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: October 1st, 2011 (36 months duration)

**Deliverable D6.2**
Ubiquitous access and mobile phone network interactions report

<table>
<thead>
<tr>
<th>Contract Due Date</th>
<th>30/09/2013</th>
</tr>
</thead>
<tbody>
<tr>
<td>Submission Date</td>
<td>30/09/2013</td>
</tr>
<tr>
<td>Version</td>
<td>1.0</td>
</tr>
<tr>
<td>Responsible Partner</td>
<td>Ericsson (EYU)</td>
</tr>
<tr>
<td>Author List</td>
<td>Srdjan Krco, Boris Pokric, Dejan Drajic, Tomislav Dimcic</td>
</tr>
<tr>
<td>Dissemination level</td>
<td>PU</td>
</tr>
<tr>
<td>Keywords</td>
<td>Internet of Things, IPv6</td>
</tr>
</tbody>
</table>

Project Coordinator: Mandat International (MI)
Sébastien Ziegler sziegler@mandint.org
Table of Contents

1  Introduction ....................................................................................................................................... 7
   1.1. Purpose and scope of the document ..................................................................................... 7
   1.2. Task T6.1 ............................................................................................................................... 7
   1.3. Structure of the document .................................................................................................... 7

2  Integration of LTE mobile devices with IoT6 ........................................................................ 9
   2.1. LTE Network in IoT6 Architecture .................................................................................... 9
   2.2. Mobile phones (IoT6 devices) as the Gateways in IoT6 ..................................................... 10
   2.3. Mobile phones (IoT6 devices) registration to LTE Network ............................................. 11
   2.4. Principles of integration LTE networks with the IoT6 architecture model ....................... 13

3  Resource and service discovery ....................................................................................................... 15

4  Experiments setups and use cases ............................................................................................... 16
   4.1. A Comparison of Discovery and Resource Directory approaches ...................................... 16
       4.1.1. Measurement setup and Testbed description ................................................................. 16
       4.1.2. CoAP RD communication .............................................................................................. 18
       4.1.3. Discovery communication ............................................................................................. 19
       4.1.4. Communication time comparison ................................................................................... 22
       4.2. Mobile Phones with its own sensors ............................................................................... 25
       4.3. Half Gateway implementation .......................................................................................... 28
       4.4. Implementation of mDNS on a mobile phone (Android based) ........................................ 31
       4.5. Implementation of mDNS when mobile phone acts as a mobile hotspot ....................... 34
       4.6. Testing of global publishing/browsing techniques ........................................................... 36

5  Conclusions and recommendations for further integration of mobile phones within the IoT6 platform ....................................................................................................................................... 38

6  References ...................................................................................................................................... 41
List of Tables

Table 1: Comparison of Digcovery and RD ................................................................. 23
Table 2: Comparison of Digcovery and RD without the tunnelling for IPv6 .......... 24

List of Figures

Figure 1: IoT6 communication channel model ......................................................... 9
Figure 2: Device Clusters in IoT6 architecture ......................................................... 10
Figure 3: High level LTE architecture ...................................................................... 11
Figure 4: Measurement setup for comparison of Digcovery and Resource Directory .... 17
Figure 5: Communication flow – sending data from MindWave to IoT LED Lamp ........ 18
Figure 6: Communication flow – sending data from MindWave to IoT LED Lamp with IPv6 enabled devices ................................................................. 21
Figure 7: Communication flow – Resource Directory / Digcovery response time comparison .......................................................................................................................... 22
Figure 8: Communication time comparison (Digcovery vs. RD) .............................. 23
Figure 9: Test set-up, IPv6 communication between laptop and CoAP Server on the Android ......................................................................................................................... 26
Figure 10: Communication flow – getting the list of available sensors from the Android phone ......................................................................................................................... 28
Figure 11: IPv6 communication between laptop and MindWave device using Android phone as a Gateway ......................................................................................................................... 30
Figure 12: Communication flow – getting the attention or meditation level from the MindWave device ......................................................................................................................... 31
Figure 13: Experiment setup ........................................................................................ 32
Figure 14: Result of mDNS query in .local domain from Android mobile phone ....... 33
Figure 15: Result of mDNS query in .local domain from VM Avahi Discovery ............ 34
Figure 16: Experiment setup for mobile phone acting as a mobile hotspot .............. 35
Figure 17: Service browsing from Lab2 ........................................................................ 35
D6.2 Ubiquitous access and mobile phone network interactions

Figure 18: Service browsing from mobile phone ................................................................. 36

Figure 19: Overview of the mDNS / DNS-SD Connector, showing how to publish globally the resources and devices found on each local network ......................................................... 37
List of Acronyms

<table>
<thead>
<tr>
<th>Acronym</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>3GPP</td>
<td>3rd Generation Partnership Project</td>
</tr>
<tr>
<td>APN</td>
<td>Access Point Name</td>
</tr>
<tr>
<td>CoAP</td>
<td>Constrained Application Protocol</td>
</tr>
<tr>
<td>DHCP</td>
<td>Dynamic Host Configuration Protocol</td>
</tr>
<tr>
<td>DHT</td>
<td>Distributed Hash Tables</td>
</tr>
<tr>
<td>DNS</td>
<td>Domain Name Server</td>
</tr>
<tr>
<td>DNS - SD</td>
<td>Domain Name Server – Service Discovery</td>
</tr>
<tr>
<td>EEG</td>
<td>Electro Encephalograph</td>
</tr>
<tr>
<td>eNodeB</td>
<td>Enhanced Node B</td>
</tr>
<tr>
<td>EPC</td>
<td>Evolved Packet Core</td>
</tr>
<tr>
<td>EPS</td>
<td>Evolved Packet System</td>
</tr>
<tr>
<td>HGW</td>
<td>Half Gateways</td>
</tr>
<tr>
<td>HPLMN</td>
<td>Home Public Land Mobile Network</td>
</tr>
<tr>
<td>HSS</td>
<td>Home Subscriber Server</td>
</tr>
<tr>
<td>HTTP</td>
<td>Hypertext Transfer Protocol</td>
</tr>
<tr>
<td>IoT</td>
<td>Internet of Things</td>
</tr>
<tr>
<td>IPv6</td>
<td>Internet Protocol Version 6</td>
</tr>
<tr>
<td>ISP</td>
<td>Internet Service Provider</td>
</tr>
<tr>
<td>IT</td>
<td>Information Technology</td>
</tr>
<tr>
<td>LAN</td>
<td>Local Area Network</td>
</tr>
<tr>
<td>LED</td>
<td>Light Emitting Diode</td>
</tr>
<tr>
<td>lmDNS</td>
<td>Lightweight mDNS</td>
</tr>
<tr>
<td>LTE</td>
<td>Long Term Evolution</td>
</tr>
<tr>
<td>M2M</td>
<td>Machine to Machine</td>
</tr>
<tr>
<td>mDNS</td>
<td>Multicast Domain Name Server</td>
</tr>
<tr>
<td>MME</td>
<td>Mobility Management Entity</td>
</tr>
<tr>
<td>MT</td>
<td>Mobile Terminal</td>
</tr>
<tr>
<td>NAT</td>
<td>Network Address Translation</td>
</tr>
<tr>
<td>NAT</td>
<td>Network Address Translation</td>
</tr>
<tr>
<td>OSA</td>
<td>Open Service Architecture</td>
</tr>
<tr>
<td>PCRF</td>
<td>Policy Control and Charging Rules Function</td>
</tr>
<tr>
<td>PDN</td>
<td>Packet Data Network</td>
</tr>
<tr>
<td>P-GW</td>
<td>Packet Data Network Gateway</td>
</tr>
</tbody>
</table>
### D6.2 Ubiquitous access and mobile phone network interactions

