Universal Integration of the Internet of Things through an IPv6-based Service Oriented Architecture enabling heterogeneous components interoperability

Grant agreement for: Collaborative project
Grant agreement no.: 288445
Start date of project: January 1st, 2011 (36 months duration)

**Deliverable 2.3**
Report on IPv6 based advanced features

<p>| | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>Contract Due Date</td>
<td>30/07/2013</td>
</tr>
<tr>
<td>Submission</td>
<td>15/07/2013</td>
</tr>
<tr>
<td>Version</td>
<td>1.0</td>
</tr>
<tr>
<td>Responsible Partner</td>
<td>University College London (UCL)</td>
</tr>
<tr>
<td>Dissemination level</td>
<td>PU</td>
</tr>
<tr>
<td>Authors</td>
<td>Socrates Varakliotis, Peter Kirstein, Rafael Marin-Perez, Antonio Jara, Antonio Skarmeta</td>
</tr>
<tr>
<td>Internal Reviewers</td>
<td>Jérôme Francois, Foued Melakessou</td>
</tr>
<tr>
<td>Keywords</td>
<td>Internet of Things, IPv6, QoS, security, mobility, ubiquitous access, scalable and efficient routing, IoT gateways, service networks</td>
</tr>
</tbody>
</table>

Task and Report Leader: University College London (UCL)
Project Coordinator: Mandat International (MI)
# Table of Contents

1 Executive Summary ................................................................................................................. 7

2 Introduction .............................................................................................................................. 7
   2.1 Purpose and scope of the document ................................................................................. 7
   2.2 Task T2.3 ......................................................................................................................... 8
   2.3 Structure of the document ............................................................................................... 8

3 Architectural advances ............................................................................................................. 9
   3.1 Introduction ...................................................................................................................... 9
   3.2 Gateway and Server Architecture to IoT6 ................................................................. 12
       3.2.1 Protocol Levels in Gateways and Servers ................................................................. 12
       3.2.2 Server, Gateways and Clusters and Information Distribution .......................... 14
   3.3 The services on the ServiceNet ....................................................................................... 15

4 IPv6-based features ..................................................................................................................16
   4.1 Introduction ...................................................................................................................... 16
   4.2 IPv6 basic and advanced functions ............................................................................... 17
   4.3 IPv6 basic functions ....................................................................................................... 17
       4.3.1 Address Features ..................................................................................................... 18
       4.3.2 Auto-Configuration .................................................................................................. 19
   4.4 IPv6 advanced functions ............................................................................................... 19
       4.4.1 Security .................................................................................................................. 19
       4.4.2 Mobility ................................................................................................................ 20
   4.5 IoT protocols supported by IPv6 .................................................................................. 20
       4.5.1 Web Services .......................................................................................................... 20
   4.6 IPv6 advanced features in IoT6 ..................................................................................... 21

5 Ubiquitous Access .................................................................................................................... 22
   5.1 Introduction ...................................................................................................................... 22
   5.2 How IoT6 architectural advances deliver ubiquitous access .................................... 22
   5.3 Specific Feature Distribution between Server Repository and Gateways .................. 23
   5.4 Scenarios of User and Server Access to Gateways .................................................... 25
       5.4.1 Resilience ............................................................................................................... 25
       5.4.2 Security ................................................................................................................ 25
       5.4.3 Energy-efficient Operation ..................................................................................... 25
   5.5 Energy efficiency ............................................................................................................ 26
       5.5.1 ‘Energy efficiency’ notion in the 3-tier architecture ............................................. 26
       5.5.2 ‘Energy efficiency’ notion by use of IPv6 on the fringes ...................................... 26
5.5.3 ‘Energy efficiency’ notion by utilising generic half-gateway abstractions........29
5.6 IPv6 based self-configuration, self-management, self-healing.........................30
  5.6.1 Self-configuration......................................................................................30
  5.6.2 Self-management.......................................................................................30
  5.6.3 Self-healing...............................................................................................31
6 Mobility in IPv6 sensors....................................................................................31
  6.1 Mobility protocol trends................................................................................31
  6.2 Design Issues...............................................................................................33
  6.3 Mobility management for the Internet of Things.........................................34
  6.4 Mobile IPv6.................................................................................................35
    6.4.1 Mobile IPv6 requirements........................................................................35
    6.4.2 Mobile IPv6 analysis...............................................................................36
  6.5 Lightweight Mobile IPv6...............................................................................37
    6.5.1 Movement detection................................................................................37
    6.5.2 Care-of address configuration and binding management.......................38
    6.5.3 Data communication..............................................................................43
  6.6 IPsec support in Lightweight Mobile IPv6....................................................48
    6.6.1 IPsec Analysis.........................................................................................48
    6.6.2 IPsec integration in Mobile IPv6.............................................................49
    6.6.3 Crypto-suites support for IPsec in lightweight Mobile IPv6....................49
    6.6.4 IPsec Format..........................................................................................49
    6.6.5 Binding management with IPsec............................................................51
    6.6.6 Data communication with IPsec............................................................53
  6.7 Evaluation......................................................................................................55
    6.7.1 Evaluation testbed..................................................................................55
    6.7.2 Memory footprint....................................................................................59
    6.7.3 Overhead................................................................................................59
    6.7.4 Movement detection..............................................................................60
    6.7.5 Handover latency...................................................................................61
    6.7.6 Transmissions........................................................................................64
  6.8 Conclusions..................................................................................................66
7 Scalable and Efficient Routing in IoT6 Wireless Networks.................................67
  7.1 Existing routing paradigms............................................................................68
    7.1.1 Data-centric routing...............................................................................69
    7.1.2 Hierarchical routing...............................................................................71
    7.1.3 Quality-of-Service routing......................................................................73
7.1.4 Geographic routing .................................................................75
7.1.5 Gradient routing .................................................................75
7.1.6 Routing paradigms comparison ..............................................78
7.2 Background on geographic routing ..........................................80
7.2.1 Network Model and Assumptions ...........................................80
7.2.2 Greedy Forwarding .............................................................80
7.2.3 Face Routing in Planar Graphs ..............................................81
7.2.4 Combined Greedy-Face Routing ..........................................83
7.2.5 Beaconless Geographic Routing .........................................84
7.2.6 Advantages and Disadvantages .........................................86
7.3 Scalable, Efficient and Reliable (SERE) Routing for IoT6 WSNs. ....87
7.3.1 Analysis of Wireless Communications .................................88
7.3.2 Design of SERE Routing Protocol .......................................90
7.4 IPv6 transition mechanisms on gateways for the IoT..................93
8 Unicast/Multicast/Anycast for IoT6 Sensor Networks ....................93
8.1 Priority management .............................................................94
8.2 Addressing redirection ..........................................................95
9 Conclusions and future work ...................................................96
10 References ..............................................................................98

List of Tables
Table 1: Lightweight protocols implemented for the IoT........................21
Table 2: Binding Update overhead analysis considered along this document ..........54
Table 3: Binding Acknowledgement overhead analysis considered along this document ..........54
Table 4: UDP CoAP communication overhead analysis considered along this document ..........58
Table 5: Overhead analysis when using ESP Security ciphered with AES-CBC or AES-CCM mode .60
Table 6: Advantages and Disadvantages of Routing Paradigms .......................78
Table 7: Traffic classes for packet prioritisation ..................................95

List of Figures
Figure 1: Network View of the Internet of Things from D1.1 and D2.1 ....................9
Figure 2: Schematic of the Networks, Servers, Gateways and Devices in the IoT and IoT6 ..........10
Figure 3: The notation used for gateways and servers .................................................................13
Figure 4: Schematic of Networks, Servers and Gateways in the Internet of Things ....................14
Figure 5: IoT6 communication architecture and ecosystem ........................................................23
Figure 6: Radio power consumption of Transmit and Receive cycles using Contiki power tracing and profiling ..................................................................................................................27
Figure 7: Power consumption of a constrained node’s resources using Contiki ‘Energist’ (in RIME stack) ........................................................................................................................................28
Figure 8: Metrics triangle for the different mobility support solutions over 6LoWPAN ..................34
Figure 9: Mobile IPv6 packet format ..............................................................................................35
Figure 10: Sequence of phases in Lightweight Mobile IPv6 ..........................................................37
Figure 11: Care-of-address registration .........................................................................................38
Figure 12: a) Binding Update Message in Mobile IPv6. b) Binding Update Message in Mobile IPv6 with 6LoWPAN header with the original Contiki OS implementation ..................................................40
Figure 13: Binding Update Message proposed for the Lightweight Mobile IPv6 with 6LoWPAN header and next header compressed for the destination option and the mobility headers ............41
Figure 14: a) Binding Acknowledgement Message in Mobile IPv6. b) Binding Acknowledgement Message in Mobile IPv6 with 6LoWPAN header with the original Contiki OS implementation ........42
Figure 15: Binding Acknowledgment Message proposed for the Lightweight Mobile IPv6 with 6LoWPAN header and next header compressed for the routing and the mobility headers ..........43
Figure 16: Communication exchange tunnelled via the HA ..........................................................44
Figure 17: a) Packet from the MN to the CN via the HA (**1) based on 6LoWPAN header compression. b) Packet from the MN to the CN via the HA (**1) based on IPv6 ........................................45
Figure 18: a) Packet from the MN to the CN via the HA (*1). b) Packet from the CN to the MN via the HA (*2) ....................................................................................................................................46
Figure 19: a) Packet from the CN to the MN via the HA (**2). b) Packet from the CN to the MN via the HA (**2) based on 6LoWPAN header compression .....................................................................47
Figure 20: Encapsulation modes for data communication during roaming ....................................48
Figure 21: IPsec packet format .....................................................................................................51
Figure 22: a) Binding Update Message in Mobile IPv6 protected with ESP. b) Binding Acknowledgement Message in Mobile IPv6 protected with ESP ........................................................................53
Figure 23: a) Packet from the MN to the CN via the HA (**1) based on IPv6 using ESP tunnel. b) Packet from the MN to the CN via the HA (**1) based on 6LoWPAN header compression using ESP tunnel ..............................................................................................56
Figure 24: a) Packet from the CN to the MN via the HA (**2) based on IPv6 using ESP tunnel. b) Packet from the CN to the MN via the HA (**2) based on actual Contiki 6LoWPAN header compression using ESP tunnel .............................................................................57
Figure 25: Testbed for mobility evaluation ....................................................................................58
Figure 26: Time for the different scan times considering all the channels ..................................61
Figure 27: Scan time with energy sensing to detect if the MN is moving and discover new networks.

Figure 28: a) Time to establish the first association in the home network. b) Time to receive the first router advertisement in the home network.

Figure 29: a) Time to establish the association after scan in the visited network. b) Time to receive the router advertisement in the visited network. c) Time to send the BU and receive the BA.

Figure 30: Handover time for the Lightweight Mobile IPv6 protocol without scan phase.

Figure 31: Full handover time for the Lightweight Mobile IPv6 protocol.

Figure 32: Time impact when using ESP security during UDP transmissions of several payload lengths.

Figure 33: Example of gradient routing where nodes are referenced by the tuple [Identifier, Height].

Figure 34: Greedy Routing Scheme (GRS), Compass Routing (CR) and Most Forward within Radius (MFR).

Figure 35: Face routing in a planar graph.

Figure 36: GG (left) and RNG (right).

Figure 37: A data packet is forwarded from the source S toward the destination D using greedy and face mode.

Figure 38: The forwarding area must be defined so that all nodes inside it can hear one another. The Reuleaux Triangle fulfils the condition of mutual possible reception for nodes located within it. Node S holding a message intended for D has three neighbours (N1, N2, N3). Only N2 is located inside the forwarding area defined by the Reuleaux Triangle. As it can be seen, transmissions from N1 cannot be overheard neither by N2 nor by N3.

Figure 39: The contention timer is used to minimise the number of transmissions in the three way handshake and fully distributed contention. In that case, the first transmission of N2 cancels the one of N1.

Figure 40: Relation of the measured parameters RSSI, LQI and PRR at varying the distance between sender and receiver.

Figure 41: PRR values at varying the packet size in distances between 44 and 51 meters.

Figure 42: Example of an IoT6 network with smart objects, wireless routers and a half-gateway.

Figure 43: Example of the reactive routing discovery scheme.

Figure 44: The 8-bit field of traffic class in IPv6 header.

Figure 45: Example of addressing redirection from sensor network to IPv6 servers through the half-gateway.
1 Executive Summary

This document reports the IPv6-based advanced features in the context of the WP1 architecture and the more generic requirements of the IoT-A Project [10]. Concretely, we reiterate the initial distinction between the set of ‘basic’ IPv6 features and mechanisms (as previously reported mainly in D2.1) and subsequently we expand on ‘advanced features’ that provide better support for functions such as mobility, ubiquitous access and multicast in IoT6 scenarios. Moreover, in the Internet of Things environment, these features must be satisfied in a scalable form. In this deliverable, how these requirements can be satisfied, with help from IPv6 wherever appropriate, on devices which are constrained in their power availability and processing capability are considered. How these requirements may be applicable to more capable devices/hosts, such as the half-gateways are also examined. What is highlighted is where the application of basic and advanced features requires support from the infrastructure (such as the use of processes on servers) and where this approach may have architectural impact in the IoT6 core architecture, as specified in D1.1 and revised in D1.3. A range of features identified has been implemented in some of the subsystems available in IoT6. Therefore, some of the analyses of the different features, and implementation details that we consider particularly relevant to IoT6 are described.

In view of the need to validate our conclusions in practical applications, e.g. in the wider context of Smart Cities, we have analysed and in some cases implemented many of the features considered including the ones below:

- Basic features of IPv6 including addressing and auto-configuration
- Advanced features of IPv6 including security and mobility
- Advanced services enabled by these and related protocols including mobility, security, proxies, web services, energy efficiency, quality of service and efficient routing.

An important aspect of the entire IoT6 project is how to integrate legacy sensor and actuator systems in our Internet of Things concept. The use of proxies will allow much of the functionality needed for the integration stated above to be located in servers rather than in the gateways themselves. We postulate that this will result in much simpler gateways that are more maintainable, and only require comparatively small additions to the current legacy systems controllers. This should make the systems easier to maintain. Our design makes heavy use of repositories for key configuration and technology information.

While implementations underpin many of the aspects of this Deliverable, there will be full experimental validation later in the project.

2 Introduction

2.1 Purpose and scope of the document

The IoT6 project is at a crucial stage – particularly with regards to its underlying technology. As the project approaches the end of its second year, our underlying concepts must be completed, and only the implementation and validation of its soundness should remain. This Deliverable has, therefore, several distinct aims and threads. Firstly, it is to present all the IP aspects, particularly those of IPv6, that are important for IoT. Secondly, it is to summarise the complete set of implementations that are available to the other IoT6 work packages for use in the integration and validation stages of the project. Thirdly, it is to describe the architectural innovations that arise from the earlier work, and which might require experimental verification.
The first thread is complete; all the elements at the network level that will be pursued further in IoT6 are presented in this Deliverable, have been presented in earlier Deliverables, or are part of the Deliverables of other work packages; the complete bundle of protocols extends at least into the use of transport levels. These implementations are complete enough for the integration into the higher-level facilities of other work packages, though both in security and mobility some further implementations may be needed for specific Use Cases. The implementations not only of WP2 and WP3, but also part of WP4 and WP6 need to be completed before it was possible to pursue a complete implementation of some aspects of the third thread. However, enough of the implementations in these WPs are covered to allow fairly complete implementation and validation of the third thread. The remaining implementation will be carried out under the last Task (2.4) of WP2. We expect to quantify some of the advantages ensuing at a later stage.

2.2 Task T2.3

Task T2.3 description taken from the IoT6 Description of Work (DoW):

| T2.3: IPv6-based mobility, ubiquitous access, multicast and scalability M11-M22 (Task leader: UCL) |
| This task will research and address the implementation of features such as mobility, ubiquitous access, multicast and scalability. Mobility is one of the most relevant features of IPv6; it needs to be supported in the Internet of Things applications to reach a continuous and seamless connectivity. It is hence of importance to provide a mobility protocol with security support for the IPv6 nodes of this architecture and extend it to the Internet of Things. This task will research mobility features enabled by IPv6. It will explore and test ubiquitous and seamless IPv6-based access to the Internet of Things across different locations (Geneva, London, Murcia, etc.), including access to mobile small and smart things such as 6LoWPAN-based devices, since mobility support for the Future Internet of Things is required to satisfy the dependability and scalability of the Future Internet and this is recommendable since such as mentioned increases the fault tolerance capacity of the network, increases the connectivity between nodes and clusters, allows to extend and adapt network to changes of their location and environment infrastructure, and the deployment of multiple controlled mobile elements can be used to provide load balancing. Finally, this task will address IPv6 multicast potential and will assess the architecture’s potential scalability. |

2.3 Structure of the document

This document starts with an in-depth discussion of architectural advances for the IoT in section 3. Section 4 revisits IPv6-based features, functions and services and makes the distinction between basic and advanced sets. The advanced features of Ubiquity and, Mobility with Security for Wireless Sensor Networks (WSNs) are analysed, and their implementation aspects described, respectively in sections 5 and 6. Section 7 delves into the Scalability aspects of the architecture, while section 8 expands on advanced addressing features concerned with multicast and anycast and demonstrates their use in a scenario with priority management. The Deliverable presents concluding comments in section 9.
3 Architectural advances

3.1 Introduction

Due to the fact that the Internet of Things (IoT) is such an all-embracing concept, it is difficult to give a detailed and useful architecture unless one is specific about what classes of object one is dealing with. Even by taking a network view of this Internet, there would still be many very distinct sets of problems and scenarios. In WP1, an architecture diagram was presented which is repeated in Figure 1 below.

![Figure 1: Network View of the Internet of Things from D1.1 and D2.1](image)

In this view, three levels of network are defined: The normal Internet on the left side (denoted with the term ‘InterNet’), the IPv6-enabled Local Network (termed, in short, ‘LocalNet’) and the various sensor networks on the right side (termed ‘SensorNets’). We took the view that we could not influence the rate of introduction of IPv6 in the general Internet, and therefore assumed that the left network could be IPv4 and/or IPv6. The right network is the set of sensor networks; again many of these are still in legacy mode, and their technology could not be impacted. Many sensor networks will not even consider IP – though they could be provided with a gateway to the LocalNet, which would be more under the control of the specific application domain. Hence, for IoT6 purposes, we postulated that we could consider situations in which this network was operating in IPv6 mode and with the IPv6 features desired.

In a more recent paper [1], a more realistic generic picture is given by modifying Figure 1 into Figure 2 as shown below (the Use Cases given as examples in Deliverables D1.1 and D2.1 match this architecture well):
There are several differences between Figure 1 and Figure 2. In addition to the normal Internet, a ServiceNet and a DevNet are shown. These terms are used, because the salient property of the DevNet is to control devices, and hence this term is considered more appropriate in this Deliverable. The middle network is called a ServiceNet, because that is really what it is. It will usually be “fairly” local – but not necessarily local as is normally implied by “Local Area Network” or “LAN”. Which services should be located on the ServiceNet and which on the Internet is very domain-, service- and application-dependent. To better understand the above nomenclature, consider the following example. A company offers a ‘LightAsAService’ (LaaS) service, which provides municipal lighting in a smart city scenario. The company is not necessarily a utility company in its own right. They rather buy/lease light from a utility and re-sell it as a service. They offer LaaS to smaller customers, such as schools, private garages, etc. Obviously, the distribution and control networks of these services can be different infrastructures, but they can also be overlays over the same infrastructure. Each overlay is managed and controlled in a different network domain, by a different owner, servicing the particular customer through this ServiceNet. This example shows clearly that the ServiceNet extends further than a LAN, in conventional networking parlance. For reasons to be discussed below and in the following paragraph, we even permit the ServiceNet to be dual-stack – though of course the trend should be to have all the components IPv6-enabled. The main reason why we will sometimes consider the ServiceNet to be IPv4 enabled is related to resource constraints and legacy systems. In many of the WPs in IoT6, we postulate gateways providing translations between the mechanisms of operation of the legacy systems and those of the Internet. This architecture is very important, and is indeed the main thrust in the other work packages. However, in WP2, a somewhat more general view of gateway services is taken. It is assumed that the high-level services will be delivered through web services – which depend critically on HTTP accessed directly on the gateway. However, in order to simplify the operation of the gateway itself, it is possible to carry out some of the
operations in servers on ServiceNet, which communicate directly with the gateway. Moreover, many of the legacy systems are now being provided by the suppliers with IP interfaces – though currently only IPv4. Dual stack operations obviates the need to perform unnecessary IPv4-IPv6 translations because the communication between these gateways and the previously mentioned servers does not require global addressing and is quite specific.

Web services depend critically on HTTP. To deal with constrained resource devices – like sensor networks (and particularly wireless ones), a protocol called CoAP [44] has been defined. There is broad agreement that the reduced functionality of CoAP [44] is well aligned to resource-poor devices and is easy to translate to HTTP. CoAP is not dependent on the IP version used and operates over IPv4 and IPv6. From the outset, CoAP allows for proxies to take over some of the functionality of other instantiations. Normally in IoT6, all gateway functionality is in a single gateway, and its access is via HTTP or CoAP over IPv6. This new approach will allow most of the configuration and control functions to be carried out in a proxy server, requiring only CoAP to be added to the controller provided by the supplier – even if that supports only IPv4. Whether the proxy server is located on the ServiceNet or on the general Internet, it will be application-dependent, but for IoT6 purposes it can certainly be assumed, that all such proxies are on IPv6-enabled devices. Any HTTP – CoAP translation would also be done on the server.

In Figure 1 and Figure 2, the access to a specific application space of interest is via the normal Internet. A potentially huge number of Things, shown on the right of the diagram, may be controlled through DevNet. Control of the devices through DevNet is called in the figure a Controller Gateway. Between the normal Internet and the DevNet there is a ServiceNet which carried out most of the functions concerned with this application domain.

Depending on the application domain, the ServiceNet may span different regions, e.g. a home, an office, a mobile telephone region, the distribution area of a smart grid, a smart city, or a building automation system. The devices themselves may have very different properties. For example, they may be the current conventional sensors and actuators in a building: e.g. temperature gauges, door locks, motion sensors or fairly dumb meters. If this is the case, they may well be accessed via the proprietary networks at the bottom of Figure 1. On the other hand, some more futuristic devices can be much smarter. They may be IP-addressable, and contain an Internet stack. This last device category is an important class of smart objects, which many of us think will become so prevalent that they will be the dominant objects in the future IoT. Indeed there is already an industrial alliance called IPSO [7] specifically targeted at small objects and in the future we will deal mainly with such “smart objects” as a set, or individually, being IPv6-enabled.

One very important aspect of this architecture is that each individual device may be represented by an individual IPv6 address, whether or not it is really directly on an IP network. This allows the full power of IPv6 address operations to be used e.g. Multicast, Anycast, Name-Address mapping, etc. It is possible that the operations can be done directly; in many cases, however, they will be done by proxies. Whether these proxies run on the control gateway directly, or on a remoter server, is an implementation and application decision. Different legacy networks will allow different types of operation with different semantics. It is also possible to virtualise many of the operations in a technology-independent fashion in applications servers. For example, Sensors as an Application Service (SaaS) has been defined in [12]. This defines many of the normal operations like read a sensor, operate an actuator, and configure a set of sensors in a technology-independent way. Indeed CoAP already has most of these operations as its primitives.

Another EC FP7 project called CALIPSO [13] is specifically targeting IP-enabled small objects and because of their number, the use of IPv6 is obligatory if they are expected to have
unique addresses. In that sort of environment, an IPv6-based network stack in each of the objects is a very desirable feature. Moreover, the technology of the network accessing the small object is of very great importance.

While our work in IoT6 encompasses this environment, a different set of problems is being re-targeted, one in which legacy systems still need to address objects. In this environment, there is a broad set of legacy networks controlling sensors and actuators with legacy interfaces, usually not IP-enabled. In this case, there is no control of the network accessing the sensors and actuators, and can interact only with their controllers. Increasingly, these have IPv4 interfaces to the outside world so that they can be controlled remotely. It is this environment that is illustrated in the bottom right-hand network of Figure 1.

In this section, some of the implications that the class of problem being tackled have on the components of interest are discussed. When dealing with IP-enabled smart objects, there is a strong interest in the features of the DevNet, and the communication-oriented protocol stacks on the devices themselves. This is the prime motivation in CALIPSO. In IoT6, the legacy networks are largely unchangeable, so that there is little reason to study DevNet or the stacks on the devices themselves. There the interest is in the operations of the controllers, and how one can interface to these controllers in a comparatively technology-independent way over an IP-enabled ServiceNet. Thus it is the set of gateways and the servers embodying the relevant services that are of particular interest. This section considers the implications of this latter class of problems, though it considers also the implications as one transition to smart objects.

