Chapter 1.

Bringing IP to Low-power Smart Objects: the Smart Parking Case in the CAL*IP*SO Project

Paolo Medagliani, Jérémie Leguay (Thales Communications & Security), Andrzej Duda, Franck Rousseau (Centre National de la Recherche Scientifique), Simon Duquennoy, Shahid Raza (Swedish Institute of Computer Science), Gianluigi Ferrari, Pietro Gonizzi, Simone Cirani, Luca Veltri (University of Parma), Màrius Monton, Marc Domingo, Mischa Dohler, Ignasi Vilajosana (Worldsensing), Olivier Dupont (Cisco System International)

1.1 ABSTRACT

The chapter describes the Calipso communication architecture for IP connectivity in wireless sensor networks and the Smart Parking application scenario developed within the Project. The use case is a real life demonstrator for traffic flow and parking monitoring deployed in the city of Barcelona, Spain. It is based on a communication infrastructure with sensors for parking and traffic detection embedded in the ground. The sensor nodes communicate parking space availability/traffic flow to neighboring sensors until they reach a gateway. Multi-hop routing is used when there is no direct communication with the gateway. A centralized control system stores and processes all data gathered from sensors. The resulting information and implemented services are offered to citizens by means of mobile applications and city panels. In the chapter, we analyze the requirements of the use case, present the communication architecture of Calipso, and show how the Smart Parking application takes advantage of different modules within the architecture.

1.2 INTRODUCTION

1.2.1 Bringing IP to Energy-Constrained Devices

The Internet of Things proposes the vision of interacting with the physical world by interconnecting objects with processing, communication, sensing, and actuating capabilities. The main IoT challenges include the integration of small Smart Objects having strong energy and processing constraints, large-scale interconnection of nodes through flexible and secure networking, as well as personalized interaction with the physical world and integration within the user-created content and applications.

Existing solutions stemming from past industrial and academic initiatives suffer from several limitations. The most important obstacle in the development of the Internet of Things was the advent of a large number of proprietary or semi-closed solutions such as Zigbee, Z-Wave, Xmesh, SmartMesh/TSMP that proposed different functionalities at several layers. Moreover, early research work in sensor networks suggested that the constrained and application-specific nature of sensor networks required networking to be based on non-IP concepts (Estrin, Govindan, & Kumar, 1999) (Hill, 2000). The result was the existence of many non-interoperable solutions addressing specific problems and based on different architectures and different protocols. Such approaches have not led to large-scale deployments nor to interconnection of products from different vendors. Interconnecting heterogeneous networks is possible via protocol translation gateways, but this approach also presents several problems (reduced scalability, potential security issues, no end-to-end services, etc.).

For a long time, using IP in constrained networks was considered as too complex or ill-suited for such environments.

However, with the increase of computing power and memory size, several successful implementations have showed the possibility of running the full-fledged IP stack (Dawson-Haggerty, Jiang, Tolle, Ortiz, & Culler, 2010) (Dunkels, 2003) (Hui & Culler, 2008) (Jiang, Dawson-Haggerty, Dutta, & Culler, 2009) (Priyantha, Kansal, Goraczko, & Zhao, 2008) (Yazar & Dunkels, 2009), showing that the performance of layered IP-based sensor network systems rivals that of ad-hoc solutions. One of the salient examples is the Contiki operating system, which was first used to explore IPv4 communications for sensor networks (Dunkels, 2003) (Dunkels, Voigt, & Alonso, 2004) and later provided the first fully certified IPv6 stack for IP-based smart objects (Durvy, 2008). Hui and Culler have developed an IPv6 architecture for low-power sensor networks based on IEEE 802.15.4 (Hui & Culler, 2008).

Running an IP stack on a Smart Object presents the advantage of easy integration with the current Internet and an easy reuse of existing applications or protocols. More specifically, IP provides several important characteristics:

- it is based on open standards, which is essential for interoperability, cost efficiency and innovation,
- intelligence is pushed outside the network, enabling not only network administrators but also users to develop new applications,
- flexibility—supporting a wide range of media and devices,
- universality—all protocols that solved very specific issues never survived,
- open support for security,
- support for auto configuration,
- scalability.

Obviously, the minimal computing and memory requirements for running the protocol limit the all IP approach to objects that may currently cost about tens of euros. Smaller, less powerful nodes may still operate without IP to perform some specialized functions, if the cost justifies such a choice.

The Pervasive Internet needs a universal alternative to the many existing techniques for connecting ordinary devices to the Internet. The other techniques all have something to recommend them; each is optimized for a special purpose. But in return for their optimality, they sacrifice compatibility. Since most device connectivity rarely requires maximum optimality, compatibility is a much more important objective. IP is the only answer. End-to-end IP architectures are widely accepted as the only alternative available to support the design of scalable and efficient networks comprised of large numbers of communicating devices. IP enables interoperability at the network layer, but does not define a common application-layer standard, thus making it optimal for use in a wide variety of applications ranging across several industries. (Harbor Research, 2011)

1.2.2 The CAL*IP*SO Project

CAL*IP*SO is a European FP7 project targeting the development of IP connected smart objects. In order to provide long lifetime and high interoperability, novel methods to attain very low power consumption are put in place. CAL*IP*SO leans on the significant body of work on sensor networks to integrate radio duty cycling and data-centric mechanisms into the IPv6 stack, something that existing work has not previously done. In the CAL*IP*SO project, we propose a number of enhancements to the standard low-power IP stack such as protocol optimizations, new network protocols, or security modules.

The context of CAL*IP*SO is the IETF/IPv6 framework, which includes the recent IETF RPL and CoAP protocols. It sets up a structure for evaluation that has not previously been available. Implementations have been carried out within the

Contiki open source OS, the European leading smart object OS. In order to drive the development, three applications have been considered: Smart Infrastructures, Smart Cities, and Smart Toys, all of which need both standardized interfaces and extremely low power operation.

CAL*IP*SO considers that smart object networks both need to communicate with other smart objects, other smart object networks, as well as Internet-based systems. The project goal is to push IP end-to-end end connectivity all the way into smart objects through compact, energy efficient, and loss/failure tolerant routing and radio protocols. CAL*IP*SO focuses on four specific layers of the IP stack: the MAC layer, the routing layer, the transport layer, and the application layer. With a deep understanding of the complex interactions between the layers, CAL*IP*SO is able to significantly increase the performance and reduce the power consumption of IP-based smart object networks, thereby removing major barriers to IP adoption in smart object networks. .

1.3 SMART PARKING

One of the key applications and entry-points to Smart Cities is the Smart Parking application. It is designed to help drivers in the tough process of finding a parking spot in a crowded city. With the help of this kind of applications, citizens can reduce the searching time by 8% on the average, allowing them to save time, fuel, and associated costs, and hence, reducing frustration, accidents, and increasing the quality of life in cities. Because urban traffic is the cause of 40% of CO2 and 70% of other contaminant in cities, smart parking applications also reduce overall city contamination.