<table>
<thead>
<tr>
<th>Acronym</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>RD</td>
<td>Resource Directory</td>
</tr>
<tr>
<td>REST</td>
<td>Representational state transfer</td>
</tr>
<tr>
<td>SaaS</td>
<td>Software as a Service</td>
</tr>
<tr>
<td>S-GW</td>
<td>Serving Gateway</td>
</tr>
<tr>
<td>SPI Bus</td>
<td>Serial Peripheral Interface Bus</td>
</tr>
<tr>
<td>STIS</td>
<td>Smart Things Information System</td>
</tr>
<tr>
<td>TE</td>
<td>Terminal Equipment</td>
</tr>
<tr>
<td>UE</td>
<td>User Equipment</td>
</tr>
<tr>
<td>WLAN</td>
<td>Wireless Local Area Network</td>
</tr>
<tr>
<td>VM</td>
<td>Virtual Machine</td>
</tr>
<tr>
<td>VPLMN</td>
<td>Visited Public Land Mobile Network</td>
</tr>
</tbody>
</table>
1 Introduction

1.1. Purpose and scope of the document

The IoT6 research project aims at researching and exploiting the potential of IPv6 to develop a service oriented architecture overcoming the current Internet of Things fragmentation.

The purpose of this deliverable is to document the activities and outcomes of Task T6.1: Ubiquitous access and mobile phone network interactions. In this document, we address the possibilities of integrating a LTE (Long Term Evolution) mobile network with the IoT6 architecture, and the feasibility of integrating Digcovery with LTE. The usage of the Resource Directory (RD) and Digcovery to discover resource/services are explored and compared. Specifically, the document describes and presents the results of experiments for selected use cases. Special attention was dedicated to the integration of sensors available on mobile phones with the IoT6 architecture, on which all the experiments and measurement setups are based.

1.2. Task T6.1

Task T6.1 researches and explores various ways to integrate an IPv6-based Internet of Things into mobile phone networks, enabling mobile devices to provide access to smart objects as well as to use mobile devices as sensors/actuators.

The integration of the IPv6-based Internet of Things into mobile phone networks is based on the IoT6 architecture, and has to fulfill the following requirements:

- Enable mobile phones to provide ubiquitous access to smart things connected to the developed architecture
- Enable smart things or systems connected to a IoT6 system to connect and send messages to the mobile phone
- Enable IoT6 systems to use mobile phones as mobile sensing tools, and to retrieve information from the mobile phones, such as temperature, motion, localization, etc.

The goal of the Task is to propose the best option to implement and test the above-mentioned forms of interactions between mobile phones and the IoT6 architecture.

1.3. Structure of the document

The document is organized as follows: the integration of LTE network with the proposed IoT6 architecture is explained in Section 2, together with an analysis of the barriers, challenges and benefits of the integration. A feasibility study and analysis of Digcovery integration with LTE is also presented. In section 3, resource and service discovery are analyzed, namely the usage of RD and Digcovery to discover resources/services. A
D6.2 Ubiquitous access and mobile phone network interactions

comparison of these two approaches is outlined. Section 4 presents selected scenarios and use cases, together with the results and detailed analysis of experiments. Finally, in Section 5, the conclusions and recommendations for further integration of mobile phone within the IoT platform are given.
2 Integration of LTE mobile devices with IoT6

The scope of this section is to analyze the inclusion of LTE devices (mobile phones) into the adopted IoT6 architecture model from [1], [2], [3]. The section is organized as follows: the position of LTE networks in the IoT6 architecture is explained; the addressing of mobile devices is analyzed and their roles as Gateways and Half Gateways as defined in the IoT6 architecture are explained; the principles of integrating LTE network with the IoT6 architecture model are analyzed and a feasibility study of the integration of LTE mobile networks and Digcovery is explored.

2.1. LTE Network in IoT6 Architecture

For the sake of completeness, a figure from [2] is presented in order to give a global picture of the IoT6 architecture (Figure 1). Detailed definitions and explanations are given in [2], however, only the related network elements are addressed here.

![Figure 1: IoT6 communication channel model](image)

At the high level, the IoT6 architecture consists of three distinct network domains. The first domain is the general Internet wide-area domain, which contains the future Internet core network and its associated components that access this network, like mobile networks, monitoring applications, cloud computing (SaaS), user interfaces and information systems. The second domain covers the IoT intranet (IPv6 local network) containing the IPv6 Services, routers and Gateways used to provide access to sensor clusters. The third domain is the sensor network domain that consists of smart things with associated sensors connected using a variety of protocols (IP or non-IP based).

Following the IoT6 definitions, a device represents a physical component i.e. hardware, with communication capabilities for interaction with other systems. Devices can be either attached to - or embedded inside - a physical entity, or monitor a physical entity in the vicinity.
Devices can belong to one of the following clusters: M2M cluster, other clusters, large IPv6 clusters and small IPv6 cluster (Figure 2).

A large IPv6 cluster represents a device (sensor) network where the multicast traffic is considered inefficient for direct Service Discovery (SD). Therefore, it is better to employ the DNS-SD methodology, where an additional DNS server is placed within the network, serving as a local database with resource records structured as per DNS-SD convention.