### 3.2 Gateway and Server Architecture to IoT6

#### 3.2.1 Protocol Levels in Gateways and Servers

In [1] we have analysed the 2013 scene with the Internet of Things, and have contrasted it with the scene thirty years ago for the interconnection of networks. In 1980, there was a proliferation of legacy networks with their own protocol suites at all levels. In an early paper on network interconnection [2], the authors showed that there were two main approaches to allow computers on one network to communicate with computers on another. The first was to leave the networks unchanged, and to **adapt** the protocols between them in a gateway; in general this required an Application Level Gateway (ALG), because there were differences between the technologies at all levels. The second approach was to **adopt** the same network technology at some level of one of the networks (or both). In general, if there were N types of network, a complete adaptation would require N x (N-1) types of gateway – though this could be greatly reduced if the concept of half-gateways was used. If one type could be used as the dominant network technology, this would reduce the work in the construction of such gateways. While at that time the international standards community was working on the OSI stack [3] as a standard, Cerf was already advocating the Internet protocols [4] as the standard of choice. If such a standard was adopted, then the number of types of gateway could be reduced to N – though connection of two non-standard topologies would require two such gateways.

If the community could be persuaded to **adopt** similar technologies at some levels, then there would be no need to **adapt** these levels at all. After a heated debate over the next decade, the Internet Protocol (IP4) won the war. It was comprehensively developed in the Internet Engineering Task Force (IETF) [5]. The gateways themselves had much less work to do due to the wholesale adoption of the Internet suite based on IPv4. In fact, the new name was just **routers**, and their adaptation was required only at the network level and below. All activities concerned with reliability, Quality of Service (QoS) and packet ordering was relegated to end-
systems, which had to implement a complete Network Stack, which included any higher-level protocols required. This allowed the routers to work at much higher speed.

As networks and host computers grew in number, it was clear that a database of network addresses was required. To allow these databases to be distributed, one defined the Domain Name System (DNS) [6]. This database was from the outside defined as distributed. Although this data needed to be universally accessible, it could be populated by organisations that were close to the end users and their networks. Moreover, although the routers needed the information from them, the databases themselves did not need to be in gateways, they could be held in servers accessed over the Internet. These servers could be located wherever suitable computing and networking resources were available, and could be replicated for resilience in case of network or server failure.

The situation in 2013 with the Internet of Things (IoT) is much the same as with networks in 1981. Again there are many legacy systems for sensors and applications. Different application domains have defined different ones, and there is little agreement which of them will survive. It is now agreed that wide-area connectivity will be done normally by Internet Protocols, though there is still discussion on whether this should be by IPv4 or IPv6. In fact, there is little doubt that eventually it will be IPv6, but the complete transition may take quite a while. For the time being, there is a broad class of situations where the IoT situation in 2013 is very similar to the network one of 1983.

To illustrate this, we must first define a notation as shown in Figure 3.

![Figure 3: The notation used for gateways and servers](image)

In Figure 3, various communications in the systems are shown: Application, High Level, Transport and Network. By “Network”, is meant both link level and the basic communications technology. The standard definition distinguishes between Servers, which provide network services, and gateways, which connect two networks adapting between the protocols at the different levels. In the normal Internet, services that need to be carried out is Name to address provision and web services, which are normally carried out in Servers. Figure 3 indicates that the physical (and link) levels on a server must be of one type as dictated by the network technology. However, several different applications, high-level transport and network levels can co-exist. Usually a remote entity will communicate with a
server by a particular stack including high-level, transport and network-level protocol. By contrast, in a gateway there will be often different physical and link-level technologies on two half-gateways. This is illustrated in Figure 4, where the Internet is assumed in the ServiceNet, and a specific Buildings Automation System (KNX) [8] is assumed for DevNet.

![Figure 4: Schematic of Networks, Servers and Gateways in the Internet of Things](image)

In Figure 4, a gateway is shown between ServiceNet and DevNet which runs a different stack in each. This gateway must be an Application Level Gateway (ALG), which must adapt all the protocol levels from one stack to the other. The Server shown is running only one stack at the link and physical levels, but different ones above. For example, the server may be dual-stack, running both IPv4 and IPv6 software, so that it can be accessed remotely from systems running either one or the other of these. The figure shows also a server called a Repository. In the normal Internet, a service that is vitally important is the DNS that maps names to network addresses; in the IoT, this functionality needs to be greatly extended to have a Repository that stores much more information. For example, the repository may store all the information about the current configuration of a sensor system, and even its network address. As will be demonstrated later, it may have a much broader and more important functionality in IoT.

In the 1978 paper [2], the main discussion concerned the extent to which different networks adopted similar protocols at different levels; it was clear that ALGs could be quite complex. Eventually, as all the networks adopted Internet protocols, the gateways no longer had to translate the transport or higher-level protocols, so that the need to even run these disappeared. The gateways became merely routers adapting only link-level and physical differences. As IPv6 came to the fore, the routers sometimes ran dual-stack, but had to mediate between the two technologies. With some physical technologies like satellite and lossy wireless, intermediate devices sometimes did much more complex transformations at their edges, but from a network point of view these were transparent. In CALIPSO, they went even further. They put functionality in interior nodes that cached information, and then communicated it autonomously instead of directly from the end sensor node.

### 3.2.2 Server, Gateways and Clusters and Information Distribution

The ALGs in 1978 had to do already so much computation that one sometimes had to use clusters of machines. While these processors used inter-process communications, they had to
be co-located because of the slow speed of networks at the time. By the ‘80s, when network speeds had increased, some gateways started using multi-processor configurations. Recent instantiations have used co-processors to off-load specific functions. Nevertheless, most current gateways between legacy systems and the Internet have carried out their functions in single systems. This has the advantage of making it possible to acquire or develop only one integrated system for each gateway to the Internet technology but this may require quite complex software to achieve full functionality. Since the DevNet gateway is at a critical place i.e., the link to the sensors and actuators, it is often vital that it is very robust and available. Sometimes the gateway may be on hardware of constrained power, and should try to conserve power as much as possible. We already have very functional gateways to legacy systems developed in IoT6 (U. Murcia board in WP2 and WP3 with various legacy interfaces [19], one tailored to home and building automation from WP4 [15], as well as others from WP5) and previously by the Universal Device Gateway project [137] whose results are used by WP4. How much of this functionality can be removed from the gateway itself and located in servers will now be ascertained. For this reason, Task T2.4 is exploring how much of the functionality can be placed in servers. These servers may now be located on the ServiceNet and it is possible that some may even be located in a Cloud somewhere else.

3.3 The services on the ServiceNet

The services on the ServiceNet may be of several sorts; some of these are discussed below:

a. Those concerned with offloading the technology of the DevNet
   An example of these is the translation of an IPv6 address into a specific address of a sensor. Another is translating a CoAP read/write command into one setting and reading from sensors or actuators. Yet another example is translating a multicast Read in CoAP into a sequence of Read commands over the DevNet.

b. Those concerned with providing data storage and a user interface
   Examples are storage servers, which preserve a history of sensor readings, and can make these available over a management interface.

c. Those providing configuration data and the location of processing services
   Just as the DNS stores the association of names with addresses, a repository for IoT might store the following details: what sensors are in a sensor deployment which could associate them with IP addresses or even possibly state something about their current state of health, and even provide a pointer to where the process is located which can process the technology-dependent features. Incidentally, just as the DNS is globally distributed, there are some IoT repositories such as the HANDLE resolver system [9] that have the same property.

d. Those that require substantial resources
   One function that is considered of paramount importance to IoT is to ensure secured operation of devices. This may be a very complex requirement. It may need to ensure different levels of authorisation for different operations, detailed authentication of those attempting to carry out an operation, protection against Denial of Service attacks and similar safeguards. All these may require relatively complex cryptographic operations and access to security repositories. By locating the main security functions in a security proxy, and requiring only secured tokens between the security proxy and the gateway, much more powerful protection can be provided for constrained-resource gateways.
4 IPv6-based features

4.1 Introduction

The number of devices that are connected to the Internet is growing exponentially. This has led to the inception of the ‘Future Internet’, which one can say was partly fueled by the new version of the Internet Protocol, IPv6. The new protocol’s basic remit was to extend the addressing space in order to support all the emerging Internet-enabled devices. It was soon realised, however, that enlarging the address space alone to accommodate the plurality of devices only addressed part of the challenges posed by the Future Internet. IPv6 has since been supplemented with protocol functions that provide a complete protocol suite (as opposed to a monolithic ‘stack’). Beyond enlarged addressing space the suite caters for secure communications to endpoints and mobility for devices and whole networks, to name but a few, thereby enabling nodes to be ‘always connected’. It is this extended set of IPv6 protocol functions that are making it possible to connect numerous objects (things) to the Internet and pave the way towards the IoT.

The objective of the IoT is to integrate and unify all communications systems that surround us. Hence, systems can interact with other systems (access) or even request to modify other systems (control) leading to ubiquitous computer communications by defining a new generation of services.

IoT is enabled by tiny and highly constrained devices, called smart objects. These devices have low-performance properties due to their constraints in terms of memory capacity, computation capability and energy autonomy. In addition, their communication capabilities are only low bandwidth, limiting reachability. These devices enjoy non-continuous connectivity because they use low duty cycles and they are highly power constrained. Their communications networks are described as the Low-power Wireless Personal Area Networks (LoWPANs).

Recently, the IETF working group defined how IPv6 packets could be transported over these LoWPANs (6LoWPAN) to extend Internet to smart devices. 6LoWPAN endeavours to offer to the LoWPANs all the advantages of IP, such as scalability, flexibility, ubiquitous, open, and end-to-end connectivity. One could argue that 6LoWPAN would enable constrained devices to be empowered with all those features and functions of the full IPv6 protocol suite, i.e., a protocol stack for mobility (such as MIPv6), a protocol for management (such as SNMP or equivalent), a suite for security (such as IPsec), etc. However, because of the aforementioned reasons of energy and resource constraints, it is unrealistic to assume that a 6LoWPAN device can be empowered with full-blown host-based mobility, complete network management features, full layer security, etc. The host-based protocol counterparts of such features require most of the signalling and functionality to run on the end-nodes. In some cases, it is possible that even a network infrastructure to support the various features may be required.

e. Those providing fairly uniform application-oriented services.

In any specific application domain, the operations required are often few in number. For example, in a sensor application, CONFIGURE, READ, WRITE, STORE are the main requirements. It is quite feasible to define an Application Server for such purposes, which carries out only technology independent operations and it is the other proxies that bring in the technology and device specifics.
The reduced capabilities of 6LoWPAN networks were not originally considered in the design of the host-based protocols. For example, a 6LoWPAN node with severely reduced packet size and aggressive techniques to conserve energy (by use of sleep schedules with long sleep periods), may not be able to run the full functionality required by an IP host, causing a faulty operation in the IP network. 6LoWPAN nodes may just wake up to receive IPv6 signalling messages; this mode of operation may introduce delays (or timeouts) in the reception of messages which are vital for the operation of a host-based IP network.

In this section, having clarified that the reduced capabilities of the LoWPAN devices cannot accommodate ‘as is’ the stacks of a full IPv6 protocol suite, what will be identified are the various protocol functions and how they have been re-engineered in order to empower the constrained devices with the stacks of Internet-enabled devices. It should be noted that such adaptation might come at the expense of reduced functionality.

### 4.2 IPv6 basic and advanced functions

In order to determine the advanced IPv6 functions that should be used in IoT6, we must determine both what protocol features and functionality might be useful, and list what features might be available in the full host environment and in the constrained environment. The set of IPv6 protocol functions that deal with the suite of the wider addressing aspects as the basic set of IPv6 features is considered. We also consider other functions in the protocol suite that are supported by advanced operations (and possibly require infrastructure support) as the advanced set of IPv6 features. In this section, both of these classes of features are tackled.

We also acknowledge that the above-mentioned sets of operations are not always implemented in a monolithic way (i.e., exclusively and completely inside a single system kernel, or device driver, with limited data structures) and therefore the terms ‘features’, ‘functions’, and ‘operations’ are used interchangeably. For example, the advanced feature of mobility requires the supporting function of a Home Agent. This is a service provided by the relevant infrastructure. It incorporates all these operations that the protocol needs to fulfill its functionality. These operations may differ slightly in their various stack implementations, i.e. a full Home Agent functionality may exist for fully capable mobile hosts, and a tailored reduced functionality may exist in another stack for a LoWPAN node. Nevertheless, both stacks will have to implement the minimum standard set of such operations to ensure interoperability.

The principal functions that are useful have been defined in the IoT-A Project [10]. These are introduced further in subsection 4.3 and have been previously mentioned in Deliverables D1.1 and D1.2. There is only very limited class of facilities that can be provided by IPv6 for the IoT. Those that seem relevant are considered further in subsection 4.5 and their intricacies and technical details are analysed extensively in the subsequent sections of this deliverable.

It is important to bear in mind, however, that there are very few intrinsic features of IPv6 that cannot be done using IPv4. Some may be much more convenient using IPv6. More significant is that many facilities and services have been standardised for IPv6 in the last decade, and not for IPv4. Thus unless IPv6 is employed, it is uncertain whether their compatibility can be assumed between the offerings of different suppliers. This will be further amplified in section 4.

### 4.3 IPv6 basic functions

We mentioned earlier that the set of IPv6 protocol functions that deal with the suite of
addressing aspects as the *basic set of IPv6 features* was considered. A full description of the basic IoT6 addressing features has been provided in detail in Deliverable D1.2. The deliverable has also comprehensively addressed the topic in the context of IoT6 and contributed the ‘IPv6 Addressing Proxy’ service as an IoT feature based on basic IPv6 functions, as published in [18].

The addressing aspects for completeness are further discussed below.

### 4.3.1 Address Features

The first and most common attribute of IPv6 is its address structure. Many aspects of this address structure are particularly valuable to IoT.

#### 4.3.1.1 Large address space and scalability

The best known, and most often quoted, advantage of IPv6 is its address space of 128 bits rather than the 32 of IPv4. While it is not a requirement of the header, it has been agreed in the IETF that only the high-order 64 bits will be used as an address, with the lower 64 bits used to characterise the device. Even a 64-bit address is four billion times as many as is available in IPv4. This would allow each device that can be foreseen even in the IoT of the distant future to be given a unique address. This would obviate the need to use Network Address Translation (NAT) as it is used in today’s IPv4 Internet due to the shortage of addresses. In most of the approaches to the IoT, addressing directly all devices is not required; in that case it is not essential to use the network address for this purpose, since it could carry the device identifier in the payload. Nevertheless, because there is so much standardisation of the address space, it is convenient to use it as a device identifier. Indeed, some deployments have gone much further, and are using parts of the address structure to represent the physical location and/or topology of end devices.

#### 4.3.1.2 IPv6 Multicast

A property of the address shared by IPv4 and IPv6 is that of a multicast address. This allows a packet to be directed to multiple addresses as defined in a multicast group. While this property exists in IPv4, services based on IPv4 have never worked well and are very rarely offered by carriers. By contrast, in IPv6 there has been considerable effort to ensure that there are other parts of the packet header to assist the rugged implementation of multicast. Moreover, many other IPv6 services rely on the availability of multicast as a base capability so that it can be relied on as a tool available to applications and will be enabled in all IPv6-ready routers. Multicast will normally be associated with a multicast group; this may be stored in a repository and cached in routers.

#### 4.3.1.3 IPv6 Link Local and Anycast

Every IPv6 address, except the unspecified address (::), has a “scope”, which specifies in which part of the network it is valid.

Two other useful IPv6 properties, which do not exist in IPv4, are Link Local and Anycast. Link Local restricts the scope of an operation to a particular set of addresses reachable on the same network link; Anycast requests a response from any node within its scope that is live. Both of these can be available to applications, and examples of their use will be given when Use Cases are considered. They are also used heavily in some of the auxiliary services that have been specified by the IETF and are very useful in the IoT.
4.3.2 Auto-Configuration

In IPv6, procedures for auto-configuration of hosts have been defined. These are to ensure that IP nodes are directly self-connected into the local environment with or without prior knowledge of the device. This is further covered in subsection 5.6.1 (Self-configuration) and also forms the basis of some mobility advanced features (Care-of address) as discussed in paragraph 6.5.2.

In legacy systems, some such system also often exists; one may arrange in the nearest gateway to the legacy system that such a device auto-configuration translates the address into an Internet one.

4.4 IPv6 advanced functions

As mentioned earlier, functions in the IPv6 protocol suite that are supported by advanced operations (and possibly require infrastructure support) are considered as the advanced set of IPv6 features.

We introduce below the set of IPv6 advanced features, which are also endorsed by the IoT-A architecture.

4.4.1 Security

Security is of paramount importance in IoT. It has many features, but there are fundamentally three important ones: authentication, integrity and confidentiality. Based on these 3 features, there are many other services like robustness to different attacks, audit trails and authorisation. While these are vital to achieve secure operation, they are beyond the scope of this section.

In IPv4, the IPsec [15] protocol suite has been defined and implemented, however its support is not mandatory. It provides for the three basic services above. In the earlier versions of the IPv6 documents, the IPv6 support was mandatory however, in the latest versions, it is advisory, but not quite mandatory. This is mainly because some major players, e.g. the mobile telephone operators, preferred to use quite different mechanisms based on the SIM card under their control. For IoT6 purposes, security is of paramount importance. For this reason we assume that IPsec is supported. One aspect of IPsec is the need of shared secrets between the two ends of an IPsec communication. There are mechanisms defined like Internet Key Exchange (IKE) [20]. There are also bootstrapping mechanisms defined such as IPsec bootstrapping [21]. These are not mandatory, but may be used in some Use Cases. It is also possible to include the relevant information when a device is being configured; this is a mechanism that is also very useful in IoT.

Based on the basic IPsec features, a number of services have been defined – e.g. SSH, TLS [22] and DTLS [23]. The first two are based on stream protocols. The TCP already ensures reliable transmission, while TLS ensures the security features. The last is designed for datagram security, and is therefore the one required for securing information between one CoAP entity and another HTTP or CoAP one.

There is one problem with DTLS. With stream communications, the stream transport protocol, e.g. TCP, ensures reliable delivery of all packets in the stream. With datagrams, there is no corresponding transport protocol. There is acknowledgement of individual datagrams, but not necessarily group management of multicast groups. For this reason, DTLS cannot guarantee reliable reception of packets by all members of a multicast group. The Standard states that this means DTLS cannot be used with multicast. It would be truer to state
that if a multicast-addressed packet is transmitted, you cannot be sure it has been received by all destinations. However, if it is received, it is still possible to carry out the relevant security functions. There is no greater danger of unauthorised operations than with unicast packets.

4.4.2 Mobility

4.4.2.1 Host mobility in Layer 2
There are a number of mechanisms for dealing with mobility. In general mobile telephone operators have a well-specified mechanism for dealing with mobile devices, which operate on Layer 2. In theory, this allows full access at Layer 3 and above. Therefore, one would expect that all Internet procedures continue unchanged, with the potential service interruption at cell handovers.

4.4.2.2 Host mobility in Layer 3
As part of the support for mobility in Layer 3, the IETF defined a Mobile IP called MIPv4. While this works quite well, it is inefficient. All traffic has to go through a Server which acts as a Home Agent, and then routes any traffic to the device through the relevant Foreign Agent. There has been a lengthy activity in the IETF to remedy the inefficiencies of the mechanisms used. In IPv6, a long time was spent in defining secure re-connection methods (using IPsec) to allow direct transmission from the mobile device to a destination on the Internet. The IETF could have re-defined MIPv4 procedures, but this has not been done.

4.4.2.3 Network mobility
In some applications, one is working not only with single remote devices, but also with mobile networks. An example might be the sensors on an airplane or train. For this there has been a further development called NEMO [60], [61], [62], for which an IPv6 version has been specified.

4.4.2.4 Mobility on the IoT
Layer 3 mobility for end hosts is analysed comprehensively in section 6. The requirements for end-hosts (full and reduced capability), as well as design principles, management of operations for the IoT and the MIPv6 protocol itself are presented in much detail. In addition, a lightweight solution for reduced capability device mobility is described as well as its interactions with IPsec in a secure mobile environment.

4.5 IoT protocols supported by IPv6
The following paragraph refers to protocol standards that have been further engineered with support by IPv6. Similar web service protocols exist based on IPv4, however, their fit to constrained environments is only possible due to recent developments in IPv6 and 6LoWPAN areas.

4.5.1 Web Services
Most of the services used in IoT are based on web services. These rely on HTTP as its Internet protocol, which is in its turn reliant on the stream transport protocol TCP. HTTP allows for proxies to carry out many of its operations.

To deal with constrained resource devices, a much reduced functionality protocol has been defined, called CoAP [44]. This still includes all the basic services required in sensor networks, but does not have the power or flexibility of HTTP and therefore, may use much
less resource. To ensure good functionality of applications, CoAP has been designed to allow easy interfacing with other implementations of CoAP and HTTP. Thus the implementation in the constrained environment may have limited functionality; however, the full functionality is provided by the proxy. One feature of CoAP, included to reduce resource utilisation, is its use of only UDP datagrams in its inter-entity communications. CoAP does work with any IP address structure including multicast.

All these protocols exist both for IPv4 and IPv6 and must be reliable between entities. In stream protocols like TCP, reliable operation can be assured; in datagram operation this is much more difficult to ensure.

### 4.6 IPv6 advanced features in IoT6

In the following Table 1, mappings between existing protocol functions (which we classified as advanced functions with the aid of IPv6) and some lightweight implementations of the same features are presented. The table is provided in order to demonstrate how the various features in the reduced domain are interoperable with, and translatable to the full implementations. This set forms the core of the advanced IPv6-based feature set.

It is obvious how the feature set extends beyond the narrow boundaries of a specific layer in computer networking parlance. In fact, the feature set extends to the architecture and several components in the table below require support from the infrastructure (e.g., the DNS features). This is exactly what a protocol suite is about and therefore, the preferred term for this set of features would be ‘the bundle of advanced features based on IPv6’ for use with the IoT.

<table>
<thead>
<tr>
<th>Protocol</th>
<th>Full version</th>
<th>Lightweight</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>IPv6</td>
<td>IPv6 (RFC 2460)</td>
<td>uIP (Contiki OS)</td>
<td>Internet protocol</td>
</tr>
<tr>
<td></td>
<td></td>
<td>6LoWPAN (RFC 6282)</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>GLoWBAL IPv6 [28]</td>
<td></td>
</tr>
<tr>
<td>Neighbor discovery</td>
<td>ND (RFC 2461)</td>
<td>ND for 6LoWPAN (RFC 6775)</td>
<td>Autoconfiguration</td>
</tr>
<tr>
<td>RESTful</td>
<td>HTTP (RFC 2616)</td>
<td>CoAP [20,33]</td>
<td>Web services</td>
</tr>
<tr>
<td>SNMP</td>
<td>SNMP (RFC 5590)</td>
<td>COMAN [31]</td>
<td>Network management</td>
</tr>
<tr>
<td>DNS</td>
<td>DNS / mDNS [39]</td>
<td>CoAP Service Discovery [27]</td>
<td>Service Discovery</td>
</tr>
<tr>
<td></td>
<td></td>
<td>lmDNS [40]</td>
<td></td>
</tr>
</tbody>
</table>

**Table 1: Lightweight protocols implemented for the IoT**

The table shows lightweight implementations of the IP stack (in network parlance), such as uIP and header compression through the 6LoWPAN adaptation layer protocol [56] in order to achieve Internet connectivity. Web Services are supported through a RESTful architecture also enhanced with a lightweight and ‘compressed’ protocol such as the Constrained Application Protocol (CoAP) [44]. Most recently, the management of constrained networks and devices (COMAN) [55] is also provided as an alternative to the Simple Network Management Protocol (SNMP).

This deliverable also exploits further other advanced features based on IPv6 such as resilience, energy efficient operation, system security, ubiquitous access and mobility.
management (with security support for the IoT). These are further presented in section 5 through section 8 and utilise many of the IPv6 basic and advanced functions as presented earlier. The requirements for some of these advanced features are better understood after the introduction of architectural enhancements discussed in the following section.

5 Ubiquitous Access

5.1 Introduction

This Deliverable combines two quite disparate threads, which are completely complementary, and are both facilitated by ubiquitous access. The first is that of allowing entities to access I/O devices for monitoring and actuation; the second is to distribute the different aspects of the subsystems required to achieve this functionality. Much of the IoT6 project is concerned with the system, including gateways, of accessing and using the IoT devices. Here the IPv6 address structure is particularly important to allow the addressing of the large number of devices. Moreover many of the other IPv6 features, like built-in facilities for multicast, anycast, mobility, security and auto-configuration are very useful. The individual features have been discussed already. How they can be put together into a complete gateway is discussed in this section. The resulting gateway is fairly complex, it needs a significant quantity of code, and often requires a reasonably powerful processor. Once we have built such a gateway, it is important to analyse its components in detail, and to consider whether any of them could be located elsewhere, where they might be replicated, updated and maintained without interfering with the real-time operation of the system. This approach is being pursued here in parallel. The provision of replicable servers, running a multitude of proxies, is a mechanism for the second approach and is also pursued here.

5.2 How IoT6 architectural advances deliver ubiquitous access

Figure 5 presents the communication architecture of IoT6 in a refined view, which demonstrates how IoT6 exploits advanced functionality based on IPv6 protocols to provide ubiquitous access, scalability, resource management, service discovery, and support for heterogeneity. The architecture and use cases also make use of IPv6-based advanced functions such as security, mobility, anycast, multicast and quality of service (smart routing).