Last but not least, deploying a smart parking application in a city with controlled parking areas, their occupation time is increased, the number of non-paying drivers drops and in conclusion, the total income of the municipality can be increased by almost 15%.

Some smart parking applications are based on cameras aiming at parking zones and streets with all the problems inherent to image processing applications (image quality, changing conditions, high bandwidth needs, etc.). The Smart Parking application specified in CAL*IP*SO is based on individual car sensor devices installed at every single parking spot in the city. Every device processes the signal received by car sensing techniques and it decides if there is a car parked above or if it is a free spot and sends this information to a central server, where the data is processed, clustered, and sent to the citizens in various ways (mobile phone application, on-street panels, information website, etc.).

In parallel to that, municipality and city traffic control can retrieve and manage the current and historical data to study how to enhance traffic management in the city, adjust fees on controlled parking areas, etc. This data is of a great value to the city as long as the information forms a new axis on the city data space.

Figure 1.1 shows the communication infrastructure with the sensors for parking and traffic detection embedded in the ground. The sensor nodes communicate the parking space availability/traffic flow to neighboring sensors until the data reach the gateway. Multi-hop routing is used when the direct contact with the gateway cannot be made. A centralized control system stores and processes all the data gathered from sensors.

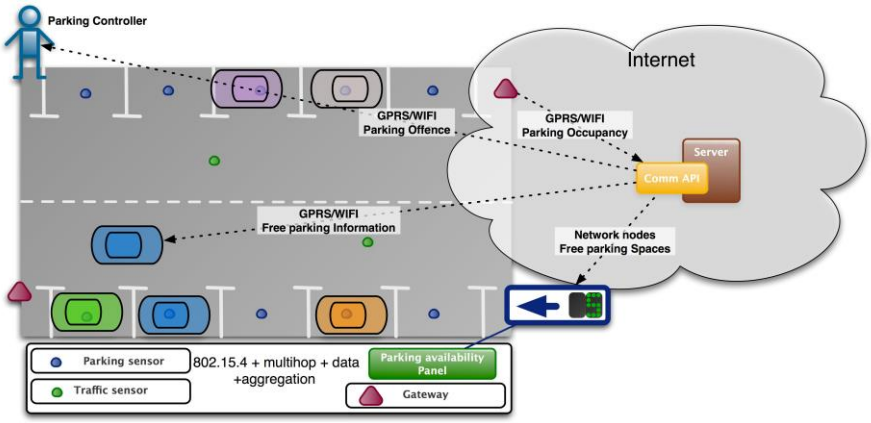


Figure 1.1 Architecture of the parking space availability control service.

The storyline of the use case is the following: a driver heading to a desired parking sub-area (i.e., within a specified walking distance to the final destination) can use this service from her mobile phone. First, the system will tell if it forecasts that free parking spaces will be available at the expected time of arrival. The availability forecast will be done at the central platform using algorithms based on context information (time of the day, day of the week, weather conditions, etc.) and historic data. Two outcomes are possible:

Case 1: available parking spaces are forecasted in this sub-area. In this case, the system will advise parking spaces with the largest numbers of free spots. Also, the system will notify the driver if traffic congestion has been detected along the route up to the selected parking space.

Case 2: no available parking spaces are forecasted in this sub-area. In this case, the system will search and recommend another sub-area with available parking spots according to user preferences. The preferences will include the proximity to the desired area and traffic status along the route.

The use case 1 will demonstrate the parking control application that consists of the following main components:

- Sensor Nodes, which are small-embedded devices containing an AMR (Anisotropic Magneto-Resistive MEMS) sensor, signal conditioning stages through FPAA, a low- power IEEE 802.15.4 wireless interface for communication. These nodes are connected to a self-organizing network for communication between nodes.
- Hybrid Gateways collecting information about parking availability from sensor nodes on the streets and transmitting the information to the centralized urban control through the Internet. They allow interconnection using different interfaces in order to be easily adapted to different urban scenarios like urban WiFi, wired municipality infrastructure, fiber optical, etc.
- Cloud Central Platform collecting the information sent by the gateways and city sources, and implementing the service to be accessed by the final users (through Information Panels and Mobile Phones). This service will identify the available parking spots and offer a forecast.
- Information Panels collecting parking availability information from the control center and display this information to guide citizens to find free parking spots.
- Mobile application to run on the users portable devices to access the information in real time and obtain recommendations.

In Figure 1.2 we show an example of real-time panel information displaying and control interface used in smart parking applications (the Worldsensing smart parking application were deployed in Barcelona and Moscow).

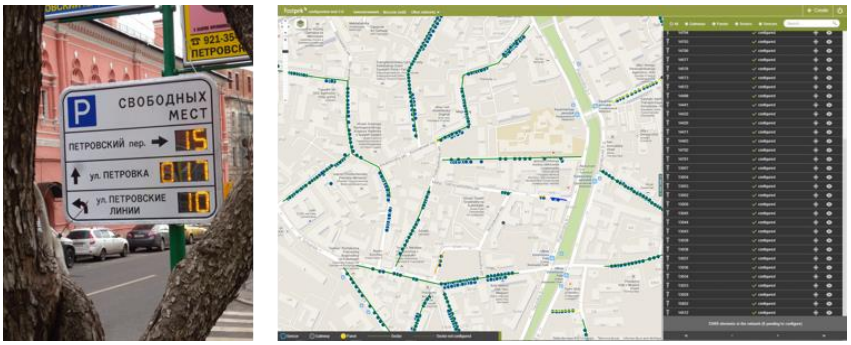


Figure 1.2: Installation and control interface of a smart parking application

The following table summarizes the requirements of the Smart Parking application developed by Worldsensing SME. The requirements are based on market demands and user experience like a long battery life (about 5 years), maximum delay (i.e. response time of the application must be less than 10 seconds), etc.

Table 1 Requirements of the Smart Parking

Aspect	**Requirement**
Physical/link layer	802.15.4 with a duty-cycling scheme
Topology	Multi-hop network
Throughput and latency	Low throughput (collection of periodic values every 4s). Support for bursty load (car footprint transmission).
Energy consumption	Battery-powered device. Lifetime required for 5 years
Duty cycling MAC	Yes. Tailored for convergecast traffic (Multipoint-to-Point) for smart parking data without in-network storage
Data aggregation/storage and in-network processing	Both required, data aggregation and in-network processing
Support of mobility	No
Traffic patterns	Multi-hop convergecast
Routing	Fairness and load balancing to mitigate hot spot problems. Evaluate metrics for RPL as Smart parking application suffers from extreme multipath that requires smart updates on the routing topology.
Transport	QoS at transport
Neighbour and service discovery	Self-discovery of capabilities provided by nodes, announced/pulled out by the gateways, and/or in the vicinity.