A small IPv6 cluster contains a small number of devices (sensors) and has an efficient multicast infrastructure, where it is possible to implement a direct Service Discovery using mDNS (multicast DNS), or lmDNS (lightweight mDNS) only.

The “other” cluster contains the IoT6 non-compliant devices based on other, non-IPv6 communication protocols, including IPv4-based devices. The IoT6 non-compliant devices, require Gateways or proxies to be connected to the rest of the IoT6 system in order to adapt the native protocols, functionality and addressing to IPv6 through a transparent mechanism.

2.2. Mobile phones (IoT6 devices) as the Gateways in IoT6

Gateways are an important component of the IoT6 architecture model, providing a number of different functionalities as indicated in [1], [2], [3]. Within the local network (Intranet), all the local Gateways are IPv6-compliant and provide functionalities such as the discovery of services and resources, protocol adaptations (e.g. HTTP to CoAP, IPv4 to IPv6), smart routing and security and privacy aspects. Gateways are also used to keep track of IoT6 devices that are reachable via Gateways. They also translate the incoming/outgoing messages or data into an internal format of the connected device. Half Gateways (HGW) provide more specific functionalities to the sensor networks in terms of protocols and addressing, taking
D6.2 Ubiquitous access and mobile phone network interactions

into account IPv6-based as well as legacy devices (IPv4 and non-IP devices).

A mobile device with IPv6 functionality, including an LTE one, can represent a Gateway for both small and large clusters of IoT IPv6 devices (sensors). Devices in the clusters are obtaining IP addresses independently from the mobile network. In addition, devices could belong to other clusters, which contain non-IP devices and sensors with IPv4 private addresses. For non-IPv6 based devices, mobile devices can acts as Half Gateways and map their physical layer identifiers to IP addresses.

The implementation of mobile phone as a Half Gateway is evaluated in section 4.3. In section 4.5 the implementation of mDNS service publishing/browsing in the mobile network, when a mobile phone serves as a mobile hotspot is described.

2.3. Mobile phones (IoT6 devices) registration to LTE Network

LTE [6] is a global standard for mobile networks defined by 3GPP (3rd Generation partnership project) for the fourth generation of mobile broadband. A brief overview of the LTE architecture, with the main entities and protocols is given in Figure 3.

![Figure 3: High level LTE architecture](image)

The main entities in a LTE network and their corresponding functionalities are the following:

- **UE** stands for User Equipment (end user device)
- **MME (Mobility Management Entity)** is the control node that processes the signalling between an UE and the core network, i.e. it handles the mobility and the session management functions
- **eNodeB (enhanced Node B)** is responsible for all radio-related functions.
- **HSS (Home Subscriber Server)** contains users’ subscription data and information about the PDNs (Packet Data Networks) to which the user can connect. The HSS also holds dynamic information such as the identity of the MME to which the user is currently attached or registered
- **P-GW (Packet Data Network - PDN Gateway)** is responsible for IP address allocation for the UE, as well as QoS enforcement and flow-based charging according to rules from the PCRF
S-GW (Serving Gateway) – All user IP packets are transferred through the Serving Gateway, which serves as the local mobility anchor for the data bearers when the UE moves between eNodeBs. PCRF (Policy Control and Charging Rules Function) is responsible for policy control decision-making, as well as for controlling the flow-based charging functionalities.

More information about LTE networks can be found in the LTE related 3GPP recommendations [6].

Next we will explain and analyze in more details the registration procedure of a mobile device to a LTE network; i.e. the attach procedure. To announce its presence in the network, the UE must initiate an attach procedure by signalling to the MME. As a result of the attach procedure, a mobility context exists in the MME and one default bearer is established between the UE and the PDN Gateway, i.e. one PDN connection is created.

During the registration procedure, LTE mobile phones (IoT6 devices or in LTE terminology UE-User Equipment) are obtaining IP addresses from the LTE network. The assignment of IP addresses is performed by the EPC (Evolved Packet Core) core network. The IP address allocation to terminals in the 3GPP Release 10 is explained in detail in the corresponding 3GPP recommendation [7]. A UE (LTE device) shall perform the address allocation procedures for at least one IP address (either IPv4 address or IPv6 prefix) after the default bearer activation. One of the following ways shall be used to allocate IP addresses for the UE:

- The Home Public Land Mobile Network (HPLMN) allocates an IP address to the User Equipment (UE) when the default bearer is activated (dynamic or static HPLMN address)
- The Visited Public Land Mobile Network (VPLMN) allocates an IP address to the UE when the default bearer is activated (dynamic VPLMN address)
- The Packet Data Network (PDN) operator or administrator allocates an (dynamic or static) IP address to the UE when the default bearer is activated (External PDN Address Allocation)

Thus, if an UE is in its home or a visited network, then HPLMN or VPLMN will allocate an IP address to the UE through regular LTE procedures, while in the third case the PDN operator/administrator will allocate IP addresses from its own pool of addresses. Addresses can be static or dynamic. The static addresses should be defined and saved on the HSS as one of the UE subscription parameters and retrieved from the HSS during the registration process. Dynamic addresses are assigned from the available DHCPs pools.

The IP address allocated to the default bearer of an UE shall be also used for the dedicated bearers within the same PDN connection. A PDN connection consists of one default bearer and, depending on the service for which the connection is used, a number of dedicated bearers. The IP address allocation for PDN connections, which are activated by the UE requesting PDN connectivity procedure is handled with the same set of mechanisms as those used within the attach procedure (procedure for registering with the network). The purpose of...
the PDN connectivity procedure is to set up a default Evolved Packet System (EPS) bearer between a UE and a packet data network.

PDN types IPv4, IPv6 and IPv4v6 are supported. An EPS Bearer of PDN type IPv4v6 may be associated with one IPv6 prefix only or with both one IPv4 address and one IPv6 prefix. PDN type IPv4 is associated with an IPv4 address, while PDN type IPv6 is associated with an IPv6 prefix. PDN types IPv4 and IPv6 are utilized for the UE and/or the PDN Gateway supporting IPv4 addressing only or IPv6 prefix only. It can be also the case that the operator preferences dictate the use of a single IP version only. In addition, the subscription can be limited to IPv4 only or IPv6 only for a specific APN (this is controlled by the operator). In addition, PDN type IPv4 and IPv6 are utilized for interworking with nodes of earlier releases (Releases before 3GPP Release 10 [6]).

