IETF Standardization in the Field of the Internet of Things (IoT): A survey

Article *in* Journal of Sensor and Actuator Networks · April 2013
DOI: 10.3390/jsan2020235

8 authors, including:

Isam Ishaq
Al-Quds University
21 PUBLICATIONS **202** CITATIONS
SEE PROFILE

David Carels
Ghent University
9 PUBLICATIONS **123** CITATIONS
SEE PROFILE

Girum Ketema Teklemariam
Ghent University
14 PUBLICATIONS **160** CITATIONS
SEE PROFILE

Jeroen Hoebeke
Ghent University
127 PUBLICATIONS **786** CITATIONS
SEE PROFILE

J. Sens. Actuator Netw. **2013**, *2*, 235-287; doi:10.3390/jsan2020235

OPEN ACCESS

Journal of Sensor and Actuator Networks

ISSN 2224-2708

www.mdpi.com/journal/jsan/

Review

IETF Standardization in the Field of the Internet of Things (IoT): A Survey

Isam Ishaq *, David Carels, Girum K. Teklemariam, Jeroen Hoebeke, Floris Van den Abeele, Eli De Poorter, Ingrid Moerman and Piet Demeester

Department of Information Technology (INTEC), Ghent University - iMinds,
Gaston Crommenlaan 8 Bus 201, Ghent 9050, Belgium; E-Mails: david.carels@intec.ugent.be (D.C.); girum.ketema@intec.ugent.be (G.K.T.); jeroen.hoebeke@intec.ugent.be (J.H.); floris.vandenabeele@intec.ugent.be (F.V.A.); eli.depoorter@intec.ugent.be (E.D.P.); ingrid.moerman@intec.ugent.be (I.M.); piet.demeester@intec.ugent.be (P.D.)

* Author to whom correspondence should be addressed; E-Mail: isam.ishaq@intec.ugent.be; Tel.: +32-933-14-900; Fax: +32-933-14-899.

Received: 15 February 2013; in revised form: 2 April 2013 / Accepted: 9 April 2013 / Published: 25 April 2013

Abstract: Smart embedded objects will become an important part of what is called the Internet of Things. However, the integration of embedded devices into the Internet introduces several challenges, since many of the existing Internet technologies and protocols were not designed for this class of devices. In the past few years, there have been many efforts to enable the extension of Internet technologies to constrained devices. Initially, this resulted in proprietary protocols and architectures. Later, the integration of constrained devices into the Internet was embraced by IETF, moving towards standardized IP-based protocols. In this paper, we will briefly review the history of integrating constrained devices into the Internet, followed by an extensive overview of IETF standardization work in the 6LoWPAN, ROLL and CoRE working groups. This is complemented with a broad overview of related research results that illustrate how this work can be extended or used to tackle other problems and with a discussion on open issues and challenges. As such the aim of this paper is twofold: apart from giving readers solid insights in IETF standardization work on the Internet of Things, it also aims to encourage readers to further explore the world of Internet-connected objects, pointing to future research opportunities.

Keywords: Internet of Things; standardization; survey; constrained devices; IETF ROLL; IETF CoRE; IETF 6LoWPAN; RPL; CoAP; Web of Things

1. Introduction

Internet protocol technology is rapidly spreading to new domains where constrained embedded devices such as sensors and actuators play a prominent role. This expansion of the Internet is comparable in scale to the spread of the Internet in the ’90s and the resulting Internet is now commonly referred to as the Internet of Things (IoT). The integration of embedded devices into the Internet introduces several new challenges, since many of the existing Internet technologies and protocols were not designed for this class of devices. These embedded devices are typically designed for low cost and power consumption and thus have very limited power, memory and processing resources and are often disabled for long-times (sleep periods) to save energy. The networks formed by these embedded devices also have different characteristics than those typical in today’s Internet. These constrained networks have different traffic patterns, high packet loss, low throughput, frequent topology changes and small useful payload sizes.

In the past few years, several innovations were developed to enable the extension of Internet technologies to constrained devices, moving away from proprietary architectures and protocols. Most of these efforts focused on the networking layer: *IPv6 over Low-Power Wireless Personal Area Networks* (RFC 4919) [1], *Transmission of IPv6 Packets over IEEE 802.15.4 Networks* (RFC 4944) [2], IETF *routing over low-power and lossy networks* [3] or the ZigBee adoption of *Internet Protocol Version 6* (IPv6) [4]. These new standards enable the realization of an Internet of Things, where end-to-end IP-based network connectivity with tiny objects such as sensors and actuators becomes possible.

However, it was not global connectivity that was at the basis of the great success of the current Internet, but the World Wide Web and the resulting web service technologies. Today, an embedded counterpart of web service technology is needed in order to exploit all great opportunities offered by the Internet of Things and turn it into a Web of Things. Recently, standardization work has started within the IETF *Constrained RESTful Environments* (CoRE) working group [5] to allow the integration of constrained devices with the Internet at the service level.

With these technologies, it has now become possible to deploy a self-organizing sensor network, to interconnect it with IPv6 Internet and to build applications that interact with these networks using embedded web service technology. Application developers using high-level programming languages can count on these standardized technologies in the realization of a Semantic Web of Things or Sensor Web, which enables data producers and users to publish and access sensor information via web- and standards based interfaces. In this paper, we will briefly review the history of integrating constrained devices into the Internet, with a prime focus on the IETF standardization work. The key realizations of the related IETF working groups will be discussed, complemented with an extensive overview of related research results that illustrate how these novel technologies can be extended or used to tackle other problems and with a discussion on open issues and challenges.

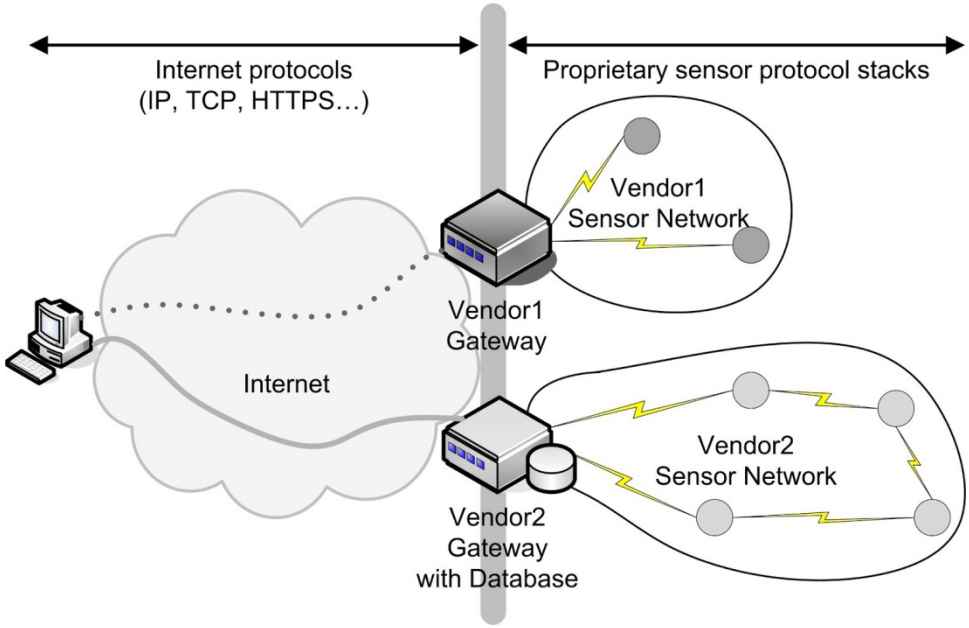
The remainder of this paper is organized as follows. In Section 2, we discuss the evolution from proprietary solutions towards IP-based integration of constrained devices using standardized protocols, introducing the most relevant standardization bodies and groups. In the following sections we will extensively discuss these standardization efforts, related work and open issues and challenges. Section 3 introduces IEEE 802.15.4 as it is the most widely used physical layer and MAC layer in constrained networks, providing the foundations for the networking protocols at the higher layers. From that point on, the focus is shifted to the IETF standardization, discussing IETF 6LoWPAN, IETF RoLL and IETF CoRE extensively in Sections 4, 5 and 6 respectively. Section 7 glues everything together, illustrating how all these standardization efforts contribute to the realization of the Internet or Web of Things. Finally, section 8 concludes this paper.

2. Integration of Constrained Devices into the Internet

In the absence of widely accepted standard protocols for resource-constrained devices, out-of-necessity many vendors developed proprietary protocols to run inside their sensor networks. Connectivity between the Internet and the sensor networks was achieved through the use of vendor-specific gateways or proxies. These gateways have to translate between protocols used in the Internet and proprietary protocols used in the sensor networks. Figure 1 displays two different sensor networks that are connected to the Internet by gateways. Users on the Internet have to connect to the gateways in order to obtain data from the corresponding sensor network. There are several ways how a gateway can handle such user requests. For example, the gateway from vendor 1 translates standard Internet protocols into proprietary sensor protocols and relays the requests to the sensors in its network. The gateway then receives the answers from the relevant sensors by means of the proprietary sensor protocols and sends back the appropriate reply to the user using standard Internet protocols. The gateway offers an API that applications should use in order to create requests that can be understood by the gateway. This approach has the benefit that direct (real-time) interaction with sensor nodes is possible, but only by using a vendor-specific interface. Alternatively, the gateway of vendor 2 contains a database with pre-collected sensor data. When it gets a request from a user on the Internet, it replies directly to the requester using the data in the database. In some cases, the gateway is simply running a web server that makes the data available to the outside world. In this case, existing database technologies can be reused, but the user does not know whether the returned data is coming in real-time from the sensors or whether it is coming from a value that has been previously stored in a database.

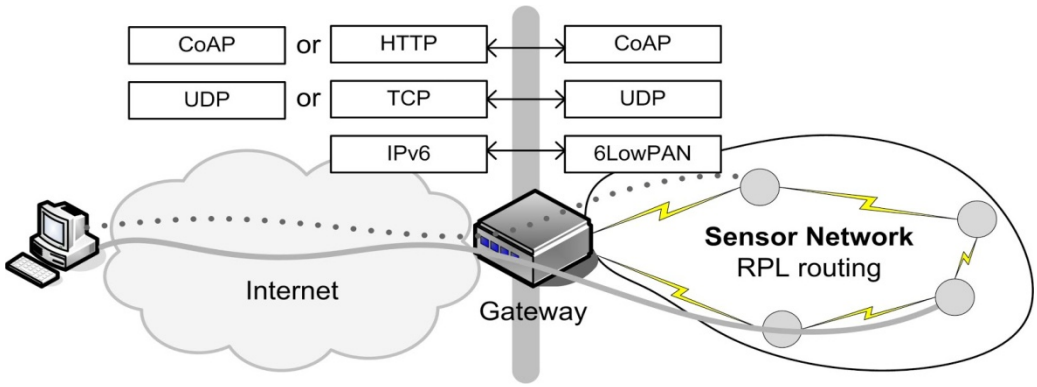
The use of standardized solutions is mostly limited to the use of a standard for the physical layer and MAC layer, although tailored MAC protocols could be used as well. It is clear that such an approach hinders the integration of sensors into the Internet. Little flexibility is offered since users can query the sensors only in the way that is allowed by the gateway. Another disadvantage is the vendor lock-in: gateways and sensors often have to be from the same vendor in order to be compatible. In addition, creating and maintaining the gateway requires significant development effort: often even adding new sensor resources requires making (administrative) changes to the gateway. Finally, due to the lack of real end-to-end connectivity, no real-time interaction with the resource-constrained devices is supported.

Figure 1. Gateways and proprietary protocols are often used to interconnect sensor networks to the Internet.



These limitations in combination with the general understanding that constrained devices will take up a prominent role in the future Internet, raised the need for standardized, open solutions for network communication with constrained devices. To ensure wide adoption, these new solutions have to be interoperable with the most widely used protocols in the Internet, initially IP and, in a later stage, HTTP. To address these needs, the IETF—responsible for the development of high-quality Internet standards—has formed several working groups: *IPv6 over Low Power WPAN* (6LoWPAN) [6], *Routing Over Low Power and Lossy Networks* (ROLL) [3] and *Constrained Restful Environments* (CORE) [7]. The 6LoWPAN group tackles the transmission of IPv6 packets over IEEE 802.15.4 networks, the ROLL group develops IPv6 routing solutions for Low Power and Lossy Networks (LLNs) and the CoRE group aims at providing a framework for resource-oriented applications intended to run on constrained IP networks. Together, these protocols allow the IP-based integration of constrained devices into the Internet in a standardized way, as shown in Figure 2. Due to the popularity of IEEE 802.15.4, this standard is used at the physical layer and the medium access control layer.

Figure 2. Internet protocols are extended to the sensor networks. The Gateway translates between the two standardized protocol stacks.



Similar to the previous approaches, gateways are still used to translate between the protocols used in the Internet and protocols used in the sensor networks, e.g., IPv6 to 6LoWPAN and vice versa. However, due to the use of standardized protocols, many of the disadvantages from the previous approaches are now solved. For example it is now possible to combine sensor devices from different vendors in the same network, or to use a gateway from a different vendor than the vendor of the sensor devices. Flexibility is also improved by this approach as users are not confined to the API offered by the gateway: users can directly query the sensors without the need for the gateway that understands the query or needs to interpret the data. The application payload can now travel directly from the client to the sensor, where it is processed and acted upon. The gateway takes care of the translation between standardized protocols. This end-to-end approach makes adding and removing sensor resources transparent to the gateway and improves interoperability of devices.

3. IEEE 802.15.4

To create a standardized protocol stack for constrained networks and devices, the IETF builds further upon the IEEE 802.15.4 standard. The IEEE 802.15.4 standard is maintained by the IEEE 802.15 working group [8] and defines low-data-rate, low-power, and short-range radio frequency transmissions for *wireless personal area networks* (WPANs). The working group aims to keep the complexity of the standard and the cost of the necessary hardware low, making it suitable for wireless communication among constrained devices such as sensors and actuators. The standard describes a Physical (PHY) layer and a *Medium Access Control* (MAC) sublayer. These will be discussed in the following two subsections. Furthermore, 802.15.4 has proven to be a popular technology for wireless communication in WPANs and a number of examples that have adopted 802.15.4 are listed in the last subsection.

3.1. Physical Layer

The IEEE 802.15.4 physical layer is responsible for (de)activating the radio transceiver, data reception and transmission, channel frequency selection, energy detection within a channel and determining whether or not the communication channel is occupied by a transmission (*i.e.*, Clear Channel Assessment, CCA). IEEE Std 802.15.4–2011 [9] defines a total of 15 different PHY modes. The PHY mode specifies which frequency band and modulation are used for data transmission.

An 802.15.4 compliant radio typically operates at one of the following license-free frequency bands: 868–868.6 MHz (e.g., Europe), 902–928 MHz (e.g., North America) or 2,400–2,483.5 MHz (worldwide, *i.e.*, ISM band). O-QPSK and BPSK are the most commonly used constellations for (de)modulating the signal. Almost all of the defined PHY modes apply a spreading technique that allows spreading out the signal so that it occupies a larger bandwidth. By using these simple constellations and the spreading technique, the communication is more robust and has an increased resistance to interference and (narrow-band) noise even when using a low transmission power.

Depending on the environment and the PHY mode, the communication range varies between 10 and 100 meters. In 802.15.4–2011 the maximum PHY data rate for the ISM band is limited to 250 kbps when using *Direct-Sequence Spread Spectrum* (DSSS) or 1,000 kbps when using *Chirp Spread*

Spectrum (CSS) respectively as a spreading technique. CSS is, however, an optional feature of the standard and in most applications 250 kbps is the maximum PHY data rate.

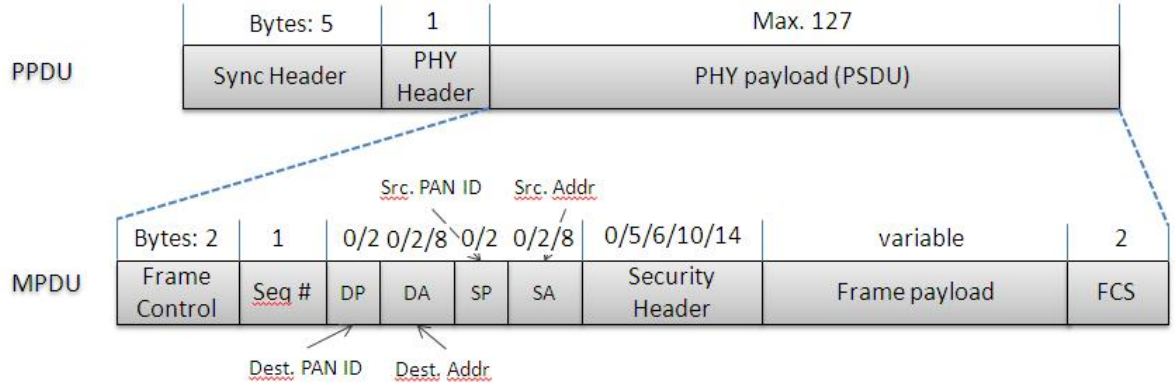
3.2. MAC Sublayer

Because devices that are in each other's proximity share the same communication medium, the access to the medium has to be controlled. When a device wants to send a packet, the MAC sublayer asks the PHY layer's CCA to check whether the medium is occupied. If the CCA indicates that another transmission is currently taking place, the MAC sublayer defers its transmission and waits for a set amount of time before it retries sending the packet. If, on the other hand, the CCA determines the medium to be free then the MAC sublayer transmits the packet immediately.

Apart from medium access control the MAC sublayer also provides acknowledgement of frame reception and validation of incoming frames. The standard defines three different network structures: star topology, mesh topology and cluster tree topology. However, these 802.15.4 topologies are hardly ever used in practice and protocols that run on top of 802.15.4 build their own networks instead. To limit energy consumption, duty cycles are often used by MAC protocols. As a result, transceivers can be in sleeping mode up to 99% of the time, which drastically reduces power consumption and increases operational lifespan.

The *Physical Protocol Data Unit* (PPDU) and *MAC Protocol Data Unit* (MPDU) formats as defined in IEEE Std 802.15.4–2011 are shown in Figure 3. The PPDU for O-QPSK consists of a SYNC header (for receiver clock synchronization), a PHY header (which contains the length of the PSDU) and a PHY payload referred to as the *Physical Service Data Unit* (PSDU). To limit packet error rates, the PSDU is limited in size to 127 bytes. An 802.15.4 MPDU or MAC frame is transported as the PSDU of the PPDU and consists of three parts: (i) the *MAC header* (MHR), (ii) the MAC payload and (iii) the *MAC footer* (MFR). The MHR comprises frame control (indicating the type of frame), sequence number (for referring to frames, e.g., in acknowledgements), addressing information (for source and destination identification) and security-related information, and is of variable length. The frame payload also has a variable length and contains information specific to the frame type. Finally, the MFR is 2 bytes long and contains a *frame check sequence* (FCS) that is calculated by the sender and that is used for frame validation by the receiver.

Figure 3. IEEE 802.15.4 The *Physical Protocol Data Unit* (PPDU) and *MAC Protocol Data Unit* (MPDU) formats.



Depending on the frame type the length of the addressing fields varies. An acknowledgement control frame for instance will not contain any addressing information. For data frames the length depends on which addressing format is being used and whether or not (optional) PAN identifiers of 2 bytes are included. The standard specifies two addressing formats: long 64-bit addresses that are globally unique and short 16-bit addresses that are unique within a PAN. When using the long addressing format, the MHR's length is 19 bytes and the maximum size of the payload is limited to 106 bytes. For the short addressing format the MHR is 7 bytes long and the maximum payload size is limited to 118 bytes. Using link-layer security can further reduce the size available for the payload size by as much as 14 bytes.