Security	Payload and header encryption

Table 2 Low level details of the Smart Parking

Application	
Maximum time since a car arrive until it is shown on the display	10 sec
Gateway	
Maximum nodes per gateway	> 50
Motes	
Maximum transmission distance between motes (car on)	Each device must reach 2 other devices, within 10 meters.
Maximum transmission distance between motes (no car on)	Each device must reach 3 other devices, within 15~20 meters.
Sampling rate	1 Hz
Radio	
Size of packets	20 bytes
Number of packets per minute	Each time a car comes in/out + keep-alive every 15 minutes
Duty cycle	<0.2%
Percentage of lost messages	<10%

1.4 CAL*IP*SO ARCHITECTURE

This section presents the CAL*IP*SO architecture. Figure 1.3 shows a layered view of the architecture for a Smart Object node. Bold shapes indicate the blocks on which the project focused by providing enhancements, optimizations, or adaptations to Smart Objects constraints.

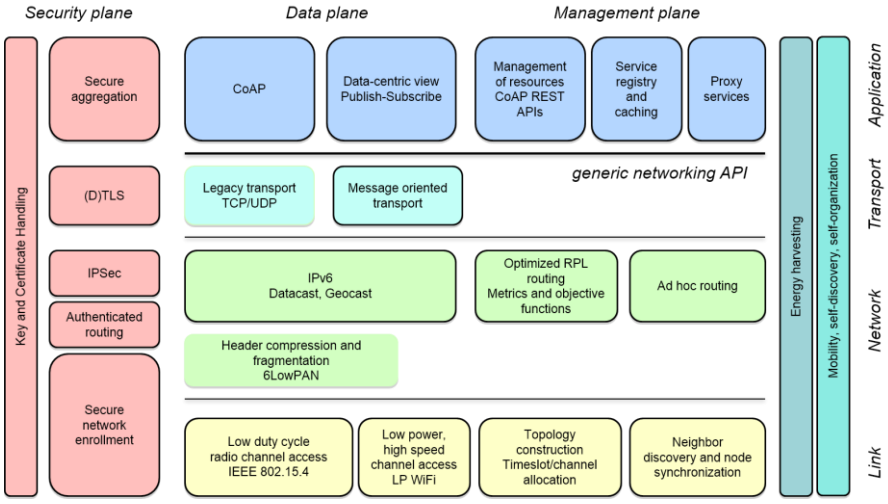


Figure 1.3 High level functional architecture of sensors.

In this figure, we do not present the gateway (or LBR, the border router in the IETF terminology) that interconnects the constrained network with the Internet. This element supports the standard TCP/IP protocol stack and adapts its operation to Smart Object nodes. The main function of the gateway relates to the 6LoWPAN layer that takes care of fragmenting IPv6 packets longer that the L2 MTU and compresses headers. It can also provide interfacing CoAP with the standard HTTP.

At PHY/MAC layer, the constrained nodes should benefit from energy efficient solutions such as IEEE 802.15.4 or Low Power WiFi, depending on application needs for bandwidth and delay. On the other side, the gateway must be able to communicate both with nodes, that is running the same PHY/MAC protocols, and with the Internet, running standard Ethernet and WiFi protocols.

Similar considerations can be carried out for the networking layer. The constrained nodes run the IPv6 protocol, coupled to 6LoWPAN to adapt to the underlying data link layer. The routing protocols and the data communication paradigms as well are specific to the constrained domain. The gateway, instead, in addition to the above mentioned solutions, must also run traditional legacy IPv4 and IPv6 to interconnect to existing network infrastructures.

At the transport layer, both gateway and nodes can run legacy protocols such as TCP or UDP, with a preference for UDP in the constrained world, due to its reduced memory and resource use. In addition, nodes can implement a publish/subscribe mechanism to improve data collection and reduce unneeded communications.

Finally, at the application layer, nodes run specific lightweight protocols to enable efficient communications, such as caching of data or REST-like interfaces adapted to the constrained world. The gateway, in addition to run the same protocols, is in charge of the interconnection with the Internet, therefore it exposes REST APIs that allow remote users to easily interact with the gateway.

Gateways do not have constraints in terms of energy and computational capabilities and this kind of issues primarily concern nodes that need to deal with energy saving, self-organization, mobility, and security. Nodes appearing in the network must be able to automatically learn about network parameters. Eventually, the running protocols must take into account that nodes can move and react to the change of a position. Securing communications is another important aspect of the constrained world since the computational capabilities do not allow for traditional security mechanisms. Since nodes are battery powered or energy harvested, saving energy becomes crucial to extend the whole network lifetime. These aspects have an impact on the whole architecture of a node, requiring a careful specific design of the running protocols.

In the following, we detail the most important project contributions included in the CAL*IP*SO stack, as well as some existing protocols and solutions that have been exploited within CAL*IP*SO.

1.4.1 CALIPSO Communication Modules

1.4.1.1 MAC Layer

RAWMAC

RAWMAC is a cross-layer mechanism in which the routing layer, for instance the RPL protocol presented in Section 1.4.1.2, is used as a management layer for organizing the asynchronous duty-cycled MAC protocols (such as ContikiMAC).

ContikiMAC (Dunkels, 2011) (Duquennoy, Österlind, & Dunkels, 2011) is the Contiki default low-power listening MAC protocol. To listen, nodes periodically wake up (e.g. at 8 Hz) and proceed with two Clear Channel Assessments (CCAs) with an interval that guarantees that any ongoing packet transmission will be sensed. Upon detecting activity on the channel, the node keeps its radio on, waiting for an incoming packet. To send, nodes transmit the data packet repeatedly as a wakeup signal, and keep doing so until they received an acknowledgement (case of unicast) or for exactly one wakeup period (case of broadcast).

ContikiMAC implements a so-called “phase-lock” optimization where senders remember the wakeup phase of each of their neighbors to optimize subsequent transmissions (wakeup signal starts briefly before expected target wakeup). Once a receiver is active, the transmitter can transmit arbitrarily long sequences of packets to amortize the wakeup procedure and allows for efficient forwarding of large data bursts (Duquennoy, Österlind, & Dunkels, 2011).
The contention-based, unscheduled nature of ContikiMAC allows it to handle random traffic patterns while sleeping more than 99% of the time.
Because it emulates an always-on link and makes no assumptions on the above layers, ContikiMAC is an ideal choice for low-power IP scenarios.

The key idea of RAWMAC is to exploit the DODAG built by RPL to make each node align its wake-up phase with that of its preferred parent, creating a data propagation “wave” from the leaves of the DODAG to the root. Once a data packet rides this wave, the latency is significantly reduced, as it depends only on small propagation delays and on the internal processing carried out at each device to forward the packet. By properly configuring the phase lock mechanism of ContikiMAC, the transmitting node wakes up only when the receiving node is ready to receive the packet, so that the energy consumption is kept as low as possible.

Figure 1.4 shows this wake-up phase alignment in RAWMAC. As long as the routing structure is established, a node shifts its wake-up phase in order to be aligned with that of its parent. More precisely, it sets the wake-up phase to the time at which it received the last link layer ACK from its RPL preferred parent. Since the preferred parent must have been awake to receive the packet, the node can assume that the reception of the ACK means that it has successfully transmitted a packet within the preferred parent wake-up window and, thus, that it has found the preferred parent wake-up phase.