The way that an UE sets the requested PDN type may be pre-configured in the device per Access Point Name (APN). Unless otherwise configured, the UE sets the PDN type during the attach or PDN Connectivity procedures, based on its IP stack configuration as follows:

- A UE, which is IPv6 and IPv4 capable, shall request for PDN type IPv4v6
- A UE, which is only IPv4 capable, shall request for PDN type IPv4
- A UE, which is only IPv6 capable, shall request for PDN type IPv6
- When the IP version capability of the UE is unknown in the UE (as in the case when the Mobile Terminal (MT) and Terminal Equipment (TE) are separated and the capability of the TE is not known in the MT), the UE shall request for PDN type IPv4v6

2.4. Principles of integration LTE networks with the IoT6 architecture model

In this section, the principles of integrating LTE networks with the IoT6 architecture model are analyzed in detail. LTE provides connectivity to IoT6 devices and make them available to the rest of the IPv6-enabled environment in the sense of discovery, access and management. Based on the described procedure in section 2.3, LTE device (i.e. a mobile phone acting as an IoT6 device, an IoT6 Gateway or an IoT6 Half Gateway) will obtain an IP address from a LTE network. It is up to the operator to decide and define if addresses for LTE devices/gateways will be assigned to a separate APN or LTE device/Gateway and whether it will be treated as a regular LTE user.

In the scope of the IoT6 architecture, a LTE network represents the part that enables communication of IoT6/M2M devices with other parts of the system via a LTE access network. The devices can be connected directly or via a Gateway.

A mobile phone itself can be an IoT6 device. In addition, there can be one or more devices connected to a mobile phone, in which case the mobile phone acts as a Gateway or Half Gateway in the IoT6 architecture (more details can be found in section 4.3.). Devices (mobile
phones) can have embedded sensors for temperature, motion, localization etc., or other external smart objects can be connected via Bluetooth, Infrared, etc.

LTE enables the deployment of new IoT6 devices and their subsequent software upgrades. New IoT6 resources that are connected to the network can register themselves in an IoT6 resource repository (that maps an identifier of IoT6 device to its address), that should be globally accessible from IoT6 applications.

In addition, LTE supports the transmission of other commands like activation, update, deactivation, reactivation of resources on IoT6 devices. QoS management is also provided within LTE networks that can be used for urgent traffic flows.

One of the main goals of the IoT6 project is to allow the autonomous registration and discovery of resources and services. At the moment, the most widely used discovery mechanism for the Internet is DNS. Nowadays, extensions of DNS in the form of mDNS and DNS-SD (DNS Service Discovery) are developed that can be used for resource and service discovery functionalities instead of the standard approach. These extensions allow to query and discover services by types and properties. However, there is no complete architecture, which manages global discovery. A global discovery architecture interoperable with DNS called Digcovery [3], [8] is proposed and presented in the project. More details about mDNS and DNS-SD mechanisms are explained in Section 3. For local self-discovery, the mDNS protocol is used based on multicast messages and commercial solution Avahi for Linux based platforms. There is also commercial solution Bonjour for Apple products for MAC OS.

In this section, we have provided a comprehensive study of integration possibilities of LTE devices in IoT6. Different aspects of integration have been considered. There are limitations that mainly depend on the approach of the mobile operator, i.e. until which extent they intend to support IPv6 in their networks. A good overview of LTE networks commercially launched worldwide or have made a commitment to it can be found in [24]. Commitment levels include ongoing LTE trials, intentions to trial, deploy, migrate, etc.
3 Resource and service discovery

In this section, resource and service discovery in IoT6 is discussed. According to the IoT6 architecture, descriptions of the resources, including those hosted by a device (UE in LTE), are stored in a Resource Directory (RD). The RD represents common storage that hosts descriptions of resources held on other servers, allowing lookups to be performed for those resources. Description of one implementation of the RD can be found in [9], where resources of ecoBus applications are register in the RD.

In order to simplify the architecture and procedures in IoT6, we utilize properties of the IPv6 protocol and re-use them within the architecture model, replacing some of the standard components. For example, parts of the service and resource discovery functionality can be replaced with the DNS-SD and mDNS based IPv6 functionalities. In this case, the discovery of services is conducted through IPv6 network-based information systems that are already deployed, such as the Domain Name System with Service Discovery (DNS-SD). Following the same approach, a Resource Directory serving a local level could be replaced with multicast DNS (mDNS), thus providing the required functionality by exploiting and extending the IPv6 functions only. More about mDNS and DNS-SD can be found in [3].

In order to find devices and resources outside of the local network, global look-up discovery mechanisms are needed for IoT6 and a global service discovery architecture is required in order to manage the different domains for a single management system. For this purpose, the Digcovery core management system [8], which is global discovery platform, is a selected for the discovery. This platform is used to locate the different domains and the widely deployed directories with the different resources. Digcovery is based on DNS (dig command in Linux OS). Digcovery is public and accessible from anywhere through digcovery.net. Digcovery allows the delegation of each domain to the end-user and/or service provider through digrectories. A more detailed description can be found in [3].

Our work is focused on resource and service discovery in small IoT6 device networks where different type of sensors/networks can be connected to a mobile device. A mobile device can have its own sensors (embedded) or different sensors can be connected wirelessly (for example via Bluetooth) or using wires (for example via an USB connection). These external sensors can be connected to a mobile phone for a longer periods of time (the user is carrying them around, an example is personal health sensors) or can connect opportunistically when a mobile device comes to the range (for example sensors deployed in the environment to monitor various parameters).
4 Experiments setups and use cases

In this section, the results and a detailed analysis of the executed experiments are presented and discussed. The scope of the tests was to provide a proof of the concept on the integration of sensors available on mobile phones within the IoT6 architecture and for the service discovery using DNS-SD and mDNS within an IPv6 network.

Different forms of interactions between mobile phones and the IoT6 architecture were explored. Two groups of tests were run; one is related to the integration between a RD and Digcovery (4.1 – 4.3), while tests 4.4. to 4.6, are related to the implementation of mDNS and DNS-SD protocols for resource and service discovery of services offered by mobile phones. It is shown how IoT6 systems can use mobile phones as mobile sensing tools, and to retrieve information from the mobile phones, and how mobile phones can provide access to smart things connected to the developed architecture.

In Serbia, since there is still no mobile operator that offers IPv6 connectivity, it was not possible to perform tests where the phones obtain IPv6 address from the mobile network, but instead IPv6 tunnelling via a WiFi connection was used.