The figure shows the snapshot of the IoT6 system architecture after the Vienna IoT6 meeting. Specifically, it demonstrates how all the system components interoperate through a protocol suite based on IPv6. This suite includes 6LoWPAN, GLoWBAL IPv6 or IPv6 Addressing Proxy for addressing services. It then stacks CoAP on top of the Network Layer, followed by JSON and oBIX for the sensors and actuators, all in the Application Layer. In addition, other protocols are defined over JSON/CoAP for the various Information Systems (such as STIS from WP6 and Discovery from WP3) in order to integrate RFID (passive resources), and active resources (sensors and actuators), respectively.

The initial work that offered IPv6 addressing to ‘All Things’ with 6LoWPAN, GLoWBAL IPv6 and IPv6 Addressing Proxy services, has been extended with the additional protocols and features of IPv6. Specifically, techniques for energy saving and quality of service have been offered based on the use of anycast/multicast and flow label, as well as security support over IPv6 (with a lightweight integration of IPsec) and mobility support (with a lightweight MIPv6 solution).
5.3 Specific Feature Distribution between Server Repository and Gateways

In Deliverable D2.1, an example of a gateway between a KNX Sensor system and the Internet was given. It was a comparatively simple example, but it had many components in the gateway. These included the following:

1. A WiFi Driver at the link level
2. An IPv6 stack at the network level
3. A TCP implementation at the transport level
4. Some simple configuration information on what sensors there were and their nature (Application Layer)
5. An address proxy to translate between IPv6 addresses and those from the sensor system (Application Layer)
6. An HTTP Server module (Application Layer)
7. A processing module that translated between HTTP Read/Write and those of the sensor system (Application Layer)
8. A whole set of software to control the legacy system (this software is considered as the Application Layer)

In the example, there were no multicast operations, and no security. It would have been possible to simplify this gateway hugely by removing functionality out of the server and
leaving just the following:

9. A WiFi Driver at the link level
10. An IP stack at the network level
11. A UDP implementation at the transport level
12. A CoAP Server module (including DTLS security below it)
13. A whole set of software to control the legacy system

The implementation of 9 - 13 would have been much simpler for many reasons. The only parts of the gateway which might not have been provided by the legacy system supplier are 9 - 12. However, these days, the supplier might even have provided also 9 - 11. It is possible that the supplier would have provided a TCP module instead of UDP with the main advantage of UDP being that it requires less processing and memory than TCP. Moreover, the CoAP module would also provide full security via DTLS. There would, of course, be another significant advantage. The gateway described here contains no software that depends on the KNX features, and would be identical, as regards its Internet portion, to any other legacy system.

Of course, there is no such thing as a “free lunch”. The simplification in the gateway comes at the cost of activity in a server cloud – most of which would be located on ServiceNet in Figure 2, but some might even be in a cloud elsewhere. The following services would be required:

1. A Configuration Data Set, which might be held in a Repository. It would contain information on the current sensor configuration, the mapping of IPv6 addresses to sensor virtual addresses and the process which would input the technology components of the legacy system. The information could be stored in a Repository, with full data security. It could verify the authorisation of the agent entering or modifying the configuration information, as well as that of the agent attempting to access it.

The dataset could be stored securely if the appropriate server was used. It might have a simple form of DTLS for access from the processing module below and the gateway using CoAP. The access from remote configuration or accessing agents might be via HTTPS, and be much more stringent, to ensure that the appropriate authorisation is provided. There is no reason why the gateway access could not be via a NAT gateway and IPv4, if this is more convenient in that case. This would automatically avoid any unauthorised access to the gateway from outside the ServiceNet.

Of course we would recommend using IPv6 even in this portion, but if the controller is provided by the supplier with only an IPv4 stack, it might be considered unnecessary and wasteful of resource to add an IPv4-IPv6 conversion in the gateway.

2. There would need to be a technology processing module that would depend only on the type of legacy system, it need not contain any configuration data; that can be provided from the configuration data in the Repository. This processing might include a translation from a multicast group operation to a set of unicast ones, if the legacy system does not support multicast. It might also include translating an IPv6 address and a virtual sensor address into a legacy system in the syntax of the legacy system. It would map between the CoAP primitives of READ/ WRITE, etc. into the syntax of commands understood by the legacy system.

Doing any multicast to multiple unicast operations in this server is both possible and may be optimum; however it comes at the cost of there being multiple CoAP transactions between the server and the gateway. Each transaction would need to be
secured individually, since DTLS is not recommended for multicast operations. Thus it may be desirable in particular cases to carry out the multicast to multiple unicast operations in the gateway. This solution would modify the otherwise simple and stateless operation of the gateway.

3. For mobility support, there may be a need for a Corresponding Agent that is capable of carrying out the binding updates required by MIPv6. If mobile sensors or sensor networks are to be supported, then the option of using IPv4 should certainly not be used for the gateway.

### 5.4 Scenarios of User and Server Access to Gateways

There are three important requirements in IoT:

1. The system should be resilient
2. Access and Operation should be secure
3. Gateway operations should be as energy-efficient as possible.

A key advantage of the Server/Repository/Gateways approach described above is that it gains over the normal gateway on all three counts. We will analyse each of the above criteria in turn.

#### 5.4.1 Resilience

The gateway itself is so much simpler because it is probably much more error-free and - since it is almost unchanged from that provided by the supplier - most of the problems will have been resolved by the supplier with the legacy system itself. Many repositories have been developed for general purposes, and have been shown to be rugged over a substantial period of time. Moreover, in many repository designs, it is possible to have replicated and distributed systems. An example of this is the HANDLE resolver system [9]. Here the data can be provided locally, but the access (for name to resource resolution) is through the whole Internet.

In this respect, one could also employ network-based mechanisms based on IPv6 to increase the resilience of the HANDLE service overall. We anticipate that a HANDLE cluster arrangement can be better load-balanced by the use of anycast addressing mechanisms in the Local Link addressing of a HANDLE domain.

#### 5.4.2 Security

It is very difficult to provide fully secure systems when there is wide access to them. In the approach outlined, the approach is through the repository. Certainly for systems like HANDLE, a very comprehensive security infrastructure has been provided. It would be difficult to provide anything comparable in a resource-limited device.

#### 5.4.3 Energy-efficient Operation

It has been shown that operations in server clusters are much more energy efficient in any case than those in isolated systems. In the IoT environment, it is actually the gateways and sensors where the resources are most critical. In our approach, the more intense operations are performed in Server Clusters; the operations in the gateways themselves are deliberately as light as possible - using protocols optimised for this environment. Thus with wireless sensors, CoAP, 6LoWPAN and RPL may well be used. All of these have been defined with resource constraints in mind.
We further analyse the wider term “Energy Efficiency” and its implications for the IoT in a separate subsection below. However, it is not just overall energy efficiency that is important, it is particularly that the gateway, which may itself be mobile and battery operated, makes minimal power demands. The distributed approach to gateway functionality helps to achieve this goal. Finally, there is a growing interest in moving as much processing as possible into clouds. The reason for this is some of the translation and configuration processes can be located in distant clouds and make better use of power in such configurations.

5.5 Energy efficiency

In the context of the IoT ecosystem it is often heard that the various protocol suites have been engineered with energy considerations in mind. This is of course true for the resource-constrained part of the IoT network, i.e. the SensorNet or DeviceNet, as presented in Figure 1 and Figure 2 of our enhanced system architecture based on Gateways and Servers. In particular, adaptation layers such as the 6LoWPAN sub-layer ensure that the potentially inefficient use of the full IPv6 stack with its overheads (long address structures and the variety of plain uncompressed header features) are utilised in a controlled fashion that fits the energy envelopes of small devices.

In this section, the term “Energy Efficiency” is revisited and its notion is analysed in the IoT architecture.

5.5.1 ‘Energy efficiency’ notion in the 3-tier architecture

As mentioned earlier, it has not been the design objective of IPv6 from the outset to reduce energy at the core scale of the network. However, as the Internet economies prevailed, energy-conscious initiatives and policies have been reflected in network protocol designs and infrastructure implementations.

In the 3-tier architecture of an Internet as shown in Figure 1, it is noted that the future network itself extends at the fringes by many more devices, gateways and servers, to what we call the Internet of Things. The additional hardware this growth entails can only lead, undoubtedly, to increased power consumption. However, this is not the notion of “Energy Efficiency” to which this project aspires. We are conscious of the increased power consumption requirements of an enlarged Internet. We are also conscious of the limited energy capabilities of the Internet of Things (as the fringe of the Future Internet). It is the improvements one can make at the intersection of the two domains (the core and the fringe) where we believe the term energy efficiency highlights the advantages of using IPv6 protocol suites over IPv4 and even over legacy and proprietary protocols.

5.5.2 ‘Energy efficiency’ notion by use of IPv6 on the fringes

In the low-power/low-energy communication domain of the IoT, the IEEE 802.15.4 standard is one designed to fulfill the low power PHY/MAC protocol requirements of small devices. However, such nodes still consume around 20mA of energy when the radio chip is active.

The following graphs of Figure 6 show a typical embedded sensor node (older Berkeley open design of a Telos B mote) utilising Contiki’s ‘powertrace’ capability to profile energy

---

1 See /apps/powertrace/powertrace.h for the exported function API and examples in /contiki/examples/powertrace.c
consumption for the receive and transmit cycles only. It is observed that the transmit cycle varies around an average of 8mA, while the receive cycle is not doing any better, yet with a smoother consumption envelope, as expected. The Peaks of the receive-transmit cycle amount to almost 20mA. This is not negligible.

Figure 6: Radio power consumption of Transmit and Receive cycles using Contiki power tracing and profiling.

To further validate the above measurements, the ‘Energest’ feature of the RIME stack in Contiki is utilised. With ‘Energest’, the above measurements have been refined to the granularity of each resource (embedded sensor, CPU, radio cycle, LED). UCL’s open mote design marketed as Orisen Prime [16] was used (hardware details available on request) and the same stock Contiki-2.6 as before.

2 See /core/sys/energest.h for the exported functions and implementation in /core/sys/energest.c
First, we observe how closely the consumption envelope of receive and transmit radio cycles are captured compared to the previous method in Figure 6 (5.5mA and 10.5mA respectively). Secondly, we now have a comparative view of how much the various resources consume and what percentage of this is attributed to the network functions. We estimate that while the listening functions (receive) are comparable to the sensor and LED profiles, the transmission is at double, albeit at a shorter duration. This is for the transmission of a single byte payload using full IPv6 headers (non-6LoWPAN, see discussion further below).

In addition, as was discussed in an earlier section, IP typically does not allow non-responsive nodes without leading to communications failures. As a result, and considering the graph observations above, one can only expect a reduced battery lifetime of an embedded small device.

The standardisation efforts within IETF have led to the development of the 6LoWPAN adaptation layer that enables IPv6 to reach the fringes of the Future Internet. Typically, this implies that the overhead of using the full IPv6 headers (40 bytes) can be significantly reduced down to 1-7 bytes per IPv6 packet. This is a significant development, if one considers that multiple such transmissions are required to transfer a message from a 6LoWPAN-enabled node to the Internet and vice versa.

This aspect of energy consumption is formally captured in the following simple formula:

\[ P_m = P_p + P_h \]

Where \( P_m \) denotes the overall power consumption for the successful transfer of one message to its destination, \( P_p \) is the power consumption attributed to the message payload only and \( P_h \) is the equivalent quantity for the headers only. It is assumed that the message fits in a single IP packet.

The above observation implies that efficient network adaptation and routing layers would be those that attempt to keep the \( P_h \) term as low as possible. If no IP layer is assumed then the \( P_h \) can be considered null and thus one obtains close to optimal efficiency by employing an
energy efficient MAC transmission mechanisms at Layer 2. However, this is outside the scope of this discussion, and a plurality of such application custom MAC protocols exists in the literature.

In the Internet world, $P_h$ is compressed to about 1-7 bytes per message, as calculated by the various validations of the 6LoWPAN standard. It is however noted, that this header compression mechanism has become optional.

On the other hand, efficient routing protocols have been introduced at Layer 3 for small devices. Notably, RPL among others attempts to reduce the $P_m$, by keeping the total number of messages low. This is achieved with a proactive mechanism that maintains a Directed Acyclic Graph (DAG), which maps to the nodes of the constrained network. As each node knows the root of the DAG at each time, it can adapt its routing directly to that node, instead of resorting to repetitive and energy-expensive route discovery methods dynamically per packet.

In addition to RPL, section 7 reviews and analyses various routing constructs that have been proposed for constrained networks. While it is not within the scope of this deliverable to fully analyse the ‘Energy Efficiency’ properties of each such protocol in turn, one can roughly estimate the energy implications by considering the amount of messages used to pass a message end-to-end. Subsection 7.3 then sets criteria that must be considered if one wants to satisfy energy efficient requirements of the IoT. The proposed solution of “Scalable, Efficient and RE liable” (SERE) routing adheres to these criteria while maintaining a low factor of $P_m$.

5.5.3 ‘Energy efficiency’ notion by utilising generic half-gateway abstractions

While the previous paragraph discussed in detail the concept of ‘Energy Efficiency’ in the fringes of the Future Internet, little has been said about the interconnection point. Such islands of small devices require gateway devices to link into the larger network. In section 3 and Figure 1 and Figure 2, the terms DevNet for the ‘Things’ in the fringes and introduced the ServiceNet to denote the immediately larger network these islands attach to have been used. The interconnecting Controller Gateway, made of two half-gateways (as comprehensively presented in [19]), poses another challenge in defining ‘Energy Efficiency’ in this domain.

Section 3 stipulated that many of the functions of the Controller Gateway can be abstracted and summarised by the use of CoAP. At best, this approach will allow resource-poor network islands of DevNets interconnect to ServiceNets by putting some of the core functionality on processes that run on servers either in the ServiceNet, if critical, or further out into the Cloud, if not.

This architectural enhancement reflects an opportunity for energy savings on the DevNets. Any function that can be abstracted, for example, encrypted communication with one node in the DevNet, does not necessarily have to involve a fully featured encryption operation on the node in the DevNet. One can supply a token-based secure communication method that gets initialised once and operates via a token-verification process that runs on a server in the ServiceNet. This approach has the potential to dramatically reduce both the $P_h$ and $P_m$ factors of subsection 5.5.2, thus leading to significant energy savings on the DevNet. These gains could potentially be even higher with the use of Cloud services and virtualisation of the abstracted services that run on the Controller Gateway. In its simplest form then, the energy profile of the Controller Gateway is close to that of a transparent CoAP proxy.
5.6 **IPv6 based self-configuration, self-management, self-healing.**

The terms self-configuration, self-management and self-healing have been used in the DoW document for the Task 2.1. They have not been adequately clarified in the previous deliverable D2.1 when only a basic set of IPv6 mechanisms was reviewed for use with the IoT. In the following paragraphs, we present for completeness a follow-on discussion on the above terms and how we believe their use can be mapped to IoT functions, as expressed by the IoT6 systems architecture view of Figure 1 and the enhanced view of Figure 2.

### 5.6.1 Self-configuration

This function maps clearly to the stateless address auto-configuration features of the basic IPv6 stack (SLAAC) as described by IETF standards [24]. In multiple occasions in the following sections, mainly when discussing advanced features based on IPv6 functions, it was realised that end hosts are required to support multiple IPv6 addresses per network interface; in addition to this requirement introduced by the new protocol suite of IPv6, every host is required to configure a Link Local address even when global addresses are available. IPv6 hosts may also self-configure multiple global addresses upon receipt of one or more Router Advertisement messages, thus eliminating the need for infrastructure support with a DHCPv6 server.

IPv6 hosts may randomly generate the host part of a SLAAC-configured address. Hosts then generally combine a prefix of up to 64 bits with a 64-bit EUI derived from the manufacturer-allocated (and assigned) 48-bit IEEE MAC address. The MAC address ensures that this portion is globally unique, a property inherited by the EUI-64. The host is normally required to ensure (through a method of broadcast queries) that the generated (fully concatenated) IPv6 address is not in use by any other host on the local network.

While this mechanism of self-configuration (also referenced in the literature under the wider term of ‘zero-configuration’) can be fully implemented on a capable host (end-system computer), it is unlikely that the resource-constrained equivalent for IPv6 stacks running on IP-based Sensor Networks will be as comprehensive.

### 5.6.2 Self-management

The main benefits of the self-management technology in systems and networks are to minimise operator involvement and operating expenditure (OPEX) in the deployment, provisioning and maintenance of the network, and increasing network reliability. This is of course a set of desired features for network operators. The recent FP7 project EFIPSANS [11] introduced an architecture reference model for self-managed networks (dubbed ‘autonomic networks’), termed GANA (Generic Autonomic Network Architecture), which defines the fundamental building blocks that should be considered when designing devices for an autonomic network. The project has instantiated the architecture by proposing and incorporating a set of IPv6 functions into a more elaborate set of operations that fulfil the requirements of autonomic networking. It is noted that the architecture mainly addresses full capability network devices that one commonly encounters in the core of the network.

However, the core principles of self-managing networks, as perceived by EFIPSANS, have matching functions in resource-poor networks of devices. In particular, most node-related functions rely on the set of auto-configuration options built around IPv6. This includes both stateful (DHCPv6) and stateless auto-configuration (SLAAC), as described in the previous paragraph. The set of stateless functions is adequately covered in IoT6 by the Neighbour Discovery mechanisms analysed in section 4, and it has been employed in the advanced IPv6 features (e.g. node mobility). The stateful functions need to be supported by the infrastructure...
(such as DHCPv6); but, this is not within the primary scope of the DevNets themselves. Their reduced IP stacks are already capable of supporting these mechanisms in the adopted Contiki framework.

One aspect of autonomic networks not directly addressed in the IoT6 architecture (being network- and service-layer focused) is the auto-configuration of radio channels in the MAC layer and below. Nevertheless, suitable mechanisms exist in the adopted platform of Contiki. Such mechanisms have not been required by any Use Case.

IoT6 has, however, addressed self-management at Layer 3 and above, by analysing the use of IPv6 extension headers. WP3 and WP5 further propose smart routing features for route management and reliability in their respective deliverables.

5.6.3 Self-healing

As already stated in the DoW, we consider ‘self-healing’ an inherent property of wireless sensor networks. This is often mainly attributed to the particular way these networks operate, where they remain in ‘sleep’ mode most of the time, while they resort to some activity (i.e., transmission, LED indication, etc.) for limited parts of their duty cycles in order to save energy. As a result, their MAC layer properties follow this paradigm of intermittent operation. In addition, ad-hoc and multi-hop routing mechanisms provided either at Layer 2 (‘route-under’, i.e. below network layer, in 6LoWPAN parlance), or routing most recently at Layer 3 (‘route-over’, i.e. RPL), ensure that the IP operation end-to-end does not lead to faults due to the intermittent nature of the message exchange. We consider this combination of properties at link and routing layers (as better supplemented by IPv6) as one of the self-healing properties of the sensor network.

An extensive analysis of various routing methods for non-legacy sensor networks has been given in subsections 7.1 and 7.2. The various healing properties of those protocols, where relevant, are clearly stated. The solution proposed in subsection 7.3 for the IoT6 architecture capitalises on such properties, by providing a mechanism for path recovery and by employing two schemes for reliable route discovery (based on ICMPv6). In IoT6 vision, these self-healing mechanisms could be either managed by the end-node itself, through the functionality of the IP stack in a DevNet, or by the half-gateway in the case of legacy systems.

It is recognised that mechanisms native and internal to a legacy network may exist, which supplement the self-healing functionality of the fringes of the Future Internet. Future work may be able to address mappings between such self-healing mechanisms and those of the IP Internet. In fact, the system view of the IoT6 architecture as enhanced by the work described in section 3 may lead to completely transparent self-healing mechanisms over the controller gateways.

6 Mobility in IPv6 sensors

6.1 Mobility protocol trends

Mobility is one of the major issues of the Future Internet and is solved in different ways. The solutions are mainly split into two trends: on one hand, a trend based on an evolutionary research following the IPv6-based approach and current Internet architecture; and on the other hand, a clean-slate, where a new architectures that requires major changes in the existing protocols and networking philosophy.

The clean-slate trend is based on new concepts such as identifier and locator split architectures such as the one presented in [42]. This kind of architectures presents the
advantage that mobility is directly supported by the separation of the session identification with the locator of the device, which is the problem of the current Internet architecture. Previous work on the IoT has been focused on this approach; the main issue is that the overhead for 6LoWPAN devices is increased due to the need to transport one additional header from the identification layer. These types of solutions are very relevant from the research point of view, but are considered inconvenient and are not feasible in the short term, since the current hardware and infrastructure deployed are not ready for this kind of approaches.

For that reason, this work is focused on the evolutionary research approach. This follows the current Internet architecture for the management of the identification and location, i.e. IPv6 continues being used for Identification of the session in the transport and application layers, and Locator of the devices for routing in the network layer. These solutions allow the continuing use of the existing infrastructure and the overcoming of the problem using a similar concept to the identifier/locator split but in an implicit way. Specifically, the main protocol following the evolutionary approach is Mobile IPv6 (MIPv6). MIPv6 uses two IPv6 addresses, on one hand, the initial address of the device, commonly denominated Home Address that is used as identifier, and the new address in the visited network, commonly called care-of address, which is used as locator.

MIPv6 protocol provides the signalling messages and IPv6 headers extensions to manage the binding between these two addresses. In addition, this defines the security mechanisms and networking requirements in order to avoid the identity supplantation and man-in-the-middle attacks. Specifically, this defines return routability mechanism to carry out route optimisation in order to avoid triangle routing and IPsec tunnelling between the mobile node and the home agent. This is to ensure the security and authentication of the mobile node for the binding updates when the node needs to register a new care-of address.

The main functions of MIPv6 are covered by the home agent, which is the identity in charge of managing the identifier, caching packets when the mobile node is in transit, and demonstrating the authenticity of the mobile node when the mobile node claim its identity from a visited network.

In previous works, the feasibility of Mobile IPv6 for constrained devices such as the considered ones for the IoT [38] has been evaluated. These works concluded that MIPv6 presents a high overhead for the data packets when the mobile node is in roaming, since this needs to include the destination option to specify its home address in case of route optimisation applied or build an IPv6 tunnel which requires an additional IPv6 header. Both cases require a high overhead.

The second problem with Mobile IPv6 is that IPsec is mandatory in order to protect the communications between the mobile node and the home agent. As mentioned previously, the trust relationship between the mobile node and the home agent is a fundamental requirement of MIPv6. This is because all the security of the binding update for the mapping between the care-of address and home address, and additional security processes such as the return routability for the route optimisation are based on this trust relationship.

Therefore, the lightweight implementation of MIPv6 for IoT is not as simple as to carry out header compression and reduction of the size for the signalling control messages as previously defined for other protocols such as IPv6 with 6LoWPAN and HTTP with CoAP. Important changes are required to solve the challenges from the emerging infrastructure deployed for the IoT. Specifically, this change of infrastructure for IoT is the previously mentioned introduction of highly constrained devices in the Internet in the level where usually nodes with high capabilities such as laptops, PCs and servers were deployed.
Therefore, the evolution is from a current infrastructure of end hosts with high capabilities, and gateways/routers with also high capabilities, to an infrastructure where the end nodes are a large number of constrained devices, and border routers with high capabilities. In addition, these emerging Internet-enabled devices require additional self-* properties in order to support the required scalability and autonomy to support dynamic environments.

The next sections present the design issues for the lightweight version of MIPv6 and the proposed solution to offer a secure and efficient mobility management for the IoT.

### 6.2 Design Issues

The following items present the requirements for the design of a mobility management protocol that satisfies the requirements from dynamic ecosystems, which represent the constraints from the devices used to build the IoT ecosystems. Each scenario presents different requirements and challenges. But all of them present the common goal of reaching a seamless handover ensuring the security and a suitable efficiency.

These design issues have been defined considering the requirements from emerging scenarios such as smart cities, Hospital Wireless Sensor Network (HWSNs) and health monitoring in critical environments from previous works [28][77].

- **Global identifier**: End devices need to be reachable globally by any other entity connected to the Internet. Thereby, end-to-end connectivity can be offered, which is a foundation of IPv6, Future Internet and IoT [77][59].

- **IPv6-based protocol**: Mobility management for IoT needs to be built over Internet protocols, such as Mobile IPv6 (MIPv6) even when they are not implementing all the functions of the host-based IPv6 protocol. Thereby, it offers an evolving approach for IoT that can be integrated with the existing Internet-based software and infrastructure.

- **Lightweight protocol**: Mobility management protocol needs to consider similar lightweight regards and implementation guidelines that have been already taking into account for 6LoWPAN and CoAP. Thereby, mobility management can be integrated into constrained devices with low memory capabilities [43]. The requirements from MIPv6 are higher than 6LoWPAN and CoAP, since this is presenting additional security requirements and an overhead for all the data packets during the roaming.