We define the phase offset P_o (dimension: [s]) as the offset between the node wake-up phase and the wake-up phase of its parent. Given the node sleeping interval C_T, it holds that $0 \leq P_o \leq C_T$. The parameter Po has indeed to be chosen carefully, since it has an impact on the system delay performance. If P_o is too short, a node relaying a packet may not be able to catch its parent wake-up, because the reception of the same packet from its child has not completed yet. If this is the case, then the child should wait the next cycle time C_T to be able to forward the packet. If P_o is too long, instead, the delay significantly increases, as the sender has to wait for the receiver to wake up to be able to transmit the packet.

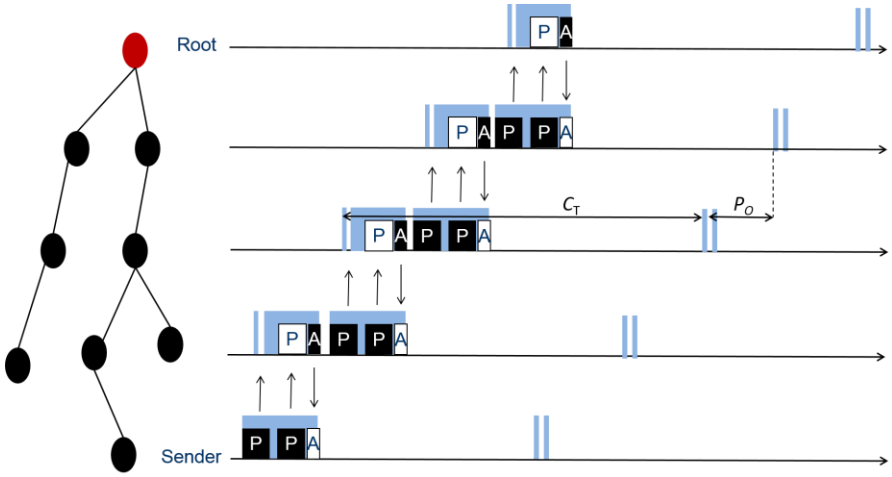


Figure 1.4: Principle of RAWMAC: the wake-up phase alignment. Once the RPL DODAG is set up, each node shifts its wake-up phase according to that of its parent in the DODAG.

1.4.1.2 Routing layer

RPL

The RPL routing protocol (Winter, 2012) (Gaddour & Koubâa, 2012) has been developed for a limited data rate and lossy environments. Actually, RPL is the most adopted routing protocol for constrained networks. RPL is a distance-vector protocol based on the creation of a routing topology referred to as the *Destination Oriented Acyclic Directed Graph* (DODAG), in which the cost of each path is evaluated according metrics defined in an objective function. The goal of this protocol is the creation of a collection tree protocol, as well as a point-to-multipoint network from the root of the network to the devices inside the network.

To keep the status of the network updated, the root of the RPL DODAG periodically broadcasts *DODAG Information Object* (DIO) control messages. The receiving nodes may relay this message or just process it, if configured as leaves of the tree.

The RPL protocol also introduces a *trickle* mechanism that allows to reduce the transmission frequency of DIO messages according to the stability of the network.

In some cases, when specific flags in DIO packets are set, the nodes receiving a DIO are stimulated for the generation of a *Destination Advertisement Object* (DAO) messages. This is a unicast data packet that can be sent either directly to the root, when a *non-storing* mode is used, or to the selected parent in the *storing* mode. The messages create downstream routes from the root to the leaves.

In the former case (non-storing mode), intermediate nodes simply add their addresses to the DAO header. Only the root stores the downward routing table of the tree. In the latter case (storing mode) instead, since DAO messages are directly processed by the parent node that receives the packets, each node stores a routing table for the children associated to it. Before sending in its turn a DAO, a node aggregates the information received by the children, so that the aggregated reachability information is sent to its parent. As unicast messages, the DAO can be acknowledged by the receiver.

The third type of message foreseen by RPL is the *DODAG Information Solicitation* (DIS) used by nodes to advertise their presence in the network so that they can join an existing DODAG.

ORPL

ORPL (Duquennoy, Landsiedel, & Voigt, 2013) is an opportunistic extension to RPL. The basic idea behind ORPL is to replace unicast transmissions to a specific next hop by anycast transmissions aimed at any node that offers progress towards the destination. Figure 1.5 illustrates the anycast operation in which traditional routing estimates link quality and sticks to links that appear to be generally good, while ORPL uses any link available at the time of transmission.

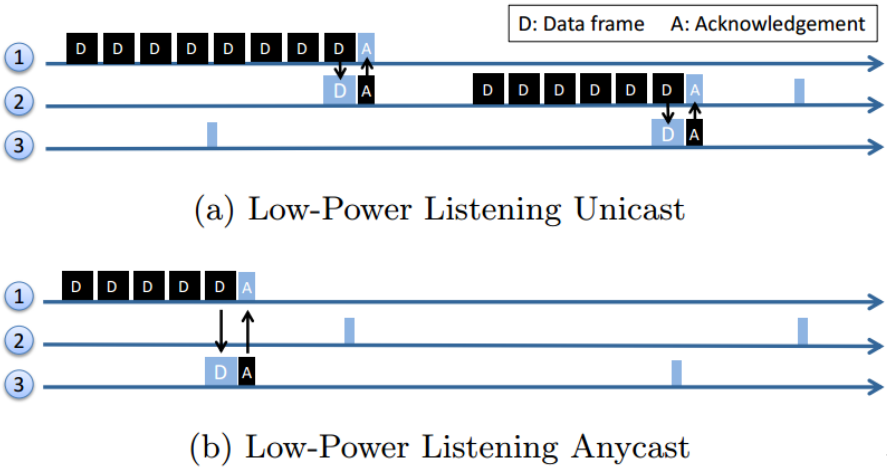


Figure 1.5: Traditionally, routing uses unicast over stable links aiming at stability. Low-power listening introduces a significant delay at every hop. ORPL uses anycast and transmits to the first awoken forwarder that receives the packet regardless of link quality estimates. Combined with ContikiMAC radio duty cycling, ORPL conciliates low energy (nodes sleep most of the time) and low latency (first awoken neighbor forwards).

Routing in ORPL is possible towards the root by simply following a gradient, but also towards any other node by going away from the gradient, directed by lightweight routing tables that merely contain the set of nodes below in the topology ("routing sets").

ORPL was tested at a large scale and has shown to attain delivery ratios over 99% together with sub-percent duty cycles and sub-second latency.

RRPL

Reactive RPL is a lightweight version of RPL that retains the collection tree structure (DODAG) of RPL, but speeds up local repair in case of link failures and allows for reactive or proactive routing for downward (P2MP) traffic. It provides a mechanism for fast local route repair through the use of a link reversal algorithm towards the sink. It takes advantage of the existing DODAG structure to enable efficient reactive route search. Moreover, RRPL does not impose the use of the additional RPL header in each packet, since it quickly detects and fixes routing loops.