3.3. IEEE 802.15.4 Based Solutions

Several higher layer protocols have been defined directly on top of the IEEE 802.15.4 MAC sublayer. For completeness, a very brief overview of the most commonly used non-IETF solutions that have been built on top of IEEE 802.15.4 is given below.

- ZigBee [10] builds upon the physical layer and medium access control defined in IEEE standard 802.15.4 for low-rate WPAN with additional network, security and application software layers. Predefined application services specify which actions a device can take, the main example being "turn the lights on" and "turn the lights off".
- Wireless HART [11] focuses on automation and industrial applications that require real time guarantees. To realize these goals, a time synchronized, self-organizing, and self-healing mesh architecture is used. The standard was initiated in early 2004 and developed by 37 *HART Communications Foundation* (HCF) companies. In April 2010, WirelessHart was approved by the *International Electrotechnical Commission* (IEC) unanimously, making it a wireless international standard as IEC 62591.
- The MiWi protocol stacks [12] are small foot-print alternatives to ZigBee (40K–100K), which makes them useful for cost-sensitive applications with limited memory. Although the MiWi software is free, there exists a unique restriction and obligation to use it only with Microchip microcontrollers.
- ISA100 [13] addresses wireless manufacturing and control systems (developed by the *Systems and Automation Society* (ISA)). They defined ISA100.11a, a wireless networking standard that builds upon IEEE 802.15.4.

Contrary to most of the above solutions, the IETF standards are fully open (no contributor fees are required). To ensure that each device is IP-addressable, an IPv6 layer (6LoWPAN layer) is defined above the IEEE 802.15.4 MAC sublayer. This adaptation layer is discussed in the next section.

4. IETF 6LoWPAN Working Group (IPv6)

The IPv6 protocol has a high overhead and restrictions that make it unsuitable for LLNs such as IEEE 802.15.4 networks. For instance, considering the limited space available for the MAC payload in an 802.15.4 MPDU, the use of a 40-byte IPv6 header would be too excessive. Therefore, the IETF 6LoWPAN (IPv6 for Low-power Wireless Personal Area Network) Working Group was formed to

work on the IPv6 protocol extensions required for such networks where the nodes are interconnected by IEEE 802.15.4 radios [14]. Meanwhile the working group has produced several proposed standards and informational documents regarding these required extensions.

Starting from a well-defined set of assumptions and a problem statement, as defined in the informational RFC 4919 [1], a solution for transmitting IPv6 packets over IEEE 802.15.4 networks was defined, resulting in RFC 4944 [15]: *Transmission of IPv6 Packets over IEEE 802.15.4 Networks*. This document describes the frame format, the methods of link-local address formation and autoconfigured addresses, simple header compression and mesh-under routing for multi-hop IEEE 802.15.4 networks. Subsequent RFCs of the 6LoWPAN working group, *Compression Format for IPv6 Datagrams over IEEE 802.15.4-Based Networks* (RFC 6282) [16] and *Neighbor Discovery Optimization for IPv6 over Low-Power Wireless Personal Area Networks (6LoWPANs)* (RFC 6775) [17], which respectively cover advanced header compression and Neighbor Discovery optimization, have updated RFC 4944. In addition to this, the working group has produced 2 other informational RFCs addressing use case descriptions (RFC 6568) [18] and routing requirements (RFC 6606) [19] and is in the process of finalizing an internet draft on the transmission of IPv6 packets over Bluetooth (draft-ietf-6lowpan-btle-11) [20].

Further, the working group is also expected to collaborate with other organizations (such as IEEE and ISA SP100) and other IETF working groups (such as ROLL) on common interest issues and is mandated with providing implementation and interoperability guides [21].

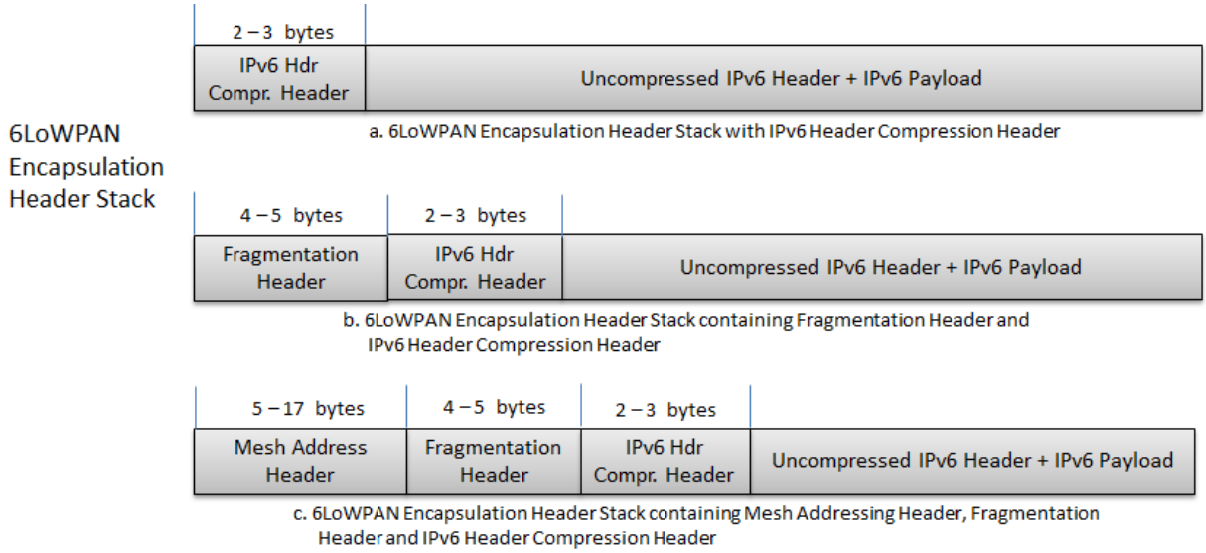
4.1. Key Protocols

As mentioned in Section 4.1, the main objective of the 6LoWPAN working group is coming up with extensions to IPv6 protocols so that IPv6 packets can be transferred in constrained networks such as IEEE 802.15.4. IPv6 forwarding routers, unlike IPv4 routers, do not support fragmenting outgoing packets. This means that the communicating hosts have to send packets with the right size (MTU) supported by the communication links of the router. To meet this requirement hosts may use Path MTU Discovery to find a suitable MTU along the path or just send packets that meet the minimum MTU requirement for all links, which is 1,280 bytes. However, this minimum value is still too big for IEEE 802.15.4 links that have an MTU of 127 bytes for the entire MPDU. To use IPv6 on top of IEEE 802.15.4 networks, a mechanism that allows transmitting IPv6 packets that are larger than the MTU of 127 bytes is required. The solution presented by the 6LoWPAN working group is to use a layer between the network and the data link layers that supports packet fragmentation and reassembly. In addition, the 40 byte IPv6 fixed header takes a significant portion of the already small protocol data unit of the LLNs, leaving little room for IPv6 data payload. To solve this problem different sorts of header compression are proposed by the working group. Further, these fragments have to be routed between the LLN nodes. Layer 2 multi-hop data transmissions should also be addressed in relation to IPv6 adaptation. Accordingly, the 6LoWPAN working group has introduced an IPv6 adaptation layer, named 6LoWPAN Adaptation Layer that lies between the data link layer and the network layer of the protocol stack. The adaptation layer delivers three basic services: packet fragmentation and reassembly, header compression, and data link layer routing (for multi-hop connections).

4.1.1. 6LoWPAN Frames

RFC 4944 states that all LoWPAN encapsulated datagrams are prefixed with an encapsulation header stack where each header in the header stack contains a header type and zero or more header fields. The LoWPAN header may contain the mesh addressing header, fragmentation header, and IPv6 Header compression header, in that order. Figure 4 shows examples of header stacks that might be used in 6LoWPAN. These frames are transported inside the frame payload field of an 802.15.4 data MPDU.

Figure 4. 6LoWPAN encapsulation header stack examples.



Similar to IPv6, the 6LoWPAN headers are added when required. The first byte of the encapsulation header, the dispatch byte, identifies the next header. More specifically, the first three bits of the dispatch byte indicate the next header type while the remaining bits are used for different purposes depending on the header type. Table 1 summarizes the different values for the dispatch byte.

Table 1. Summary of Dispatch Byte values.

First 3 Bits	Header Type	Description
00x	NLAP	This is not a 6LoWPAN frame. This is important for 6LoWPAN to co-exist with other protocols. The remaining 6 bits are ignored.
010	Uncompressed/HC1 Compressed IPv6 Addressing Header	The address type is determined depending on the remaining 5 bits. E.g.: 00001 = uncompressed IPv6 Address 00010 = HC1 Compressed Header
011	IPHC Compressed Header	The remaining 5 bits are added to the rest of IPHC compression header to optimize IPv6 header compression
10x	Mesh Header	The next header is the mesh header. The last bits are used for other purposes related to mesh-under routing.
11x	Fragmentation Header	The next header is a fragment header. The fragment type is determined by the remaining 6 bits. The bit sequence 000xxx indicates first fragment while 100xxx indicates Non-first fragments. The last three bits in both types of fragments will be used for other purposes. The other bit sequences are reserved.

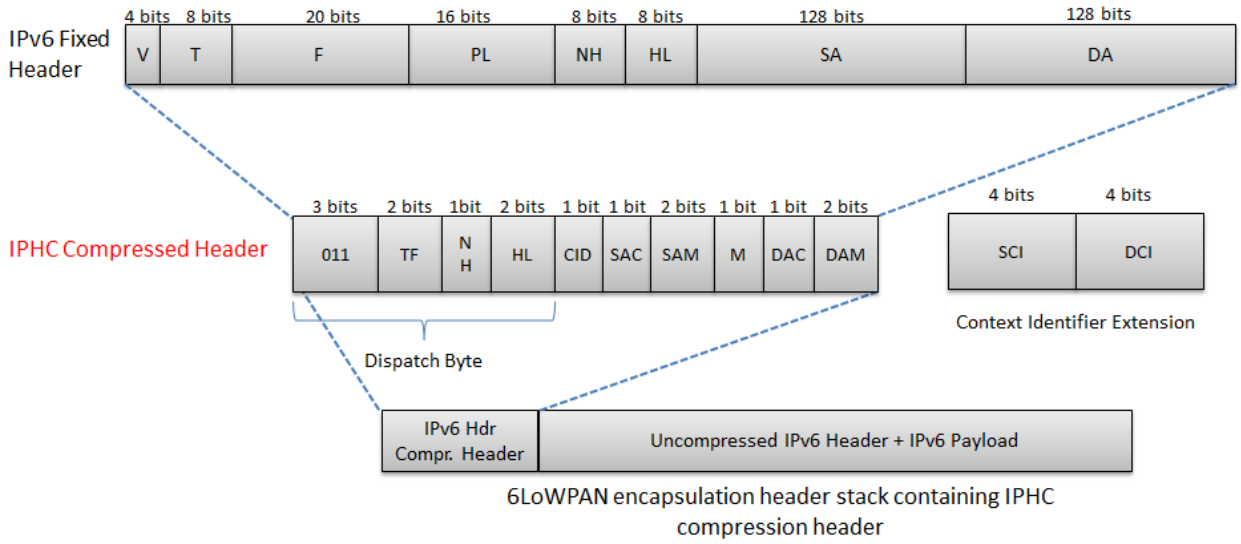
4.1.2. Header Compression

One of the services provided by the 6LoWPAN adaptation layer is header compression. RFC 4944 defines header compression techniques that can be used to compress IPv6 headers and to allow more data from layers on top of IPv6 to be included in a single 802.15.4 frame.

The LOWPAN_HC1, the main compression technique specified in RFC 4944, is optimized for compressing IPv6 packets that contain link-local IPv6 addresses. The technique attempts to reduce the size of the packet by removing common fields (Version, TC, Flow label), inferring the IPv6 addresses from the link-layer addresses in the 802.15.4 header and the static IPv6 link-local prefix (fe80::/64), inferring the IP packet length from the layer 2 frame length (or the fragmentation header) and limiting the values of the next header field to TCP, UDP and ICMP. [22]. The RFC also defines HC2 compression for transport layer compression, which allows compressing UDP, TCP and ICMP. Other transport layer protocols cannot be compressed by HC2. HC1 compression has very low compression factors for global and multicast addresses, which are needed for direct host-to-host interactions between constrained devices and clients as envisioned by the Internet of Things, therefore its use is limited and not further detailed here.

RFC 6282 specifies two new compression mechanisms named LOWPAN_IPHC and LOWPAN_NHC. According to the RFC, LOWPAN_IPHC, uses 13 bits for compression (the last 5 bits of the dispatch byte and an additional byte) and an extra 8 bits to store context information, when necessary. Figure 5 shows the IPHC header format and Table 2 summarizes the address compression fields.

Figure 5. IPHC Compression.



IPHC is based on a number of basic assumptions about common 6LoWPAN communication cases. The first assumption regards commonly used fields. It is assumed that IPv6 header fields such as Version, Traffic Class and Flow Label have fixed values and do not have to be transmitted. Therefore, IPHC totally ignores the 4 bit version and attempts to compress the Traffic Class and Flow Label into the TF bits (2 bits). By assuming hop limits will be set to well-known values such as 1, 64 and 255 by the host, IPHC compresses the hop limit from 8 bits to 2 bits.

Table 2. Summary of Dispatch Byte values.

Field	Description
Context ID Extension (CID) (1bit)	0 = No Additional Context Identifier Extension is used. 1 = An additional 8-bit field follows the DAM field
Source Address Compression (SAC) (1 bit) **Dest. Address Compression (DAC) (1 bit)**	0 = Stateless source/destination address compression 1 = Stateful, context based source/destination address compression
Source Address Mode (SAM) **If SAC = 0**	00 = The full 128 bits address is sent inline 01 = The last 64 bits are sent inline 10 = The last 16 bits are sent inline. 11 = The entire source address is elided
If SAC = 1	00 = The unspecified address, :: . Nothing is sent inline 01 = 64 bits are carried inline 10 = 16 bits are carried inline 11 = The address is fully elided
Multicast Compression (M)	0 = Destination address is not a multicast address 1 = Destination address is a multicast address
Destination Address Mode (DAM) **If M = 0 and DAC=0**	Same as SAM with SAC = 0
If M = 0 and DAC = 1	Same as SAM with SAC = 1
If M = 1 and DAC = 0	00 = The full address is sent inline 01 = Only 48 bits are sent inline 10 = Only 32 bits are sent inline 11 = Only 8bits are sent inline
If M = 1 and DAC = 1	00 = Only 48 bit s are sent inline. 01,10,11 = Reserved

Assuming that addresses could be generated from link-layer addresses and that mostly link local addresses are used inside LLNs, IPHC attempts to compress the IPv6 address fields. To further improve the efficiency of compression of global and multicast addresses, IPHC uses context information. The *Context ID* Extension bit (CID) bit indicates whether an additional 8-bit Context Identifier Extension field is added or not. IPHC supports stateless and stateful methods of addresses compression. The *Source Address Compression* (SAC) and *Destination Address Compression* (DAC) bits indicate which of these two compression methods is used for the source and destination address, respectively. The *Source Address Mode* (SAM) bits in combination with the SAC bit determine how many of the source address bits are actually elided and how many bits are sent in-line. The number of bits sent inline can be 128, 64, 16 or 0 bits for stateless compression and 64, 16, or 0 bits for stateful compression. In all cases, the bits that are elided are assumed to be calculated from the link-local prefix, link layer address and stored context. The destination address compression varies based on the address type. Multicast destination addresses are indicated by the M bit in the compression header. Unicast destination addresses are compressed in the same way as unicast source addresses. However, multicast addresses follow a different rule based on the DAC bit. Accordingly, 128, 48, 32, or 8 bits has to be sent inline when multicast destination addresses are compressed. The 8 bit context identifier

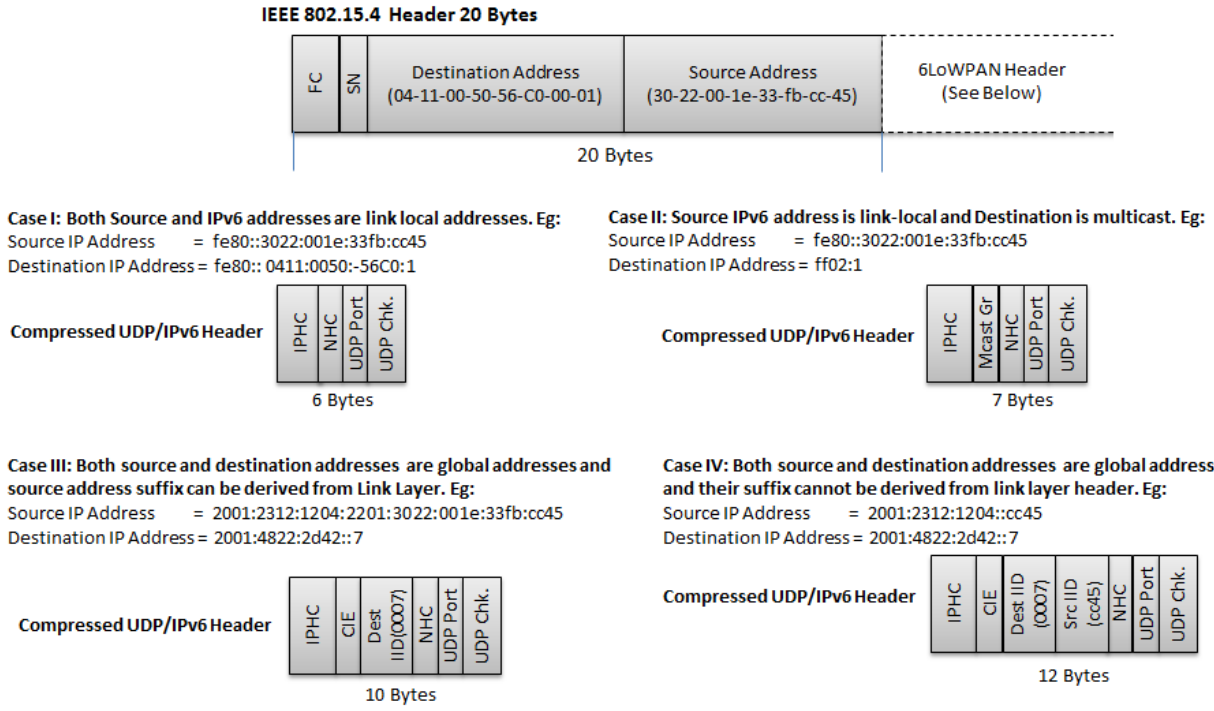
is added only if the SAC and/or DAC bits indicate its presence. If it exists, the first 4 bits indicate store context for the source address while the remaining bits store destination address context. The RFC does not specify what is stored in these fields nor how communicating parties exchange this information.

The IPv6 Payload Length field can be inferred from the fragment header or from the link layer header and has to be fully elided.

As was already mentioned HC2 can only compress UDP, TCP and ICMPv6 headers. To alleviate this issue LOWPAN_NHC introduces a variable length next header identifier which could be used for future next header compressions.

Figure 6 illustrates the IPHC header compression when link-local, global and multicast IPv6 addresses are used for communication.

Figure 6. Internet Protocol Version 6 (IPv6) Header Compression Example.