- **Communication cost**: Mobility headers and related signalling must be optimised to reduce the impact in the power consumption and overhead ratio. Specifically, signalling messages should fit within a single frame to avoid fragmentation [77]. In addition, broadcast and multicast usage should be reduced since smart objects have a low duty cycle and consequently the reachability and impact in power consumption of these kinds of communications create additional challenges. The overhead impact needs to be reduced mainly for the data communication which needs to include an additional IPv6 header (tunnelled packets through the home agent) or the destination option (when route optimisation is applied and the communication can be directly established with the correspondent node). The other binding-related messages are not presenting a major challenge, since they are requiring less than a frame and the piggyback payload packets on the binding-related messages are not applied.
• **Packet encapsulation**: Packet encapsulation used by Mobile IPv6 (when tunnelling packets between a mobile node and its home agent) reduces the frame size left for data and thus may generate fragmentation. For that reason, new challenges arise for enabling mobility management in this kind of devices to reduce the overhead from Mobile IPv6 for data packets.

• **Security**: Node authentication and authorisation must be supported so as to offer security capability, ensure protection of the resources, integrity and confidentiality of the information. Many security challenges exist in dynamic IoT ecosystems, mainly due to the resource constraints of mobile users, the authentication delay constraint, and the demanding security requirements of applications when the nodes are in roaming, i.e. visiting foreign networks.

• **Movement detection**: Mobile IPv6 relies on neighbour discovery for movement detection and care-of address creation. This movement detection based on neighbour discovery is very slow since it depends on the router advertisement frequency and it is not effective in wireless networks when different channels are used for different LoWPANS. Solutions for movement detection can be optimised for specific use cases, for example in HWSNs where continuous monitoring generates continuous and periodical traffic. This traffic can be used for the RSSI evolution analysis and consequently avoid the usage of extra signalling messages. Examples of this kind of movement detection techniques have been presented in [28].

### 6.3 Mobility management for the Internet of Things

The proposed mobility management protocol needs to present a high efficiency in terms of low computation complexity and communication cost, but at the same time it needs to be compatible with the existing IPv6 infrastructure and offer a suitable security level. Figure 8 presents the main metrics for the mobility management protocol for IoT and its relation with the solutions proposed and evaluated in this work. The integration and interoperability with the existing infrastructure is one of the main requirements for mobility management in dynamic ecosystems, since mobile nodes require the capability to use other networks during the roaming, i.e. during the time out of the home network. For that reason, it is important to offer a solution highly compatible with the available access points and routers.
The security is a high requirement for mobility, since this offers the capability to redirect traffic to a new address (the care-of address) and claim the identity of a node. Therefore, both features open a high number of vulnerabilities for man-in-the-middle attacks, identity supplantation, and data integrity. In order to avoid these vulnerabilities, it requires the authentication of the mobile node such as it is carried out in Mobile IPv6 with the trust relationship between the mobile node and its home agent.

Efficiency is always a desirable feature but in IoT, it presents a higher relevance, since this marks the difference between solutions suitable for the constrained capabilities of the devices relevant to IoT solutions.

An optimal solution satisfying the three described metrics cannot be defined for IoT environments with the existing solutions and protocols, due to the unsuitability of the full Mobile IPv6 protocol and the lack of standardisation for new proposals. For that reason, this work analyses the three approaches and discusses each one in relationship to the function of the scenario requirements.

The following subsections present the approaches evaluated in this work. First, Mobile IPv6 is analysed since this offers a high integration potential. However the main problem with this approach is that it presents requirements which make it unsuitable for constrained devices. For that reason, a lightweight version of Mobile IPv6 is suggested, which presents a higher efficiency for IoT environments and maintains the interoperability with the original Mobile IPv6 protocol. Finally, lightweight Mobility support with security support (i.e. IPsec), following the requirements and design considerations of Mobile IPv6 is analysed.

### 6.4 Mobile IPv6

Mobile IPv6 offers an extension header for the IPv6 protocol to support the binding management. Figure 9 presents the integration of the mobility header as an option for the IPv6 header.

#### 6.4.1 Mobile IPv6 requirements

The main problem of Mobile IPv6 over 6LoWPAN is the overhead due to the mobility options, Home Agent (HA) address specification in all the data packets, and the tunnelling costs for the communications through the HA, and on the other hand, the viability of the required security for the communication between the Mobile Node (MN) and its HA is a problem.
Mobile IPv6 requires a set of mandatory capabilities for the MN, which is not feasible for the constrained devices used in the IoT. Specifically the requirements of Mobile IPv6 for MNs are as follows:

1. The MN must be able to process Mobility Headers.
2. The MN must maintain a Binding Update List.
3. The MN must be able to send Binding Updates, and receive Binding Acknowledgment and Binding Refresh Request.
4. The MN must support receiving Mobile Prefix Advertisements (Router Advertisements) and re-configuring its home address based on the prefix information contained therein.
5. The MN must support movement detection and care-of address formation.
6. The MN must support the Destination Option header to include the Home Address in the Binding Updates.
7. The MN must perform IPv6 encapsulation and decapsulation for the communications based on triangle routing through the HA.
8. The MN must support IPsec, since the communication between the MN and the HA needs to be protected.
9. The MN must support the return routability procedure.
10. The MN must be able to process type 2 routing header in order to receive packets directly from the Correspondent Node (CN) and include the Home Address option to send the packets directly to the CN, both options when the Route Optimisation procedure is carried out.

All the presented requirements are mandatory following the RFC 6275 [47], therefore, they need to be supported in order to be interoperable and fully compliant with the existing Mobile IPv6 implementations.

### 6.4.2 Mobile IPv6 analysis

The initial seven requirements are based on the basic functionality of Mobile IPv6 in order to perform the binding management, set-up of the care-of address, and exchange of data with the correspondent node when the MN is in roaming mode. Specifically, the *destination* option is required to indicate the home address during the binding management when the source address is the care-of address instead of the home address, and the encapsulation/decapsulation is required to continue using the home address as source address during the roaming.

According to the last three requirements, even when they are mandatory, a Mobile IPv6 scenario can be established without really requiring them. These requirements are related with security aspects of the following: IPsec for the communication between the MN and the HA; and the return routability procedure used to exchange binding key, ensure the reachability of the CN from the MN and authenticate the MN for the route optimisation, in order to avoid the triangle routing through the HA.

The Mobile IPv6 communications with the correspondent nodes can be carried out in two different ways: with suboptimal traffic flow in the case that the CN does not support MIPv6 and directly with the CN in the case that the route optimisation process can be performed.

The route optimisation and consequently return routability process are only carried out when the CN supports Mobile IPv6. For that reason, a mobility scenario can be established without
these two functions and ignore them when the MN supports Mobile IPv6, since it should be required to carry out triangle routing.

Therefore, the IPsec requirement presents the main concerns about the feasibility of Mobile IPv6 over 6LoWPAN, since IPsec was not initially considered suitable for constrained devices, due to its high overhead and processing requirements [41]. Therefore, this work analyses the feasibility to integrate IPsec and its limitations.

Figure 10: Sequence of phases in Lightweight Mobile IPv6

For that reason, this work proposes a lightweight version of Mobile IPv6 without security support, when the initial seven requirements are satisfied and when it is compatible and functional with the existing Mobile IPv6 solutions. On the other hand, IPsec is analysed and integrated in the lightweight version of Mobile IPv6, thereby it offers a solution with security support satisfying the requirement number eight.

The next two subsections present both the proposed lightweight Mobile IPv6 and the IPsec support in the lightweight Mobile IPv6.

6.5 Lightweight Mobile IPv6

Lightweight Mobile IPv6 is lightweight because it does not support route optimisation, return routability, and IPsec. In addition, its implementation has been optimised to be integrated into constrained devices with a low capacity in terms of memory and communication capabilities.

Figure 10 presents the sequence of phases in Lightweight Mobile IPv6, the main difference with respect to full Mobile IPv6 is that Return Routability Procedure and Correspondent Binding Procedure are not carried out; consequently Route Optimisation is not supported. The phases are described as follows.

6.5.1 Movement detection

Movement detection in Mobile IPv6 is based on Neighbour Discovery. Neighbour Discovery has been redefined for 6LoWPAN in the RFC6775 [49]. The revision of Neighbour Discovery for 6LoWPAN presents a serious inconvenience for the Mobile IPv6, since the router
Advertisements are only sent upon reception of router solicitation, therefore, the movement detection cannot be based on the router advertisement frequency or appearance of router advertisements with a different prefix.

The revised Neighbour Discovery is also removing the duplicated address detection procedure but the MN continues requiring the registration of its new care-of address on the router.

![Figure 11: Care-of address registration](image)

This registration only consists of exchanging neighbour solicitation and advertisement messages between the MN and the router. Therefore, several solutions have been proposed for 6LoWPAN for movement detection based on cross-layer movement detection through values from the link layer such as RSSI and additional signalling packets such as keep-alives [32].

Signalling for the movement detection should be based on passive overhearing of messages from other protocols instead of active keep-alives. The reason is that the number of messages arising from the mobility protocol needs to be reduced as much as possible in order to ensure the low-power features desirable in the IoT. The passive overhearing analysis to trigger the movement detection is dependent on each solution. For the evaluation, an active scan to associate to the new network and then the use of neighbour discovery, as defined in the standard, is considered first. An approach based on direction determination and RSSI evolution has been presented in our previous works for critical environments and Hospital Wireless Sensor Networks [29][30].

### 6.5.2 Care-of-address configuration and binding management

Care-of-address configuration is based on stateless auto-configuration. Stateless auto-configuration is one of the advantages of neighbour discovery through the router advertisement, where the prefix of the new network is announced.

Care-of-address registration uses the mobility headers defined by the MIPv6 protocol. Specifically, care-of-address registration is carried out through the Binding Update message and its respective Binding Acknowledgment, such as presented in the Figure 11.
6.5.2.1 Binding Update (BU)

A BU message is used when the MN is in its home network to indicate to the HA that it needs to take care of its home address; it is the called HA Registration process. Secondly, the BU is used by the MN to notify the HA of a new care-of address assigned in the visited network for itself. Thereby, the HA is able to map and forward the messages from the active communications.

The main difference between the registration and update is in some bits defined over the mobility header. For that purpose, the Mobile IPv6 header for BU will be analysed in more details. The BU is composed of two IPv6 header extensions such as presented in the Figure 12(a).

First, the Destination Options for IPv6 header is used to indicate its Home Address. Thereby, the Home Agent can link the new care-of address with its home address. Then the mobility header is sent, in this case the mobility header 5 is the binding update. The fields as described are as follows:

- **Sequence Number**: The sequence number is used to avoid duplicate packets and match the binding update with the binding acknowledgment.
- **A**: The bit A is set to request to an acknowledgement from the Home Agent.
- **H**: The bit H is set to indicate to the received node that it should act as its Home Agent, i.e. it is presenting the home agent registration process. For that reason, when this packet is sent, the MN needs to be in the same network (included subnet prefix) as the Home Agent.
- **L**: The bit L is set to indicate that the home address is the same to its link local address; it is mainly for the Home Agent registration process.
- **K**: The bit K is set to indicate that the dynamic key management for IPsec is supported. Otherwise, if IPsec is established static or non-security is defined for the communications with the Home Agent then it needs to be cleared.
- **Lifetime**: The lifetime of the binding update indicates when the update from the MN can be considered expired.
- **Options**: Some mobility options such as alternative care-of address can be added.

For the Lightweight Mobile IPv6, the Binding Update is equivalent, but with the usage of the 6LoWPAN and the extension header format proposed by RFC6282.

Figure 12(b) is presenting the BU for the 6LoWPAN version available in the Contiki implementation of 6LoWPAN, which is neither supporting the Next Header compression for the destination option nor for the mobility header. For that reason, this carries the original Next Header value of the destination option in-line (i.e. Next Header with value 60), and this also requires to carry the hop limit field, since this values is equal to 128 and the header compression to elide this value is only considering 1, 64 and 255 values.
Figure 12: a) Binding Update Message in Mobile IPv6. b) Binding Update Message in Mobile IPv6 with 6LoWPAN header with the original Contiki OS implementation

Figure 13 presents the proposed version of the BU with the compression mechanism for the IPv6 Next Header (NHC) of which this work has been considered as a novelty. It is the first implementation of the 6LoWPAN NHC for the destination option and the Mobile IPv6 headers. The NHC for the IPv6 Destination Options Header has the value reserved in the RFC6282 [56], 1110011N, where N indicates if one additional header is presented with the NHC format, in this case since the IPv6 Mobility Header is also added. The value of N is 1, and NHC=0xE7. The fields included for the compressed version of the IPv6 Destination Options Header are:

- **Header Length**: This field indicates the length of this header in the same way as in the original IPv6 Destination Options Header. This is required since several options can be included, and consequently the length is variable.
• **Option Type**: This field indicates the option type in the same way as the original IPv6 Destination Options Header. For example, the value 201 means that the Home Address Options is required to indicate to the Home Agent to which MN this BU belongs to.

• **Option Length**: This field indicates the length of the option, since each option has a different value.

• The NHC for the IPv6 Mobility Header has the value reserved in the RFC6282 [56], 1110100N, in this case N is 0, and NHC=0xE8.

The fields included for the compressed version of the IPv6 Mobility Header are:

• **Header Length**: This field indicates the length of this header in the same way as in the original IPv6 Mobility Header. This is required since several mobility header types are defined.

• **Mobility Header Type (MH)**: This field indicates the MH type. This has been reduced from 8 to 4 bits, since the considered MH Types are under 16, and this allows re-using the reserved 4 bits after the flags.

• **Flags (A, H, L, and K)**: The flags keep the same semantic as the original one.

• **Sequence Number**: This field has the same semantic as in the original mobility header.

• **Lifetime**: This field has the same semantic as in the original mobility header.

• **Mobility options**: This offers in the same way as the original Mobile IPv6 header, the option to add mobility options such as the alternative care-of address option.

The compressed version of the BU presented in the Figure 13 is converted to the version presented in the Figure 12(a), when this goes through the 6LoWPAN Border Router. Thereby, it is totally interoperable with Mobile IPv6.

![Figure 13: Binding Update Message proposed for the Lightweight Mobile IPv6 with 6LoWPAN header and next header compressed for the destination option and the mobility headers](image)

### 6.5.2.2 Binding Acknowledgment (BA)

The BA is presented in the Figure 14(a). The BA is very similar to the BU, but the difference
is the inclusion of the Routing Header type 2 to hold the Home Address of the MN and the field Status to indicate if the binding update has been accepted, or if some problems have been raised.

In the same way as the Binding Update, the Figure 14(b) presents the version of the Binding Acknowledgment with the usage of 6LoWPAN, which is offered by the current Contiki OS implementation. Since the extension header format proposed by the RFC6282 should be used, Figure 15 presents the version proposed with the implementation of the IPv6 Routing Header Type 2 using the NHC.

![Figure 14: a) Binding Acknowledgement Message in Mobile IPv6. b) Binding Acknowledgement Message in Mobile IPv6 with 6LoWPAN header with the original Contiki OS implementation](image)

Mobile IPv6 with a 6LoWPAN header with the original Contiki OS implementation is presented. The NHC for the IPv6 Routing Header has the value reserved in the RFC6282 [56], 1110001N, where N is equal to 1, since the IPv6 Mobility Header is added. The value is
NHC=0xE4. The fields included for the compressed version of the IPv6 Routing Header are:

- **Header Length**: This field indicates the length of this header in the same way as in the original IPv6 Routing Header.
- **Routing Type**: This field indicates the type of the routing header, in this case the routing header type 2 is used, which is reserved for the mobility purpose. The other options of mobility are being deprecated because of security issues. Therefore, this field could be elided in the future.
- **Segment left**: The field indicates the number of hosts that this message still has to visit before reaching its final destination. This value is equal to 1 in the case of mobility purpose, since this is only used to identify the MN Home Address, in the same way as the destination option identified to the MN Home Address in the BU message. Therefore, since it will be always equal to 1, this field can be elided in the future.

The NHC for the IPv6 Mobility Header for the BA changes with respect to the BU. Specifically, this has changed the A, H and L flags to the status field. This change has been applied for the Lightweight version presented in the Figure 15.

![Diagram](image)

*Figure 15: Binding Acknowledgment Message proposed for the Lightweight Mobile IPv6 with 6LoWPAN header and next header compressed for the routing and the mobility headers*

### 6.5.3 Data communication

The traffic generated for the data communication is tunnelled via the HA was presented in Figure 14. The triangular routing through the HA is required when the CN is not supporting Mobile IPv6 and in particular for Lightweight Mobile IPv6 since the support of Route Optimisation has not been considered.
The encapsulation is from the MN to the HA and vice versa. For that reason, two versions of the same packet are defined. First, the “**” version of the packet is presented for the version encapsulated and the “*” version for the decapsulated version. The Figure 17(a) presents the format of a CoAP packet sent from the MN to the CN via the HA. This includes the outer 6LoWPAN header with the source address of the current address of the MN (care-of address) and destination address set to the HA address. The inner 6LoWPAN headers with the source address the home address of the MN and destination address of the CN address. Finally, this includes the useful data with the CoAP packet or the transport/application used.

This uses the NHC defined for IPv6 packets in the RFC6282, where such as presented, the NHC value for the tunneled IPv6 packet uses the reserved value 1110111N. N is 1 since the UDP next header is required. Therefore, NHC=0xEF. This packet based on 6LoWPAN is translated to the IPv6 version presented in the Figure 17(b) after its decompression by the 6LoWPAN Border Router.

This compression of both the outer and inner headers from IPv6 to 6LoWPAN has been one of the optimisations carried out by the Lightweight Mobile IPv6, since the original implementation as found in the Contiki OS just compresses the outer header and carries out the inner header with the full IPv6 header.

CoAP has been optimised for the integration of the REST architecture in constrained networks. It requires just 4 bytes to specify version (Ver), Type to indicate if this message is of Confirmable (CON), Non-Confirmable (NON), Acknowledgment (ACK) or Reset (RST), Token Length in case some option is added, Code similar to HTTP in terms of GET, PUT, POST and DELETE, and finally the message ID for the detection of message duplication, and to match messages of type ACK/RST to messages of type CON/NON. More details about CoAP can be found in [44]. The packet described in Figure 17(b) is decapsulated by the HA and only then is it sent as if the MN was in its home address. This packet is presented in Figure 18(a), where the outer IPv6 header is removed.
Figure 17: a) Packet from the MN to the CN via the HA (**1) based on 6LoWPAN header compression. b) Packet from the MN to the CN via the HA (**1) based on IPv6

In the same way, the traffic generated by the CN is sent to the MN via the HA. Figure 18(b) presents the packet transmitted from the CN to the MN, which is not encapsulated. This packet arrives to the HA, which is encapsulated as presented in the Figure 19(a), and finally the 6LoWPAN Border Router adapts the outer and inner header to 6LoWPAN instead of IPv6, see Figure 19(b). Therefore, the MN always needs to encapsulate and decapsulate all the packets, which means an extra overhead due to the encapsulated IPv6 header. The presented tunnelling is assuming IPv6 headers and it is not taking into account security. Figure 20 presents the different options to implement the tunnel between the MN and the HA.
Figure 18: a) Packet from the MN to the CN via the HA (*1). b) Packet from the CN to the MN via the HA (*2)
Mainly, two modes are defined: encapsulation without security, such as has been presented; and encapsulation with IPsec ESP. The Lightweight Mobile IPv6 implementation can consider IPv6 header compression instead of full IPv6 such as 6LoWPAN. For that reason, the presented figures are based on the 6LoWPAN header in all the versions. The next subsection presents the case where there is an IPsec tunnel between the MN and HA.

Figure 19: a) Packet from the CN to the MN via the HA (**2). b) Packet from the CN to the MN via the HA (**2) based on 6LoWPAN header compression
6.6 **IPsec support in Lightweight Mobile IPv6**

6.6.1 **IPsec Analysis**

IPsec is mandatory with IPv6, making it available in the majority of operating systems and networking hardware. IPsec is not one protocol but rather three: Authentication Header (AH) and Encapsulating Security Payload (ESP) are used for traffic security and Internet Key Exchange (IKE) is used for the establishment of keying material and other traffic security parameters.

AH and ESP are usually supported by the kernel as part of the IP stack, while IKE is implemented as a user daemon.

![Figure 20: Encapsulation modes for data communication during roaming]

IPsec offers end-to-end security in the network layer. It was expected to be a suitable security protocol for datagram traffic generated by client-server applications, but at the end it has not been exploited as much as expected since it is not very suitable for Web-based client-server application models. This needs to be managed in the kernel. For that reason, protocols such as DTLS have gained attention for their application in the datagram environment. In particular for the IoT, DTLS 1.2 [23] has been considered as the security protocol for the constrained application protocol.

IPsec is used mainly to build tunnels between hosts such as the one used for MIPv6. The traffic exchanged between the MN and its’ HA is IPsec protected (tunnel mode). In all cases, its signalling traffic is protected using transport mode (ESP). IPsec has to be used for traffic through the Home Agent tunnel; this solves most of the security challenges introduced by mobility. MIPv6 introduces no new security threats. The main problem of IPsec is the overhead and extra memory requirements for its development.

The integration of IPsec over constrained devices can be carried out through the usage of specific crypto-suites such as AES-CCM that are directly supported by hardware in the majority of the transceivers used in IoT solutions such as IEEE 802.15.4 and 6LoWPAN. Thereby, the impact coming from the cryptographic primitives is avoided. However, it can use more common and supported crypto-suites such as AES-CBC, which offer encryption and can be used in conjunction with other authentication mechanisms when that authentication can be also provided. Both of them continue to present an inconvenient overhead for the communications. The next subsection presents the new formats including the IPsec ESP
6.6.2 IPsec integration in Mobile IPv6

Mobile IPv6 considers IPsec as the security protocol to protect the binding management, the communications between the MN and the HA, and the communications with the CN sent via the HA.

Binding management requires the use of IPsec ESP in transport mode to provide data origin authentication, connectionless integrity, and optional anti-replay protection.

Regarding the data communication with the CN via the HA, IPsec ESP is used in tunnel mode to protect the inner IPv6 address. Thereby, intruders between the MN and the HA cannot determine what is the real destination address of the packet.

One inconvenient aspect of IPsec for Mobile IPv6 is that Route Optimisation is not compatible with IPsec Tunnel payload (i.e. ESP), since the Security Association (SA) with all the CNs is not established. The SA is established mainly between the MN and its HA. For that reason, the security is carried out through Return Routability procedure when Route Optimisation is applied. In the case of lightweight Mobile IPv6, Route Optimisation with the CN is not used; consequently the IPsec protection is not lost on traffic leaving/entering the foreign network.

6.6.3 Crypto-suites support for IPsec in lightweight Mobile IPv6

The crypto-suites considered for the IPsec integration with Mobile IPv6 can be based on different approaches. IPsec defines as mandatory AES-CBC for encryption and HMACSHA1-96 for authentication [58], although AES-CCM support has been defined in RFC6275 [47]. AES-CCM can be considered highly relevant for IoT, since the majority of the transceivers support these functions by hardware. For example, the IEEE 802.15.4 standard includes support for AES-CTR for encryption, AES-CBC-MAC for message authentication and AES-CCM which combines encryption and message authentication. This offers blocks of 32, 64 or 128 bits. Some implementations also support 96 bits. The mandatory mode by the standard is AES-CCM. For that reason, this work is focused on AES-CCM.

CCM is an authenticate-and-encrypt block cipher mode for the Advanced Encryption Standard (AES) block cipher available in the majority of the hardware used to build IoT devices such as IEEE 802.15.4 and IEEE 802.11 transceivers. AES-CCM, with an explicit Initialisation Vector (IV) is used as an IPsec Encapsulating Security Payload (ESP) mechanism, such as required to build the IPsec tunnel between the MN and the HA. AES-CCM provides data integrity and data origin authentication for the payload and for additional information included in the Additional Authentication Data (AAD) section of the ESP payload. Thereby, the ESP payload is composed by the IV, encrypted payload and the authentication data.

AES-CCM requires a different IV for each encryption in order to avoid vulnerabilities. IV needs to be generated by the encryptor and to be transferred to the decryptor. IV collision can lead to the acquiring of plain-text information from both packets. For that reason, the dynamic change of keys is suggested through the Internet Key Exchange (IKE) offered by IPsec or other solutions such as the temporal key integrity protocol (TKIP) used in IEEE 802.11.

6.6.4 IPsec Format

Figure 21 presents the total headers integrated in a Lightweight Mobile IPv6 packet. This is assumed that the outer IPv6 header is 6LoWPAN, but IPv6 could also be considered. Such as
Presented in the Figure 21, the overhead of ESP in tunnel mode depends on the crypto algorithm used. The overhead introduced by AES-CCM mode is at least 18 bytes, 16 bytes from ESP Header (SPI, Sequence number and IV), 2 bytes from ESP trailer (Pad length and next header) when the block alignment is perfect and consequently padding is not required. The use of AES-CBC introduces at least 26 bytes, 24 bytes from ESP Header (SPI, Sequence number and IV) and 2 bytes from ESP Trailer (Pad length and next header) following the previously mentioned conditions.