Featurecast

Featurecast is a new address-centric communication paradigm especially well suited for sensor networks. It uses a data-centric approach to select destination nodes: destination addresses correspond to a set of features characterizing sensor nodes. For instance, we can reach a group of nodes satisfying the featurecast address [`4th floor and temperature`]. In this way, the communication mode closely reflects how application developers reason about sensors, actuators, and their

interaction with the real world. Features may be freely defined and specific to a particular application or a given deployed network.
Featurecast extends the standard notion of multicast with a more general definition of groups: instead of one address representing a multicast group, a featurecast address defines the group membership based on a set of features. With featurecast, a node has implicitly an address for every subset of its advertised features. We propose a scheme for efficiently routing packets to all such addresses as well as an address coding compatible with the multicast IPv6 address format.

1.4.1.3 Application layer

CoAP

RESTful web services and the HTTP protocol are widely used to publish the status of resources. However, web services are not suitable for constrained networks due to the high overhead introduced by HTTP and the presence of the TCP congestion window. For these reasons, the CoRE IETF working group aimed at adapting the REST architecture to constrained networks through the definition of a protocol referred to as Constrained Application Protocol (CoAP) (Shelby, 2013). The RESTful architecture, as the idealized model of the Web, is usually implemented with HTTP, and its 4 basic methods: GET, POST, PUT, and DELETE.
In CoAP, the same four REST verbs have been implemented for constrained devices. CoAP is by nature more lightweight than HTTP, supports multicast natively, as well as the publish-subscribe model. It is for example perfectly possible with CoAP to subscribe with a single request to all smoke detectors in a building via a multicast request. In case of fire, an alarm can then be multicasted to all subscribers. CoAP supports reliable communication as an option on top of UDP. Actually, CoAP has become a de facto standard to expose webservices in constrained networks.

HTTP/CoAP Proxy

The HTTP/CoAP (HC) proxy with caching functionality has been implemented relying on the IETF protocols. We made this choice both because some libraries, which can be reused to speed up the implementation, are already available and because IETF protocols have been specifically designed to meet the constraints typical of IoT architectures.

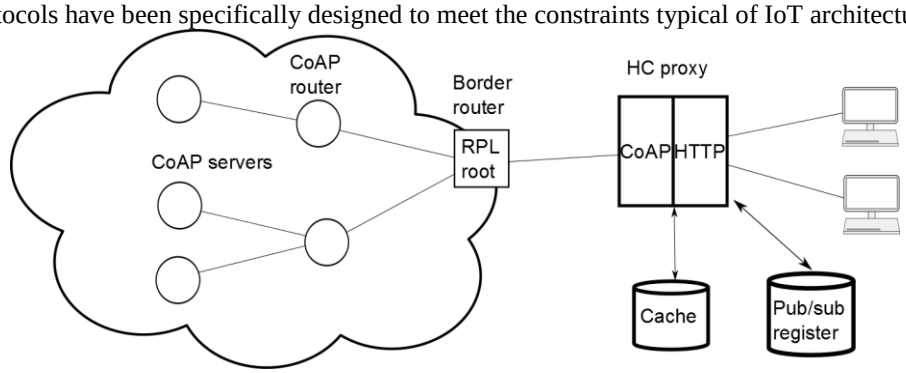


Figure 1.6: Scheme of the implemented HTTP/CoAP caching system.

In Figure 1.6, we show a logical scheme of the implemented caching system, where the existing CoAP servers expose some resources to be queried. The running routing protocol is RPL. All the requested information and the notifications from the nodes are gathered to the root of the RPL tree, which overlaps with the WSN gateway. The HC proxy will then send the requested information to the final user.
Our HC proxy implementation is based on the Californium open source framework (Kovatsch, Mayer, & Ostermaier, 2012). The main reason of this choice is that Californium already implements in JAVA a basic set of CoAP functionalities. We have then introduced two information storing mechanisms. The first one is the caching database,

where all the information from the WSN are gathered and stored and for which we have chosen CouchDB, since it offers a REST HTTP API to query stored results. The second one is a subscription register to efficiently manage the subscribers to CoAP resources.
Requests are intercepted by the HC proxy that also handles the eventual response from a given CoAP server. If the proxy has a stored value that is fresh enough, that is whose lifetime is smaller than a given value, it directly replies to the request of a remote client, without forwarding it into the WSN. Otherwise, if the required value is not present or it is too old, it transfers the request to the intended CoAP server. Additionally, the proxy stores the sensor responses in the cache to make them available for other eventual incoming requests. A similar approach is used for the publish-subscribe register. In the case of observation requests issued by a remote client, the proxy handles them by maintaining a list of observed resources and a list of interested clients. Each time a notification for a resource update is sent from the node to the proxy, the proxy will forward this message to all interested subscribers.
Caching and observe mechanisms can operate together as well. For instance, the information already requested by an observe request can be cached and made available to another request not related to the previous observation. In fact, the caching and the observation mechanism are independent from each other. Nevertheless, the two systems should be used together because the information obtained by observation can help the caching operations.

1.4.2 CAL*IP*SO Security Modules

Compression of IPsec AH and ESP Headers for Constrained Environments

The security in 6LoWPAN networks is particularly important as we connect smart objects to the insecure Internet. The standardized and mandatory security solution for IPv6 is IPsec. We have proposed an extension of 6LoWPAN defining header compression for IPsec datagrams (Raza, Duquennoy, & Selander, 2014) (Raza, et al., June 2011).

We focus on the IPsec transport mode that provides end-to-end security, both from device to device and from device to traditional Internet hosts.

Our solution supports both the AH and ESP protocols, where:

- AH authenticates the IPv6 header and payload between two end hosts;
- ESP authenticates and encrypts the IPv6 payload (but not the header) between the end hosts.

When needed, IPsec AH and ESP can be combined to link-layer security to encrypt and authenticate the entire payload of the frame in a hop-by-hop fashion.
Our solution brings standard Internet-class security to the most constrained devices, i.e. devices running Contiki on 16-bit MCUs and with only tens of kilo-bytes of memory (Raza, et al., June 2011).

Distributed key verification and management

Distributed key certification renders sensor nodes fully autonomous and does not require the use of central authorities and certificates. We have designed a protocol based on asymmetric cryptography and one-way accumulators. It provides secure node enrollment and key certification suitable for other security protocols like IPsec and DTLS.

The protocol relies on Elliptic Curve Cryptography (ECC) for public/private keys as well as the accumulator. Each node is assigned a pair of keys and the material needed for the one-way accumulator: its witness, and the accumulator containing all the public keys of the nodes in the network, including the gateway. Two nodes can then assess that they are part of the same network by sending their public key and the witness to each other for mutual verification. Once the nodes have mutually verified their public keys, they can establish a symmetric key through the Elliptic Curve Diffie-Hellman (ECDH) key agreement protocol.