4.1. A Comparison of Digcovery and Resource Directory approaches

The goal of this test case was to measure performance and compare implementations of a Digcovery system and the Resource Directory approach. Although in the conducted test Resource Directory worked over the IPv4 network, it is significant to compare the existing approach with the new proposed Digcovery system that works over IPv6. The focus of the test was on evaluating the benefits of one over the other. To do a comparison, the testbeds were set up in the same manner. This means that the local installations were the same, only the resource discovery mechanism was different.

4.1.1. Measurement setup and Testbed description

The IoT LED (Light Emitting Diode) Lamp used in this experiment is a smart lamp connected to the Internet and designed to be able to adapt to the environment in the room or to visualize some interesting and fun symbols, like smiley face, heart, space invaders, etc. The lamp was originally developed and used in the demonstration of the FP7 HOBNET project. Here, we first provide a short description of the setup, which is then followed by changes and extensions, introduced for the purpose of IoT6 project.

The lamp is controlled by a very popular ARM based Linux embedded system i.e. Raspberry Pi [10], that is connected to the Internet via a WiFi dongle. On the Raspberry Pi, CoAP and HTTP servers are up and running and they are able to receive information coming from the Internet, process them, control the lamp and respond with the right messages. This enables control of the lamp using various devices (PCs, phones, tablets and complex devices) or systems that only use lamp for the visualization of the current state. More information about
the usage of the IoT LED Lamp in smart classrooms can be found in [12].

One system that can be used to control the lamp is the MindWave device [13] that is connected to an Android phone. The MindWave device is able to read brain wave activity and to send the raw data to an Android phone. A connection between the MindWave and a phone is established using Bluetooth technology. The application running on the phone can process raw data coming from the MindWave device and interpret it according to the level of attention and meditation.

Since both, the Android phone and the Raspberry Pi, are connected to the Internet, they could be used for this measurement (Figure 4). First, the Raspberry Pi sends the registration POST message to the Digcovery or to the RD in order to register the IoT LED Lamp. This registration message contains the appropriate description of the resource adapted to the system that is being used.

In order to communicate with the IoT Lamp, the Android application that can read and process raw data from the MindWave device, first needs to get a description of the lamp from either Digcovery or a Resource Directory. The Android application, when started, reads the IP address of the IoT Lamp from the resource description and according to the level of attention (gathered from the MindWave) sends the right message to the IoT Lamp, through the Raspberry Pi. The IoT Lamp shows happy (smile) or sad face, based on the level of attention captured by MindWave.

*Figure 4: Measurement setup for comparison of Digcovery and Resource Directory*
4.1.2. CoAP RD communication

In the HOBNET project, the Resource Directory is used as a key component of the HOBNET architecture, with all resources available to other components in the domain [25]. The RD is implemented as one lightweight system that provides mechanism for easy implementation of different front-ends, different parsers, serializers and different databases. The RD supports both HTTP and CoAP protocols at the application layer. As one of the scenarios in the HOBNET project, the IoT LED Lamp is used, as described in the following chapter.

Communication with the RD is via the IPv4 network using the CoAP as an application layer protocol. Raspberry Pi has a CoAP server that first updates the description on the RD and after that waits for the data. Android application has a CoAP client that sends data to the Raspberry Pi. RD is a database that contains descriptions of all the resources and its services and capabilities and can communicate using CoAP. An example of the communication flow is given in Figure 5.

![Figure 5: Communication flow – sending data from MindWave to IoT LED Lamp](image)

The steps in the communication flow are as follows:

- After the first boot, the CoAP server on the Raspberry Pi sends a PUT message to the RD in order to update its description (mainly the IP address):

  \[\text{CoAP PUT on coap://89.216.116.166:5684/rd/ep/ericsson.org.iotlamp}\]

  with the resource description in the payload.

- The response from the RD is the end point UID made from the domain name and end point name:
Ericsson.org.iotlamp,

where the domain is ericsson.org and end point name is iotlamp. Within one domain end-
point name must be unique. One end point can have more than one service (or resource in
the CoRE terminology [http://tools.ietf.org/html/draft-ietf-core-resource-directory-00])
and every service has its UID, i.e. ericsson.org.iotlamp.coapLight, where the ‘coapLight’
is unique within one end point. In this way, every end point can have more than one
service in its description, i.e. an end point could have a light actuator and a temperature
sensor:

<coap://212.200.203.135:5683>;ep="iotlamp";d="ericsson.org";loc="40.7292053820548
74.2066757475036";cat="light","/light";title="iflamp";if="i";rt="actuator";tag="light";ui
d="coapLight";</temp>;title="temperature";if="i";rt="sensor";tag="temp";uid="coapTemp
p"

- After receiving a message from the MindWave device about the level of brain activity,
  the Android application sends a GET message to the RD in order to find the
  appropriate address of the Raspberry Pi.

GET on the coap://89.216.166:5684/rd/ep/ericsson.org.iotlamp,

where the response is the resource description in the CoRE Link Format:

<coap://212.200.203.135:5683>;ep="iotlamp";d="ericsson.org";loc="40.7292053820548
74.2066757475036";cat="light","/light";title="iflamp";if="i";rt="actuator";tag="light";ui
d="coapLight"

- At this point the Android application gets an IP address and sends a PUT message to
  the Raspberry Pi:

CoAP PUT on the coap://212.200.203.135:5683/light,

with the state of the brain activity as the payload.

- According to the received message, the Raspberry Pi sends to the lamp through its SPI
  bus (Serial Peripheral Interface Bus) instructions to display a happy or a sad looking
  face.

4.1.3. Digcovery communication

Based on the findings in the HOBNET project, the Digcovery system is now introduced in
order to perform a comparison of the resource discovery mechanisms.

For enabling communication using Digcovery, some specific network configuration needs to
be made. In order to communicate over the IPv6 network, every device in the communication
needs to have an IPv6 address and needs to be IPv6-enabled. In addition, Internet Service
Providers needs to support IPv6 in order to be able to route communication in a proper way.
D6.2 Ubiquitous access and mobile phone network interactions

Because ISP’s in Serbia do not support the IPv6 addressing system, a tunnelling system was used provided by Gogo6 [14] with its Freenet6 Tunnel Broker [15]. A tunnelling mechanism enables IPv6 networking even if there is no support in the ISP. Within this mechanism, the IPv6 address is carried by the IPv4 network to the Freenet6 server, where the communication transfers completely to the IPv6 network. On Freenet6, a single, permanent IPv6 address and a DNS name are assigned to each user, making their device reachable from anywhere on the IPv6 Internet.