The difference resides on the IV vector, since AES-CBC [45] introduces the full IV vector length (16 bytes) and AES-CCM [46] the half-length of the IV vector (8 bytes). To construct the sequence of counter blocks used by counter mode to generate the key stream, AES-CCM uses 1 byte for CCM flags, 4 bytes for the block counter, and the remaining bytes are composed of 3 bytes of salt assigned at the beginning of the security association and 8 bytes transmitted in the packet.

This overhead introduced in the packet only provides confidentiality. The ESP Authentication Header could optionally be introduced in order to provide integrity. AES-CCM algorithm is prepared to provide integrity contrary to AES-CBC that needs the use of an additional algorithm such as HMAC-SHA1.

However, these fields could be compressed such as described in [48] when ESP Authentication Data is not included. The main motivation is that ESP Authentication Data considers the ESP header for its calculation; therefore in case of the use of a compressed version of the ESP header, this ICV field will be corrupt in the 6LoWPAN network. Otherwise, when it is not used, a compressed version of the ESP header could be considered, which is uncompressed in the border router, in the same way as 6LoWPAN is uncompressed to the IPv6 header. The ESP fields meaning are described below:

- **Security Parameter Index (SPI)**: This identifies the security association (SA) used in IPsec. This requires 4 bytes. This can be elided in case that the SA for a host (IP address) is well-known by the border router, e.g. it is known by the border router by a previous usage. Other option is to reduce it to 4 bits in order to support 16 different SPIs assuming that the IoT devices will not keep multiple SAs according to its constrained capabilities. Thereby, the border router does not keep a status; or it just extends from 4 bits to 32 bits by adding zeros.

- **Sequence Number**: The sequence number is used to prevent replay attacks. This can be determined by the border router or reduced to a lower number of bits. The main problem of the reduction of bits is the coherence with the home agent, which can set sequence numbers of until 32 bits. For that reason, the options for this field use 32 bits. The border router is tracking and storing the initial 28 bits of the sequence number.

- **Initialisation Vector (IV)**: These 8 bytes need to be included, since the initialisation vector is used for the decoding of the packet, and this is changed for each transmission in order to avoid the vulnerabilities of block ciphering when the IV is re-used.

- **Padding Length**: This field indicates the number of bytes added in the ESP trailer to align the payload block to the multiple of the AES-CCM, in our case blocks of 16 bytes (128 bits).

- **Next header**: This field indicates the next IPv6 header option.

The overhead with IPsec ESP in tunnel mode is excessive since this does not allow for the compression of the inner IPv6 header, as the HA does not need to be aware of the 6LoWPAN header compression. Therefore, even when the ESP header is compressed, it continues carrying 40 bytes of the inner IPv6 header. The section 8 presents the evaluation of the
overhead of IPsec for its different configurations. But, it can be seen in advance that for the presented configuration with the outer header based on 6LoWPAN, the final available payload is 20 bytes out of the original 127 Bytes, which corresponds to 16% efficiency.

Therefore, since the main overhead of Mobile IPv6 is coming by the tunnel, i.e. the inner IPv6 header, two options can be considered. The first one is the non-usage of security and consequently the header compression such as 6LoWPAN or GLoWBAL IPv6 for both the outer and inner headers. The second one corresponds to the definition of an alternative mobility protocol where the header compressed is optimised and security supported through other mechanisms. The next section presents mobile GLoWBAL IPv6, which is an alternative mobility mechanism that is not compatible with Mobile IPv6 but this offers a very lightweight mechanism with a low overhead and suitable security level [52].

**Figure 21: IPsec packet format**

### 6.6.5 Binding management with IPsec

The format of the IPsec-protected Binding Update (BU) message sent by the MN to the HA from a foreign network to register its new CoA, is presented in the Figure 22(a). The BU message is sent using the current CoA of MN (address in source address field of the IPv6 header). The HoA is found in the Home Address Option in the Destination Option Header extensions following the IPv6 header. The BU message (Mobility Header type 5) is IPsec-protected. It contains an AltCoA option which provides the current CoA of the MN. Unlike the version found in the IPv6 header, this one undergoes IPsec protection and cannot be tampered with en route.

After reception and processing of the BU, the HA replies with an IPsec protected Binding Acknowledgment (BA) message. The format of the message is provided below on Figure 22(b). The Routing Header Type 2 contains the final destination of the packet, i.e. the HoA. The destination address of the packet (the one in the IPv6 header) is the MN’s CoA.
### IPv6-based advanced features

#### IPv6 Header

<table>
<thead>
<tr>
<th>Field</th>
<th>Bits</th>
</tr>
</thead>
<tbody>
<tr>
<td>Ver</td>
<td>6</td>
</tr>
<tr>
<td>Traffic Class</td>
<td>16</td>
</tr>
<tr>
<td>Flow Label</td>
<td>24</td>
</tr>
<tr>
<td>Payload Length</td>
<td>32</td>
</tr>
<tr>
<td>Next Header</td>
<td>60</td>
</tr>
<tr>
<td>Hop Limit</td>
<td></td>
</tr>
</tbody>
</table>

#### Source Address
(Care-of Address of Mobile Node 128bits)

#### Destination Address
(Home Agent Address 128bits)

- **Next Header**: 135
- **Header Length**: 8
- **Type**: 1
- **Option Length**: 2
- **Option Data**: 0

Home Address (128bits)

#### ESP Header (AES-CBC)

- **Payload Proto**: 59
- **Header Length**: 8
- **MH Type**: 5
- **Reserved**: 16

#### Mobility Header

- **Checksum**: 8
- **Reserved**: 8
- **Sequence #**: 16

- **MobOpt. Type-1 [PadN]**
- **Option Length**: 0
- **MobOpt. Type-3 [CoA]**
- **Option Length**: 16

#### Alternate Care of Address
(Care-of Address of Mobile Node 128bits)

#### ESP Trailer

- **Padding**: 14
- **Next Header**: 60
53

Figure 22: a) Binding Update Message in Mobile IPv6 protected with ESP. b) Binding Acknowledgement Message in Mobile IPv6 protected with ESP

Tables 2 and 3 make a comparative about the size, security overhead and need of fragmentation regarding the Binding Update and Binding Acknowledgement packets on the several versions mentioned before in this document. This information reflects that the overhead caused by the use of IPsec on Binding Updates and Binding ACK represents between 23.3% and 28.6% of these signalling packets; in addition, the use of IPsec requires fragmentation to send the signalling packets because packet sizes are bigger than 127 bytes, the maximum frame size for 802.15.4.

### 6.6.6 Data communication with IPsec

IPsec security using ESP tunnel mode was used in order to avoid attacks on the mobility. The use of security includes 2 new headers in the packet, the ESP Header and the ESP Trailer which were discussed previously. Following the data communication example previously presented in Figure 16, Figure 23(a) and Figure 24(a) present the version "**" of the packet using ESP tunnel mode when they are sent from the MN to the CN via HA and vice versa. Figure 23(b) and Figure 24(b) presents the version "***" of the packet using ESP tunnel mode when they are sent from the MN to the CN via HA and vice versa using 6LoWPAN Header Compression. These figures presents an ESP tunnel mode encrypted using AES-CBC algorithm as has been developed to carry out the analysis presented in the next section. Decapsulated version (**"**) is the same as presented on Figure 18(a) and Figure 18(b).
Table 2: Binding Update overhead analysis considered along this document

<table>
<thead>
<tr>
<th>Packet</th>
<th>Headers (Bytes)</th>
<th>Total Bytes</th>
<th>IPsec Overhead</th>
<th>Fragmentation Required?</th>
</tr>
</thead>
<tbody>
<tr>
<td>Contiki Figure 5(a)</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Contiki Figure 5(b)</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Binding Update</td>
<td>LL (25) + 6LoWPAN (35) + Dest. Opt. HC (20) + Mob Header HC: BU (26) + ESP Trailer</td>
<td>106</td>
<td>N/A</td>
<td>NO</td>
</tr>
<tr>
<td>Contiki UMU Figure 6</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Contiki ESP Figure 15(a)</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Contiki ESP No Image</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Binding Update</td>
<td>LL (25) + 6LoWPAN (35) + Dest. Opt. HC (20) + ESP Header HC (18) + Mob Header: BU (32) + ESP Trailer (16)</td>
<td>146</td>
<td>23.3%</td>
<td>YES</td>
</tr>
<tr>
<td>Contiki ESP UMU No Image</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Table 3: Binding Acknowledgement overhead analysis considered along this document

<table>
<thead>
<tr>
<th>Packet</th>
<th>Headers (Bytes)</th>
<th>Total Bytes</th>
<th>IPsec Overhead</th>
<th>Fragmentation Required?</th>
</tr>
</thead>
<tbody>
<tr>
<td>Binding ACK</td>
<td>LL (25) + IPv6 (40) + Routing Header (24) + Mob Header: BA (16)</td>
<td>105</td>
<td>N/A</td>
<td>NO</td>
</tr>
<tr>
<td>Figure 7(a)</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Binding ACK</td>
<td>LL (25) + 6LoWPAN (35) + Routing Header (24) + Mob Header: BA (16)</td>
<td>100</td>
<td>N/A</td>
<td>NO</td>
</tr>
<tr>
<td>Contiki Figure 7(b)</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Binding ACK</td>
<td>LL (25) + 6LoWPAN (35) + Routing Header HC (20) + Mob Header HC: BA (10)</td>
<td>79</td>
<td>N/A</td>
<td>NO</td>
</tr>
<tr>
<td>Contiki UMU Figure 8</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Binding ACK</td>
<td>LL (25) + IPv6 (40) + Routing Header (24) + ESP Header (24) + Mob Header: BA (16) + ESP Trailer (16)</td>
<td>145</td>
<td>27.6%</td>
<td>YES</td>
</tr>
<tr>
<td>ESP Figure 15(b)</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Binding ACK</td>
<td>LL (25) + 6LoWPAN (35) + Routing Header (24) + ESP Header (24) + Mob Header: BA (16) + ESP Trailer (16)</td>
<td>140</td>
<td>28.6%</td>
<td>YES</td>
</tr>
<tr>
<td>Contiki ESP No Image</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Binding ACK</td>
<td>LL (25) + 6LoWPAN (35) + Routing Header HC (20) + ESP Header HC (18) + Mob Header: BA (16) + ESP Trailer (16)</td>
<td>130</td>
<td>26.1%</td>
<td>YES</td>
</tr>
<tr>
<td>Contiki ESP UMU No Image</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

These example packets contain UDP-CoAP communications. Table 4 shows several versions of UDP-CoAP packets with and without security on 802.15.4 or 802.3 with information related to the available payload, IPsec overhead when it is used and the need of fragmentation 802.15.4 to be sent.
6.7 Evaluation

6.7.1 Evaluation testbed

The evaluation testbed is presented in Figure 25. The testbed is composed of a HA, a MN, a CN and two 6LoWPAN Border Routers announcing two different networks.

The HA implementation is based on Mobile IPv6 in order to be compatible with the IPv6-enabled backbone. Specifically, it is based on the UMIP implementation of Mobile IPv6 (http://www.umip.org/). UMIP is an open-source Mobile IPv6 stack for the GNU/Linux Operating System.
Figure 23: a) Packet from the MN to the CN via the HA (**1) based on IPv6 using ESP tunnel. b) Packet from the MN to the CN via the HA (**1) based on 6LoWPAN header compression using ESP tunnel
Figure 24: a) Packet from the CN to the MN via the HA (**2) based on IPv6 using ESP tunnel. b) Packet from the CN to the MN via the HA (**2) based on actual Contiki 6LoWPAN
The HA has integrated a 6LoWPAN Ethernet bridge presented in Figure 25, which builds a virtual network interface (tun/tap) for the 6LoWPAN network. The 6LoWPAN Border Router for the foreign network (visited network) has been developed with a Cisco Router enabled with OpenWRT (www.openwrt.org). OpenWRT has been extended to support the previously mentioned 6LoWPAN Ethernet bridge in order to extend it with 6LoWPAN connectivity.

<table>
<thead>
<tr>
<th>Packet</th>
<th>Headers (Bytes)</th>
<th>Total Bytes</th>
<th>Payload Size</th>
<th>IPsec Overhead</th>
<th>Fragmentation Required?</th>
</tr>
</thead>
<tbody>
<tr>
<td>*1 and *2</td>
<td>LL (25): IPv6 (40) + UDP (8) + CoAP (4)</td>
<td>77</td>
<td>50</td>
<td>N/A</td>
<td>NO</td>
</tr>
<tr>
<td>10(a)</td>
<td>LL (25) + 6LoWPAN (35) + 6LoWPAN (35) + UDP (8) + CoAP (4)</td>
<td>107</td>
<td>20</td>
<td>N/A</td>
<td>NO</td>
</tr>
<tr>
<td>10(b)</td>
<td>LL (25) + IPv6 (40) + IPv6 (40) + UDP (8) + CoAP (4)</td>
<td>117</td>
<td>19</td>
<td>N/A</td>
<td>NO</td>
</tr>
<tr>
<td>16(a)</td>
<td>LL (25) + IPv6 (40) + ESP Header (24) + IPv6 (40) + UDP (8) + CoAP (4) + ESP trailer (16)</td>
<td>157</td>
<td>25.5%</td>
<td>YES</td>
<td></td>
</tr>
<tr>
<td>16(b)</td>
<td>LL (25) + 6LoWPAN (35) + ESP Header (24) + IPv6 (40) + UDP (8) + CoAP (4) + ESP trailer (16)</td>
<td>152</td>
<td>26.3%</td>
<td>YES</td>
<td></td>
</tr>
<tr>
<td>18(b)</td>
<td>LL (25) + 6LoWPAN (35) + ESP Header (18) + IPv6 (40) + UDP (8) + CoAP (4) + ESP trailer (16)</td>
<td>146</td>
<td>23.3%</td>
<td>YES</td>
<td></td>
</tr>
</tbody>
</table>

Table 4: UDP CoAP communication overhead analysis considered along this document

Figure 25: Testbed for mobility evaluation
The Mobile Node is a sensor board powered with batteries in order to make it mobile. Finally, the Correspondent Node is a common Linux Machine.

6.7.2 Memory footprint

The current Lightweight Mobile IPv6 implementation has been developed under Contiki OS 2.4 version, to compete with the Jennic port on the JN5148 chipset. Movement detection is based on a simple energy scan on all frequencies over 2 seconds and this could be improved with some technique within the next versions. Signalling packets and tunnel mode works in both modes, with and without ESP security. Code size of this implementation is about 7.4K Bytes with AES-CCM, something less without security and 15.5K Bytes with AES-CBC (software) mode.

6.7.3 Overhead

The encapsulated packet for the data communication during the roaming has a size of 92 bytes plus the payload of the CoAP Packet such as presented in Figure 24. This packet is encapsulated into a MAC frame of IEEE 802.15.4, which introduces 25 bytes of the IEEE 802.15.4 header.

Therefore, the total size is 117 bytes out of the 127 bytes of the MAC frame size. This means that when the CoAP payload is over 10 bytes, fragmentation is required.

For the version based on the 6LoWPAN headers, the reduction is 10 bytes, i.e. the size is 82 bytes plus the payload of the CoAP Packet as presented in the Figure 24. The reduction is very low since the global addressing requires carrying out in-line the source and destination address for both IPv6 headers; in addition, since the ports considered for the UDP header are not in the range of the compressed ones, this requires also carrying them in-line. Consequently, the reduction is very limited.

For that reason, the future work is going to be focused on the analysis of the feasibility of new techniques such as GLoWBAL IPv6 for the header compression of IPv6 and UDP, instead of 6LoWPAN.

Table 5 compares the overhead introduced by IPsec security in both encryption modes (AES-CCM and AES-CBC). The use of secure communications imposes the fragmentation at link layer in 802.15.4 for all the secure communications.
Table 5: Overhead analysis when using ESP Security ciphered with AES-CBC or AES-CCM mode

<table>
<thead>
<tr>
<th>Packet</th>
<th>Headers (Bytes)</th>
<th>Total Bytes</th>
<th>Payload Size</th>
<th>IPsec Overhead</th>
<th>Fragmentation Required?</th>
</tr>
</thead>
<tbody>
<tr>
<td>IPv6 ESP (AES-CBC)</td>
<td>LL (25) + IPv6 (40) + ESP Header (24) + IPv6 (40) + ESP Trailer (16)</td>
<td>145</td>
<td></td>
<td>27.6%</td>
<td>YES</td>
</tr>
<tr>
<td>Contiki ESP (AES-CBC)</td>
<td>LL (25) + 6LoWPAN (35) + ESP Header (24) + IPv6 (40) + ESP Trailer (16)</td>
<td>140</td>
<td></td>
<td>28.5%</td>
<td>YES</td>
</tr>
<tr>
<td>Contiki UM ESP (AES-CBC)</td>
<td>LL (25) + 6LoWPAN (35) + ESP Header HC (18) + IPv6 (40) + ESP Trailer (16)</td>
<td>134</td>
<td></td>
<td>25.2%</td>
<td>YES</td>
</tr>
<tr>
<td>IPv6 ESP (AES-CCM)</td>
<td>LL (25) + IPv6 (40) + ESP Header (16) + IPv6 (40) + ESP Trailer (16)</td>
<td>137</td>
<td></td>
<td>23.3%</td>
<td>YES</td>
</tr>
<tr>
<td>Contiki ESP (AES-CCM)</td>
<td>LL (25) + 6LoWPAN (35) + ESP Header (16) + IPv6 (40) + ESP Trailer (16)</td>
<td>132</td>
<td></td>
<td>24.2%</td>
<td>YES</td>
</tr>
<tr>
<td>Contiki UM ESP (AES-CCM)</td>
<td>LL (25) + 6LoWPAN (35) + ESP Header HC (10) + IPv6 (40) + ESP Trailer (16)</td>
<td>126</td>
<td>1</td>
<td>20.6%</td>
<td>YES</td>
</tr>
</tbody>
</table>

6.7.4 Movement detection

Movement detection has been described in the subsection 6.1. The ideal movement detection is based on passive overhearing but this solution is not realistic since in real environments different channels will be used between the home network and the visited ones.

For example, smart cities will use an extended range of channels in order to avoid interference with other 6LoWPAN networks and existing WiFi networks working at 2.4 GHz.

In the same way, HWSN will distribute the channels in order to avoid collisions, for that reason it will be more motivated that two consecutive networks use different channels.

This work analyses the scan time that is required for movement detection, since a periodic scan is scheduled with energy sensing to discover new networks and measure the current link quality [53].

The total scan time depends on the number of channels to be scanned and the time spent for listening to a specific channel. The scan time per channel is an important value, which needs to be synchronised with the beacon frequency in order to detect the networks.

Figure 26 presents the time spent with the different scan times for scanning all the channels. For example, Contiki OS brings the scan time to 4 by default, which means almost 4 seconds. For the mobility purpose the scan is not only used the first time when joining the network, it is used also periodically. In our case, this value should be reduced for the lightweight Mobile IPv6, in which a scan time of 3 has been used, which is a time of around 2 seconds as presented in the Figure 27 for the 50 evaluation tests carried out.
Figure 26: Time for the different scan times considering all the channels

The scan during the connection time brings the problems, on the one hand, that the node is unreachable during the 2 seconds when the node is scanning, and on the other hand, the power consumption of active listening during the 2 seconds for each scan.

Figure 27: Scan time with energy sensing to detect if the MN is moving and discover new networks

6.7.5 Handover latency

The handover latency depends on multiple phases. First, movement detection is carried out through the active scan, since the other techniques are not feasible when multiple channels are used in an ecosystem. Second, the MN requires associating the new network in the link layer, i.e. link to the IEEE 802.15.4 network. Third, the MN requires the set-up of the new CoA after that the Router Advertisement (RA) is received from the visited network, and finally, the MN requires the sending of the BU to the Home Agent and waits for the BA.

Figure 28 presents the association to the link layer time and the network layer configuration for the first connection of the MN to its Home Network. These values are presented as a reference. The home association spends over 4 seconds since this is based on a scan time with value equal to 4. The networking configuration is very random from 78 milliseconds because the RA has just received 78 milliseconds after joining, to 3,658 seconds because the RA was sent 300 milliseconds before joining to the network.
Regarding the time during the mobility, the scan time is presented in the Figure 27 with a time around 2,083 seconds. After time spent to find a foreign network, it takes an average of 511ms to associate at link layer with the new Border Router (see Figure 29(a)).

Once the link layer has been established, it is necessary to wait in order to receive the RA in so that the network layer configuration can be obtained. This time usually is around 100ms since after the link layer association, a Router Solicitation is sent to improve the time it takes to receive the network layer configuration. The time to receive the Router Advertisement is from 36 milliseconds with some random high picks of 3,151 seconds that set the average time to receive the Router Advertisement on 473 milliseconds. Figure 29(b) presents the time for receiving the RA after joining to the visited network.

Finally, the MN needs to send the BU and wait for the BA. Figure 29(c) presents the time spent for the Round Trip Time spend to send the BU and receive the BA; it is around 1 second with a minimum of 1,028 seconds and a maximum of 1,338 seconds.

Therefore, the handover time without considering the scan phase is presented in the Figure 31, since the scan phase has been part of the movement detection. This presents a minimum time of 1,592 seconds, and a maximum of 4,700 seconds with an average of 2,037 seconds.

This result is very close to the handover times from other works such as the found in the soft handover [39][40], which presents a value of 2.1047 seconds.

This result shows that some values state as the 13,7 seconds mentioned in the work found in [34] for its comparison of Mobile IPv6 with respect to their MOBINET proposal can be reduced following the design issues and the optimisation carried out with Lightweight Mobile IPv6. Figure 31 presents the value also considering the scan time. The handover latency depends on very issues related with the implementation, platform or operating system, since all these issues are ignored in simulators [35]. For that reason, this work has carried out all the evaluation with an implementation over real nodes using the Contiki OS.

The handover latency to start the process in Mobile IPv6 is influenced by the RA frequency, since the MN needs to wait for the expiration of the default router lifetime before sending new router solicitations. The other option is to wait for the router advertisement from the 6LoWPAN Border Router, which used to be very low in order to optimise lifetime from the sensors, since this packet needs to be treated by all the nodes, even after that they have already being connected.

In our own implementation, the measure is taking into account the association time which is the union of the RA reception and the auto-configuration from the MN. Therefore, this also presents an extra time. The main factor is the movement detection; the best case could be that
all the networks are working in the same channel in order to be able to see other RAs from networks located at the neighbourhood before losing the connectivity with the current network, and requiring an active scan.

Figure 29: a) Time to establish the association after scan in the visited network. b) Time to receive the router advertisement in the visited network. c) Time to send the BU and receive the BA

Figure 30. Handover time for the Lightweight Mobile IPv6 protocol without scan phase.
6.7.6 Transmissions

The use of triangular routing presents an impact on downstream and upstream time transmissions since there is an intermediate routing of the packets to the foreign network. However, the use of security has also an impact not only in the packet size, but also in time. This increased time is related with the time spent by the HA and the MN to include/remove ESP headers and encrypt/decrypt the packet. Home Agent and Mobile Node must include/remove 24 bytes from ESP Header (SPI, sequence and nonce using AES-CBC) and 16 bytes from ESP Trailer (Padding, Padding length and Next Header). In order to demonstrate the impact of the security during transmissions, a set of tests has been carried out. This test takes measures from Round Trip time of UDP packets with several lengths, up to 1100 bytes of payload since when headers (outer IPv6, ESP Header, inner IPv6 and ESP Trailer) are added, the length of the packets is near to the IPv6 MTU.

Figure 32 shows RTT times measured from the test; these reflect the increased time when ESP security is used to avoid attackers along the mobility. The increased time along the several payload length goes from 19.9 ms as minimum up to 36.6 ms with an average of 25.8 ms.
6.7.6.1 Related works

Wireless Sensor Networks (WSN) was one of the basis areas for the IoT. Nowadays, several of IoT resources are based on wireless devices designed over protocols such as IEEE 802.15.4, which is the main protocol to develop WSN.

Mobility for WSN was addressed in the previous work. However, the majority of these proposals were defined from a point of view where IP integration was not considered. The IP integration is the main difference between the previous WSN solutions and the current IoT.

This related work is focused on the IP-based solutions. For this purpose, this section analyses, on the one hand, the different solutions for mobility supporting IPv6 from a general point of view, i.e. not considering the features and constraints of the IoT resources.

On the other hand, the first approaches for mobility support with consideration of the specific features, requirements and constrains of IoT resources are analysed. MIPv6 protocol is the most studied and well-known protocol to provide mobility in IPv6 networks. It was considered not suitable for 6LoWPAN nodes, since it brings an enormous overload for MN, because MN is involved during all the handover processes, with very weighty messages, and high processing requirements [71].