Our protocol inherits the major security properties of one-way accumulators: “one-way-ness” and resistance to forgery. Regarding node injection, the security of our protocol is reduced to the security of the accumulator. Due to the one-way-ness of accumulators, the capture of a node does not compromise the communications of other nodes: only the

keys of the captured node are compromised. Finally, we deal with denial-of-service by triggering the most expensive operation (ECDH) using a table lookup, hence preventing the replay of correct messages that would cause the exhaustion of the node resources otherwise.

OAuth

Open Authorization (OAuth) (Hammer-Lahav, 2010) is an open protocol that allows secure authorization in a simple and standardized way from third-party applications accessing online services, based on the REST web architecture. OAuth has been designed to provide an authorization layer, typically on top of a secure transport layer such as HTTPS.

OAuth defines three main roles:

- the User (U) is the entity who generates some sort of information;
- the Service Provider (SP) hosts the information generated by the users and makes it available through APIs;
- the Service Consumer (SC), also referred to as "client application", accesses the information stored by the SP for its aims.

In order to allow a client application access information on his/her/its behalf, a user must issue an explicit agreement. The agreement results in the grant of an access token, containing the user's and client application's identities, the client. The client must exhibit the access token in every request as an authorization proof. The OAuth 2.0 protocol enhances the original OAuth protocol focusing on the easiness of client development (Hardt, 2012).

Smart objects providing CoAP-based services might also require some authorization for permitting access by third-party applications. In order to meet this kind of security requirement, smart objects might benefit by the use of the OAuth protocol. However, due to the need to execute heavy cryptographic computation and memory footprint issues (both in terms of available ROM and RAM on smart objects), it is not feasible to implement the OAuth logic and access-token management directly on the device.

For this reason, a novel OAuth-based authorization framework targeted to IoT scenarios has been designed and implemented. The authorization framework, denoted as "IoT-OAS", allows smart objects to delegate authorization-related operations in order to minimize the memory occupation due to the implementation of specific software and storage modules.

The procedure takes an incoming OAuth-secured request and asks the IoT-OAS authorization service to verify the access token included in the request against a set of client and user credentials it stores, by using the appropriate digital signature verification scheme, as specified in the OAuth protocol definition (either PLAINTEXT with secure transport, HMAC, or RSA). Upon reception of the request, the IoT-OAS service computes the digital signature for the incoming message and performs a lookup in its internal credential store to see if the request matches client identity, user grants, and requested resource access. If the signature is verified and the resource is set to be accessible, the IoT-OAS service replies with a success response and the request can then be served. Otherwise, the request is blocked and a client error response is sent back to notify that the client is not authorized to access the requested resource.

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

1. the time required to build new OAuth-protected online services, thus letting developers focus on service logic rather than on security and authorization issues;
2. the simplicity of the Smart Object, which does not need to implement any authorization logic but must only invoke securely the authorization service in order to decide whether to serve an incoming request or not;
3. the possibility to configure the access control policies (dynamically and remotely) that the smart object (acting as SP) is willing to enforce, especially in those scenarios where it is hardly possible to intervene directly on the device.

The IoT-OAS architecture supports HTTP (secured through TLS transport) and CoAP (secured through DTLS transport) requests sent by external clients targeting services provided by smart objects, thus enabling the possibility to perform access control either on the device (before serving requests) or on HTTP/CoAP proxies at the border of a constrained network (filtering incoming requests, which are consequently delivered or not to the target smart object).

1.5 CALIPSO IMPLEMENTATION AND EXPERIMENTATION WITH SMART PARKING

This section presents the implementation of the Calipso modules, described in the previous section, and the experimentation plan for the Smart Parking application.

1.5.1 Implementation of Calipso modules

Figure 1.7 shows the integration of the different modules. As Calipso targets several IoT-based applications, the integration of the modules is made as flexible as possible, to allow the choice between different protocols depending on the deployed application.

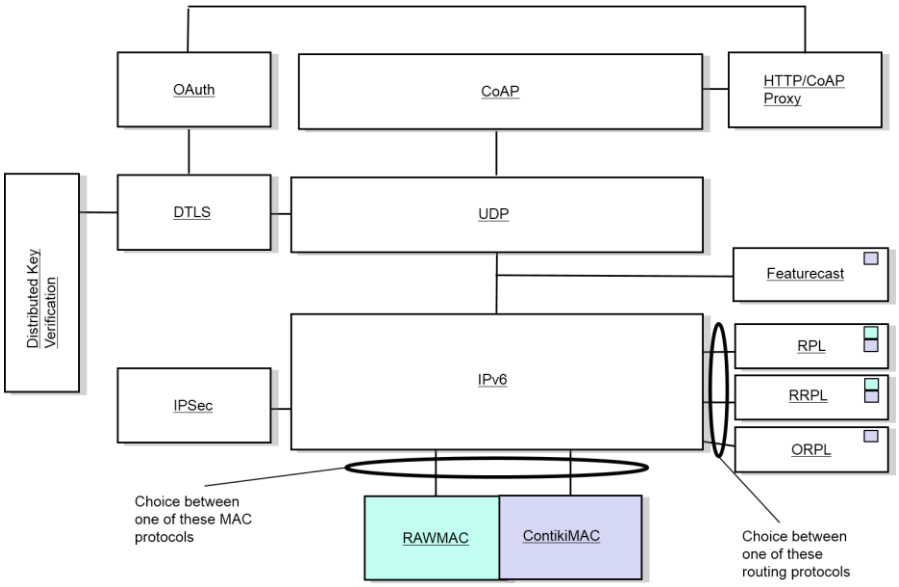


Figure 1.7 Software architecture for smart parking.

At the MAC layer, we can choose between two different MAC protocols: (i) RAWMAC and (ii) ContikiMAC. RAWMAC is suitable for efficient data collection (i.e., mostly mono-directional traffic). In order to create a "wave", this protocol requires a routing layer that does not vary routing tables very often. For instance, it can rely on RPL since routes are created and maintained proactively during time. On the other side, ContikiMAC is suitable in scenarios where low power communications are mostly bi-directional. Differently from RAWMAC, it does not require any information from the upper layer. Clearly, the choice of a specific MAC layer has an impact on the routing plane.

At the networking layer, we have the choice between three routing protocols: (i) ORPL, (ii) Reactive RPL, and (iii) RPL. ORPL is an extension of RPL where packets are forwarded opportunistically towards their destination, leading to increased reliability, shorter delay and reduced energy consumption. ORPL relies on anycast at the MAC layer (ContikiMAC), making it incompatible with the unicast-oriented RawMAC. ORPL performs at best in static, dense network, and supports random channel access. In use cases where only a few nodes are connected to the network at the same time and the impact of opportunistic strategies would be limited, it is preferable to adopt more conservative mechanisms providing anyway interesting capabilities such as fault tolerance and slow mobility management. RRPL instead, provides on-demand route recalculation. This allows to handle node (limited) mobility and to react to network failures. RRPL can coexist with RAWMAC since, as soon as a route is recalculated, RAWMAC follows route modification.