The Raspberry Pi was set with a Freenet6 client, which enables IPv6 communication. The client keeps the IPv4 tunnel constantly open with the Freenet6 server. Besides that, the client is in charge to get the static IPv6 address from the Freenet6 server. Freenet6 has a unique IPv6 address for every account created on its site.

In order to send data to the Digcovery, the Android phone also needs to have an IPv6 address. Raspberry Pi is basically a Linux machine and therefore it could be set to be a router for the local network enabling Internet access to the local devices. Since Raspberry Pi already has a tunnelling mechanism that provides IPv6, it is converted to be a Wi-Fi hot spot for the IPv6 network. In this way IPv6 enabled devices could get the IPv6 address through the Raspberry. A full /56 prefix is assigned to a Raspberry, enabling the distribution of IPv6 connectivity to an entire network. This system gave the global-dynamic and local-static IPv6 address to the Android phone.

The Raspberry Pi is acting as the IoT LED Lamp controller that has an IPv6 address on one side, and as an IPv6 border router that provides IPv6 connectivity to the local devices, on the other.

Since the Digcovery system has a built-in CoAP interface, our Raspberry Pi is able to register its services simply by sending the following PUT message to the Digcovery:

```
coap://[2001:720:1710:10::1000]:5683/dig?ep=iotlamp&proto=coap&port=5683&d=
example.com &lat=25&long=45&z=Belgrade,
```

where:
- ‘ep’ is the end-point name,
- ‘proto’ is protocol,
- ‘d’ is domain name (or in this case IPv6 address),
- ‘lat’ is latitude,
- ‘long’ longitude,
- z is the zone where the service is.

After registration, IPv6 enabled devices are able to discover available services and to use
D6.2 Ubiquitous access and mobile phone network interactions

them as shown in Figure 6.

![Diagram](image)

Figure 6: Communication flow – sending data from MindWave to IoT LED Lamp with IPv6 enabled devices

The steps in the communication flow are as follows:

- In order to discover the service and communicate with it, the Android application developed for this purpose sends a GET message, encapsulates the IPv6 address from the response and sends a PUT message to the Raspberry Pi service. The following GET message is sent to the Discovery system:

  `coap://[2001:720:1710:10::1000]:5683/dig?qt=2&ep=iotlamp`

- The response in json format is as follow:

  `[{"name":"coap.iotlamp","port":5683,"addr":"2001:0:5ef5:79fd:1465:a7a3:4d21:b96b","values": [{"value":"25@45","nameField":"geo"}, {"value":"Belgrade","nameField":"gps"}], "gps":"Belgrade","loc":[45.0,25.0],"domainName":"example.com"}]`

- The application extracts the ‘addr’ parameter from this message (which is the IPv6 address of the Raspberry Pi) and sends the following PUT message:


  with the state of the brain activity as the payload. Based on the content of the received message, the Raspberry Pi sends instructions to the lamp to display a happy or a sad face look.


4.1.4. Communication time comparison

In this sub-section, we compare the time needed to finish the interactions described above, for an IPv6 network and Digcovery system on one side and an IPv4 network and a CoAP Resource Directory system on the other.

In order to perform measurements, an installation similar to the one previously described above is configured. The Raspberry Pi is set as a CoAP server that is able to respond with the current status of the lamp (i.e. if a smiley or a sad face is displayed). After the first boot, a resource description with the current IP address of the Raspberry Pi is sent to the Digcovery or RD (depending on the system used). In addition to being a CoAP server, the Raspberry Pi acts as an IPv6 border router enabling any other device to get an IPv6 address and to communicate over the IPv6 network.

The communication flow is presented in Figure 7, indicating that both setups are topologically the same. A Smartphone application is able to request resource descriptions from the RD when the CoAP RD installation is tested or from the Digcovery when its system is being tested. When the application gets a resource description from the RD or Digcovery, it encapsulates the IP address from the received response and sends a GET message to the Raspberry Pi which responds with the current state of the lamp.

![Figure 7: Communication flow – Resource Directory / Digcovery response time comparison](image)

- If the Smartphone is attached to an IPv6 network, the application sends a GET message to the Digcovery:

  \[\text{coap://[2001:720:1710:10::1000]:5683/dig?qt=2&ep=iotlamp}\]

  and obtains the lamp’s resource description from the response.

- If the communication goes over an IPv4 network, a request is sent to the Resource
D6.2 Ubiquitous access and mobile phone network interactions

Directory:

`coap://89.216.116.166:5684/rd/ep/ericsson.org.raspiCoAPServer`,

the response is obtained in the resource description using the CoRE Link Format.

- After the communication with the Digcovery/RD is finished, an appropriate GET request is sent to the RaspberryPi:

  ```
  GET on coap:// IPAddress/light,
  ```

  where the IP Address is the IPv6 address obtained from the Digcovery or an IPv4 address obtained from the RD. The Raspberry Pi responds with the current status of the lamp.

The total response time was measured for the entire communication, i.e. Smartphone – Digcovery/RD – Smartphone – Raspberry Pi – Smartphone. In order to avoid the influence of the current conditions on the network, the measurements were repeated 100 times for every test. The CoAP RD installation was tested over a 3G network with 50 repeats.

The following graph (Figure 8) shows the results of the tests:

![Graph showing communication time comparison](image)

**Figure 8: Communication time comparison (Digcovery vs. RD)**

The average end to end times for different configurations are outlined in Table 1.:

<table>
<thead>
<tr>
<th></th>
<th>IPv6</th>
<th>IPv4</th>
<th>IPv4</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Network</strong></td>
<td>WiFi</td>
<td>3G</td>
<td>WiFi</td>
</tr>
<tr>
<td><strong>Number of repeats</strong></td>
<td>100</td>
<td>50</td>
<td>100</td>
</tr>
<tr>
<td><strong>Average time (ms)</strong></td>
<td>387.18</td>
<td>46.6</td>
<td>35</td>
</tr>
</tbody>
</table>

**Table 1: Comparison of Digcovery and RD**
As it can be seen, the best results, i.e. the shortest end to end communication time is achieved when WiFi network in combination with the IPv4 were used. Interaction over a 3G network is a little bit slower (46.6ms), while the worst results were obtained when IPv6 was used (387.18ms).