For this reason, this work has carried out with a lightweight version of Mobile IPv6 in order to make it suitable for IoT resources such as 6LoWPAN nodes. Hierarchical Mobile IPv6 (HMIPv6) [72] is an optimisation of the MIPv6 regarding the subject of micro-mobility in a well-known architecture that is composed of a Home Agent (HA), gateways and several access routers to increase coverage. When MN changes access point, it only needs to update its local short 16 bit address with the gateway. Its IPv6 Care of Address (CoA) remains the same. Short addresses are managed by the topology control algorithm. Mobile IP Fast Authentication Protocol (MIFA) [73] introduces a very simple concept on how to support macro-mobility with authentication. It defines a group known as L3-FHR (Layer 3 Frequent Handover Region) composed of the neighbours of a network, where a mobile device is able to move. This protocol also increases the functionality of the mobility entities in the visited networks, making them responsible for the authentication of the mobile nodes.

Fast Handover for Mobile IPv6 (FMIPv6) [74] is characterised by the MN being able, through the use of link layer specific mechanisms, to find available access points to request subnet information. Thereby, MN is capable of configuring its CoA while it is still located in its current network. This considerably reduces the handover latency.

The solution proposed in this work has been focused on the main protocol, i.e. Mobile IPv6, for 6LoWPAN networks. Initial approaches have been defined to support mobility in 6LoWPAN. For mobility based on node, a solution based on 6LoWPAN Neighbour Discovery [76][77] has been defined, which supports micro-mobility, since it supports Extended 6LoWPANs, i.e. a group of 6LoWPAN networks interconnected through a backbone.

Regarding Mobile IPv6, a lightweight version of the Mobile IPv6 messages was suggested in [78]. This approach was similar to the idea of header compression used for IPv6 messages over IEEE 802.15.4 [56].

Finally, other approaches can be found based on Network Mobility (NEMO) [30][60][61][62] to reduce overload in MN, and Proxy Mobile IPv6 (PMIPv6) [30][75], where MN does not require mobile functionality in its IPv6 stack, because exchange of messages between MN and HA are delegated to a new network device, which acts as Proxy between them. These protocols are specifically appropriate for 6LoWPAN, because this avoids the involvement of
MN in mobility-related signalling, but they are not applied in our approach since we are focused on end device mobility.

### 6.8 Conclusions

Smart Objects are highly capable of integrating and transferring enriched data from environmental sensors, parking, activities, behaviours, home automation, intelligent transportation systems, clinical devices from mobile health, and Ambient Assisted Living (AAL) environments [30].

Wireless Sensor Networks (WSNs) are usually appointed as the missing extension to connect the virtual to the real world. Constituted by low power and low cost small nodes, WSNs have been projected for thousands of applications in several areas, such as military, healthcare, education, environment, transport, and industrial automation. Although the number of uses is increasing daily, the existent WSNs do not cover the half of them.

Such situations happen because the technology evolution is not following the theoretical WSNs potential. The network technology should not limit the application; instead, it should adequately respond to its requirements. Therefore, independent of people’s activity within a specific scenario, WSNs must be able and ready to support it and to provide the required reliability. In real-time monitoring scenarios such as the above, high latencies and packet losses might signify failure to detect a critical anomaly, potentially leading to disaster and/or casualties. In this context, there is a strong motivation to develop solutions for mobility support [66], since WSNs are seen as linking the virtual to the real world, and consequently it is natural that the probability of monitoring mobile bodies is truly a high one.

Mobility is one of the most important issues in next generation networks. Mobility based communication increases the fault tolerance capacity of the network, increases the connectivity between nodes and clusters, and deployment of multiple controlled mobile elements can be used to provide load balancing and gathering data.

Wireless Sensor Networks used for our research are IP-based in order to provide features from Internet to WSN such as global connectivity, flexibility, open standards and end-to-end communication with other systems. Particularly, it is based on IPv6 Low-Power Personal Area Networks (6LoWPANs), which are low cost communication networks that allow wireless connectivity in applications with limited power and relaxed throughput requirements. 6LoWPAN networks are constrained by their link layer technology i.e. IEEE 802.15.4, which is characterised as lossy, low-power, low bit-rate, short range and with many nodes saving energy which means long deep sleep periods. Moreover, IEEE 802.15.4 links are asymmetric and non-transitive in nature, and finally they do not define a common domain broadcast; a 6LoWPAN network is potentially composed of a large amount of overlapping radio ranges, eventually federated by either a backbone or a backhaul link.

For the mentioned constrains for 6LoWPAN, the use of classic IPv6 protocols such as Neighbour Discovery (ND) [69], IP Security (IPsec) [70], and Mobile IPv6 (MIPv6) [71] encounter several problems. For example, ND was not designed for non-transitive wireless links, the assumption of traditional IPv6 link concept i.e. a single domain broadcast and heavy use of multicast makes it infeasible [76].

Mobile IPv6 was originally not considered feasible because the overhead and requirements of security based on IPsec, but this work has presented a Lightweight Mobile IPv6 version of the problem that is an optimised version with both minimal yet sufficient for IoT nodes constraints and use cases requirements.

In the same way, IPsec was not considered feasible, since IPsec requires cryptographic primitives, which are very expensive in relation to the number of CPU cycles and memory.
But, it has been presented how to make it feasible with the AES-CBC and AES-CCM cryptosuites, since they are supported by HW and they present a low memory footprint. In this context, we presented a lightweight implementation of Mobile IPv6 with header compression, reduced footprint requirements, and the support for IPsec.

Lightweight Mobile IPv6 has been implemented over the Contiki OS and evaluated successfully offering mobility with a handover under 2 seconds and an integral compatibility with the existing Mobile IPv6 implementations such as MIPv6. For the evaluation of the compatibility with the standard MIPv6, it has been validated with the tests defined by TAHI [65].

In conclusion, protocols such as MIPv6 and IPsec have presented a big challenge for its integration in constrained devices, but this work has presented that it is feasible with a memory footprint around 7,4KB for MIPv6 and 15,5KB for IPsec. The support and comparability with the existing Internet-based protocols is a major requirement in order to reach a proper IoT convergence. The link layer MUST fragment packets in 802.15.4 when IPsec security is used. At the same time, IPsec introduces a brief latency due to the added time to encrypt and to send the over headed packet.

7 Scalable and Efficient Routing in IoT6 Wireless Networks

In IoT6 context, one of the most important elements is Wireless Sensor Networks (WSN). The connecting of WSNs in the Future Internet will enable global remote access from heterogeneous information systems via common information services. To achieve that, recent advances in mechanical and microelectronic circuits enable smaller and cheaper WSN devices. In WSNs, devices distribute the captured information without the need of expensive communication infrastructure. WSNs will allow easy deployment of a large number of autonomous devices being able to monitor and control all type of environments.

However, the wireless connection of WSNs to Internet presents important challenges. WSN devices employ wireless radios to communicate the information from a source to a sink following a multi-hop forwarding approach. To establish multi-hop communications, these devices require routing algorithms adapted to specific properties of WSNs. These networks are formed by thousands of nodes equipped with constrained resources in terms of computing and energy. Wireless communication is the energy bottleneck limiting the autonomy of WSNs. For these reasons, routing protocols must be scalable for an increasing number of nodes and efficient in the number of transmitted messages.

This section studies the main requirements affecting routing algorithms in WSNs [79]. These requirements often are interrelated; sometimes the performance of one requirement is opposite to another one. For instance, the increment of connectivity produces the reduction of the energy efficiency. In WSNs, the design of routing protocols is influenced by the following requirements.

- Energy is the most critical requirement. Sensor nodes are placed in distant positions where electric energy is unavailable. Thus, nodes using batteries must operate autonomously during months or even years. The battery lifetime is determined by the power consumption of three main components (i.e. processing, sensing and communication). These components are disabled during idle states in order to minimise the power consumption. However, wireless communication consumes the majority of the energy, in particular the transmission is the most energy consuming task [88]. To maximise the WSNs lifetime, efficient routing protocols must minimise the usage of wireless radio.
• Scalability is a specific property of WSNs where thousands or hundreds of nodes are deployed to sense a target area. For the limited sensing coverage of nodes, the density of neighbour inside the same radio range is from tens to hundreds [89]. This factor requires distributed protocols where nodes take routing decisions using only local neighbourhood information.

• Fault tolerance is a relevant factor for WSNs due to common problems in device hardware and wireless communication. Failures in nodes may happen due to physical damage or power lack. Another important factor is the fluctuation of link qualities between wireless radios. These failures break multi-hop paths and damages routing protocols reducing their performances in terms of latency, efficiency and reliability [90].

• Connectivity is of great importance in dense WSNs. The connectivity is defined as the capacity of establishing communication between any two individual nodes. To achieve connectivity among all nodes, WSNs require localised communication protocols avoiding the overhead of routes maintenance techniques and flooding discovery mechanisms [87].

### 7.1 Existing routing paradigms

This section summarises the state-of-the-art of the main routing paradigms for WSNs. According to several studies [90][93][94][95] routing protocols are classified into four types: data-centric (or attribute-based), hierarchical, QoS (Quality of Service) and geographic (or location-based). In data-centric routing all nodes play the same role and provide the same operations, and the routing process is based on query messages. In hierarchical routing, there are two types of nodes: cluster-head and normal nodes. Cluster-head nodes aggregate data from normal nodes to decrease data traffic transmitted toward the sink. In QoS routing sensor nodes forward packets in order to guarantee network parameters such as energy efficiency, effective sample rate and bounded delay. Geographic routing nodes exploit their position information to forward packets toward the destination. Nodes take routing decisions according to the positions of their neighbours and the position of the destination. For each paradigm, we present several examples of routing protocols below.

<table>
<thead>
<tr>
<th>Paradigms</th>
<th>Protocols</th>
</tr>
</thead>
<tbody>
<tr>
<td>Data-centric</td>
<td>SPIN</td>
</tr>
<tr>
<td></td>
<td>Directed-Diffusion</td>
</tr>
<tr>
<td></td>
<td>Rumor-Routing</td>
</tr>
<tr>
<td></td>
<td>COUGAR</td>
</tr>
<tr>
<td></td>
<td>ACQUIRE</td>
</tr>
<tr>
<td>Hierarchical</td>
<td>LEACH</td>
</tr>
<tr>
<td></td>
<td>PEGASIS</td>
</tr>
<tr>
<td></td>
<td>TEEN and APTEEN</td>
</tr>
<tr>
<td></td>
<td>SOP</td>
</tr>
</tbody>
</table>
Table 6: Routing Paradigms for Wireless Sensor Networks

<table>
<thead>
<tr>
<th>QoS</th>
<th>SPEED</th>
</tr>
</thead>
<tbody>
<tr>
<td>Geographic</td>
<td>SAR</td>
</tr>
<tr>
<td></td>
<td>GFG</td>
</tr>
</tbody>
</table>

7.1.1 Data-centric routing

Data-centric routing is based on the sending of query messages from a single sink to request information of multiple sensor nodes. In dense WSNs, the sink has severe difficulty in selecting sensor nodes since the use of global identifiers is impossible because of the large number of deployed nodes. Unlike traditional address-based routing where routes are created using network addresses of nodes, in data-centric routing query messages contain attributes to specify requested data. So, the sink sends a query message to sensor nodes placed in a concrete region and waits for their data answers. Moreover nodes aggregate data during the forwarding to decrease the information redundancy.

Two earlier data-centric proposals SPIN [96] and directed diffusion [97], considered data negotiation among nodes to avoid redundancy and save energy. The negotiation concept motivated several protocols such as Rumor-routing [98], COUGAR [99] and ACQUIRE [100]. These protocols and their main ideas are presented in the following section.

Content-Based Network (CCN) is another approach for data-centric routing. In CCN, the communication network features a new advanced communication model where messages are not given explicit destination addresses, and where the destinations of a message are determined by matching the content of the message against specific predicates (“interests”) declared by nodes. Routing in a content-based network aims at propagating these predicates to create a path for the return data and the necessary topological information in order to maintain loop-free and possibly minimal forwarding paths for messages. CCN is described widely in D3.2 for Smart Routing.

7.1.1.1 SPIN (Sensor Protocols for Information via Negotiation)

Heinzelman et al. [96] proposed SPIN which is a family of adaptive protocols for data-centric routing. The main idea is to distribute data from sources in the whole network. SPIN considers that sensor nodes located close to each other may obtain similar data which can be aggregated. The protocol identifies data by assigning high-level names, called meta-data. Metadata are determined by each application to increase the flexibility. Nodes employ metadata to negotiate the data transmission with their neighbours in order to eliminate redundancy.

The SPIN communication follows 3 phases using three types of messages: ADV, REQ and DATA. First, each node obtaining new data advertises the specific metadata in an ADV message. Neighbours interested in the incoming metadata send a REQ message to request the DATA message with the data. The communication process is repeated till the data is fully diffused in the network.

The aggregation process of SPIN decreases the network traffic and power consumption. Moreover SPIN behaves well with topological changes owning to nodes require only 1-hop neighbourhood information. Nevertheless, the advertisement scheme does not ensure data delivery when neighbours are not interested.
7.1.1.2 Directed Diffusion

Intanagonwiwat et al. [97] proposed directed diffusion; a data-centric protocol for WSNs. Directed diffusion enables data distribution by a naming scheme of attribute-value pairs. This scheme permits the data aggregation from various sources in order to eliminate redundancy.

Directed diffusion uses 3 phases: interests requesting, gradients building and data dissemination. Using attribute-value pairs the sink floods a query message with a data interest in the whole network. During the interest propagation each node stores a gradient which indicates the previous query sender. The propagation process constructs various paths between the sink and sources. Finally, the sink chooses the best path which is strengthened by the initial interest retransmission. Sources employ the chosen paths to transmit data toward the sink. Intermediate nodes aggregate data from various sources in order to decrease the transmission overhead and power consumption. In addition, directed diffusion provides a repairing technique for broken paths.

Unlike SPIN, direct diffusion is an on-demand protocol without global network information.

Direct diffusion provides an aggregation tree to communicate data from sources to the sink. The usage of the best paths is ideal to support applications needing high-rate data flows. Nevertheless, the reactive query technique is inefficient in scenarios where there is periodic data delivery (i.e. environmental monitoring) or the route is utilised only once (i.e. alert systems).

7.1.1.3 Rumor-Routing

Braginsky et al. presented a variant of directed diffusion, called Rumor routing [98]. The objective is to decrease the high overhead of queries flooding. Rumor routing is designed to request few interest events. In rumor routing, events are distributed in the network utilising long-lifetime packets, called agents. Each sensor node observing an event stores it in a table and produces an agent. Agents traverse the network and propagate data of local events to far away nodes which store the events in their tables. Employing their events table, sensor nodes answer to particular event queries of the sink node.

Unlike directed diffusion, rumor routing requires no flooding to distribute events minimising the energy consumption and communication overhead. Results through simulation showed that rumor routing enhances significantly power efficiency and supports nodes failures. Nevertheless if the application needs many events, the maintenance cost of agents and event tables becomes impracticable.

7.1.1.4 COUGAR

Yao et al. introduced an alternative data-centric protocol called COUGAR [99] which considers the network as a distributed database system. COUGAR utilises declarative queries to abstract the queries computation from the network layer.

The abstraction is provided through a new query layer between the application and network layers. Based on database systems, COUGAR proposes an architecture where nodes choose a leader to aggregate and transmit the information to the sink. The sink is in charge of producing a query plan that incorporates the information of data flows. Also, this plan explains how to choose a leader for a specific query. The proposed architecture considers the data aggregation in all nodes to decrease the power consumption and the traffic toward the leaders.

Nevertheless, the independent layer for queries computation possesses some disadvantages. First, including query layer on nodes might add an extra overhead in terms of routes storage.
Second, the data aggregation in all nodes needs time synchronisation. Third, leader nodes must be kept to avoid communication failures.

7.1.1.5 ACQUIRE

In [100], Sadagopan et al. proposed a data-centric technique for querying sensor networks called ACtive QUery forwarding In sensoR nEtworks (ACQUIRE). Like COUGAR, ACQUIRE sees the network as a distributed database where a query message contains multiple sub-queries. In ACQUIRE, the sink node disseminates a query in the network. During the dissemination each sensor node tries to answer utilising its pre-cached information. When the pre-cached information is not updated, the node transmits the query to its d-hops neighbours. Once d-hop neighbours resolve fully the query, they answer to the sink. This process permits simple and complex queries.

Unlike data-centric approaches employing flooding, ACQUIRE provides a power efficient querying mechanism by adjusting the variable d. Note that when d is equal to the network diameter: the algorithm behaves similar to flooding. Otherwise if d is too small, the query travels only few hops. Authors presented a mathematical modeling which finds an optimal value of d = 4 in well-distributed networks. Nevertheless, results of the mathematical model are not validated through simulations.

7.1.2 Hierarchical routing

Hierarchical or cluster-based routing is a well-known technique to provide scalability and power efficiency in WSNs. The idea is to aggregate data from various nodes which belong to a specific cluster in order to decrease the traffic toward the sink. Hierarchical routing consists of two stages: first-stage is employed for forming cluster heads (CHs) and second-stage is utilised for routing. Cluster formation is based on the power and proximity of nodes to the observed region. In cluster-based architectures high-power nodes act as CHs to aggregate and transmit the information from low-power nodes sensing the target phenomenon.

For WSNs one of the first hierarchical routing protocols is LEACH [101]. Based on LEACH, several algorithms have been developed such as PEGASIS [102] and TEEN [103]. Moreover some alternative hierarchical protocols have been proposed (i.e. SOP [104]). Below, we describe these hierarchical routing protocols.

7.1.2.1 LEACH

Heinzelman et al. [101] proposed Low Energy Adaptive Clustering Hierarchy (LEACH). LEACH forms clusters of sensor nodes according to the received signal strength and employ local cluster heads (CHs) as relays toward the sink. It employs a combined TDMA/CDMA scheme to decrease the number of intra-cluster and inter-cluster collisions. In LEACH, a distributed formation scheme chooses arbitrarily a few nodes as CHs and rotates the CH role among all nodes to balance the power consumption and maximise the network lifetime.

In LEACH each CH aggregates data received from sensor nodes belonging to the specific cluster and transmits a compressed packet to the sink.

The operation of LEACH is divided into two stages: CHs organisation and data transmission. The duration of data transmission is longer than the duration of CHs organisation in order to reduce the control overhead.

The CHs flood advertisement packets to the remaining sensor nodes. According to the signal strength of the advertisements received, each sensor node chooses the best CH and transmits a joining packet to the chosen CH. Once a CH received all joining packets, this CH creates a TDMA scheme considering the number of joining nodes and informs them about their
transmission time slot. During the transmission stage each node senses and sends data to its
chosen CH node. Each CH aggregates the received data and transmits the compressed
information to the sink. To decrease collisions and interferences each CH communicates
employing different CDMA codes. Periodically, the network runs again the organisation
process to choose new CHs.

LEACH is fully distributed and needs no global network knowledge. When some CH nodes
die, LEACH enables a dynamic cluster formation increasing the network lifetime.
Nevertheless LEACH requires that sensor nodes have high computing capacity to support two
MAC layers. Moreover the dynamic formation requires extra overhead (i.e. CH changes,
advertisement packets, etc.) increasing the power consumption.

7.1.2.2 PEGASIS

Lindsey et al. [102] presented an improved version of the LEACH protocol named Power-
efficient GAthering in Sensor Information Systems (PEGASIS). PEGASIS objectives are
increasing the network lifetime and decreasing the bandwidth consumption. Unlike LEACH
which forms multiple clusters, PEGASIS creates near optimal chains where nodes require
only local communication with their 1-hop neighbours. PEGASIS constructs the chain in a
greedy fashion choosing in each hop the neighbour closest to the sink. To choose the closest
neighbour, each node utilises the 1-hop neighbours distances estimated according to their
signal strength. Once the chain construction finishes, each node using its closest neighbour
transmits data which are aggregated in each hop till reaching the sink. When the sink receives
data from all nodes, the round finishes and the chain-creation process is repeated. This
process decreases the necessary power for transmitting data to the sink and distributes the
energy consumption among all nodes.

Simulation results showed that PEGASIS outperforms LEACH about 100–300\% for different
network topologies and sizes. So, PEGASIS avoids the computing overhead of dynamic
cluster formation and decreases the number of transmissions by optimising the data
aggregation. However PEGASIS introduces excessive packet delay from distant nodes on the
chain. Also, nodes require energy neighbourhood information to take routing decisions
increasing the control traffic.

7.1.2.3 TEEN and APTEEN

Manjeshwar et al. proposed two hierarchical routing protocols called Threshold-sensitive
Energy Effi cient sensor Network (TEEN) [103] and Adaptive Periodic Threshold-sensitive
Energy Effi cient sensor Network (APTEEN) [105]. TEEN and APTEEN are based on a
hierarchical architecture where nodes closer to the sink form clusters. Both protocols were
designed for time-critical applications where the network operates reactively to the changes of
sensed parameters (i.e. temperature, humidity, etc.).

In TEEN, sensor nodes measure periodically parameters of the environment, but data
transmission is done only for relevant information. After the clusters formation, each cluster
head (CH) diffuses two thresholds for each measured parameter to its members. First, the hard
threshold represents a minimum interesting value of the sensed parameter. Second, the soft
threshold indicates a small value change of the sensed parameter. The data transmission is
only generated when the sensed value is bigger than the hard threshold and the value change
is equal to or greater than the soft threshold.

The main advantages of TEEN are the suitability for time critical applications and the
reduction in power consumption. As data transmission consumes more power than data
sensing, the energy consumption in TEEN is less than proactive protocols such as LEACH.
Moreover in the formation of each cluster, the CH distributes the thresholds, and the user can
adjust both hard and soft thresholds in order to optimise the trade-off between power efficiency and data accuracy. Smaller value of the soft threshold produces more accurate information, but more power consumption.

On the other hand, APTEEN is an extension of TEEN and provides both proactive and reactive techniques for periodical collection and time-critical events, respectively. In APTEEN, each CH broadcasts the sensed parameters, their thresholds, the TDMA schedule and a count time to its members. The count time represents the maximum period in which a node cannot send data, after that time data transmission is forced. Each CH also performs data aggregation in order to reduce consumption. APTEEN offers a high flexibility by supporting three different query types: historical for analysing past data values, one-time for taking a snapshot view of the network and persistent for monitoring an event during a specific period. However the main drawback of APTEEN is the extra complexity required to develop the threshold functions and the count time.

Simulated results of TEEN and APTEEN have demonstrated that both algorithms outperform LEACH. Moreover the APTEEN performance is somewhere between TEEN and LEACH in terms of network lifetime and energy dissipation. In most of the tested scenarios, TEEN obtains the best results because it decreases the transmission overhead. The main disadvantages of both TEEN and APTEEN are the overhead and complexity related with the method of developing threshold-based functions, the way of dealing with parameter-based naming of queries and the technique of forming clusters at multiple levels.

### 7.1.2.4 SOP

Subramanian et al. [104] presented a Self-Organising Protocol (SOP) and a taxonomy of WSNs applications. According to such taxonomy, authors make an architecture supporting heterogeneous sensor nodes that can be mobile or stationary. In SOP, nodes monitor the environment and transmit their data to a pre-configured set of nodes acting as routers. These routers forming the communication backbone are stationary and forward monitoring data to sink nodes. Each sensor node is connected to a router to participate in the network. Authors proposed a routing architecture which needs node identification. In this architecture each node is identifiable by means of the address of its router. The routing architecture follows a hierarchical model where clusters of sensor nodes are created.

To support fault tolerance SOP utilises for packet broadcasting the Local Markov Loops (LML) algorithm which performs a random walk on spanning trees of a graph. The LML algorithm employs routers to keep all sensor nodes connected. In this algorithm nodes can be identified individually in the network.

Thus SOP is suitable for applications where any node may be the communication destination. Due to the reduced number of routers, the proposed architecture achieves power efficiency.

As authors’ results show, SOP requires a small cost for keeping routing tables and forming the hierarchical architecture. Owning to LML broadcasting trees, SOP decreases the energy consumption for broadcasting which is less than the energy required by the SPIN protocol [96]. In addition LML broadcasting trees enable fault tolerance (i.e. died nodes). However the main disadvantage is the proactive organisation of LML which produces additional overhead. Another problem is associated to the hierarchy formation when there are many disconnected points in the network. In these cases the increment of reorganisation overhead reduces significantly the communication efficiency.

### 7.1.3 Quality-of-Service routing

QoS-based routing protocols balance their performances between traffic quality and power
efficiency. In these protocols, the path formation between sensor nodes and the sink is addressed as a network flow problem. Concretely, nodes delivering data to the sink must satisfy particular metrics such as delay, energy, bandwidth, etc. Various examples of QoS routing protocols are described below.

### 7.1.3.1 SAR

Sohrabi et al. [106] introduced one of the first QoS-based routing protocols for WSNs, called Sequential Assignment Routing (SAR). SAR is a table-driven multi-path solution with a local path restoration technique to prevent failures of single routes. To form multiple paths from source nodes to the sink, SAR constructs trees rooted at 1-hop neighbours of the sink. The path formation results in a multi-route tree composed by all nodes. Each node chooses one of these paths to transmit data according to three factors: the priority level of each message, energy resources of neighbours and QoS on each route. The aim of SAR is minimising the average weighted factors and optimising the network lifetime. The weighted factors are estimated as the product of a weight coefficient related with the priority level of the packet and the additive QoS factors.