All the MAC and routing protocols presented above are perfectly compatible with IPv6, as well as with the IPSec security mechanism that allows providing secured communications. Featurecast instead is only compatible with ContikiMAC. In fact, Featurecast has the objective of providing an alternative to group communications for networks where no or limited mobility is provided. Since the considered routing protocols address unicast communications, whereas Featurecast is developed for multicast communications, these protocols can coexist natively. In contrast, Featurecast is not perfectly compatible with RAWMAC as the latter has been designed for aligning activity phases for efficient many-to-one communications. In Featurecast , instead, communications follow a many-to-many pattern. The two protocols could then coexist, but the delay introduced by RAWMAC would be significant.

At the transport layer, UDP is compatible with all the above-mentioned protocols, due to its stateless operation. In order to provide end-to-end security over UDP, we use DTLS. Since we want to reduce the overhead of cryptographic computation, we introduce a distributed key validation mechanism. In addition, as it could be important to authenticate users accessing the network, we use the OAuth solution. Because an authentication server cannot be executed directly on nodes for a matter of resource consumption, it has been integrated with the proxy node.

At the application layer we choose CoAP, that offers RESTful primitives to constrained networks. The most used verbs of CoAP are GET and Observe. In the former case, we deal with bi-directional communications, whereas in the latter, we mainly have mono-directional data transfer, matching very well a protocol for data collection such as RAWMAC.
When many requests from external users are carried out on the same node, it would be more efficient to have a caching node outside the constrained network replying on behalf of the in-network node. In addition, since requests from the Internet are coming in HTTP, it necessary to translate request (and responses) from HTTP to CoAP (and from CoAP to HTTP). The HTTP/CoAP proxy is then in charge of carrying out this protocol translation. The proxy is also able to publish data on remote webservers in case this is required by the use case.

1.5.2 Experimentation plan for Smart Parking

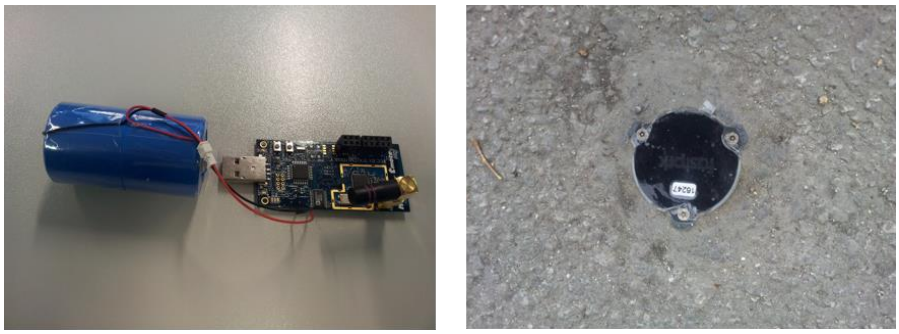
The integrated modules will be evaluated in the Smart Parking application, whose requirements have been already defined in previous sections.

1.5.2.1 Prototype Description

In the experiments, it is planned to use the Tmote Sky platform. This mote is an ultra low power wireless module that leverages industry standards like USB and IEEE 802.15.4 to interoperate with other devices. The Tmote Sky has been the base for most of the developments done in Calipso.
The Tmote Sky has the following hardware characteristics:

- 250kbps 2.4GHz IEEE 802.15.4 Chipcon Wireless Transceiver
- 8MHz Texas Instruments MSP430 microcontroller (10k RAM, 48k Flash)
- Integrated onboard antenna with 50m range indoors / 125m range outdoors
- Integrated Humidity, Temperature, and Light sensors
- Ultra-low current consumption
- Programming and data collection via USB
- optional SMA antenna connector
- Contiki support

The Tmote is connected to a standard battery (see Figure 1.8 (a)) and placed into a Worldsensing box to be buried into the tarmac (see Figure 1.8 (b)).



(a) (b)

Figure 1.8: (a) Tmote with battery ready for boxing, and (b) Tmote in the Worldsensing box in the real deployment.

1.5.2.2 Description of the Scenario

Each device will be integrated with the Worldsensing box in order to be buried in tarmac in the same way a real installation is done. This installation procedure will mimic the exact conditions for a real deployment.
The test facility is situated in a corner street in Barcelona in the 22@ neighborhood. Its location can be seen in Figure 1.9. The corner is Parking Load Zone that it is restricted for loading and unloading activities, with a maximum stop time of 30 minutes. Each device is buried about 2 meters a part of each other, for a total of 6 devices to cover the entire corner. This zone is selected due to its high car churn (car replacement) and its lack of marked parking space. This lack of markings causes different placement of any truck or car parking on the zone, changing the radio link between motes each time a car enters or leave the zone. These changes will be reflected on changes on the routing and the topology of the network created.
The network topology would be available through tools consulting the base station (Figure 1.10).

In order to test RAWMAC performance, nodes are configured in a chain topology as shown in Figure 1.10. As soon as a car is moved, a message is sent to the web application. We will measure the delay of transmission and the packet delivery ratio. In addition, we will fix a threshold on delay and we will compare the energy consumption of RAWMAC and ContikiMAC. We will also measure the convergence time while RRPL is establishing new routes, the lengths of the established routes/paths, as well as the traffic overhead of RRPL signaling.

Figure 1.9: Picture of the selected smart parking test facility

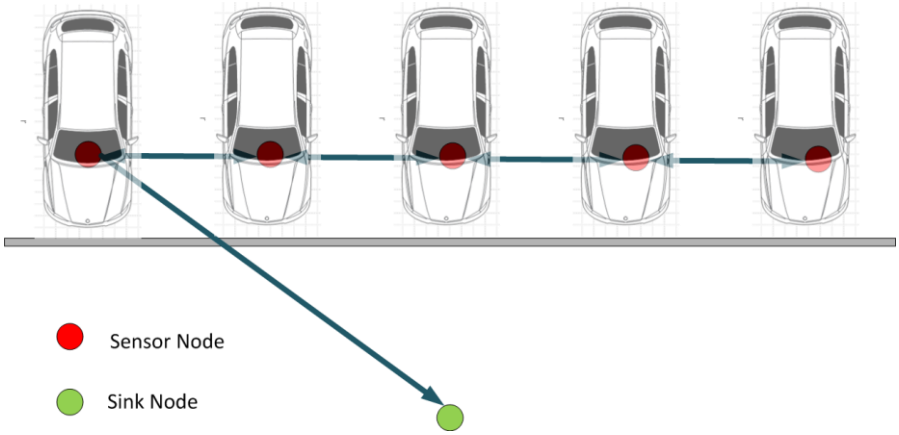


Figure 1.10: Configuration of the network topology

1.5.2.3 Performance indicators

For each of the tested modules, we provide the performance indicators that will characterize the operation of the proposed mechanisms.