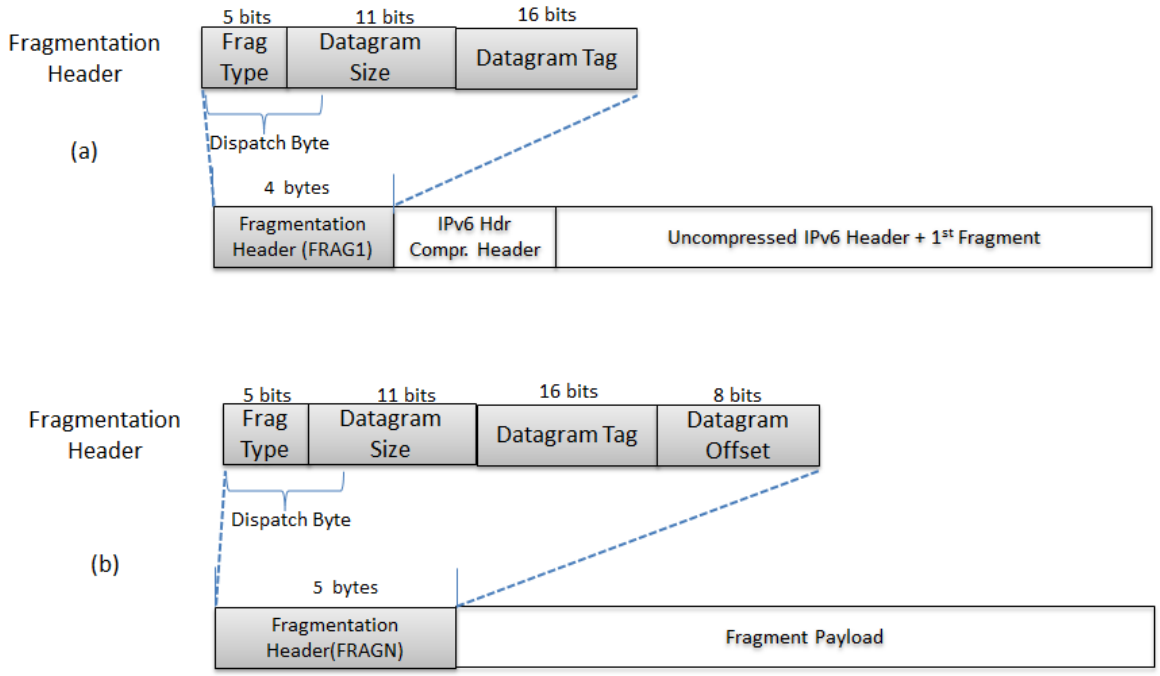
In case I, since both source and destination IP addresses are link-local unicast addresses, the prefix is fixed (*i.e.*, fe80::/64) and the suffix can be inferred from the IEEE 802.15.4 source and destination addresses. As shown in the figure, the entire IPv6/UDP header can be compressed from 48 bytes to just 6 bytes. The second case illustrates IPHC compression when the destination IP address is a multicast address. Here the IPv6/UDP header is compressed to 7 bytes by sending only the multicast group inline and deriving all other information from the IEEE 802.15.4 header. Cases III and IV are typical to IoT interactions where both source and destination IP addresses are global unicast addresses. In case III, the source suffix can be derived from the IEEE 802.15.4 source address and, hence, does not have to be sent in-line. The destination suffix however cannot be derived from the IEEE 802.15.4 destination address and has to be sent uncompressed. In case IV, the suffixes of both source and destination addresses cannot be derived and have to be sent inline. In cases III and IV, the IPv6 address prefixes and other information is derived based on the shared context information that is stored in the

Context Identifier Extension (CIE) byte. In case 3 and case 4, the IPv6/UDP headers are compressed from 48 bytes to 10 and 12 bytes respectively.

4.1.3. Fragmentation

The other service provided by the 6LoWPAN adaptation layer is fragmentation and reassembly. Fragmentation is only required when the entire IPv6 packet cannot fit in a single IEEE 802.15.4 frame, *i.e.*, when it is larger than the available space for the MPDU payload (typically 106 bytes, see Section 3.2). According to RFC 4944, fragmentation breaks a single IPv6 packet into smaller pieces and a fragmentation header is included in every fragment. The document further specifies using two different types of fragmentation headers. The fragment header of the first fragment contains only the datagram size (11 bits) and datagram-tag (16 bits) fields, while subsequent fragments of the same IPv6 packet also include the datagram-offset (8 bits) field. Datagram size is the length of the entire unfragmented IPv6 packet, datagram-tag identifies to which datagram (*i.e.*, packet) a particular fragment belongs. Therefore, these values have to remain the same for all fragments of a single IPv6 packet. The datagram-offset indicates the offset of the fragment from the first fragment. Figure 7 shows the 6LoWPAN fragmentation header.

Figure 7. 6LoWPAN Fragmentation Header. (**a**) 6LowPAN encapsulation header stack for the first fragment, containing the Fragmentation header (FRAG1) and IPv6 Header Compression header (**b**) 6LowPAN encapsulation header stack for subsequent fragments, containing the Fragmentation header (FRAGN).



4.1.4. Mesh-Under Routing Support

Routing involves calculating best paths (according to some metric) to a destination in a multi-hop network. If a single link layer technology (such as IEEE 802.15.4) is used at layer 2, the path

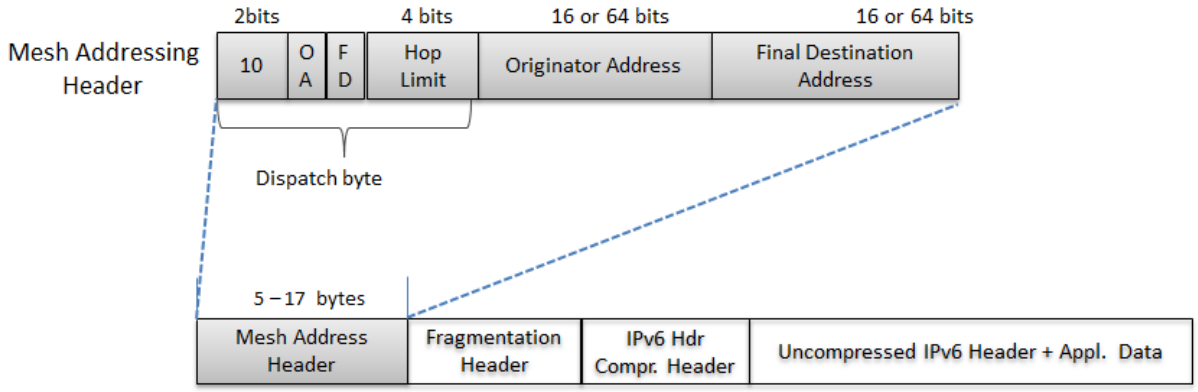
calculation could be done at either layer 3 or layer 2. Layer 3 routing is usually referred to as route-over routing while forwarding at layer 2 is called mesh-under routing.

Layer 3 routers build and maintain a routing-table which contains the next-hop to all known destination networks based on their network prefixes. This means that when a packet arrives at a router, the link-layer encapsulation is removed and the destination address in the IPv6 header is used to do a longest prefix match in the routing table to determine the next hop. Once the next hop router is known, the router re-encapsulates the packet with layer 2 headers and trailers. The layer 2 addresses indicate the current router as a source and the next hop router as the destination while the source and destination IP addresses remain unchanged. This is done at every forwarding router along the path of the packet.

Mesh-under routing on the other hand, uses link layer addresses to make forwarding decisions. Every forwarding layer 2 router along the path of a packet is expected to maintain its own forwarding-table based on link layer addresses in order to make forwarding decisions based on these addresses. Just as in route-over routing, four addresses are required to forward the packet at an intermediate node—the originator address, the final destination address, the current forwarding router address and the next hop router address. The IEEE 802.15.4 source and destination MAC addresses of the frame indicates the current forwarding router and the next-hop router, respectively. A way of transmitting the originator and final destination addresses is required to fully support mesh-under routing.

RFC 4944 introduces the Mesh Address Header (Figure 8) for this purpose. The mesh address header contains the dispatch byte and the originator and final destination link layer addresses. The OA and FD bits in the header indicate which of the two 802.15.4 addressing formats are used for the Originator and Final Destination address fields: short 16-bit or long 64-bit addresses. The hop count takes up an additional 4 bits, supporting up to 14 hops. The value 0xF is reserved to indicate that one more byte follows to support more than 14 hops.

Figure 8. 6LoWPAN Mesh Addressing Header.



6LoWPAN encapsulation header stack containing Mesh address header, fragmentation header and IPv6 Header compression header

As an example of mesh-under routing, assume that node A sends a packet to node B which is located some hops away from node A in an IEEE 802.15.4 network. First, node A, construct a mesh addressing header with its own link-layer address as Originator address and the link-layer address of node B as Final Destination address. Based on its layer 2 "routing-table", node A identifies its next hop

router (node C). Accordingly, it puts node C's link layer address as the destination address and its own link-layer address as the source address in the 802.15.4 Mac Header. The resulting frame contains the following addresses in the 6LoWPAN mesh addressing header: Originator address (node A), final destination address (node B) and in the IEEE 802.15.4 MHR: Source address (node A) and destination address (node C). Once the router, node C, receives the frame, it checks the Mesh Addressing Header to check whether it is the final destination. Since node B is the final destination, node C will also forward the packet to its next hop. After determining the next hop, node C re-encapsulates the frame with its own link layer address as the source and the next hop's link layer address as the destination address in the MHR. This process continues until the packet reaches node B. The addresses in the 6LoWPAN Mesh Addressing Header remain the same along the path of the packet.

4.2. Implementation and Evaluation

4.2.1. Implementation

Many organizations and companies are incorporating the 6LoWPAN adaptation layer in their protocol stacks. Additionally, there are also a couple of network simulators that support 6LoWPAN. Table 3 summarizes the features of six different implementations of 6LoWPAN.

Table 3. 6LoWPAN implementations.

Implementation	Operating System /Simulator	License	RFC 4944	RFC 6282	RFC 6775
SICSLOWPAN	ContikiOS/Cooja Simulator	Open Source	X	x	x
BLIP (Berkley Low-power IP)	TinyOS	Open Source	X		
Arch Rock 6LoWPAN	TinyOS	Open Source	X		
NanoStack 6lowpan	FreeRTOS	Open Source	X	x	x
Hitachi	-	Commercial	X		
NS-3	Simulator	Open Source	X		

4.2.2. Evaluation

Very few performance evaluations of the 6LoWPAN adaptation layer have been published thus far. The most notable performance evaluation of the 6LoWPAN protocol was made by [23] in which the author qualitatively and quantitatively explains the advantages of 6LoWPAN as compared to different industry standards. According to [23], in terms of memory footprint, ease of deployment, scalability, energy efficiency and other features, 6LoWPAN has significant advantage over other standards, such as Zigbee.

When analyzing 6LoWPAN in detail, it can be seen that the adaptation layer solves many of the issue relating to supporting IPv6 on constrained devices and LLNs. One of the functions of the adaptation layer is fragmentation and reassembly. In order to fit large IPv6 packets into the small MTU of IEEE 802.15.4 networks and improve efficiency of communication, packets must be fragmented at the adaptation layer before it is passed on to the 802.15.4's MAC sublayer. Similarly, during reception,

these fragments should be reassembled at the same layer before passing them onto upper layer protocols. This approach has enabled smart objects to participate in any IPv6 based communication with some extra processing overhead. In [24], the authors attempted to test performance of 6LoWPAN on TelosB and MikaZ motes. The authors tried to measure the impact of packet size on RTT and packet loss by increasing the packet size. As expected, the RTT increased for larger packets due to fragmentation and reassembly of packets.

Both route-over and mesh-under routing mechanisms can be used in 6LoWPAN networks. Due to fragmentation and header compression at the 6LoWPAN adaptation layer, packets in a network that uses route-over routing have to be reassembled at every intermediate node to get the destination IP address for determining the next hop. These fragments have to go through compression and fragmentation processes before they are forwarded to the next hop router. This approach requires extra buffer space to store all fragments of a particular packet at each intermediate node. As memory is one of the constrained resources, this may not be an optimal solution for Internet of Things nodes. On the other hand, mesh-under routing may solve the issue of reassembly and fragmentation by routing the fragments independently. To the best of our knowledge, there is no performance comparison of the two routing approaches in literature. Also, no standardized solution for mesh-under routing has been proposed; as opposed to the RPL route-over routing protocol that is being defined in the IETF ROLL working group. However, examples of mesh-under routing can be found in literature, e.g., in [25].

IPv6 headers have a minimal length of 40 bytes. This is a too large overhead for the 127 byte MTU of IEEE 802.15.4 networks. 6LoWPAN uses different mechanisms to compress IPv6 headers by removing redundant information from the headers and inferring values from the link layer information. The original compression mechanism proposed in RFC 4944, called HC1, compresses source and destination addresses by assuming that they are link local addresses. If addresses are global, they have to be sent uncompressed. This makes HC1 compression less efficient for IoT solutions which mostly involve global IPv6 addresses. In addition to this, the transport layer protocol compression mechanism, HC2, compresses only limited numbers of protocols. New compression mechanisms, called IPHC and NHC, were introduced in RFC 6282. These mechanisms are significant improvements to the HC1 and HC2 compression mechanisms and new implementations are urged to use the new compression techniques instead of HC1 and HC2. To our knowledge, no detailed evaluations regarding the footprint and performance of these compression mechanisms have been published in the literature to date.

4.3. Leveraging upon 6LoWPAN to Realize the IoT

As described in the previous sections, the core specifications of 6LoWPAN are RFC 4944, RFC 6282 and RFC 6775. As 6LoWPAN is part of a full protocol stack, it is used in combination with different protocols, from the physical to the application layer. Further standardization work and research efforts are therefore focusing on, amongst others, improvements to the core specifications, extensions for non-IEEE 802.15.4 networks and the adoption of 6LoWPAN in practical use cases.

4.3.1. Improvements to Core Specifications

In addition to the core specifications, other sub-protocols are being discussed in the working group. As was already mentioned, the improved compression mechanism specified in RFC 6282 introduces a

next header identifier to enable any arbitrary next header compression. The *Generic Compression of Headers and Header-like payloads* (draft-bormann-6lowpan-ghc-05) [26] is one of the internet drafts that utilizes the identifier to perform compression of arbitrary headers.

Adaptation Layer Fragmentation Indication is another area that is under discussion. The draft, (draft-bormann-intarea-alfi), proposes a mechanism to indicate the presence of fragmentation at the adaptation layer and to indicate preferred MTU for the communication. This draft aims at improving application performance by limiting packet sizes to the smallest possible size to avoid fragmentation on link layers with small MTU values.

4.3.2. 6LoWPAN over Non IEEE 802.15.4 Technologies

The working group also adopted internet draft-ietf-6lowpan-btle, which applies 6LoWPAN technology to Bluetooth Low Energy. This draft is expected to be a companion of RFC 4944 by removing unnecessary features such as Mesh header and Fragmentation. This extension is the first draft that extends 6LoWPAN to non-IEEE 802.15.4 networks. Draft-mariager-6lowpan-v6over-dect-ule, is another draft proposed to extend 6LoWPAN to another non-IEEE 802.15.4 technologies. The draft proposes 6LoWPAN technology for DECT ULE (*Digital Enhanced Cordless Technology Ultra Low Energy*). The last two drafts indicate that the working group is considering link-layer technologies other than 802.15.4 to use with 6LoWPAN. As such, it is explored how 6LoWPAN can operate over heterogeneous low-power technologies, in a similar way as how IP can operate over different underlying technologies.

4.3.3. Adoption of 6LoWPAN in Real Life Use Cases

As 6LoWPAN is a key component in order to realize the IP-based integration of constrained devices, it is used in a multitude of projects, exploring a wide range of use cases such as smart infrastructures, smart buildings, smart environments, smart cities, ... In all cases, constrained devices, forming 6LoWPAN networks, are used to collect information from the real world and this information is used to generate intelligence and make the world around us smarter. Enumerating them all would be impossible, but in order to illustrate some application domains a limited number of specific examples is listed.

The Smart Energy and Home Automation: Restful Architecture (Sahara) project aims at using web-services for smart energy and home automation. The project uses 6LoWPAN on IEEE802.15.4, CoAP and other IETF standardized protocols [27]. Similarly, the European Union FP7 HOBNET project deploys an IPv6/6LoWPAN infrastructure to be used for the automation and energy management of smart and green building [28]. Another example is the FP7 Outsmart project [29], where 6LoWPAN networks are used for instance to optimize waste management [30]. CALIPSO is another FP7 project that aims at building IP-connected networks of smart object in the area of smart infrastructure, smart cities and smart toys by utilizing IETF standards (including 6LoWPAN) and other start-ups [31]. Finally, in [32] 6LoWPAN networks are being used for livestock monitoring applications and other similar use cases.

4.3.4. Other Efforts

Management of (6LoWPAN) networks is another important aspect of 6LoWPAN that is being investigated. In line with this, 6LoWPAN MIB (draft-schoenw-6lowpan-mib) [33] is being proposed. The draft demonstrates a JSON format using a version of the Management Information Base database for management purposes along with the normal SMIv2 format.

From the previous discussion, it has become clear that addressing is also a key aspect when extending IPv6 to the constrained world through the use of 6LoWPAN. In this area, other IETF working groups are also conducting research activities in the area. The IETF Working group AUTOCONF (*Ad hoc* Network Autoconfiguration) is also working on Neighbor Discovery and stateless address autoconfiguration. Finally, as 6LoWPAN is meant as an enabler to bring IPv6 functionality to constrained networks, many higher layer protocols, such as routing, will run on top of 6LoWPAN networks. During the design of these solutions, the relation and interaction with 6LoWPAN is considered. For example, the routing protocols and mechanisms being developed by ROLL working group can also be seen as other 6LoWPAN related efforts [22]. The ROLL working group is discussed in the next section.

4.4. Research Challenges

One of the key issues in connecting constrained networks with the Internet is fragmentation, caused by the 127 byte MTU of IEEE 802.15.4 versus the minimum MTU of 1280 bytes of links in IPv6 networks. Avoiding low-level fragmentation is important and requires knowledge about the MTU (*i.e.*, via discovery), knowledge about other protocols (type of routing) and interaction with the applications (and the potential use of fragmentation at this layer, such as block transfers in CoAP as will be discussed later). A good understanding and evaluation of the impact of fragmentation and solutions to avoid it is needed. For example, when using route-over solutions, additional buffer space is needed for packet reassembly and fragmentation at each intermediate node, since the information needed for routing can only be found in the first fragment [34]. The need of this additional buffer space can be avoided, by providing a small buffer to store the first few bytes of each packet. As the delivery of the first fragment could be handled by looking at the destination address, subsequent fragments may be routed based on the datagram-tag which is stored in the small buffer we set aside. Different mechanisms may be considered for first fragments of a packet arriving late.

Regarding compression, advanced compression techniques have been proposed, including the compression of next headers. Until now, a thorough evaluation of the performance of these mechanisms for various communication patterns, address sizes and setups is missing. Transport layer compression methods are currently defined for UDP only. The 6LoWPAN working group is discussing generic next header protocol compression. Nevertheless, no compression mechanism is defined for TCP and ICMPv6 yet. Such a mechanism for TCP could be considered irrelevant for the majority of constrained devices, since they typically use UDP with an application-layer protocol such as CoAP that provides some of the missing features of TCP. However the use of CoAP on top of TCP has been explored [35].

There are still a number of other issues to be investigated. Layer 3 route-over and layer 2 mesh-under routing protocols are supported on 6LoWPAN networks. Mixing these protocols in a given 6LoWPAN network might be a path worth exploring [23].

Finally, the 6LoWPAN working group has already published a proposed standard document relating to Neighbor Discovery. Yet, Secure Neighbor Discovery is still unexplored territory. IPv6-like security up to level of constrained devices is being researched. Despite several efforts, key distribution to constrained devices remains one of the biggest challenges so far.

5. IETF ROLL Working Group

5.1. Group Description and Key Protocols

5.1.1. Description

The specific properties of LLNs imply specific routing requirements for these networks. The ROLL working group [36] focuses on building routing solutions for LLNs because evaluation of existing routing protocols like OSPF, IS-IS, AODV, and OLSR indicate that they do not satisfy all of the specific routing requirements. More specific, the working group focuses on industrial (RFC 5673) [37], connected home (RFC 5826) [38], building (RFC 5867) [39] and urban sensor networks (RFC 5548) [40] for which different routing requirements were specified.