SAR provides a route re-computation technique for supporting any topological changes (i.e. nodes failures). The sink triggers periodically the path re-computation, and nodes employ a localised handshake scheme to recover from network failures. The localised handshake recovery keeps the consistency of routing tables between downstream and upstream nodes on each path. Each node detecting any local failure performs automatically a localised path restoration.

As authors’ results demonstrated, SAR provides tolerance to failures and easy recovery. Nevertheless, the algorithm needs an additional cost of routing table maintenance which is infeasible in dense networks.

### 7.1.3.2 SPEED

He et al. presented a sophisticated QoS routing protocol called SPEED [107] which supports soft real-time end-to-end guarantees in WSNs. In SPEED, nodes maintain neighbourhood information and employ geographic routing to find the paths toward the sink. SPEED strives to ensure a pre-configured data rate in the network. To guarantee the data rate they divide the packet delay by the distance to the sink. The scheme prevents congestion in networks carrying high traffic.

SPEED contains a routing algorithm called Stateless Geographic Non-Deterministic forwarding (SNFG) which consists of four extra mechanisms. First, a beacon exchange mechanism provides neighbours’ information such as positions. Second, nodes estimate the 1-hop neighbours’ delay by the elapsed time from data transmissions till the reception of ACK packets. Employing 1-hop delays, each node chooses the neighbour that satisfies the desired rate. Third, a failure detection mechanism checks the forwarding ratio of each neighbour considering a miss when the wished rate is not achieved. If the forwarding ratio is less than a random number between 0 and 1, the packet is dropped. Fourth, SPEED utilises a local rerouting technique to prevent voids when a node cannot find a next-hop neighbour. This technique avoids congestion by transmitting packets back to the sources, when they look for new paths.

Simulation results showed that SPEED outperforms two well-known ad-hoc routing protocols (dynamic source routing (DSR) [108] and ad-hoc on-demand vector routing (AODV) [109]) in terms of miss ratio and end-to-end delay. SPEED consumes less power because of its efficient design in aspects such as control packet overhead and uniform traffic distribution. The good balance is possible through the SNGF routing module which distributes packets in a
reduced forwarding region. Nevertheless, the SNGF routing module does not consider any energy metric. Moreover DSR and AODV were not designed for WSNs, thus SPEED must be compared with efficient routing protocols to understand its realistic performance.

7.1.4 Geographic routing

In WSNs, the location of an event is crucial information for the majority of applications. In most applications, nodes are manually deployed ensuring that routes exist to forward data toward the sinks [110].

To obtain positions, different localisation systems exist such as Global Positioning System (GPS), infrastructure-based localisation techniques, and ad-hoc localisation algorithms [111][112][113][114][115][116][117]. Once location information is available, the operation of communication protocols is simplified improving the energy efficiency considerably.

Geographic routing employs location information of sensor nodes to take forwarding decisions. The objective of geographic routing is to reduce the distance towards the destination in each hop. So, nodes forward the packet through their closer 1-hop neighbour toward the destination. To take forwarding decisions, nodes need only local neighbourhood information. Unlike previous algorithms, geographic routing requires neither routing tables nor flooding discovery activities. In fact, geographic routing prevents the extra cost of routing information maintenance which involves high energy consumption and high fault tolerance. Given that geographic routing is the base for our contribution, these protocols are analyse din detail in the next section.

7.1.5 Gradient routing

Following a previous review [81], the main concept of Gradient routing and its most important protocols is presented. The concept of gradient is particularly useful for convergecast networks such as WSNs. In the simplest convergecast scenario, all traffic is sent to a single sink node. In this Schurgers case, a single gradient - rooted at the sink node - is built and maintained in the network. The next figure depicts a topology where nodes are assigned heights calculated as a function of hop count. When node Y at height 3 sends a message, it sends it to its neighbour of smallest height I; similarly I relays the message to G, and G to A.
7.1.5.1 Gradient-Based Routing (GBR)

GBR [83] is the canonical gradient routing protocol. On top of the basic idea described above, an energy-based scheme can be used as a data dissemination technique, where a node increases its height when its energy drops below a certain threshold so that other sensors are discouraged from sending data to it.

7.1.5.2 GRAdient Broadcast (GRAB)

GRAB [84] enhances the reliability of data delivery through path diversity. GRAB builds and maintains a gradient, providing each sensor the direction to forward sensing data. However, unlike all the previous approaches, GRAB forwards data along a band of interleaved mesh from each source to the receiver.

To collect data reports, the sink first builds a gradient by propagating advertisement (ADV) packets in the network. The height at a node (dubbed “cost” in GRAB) is the minimum energy overhead to forward a packet from this node to the sink along a path; nodes closer to the sink have a smaller cost. GRAB makes the assumption that each node has the means to estimate the cost of sending data to nearby nodes, e.g., through SNR measurements of neighbours’ transmissions. Each node keeps the cost of forwarding packets from itself to the sink. Since only receivers with smaller costs may forward the packet at each hop, the packet is forwarded by successive nodes to follow the decreasing cost direction to reach the bottom of the cost field, which is the sink.

Multiple paths of decreasing cost can exist and interleave to form a forwarding mesh. To limit the width of this mesh in order to avoid creating excessive redundancy and wasting resources, a source assigns a credit to its generated packet. The credit is some extra budget that can be consumed to forward the packet. The sum of the credit and the source’s cost is the total budget that can be used to send a packet to the sink along a path. A packet can take any path...
that requires a cost less than or equal to the total budget. Multiple nodes in the mesh make collective efforts to deliver data without dependency on any specific node.

Performance analysis of GRAB shows the advantage of interleaved mesh over multiple parallel paths and shows that GRAB can successfully deliver over 90% of packets with relatively low energy cost, even under the adverse conditions of node failures and link message losses.

### 7.1.5.3 Collection Tree Protocol (CTP)

The Collection Tree Protocol (CTP) [85] uses Expected Transmission Count (ETX) as a link metric for setting up the gradient. Using ETX, the height of a node indicates how many times a message originated at that node is transmitted before it reaches the sink. These transmissions include the hops from node to node, as well as the retransmissions needed upon link failure.

CTP piggybacks gradient setup information in beacon messages, and uses the Trickle algorithm [86] to regulate the beaconing interval. In the absence of topological changes, this interval is regularly doubled until it reaches a maximum value which triggers only a few beacons per hour. Upon topological changes, the interval is reduced to allow for fast gradient re-convergence. Experimental results on 12 different testbeds show that CTP requires 73% fewer beacons than a solution with a fixed 30-second beacon interval, for an idle duty cycle of 3%.

### 7.1.5.4 Routing Protocol for Low Power and Lossy Networks (RPL)

RPL (pronounced “Ripple”) [82] is a routing protocol designed for LLN with the expectation of joining with thousands of nodes network. It supports three traffic patterns: multipoint-to-point (MP2P), point-to-multipoint (P2MP) and point-to-point (P2P). RPL captures most of the ideas exploited by the previous gradient routing proposals for convergecast WSNs.

In RPL, a gradient (called Directed Acyclic Graph, DAG) is defined by the following fours elements: (1) a set of sink node(s), (2) the set of atomic metrics collected on each link (e.g. bandwidth, Packet Deliver Ratio – PDR, etc.), (3) how these atomic metrics are combined to obtain the link’s cost (by adding, multiplying, etc. the atomic metrics) and (4) how link costs are combined to form a multi-hop path cost (by adding, multiplying, etc. the link costs). A given network can contain multiple gradients.

The main idea of RPL is the high degree of autonomy in the nodes level through building a Destination Oriented DAGs (DODAGs) rooted towards one sink (DAG ROOT) identified by a unique identifier DODAGID. The DODAGs can be optimised according to an Objective Function (OF) based on differential application specifications and identified by an Objective Code Point (OCP), which indicates the dynamic constraints and the metrics such as hop count, latency, expected transmission count, parents’ selection, energy etc. A rank number is assigned to each node which can be used to determine its relative position and distance to the root in the DODAG.

A set of multiple DODAGs can be in a RPL INSTANCE which is a very important concept in RPL. A node can be a member of multiple RPL INSTANCES but can belong to at most one DODAG per DAG INSTANCE. DODAG Information Object (DIO) messages are used to construct and maintain the upwards routes of the DODAG with the information, such as RPL INSTANCE, DODAGID, RANK and DODAGVersionNumber. A trickle timer of RPL can regulate the transmission of DIO messages and help to eliminate redundant control messages. Each node has to monitor its neighbours’ DIO messages before joining a DODAG. Then, it selects a DODAG parent set from its neighbours according to the latency they advertise, OF and computes its RANK. Destination Advertisement Object (DAO) messages are used to
maintain downward routes by selecting the preferred parent with lower rank and sending a packet to the DAG ROOT through the parents set. Another common message is DODAG Information Solicitation (DIS) that can be sent by any node in RPL to solicit DIO messages from its neighbourhoods for update routing information.

RPL has two mechanisms to repair the topology of the DODAG, one is to avoid the loops and allow nodes to join or rejoin a new position and another one is called global repair. Global repair is an operation mode that the DODAG ROOT increments the DODAGVersionNumber to create a new DODAGVersion. Another mechanism is local repair which can allow the DODAG repaired within the DODAG Version. For example, the node can detach from the DODAG, advertise a rank of INFINITE RANK to inform its sub-DODAG, and finally re-attach to the original or a brand-new DODAG.

### 7.1.6 Routing paradigms comparison

This section provides a comparison of the four routing paradigms designed for WSNs: data-centric, hierarchical, QoS and geographic. For these four routing paradigms, their advantages and disadvantages summarised in the next table are discussed.

<table>
<thead>
<tr>
<th>Paradigms</th>
<th>Advantages</th>
<th>Disadvantages</th>
</tr>
</thead>
<tbody>
<tr>
<td>Data-centric</td>
<td>High Energy-Efficiency</td>
<td>Low Scalability</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Low Fault-Tolerance</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Low Connectivity</td>
</tr>
<tr>
<td>Hierarchical</td>
<td>High Energy-Efficiency</td>
<td>Low Fault-Tolerance</td>
</tr>
<tr>
<td></td>
<td>High Scalability</td>
<td>Low Connectivity</td>
</tr>
<tr>
<td>QoS</td>
<td>High Fault-Tolerance</td>
<td>Low Energy-Efficiency</td>
</tr>
<tr>
<td></td>
<td>High Connectivity</td>
<td>Low Scalability</td>
</tr>
<tr>
<td>Gradiant</td>
<td>High Energy-Efficiency</td>
<td>Low Fault-Tolerance</td>
</tr>
<tr>
<td></td>
<td>High Scalability</td>
<td>Low Connectivity</td>
</tr>
<tr>
<td>Geographic</td>
<td>High Energy-Efficiency</td>
<td>Low Fault-Tolerance</td>
</tr>
<tr>
<td></td>
<td>High Scalability</td>
<td>Low Connectivity</td>
</tr>
</tbody>
</table>

*Table 6: Advantages and Disadvantages of Routing Paradigms*

Data-centric routing decreases significantly the communication overhead in networks where sensed data is of interest to a unique sink. In these scenarios, data-centric algorithms construct power efficient paths between sensor nodes and the sink. The path formation depends on attribute-value pairs often determined in the application layer. However, the limitation of pre-configured attribute-value pairs avoids sophisticated queries. These attribute-based protocols are inefficient in applications requiring connectivity to multiples sinks. Moreover, the path
formation requires flooding mechanisms that are not scalable and robust in dense WSNs with dynamic topologies.

Hierarchical routing is proposed as an efficient and scalable technique to route monitoring data of node groups to the sink. In these protocols each cluster-head node aggregates monitoring data of its region to decrease the traffic toward the sink. The additional overhead of cluster formations is traded for the energy saving in the data transmission phase in dense networks. Nevertheless this does not pay off in small networks with multiple sinks. Hierarchical protocols are inefficient in dynamic topologies (i.e. most of WSNs) where cluster updates are frequently producing excessive overhead.

Unlike data-centric and hierarchical paradigms proposed for power efficiency, QoS routing is focused on ensuring minimum performances in terms of delay and bandwidth. QoS techniques keep multiple paths between source nodes and sinks to ensure the network connectivity even when there are communication failures. The main disadvantage is the additional cost of keeping routing tables which comprises resource constraints, i.e. memory and energy, in dense WSNs.

Gradient protocols are focused on improving the efficiency and scalability in multipoint-to-point routing where RPL is the most important proposal. In RPL, DODAG formation and multipoint-to-point forwarding mechanisms are elegant and provide good performance. Moreover, DIO message generation/processing rules and the Trickle timers can be implemented easily and the state required in each router is minimal and bounded.

However, recent studies [80], [81] of RPL present several relevant issues regarding bi-directionality of links and the possibility of loops. The DAO mechanism is less efficient and scalable to enable downward routes, bi-directional traffic flows and sensor-to-sensor flows because all traffic has to go through the root. Also, RPL provide two incompatible modes-of-operation: storing mode, wherein all routers are expected to have “unbounded” memory (or, at least, enough to store complete routing tables), and non-storing mode necessitating source-routing thus the possibly of more fragmentation and the higher probability of IP packets being lost. Both operation modes do not support the scalability and efficiency required for resource-constrained devices.

Moreover, experimental tests show that RPL generates routing loops in real lossy testbeds. Even in non-storing mode, where the DODAG root performs source-routing to destinations inside the network, loops are a problem: while they can be detected when constructing the source route, the only corrective measure that the DODAG root can take is to trigger global reconstructions of the DODAG producing a complete “reboot” which is unfeasible in dense WSNs.

Geographic routing exploits location information of sensor nodes to take forwarding decisions. By the knowledge of location information, geographic routing provides the most efficient and scalable scheme to forward packets in comparison to previous paradigms. Unlike previous paradigms geographic routing needs neither maintaining routing tables nor flooding discovery activities. This paradigm decreases memory, traffic, computation and energy consumption because routing decisions are performed based on the positions of 1-hop neighbours and the destination. Geographic routing needs local neighbourhood information supporting fast responses to topology changes. Taking localised routing decisions also offers a scalable solution in dense WSNs. Geographic routing using efficient destination discovery mechanisms [118] enables full connectivity among all nodes.
7.2 Background on geographic routing

Based on previous research studies [119][120][121], an extensive overview of Geographic Routing (GR) and its two main forwarding strategies are presented: greedy and face. In GR, forwarding decisions are based on the location of the destination and the positions of neighbouring nodes in each hop. In greedy strategy, the current forwarder holding the packet chooses a closer neighbour as next hop reducing the distance toward the destination. Nevertheless, a greedy strategy does not ensure the packet delivery in sparse networks with void areas. A void area appears, when the current forwarder has no neighbours closer to the destination than itself and becomes a local maximum. To resolve a local maximum, face strategy enables that the packet advances around the perimeter of the void area. Combinations of greedy and face strategies provide the most efficient and scalable GR solutions.

7.2.1 Network Model and Assumptions

As most of routing solutions, GR considers that wireless networks follow a Unit Disk Graph model (UDG)[122]. Each link between two nodes \( u \) and \( v \) is represented as an edge \( e = (u, v) \), if the distance \( |uv| \) is lower than the radio range \( r \). The UDG model assumes that all nodes have the same radio range, and all links are bidirectional. Moreover GR assumes the following conditions:

- Each node determines its geographic location using an existing positioning technique. As location information is key in many applications, many positioning mechanisms have been proposed. To obtain their locations, nodes may employ any positioning system based on extra hardware (i.e low-power GPS), distributed algorithms (i.e., APS [115], RPE [116], DPE [117]) or virtual coordinates [135][136]. Otherwise in stationary network, each node can be pre-established with its fixed position.
- Each node knows the position of its 1-hop neighbours. Nodes can discover the neighbourhood information by a beaconsing mechanism. Using the beaconsing mechanism, nodes broadcast periodically control messages, called beacon that includes their positions and identifiers.
- The source node knows the position of the destination. To map node identifiers to geographic locations, several location service mechanisms have been introduced in the literature [118]. Moreover in some applications, nodes know the pre-configured destination position (i.e. a fixed gateway).

7.2.2 Greedy Forwarding

The first greedy forwarding algorithms were introduced in the 1980s for grid networks [125][126][127]. These strategies were designed to guarantee the packet delivery in uniformly dense networks without void areas. The main idea of these strategies is reducing the distance to the destination in each hop. The current forwarder chooses the next hop among neighbours closer to the destination than itself maximising a local forwarding criterion. In the next subsections, existing greedy strategies according to the criterion they use are described.

7.2.2.1 Greedy Routing Scheme (GRS)

Finn et al. proposed Greedy Routing Scheme (GRS) [126] based on the criterion of the advance, which is equal to minimise the distance toward the destination in each hop. To maximise the advance toward the destination, the current forwarder chooses the neighbour located closest to the destination. The aim of GRS is providing the largest advance per hop in order to follow the shortest path. In GRS, backward forwarding is not allowed to avoid routing cycles. However, this scheme may produce that the packet follows a deviated path for the straight line from the source to the destination. Lateral deviations are common in networks
with low density.

### 7.2.2.2 Most Forward within Radius (MFR)

Takagi and Kleinrock presented the Most Forward within Radius (MFR) routing strategy [125]. MFR introduces the notion of progress as the projection of the neighbour position on the line drawn from the current forwarder to the destination.

In MFR, the next hop selection provides the maximum progress in the destination direction. MFR seeks a double objective: minimising the path length and optimising the number of radio transmissions. The disadvantage of MFR is that a packet may move away from the destination even though there are nodes located on a more direct trajectory or are physically closer.

### 7.2.2.3 Compass Routing (CR)

Kranakis et al. proposed Compass Routing (CR) [127] including the concept of angular distance. CR chooses the neighbour minimising the angle distance based on the forwarder-destination line. In CR, the packet follows the straightest direction from the source to the destination. The main advantage of CR is that the packet is able to go around a void area in certain situations. However, this technique based on the direction is prone to routing cycles. These greedy strategies are illustrated in Figure 34.

![Figure 34: Greedy Routing Scheme (GRS), Compass Routing (CR) and Most Forward within Radius (MFR)](image)

### 7.2.3 Face Routing in Planar Graphs

Based on geographic positions, face routing is the most used strategy to guarantee the packet delivery. The key concept is the traversal of adjacent faces in a planar graph (see Figure 35). A planar graph represents the network topology removing all crossing edges. Using the planar graph, the packet is routed along the edges of the faces. To do that, face forwarding employs the right-hand rule (clockwise rotation) according to the source-destination line. The successful application of this strategy requires the previous planarisation of the network graph using some distributed algorithms described below.
7.2.3.1 Planarisation Algorithms: GG, RNG and CLDP

The planarisation of the network graph is needed for ensuring the packet delivery, because crossing edges cause cycles in face routing. Distributed algorithms are able to construct a planar sub-graph of its local topology by eliminating crossing links among its 1-hop neighbours.

For local planarisations, several distributed algorithms were proposed depending on the geometric criterion employed such as Gabriel Graph (GG) [128] or Relative Neighbourhood Graph (RNG) [129]. In GG, a node u keeps the link to a neighbour v if no nodes exist within the circle whose distance is the segment uv. In RNG, a node u keeps the link to neighbour v if the distance to v is lower than or equal to the distance from both u and v to every other neighbour, otherwise the link is removed. The GG or RNG criteria enable distributed algorithms to obtain planar sub-graphs with only 1-hop neighbourhood information. However, GG and RNG algorithms eliminate many crossing links increasing the path length in face routing. This makes it less efficient than greedy routing.

The main problem of planarisation algorithms is the assumption of an idealised UDG model to represent realistic WSN. WSN is sensitive to inaccurate positions [117] and irregular radio ranges [33]. In real networks, a link between two nodes u and v may not exist even when their estimated distance |uv| is lower than the theoretical radio range. Planarisation algorithms fail to eliminate crossing links and to construct inaccurate planar graph, as described in [130][131].

For realistic WSNs, Kim et al. proposed an alternative solution named Cross Link Detection Protocol (CLDP). CLDP eliminates crossing links and guarantees planar sub-graphs, but increases the control overhead. To detect crossing links, nodes employ the right-hand rule to probe packets with their checked links. All links are probed several times till reaching a convergence situation where they are marked as routable or non-routable. However CLDP needs multi-hop exchanges to identify crossing links, thus the probing overhead grows as the network density increases. For this reason, the authors proposed an on-demand variant [132] that starts the probing process, when the face protocol must choose a link.

7.2.3.2 Face Routing: Compass II and Face-2

Kranakis et al. and Bose et al. introduced the earlier face routing strategies called Compass II [127] and Face-2 [133], respectively. Both strategies employ the right-hand rule to traverse a faces sequence of a planar graph till finding the destination. Compass Routing II constructs a planar sub-graph based on the Delaunay triangulation criterion. Using this sub-graph, the packet traverses fully each face until reaching the closest edge that intersects the source-
destination line. In the intersection, the packet is forwarded to the end-node of this edge where the face is changed to continue the traversal. This process is repeated till the packet reaches the destination. Unlike Compass Routing II, Face-2 constructs a planar sub-graph employing the Gabriel graph criterion. To reduce the traversal process, Face-2 changes to the next face at the first edge that intersects the source-destination line.

![Figure 36: GG (left) and RNG (right)](image)

### 7.2.4 Combined Greedy-Face Routing

Several combined approaches of greedy and face forwarding have been proposed to provide efficient and robust geographic solutions such as Greedy-Face-Greedy (GFG) [133] and Greedy Perimeter State Routing (GPSR) [134]. GFG combines greedy and face strategies based on Compass routing and Face-2, respectively. In GFG, greedy strategy is applied until reaching the destination or a local maximum.

When a local maximum is reached, the local maximum position and the first edge are stored in the packet which is forwarded in face mode. The face mode employs the Gabriel-graph criterion to obtain planar sub-graphs. According to planar sub-graph, Face-2 algorithm is utilised till finding a next hop closer to the destination than the local maximum where the greedy mode is resumed. On the other hand, GPSR provides greedy and face forwarding through Greedy Routing Scheme (GRS) and Face-2, respectively. GPSR constructs planar sub-graphs by the RNG algorithm. Like GFG, packets are first forwarded in greedy mode, when a local maximum is reached, face strategy is then used until greedy forwarding can be resumed.
7.2.5 Beaconless Geographic Routing

Previous geographic protocols assume that nodes know their 1-hop neighbourhood information by exchanging short control messages, called beacons. Each node broadcasts periodic beacons with its identifier and position. Beacons are not forwarded, thus only 1-hop neighbours can receive them. However, the beaconing mechanism reduces the efficiency of geographic protocols. For instance, periodic beacons can interfere with regular data transmission. When the nodes are not taking part in any routing process, the bandwidth and power consumption represent a total waste of resources. To overcome such issues various beacon-less routing protocols have been proposed for WSNs.

Beacon-less routing algorithms employ a reactive scheme to discover 1-hop neighbours and select the next hop. In particular the current forwarder broadcasts the packet to discover its (unknown) neighbours. Neighbours receiving the packet participate as next-hop candidates. They wait for a delay time according to one routing metric (i.e. the destination distance). So, the candidate closest to the destination has the shortest timeout. When the timer expires, the respective candidate transmits first the packet and becomes the next hop. There are two main variants for this reactive scheme. In the first one, the next hop selection is based on candidates’ timers, thus the first transmission cancels the rest. In the second one, the current forwarder after receiving transmissions from candidates chooses explicitly the next hop. The two most representative beacon-less algorithms are Beacon-Less Routing (BLR) [123] and Implicit Geographic Forwarding (IGF) [124], described in the following.

**7.2.5.1 Implicit Geographic Forwarding (IGF)**

Implicit Geographic Forwarding (IGF [124]) is one of the first beaconless geographic routing protocols proposed in the literature. IGF combines MAC and network layers. The selection of the next hop is carried out at the MAC layer, and the actual delivery is done at the network layer. In IGF the node currently holding the packet broadcasts a Request to Send (RTS) frame and waits for the first Clear to Send (CTS) response. Each neighbour receiving the RTS frame
evaluates its own suitability as next hop. The neighbour providing the largest advance towards
the destination is preferred and should answer first. Finally, at the Network layer, the
forwarding node transmits the data message and the selected neighbour confirms the reception
by answering with an Acknowledgment message (ACK).

IGF includes two optimisations to reduce the number of responses and collisions. The first
mechanism avoids simultaneous responses from neighbours based on timers. The second
scheme cancels unnecessary responses when other neighbours’ responses are overheard.

Upon receiving a RTS message, each neighbour sets a timer to wait before answering with a
CTS message. The timer value depends on the reduction in distance towards the destination
provided by the node plus a random component. Thus, neighbours located closer to the
destination answer first. Besides, neighbours overhearing an earlier CTS from another
neighbour cancel their own timers.

IGF defines a forwarding area so that all nodes within that area are separated by a distance
lower than the theoretical radio range. That is, in theory, all nodes inside it can hear one
another. Only those nodes located inside the forwarding area can take part in the selection
process. This is defined that way to allow neighbours to overhear other neighbours’ answers.
However, in practice, radio propagation can make nodes within the forwarding area not to
overhear some answers. Also, as a side effect, the use of a forwarding area may neglect some
neighbours providing a higher advance because of being outside that area.