There are several reasons for such an outcome. First, the frame size is typically 127 bytes as per IEEE 802.15.4 standard [16], compared to 1280 bytes in IPv6. The larger the frame size the slower the communication is because of the amount of bytes that needed to be transferred. Although the IPv6 addressing system has a larger frame this is not the only reason for slower communication. In our experiment, the IPv6 network was established using a tunnelling mechanism, which enables IPv6 communication, but using IPv4 as the underlying carrier. The information carried, using the tunnelling, goes first to the Freente6 server through the IPv4 network, where it is transferred to IPv6 and then it is routed to its final destination through IPv6.

In the tunnelling mechanism, the IPv6 address is packed inside the IPv4 packet and the Freenet6 server needs to be contacted in order to switch completely to IPv6. This is needed in countries where the ISP has no support for IPv6. In order to see what influence the tunnelling has on the communication, the same tests were performed in Spain where ISP supports IPv6 and tunnelling is avoided.

The same application on the phone was used for these tests. The application first contacts the Dicovery for the test performed using IPv6 and the Resource Directory for the IPv4 test. Then, it gathers the description containing the address of the service and contacts the end point.

The results of the tests are shown in the Table 2:

<table>
<thead>
<tr>
<th>Network</th>
<th>IPv6</th>
<th>IPv4</th>
</tr>
</thead>
<tbody>
<tr>
<td>WiFi</td>
<td>55.6</td>
<td>64.4</td>
</tr>
</tbody>
</table>

*Table 2: Comparison of Dicovery and RD without the tunnelling for IPv6*

This result represents the ideal case for the IPv6 test, because Dicovery is close to the Smartphone and little routing is needed. In comparison with the previous tests, the average end to end delay is more than 300 ms lower which shows that the tunnelling mechanism had a large influence on the communication speed.

The communication speed for the IPv4 network is about 20 ms slower than in previous tests. This is because the Resource Directory used in these tests is located in Serbia. In the first case, the Smartphone application was also in Serbia while in the second case, it was in Spain.
D6.2 Ubiquitous access and mobile phone network interactions

It is obvious that communication speed is a little bit slower in the second case because of the higher impact of routing.

We intend to perform an update of the tests and also add more tests performed on real IPv6 networks in cooperation with UMU (until M30).

4.2. Mobile Phones with its own sensors

Every Smartphone has a number of embedded sensors, like GPS, microphone, speaker, camera, etc. If the data from the phones sensors could be accessed from the Internet they could be combined and used for getting a bigger picture about the environment. The potential is huge because there are so many Smartphones in the world. In this test case, we accessed the sensors on the phone over the Internet, through the IPv6 network. A Smartphone application is responsible for registering the phone’s sensors into a Digcovery directory. Another device (a laptop in this test case) searched the directory for the required service. After receiving the required description, a client application on the laptop communicated with the phone and collected measurements from the sensors on the phone.

While working over the mobile network (3G network), the mobile phone has no static IP address, therefore it is practically impossible to access the phones sensors over the Internet. If the phone is connected to the Internet over the WiFi, it would have a local IP address that could be static but it is an address in the local network and the port forwarding should be done for every WiFi network where the phone is connected. These problems are the reason why it is not practical and why it is not possible to implement any server on the phone that could be accessed from the Internet. Google has service for their notification system that they offer. In order to enable this function, the constant communication with their service needs to be open, which has a large influence on the battery consumption [17]. The problems with the accessibility over the Internet could be overcome with an IPv6 network because every device could have a unique IP address and access to the phone could be done directly.

In this test setup, an Android-based phone was used for the implementation of the IPv6 CoAP server, a Raspberry Pi was acting as an IPv6 border router and a laptop as an IPv6 client. The same mechanism for obtaining IPv6 addresses as in experiment 4.1.3 was used, i.e. a tunnelling system was used with the Freenet6 Tunnel Broker [15]. On the Freenet6, a single, permanent IPv6 address and a DNS name were assigned to each user, making their device reachable from anywhere on the IPv6 Internet.

A Freenet6 client was installed on the Raspberry Pi to enable the IPv6 communication. A static IPv6 address, accessible from the web was assigned to the Raspberry Pi. As explained in the previous section, the Raspberry Pi acts as a border router for the IPv6 network and enables the Smartphone to get its IPv6 address. This is to enable that a DHCP server is built on the Raspberry Pi, which assigns unique IPv6 addresses to every device that tries to connect with it.
The IP address assigned to the Smartphone is the following:

2001:5c0:1504:b800:50d0:f101:83cc:b8b4

In order to enable CoAP support, a Californium library was used. On the Smartphone, a CoAP server is built in and listens to port 5683.

Since this is a REST CoAP server in which there are several available interfaces: /sensor, /sensor/gps, /sensor/light, etc.

When started, in order to enable service discovery (Figure 9.) the Smartphone’s CoAP server registers its available resources and services on the Discovery, by sending the following PUT requests:

Coap://[2001:720:1710:10::1000]:5683/dig?ep=sensors&proto=coap&port=5683&d=example.com &lat=25&long=45&z=Belgrade,

and

coop://[2001:720:1710:10::1000]:5683/dig?ep=gps&proto=coap&port=5683&d=example.com &lat=25&long=45&z=Belgrade,

Figure 9: Test set-up, IPv6 communication between laptop and CoAP Server on the Android

The other end of the communication link is embodied in an IPv6 Client installed on a laptop. Similarly to the Raspberry Pi, the laptop obtained an IPv6 address through the Freenet6 Tunnel Broker and is able to make a request over IPv6. In order to read the available services
on the phone, the IPv6 client sends a GET request to the Digcovery server:

```
coap://[2001:720:1710:10::1000]:5683/dig?qt=2&ep=sensors
```

Digcovery responds with a description of those services:

```
[
  {"name":"coap.sensors","port":5683,"addr":"2001:5c0:1504:b800:50d0:f101:83cc:b8b4","values":[]},
  {"name":"coap.gps","port":5683,"addr":"2001:5c0:1504:b800:50d0:f101:83cc:b8b4","values":[]}
]
```

The description of both services is sent because they are registered from one domain.