RAWMAC
The performance of RAWMAC will be measured in terms of the following metrics:

- Packet delivery ratio: the ratio of packets successfully delivered to the base station.
- Energy consumption of the sensors.
- Upward delay (Optional): delay of a packet sent from a sensor to the base station.

Reactive RPL
The performance of reactive RPL will be measured in terms of the following metrics:

- Code footprint: the amount of ash memory the routing protocol uses when compiled into the node firmware.
- Convergence time and route length measure how quickly the routes are established and maintained as well as their length.
- Traffic overhead to maintain connectivity. Routing packets consume radio resources, but also computation time at the sender and receiver.

ORPL
The performance of ORPL will be measured in terms of the following metrics:

- Packet delivery ratio: the ratio of packets successfully delivered to the base station.
- End-to-end delay: delay from a node transmitting to the base station receiving it.
- Duty cycle: proportion of time with radio turned on, used as a proxy for power.
- Hop count: the number of hops for transmitted packets to reach the root (this metric does not reflect the end goal, but is interesting for network monitoring).

1.6 CONCLUDING REMARKS

In this chapter, we have first presented the CAL*IP*SO communication architecture and the Smart Parking use case. Then, we have detailed the most relevant modules developed in CAL*IP*SO showing how they contribute to meet the use case objectives and which interfaces are necessary to make the modules interoperable.

The example application clearly shows how IP can foster communications among constrained smart objects. Currently, the CAL*IP*SO protocol stack is being tested and validated in the Smart Parking application. Our preliminary results show significant improvements in performance and energy consumption while extending IP reachability to small constrained devices. Other applications may benefit from the functionalities to bring new innovative services to citizens.

Future research extensions of CAL*IP*SO project will encompass the inclusion of energy harvesting techniques into the developed stack and the use of heterogeneous networks. In order to let the devices recharge, even more severe duty cycles at the nodes must be used. Thus, the solutions developed must be adapted to take into account this additional functionality.
Actual solutions for constrained devices barely allow the transmission of large amount of data due to the limited bandwidth offered, so coupling a high speed and high energy-consumption interface with a low power and low bandwidth interface will boost the overall network capacity.

Acknowledgements
The work of the authors is funded by the European Community's Seventh Framework Program, area "Internetconnected Objects", under Grant no. 288879, CAL*IP*SO project - Connect All IP-based Smart Objects. The work reflects only the authors views; the European Community is not liable for any use that may be made of the information contained herein.

References

Dawson-Haggerty, S., Jiang, X., Tolle, G., Ortiz, J., & Culler, D. (2010). SMAP: a simple measurement and actuation profile for physical information. *Proceedings of the 8th ACM Conference on Embedded Networked Sensor Systems,*, 197-210.

Dunkels, A. (2003). Full TCP/IP for 8-bit architectures. *Proceedings of The International Conference on Mobile Systems, Applications, and Services (MobiSys)*, 14 pages.

Dunkels, A. (2011, December). The ContikiMAC Radio Duty Cycling Protocol. *Technical Report T2011:13*. Swedish Institute of Computer Science.

Dunkels, A., Voigt, T., & Alonso, J. (2004). Making TCP/IP Viable for Wireless Sensor Networks. *Proceedings of the European Conference on Wireless Sensor Networks (EWSN)*, 4 pages.

Duquennoy, S., Landsiedel, O., & Voigt, T. (November 2013). Let the Tree Bloom: Scalable Opportunistic Routing with ORPL. *Proceedings of the International Conference on Embedded Networked Sensor Systems (ACM SenSys 2013).* Rome, Italy.

Duquennoy, S., Österlind, F., & Dunkels, A. (November 2011). Lossy Links, Low Power, High Throughput. *Proceedings of the International Conference on Embedded Networked Sensor Systems (ACM SenSys 2011).* Seattle, WA, USA.

Durvy, M. et al. (2008). Making Sensor Networks IPv6 Ready. *Proceedings of the International Conference on Embedded Networked Sensor Systems (ACM SenSys)*, 421-422.

Estrin, J. H., Govindan, R., & Kumar, S. (1999). Next century challenges: scalable coordination in sensor networks. *Proceedings of the fifth annual ACM/IEEE international conference on Mobile computing and networking*, 263-270.

Gaddour, O., & Koubâa, A. (2012). RPL in a nutshell: A survey. *Computer Networks: The International Journal of Computer and Telecommunications Networking*, 3163-3178.

Hammer-Lahav, E. (2010, April). The OAuth 1.0 Protocol - RFC 5849.

Harbor Research. (2011). *M2M & Smart Systems, Machine-To-Machine (M2M) & Smart Systems.* London: Harbor Research.

Hardt, D. (2012, October). The OAuth 2.0 Authorization Framework - RFC 6749.

Hill, J. (2000). System Architecture Directions for Networked Sensors. *ACM SIGOPS Operating Systems Review*, 93–104.

Hui, J., & Culler, D. (2008). IP is Dead, Long Live IP for Wireless Sensor Networks. *Proceedings of the International Conference on Embedded Networked Sensor Systems (ACM SenSys)*, 14 pages.

Jiang, X., Dawson-Haggerty, S., Dutta, P., & Culler, D. (2009). Design and implementation of a high-fidelity ac metering network. *Proceedings of the International Conference on Information Processing in Sensor Networks (ACM/IEEE IPSN)*, 12 pages.

Kovatsch, M., Mayer, S., & Ostermaier, B. (2012). Moving Application Logic from the Firmware to the Cloud: Towards the Thin Server Architecture for the Internet of Things. *Proc of the 6th Int Conf on Innovative Mobile and Internet Services in Ubiquitous Computing (IMIS 2012)*, 6 pages.

Priyantha, B., Kansal, A., Goraczko, M., & Zhao, F. (2008). Tiny web services: design and implementation of interoperable and evolvable sensor networks. *Proceedings of the International Conference on Embedded Networked Sensor Systems (ACM SenSys)*, 253-266.

Raza, S., Duquennoy, S., & Selander, G. (2014). *Compression of IPsec AH and ESP Headers for Constrained Environments*. Retrieved from http://tools.ietf.org/html/draft-raza-6lo-ipsec-00

Raza, S., Duquennoy, S., Chung, T., Yazar, D., Voigt, T., & Roedig, U. (June 2011). Securing Communication in 6LoWPAN with Compressed IPsec. *7th IEEE International Conference on Distributed Computing in Sensor Systems (DCOSS '11).* Barcelona, Spain.

Shelby, Z. et al. (2013). Constrained Application Protocol (CoAP). draft-ietf-core-coap-18.

Winter, T. et al. (2012, March). RPL: IPv6 Routing Protocol for Low-Power and Lossy Networks. RFC 6550.

Yazar, D., & Dunkels, A. (2009). Efficient Application Integration in IP-Based Sensor Networks. *Proceedings of the ACM BuildSys 2009 workshop, in conjuction with ACM SenSys 2009*, 6 pages.