The working group focuses only on an IPv6 routing architectural framework while also taking into account high reliability in presence of time varying loss characteristics and connectivity with low-power operated devices with limited memory and CPU in large scale networks. The main realization of this working group is the design of the IPv6 route-over Routing Protocol for LLNs, also called RPL, which covers the routing requirements of all the application domains.

5.1.2. IPv6 Routing Protocol for Low Power and Lossy Networks

With the specification of RPL in *RPL: IPv6 Routing Protocol for Low-Power and Lossy Networks* (RFC 6550) [41], the IETF has specified a proactive "route-over" architecture where routing and forwarding is implemented at the network layer, according to the IP architecture. The protocol provides a mechanism whereby multipoint-to-point, point-to-multipoint and point-to-point traffic are supported. Although RPL was specified according to the routing requirements for LLNs, its use is not limited to these applications. RPL routes are optimized for traffic to or from a root that acts as a sink/root for the topology.

The functioning of the RPL routing protocol is based on the construction of a *Directed Acyclic Graph* (DAG), which consist of one or more DODAGs (*Destination Oriented DAGs*), for each sink/root a DODAG. The position of an individual node in a DODAG is determined by the rank of the node. This rank is calculated based on the *Objective Function* (OF), which defines how to translate one or more metrics and constraints, defined in *Routing metrics used for path calculation in low power and lossy networks* (RFC 6551) [42], into a rank. The OF also specifies how a node has to select his parent. Different DODAGs based on a same OF are represented by a RPLInstanceID. More details concerning OFs can be found in *Objective Function Zero for the Routing Protocol for Low-Power and*

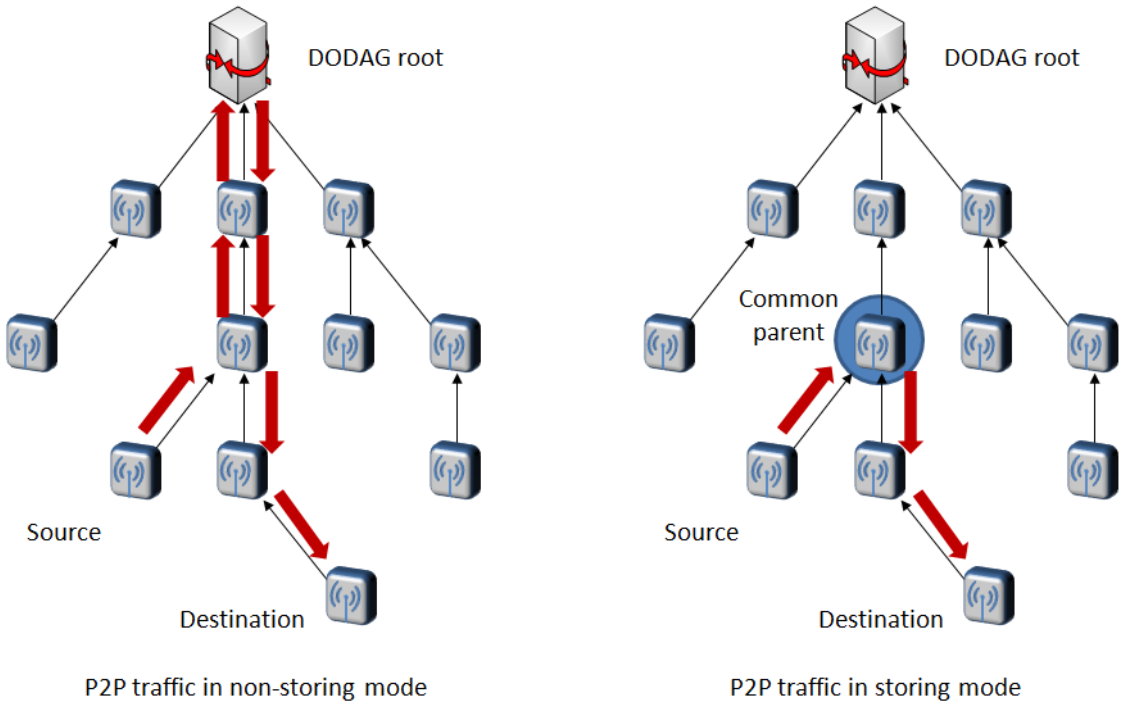
Lossy Networks (RPL) (RFC 6552) [43], *The Minimum Rank with Hysteresis Objective Function* (RFC 6719) [44] and RFC 6551.

When a new node joins a RPL network, it first listens to receive *DODAG Information Object* (DIO) messages, which are broadcasted periodically when the trickle timer [45] of neighboring nodes fires. When no DIO message is received, the node will broadcast a *DODAG Information Solicitation* (DIS) message, which will force the neighboring nodes to send a DIO message. Based on the information of the DIO message from the neighboring nodes, the *Objective Function* (OF) will select the preferred parent. When every node in the network has selected a preferred parent, the DODAG (for a specific OF and for each sink) has been constructed. Routing from a node towards the sink is established by forwarding each message, for collection by the sink, to each nodes parent, until packets reach the sink.

Downward routes (root to leaf) are constructed using *Destination Advertisement Object* (DAO) messages. When a node has selected a preferred parent it will send a DAO message to his preferred parent, which will be forwarded, via the parent's parent, towards the root. Two modes of operation are supported: Storing (fully stateful) or Non-Storing (fully source routed) mode.

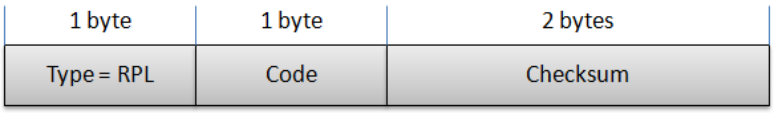
The mode of operation, for construction of downward routes will also influence the operation for point-to-point routes (Figure 9). In the Non-Storing case, the packet will travel all the way to a DODAG root before traveling down. In the Storing case, the packet may be directed down towards the destination by a common ancestor of the source and the destination prior to reaching a DODAG root. If the destination is on the route towards the root, the destination node of course will not forward the message.

Figure 9. Packet flow for point-to-point traffic between two nodes in an RPL network.



RPL messages are encapsulated into a new ICMPv6 message, defined in *Internet Control Message Protocol (ICMPv6) for IPv6 Specification* (RFC 4443) [46]. For RPL control messages this results in a message structure illustrated by Figure 10, consisting of an ICMPv6 header followed by the actual message body (base) and some (optional) options.

Figure 10. ICMPv6 header for RPL control messages.



The code field identifies the type of the RPL control messages. Table 4 gives an overview of the different codes and their corresponding RPL message type.

Table 4. Code fields in RPL ICMPv6 messages.

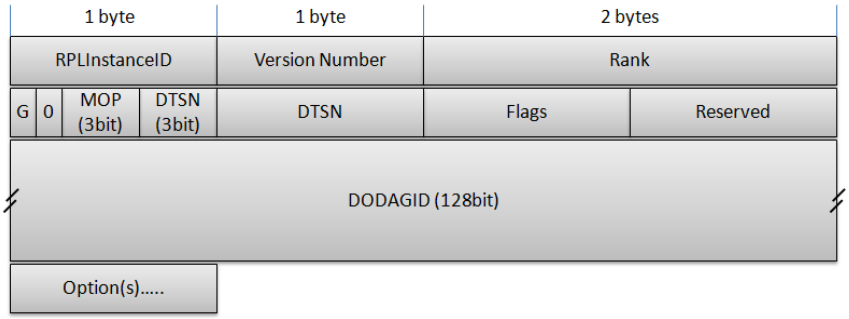
Code Field	RPL Message Type
0x00	DODAG Information Solicitation (DIS)
0x01	DODAG Information Object (DIO)
0x02	Destination Advertisement Object (DAO)
0x03	Destination Advertisement Object Acknowledgment
0x80	Secure DODAG Information Solicitation
0x81	Secure DODAG Information Object (DIO)
0x82	Secure Destination Advertisement Object (DAO)
0x83	Secure Destination Advertisement Object Acknowledgment

When the high order bit (0 × 80) in the code field is set, it indicates that security is enabled. In this case between the header and the base field a security field will be added.

In a DODAG Information Solicitation message the base field will consist of a reserved 1 byte flag field, a 1 byte reserved field. Together, with the flag and reserved field, unassigned bits must be set to zero on transmission and must be ignored on reception.

The DODAG Information Object (Figure 11) contains information that allows receiving nodes to learn and configure for joining a DODAG. The message can indicate if the DODAG is grounded (1 bit G-field), which *mode of operation* (MOP) is followed and how preferable the root of this DODAG is compared to other DODAG roots (Prf-field). The message also contains the version number, the RPLInstanceID, DODAGID (128-bit IPv6 address that uniquely identifies a DODAG), a number of flags and the rank of the sending node.

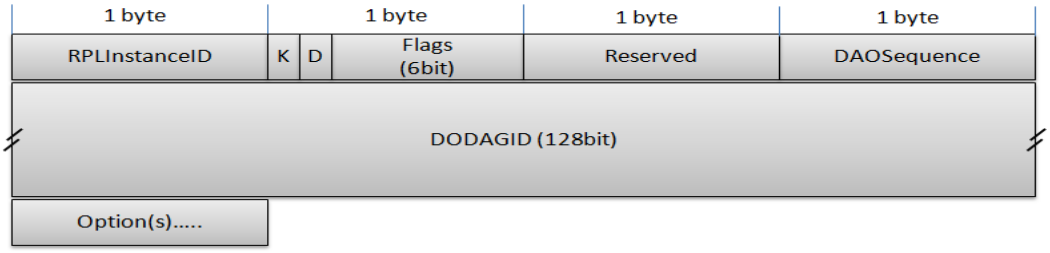
Figure 11. The DODAG Information Object (DIO) Base Object.



The *Destination Advertisement Object* (DAO) message is illustrated in Figure 12. The RPLInstanceID, learned from the DIO, is copied into the DAO message. When the sender expect the receiver to send a DAO-ACK the K-flag is set. To indicate the presence of the DODAGID field, the

D-flag is used. The DAOSequence indicates the incremented sequence number incremented every time a node sends out a unique DAO message.

Figure 12. The DAO Base Object.



5.2. Implementation and Evaluation

5.2.1. Implementation

Already during the standardization process of the RPL protocol, the draft of the protocol was implemented in several platforms and simulators. An overview of the different implementations is presented in Table 5.

Table 5. Implementations incorporating RPL.

Name	OS	Protocol Version	Notes (Extensions, ..)
TinyRPL [47]	TinyOS	draft-ietf-roll-rpl-17	- uses BLIP 2.0 - only storing mode - only single RPLInstanceID - security options not supported - only telosb and epic platform support
ContikiRPL [48]	Contiki	RFC 6550	by default enabled on Tmote sky platform
OpenWSN [49]	OpenWSN	RFC 6550	
Nano-RK [50]	Nano-RK	draft-ietf-roll-rpl-07	
NanoQplus [51]	NanoQplus	draft-ietf-roll-rpl-13	

In Table 6 an overview is given of the network simulators that implement the RPL protocol. In the table the TOSSIM simulator, standard for TinyOS, is not mentioned because TOSSIM requires micaz support, which is not available for TinyRPL. Several research papers such as [52] and [53] also implemented the RPL protocol into the WSNet simulator. In [53] this implementation is based on draft-ietf-roll-rpl-05.

Table 6. Simulators incorporating RPL.

Name	Language	Protocol version	Notes (extensions,..)
Cooja [54]	C with limited libs	RFC 6550	MSPsim (TinyOS + Contiki)
NS-3 [55]	C++ and Python	draft-ietf-roll-rpl-19	
OMNET++/Castalia [56]	C++ (wrapped together with NED)	draft-ietf-roll-rpl-19	
J-SIM [57]	Tcl/Java	draft-ietf-roll-rpl-19	EU-funded FP7 ICT-257245 VITRO project

5.2.2. Using the Protocol

RPL is designed to be widely applicable; therefore many configuration options are available. Based on experience, *Recommendations for Efficient Implementation of RPL* (draft-gnawali-roll-rpl-recommendations-04) [58] presents different design choices and configuration parameters that envision an efficient RPL implementation and operation. In *Performance Evaluation of the Routing Protocol for Low-Power and Lossy Networks (RPL)* (RFC 6687) [59], an overview and evaluation is given of how the protocol can handle two different use cases (a small outdoor deployment of sensor nodes for building automation and a large-scale smart meter network) to meet the desired requirements.

It is worth noting that a difference in design choices and configuration parameters can lead to interoperability problems. In [60] the interoperability between ContikiRPL and TinyRPL is investigated. The paper shows that, despite having two good performing independent implementations, a combination can lead to large scale differences in behavior when combining them in a mixed set up. It is also illustrated that subtle difference in lower layers can affect the system performance in unexpected ways.

5.2.3. Sensor-to-Sensor Traffic

The routing for sensor-to-sensor communication is based on the communication paths between the root and the sensors. These paths are initiated by the communication of DAO-messages to their preferred parent. According to [61] this path setup can lead to congestion on the nodes around the root. In addition, the limitations on packet sizes for constrained devices for non-storing mode introduce the risk of fragmentation for paths with multiple hops. In contrast, for operation in storing mode, the limited memory influences the number of paths a node close to the root can store.

The authors of [62] state that the importance of point-to-point traffic flows in low-power and lossy networks is underestimated. In the paper it is demonstrated for a network consisting of about 1,000 nodes that the shortest cost *peer-to-peer* (P2P) routes performs significantly better than the current RPL standard (using up and down P2P routes). This illustrates the need of additional point-to-point routing mechanisms. This conclusion is also confirmed by [63]. In this paper a solution is provided, called P2P-RPL, which extents RPL and performs better in a network of 27 fixed nodes running Contiki that has an average node degree of 4.39. While data packets in standard RPL traverse approximately 5 links on average, the links that are traversed when using P2P-RPL are halved for the same network. For even deeper nested DODAG trees, even higher gains are expected. P2P-RPL also decrease the traffic load around the sink: in storing mode with standard RPL 74,53% of the routes traverse the root, while for P2P-RPL this is only 16,03%.

5.2.4. Multipoint-to-Point Traffic

In many IoT use cases, different sensors send their sensed data to a central sink. For this type of traffic RPL requires very limited control overhead. This overhead is further reduced by the use of the Trickle timer [45], which decreases the frequency of the sending of DIO messages when the network is

stable. Also the responsibility of maintaining routes from leafs to the sink is delegated to each node by only selecting a parent which is closer (according to the OF) to the sink.

5.2.5. Multicast

Possibilities for the usage of RPL routing of multicast messages are stated in *Multicast Protocol for Low power and Lossy Networks (MPL)* (draft-ietf-roll-trickle-mcast-02) [64]. According to [65] the proposed draft solution has the advantage that it will work without modifications and reliability is increased by the per datagram state information maintenance, but it also has a number of drawbacks. A first drawback is that, instead of storing only destination information, for each packet a state has to be stored, which can result in scalability issues. The use of caching of messages, to avoid duplicates, can possibly improve the performance. Other drawbacks are an increase in complexity, susceptibility for out-of-order datagram arrival and energy and bandwidth inefficiencies due to the forwarding of messages to all parts of the network instead of only to parts with interested nodes (lack of group registration). To solve these issues, [65] introduces an alternative protocol, called SMRF.

5.2.6. Anycast

Currently, no support for anycast is provided in RPL. The use of anycast could be very efficient, especially when multiple sinks are available.

5.2.7. Link Estimation

The estimation of the link quality with neighboring nodes is done by datapath validation via *Expected Transmission Count* (ETX) link cost estimation. The ETX equals the average number of transmissions for a packet to be successfully delivered to its next hop. The current selection mechanism prefers parents with the lowest rank. When there is more than one parent with the lowest rank, the first node is chosen as preferred parent, this has the advantage that no energy is consumed for evaluating the link quality to other neighboring nodes. As a downside, it only evaluates the currently used link; no alternative paths via newly discovered links are investigated. After some time this can lead to a sub-optimal routing topology. In [66] the solution of passive probing is introduced. Hereby the quality of a newly discovered link is initialized with the best value. This will force nodes to investigate all neighboring nodes as possible parent. At startup this solution will require more energy, but this energy is used to investigate the most optimal path, which can lead to a better overall energy efficiency.

However, in dense networks, the limited node memory results in constant re-evaluation of neighboring nodes, because neighbors in the neighbor table are continuously replaced by more recently used or heard neighbors. The authors of [66] suggest to use cache management to solve rediscovery and re-evaluation.

End-to-end link quality for a point-to-point route is currently not monitored in the RPL framework. A point-to-point link and its quality are currently determined by the individual links between the intermediate nodes. The nodes decide when to switch to a different parent based on the objective function. In *A Mechanism to Measure the Routing Metrics along a Point-to-point Route in a Low*

Power and Lossy Network (draft-ietf-roll-p2p-measurement-07) [67] a mechanism is described to analyze the quality of the current route and to allow the router to initiate the discovery of a better route.

Similar to the previous draft, an on demand discovery mechanism for routes with specified metric constraints is presented in *Reactive Discovery of Point-to-Point Routes in Low Power and Lossy Networks* (draft-ietf-roll-p2p-rpl-15) [68].

5.2.8. General Performance

For the implementation of RPL in Contiki, according to [48], 3224 bytes of the ROM are used and 800 bytes of the RAM. For a Tmote Sky mote this implies 6.5% of the ROM- and 8% of the RAM-resources.

The usage of the Trickle timer helps to minimize the amount of routing overhead. In a stable network the Trickle timer allows the beacon intervals to exponentially increase and when noticeable changes in the network conditions are detected the beacon interval quickly decreases to the minimum interval. Still, for scenarios with bidirectional traffic, the RPL protocol generates a larger traffic overhead (albeit with comparable delivery ratios) when compared to a protocol like LOAD [69]. It also has to be noted that LOAD is a reactive protocol, which has the additional advantage that the overhead is dependent of the traffic load, while RPL is proactive and has a constant overhead. When comparing TinyRPL with CTP [70], the well-known point-to-sink routing protocol for TinyOS, simulations indicate that the performance of both protocols has a comparable packet reception ratio and overhead. The benefit of RPL, compared to CTP, is the support for various types of traffic patterns (*i.e.*, multipoint-to-point, point-to-multipoint and point-to-point traffic) and the ability to directly connect to Internet nodes.

Finally, [71] analyzes the performance of RPL based on different simulations and concludes that RPL can ensure a fast network setup, which makes it a candidate protocol for monitoring applications in critical conditions. However they conclude that optimizations are required concerning the signaling in order to decrease protocol overhead.

5.3. Leveraging upon RPL to Realize the IoT

5.3.1. Real Life Use Cases

The RPL framework is currently used in different research projects. However in most cases the protocol is adapted to the specific requirements of the network. Most often, the RPL framework is used as a basis for the development of a specific protocol. Examples of such adaptations can be found in the previous evaluation section.

Additional examples include the following: in [72] the implementation of a smart monitoring system over a wireless sensor network is presented. The implementation focuses on the use of RPL to create an efficient and reliable routing structure. The paper also shows how, by changing some key parameters, the performance of RPL routing in a smart grid scenario can be influenced. In [73] a case study for the usage of RPL in an agricultural context is presented. Further, in [74] an example of the usage of RPL in transport logistics can be found.

5.3.2. Loop-free Repair Mechanisms

When link quality decreases and/or failure of a parent occurs, the repair mechanism of RPL, can introduce DODAG loops. In *Loop Free DODAG Local Repair* (draft-guo-roll-loop-free-dodag-repair-00) [75] an adaptation to the repair mechanism is proposed. In *Loop Free RPL* (draft-guo-roll-loop-free-rpl-01) [76] adaptation to the rank mechanism and objective function are presented. These drafts envision a loop free repair mechanism.