Figure 38: The forwarding area must be defined so that all nodes inside it can hear one
another. The Reuleaux Triangle fulfils the condition of mutual possible reception for nodes
located within it. Node S holding a message intended for D has three neighbours (N1, N2,
N3). Only N2 is located inside the forwarding area defined by the Reuleaux Triangle. As it
can be seen, transmissions from N1 cannot be overheard neither by N2 nor by N3

7.2.5.2 Beacon-Less Routing (BLR)

Beacon-Less Routing (BLR [123]) relies on a distributed contention process as the only way
of determining the next hop. BLR selects a next forwarder in a distributed manner among all
its neighbouring nodes without having information about their positions or about their
existence. Data packets are broadcasted, and the protocol takes care that just one of the
receiving nodes forwards the packet. This is accomplished by computing a Dynamic
Forwarding Delay (DFD) at each neighbour depending on its position relative to the current
forwarder and the destination.

Among all neighbours providing advance, the one in the best position forwards the data packet first. The remaining neighbours cancel their scheduled transmissions, when they overhear the data packet. To ensure that all nodes detect the forwarding, only nodes within a certain forwarding area take part in the contention to forward the packet. Furthermore, passive acknowledgments are used. That is, by detecting the transmission of the packet, the previous forwarder concludes that it was successfully received by its next hop.

Additionally, BLR includes a face strategy to deal with local maxima. The current forwarder broadcasts a short request, and all neighbours reply with a packet indicating their positions. If there is a neighbour located closer to the destination than the current forwarder, the neighbour is chosen as the next hop. Otherwise the actual forwarder extracts a planar sub-graph (e.g. Gabriel Graph) for its neighbourhood and forwards the packet according to the right-hand rule.

![Figure 39](image)

Figure 39: The contention timer is used to minimise the number of transmissions in the three way handshake and fully distributed contention. In that case, the first transmission of N2 cancels the one of N1

### 7.2.6 Advantages and Disadvantages

Geographic routing (GR) is considered as one of the most scalable solutions for WSNs [30][31]. In GR, nodes require low computation capacities to take routing decisions. Routing decisions are based on the positions of the destination, the current forwarder and its 1-hop neighbours. Each node needs a minimum state to store only 1-hop neighbours positions. Thus, GR performance is not affected by the number of nodes and the neighbour density. For this localised design, geographic protocols provide a fully-distributed way to route packets. A node knowing its 1-hop neighbourhood information is able to decide independently the next hop.

Combining greedy and face strategies, geographic algorithms are transmission efficient and robust solutions for WSNs. Greedy strategy employs the most efficient paths from the source to the destination in uniformly-dense networks. While face strategy guarantees the packet delivery even in sparse networks with void areas. Moreover, these algorithms support networks failures and topological changes. The reason is that both beacon-based schemes permit detecting topological changes by proactive neighbourhood discovery.
However, in realistic WSNs geographic routing, protocols suffer due to the assumptions of perfect wireless communications. Geographic protocols are designed and simulated considering the perfect unit disk graph model to represent wireless sensor networks. However, there are huge differences between a simulated link and a real one as demonstrated in recent studies [32][33]. Under realistic wireless networks, the performance of geographic protocols is severely damaged by frequent communication problems such as interferences, collisions, packet losses, etc.

According to the communication problem of existing routing protocols, we focus on providing scalable, efficient and reliable routing algorithm for realistic wireless networks.

7.3 Scalable, Efficient and Reliable (SERE) Routing for IoT6 WSNs.

In IoT6 wireless networks, many smart objects are connected to Internet through wireless-routers and half-gateways. Smart objects usually have constrained resources in terms of computation, memory, and battery energy. Thus, smart objects need energy-efficient operation based on duty-cycle schedules. Wireless routers and the half-gateway often possess rechargeable energy sources (i.e. solar panel) and more storage capacity (i.e. SD card). Therefore, routers and gateways are able to provide full-time operation to support the interconnection of the overall network. These networks may contain up to tens or hundreds smart objects. Therefore, the scalability is essential in the communication protocol. Moreover, the wireless communication is error-prone due to radio interferences and signal attenuation. Traffic flows consist of periodic data from smart-objects to half-gateways, on-demand queries from half-gateways to a specific smart object and also information exchanging among smart objects inside the same IoT6 network. In addition, the communication must be compatible to the IPv6 standard to support complete interoperability in Future Internet. Summarising, IoT6 wireless networks require the following features of routing protocols:

- Energy efficiency for smart objects.
- Scalable forwarding decisions.
- Distribution of limited overhead.
- Reliable and robust communication.
- Failure tolerance and path recovery.
- Pull/push communication via query/response messages.
- Peer-to-peer communication between smart objects.
- IPv6 compatibility.

In this section, a Scalable, Efficient and RELiable (SERE) routing protocol for IoT6 wireless networks is proposed. SERE is based on the scalable geographic approach that combines the forwarding strategies: greedy and face. Two reliable routing discovery schemes to support communication errors are proposed. These routing discovery schemes enable energy-efficient operation of smart objects. Both schemes are designed as extensions of IPv6 ICMP messages. In addition, SERE incorporates two simple mechanisms to provide failure tolerance and path recovery. In addition, SERE reduces the computation overhead in the IoT6 network to support query/response communication and peer-to-peer connectivity.

SERE considers packet losses which are common in realistic communications of wireless networks. A relevant contribution of SERE is the usage of maximum transmission packet to discover routers. The reason is that bigger messages are often more error-prone than shorter ones. For this reason, short control messages can traverse a link that a big data message cannot. So, only those routers that successfully received the maximum packet before are considered as reliable forwarding candidates. This design is justified by the results of an
experimental analysis described in the next section that show the strong relationship between the message size and the reception error probability.

### 7.3.1 Analysis of Wireless Communications

This analysis assumes that wireless networks follow an Extended Unit Disk Graph model including communication errors. This model considers that all nodes have the same radio range. Each link between two nodes $u$ and $v$ is represented as an edge $e = (u, v)$, if the distance $|uv|$ is lower than the radio range $r$. Moreover, GR considers the packet losses for communication failures such as collisions and interferences. A link between $(u, v)$ does not guarantee that the transmission of the node $u$ is received by the node $v$.

SERE is based on the assumption that the packet size has a direct relationship with the error probability in which bigger packets have lower probability of being received correctly than smaller ones. In this case, discovering the routers using one small control packet may cause routing protocols to select a next-hop which is not able to receive the bigger data packet. The goal is to validate such assumption. Therefore, we run a set of real experiments in order to obtain the relation between Packet Size (PS) and the Packet Reception Ratio (PRR).

In the analysis, the well-known sensor device called Tmote-Sky is employed. The Tmote-sky integrates the following elements: a MSP430 microcontroller (with 10kB RAM memory and 48kB Flash memory), a CC2420 radio chip (based in standard IEEE 802.15.4), a wireless antenna (that provides a radio range of up to 50 meters indoor and 125 meters outdoor according to the manufacture specification) and optionally sensors of humidity, temperature, and light to monitor the environment.

The experimental analysis was performed in an outdoor area of 100x100 meters. Two Tmote-sky nodes (sender and receiver) were placed at 0.5 meter above the ground and connected via USB to laptops. In each experiment the sender node transmits at maximum power (0 dBm) 50 sequences of 100 packets for each packet size. The experiments consider 8 different payload sizes (10, 25, 40, 55, 70, 85, 100 and 115 bytes) and varying the distance between the sender and receiver (from 5m to 120m). For each test the drawn results are the average of 50 sequences in order to achieve a sufficient small 95% confidence interval. The receiver reports to its connected laptop the following measured parameters:

- **PRR.** The Packet Reception Ratio computed in the laptop is defined as the ratio between the number of packets received and the total number of packets sent.
- **RSSI.** The Radio Signal Strength Indicator is a 8-bits value given by the CC2420 chip that indicates the received signal strength in dBm.
- **LQI.** The Link Quality Indicator can be viewed as the chip error ratio. It is calculated over 8-bits following the Start Frame Delimiter (SFD). The LQI values are usually between 110 and 50 and correspond to maximum and minimum quality frames respectively.
- **PS.** The Packet Size is the sum of the payload size and the sizes of headers in the MAC and link layers.
Figure 40: Relation of the measured parameters RSSI, LQI and PRR at varying the distance between sender and receiver

Figure 40 presents the values of RSSI, LQI and PRR at varying the sender-receiver distance. As we can see, the RSSI and LQI decrease progressively when the distance increases. However, the PRR obtains almost 100% in distances shorter than 44 meters and decreases significantly in distances from 44 to 52 meters. For distances longer than 52 meters, the PRR is always zero.

Figure 41: PRR values at varying the packet size in distances between 44 and 51 meters

To better understand the results, the relation between the PRR and PS in distances between 44 and 51 meters is shown. Each curve represents the PRR obtained at each distance by increasing the PS from 10 to 100 bytes. For each distance, the increment of PS decreases the PRR value. As we anticipated, the results demonstrate that bigger packets have less probability of being received than smaller ones.
Based on these conclusions, we propose SERE with two reliable routing discovery schemes for geographic data forwarding. Instead of using small control messages, each node broadcasts big messages with the maximum transmission size to discover its neighbouring routers. Bigger messages are more error-prone than shorter messages. Thus, only neighbours receiving the big message participate in the forwarding process. These schemes permit to select neighbours as next-hops providing high reception probability. The design of our SERE protocol is presented below.

### 7.3.2 Design of SERE Routing Protocol

SERE applies a scalable and localised geographic routing approach based on the positions of nodes. To determine its position, each node may employ an existing positioning technique based on extra hardware (i.e., low-power GPS), distributed algorithms (i.e., DPE, APS, RPE), virtual coordinates or pre-established configuration. In geographic approach, every routing packet contains the position and identifier of the destination. So, routing decisions are taken according to the positions of the destination, the current router and its 1-hop neighbouring routers. Each router knowing its 1-hop neighbouring routers is able to decide independently the next hop. For this localised design, SERE provides a fully-distributed way for routing decisions. This geographic approach requires a minimum state in each router that stores only 1-hop neighbours positions. Thus, SERE scales perfectly with the number of nodes of the whole network.

![Example of an IoT6 network with smart objects, wireless routers and a half-gateway](image)

Based on the experimental results of realistic wireless communications, SERE proposes two scalable and reliable schemes to discover neighbouring routers based on ICMPv6 messages. In particular, smart objects and wireless routers employ routing solicitations and routing advertisements, respectively. Both ICMPv6 messages discover 1-hop neighbouring routers.
and are not forwarded in order to maximise the scalability. Both messages are sent inside the maximum transmission frame (i.e. 127 bytes in IEEE802.15.4). Since our analysis shows that higher frame size are more of a probability due to reception failure than lower ones. Unlike SERE, the traditional routing mechanism employs small control messages for discovering routers and may select a router whose reception probability of a bigger data packet may be very low. Thus, SERE discovery schemes permit routers with weak links that may cause retransmissions, hop-by-hop delay, and data packet losses during the data communication to be discarded. In the following, the operation of these discovery schemes is described.

Smart objects employ a reactive energy-efficient scheme to select its best neighbouring router. The reactive scheme follows a duty-cycle schedule to keep smart objects in sleep mode for the most time in order to minimise the power consumption. When a smart object wakes up to transmit a data message, it first broadcasts a routing solicitation message to discover its 1-hop neighbouring routers. Then, the smart object waits for the routing advertisement of its reachable routers. In SERE, only routers receiving the routing solicitation respond with a Routing Advertisement for the specific smart object. When its timer expires, the smart object selects the next-hop among the responses of neighbouring routers according to a specific metric. For instance, a power-aware metric chooses the router whose response has the highest radio signal strength (RSSI) in order to reduce the power transmission of the smart object. Once the router is selected, the smart object is able to transmit data. This reactive scheme discards unreachable routers that may generate unidirectional links and data retransmissions. In addition, this scheme guarantees that the selected router is available during the transmission time of the smart object. Both reactive discovery and power-aware selection enable a reliable data communication at the same time maximising the battery autonomy.

Wireless routers use a proactive discovery scheme to obtain the position information of 1-hop neighbouring routers for the forwarding process. Every router broadcasts periodically a big advertisement message including its identifier and position to inform its 1-hop neighbours. Thus, only each router receiving the big beacon message stores the sender as reliable forwarding candidate. So, only reliable routers participate in the forwarding of data packets. SERE enables to discover routers providing high reception probability in order to achieve a reliable data forwarding process.

Knowing the neighbourhood positions, wireless routers can forward data packets by a geographic next-hop selection. The next-hop selection combines greedy and face strategies based on Greedy Routing Scheme (GRS) and Face-2, respectively. First, the current router holding the packet applies GRS scheme to select the closest neighbour toward the destination in order to maximise the advance and reduce the number of hops. The GRS strategy is employed until reaching the destination or a local maximum router without neighbours providing advance. If a local maximum is reached, the position of the maximum router is
stored in the packet which is forwarded in face mode. To do face forwarding, in each hop the current router constructs a planar sub-graphs of neighbourhood topology by the GG algorithm. Face-2 algorithm is used till finding a wireless router closer to the destination than the local maximum where the greedy mode is resumed. This forwarding process is repeated until the packet reaches the destination. In SERE, both greedy and face strategies provide scalable and robust forwarding decisions.

In realistic wireless network, the forwarding process is damaged severely by collisions, interferences and packets losses. To provide failure tolerance and path recovery, the forwarding process of SERE includes two simple mechanisms: acknowledgement and retransmissions. During the forwarding process, a router needs the confirmation of the reception of a data message which may be lost. To do that, two different techniques are used: passive acknowledgement (PACK) and an active one (ACK). The use of ACK introduces a message in the forwarding process incrementing the protocol overhead. Thus, each router overhears the transmission of data message from its selected next hop as PACK to confirm the data reception. The ACK is always needed when the data message arrives to the destination. On the other hand, when the current router of the data packet does not receive a PACK or an ACK, it retransmits the data packet up to a maximum of 3 times. After that, the current router repeats the next-hop selection according to greedy and face strategies. For the new next-hop, the current sender starts again the forwarding process. However, the experimental results show that the SERE design based on big routing discovery messages discards unreachable routers. For this reason, the probability of needing retransmissions is very low. In SERE, these two simple mechanisms (passive acknowledgement and next-hop re-selection) enable to support failure tolerance and path recovery with minimal transmission overhead.

Pull communication of query messages can be supported from Internet to IoT6 network. In SERE, the half-gateway collects automatically the topology of the IoT6 network. To do that, when the half-gateway receives a packet (i.e. mDNS) from a smart object, its identifier and position (included as source node) are stored. The half-gateway stores the information of smart object to map identifiers and coordinates. Once, the half-gateway receives a query message to a smart object, the half-gateway incorporates its position as the destination of the message to enable the geographic routing. This dynamic scheme permits efficient pull communication avoiding the path discovery overhead of control messages in the whole network.

To permit peer to peer communication between smart objects, SERE can provide a location service mechanism. When a source smart object wants to communicate to a destination whose position is unknown, the source sends a location request with the destination identifier to the half-gateway. The half-gateway responds to a message with the destination position to the source. Once, the source receives the responses with the destination position, the peer-to-peer communication is available. This simple mechanism is only needed the first time to discover the destination position avoiding routes tables maintenance in intermediate routers.

SERE distributes its limited overhead among all network nodes to support energy-efficient smart objects and full connectivity. Each router needs proactive transmissions to know local topology about the 1-hop neighbouring routers. Smart objects select the next-hop router using a reactive discovery avoiding the information storage of all neighbouring routers. Both proactive and reactive neighbourhood schemes support topological changes without the overhead of path repairing mechanisms. The half-gateway requires the information of smart objects to be saved. The storage size of the half-gateway is limited by the number of smart objects. In addition, wireless routers should store temporally messages toward neighbouring smart objects that are sleeping. Unlike the RPL standard, SERE avoids that path tables in intermediate routers from each pair of smart object have to be created and maintained. This
proposal also minimises the query losses of smart objects at the same time maximising their battery lifetime.

### 7.4 IPv6 transition mechanisms on gateways for the IoT

In section 3 the enhanced instance of the IoT6 network architecture view that relies on abstractions of gateway-based functions and which places these abstractions into cloud servers was reviewed. In this architectural view, ServiceNet to extend beyond the narrow meaning of a Local Area Network (LAN) into a larger domain where the Service is the main primitive (thus, one could refer to ServiceNet as the ‘Service Area Network’) was defined. Sensor and Actuator Services can be provided in some form of server/cloud infrastructure, which we have termed SAaaS (Sensors and Actuators as a Service).

Having reviewed the powerful sets of basic and advanced IPv6-based network mechanisms which combined, can lead to advanced network services, we concluded that there is an obvious advantage in using IPv6 as the preferred network protocol suite across all tiers of the IoT architecture.

We acknowledge, however, that many, if not all, of the services one could build with the IPv6 suite can also be made available in their IPv4 equivalents (possibly at reduced functionality). We further acknowledge the existence of multiple legacy DevNets and their corresponding control gateways, which hide the implementation and technology-specific details from the network core. It is common that almost all legacy DevNets are only IPv4-based. It is also common that several cloud service infrastructures may not support IPv6 (for example, the Amazon AWS cloud service). This poses the fundamental question of how best to support advanced IoT functions, if the core network does not support IPv6, or the DevNets are not IPv6-enabled.

In these cases, the use of various translation and transition mechanisms between protocol families and types of service is proposed. There is a rather long list of IPv4-IPv6 transition mechanisms in the literature (see [14] for a full list), which is not the main focus of this Deliverable. We highlight however, that the requirement of IoT systems scalability is highly related to (and relied upon) such mechanisms. We believe that translation mechanisms of benefit to the IoT can be categorised under the set of advanced IPv6 functions and it is envisaged that such mechanisms will become very relevant in the implementation of components of the enhanced IoT architecture in T2.4. A deeper analysis of the relevant features will then be provided.

### 8 Unicast/Multicast/Anycast for IoT6 Sensor Networks

In IoT6 networks, there are different traffic flows that require IPv6 advance features to provide quality of services (QoS). IPv6 features provide flow label and traffic class to manage the communication priority in each intermediate routers. For instance, a source node can establish a specific flow label in the IPv6 header of the packet sent to indicate the required QoS. Moreover, IPv6 features enable different addressing (unicast, multicast and anycast) to guarantee the traffic delivery to one or more destination nodes. In particular, the traffic flows of IoT6 networks are:

- Smart objects send periodical MDNS messages to register their services and resources in directory servers. These periodical MDNS messages need minimal priority in terms of delay and reliability. MDNS messages can be sent to a multicast address to allow the redundancy in the registering process.
- Smart objects send normal reports about the measurements of the environment. These information messages require normal priority. Normal data messages are transmitted to a monitoring server.
- When a smart object takes an abnormal measurement, it detects an alert situation and must communicate as fast as possible. An alarm message needs a high priority to reduce the hop-to-hop delay. Moreover, this alarm message must be transmitted to an anycast address to guarantee the delivery in some available control servers.

To provide QoS in IoT6 communications, two IPv6 techniques based on packet priority and destination redirecting are proposed. First, a smart object originating a packet establishes the traffic class to indicate the priority required. Second, half-gateways are able to manage the outgoing packets to IPv6 Internet for redirecting the destination address in order to support several delivery policies. Both techniques enable the packets in intermediate routes to be prioritised and guarantee the service delivery to different destination servers. In the following, both aspects will be described at the smart object side and within the half gateway.

### 8.1 Priority management

In IoT6 communications, smart objects that originate IPv6 packets can establish different traffic priorities. To specify a packet priority, smart objects employ the 8-bits Traffic Class field in the IPv6 header. The traffic class identifies a sequence of packets originated from any source that needs a special routing priority until reaching the destination. Its default value is zero for all of the 8 bits. Only smart objects that support packet priority can change the value of the Traffic Class bits.

![Figure 44: The 8-bit field of traffic class in IPv6 header](image)

In IoT6 networks, three IPv6 messages that need QoS (periodical mdns, normal data, and alarm report) are determined. A QoS policy to mark each IPv6 message with appropriate priority values is proposed. The priority values can be chosen from the range 1 to FF hex. The following table shows the used priority values. These priority values are recommended for IPv6 protocol for Internet communications. Packets that do not belong to this table have a traffic class of zero. A simple priority policy (no more than about three classes) is used; it will be easier to manage for Internet routers. This simple policy enables traffic originated by smart objects on the rest of the IPv6 network segments to be classified.
### Table 7: Traffic classes for packet prioritisation

<table>
<thead>
<tr>
<th>Traffic Class</th>
<th>Messages</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>0x1</td>
<td>MDNS registration</td>
<td>Low priority for &quot;Filler&quot; traffic</td>
</tr>
<tr>
<td>0x6</td>
<td>Normal data</td>
<td>Medium priority for &quot;Interactive&quot; traffic</td>
</tr>
<tr>
<td>0x7</td>
<td>Alarm report</td>
<td>High priority for “Internet control” traffic</td>
</tr>
</tbody>
</table>

Packets marked with non-zero Traffic Class has a special handling by the intervening routers. The nature of the special handling might be conveyed to the routers by a control protocol. The control protocol may decide how to queue the packet. The decision to queue the packet is based on the Traffic Class field of the packet. Internet routers often support class-based queuing for IPv6 packets. In those cases, a router does not support the functions for packet prioritisation. This router ignores the value of the Traffic Class field and forwards the packet without class-based queuing.

While the DiffServ network architecture recommends a standardised set of traffic classes, it does not incorporate predetermined universal values of what types of traffic should be given priority treatment.

The above approach by IoT6 priority management fully respects the DiffServ network architecture and should be interoperable with any system implementing the same priority values of Table 7. As long as each router (and IoT6 gateway) follows the above traffic class values of Table 7, the priority requirements of individual flows are respected, regardless of Use Case or Scenario. Each traffic class is managed differently by the network infrastructure, ensuring preferential treatment for higher-priority traffic on the network.

### 8.2 Addressing redirection.

IPv6 addressing enables 128-bit identifiers for interfaces and sets of interfaces [78]. IPv6 protocol defines three types of addresses: unicast, anycast and multicast. A unicast address identifies a single interface to establish point-to-point communications. Packets sent to a unicast address are delivered to the destination node identified by that address. A multicast address identifies a set of interfaces belonging to different nodes to permit point-to-multipoint communications. Packets sent to a multicast address are delivered to all destination nodes identified by that address. An anycast address identifies a set of interfaces belonging to different nodes to allow point-to-point-of-multipoint communications. Packets sent to an anycast address are delivered to one of the destination nodes identified by that address (the “nearest” one, according to the routing protocols' measure of distance).

IoT6 applications employ these three destination addresses to deliver different traffic flows. First, normal data must be sent to a unicast address to reach a particular monitoring server for storing the information of all sensor devices. Second, MDNS registers must be sent to a multicast address to reach a group of resource directories for saving the resources and services of the local sensor network. Third, alarm reports must be delivered to an anycast address to reach the nearest host of a set of control servers for triggering the alert mechanisms and informing the security staff. These three traffic flows are shown in Figure 45.
Regarding these three traffic flows, a destination redirection technique from sensor objects to Internet hosts through the half-gateway is proposed. Smart objects only need to know the unicast address of the half-gateway to reduce their computation overhead. Smart objects can obtain the IPv6 address of the half-gateway using routing advertisement and routing solicitation. The half-gateway is responsible to redirect the packets to the correct destination according to the contents of the packets. In IoT6 environment, the contents of the packets (i.e. MDNS, data and alarm) are identified by the priority values of traffic class field. Therefore a half-gateway receiving a packet can process the traffic class field in order to identify the destination address. To do that, the half-gateway only needs a small table to map each traffic class to a particular IPv6 address. This technique enables easy-maintenances of IPv6 servers addressing the update in order to reduce overhead of all smart objects in a sensor network.

9 Conclusions and future work

We have shown the importance of using IPv6 features in the IoT environments being considered, but have outlined some significant alternative implementations which provide many significant advantages. The principal components of single gateways for IoT have been defined and implemented. We have then discussed how distributed gateway functionality can be developed – which maintains all the advantages of the IPv6 facilities – and provides a better system security that would normally be practical in a constrained resource gateway.

Extensive detail of how the relevant IPv6 features can be implemented in specific environments has been presented. We have shown also that while certain features of IPv6 are desirable in principle, their use is not always feasible in gateways with constrained resources for the IoT as shown in the examples in the use of IPsec and MIPv6 in some environments.

In on-going work, we expect to provide the missing facility of access to advanced
repositories. We will then try to provide IPv6 facilities in a gateway distributed between a
gateway device and auxiliary functions on other servers. We will then quantify the differences
experienced from the two approaches.
10 References

European Project 287901, IERC European Research Cluster (2013).


[60] Bag, G., Shams, S.M.S., Akbar, A.H., Raza, H.M.M.T., Ki-Hyung Kim, Seung-Wha


[137] Universal Device Gateway Project, www.devicegateway.com