S Corson. A Highly Adaptive Distributed Routing Algorithm for Mobile Wireless Networks. In *Proceedings of IEEE INFOCOM*, 1997. 48

[80] P. Levis, T. Clausen, J. Hui, O. Gnawali, and J. Ko. The Trickle Algorithm. RFC 6206, IETF, March 2011. 53, 65

[81] Jonathan Hui and Richard Kelsey. Multicast Protocol for Low power and lossy networks (MPL). 2014. 53, 65

[82] C Adjih, P Jacquet, and L Viennot. Computing Connected Dominated Sets with Multipoint Relays. In *Ad Hoc & Sensor Wireless Networks, Vol. 1*, 2005. 65

[83] Ghalem Boudour, Martin Heusse, and Andrzej Duda. Improving Performance and Fairness in IEEE 802.15.4 Networks with Capture Effect. In *Proceedings of the IEEE ICC 2013 (International Conference on Communications)*, Budapest, Hungary, June 2013. IEEE. 67

[84] Ghalem Boudour, Martin Heusse, and Andrzej Duda. An Enhanced Capture Scheme for IEEE 802.15.4 Wireless Sensor Networks. In *Proceedings of the IEEE ICC 2013 (International Conference on Communications)*, Budapest, Hungary, June 2013. IEEE. 67

[85] P. Levis, A. Tavakolli, and S. Dawson-Haggerty. Overview of Existing Routing Protocols for Low Power and Lossy Networks. Internet draft, IETF, 2009. 71

[86] L.-O. Varga, G. Romaniello, M Vučinić, M. Favre, A. Banciu, R Guizzetti, C. Planat, P. Urard, M. Heusse, F. Rousseau, O. Alphand, E. Dublé, and A. Duda. GreenNet: an Energy Harvesting IP-enabled Wireless Sensor Network. *IEEE Internet of Things Journal*, 2015. 71, 92

[87] Omprakash Gnawali, Rodrigo Fonseca, Kyle Jamieson, David Moss, and Philip Levis. Collection Tree Protocol. In *Proceedings of ACM SenSys*, 2009. 71

[88] Ulrich Herberg and Thomas Clausen. A Comparative Performance Study of the Routing Protocols LOAD and RPL with Bi-directional Traffic in Low-Power and Lossy Networks (LLN). In *Proceedings of ACM PE-WASUN*, 2011. 71

[89] Zigbee-Alliance, "Smart Energy Profile 2", `http://www.zigbee.org/Standards/ZigBeeSmartEnergy/SmartEnergyProfile2.aspx`. 77

[90] T. Watteyne, S. Lanzisera, A. Mehta, and K.S.J. Pister. Mitigating Multipath Fading through Channel Hopping in Wireless Sensor Networks. In *Communications (ICC), 2010 IEEE International Conference on*, pages 1–5, May 2010. 87