5.3.3. Heterogeneity

RPL specifications recommend forcing nodes with more constrained characteristics to operate only as a leaf node in a RPL network. A solution to force storing mode nodes to act as a non-storing mode in the presence of non-storing modes is presented in *RPL Routing Pathology In a Network With a Mix of Nodes Operating in Storing and Non-Storing Modes* (draft-ko-roll-mix-network-pathology-01) [77].

Most often, a simple objective function for a RPL based network is used. Objective functions however can also include device information to enforce specific routes. In *The Node Ability of Participation (NAP)* (draft-baryun-roll-nap-00) [78] the incorporation of node ability information is proposed to enable heterogeneity in terms of device capabilities.

5.3.4. DIS Handling

In *DIS Modifications* (draft-goyal-roll-dis-modifications-01) [79] DIS options are presented to influence the control of responses to the solicitation for DIOs. A first option is the adding of a metric container field, which contains routing constraints a router must satisfy in order to respond, in a DIS message. A second option is the response spreading for preventing collisions with responses on a DIS multicast message. In this draft two cases (leaf node joining a DAG and identification of defunct DAGs) which make use of the DIS mechanism are described and compared.

5.4. Research Challenges

Despite the standardization of RPL, some questions and gaps remain unsolved.

5.4.1. Interaction with MAC Protocols

First of all, the RPL interaction with different MAC protocols and duty cycles is not yet determined. Especially the behavior at startup, when lots of configuration traffic is sent out, hasn't been researched yet. Since most IoT devices are, by definition, energy constrained, this aspect requires significant additional research.

5.4.2. Asymmetric Links

The current RPL standard assumes the use of symmetrical links. For the selection of an optimal parent, the current RPL standard uses an approach based on the receipt of DIO messages from neighboring nodes. The node will select a parent based on the information in this message. However a node will use this link in the opposite direction of the receival of the DIO message. In case of

asymmetric links or unidirectional links the node will not be able to reach his preferred parent, which results in the reselection of another parent after some time. As such, this mechanism can be extended to also cope with asymmetric links.

5.4.3. Mobility

Currently RPL does not support node mobility. By supporting mobile nodes, the protocol can also be used for other application domains such as monitoring and safety systems for diverse traffic situations. In the field of mobile nodes in an RPL network, research activities are just starting. This means that lots of problems still have to be solved.

In [80] a multipath routing protocol for mobile sensor networks is presented. The protocol, called DMR, is based on the RPL framework. The performance of the protocol was compared against AODV and AOMDV by simulations in the NS-2.34 simulator. An improvement of energy efficiency with 25% and 20%, compared to AODV and AOMDV is achieved, while maintaining a delivery ratio of more than 97%.

Because a vehicular environment incorporates design elements of RPL, [81] provides a simulation performance study of adaptations of RPL for VANETs, by tuning different parameters of RPL. The test setup consists of nodes traversing in a line with an interdistance equal to the transmission range. The sink is positioned in the middle of the line. Information is transferred by multi-hopping with neighboring moving nodes.

In [82] adaptations are presented to enable data collection of a mobile node traversing, in a random pattern, in an environment of fixed nodes, running standard RPL. This is achieved mainly by having the mobile node continuously monitor his parents and neighborhood.

The selection and evaluation of the preferred parent, for mobile nodes, is still an open issue. For mobile nodes the link quality and neighboring nodes will vary due to the movement of the node. Therefore constant discovery of new neighbors and constant link analysis will be necessary. Because estimations are based on measurements, they also include an estimation of the past link quality. For fixed networks this estimation will be strongly correlated with the current link quality. For mobile nodes this estimation is typically outdated. An adaptive configuration for link estimation could help solve this problem, for instance by taking into account the movement (direction, speed,…) of the node, by reasoning on the succession of estimations (increasing values can indicate a node is approaching a parent or decrease of values a the opposite) or by adapting monitoring and/or discovery of neighbors.

Choosing a parent is important for reliably sending and receiving information. For fixed networks the parent with the best (stable) link quality is indeed the best choice. For mobile nodes, switching too frequently to a different parent also influences the reliability and energy consumption (extra control traffic for downward traffic). Therefore a selection of a parent which will be a good parent for a longer time can be a better choice.

5.4.4. Multi-Sink Support

In the standard, the possibilities for multi-sinks are briefly mentioned. However, currently no complete implementations are publicly available to our knowledge.

5.4.5. Scalability of the Non-Storing RPL Approach

As already mentioned the non-storing mode is a good mechanism to limit the usage of memory for nodes in dense networks, but it also has a negative effect on the depth of routing tree. In non-storing mode, the addresses of all the intermediate nodes are added to the packet. This includes the danger that these messages will get fragmented and limits the available payload [62]. This is definitely a challenge that has not been investigated thoroughly.

6. IETF CoRE Working Group

More recent, in 2010, an IETF working group, called Constrained RESTful Environments (CoRE), was founded specifically to work on the standardization of a framework for resource-oriented applications, allowing realization of RESTful embedded web services in a similar way as traditional web services, but suitable for the most constrained nodes and networks. Their work resulted in the Constrained Application Protocol (CoAP), a specialized RESTful web transfer protocol for use with constrained networks and nodes.

6.1. Key Protocols

CoAP is defined in draft-ietf-core-coap [83] in conjunction with a number of additional specifications. At the time of writing this article, there were three CoRE Internet-Drafts that are currently nearing completion and one completed RFC. In addition there were one more group Draft and 35 other individual Internet-Drafts listed on the CoRE website [5]. In this subsection we will describe briefly the (almost) completed CoRE Internet-Drafts and the published RFC. In the next section we will provide a brief summary of the topics covered by the individual Internet-Drafts in the CoRE working group.

6.1.1. Base CoAP

CoAP uses the same RESTful principles as HTTP, but it is much lighter so that it can be run on constrained devices [84,85]. To achieve this, CoAP has a much lower header overhead and parsing complexity than HTTP. It uses a 4-bytes base binary header that may be followed by compact binary options and payload. Figure 13 shows the CoAP message format as specified in version 13 of the draft. This version introduced a breaking change in the message format. However, it is expected that this will be the final change of the format.

Figure 13. CoAP Message Format consisting of a 4-bytes base binary header followed by optional extensions.

1 byte			1 byte	2 bytes	TKL bytes	variable	1 byte	variable
V	T	TKL	Code	Message ID	Token (if any)	Options (if any)	0xFF (if payload)	Payload (if any)
2	2	4 bits						

The CoAP interaction model is similar to the client/server model of HTTP. A client can send a CoAP request, requesting an action specified by a method code (GET, PUT, POST or DELETE) on a resource (identified by a URI) on a server. The CoAP server processes the request and sends back a response containing a response code and payload. Unlike HTTP, CoAP deals with these interchanges asynchronously over a datagram-oriented transport such as UDP and thus it also supports multicast CoAP requests. This allows CoAP to be used for point-to-multipoint interactions which are commonly required in automation. Optional reliability is supported within CoAP itself by using a simple stop-and-wait reliability mechanism upon request. Secure communication is also supported through the optional use of Datagram Transport Layer Security (DTLS).

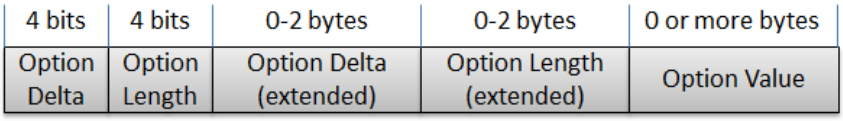
As can be seen in Figure 13 all CoAP messages start with a 4-bytes base binary header that consists of the following fields:

- **Version (V)**: A 2-bit unsigned integer indicating the CoAP version number. Current version is 1. Other values are reserved for future versions.
- **Type (T)**: A 2-bit unsigned integer indicating if this message is of type Confirmable (0), Non-Confirmable (1), Acknowledgement (2) or Reset (3).
- **Token Length (TKL)**: A 4-bit unsigned integer indicating the length of the variable-length Token field (0-8 bytes). Lengths 9-15 are reserved.
- **Code**: An 8-bit unsigned integer indicating if the message carries a request (1–31) or a response (64–191), or is empty (0). (All other code values are reserved.) In case of a request, the Code field indicates the Request Method (1: GET; 2: POST; 3: PUT; 4: DELETE); in case of a response a Response Code. Possible values are maintained in the CoAP Code Registry (see section 12 of the draft).
- **Message ID**: A 16-bit unsigned integer in network byte order used for the detection of message duplication, and to match messages of type Acknowledgement/Reset to messages of type Confirmable/ Non-confirmable.

The base 4-bytes header may be followed by one or more of the following optional fields:

- **Token**: 0 to 8 bytes, as given by the Token Length field. The Token value is used to correlate requests and responses. The rules for generating a Token and correlating requests and responses are defined in Section 5 of the draft.
- **Options**: An Option can be followed by the end of the message, by another Option, or by the Payload Marker and the payload. The format of the Options field is shown in Figure 14 and is described in more detail in the next paragraph.
- **Payload**: If present and of non-zero length, it is prefixed by a fixed, one-byte Payload Marker (0xFF) which indicates the end of options and the start of the payload. The payload data extends from after the marker to the end of the UDP datagram, *i.e.*, the Payload Length is calculated from the datagram size. The absence of the Payload Marker denotes a zero-length payload.

Figure 14. CoAP Option Format.



To be able to offer communication needs that cannot be satisfied by the base binary header alone, CoAP defines a number of options which can be included in a message. Each option instance in a message specifies the Option Number of the defined CoAP option. Instead of specifying the Option Number directly, the instances must appear in order of their Option Numbers and a delta encoding is used between them (The Option Number for each instance is calculated as the sum of its delta and the Option Number of the preceding instance in the message. For the first instance in a message, a preceding option instance with Option Number zero is assumed. Multiple instances of the same option can be included by using a delta of zero). The fields in an option are:

- **Option Delta**: 4-bit unsigned integer. A value between 0 and 12 indicates the Option Delta. A value of 13 indicates that an 8-bit unsigned integer follows the initial byte and indicates the Option Delta minus 13. A value of 14 indicates that a 16-bit unsigned integer in network byte order follows the initial byte and indicates the Option Delta minus 269. The value 15 is reserved for the Payload Marker and cannot be used here. The resulting Option Delta is used as the difference between the Option Number of this option and that of the previous option (or zero for the first option).
- **Option Length**: 4-bit unsigned integer. A value between 0 and 12 indicates the length of the Option Value, in bytes. A value of 13 indicates that an 8-bit unsigned integer precedes the Option Value and indicates the Option Length minus 13. A value of 14 indicates that a 16-bit unsigned integer in network byte order precedes the Option Value and indicates the Option Length minus 269. The value 15 is reserved for future use.
- **Value**: A sequence of exactly Option Length bytes. The length and format of the Option Value depend on the respective option, which may define variable length values.

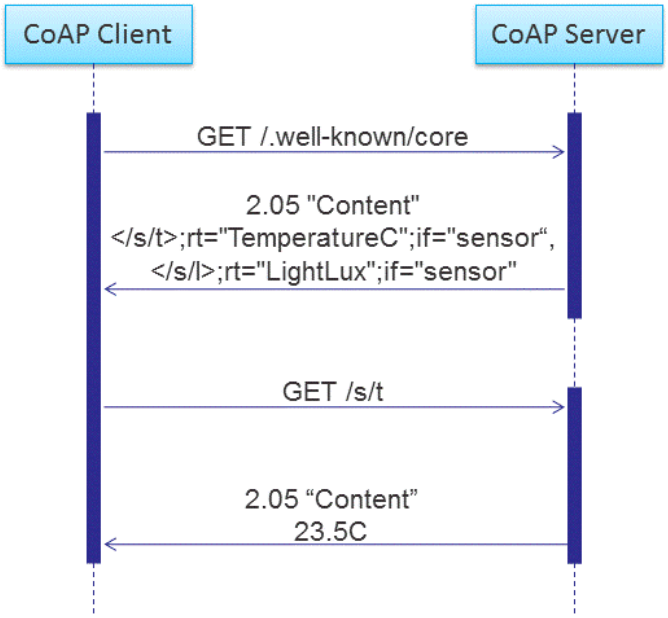
As an example of a simple CoAP option consider the Content-Format option. This option indicates the representation format of the message payload. This option has the Option Number 12 and its Option Length is between zero and two bytes. The Option Value itself is an unsigned integer that is defined in the CoAP Content Format registry (Section 12 of the draft).

The IETF CoRE working group considers the constrained restful environments as an extension of the current web architecture. The group envisions that CoAP will complement HTTP and that CoAP will be used not only between constrained devices and between servers and devices in the constrained environment, but also between servers and devices across the Internet [86]. An important requirement of the CoRE working group is to ensure a simple mapping between HTTP and CoAP so that the protocols can be proxied transparently. Thus proxies and/or gateways play a central role in the constrained environments architecture. These proxies have to be able to communicate between the Internet protocol stack and the constrained environments protocol stack and to translate between them as needed.

6.1.2. CoRE Link Format

Resource discovery is important for machine-to-machine (M2M) interactions, and is supported in CoAP using the *CoRE Link Format* as described in RFC 6690 [87]. A well-known relative URI "/.well-known/core" is defined as a default entry-point for requesting the list of links about resources hosted by a server. Once the list of available resources is obtained from the server, the client can send further requests to obtain the value of a certain resource. The example in Figure 15 shows a client requesting the list of the available resources on the server (GET/.well-known/core). The returned list shows that the server has, amongst others, a resource called/s/t that would return back the temperature in degrees Celsius. The client then requests the value of this resource (GET/s/t) and gets a reply back from the server (23.5C).

Figure 15. An example of Constrained RESTful Environments (CoRE) resource discovery and Constrained Application Protocol (CoAP) request.



6.1.3. Block Transfer

In many cases the payloads that CoAP needs to carry is very small (e.g., just a few bytes for temperature sensor, door lock status or toggling a light switch). In these cases the basic CoAP message provides very efficient means of communication and work very well. However in some cases CoAP needs to handle larger payloads (e.g., firmware update). Since CoAP is based on datagram transports such as UDP or DTLS, data fragmentation and reassembly is not offered by these transport protocols. Relying on IP fragmentation is also not very helpful, because fragmentation and reassembly does not perform well in LLN due to memory requirements imposed by route-over routing as described in Section 4. Additionally, IP fragmentation can handle only payloads up to 64 KiB. Thus, providing a mechanism at the application layer that is able of transferring large amounts of data in smaller pieces becomes a necessity. This will not just help avoid the 64KiB UDP datagram limit, but also will help avoid both IP fragmentation (MTU of 1280 for IPv6) and also adaptation layer fragmentation in LLNs (60–80 bytes for 6LoWPAN).

To overcome the payload size limitation, draft-ietf-core-block defines two CoAP options: Block1 and Block2 [88]. By using this pair of options CoAP becomes capable of transferring a large payload in multiple smaller CoAP messages. Both Block1 and Block2 options can be present both in request and response messages. In either case, the Block1 Option pertains to the request payload, and the Block2 Option pertains to the response payload. An important aspect of this mechanism is that often the server can handle block transfers in a stateless fashion—it does not require connection setup and the server does not need to track each transfer separately and thus conserve memory.

6.1.4. Observation of Resource

The state of a CoAP resource can change over time. In draft-ietf-core-observe [89] a simple CoAP extension is defined that enables a server to inform interested clients about the state change. A client interested in observing a resource includes the observe option in its GET request to the server. Whenever there is a change of the resource state, the server sends a notification to the client. Also, in case the state of the resource does not change, but the time since the last notification exceeds the max-age value of the resource, a notification is sent. As such, observe offers the possibility for a client to have an up-to-date representation of the resource without the client having to constantly poll for changes. New resource states are transmitted from the server to the clients according to a best-effort approach. The observe protocol foresees mechanisms to ensure consistency between the state observed by each client and the actual resource state.

6.2. Implementation and Evaluation

6.2.1. CoAP Implementations

At the time of writing this article, the CoAP protocol still is not yet finalized. However it is considered in its final stages before being finalized. Nevertheless several implementations of the CoAP protocol for various platforms and programing languages already exist. Some of these implementations are open source and others have commercial licenses. Interoperability between many of these implementations has been formally tested by the *European Telecommunications Standards Institute* (ETSI), a non-profit standards organization. ETSI organizes a series of events called ETSI Plugtests to test interoperability of telecommunication technologies in a multi-vendor, multi-network or multi-service environment. ETSI Plugtests, the IPSO Alliance and the FP7 Probe-it project have organized two IoT CoAP Plugtests. In addition to assessing the interoperability of participating products, these Plugtests events aimed to validate the CORE base standards. The first Plugtests event (March 2012) was attended by 18 companies testing the world's first CoAP client and server implementations. The features tested at this event included the base CoAP specification, CoAP Block Transfer, CoAP Observation and the CoRE Link Format. More than 90% of 3,000 tests executed in this event were successful. According to ETSI Plugtests this result is classified as a high level of interoperability [90].

At the second IoT CoAP Plugtests event (November 2012), some additional tests were added to cover previously untested aspects of CoAP in addition to introducing new optional tests for proxy functionality and M2M communication. In this event 1,775 test cases were performed, in 60 pairing test

sessions, with a success rate of 97.8% [91]. The improvement in interoperability compared to the first IoT CoAP Plugtests event indicates that the base CoAP protocol and its main options are getting more robust.

With insights of the first IoT Plugtests a survey on the current state of the art of lightweight REST implementations was presented in [92]. In Table 7 we present an adapted version of the survey results. In this adapted version we concentrate on publicly available implementations and servers and commercial implementations that are publicly announced (See [92] for other implementations). This table uses device classes as defined in [34]. Class 1 devices have ~10 KiB of RAM, and ~100 KiB of ROM. Class 2 devices have ~50 KiB of RAM, and ~250 KiB of ROM.

Table 7. CoAP implementations. Please note that these implementations are work in progress since CoAP itself is work in progress.

Name/ Company	License	Language	Platform	Notes
Consorzio Ferrara Ricerche [93]		NesC/C	TinyOS	Own "SiGLoWPAN" IPv6/6LoWPAN stack for Class 1 devices
Californium [94]/ETH Zurich	3-clause BSD	Java	JVM	Framework for unconstrained devices; provides client, server, and proxy stubs
Copper [95]/ETH Zurich	3-clause BSD	JavaScript	Firefox	Management and testing tool as a browser extension; focus on user interaction
Erbium [96]/ETH Zurich	3-clause BSD	C	Contiki	For class 1 devices such as sensor nodes
CoAP++ [97]/iMinds		C++	Click Modular Router	Framework for unconstrained devices; provides client, server, proxy and gateway
Evcoap [98]/KoanLogic	2-clause BSD	C	Linux	General purpose protocol implementation
Patavina Technologies [93]	Commercial	C++	proprietary OS	Wired and wireless embedded devices and sensor nodes; working on a port to uC/OS by Micrium
NanoService Device Library [99]/Sensinode	Commercial	C		OS-independent C library for Class 1 and 2 devices. Also available a JAVA SDK for unconstrained devices
libcoap[100]/Universität Bremen TZI	GPLv2, 2-clause BSD	C	POSIX and Contiki	General purpose library for Class 1 and 2 devices and up
CoapBlip [101]/Universität Bremen TZI	BSD-style	C	TinyOS	TinyOS-port of "libcoap"; runs on Class 1 devices.
coap.me [102]/Universität Bremen TZI		Ruby		http://coap.me provides an HTTP front-end to crawl CoAP servers, and a CoAP server for interoperability testing
jCoAP [103]/Universität Rostock	Apache 2.0	Java	JVM	For unconstrained devices; also targets mobile and embedded platforms
Scuola Superiore Sant'Anna [104]		Erika API	Erika OS	A middleware for building an infrastructure of wireless sensor nodes.
CoAPy [105]/People Power	BSD	Python		Last updated on July 2010
CoAP in wiselib [106]/wisebed project	GNU Lesser GPL v3	c++		Wiselib algorithm classes can be compiled for several sensor platforms such as iSense or Contiki, or the simulator Shawn.

The existence of such a wide range of implementations across a broad range of programing languages and most importantly platforms (constrained and not) demonstrates the feasibility of the protocol implementation.

6.2.2. CoAP Performance Evaluation

Going beyond mere compatibility tests and the ability to implement the protocol on constrained devices, a few studies have been made in order to evaluate the protocol's behavior and suitability for certain IoT use cases. In particular the comparison between CoAP and HTTP has been studied a few times. The results of such comparisons are considered preliminary since the CoAP protocol itself is not fully standardized yet, and because the studies used the protocol at different stages of its development. However these results still provide a good indication of what to expect when the protocol is fully standardized and the used implementations are further optimized.

For example, the work described in [107] provides an evaluation of CoAP compared to HTTP in terms of mote's energy consumption and response time in wireless sensor networks. The results for energy consumption, obtained by simulations for a fixed 10 second client request interval, show that while receiving and processing packets, the energy consumed when using CoAP is approximately half compared to the one consumed when using HTTP. While transmitting packets, the energy required by CoAP is 4 times lower than the energy required by HTTP. On the other hand, while being in idle mode, CoAP and HTTP result in similar values of energy consumption. When testing the response time on real sensor motes, the results show that CoAP/UDP introduces 9 to 10 fold lower response times than HTTP/TCP.

Similarly [108] evaluates the performance of HTTP/TCP and CoAP/UDP over a duty cycled radio layer. With a small modification to the duty cycling layer the authors achieved great improvement in performance at retained low power consumption. Furthermore they introduced an in-network caching mechanism that significantly improves the performance of software updates in incrementally deployed sensor networks.

In a third evaluation [109] the author finds that CoAP/UDP perform better than HTTP/TCP for the intelligent cargo container use case he evaluated. In particular the author reports a 6 times lower message size and a 4 times lower *Round Trip Time* (RTT). This is mainly due to CoAPs compressed header and the avoidance of the TCP handshaking mechanisms. A further study of the same use case [110] compares using CoAP/UDP to HTTP/UDP in addition to the typical HTTP/TCP. The study shows that generally UDP based protocols perform better for constrained networks due to using lower number of messages when retrieving resources. When comparing both protocols (CoAP and HTTP) when run over UDP, the study shows that CoAP performs better than HTTP. CoAP also has the added value of optional reliability since it has its own simple retransmission capability.

6.3. Leveraging Upon CoAP to Realize the IoT

The base CoAP protocol along with the three main complementary extensions (block, observe and Core Link Format) provide a basic set of protocols to solve a wide range of communication needs in constrained environment. However, since the protocol tries to be minimalistic and yet extendable at the same time, a couple of needs remain unaddressed in these base protocols. To try to address the

unsolved issues, a few individual Internet-Drafts were proposed in the CoRE working group. These drafts are in various states of development, with various levels of CoRE review and interest. A thorough overview of all these individual Internet-Drafts is given in draft-bormann-core-roadmap [111]. In this subsection we highlight some of these individual Internet-Drafts and complement them with additional literature that tries to leverage upon CoAP to realize the IoT.

6.3.1. Discovery and Naming

Resource Directory (RD) servers provide a way to collect resource descriptions from multiple servers into one central location. To facilitate setting up such a RD, draft-shelby-core-resource-directory identifies needed protocol elements [112].

In [97], CoAP is used to facilitate discovery and deployment of constrained devices. The proposed approach makes use of CoAP and combines it with DNS in order to enable the use of user-friendly fully qualified domain names (FQDN) for addressing sensor nodes. It also includes the translation of HTTP to CoAP, thus making the sensor resources globally discoverable and accessible from any Internet-connected client using either IPv6 addresses or DNS names both via HTTP or CoAP. This approach can be enhanced by publishing the discovered resource to one or more RDs.

Another example of how DNS can be used in a hierarchical fashion to enable easier access to resources is described in draft-ietf-core-groupcomm [113]. Here access to groups of resources can be provided via the use of Group FQDN that are uniquely mapped to a site-local or global multicast IP address via DNS resolution. For example the Group FQDN all.bldg6.example.com would refer to *all nodes in building 6* and the Group FQDN all.west.bldg6.example.com would refer to *all nodes in the west wing of building 6*.

6.3.2. Congestion Control

The base CoAP draft only defines a very basic congestion control when using reliable message transmissions and does not provide any congestion control when using the non-reliable transmissions mode, that is likely to carry the majority of traffic. To overcome this shortcoming a few proposals try to provide more advanced congestion control schemes. These proposals can generally provide more optimized performance in exchange for more implementation complexity and/or a narrower field of application. For example, draft-bormann-core-cocoa [114] defines some more advanced CoRE congestion control mechanisms. The main idea here is to provide a way to better estimate the RTT than that implied by the default initial timeout of 2 to 3 seconds. Further suggestions for the enhancements to this estimation of the RTT are presented in [115].

Another mechanism for congestion control is proposed in [116] by adding an option that allows a server to indicate its desire for some pacing of the requests sent to it by one client; enabling a form of server load control.

6.3.3. Advanced Interaction Patterns

The base CoAP provides good support for the simple interaction patterns between clients and servers. However more advanced interaction patterns such as the communication between a group of

devices requires extensions to the base protocol. In fact, it is anticipated that constrained devices will often naturally operate in groups (e.g., all window shutters on a given side of building may need to be lowered or raised as a group). Draft-ietf-core-groupcomm [113] discusses fundamentals and use cases for group communication patterns with CoAP. Building upon IPv6 multicast capabilities, the draft describes how CoAP should be used in both constrained and unconstrained networks and provides guidance for deployment in various network topologies. Although this draft has been adapted as a working group draft, it is still (at least in certain parts) in an explorative mode and will require additional investigation before conclusive results become available.

The CoAP observe option allows clients to register with servers to be notified whenever the state of a resource changes [89], much like the publish/subscribe paradigm in conventional web services. In [117] a new CoAP option "Condition" was proposed to extend the observe option. This option can be used by a CoAP client to specify the conditions the client is interested in. Several condition types, *i.e.*, filtering options, have been identified based on realistic use cases. Using conditional observations, the CoAP server will send a notification response with the latest state change only when the criterion is met. Using this mechanism a client can for example indicate that it is only interested in temperature values above 25 °C and not in all state changes. The feasibility of implementing this conditional observe on a constrained device is evaluated and proven in [118]. The correct operation for a simple scenario showed that the use of conditional observations can result in a reduced number of packets and power consumption compared to that which is normally observed in combination with client-side filtering.

6.3.4. Communication with Sleepy Nodes

Sleepy Nodes are network nodes that sleep most of the time in order to save energy and thus achieve longer battery life times. The base CoAP standard assumes that the communication layers below the application provide support functions for sleeping nodes. Adding better support for sleepy nodes at the application layer might be able to further reduce the power requirements of these nodes. This support is currently a very active subject of discussion in the CoRE working group; this is apparent from the relatively high number (at least seven) of individual Internet Drafts in the group that try to address this issue in one way or another.

The base CoAP provides minimal support for sleepy nodes by supporting caching in intermediaries. Resources from a sleepy node may be available from a caching proxy (if previously retrieved) even though the node is asleep. This support is enhanced by using the observe option and thus allowing sleepy nodes to update caching intermediaries according to their own schedule.

Most of these proposals try to achieve better support for sleepy nodes either by extending the functionality of the intermediaries or by extending the CoAP observe option or by a combination of both. For example, [119] proposes to store the actual resource representations in a special type of RD called the Mirror Server. Clients can then fetch the resource from the Mirror Server regardless of the state of the sleepy server. One the other hand, by using the conditional observe option as proposed in [120], the nodes may be allowed to sleep even longer. Similarly the approach of [121] is to introduce storing of sleep characteristics in the RD. Clients can then query the RD to learn the sleep status of the sleepy node before attempting communications. Both [120] and [121] include using/extending the observe option as part of their overall approaches.

A related patent [122], describes inserting sleep information into a header option or into a payload of an application layer message. Whereas the application layer message may be conveyed in HTTP or CoAP.

6.3.5. Security

Security is another hot topic on the CoRE working group with many drafts trying to tackle various security aspects of the Things and the Information they reveal about the physical world. It is anticipated that most real deployments of the IoT will require security services (e.g., confidentiality, authentication, authorization). However it is also argued that there is no single security architecture for the IoT [123]. A good description of the Thing Lifecycle is provided in [124] along with resulting architectural considerations.

The authors of [125] present a three-phase protocol to bootstrap constrained devices in a wireless sensor network based on IPv6 and CoAP. The protocol phases include service discovery, distribution of security credentials, and application-specific node configuration.

CoAP proposes to use DTLS to provide end-to-end security to protect the IoT. However DTLS is a heavyweight protocol and its headers are too long to fit in a single IEEE802.15.4 MTU. The works presented in [126–129] look specifically into the use of DTLS in constrained networks from different angles. As an example, while [128] shows how to build minimal implementations of TLS, the approach used in [129] relies on providing 6LoWPAN header compression mechanisms to reduce the size of the DTLS security headers. The authors report as an example that the number of additional security bits needed for the DTLS Record header that is added in every DTLS packet, can be reduced by 62%.

Another relevant security protocol is the Internet Key Exchange version 2 (IKEv2) which is a used for setting up IPsec security associations. IKEv2 includes several optional features, which are not needed in minimal implementations. The Minimal IKEv2 draft [130] shows how to build minimal implementations of the security protocols IKEv2 for constrained environments.

6.3.6. Intermediaries

The base CoAP draft defines basic mapping between CoAP and HTTP. However it is expected that Intermediaries will continue to play a big role in the IoT and that the basic mapping needs to be enhanced to support advance features. Some ideas about these enhancements are presented in [131]. Additional useful examples for more advanced forms of mapping and usages are described in [132].

6.3.7. CoAP in Cellular Networks

The *Short Message Service* (SMS) of mobile cellular networks is frequently used in M2M communications. The service offers small packet sizes and high delays just as other typical types of LLNs. Since the design of CoAP takes the limitations of LLNs into account, it is expected that CoAP can be nicely used with SMS. The adaptation of CoAP to the SMS transport mechanisms and the combination with IP transported over cellular networks is described in [133].

6.3.8. Real Life Use Cases of CoAP in the IoT

Some recent publications show that CoAP is being considered as a good candidate to solve current issues in real life application of the IoT. For example, [110] presents an IP based solution to integrate sensor networks used in a cargo container with existing logistic processes, highlighting the use of CoAP for the retrieval of sensor data during land or sea transportation.

Kovatsch *et al*. [134] propose an IoT architecture where the infrastructure is agnostic of applications and application development is fully decoupled from the embedded domain. This is achieved by creating a common application layer that fosters the development of novel applications. The application logic of devices is running on application servers, while thin servers embedded into devices export only their elementary functionality using CoAP resources.

GlobalPlatform is a cross industry not-for-profit association which publishes specifications facilitating the secure and interoperable deployment and management of multiple embedded applications on secure chip technology. According to a recent presentation [135], GlobalPlatform is considering the use of CoAP for the management of secure environments and other aspects covered by GlobalPlatform standards.

Castro *et al*. [136] present how the IoT is integrated for improving terrestrial logistics offering a comprehensive and flexible architecture, with high scalability, according to the specific needs for reaching an item-level continuous monitoring solution. CoAP is used here to provide tracking and monitoring services at any time during the transportation of goods. The solution makes use of observe and blockwise transfer options to optimize data transfers.

Optimizing energy policies requires monitoring, analyzing, and controlling of power consumption. Smart metering is an emerging topic for realizing such modern energy policies. In this field a large number of proprietary and open standards for communication (with low or no interoperability to each other) exist today. Therefore, it is very difficult to integrate multi-vendor solutions using one sustainable holistic approach. To this end the authors of [35] propose to use Web Service technology as an open widespread Internet standard for the creation of a heterogeneous network for smart metering devices. This work uses CoAP over TCP transport instead of using it over UDP, which is still not specified in the current CoAP drafts. Nevertheless, traffic overhead was reduced by over 90% by using CoAP instead of HTTP and EXI for XML binary encoding.

Using CoAP and *REsource LOcation And Discovery* (RELOAD), [137] proposes a new architecture for wide area sensor and actuator networking. RELOAD is a P2P signaling protocol for use on the Internet that is currently being standardized by the IETF. The architecture provides a decentralized peer-to-peer rendezvous service for CoAP nodes in WSNs and enables a P2P federation of geographically distributed WSNs. This is achieved by the use of proxy nodes that are part of the WSN but also connect to a RELOAD overlay network via cellular Internet access. The authors conclude that such architecture is most beneficial for large-scale networks having from moderate to high levels of interdevice communication.

Rahman *et al*. [138] present a smart object gateway architecture that allows for efficient service delivery between the Smart Object and an endpoint on the Internet such as an application server. A survey of some other examples of how CoAP is been used to realize the IoT can be found in [139].

6.4. Research Challenges

Since CoAP is not fully standardized yet, many related aspects remain to be examined before definite conclusions can be drawn. Some of these aspects are currently being researched and have been documented in Section 6.3. Some other aspects have been identified but no solutions were proposed yet. In this section we summarize the main aspects of the CoAP protocol where we think that extensive research is still needed.

Although a few suggestions for congestion control were already proposed (see Section 6.3.2.), we think that this area requires more research attention. The primary reason being that UDP is the main transport mechanism for CoAP. Unlike TCP, UDP does not provide any reliability mechanism or congestion control. CoAP has its own optional light weight reliability mechanism, but virtually no congestion control. Simple mechanisms for congestion control, that are optimized for LLNs and that take into account the limited amount of resources available on constrained devices are still lacking.

Another research aspect is related to the MTU size. The IPv6 minimum MTU is 1,280 bytes, whilst the maximum MTU for IEEE 802.15.4 networks is 127 bytes. The 6LoWPAN adaptation layer provides an efficient fragmentation method to allow the transmission of larger packets. However large packet sizes still sometimes impose a big problem to the receiving party. If the received message does not fit in the input buffer it could cause unpredictable effects on the receiving side, e.g., operating system restarting because of buffer overflow. For this reason, it can be useful to specify during the resource discovery sequence the maximum accepted length of the response message with the resource description [140]. If both parties support the block option, agreeing on the optimal block transfer size in a way that avoids fragmentation and assembly of the packet at the lower layers will possibly have a high positive impact on the overall performance. This is something that needs further research and experimentation.

Security of the IoT remains a challenge despite the fact that several proposals have already been made to cover certain aspects of security. Issues such as secure distribution of encryption keys in LLNs still need to be explored.

A basic mapping between HTTP and CoAP is well defined in the core CoAP draft. However since CoAP is by design highly extendable and new options are being regularly added, such extensions might impose new challenges to the HTTP/CoAP mapping. The same consideration also applies in general to all intermediaries, that need to be updated if they wish to understand new options as well (some basic information for intermediaries is contained in an option's option number, which allows intermediaries to correctly handle unknown options). Caching intermediaries have an even bigger challenge to find out and implement optimal caching strategies.

Group communication in a LLN context still needs more attention as well. The use of multicast satisfies many group communication needs, but not all of them. Multicasts are transported unreliable and since LLNs are lossy by nature, many applications will opt not to use multicast if they want to make sure that their messages are indeed delivered. For example, if multicast is used to turn on all the lights in a room, it wouldn't be acceptable if one light stays off because it did not receive the multicast message. Alternatives to the use of multicasts might be needed for such use cases. Maybe status synchronization of resources can also be used in certain use cases to make sure that the message did indeed get through. The efficiency of such and other approaches need to be examined.

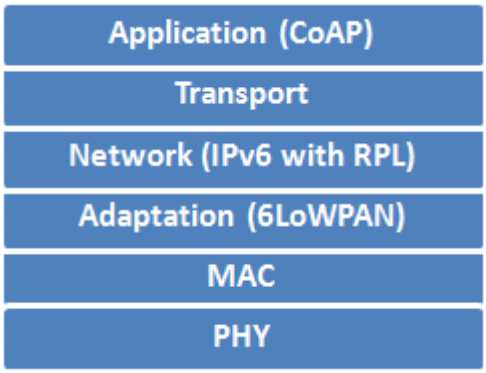
Finally, some challenges are related to the fact that resources on constrained devices often need to be accessed by other machines as well as by humans. These two types of "clients" have somewhat contradictory requirements. Humans for example prefer easy to remember (and a bit lengthy) resource names, while M2M communication is better served by providing concise names. Some more research is needed in order to satisfy the needs of both types of "clients" without adding many requirements on the constrained devices. The same considerations also apply to the use of extended semantics, e.g., a *Web Service Definition Language* (WSDL). WSDL provides a machine-readable description of how a web service can be used, what parameters it expects, and what data structures it returns. However WSDL tends to become relatively verbose and long for constrained nodes to handle, thus alternatives need to be researched.

7. Using IETF Standards to Realize the Internet of Things

7.1. Overview of the IETF LLN Protocol Stack

The previous sections provided an overview of different standards, their current status and open research topics. Combined, these individual standards form an *IETF LLN protocol stack* to support the realization of an interoperable Internet-of-things. In Figure 16 a representation is given of how the different LLN standards fit together. Currently the entire IETF LLN protocol stack is available in the Contiki OS, which is used in, for example, the CALIPSO FP7 project [31]. For simulating the IETF LLN protocol stack OpenWSN [49] can also be used.

Figure 16. IETF LLN protocol stack.



The LLN protocol stack provides end-to-end access to embedded web services, thus enabling new functionalities or building novel services involving IoT objects. This section focuses on the next steps: which additional (new or existing) research, protocols and/or standards are needed to realize a fully-automated, all-encompassing Internet of Things.

7.2. Realizing the Web of Things

To allow the integration of an increasingly large number of IoT devices, self-configuration protocols will be required. Solutions for self-configuration such as [97] allow newly deployed constrained devices to be automatically discovered, automatically assign DNS hostnames and transparently make the IoT resources directly accessible and browsable over IPv6 via HTTP or CoAP.

Alternatively, devices can add their resources to a publicly accessible directory service. This sort of solution forms an important building block that facilitates the actual usage of embedded web services (without human intervention) as is required for realizing the Web of Things (WoT). For instance, a client can automatically be notified about new resources and continuously observe the state of a resource using the CoAP observe extension [89], leading to a consistent representation of all resources of interest. Using conditional observations [117], interested parties can be notified about resource states that satisfy specific conditions, thereby acting as an enabler to build applications such as sensor—actuator interactions. These extensions enrich the capabilities of the basic CoAP protocol and contribute to the realization of the WoT.

Another important research domain focuses on web service composition whereby different IoT services are combined to realize complex goals. For instance, embedded web services can automatically be combined to create complex interaction scenarios where knowledge about the real world is used, linked with other services and processed to act again upon the physical world. Existing composition and orchestration frameworks such as described in [141], need to be extended in order to realize the WoT. The main challenge is the adaptation of existing web service composition models to take into account the limitations of constrained devices. Also, when time varying data from constrained objects is incorporated or web services act upon the real world, issues such as consistency, failures, correct execution of all transactions as described in [142] need to be explored in view of a constrained environment.

The link between the IoT and state-of-the-art cloud technology solutions is made clear in [143]. Cloud technology can be used for collecting, storing and processing the enormous amount of sensor data. Tiny objects can also be introduced as part of grid computing e.g., for the collection and processing of environmental information. In [144] an extensive overview of the introduction of mobile devices into Grid systems is given and an extension to the constrained world seems feasible with the advent of embedded web service technology.

It is clear that the step from Internet of Things towards a Web of Things will be taken sooner rather than later. The IoT can facilitate the realization of the WoT, opening up access to sensor data and stimulating their widespread usage, while at the same time avoiding vertically integrated and closed systems. As such, it presents great opportunities to researchers active as well in the field of web service technology as in the field of embedded distributed systems.

7.3. Interoperability

One of the key factors to ensure the widespread use of IoT devices is to support end-to-end interoperability between different devices. Currently interoperability between embedded devices is enforced by use of a standardized IETF LLN protocol stack. Despite the efforts of the IETF working groups, different interpretations of the standards remain a threat that causes interoperability problems between implementations from different parties. Solving these issues requires identification of the possible problems and clarification of the implementation details where necessary to prevent possible ambiguities in the standards itself. Extensive interoperability testing events, like the ETSI Plugtests for CoAP and in the future possibility for 6LoWPAN, significantly help to improve the quality of the standards and the implementations.

Another more fundamental issue to consider is that of low-power interoperability, *i.e.*, interoperability when communicating parties employ (possibly different) MAC radio duty cycling techniques. The authors of [145] identify two open issues, namely that existing protocols for LLNs typically have not been designed for MAC duty cycling and that existing MAC duty cycling mechanisms have not been designed for interoperability. While most of the IETF work is situated above the MAC sublayer, it is important that these concerns are addressed in the future so that end-to-end IPv6 connectivity with embedded devices, that employ heavy MAC duty cycling, is possible.

7.4. Bringing Semantics to the Web of Things

Semantics define a globally interpretable significance to data. Adding semantics to the IoT allows data that originates from different sources to be unambiguously accessible and processable across different domains and by different users. This allows describing data that is collected from the real world which helps automated processing and integration of said data into applications. Semantic descriptions are particularly useful in M2M environments where a high level of autonomy is assumed. Said descriptions can also help to facilitate discovery and management of IoT devices and their resources. In [146], the authors mention that the dynamicity and pervasiveness of the IoT domain poses additional challenges for semantic description when compared to conventional web service environments. More specifically, the volatility of the sensor data and the large scale of the IoT are new challenges when defining semantics when compared to traditional web services.

When looking at the resources on the devices themselves (e.g., a temperature) several resource representations can be explored: ranging from plain text formats, through formats defined by the IPSO alliance [147] to complex semantic representations using ontologies that are adapted to the specific applications and domains as described for instance in [148]. Furthermore, the SPITFIRE project [149] has defined vocabularies to describe sensors and to integrate them with W3C's Linked Open Data cloud [150]. This allows linking sensor data with other data that is already available on the World Wide Web (WWW). This brings the potential of semantic web technology (e.g., searching and reasoning) to constrained devices, realizing a Semantic Web of Things.

Similar to search engines in the WWW, sensors and their resources could be indexed just like regular web pages and made available to Internet users and other IoT devices. Of course, issues such as time dependent aspects should be taken into consideration (e.g., indexing a temperature sensor) introducing novel challenges and opportunities. Platforms like Cosm [151] already allow to make sensor data publicly available, browsable and searchable but lack the ability to actively crawl and index sensors.

7.5. Security and Privacy in the Web of Things

Applications running on embedded devices and LLNs often require confidentiality and integrity protection. These security mechanisms can be provided at the application, transport, network, and/or at the link layer. In all these cases, prevailing constraints will influence the choice of a particular protocol. Some of the more relevant constraints are small code size, low power operation, low complexity, and small bandwidth requirements.

When securing embedded environments by using DTLS (as assumed within CoAP) at the transport layer, IPSec at the network layer, or reusing IEEE 802.15.4 security primitives within the 6LoWPAN adaptation layer; the problem of key distribution remains mostly untackled. It is often assumed that each device has an appropriate asymmetric public and/or symmetric key installed. How these keys actually are distributed and installed in a safe way, is often left open and is not part of current IETF standards. To help solve this issue, work is executed within IETF's *Smart Object Lifecycle Architecture for Constrained Environments* (SOLACE) [152].

Apart from enabling secure communications between parties, privacy of WoT users is also an important concern. As embedded devices measure and control the physical environment that surrounds a WoT user, they can be (ab)used to gain insight into a user's habits and whereabouts. Therefore it is vital to enforce adequate access control to this sensitive information, in order to protect the user's privacy. As more and more internet-enabled things will gain an online presence, transparent and easy-to-use management of what information devices may expose to the outside world will be needed. For instance, you might want to give your AC vendor access to the temperature sensors in your house without allowing them to turn on and off your lighting. While you might want to allow a different vendor to only control your lighting. One approach as discussed in [153] is to group embedded devices into virtual networks and only expose certain resources within each virtual network, thus achieving access control at the network layer. Another option is to provide access control at the application layer, where access to resources on embedded devices is granted/revoked on a per-vendor basis and the application itself enforces the access control.

7.6. Reprogrammability

Finally, many IoT devices will have a long lifetime and will be deployed at locations that are difficult to reach. As such, it is clear that some mechanism will be required to (wirelessly) update IoT devices with new or updated firmware, software, protocols and/or security keys. As an example, the DisSeNT project [154] already provides solutions to wirelessly add new application level components at run-time to embedded devices.

Providing wireless updates to IoT devices is at the moment quite difficult because it requires additional overhead in terms of the memory footprint of the software. In addition, (multi-hop) transmitting a firmware image to multiple embedded devices quickly depletes the batteries of involved devices. At the moment, very few embedded operating systems have the capability to execute these kinds of updates: additional research and standards will be needed before future-proof IoT networks can be deployed.

8. Conclusions

The popularity of sensor networks (and in a broader sense Internet of Things) has increased significantly over the last ten years. Integration of these embedded devices into the Internet is challenging, since they have characteristics that differ strongly from traditional internet devices, such as very limited energy, memory and processing capabilities. Initially, research focused on developing proprietary solutions that were typically vendor-specific and did not allow end-to-end connectivity between client devices and sensor devices. However, the use of standardized protocols

enables the integration of constrained devices in the IPv6 Internet, both at the network level and at the service level.

In this paper, a high-level overview was given of the ongoing IETF standardization work that focuses on enabling direct connectivity between clients and sensor devices. To this end, different IETF groups are currently active. The IETF groups 6LoWPAN and ROLL focus on the IPv6 addressability and routing, whereas the IETF CoRE group focuses on realizing an embedded counterpart for RESTful web services. By combining these protocols, an embedded protocol stack can be created that has similar characteristics to traditional internet protocol stacks. In fact, the IETF protocols are designed to enable easy translation from internet protocols to sensor protocols and vice-versa.

The paper describes how the combination of IETF protocols enables flexible, direct interaction between internet clients and embedded Internet of Things devices. However, the paper also shows that the advent of standardized protocols is not an end point, but only a starting point for exploring additional open issues that should be solved to realize an all-encompassing Internet of Things. Several open challenges remain such as resource representation, security, dealing with sleeping nodes, energy efficiency, integration with existing web service technologies and tools, linking with Cloud services, use of semantics, easy creation of applications, scalability, interoperability with other wireless standards, maintainability, *etc*.

Anyone involved in Internet of Things research (whether dealing with network layer aspects or service layer aspects) will, sooner or later, be confronted with the IETF protocols. This paper merely touches the surface of this broad domain and tries to encourage others to further explore the world of Internet-connected objects and tackle the mentioned remaining open issues and challenges.

Acknowledgments

The research leading to these results has received funding from the European Union's Seventh Framework Programme (FP7/2007-2013) under grant agreement no 258885 (SPITFIRE project), from the iMinds ICON projects GreenWeCan and O'CareCloudS, a FWO postdoc grant for Eli De Poorter and a VLIR PhD scholarship to Isam Ishaq.

Conflict of Interest

The authors declare no conflict of interest.

References

1. Kushalnagar, N.; Montenegro, G.; Schumacher, C.P.P. IPv6 over Low-Power Wireless Personal Area Networks (6LoWPANs): Overview, assumptions, problem statement, and goals. IETF RFC 4919. 2007; pp. 1–2.
2. Montenegro, G.; Kushalnagar, N.; Hui, J.; Culler, D. Transmission of IPv6 Packets over IEEE 802.15.4 Networks. IETF RFC 4944. 2007; pp. 1–30.
3. Routing Over Low power and Lossy networks (roll). Available online: http://datatracker.ietf.org/wg/roll/ (accessed on 3 October 2012).

4. ZigBee Alliance Plans Further Integration of Internet Protocol Standards. Available online: https://docs.zigbee.org/zigbee-docs/dcn/09-5003.pdf (accessed on 3 October 2012).
5. Constrained RESTful Environments (core). Available online: http://datatracker.ietf.org/wg/core/ (accessed on 28 December 2012).
6. IPv6 over Low power WPAN (6lowpan). Available online: http://datatracker.ietf.org/wg/6lowpan/ (accessed on 28 December 2012).
7. Constrained RESTful Environments (core). Available online: http://datatracker.ietf.org/wg/core/ (accessed on 28 December 2012).
8. IEEE 802.15.4. Available online: http://www.ieee802.org/15/pub/TG4.html (accessed on 28 December 2012).
9. IEEE Standard for Local and Metropolitan Area Networks—Part 15.4: Low-Rate Wireless Personal Area Networks (LR-WPANs). 5 September 2011; pp. 1–314.
10. ZigBee Alliance. Available online: http://www.zigbee.org/ (accessed on 28 December 2012).
11. HART Communication Protocol and Foundation. Available online: http://www.hartcomm.org/ (accessed on 28 December 2012).
12. MiWi Development Environment. Available online: http://www.microchip.com/miwi (accessed on 28 December 2012).
13. ISA100 Wireless Systems for Automation. Available online: http://www.isa.org/isa100 (accessed on 28 December 2012).
14. Vasseur, J.-P.; Dunkels, A. *Interconnecting Smart Objects with IP: The Next Internet*; Morgan Kaufmann: Amsterdam, Holand, 2010.
15. Montenegro, G.; Kushalnagar, N.; Hui, J.W.; Culler, D.E. Transmission of IPv6 Packets over IEEE 802.15.4 Networks. IETF RFC 4944. 2007.
16. Hui, J.; Thubert, P. Compression Format for IPv6 Datagrams over IEEE 802.15.4-Based Networks. IETF RFC 6282. 2011; pp. 1–24.
17. Chakrabarti, S.; Nordmark, E.; Bormann, C. *Neighbor Discovery Optimization for IPv6 over Low-Power Wireless Personal Area Networks (6LoWPANs)*; Shelby, Z., Ed.; IETF RFC 6775. 2012; pp. 1–55.
18. Kim, E.; Kaspar, D. Design and Application Spaces for IPv6 over Low-Power Wireless Personal Area Networks (6LoWPANs). IETF RFC 6568. 2012; pp. 1–28.
19. Gomez, C.; Kim, E.; Kaspar, D.; Bormann, C. Problem Statement and Requirements for IPv6 over Low-Power Wireless Personal Area Network (6LoWPAN) Routing. IETF RFC 6606. 2007.
20. Nieminen, J.; Savolainen, T.; Isomaki, M.; Shelby, Z.; Gomez, C. Transmission of IPv6 Packets over BLUETOOTH Low Energy. draft-ietf-6lowpan-btle-11. 2012.
21. IPv6 Over Low power WPAN (6LoWPAN) Charter. Available online: http://datatracker.ietf.org/wg/6lowpan/charter/ (accessed on 13 December 2012).
22. Hui, J.; Culler, D.; Chakrabarti, S. 6LoWPAN: Incorporating IEEE 802.15.4 into the IP architecture 2009, p. 17.
23. Mulligan, G. The 6LoWPAN Architecture. In Proceedings of the 4th Workshop on Embedded Networked Sensors; Cork, Ireland, 25–26 June 2007; ACM: New York, NY, USA, 2007; pp. 78–82.

24. Cody-Kenny, B.; Guerin, D.; Ennis, D.; Carbajo, R.S.; Huggard, M.; McGoldrick, C. Performance Evaluation of the 6LoWPAN Protocol on MICAz and TelosB Motes. In Proceedings of 4th ACM Workshop on Performance Monitoring and Measurement of Heterogeneous Wireless and Wired Networks; Tenerife, Canary Islands, Spain, 26 October 2009; ACM: New York, NY, USA, 2009; pp. 25–30.
25. Sulthana, M.R.; Bhuvaneswari, P.T.V.; Rama, N. Routing Protocols in 6LoWPAN: A Survey. *Eur. J. Sci. Res.* **2012**, *85*, 248–261.
26. Borman, C. 6LoWPAN Generic Compression of Headers and Header-like Payloads. draft-bormann-6lowpan-ghc-05. 2012.
27. Sahara Project. Available online: http://sahara.tzi.org/ (accessed on 10 December 2012).
28. HOBNET project. Available online: http://www.hobnet-project.eu/ (accessed on 10 December 2012).
29. Outsmart: FP7 Framework Project. Available online: http://www.fi-ppp-outsmart.eu/en-uk/Pages/default.aspx (accessed on 10 December 2012).
30. Cassaniti, D. A Multihop 6LoWPAN Wireless Sensor Network for Waste Management Optimization, M.Sc. thesis, University of Padova, Padova, Italy, 2012, p. 154.
31. Calipso Project. Available online: http://www.ict-calipso.eu/ (accessed on 10 December 2012).
32. Khoshdelniat, R. 6LoWPAN Applications and Internet of Things. Available online: http://www.apan.net/meetings/Hanoi2010/Session/SensNet.php (accessed on 10 December 2012).
33. Schoenwaelder, J.; Sehgal, A.; Tsou, T.; Zhou, C. Definition of Managed Objects for IPv6 over Low-Power Wireless Personal Area Networks (6LoWPANs). draft-schoenw-6lowpan-mib-01. 2012.
34. Bormann, C. Guidance for Light-Weight Implementations of the Internet Protocol Suite. draft-ietf-lwig-guidance-02. 2012.
35. Altmann, V.; Skodzik, J.; Golatowski, F.; Timmermann, D. Investigation of the Use of Embedded Web Services in Smart Metering Applications. In Proceedigns of the 38th Annual Conference of the IEEE Industrial Electronics Society (IECON2012), Montréal, PQ, Canada, 25–28 October 2012.
36. Routing over Low power and Lossy networks (roll)—Charter. Available online: http://datatracker.ietf.org/wg/roll/charter/ (accessed on 27 December 2012).
37. Pister, K.; Thubert, P.; Phinney, T. Industrial Routing Requirements in Low-Power and Lossy Networks. IETF RFC 5673. 2009; pp. 1–27.
38. Buron, J.; Brandt, A.; Porcu, G. Home Automation Routing Requirements in Low-Power and Lossy Networks. IETF RFC 5826. 2010; pp. 1–17.
39. Martocci, J.; Mil, P. De; Riou, N.; Vermeylen, W. Building Automation Routing Requirements in Low-Power and Lossy Networks. IETF RFC 5867. 2010, pp. 1–26.
40. Watteyne, T.; Berkeley, U.C.; Winter, T.; Barthel, D. Routing Requirements for Urban Low-Power and Lossy Networks. IETF RFC 5548. 2009, pp. 1–21.
41. Winter, T.; Thubert, P.; Brandt, A.; Hui, J.; Kelsey, R.; Levis, P.; Pister, K.; Struik, R.; Vasseur, J.P.; Alexander, R. RPL: IPv6 routing protocol for low-power and lossy networks. IETF RFC 6550. 2013; pp. 1–157.

42. Vasseur, J.; Kim, M.; Pister, K.; Dejean, N.; Barthel, D. Routing metrics used for path calculation in low power and lossy networks. IETF RFC 6551. 2011; pp. 1–30.
43. Thubert, P. Objective Function Zero for the Routing Protocol for Low-Power and Lossy Networks (RPL). IETF RFC 6552. 2012; pp. 1–14.
44. Gnawali, O.; Levis, P. The Minimum Rank with Hysteresis Objective Function. IETF RFC 6719. 2012, 1–13.
45. Clausen, T.; Gnawali, O.; Ko, J.; Hui, J. The Trickle Algorithm. IETF RFC 6206. 2011; pp. 1–13.
46. Conta, A.; Gupta, M. Internet control message protocol (icmpv6) for the internet protocol version 6 (ipv6) specification. IETF RFC 4443. 2006; pp. 1–24.
47. TinyRPL—TinyOS Documentation Wiki. Available online: http://docs.tinyos.net/tinywiki/index.php/TinyRPL (accessed on 27 December 2012).
48. Tsiftes, N.; Eriksson, J.; Dunkels, A. Low-power Wireless IPv6 Routing with ContikiRPL. In Proceedings of the 9th ACM/IEEE International Conference on Information Processing in Sensor Networks, IPSN ’10, Stockholm, Sweden, 12–16 April 2010, ACM Press: New York, NY, USA, 2010; p. 406.
49. Berkeley’s OpenWSN Project. Available online: http://openwsn.berkeley.edu/ (accessed on 28 December 2012).
50. Nano-RK. Available online: http://www.nanork.org/projects/nanork (accessed on 28 December 2012).
51. Jeong, J. Design and Implementation of Low Power Wireless IPv6 Routing for NanoQplus. In Proceedings of the 13th International Conference on Advanced Communication Technology (ICACT), Daejeon, South Korea, 13–16 February 2011; pp. 966–971.
52. Pavković, B.; Theoleyre, F.; Duda, A. Multipath Opportunistic RPL Routing over IEEE 802.15.4. In Proceedings of the 14th ACM International Conference on Modeling, Analysis and Simulation of Wireless and Mobile Systems, MSWiM ’11, Miami Beach, FL, USA, 31 October 2011; ACM Press: New York, NY, USA, 2011; p. 179.
53. Saad, L.; Chauvenet, C.; Tourancheau, B. Simulation of the RPL Routing Protocol for IPv6 Sensor Networks: Two Cases Studies. In Proceedings of the International Conference on Sensor Technologies and Applications, Nice, France, 21–27 August 2011; Volume 2011.
54. Contiki: The Open Source Operating System for the Internet of Things. Available online: http://www.contiki-os.org/ (accessed on 20 December 2012).
55. Bartolozzi, L.; Pecorella, T.; Fantacci, R. ns-3 RPL module: IPv6 routing protocol for low power and lossy networks. In Proceedings of the 5th International ICST Conference on Simulation Tools and Techniques (SIMUTOOLS), Desenzano, Italy, 19–23 March 2012; pp. 359–366.
56. Hammerseth, S.K. Implementing RPL in a Mobile and Fixed Wireless Sensor Network with OMNeT++. M.Sc. Thesis, University of Oslo, Oslo, Norway, 29 November 2012, pp. 1–101.
57. rpl-jsim-platform—Implementation of RPL functionality in JSim platform—Google Project Hosting. Available online: http://code.google.com/p/rpl-jsim-platform/ (accessed on on 20 December 2012).
58. Gnawali, O.; Levis, P. Recommendations for Efficient Implementation of RPL. draft-gnawali-roll-rpl-recommendations-04. 2012.

59. Tripathi, J.; Oliveira, J.; Vasseur, J.-P. Performance Evaluation of the Routing Protocol for Low-Power and Lossy Networks (RPL). IETF RFC 6687. 2012; pp. 1–26.
60. Ko, J.; Eriksson, J.; Tsiftes, N.; Dawson-haggerty, S.; Terzis, A.; Dunkels, A.; Culler, D. ContikiRPL and TinyRPL: Happy Together. In Proceedings of the workshop on Extending the Internet to Low power and Lossy Networks (IP+SN), Chicago, IL, USA, 11 April 2011.
61. Clausen, T. H.; Herberg, U.; Philipp, M. A Critical Evaluation of the IPv6 Routing Protocol for Low Power and Lossy Networks(RPL). In Proceedings of the 2011 IEEE 7th International Conference on Wireless and Mobile Computing, Networking and Communications (WiMob), Wuhan, China, 23–25 September 2011; pp. 365–372.
62. Xie, W.; Goyal, M.; Hosseini, H.; Martocci, J.; Bashir, Y.; Baccelli, E.; Durresi, A. A Performance Analysis of Point-to-Point Routing along a Directed Acyclic Graph in Low Power and Lossy Networks. In Proceedings of the 13th International Conference on Network-Based Information Systems (NBiS), Takayama, Japan, 14–16 September 2010; pp. 111–116.
63. Baccelli, E.; Philipp, M.; Goyal, M. The P2P-RPL Routing Protocol for IPv6 Sensor Networks: Testbed Experiments. In Proceedings of the 19th International Conference on Software, Telecommunications and Computer Networks (SoftCOM), Split, Croatia, 15–17 September 2011; pp. 1–6.
64. Hui, J.; Kelsey, R. Multicast Protocol for Low power and Lossy Networks (MPL). draft-ietf-roll-trickle-mcast-02. 2012; pp. 1–24.
65. Oikonomou, G.; Phillips, I. Stateless Multicast Forwarding with RPL in 6LowPAN Sensor Networks. In Proceedings of the 2012 IEEE International Conference on Pervasive Computing and Communications Workshops (PerCom), City, Country, 19–23 March 2012; pp. 272–277.
66. Dawans, S.; Duquennoy, S.; Bonaventure, O. On Link Estimation in Dense RPL Deployments. In Proceedings of the International Workshop on Practical Issues in Building Sensor Network Applications (IEEE SenseApp 2012), Clearwater, FL, USA, 22–25 October 2012; pp. 956–959.
67. Goyal, M.; Baccelli, E.; Brandt, A.; Martocci, J. A Mechanism to Measure the Routing Metrics along a Point-to-point Route in a Low Power and Lossy Network. draft-ietf-roll-p2p-measurement-07. 2013; pp. 1–26.
68. Goyal, M.; Baccelli, E.; Philipp, M.; Brandt, A.; Martocci, J. Reactive Discovery of Point-to-Point Routes in Low Power and Lossy Networks. draft-ietf-roll-p2p-rpl-15. 2012; pp. 1–36.
69. Herberg, U.; Clausen, T. 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 the 8th ACM Symposium on Performance Evaluation of Wireless ad Hoc, Sensor, and Ubiquitous Networks (PE-WASUN), Miami Beach, FL, USA, 31 October–4 November 2011; ACM Press: New York, NY, USA, 2011; pp. 73–80.
70. Ko, J.; Gnawali, O.; Culler, D.; Terzis, A. Evaluating the Performance of RPL and 6LoWPAN in TinyOS. In Proceedings of the Workshop on Extending the Internet to Low power and Lossy Networks (IP+SN), Chicago, IL, USA, 11 April 2011.
71. Accettura, N.; Grieco, L.A.; Boggia, G.; Camarda, P. Performance analysis of the RPL Routing Protocol. In Proceedings of IEEE International Conference on Mechatronics, Istanbul, Turkey, 13–15 April 2011; pp. 767–772.

72. Bressan, N.; Bazzaco, L.; Bui, N.; Casari, P.; Vangelista, L.; Zorzi, M. The Deployment of a Smart Monitoring System Using Wireless Sensor and Actuator Networks. In Proceedings of the First IEEE International Conference on Smart Grid Communications (SMARTGRIDCOMM), Gaithersburg, Maryland, USA, 4–6 October 2010; pp. 49–54.
73. Chen, Y.; Chanet, J.P.; Hou, K.M. RPL Routing Protocol a case study: Precision agriculture. In Proceedings of the First China-France Workshop on Future Computing Technology (CF-WoFUCT 2012), Harbin, China, 16–17 February 2012.
74. Becker, M.; Pötsch, T.; Kuladinithi, K.; Görg, C. Deployment of CoAP in Transport Logistics. In Proceedings of 36th IEEE Conference on Local Computer Networks (LCN), Bonn, Germany, 4–7 October 2011; pp. 1–3.
75. Guo, J.; Orlik, P.; Bhatti, G. Loop Free DODAG Local Repair. draft-guo-roll-loop-free-dodag-repair-00. 2012; pp. 1–17.
76. Guo, J.; Orlik, P.; Bhatti, G. Loop Free RPL. draft-guo-roll-loop-free-rpl-01. 2013; pp. 1–20.
77. Ko, J.; Jeong, J.; Park, J.; Jun, J.; Kim, N. RPL Routing Pathology In a Network With a Mix of Nodes Operating in Storing and Non-Storing Modes. draft-ko-roll-mix-network-pathology-01. 2012, 1–9.
78. Baryun, A. The Node Ability of Participation (NAP). draft-baryun-roll-nap-00. 2013; pp. 1–9.
79. Goyal, M.; Barthel, D.; Baccelli, E. DIS Modifications. draft-goyal-roll-dis-modifications-01. 2013; pp. 1–11.
80. Hong, K.-S.; Choi, L. DAG-based multipath routing for mobile sensor networks. In Proceedings of the International Conference on ICT Convergence (ICTC), Seoul, Korea (South), 28–30 September 2011; pp. 261–266.
81. Lee, K.C.; Sudhaakar, R.; Ning, J.; Dai, L.; Addepalli, S.; Vasseur, J.P.; Gerla, M.A Comprehensive Evaluation of RPL under Mobility. *Int. J. Veh. Technol.* **2012**, *2012*, 1–10.
82. Carels, D.; Poorter, E.De; Moerman, I.; Demeester, P. Extending the IETF RPL routing protocol with mobility support. 2013. In press.
83. Shelby, Z.; Hartke, K.; Bormann, C.; Frank, B. Constrained Application Protocol (CoAP). draft-ietf-core-coap-13. 2012.
84. Colitti, W.; Steenhaut, K.; Caro, N. De Integrating Wireless Sensor Networks with the Web. In Proceedings of Workshop on Extending the Internet to Low power and Lossy Networks, Chicago, IL, USA, 11 April 2011.
85. Yazar, D.; Dunkels, A. Efficient Application Integration in IP-based Sensor Networks. In Proceedings of the First ACM Workshop on Embedded Sensing Systems for Energy-Efficiency in Buildings, BuildSys ’09, Berkeley, USA, 4–6 November 2009; ACM Press: New York, NY, USA, 2009; p. 43.
86. Shelby, Z. Embedded web services. *IEEE Wirel. Commun.* **2010**, *291*, 76–81.
87. Shelby, Z. Constrained RESTful Environments (CoRE) Link Format. IETF RFC 6690. 2012.
88. Bormann, C.; Shelby, Z. Blockwise transfers in CoAP. draft-ietf-core-block-10. 2012.
89. Hartke, K. Observing Resources in CoAP. draft-ietf-core-observe-07. 2012.
90. *1st CoAP Plugtest*; Technical Report CTI Plugtest Report 1.1.1; 2012.

91. Velez, L. IoT COAP#2 Interop Event Preliminary Report. Available online: http://svn.tools.ietf.org/svn/wg/core/Preliminary-Results-CoAP%232.pdf (accessed on 28 December 2012).
92. Lerche, C.; Hartke, K.; Kovatsch, M. Industry Adoption of the Internet of Things: A Constrained Application Protocol Survey. In Proceedings of the 7th International Workshop on Service Oriented Architectures in Converging Networked Environments (SOCNE 2012); Kraków, Poland, 17–21 September 2012.
93. Castellani, A.P.; Gheda, M.; Bui, N.; Rossi, M.; Zorzi, M. Web Services for the Internet of Things through CoAP and EXI. In Proceedings of IEEE International Conference on Communications Workshops (ICC), Kyoto, Japan, 5–9 June 2011; pp. 1–6.
94. Californium (Cf) CoAP framework in Java. Available online: http://people.inf.ethz.ch/mkovatsc/californium.php (accessed on 28 December 2012).
95. Copper (Cu) Add-ons for Firefox. Available online: https://addons.mozilla.org/en-us/firefox/addon/copper-270430/ (accessed on 28 December 2012).
96. Erbium (Er) REST Engine and CoAP Implementation for Contiki. Available online: http://people.inf.ethz.ch/mkovatsc/erbium.php (accessed on 28 December 2012).
97. Ishaq, I.; Hoebeke, J.; Rossey, J.; De Poorter, E.; Moerman, I.; Demeester, P. Facilitating Sensor Deployment, Discovery and Resource Access Using Embedded Web Services. In Proceedings of the Sixth International Conference on Innovative Mobile and Internet Services in Ubiquitous Computing, Palermo, Italy, 4–6 July 2012; pp. 717–724.
98. evcoap. Available online: https://github.com/koanlogic/webthings/tree/master/bridge/sw/lib/evcoap (accessed on 29 December 2012).
99. NanoService. Sensinode Ltd. Available online: http://www.sensinode.com/EN/products/nanoservice.html (accessed on 22 December 2012).
100. libcoap: C-Implementation of CoAP. Available online: http://libcoap.sourceforge.net/ (accessed on 22 December 2012).
101. CoAP. TinyOS Documentation Wiki. Available online: http://docs.tinyos.net/tinywiki/index.php/CoAP (accessed on 22 December 2012).
102. coap.me. Available online: http://coap.me/ (accessed on 22 December 2012).
103. jcoap is a Java Library implementing the Constrained Application Protocol (CoAP)—Google Project Hosting. Available online: http://code.google.com/p/jcoap/ (accessed on 22 December 2012).
104. Constraint Application Protocol (CoAP) for ERIKA embedded OS. Available online: http://rtn.sssup.it/index.php/research-activities/middleware-of-things/middleware-of-things/11-research-activities/35-coaperika (accessed on 22 December 2012).
105. CoAPy: Constrained Application Protocol in Python—CoAPy v0.0.2 documentation. Available online: http://coapy.sourceforge.net/ (accessed on 29 December 2012).
106. ibr-alg/wiselib. GitHub. Available online: https://github.com/ibr-alg/wiselib (accessed on 22 December 2012).

107. Colitti, W.; Steenhaut, K.; De Caro, N.; Buta, B.; Dobrota, V. Evaluation of Constrained Application Protocol for Wireless Sensor Networks. In Proceedings of the 18th IEEE Workshop on Local & Metropolitan Area Networks (LANMAN), Chapel Hill, NC, USA, 13–14 October 2011; pp. 1–6.
108. Duquennoy, S.; Wirström, N.; Tsiftes, N.; Dunkels, A. Leveraging IP for Sensor Network Deployment. In Proceedings of the workshop on Extending the Internet to Low power and Lossy Networks (IP+SN 2011), Chicago, IL, USA, 11 April 2011.
109. Pötsch, T. Performance of the Constrained Application Protocol for Wireless Sensor Networks. Available online: http://www.comnets.uni-bremen.de/itg/itgfg521/aktuelles/fg-workshop-29092011/ITG_HH_thomas_poetsch.pdf (accessed on 29 December 2012).
110. Kuladinithi, K.; Bergmann, O.; Pötsch, T.; Becker, M.; Görg, C. Implementation of CoAP and its Application in Transport Logistics. In Extending the Internet to Low power and Lossy Networks' (IP+SN 2011), Chicago, IL, USA, 11 April 2011.
111. Bormann, C. CoRE Roadmap and Implementation Guide. draft-bormann-core-roadmap-03. 2012.
112. Shelby, Z.; Krco, S.; Bormann, C. CoRE Resource Directory. draft-shelby-core-resource-directory-04. 2012.
113. Rahman, A.; Dijk, E. Group Communication for CoAP. draft-ietf-core-groupcomm-04. 2012.
114. Bormann, C. CoAP Simple Congestion Control/Advanced. draft-bormann-core-cocoa-00. 2012.
115. Gurtov, A.; Dashkova, E. Computing the Retransmission Timeout in COAP. Available online: http://www.etsi.org/plugtests/COAP2/Presentations/08_Computing_Retransmission_Timeout.pdf (accessed on 20 December 2012).
116. Greevenbosch, B. CoAP Minimum Request Interval. draft-greevenbosch-core-minimum-request-interval-00. 2012.
117. Li, S.; Hoebeke, J.; Jara, A.J. Conditional observe in CoAP. draft-li-core-conditional-observe-02. 2012.
118. Ketema, G.; Hoebeke, J.; Moerman, I.; Demeester, P. Efficiently observing Internet of Things Resources. In Proceedings of The IEEE International Conference on Cyber, Physical and Social Computing, Besançon, France, 20–23 November 2012.
119. Vial, M. CoRE Mirror Server. draft-vial-core-mirror-server-00. 2012.
120. Hoebeke, J.; Carles, D.; Ishaq, I.; Ketema, G.; Rossey, J.; De Poorter, E.; Moerman, I.; Demeester, P. Leveraging upon Standards to Build the Internet of Things. In Proceedings of the 19th IEEE Symposium on Communications and Vehicular Technology in the Benelux; Eindhoven, The Netherlands, 16 November 2012.
121. Rahman, A. Enhanced Sleepy Node Support for CoAP. draft-rahman-core-sleepy-01. 2012.
122. Application Layer Protocol Support for Sleeping Nodes in Constrained Networks. US Patent 20120151028. Available online: http://www.google.com/patents/US20120151028 (accessed on 23 December 2012).
123. Tschofenig, H. Report from the Smart Object Security Workshop. Available online: http://www.ietf.org/proceedings/83/slides/slides-83-saag-3.pdf (accessed on 18 December 2012).
124. Garcia-Morchon, O.; Keoh, S.; Kumar, S.; Hummen, R.; Struik, R. Security Considerations in the IP-based Internet of Things. draft-garcia-core-security-04. 2012.

125. Bergmann, O.; Gerdes, S.; Schafer, S.; Junge, F.; Bormann, C. Secure Bootstrapping of Nodes in a CoAP Network. In Proceedings of the IEEE Wireless Communications and Networking Conference Workshops (WCNCW), Paris, France, 1 April 2012; pp. 220–225.
126. Hartke, K.; Bergmann, O. Datagram Transport Layer Security in Constrained Environments. draft-hartke-core-codtls-02. 2012.
127. Raza, S.; Trabalza, D.; Voigt, T. 6LoWPAN Compressed DTLS for CoAP. In IEEE 8th International Conference on Distributed Computing in Sensor Systems, Hangzhou, China, 16–18 May 2012; pp. 287–289.
128. Tschofenig, H.; Gilger, J. A Minimal (Datagram) Transport Layer Security Implementation. draft-tschofenig-lwig-tls-minimal-01. 2012.
129. Brachmann, M.; Garcia-Morchon, O.; Kirsche, M. Security for Practical CoAP Applications: Issues and Solution Approaches. In Proceedings of the 10th GI/ITG KuVS Fachgespraech Sensornetze (FGSN11), Paderborn, Germany, 15–16 September 2011.
130. Kivinen, T. Minimal IKEv2. draft-kivinen-ipsecme-ikev2-minimal-01. 2012.
131. Castellani, A.; Loreto, S.; Rahman, A.; Fossati, T.; Dijk, E. Best Practices for HTTP-CoAP Mapping Implementation. draft-castellani-core-httSp-mapping-05. 2012.
132. Castellani, A.; Loreto, S.; Rahman, A.; Fossati, T.; Dijk, E. Best Practices for HTTP-CoAP Mapping Implementation. draft-castellani-core-advanced-http-mapping-00. 2012.
133. Becker, M.; Li, K.; Kuladinithi, K.; Poetsch, T. Transport of CoAP over SMS, USSD and GPRS. draft-becker-core-coap-sms-gprs-02. 2012.
134. Kovatsch, M.; Mayer, S.; Ostermaier, B. Moving Application Logic from the Firmware to the Cloud: Towards the Thin Server Architecture for the Internet of Things. In Proceedings of the Sixth International Conference on Innovative Mobile and Internet Services in Ubiquitous Computing, Palermo, Italy, 4–6 July 2012; pp. 751–756.
135. Hans, S. Secure Environment management based on CoAP. In Proceedings of the *ETSI CoAP Workshop*; Sophia Antipolis, France, 27–30 November 2012.
136. Castro, M.; Jara, A.J.; Skarmeta, A. Architecture for improving terrestrial logistics based on the Web of Things. *Sensors* **2012**, *12*, 6538–6575.
137. Mäenpää, J.; Bolonio, J.; Loreto, S. Using RELOAD and CoAP for wide area sensor and actuator networking. *EURASIP J. Wirel. Commun. Netw.* **2012**, *2012*, 121.
138. Rahman, A.; Gellert, D.; Seed, D.N. A Gateway Architecture for Interconnecting Smart Objects to the Internet. In Proceedings of the Workshop Interconnecting Smart Objects with the Internet, Prague, Czech Republic, 25 March 2011.
139. Villaverde, B.C.; Pesch, D.; De Paz Alberola, R.; Fedor, S.; Boubekeur, M. Constrained Application Protocol for Low Power Embedded Networks: A Survey. In Proceedings of the Sixth International Conference on Innovative Mobile and Internet Services in Ubiquitous Computing, Palermo, Italy, 4–6 July 2012; pp. 702–707.
140. Barbieri, D. CoAP Improvements to Meet Embedded Device Hardware Constraints. In Proceedings of the Workshop Interconnecting Smart Objects with the Internet, Prague, Czech Republic, March 2011.
141. Yahyaoui, H.; Maamar, Z.; Boukadi, K. A framework to coordinate web services in composition scenarios. *Int. J. Web Grid Serv.* **2010**, *6*, 95–123.

142. Gao, L.; Urban, S.D.; Ramachandran, J. A survey of transactional issues for Web Service composition and recovery. *Int. J. Web Grid Serv.* **2011**, *7*, 331.
143. Kim, W. Cloud computing adoption. *Int. J. Web Grid Serv.* **2011**, *7*, 225–245.
144. Rodriguez, J.M.; Zunino, A.; Campo, M. Introducing mobile devices into Grid systems: A survey *Int. J. Web Grid Serv.* **2011**, *7*, 1–40.
145. Dunkels, A.; Eriksson, J.; Tsiftes, N. Low-power Interoperability for the IPv6-Based Internet of Things. In Proceedings of the 10th Scandinavian Workshop on Wireless *Ad-Hoc* Networks (ADHOC'11), Stockholm, Sweden, 10–11 May 2011.
146. Barnaghi, P.; Wang, W.; Henson, C.; Taylor, K. Semantics for the Internet of Things. *Int. J. Semant. Web Inf. Syst.* **2012**, *8*, 1–21.
147. Shelby, Z.; Chauvenet, C. The IPSO Application Framework. draft-ipso-app-framework-04. 2012.
148. Abdulrazak, B.; Chikhaoui, B.; Vallerand, C.G.; Fraikin, B. A standard ontology for smart spaces. *Int. J. Web Grid Serv.* **2010**, *6*, 244.
149. Pfisterer, D.; Romer, K.; Bimschas, D.; Kleine, O.; Mietz, R.; Truong, C.; Hasemann, H.; Kröller, A.; Pagel, M.; Hauswirth, M.; *et al*. SPITFIRE: Toward a semantic web of things. *IEEE Commun. Mag.* **2011**, *49*, 40–48.
150. SweoIG/TaskForces/CommunityProjects/LinkingOpenData. W3C Wiki. Available online: http://www.w3.org/wiki/SweoIG/TaskForces/CommunityProjects/LinkingOpenData (accessed on 21 December 2012).
151. Cosm. Internet of Things Platform Connecting Devices and Apps for Real-Time Control and Data Storage. Available online: https://cosm.com/ (accessed on 21 December 2012).
152. solace Info Page. Available online: https://www.ietf.org/mailman/listinfo/solace (accessed on 27 December 2012).
153. Ishaq, I.; Hoebeke, J.; Moerman, I.; Demeester, P. Internet of Things Virtual Networks: Bringing Network Virtualization to Constrained Devices. In Proceedings of the IEEE International Conference on Cyber, Physical and Social Computing, Besançon, France, 20–23 November 2012.
154. The DisSeNT project. Available online: http://distrinet.cs.kuleuven.be/software/dissent/ (accessed on 10 December 2012).