After receiving this message, the IPv6 CoAP client installed on the laptop sends a GET request to the following address:

```
coap://[2001:5c0:1504:b800:50d0:f101:83cc:b8b4]:5683/sensors/
```

The request is tunnelled through the Freenet6 over the Internet to the Raspberry Pi which then transmits the request to its final destination, a Smartphone. Because the Smartphone listens on the 5683 port, it gets the request, processes it and sends back a response to the laptop through the same communication channel (Figure 10.). The response holds information about the available sensors on the phone:

```
“This is a list of available sensors:
KR3DM 3-axis Accelerometer
AK8973 3-axis Magnetic field sensor
GP2A Light sensor
GP2A Proximity sensor
K3G Gyroscope sensor
Rotation Vector Sensor
Gravity Sensor
Linear Acceleration Sensor
Orientation Sensor
Corrected Gyroscope Sensor
GPS”
```
4.3. Half Gateway implementation

In this section, an experiment when a mobile device acts as a Half Gateway is presented. A mobile phone represents a Half Gateway for sensors from devices that do not support the IPv6 protocol. These devices are connected to the Gateway via Bluetooth, infrared, etc. Since IoT implies connected devices via the Internet, it is crucial to show how these devices could have an Internet access over the IPv6 network. In order to do this there should be a Gateway able to communicate through IPv6 but still to be able to connect to a device via Bluetooth or Infrared. The mobile phone performs registration of these devices in Discovery or a Resource Directory, thus allowing their discovery and obtaining measurements.

In this setup, an Android phone with a CoAP Server on it, is used as an IPv6 Gateway for the Bluetooth enabled device, MindWave.

As explained in sub-sections 4.1. and 4.2, ISPs in Serbia do not support IPv6, and a tunnelling mechanism is provided in order to enable it. On Freenet6, a single, permanent IPv6 address and a DNS name are assigned to each user, making their device (PC, laptop) or any other
D6.2 Ubiquitous access and mobile phone network interactions

device reachable from anywhere on the IPv6 Internet.

The Raspberry Pi gets its IPv6 address with the Freenet6 client which enables the IPv6 communication. Beside this, the Raspberry Pi is set as the Border Router for IPv6 and therefore an Android phone could be connected to Internet via IPv6 through the Raspberry Pi. The IP address on the phone is dynamic and in this test it was:

```
2001:5c0:1504:b800:71bc:54c0:eee3:3b83
```

An application was installed on the phone. This application is able to communicate over the IPv6 network, reads and processes the EEG (Electro Encephalograph) data from the MindWave device and has a CoAP server that waits for the request from the Internet. The Californium library was used for the server implementation. The server listens to the port 5683. The MindWave device is able to read brain wave activity and to send raw measurements to the Smartphone. A connection was established between the MindWave device and the Smartphone using Bluetooth. A Smartphone application running on the phone can process that data coming from the MindWave device and interpret it as the level of attention and meditation. This setup enables access to the Bluetooth device via IPv6 network.

After running the IPv6 EEG CoAP Server Application on the phone (Figure 11.), it sends the registration data to the Discover:

```
coap://[2001:5c0:1504:b800:71bc:54c0:eee3:3b83]:5683/dig?ep=attention&proto=coap&port=5683&d=example.com&lat=25&long=45&z=Belgrade
```

and

```
coap://[2001:5c0:1504:b800:71bc:54c0:eee3:3b83]:5683/dig?ep=meditation&proto=coap&port=5683&d=example.com&lat=25&long=45&z=Belgrade
```
The IPv6 Client on the laptop is on the other side of the communication. It has an IPv6 address also through the Freenet6 Tunnel Broker and it is able to make a request over the IPv6. In order to read available services that a phone can provide, the IPv6 client sends a GET request to the Digcovery Server:

```
GET on coap://[2001:720:1710:10::1000]:5683/dig?qt=2&ep= attention
```

Digcovery responses with the description of the service:

```
[{
  "name":"coap.attention","port":5683,"addr":"2001:5c0:1504:b800:71bc:54c0:eee3:3b83","values": [{"value":"25@45","nameField":"geo"}], "value": "Belgrade", "nameField": "gps" },
{
  "name":"coap.meditation","port":5683,"addr":"2001:5c0:1504:b800:71bc:54c0:eee3:3b83","values": [{"value":"25@45","nameField":"geo"}], "value": "Belgrade", "nameField": "gps" }
]
```

In order to read the data from the MindWave device, the IPv6 CoAP Client on the laptop sends a GET request to the Android application that can handle the CoAP request and it is connected to the MindWave device via Bluetooth:

```
GET on coap://[2001:5c0:1504:b800:71bc:54c0:eee3:3b83]:5683/attention/
```
D6.2 Ubiquitous access and mobile phone network interactions

or

```
GET on coap://2001:5c0:1504:b800:71bc:54c0:eee3:3b83]:5683/meditation/
```

The request is tunnelled through the Freenet6 over the Internet to the Raspberry Pi, which then transmits the request to its final destination, the Android phone. Because the Android phone listens to the port 5683, it gets the request, processes it and then sends back the response to the laptop through the same communication canal (Figure 12.). The response holds information about current data gathered from the MindWave device.

![Communication flow diagram](image)

*Figure 12: Communication flow – getting the attention or meditation level from the MindWave device*

If the request was sent to the attention interface, the response holds information about the current attention level, and if the request was sent to the meditation interface, the response holds information about the current meditation level. These levels are relative values between 0 and 100.

### 4.4. Implementation of mDNS on a mobile phone (Android based)

The goal of this use case was to perform an initial exploration of the feasibility of implementing mDNS [18] on a mobile phone (Android based).

The purpose of the experiment was to prove that services offered by an IPv6 client can be browsed using an Android mobile phone, and vice versa, that services offered by Android mobile phone can be browsed from another phone or workstation.
We used the same Virtual Machine (VM) settings as in the experiment described in sub-section 2.1.4. in deliverable D3.1 [3]. In this experiment the settings of the VM network adapter had to be set to the bridged mode. All tested devices were connected to the same wireless router.

![Figure 13: Experiment setup](image)

The experiment had two phases. In the first phase of the experiment, services were browsed from a mobile phone. On the VM Lab2 the _ecobus._tcp. service was defined, using the resource directory description of the ecobus resource [19]. When the Avahi Browser application was started on a mobile phone, the mDNS query in the .local domain was initiated and a list of the available services was obtained.

In Figure 14, several screenshots showing results of mDNS query in the .local domain are given:

- The first shows list of services offered from VM
- The second shows which IP addresses of found workstations are shown
- The third shows the attributes (location) of the _ecobus._tcp. service
In the second phase of the experiment, the purpose was to prove that services offered by an Android mobile phone can be browsed using mDNS. For this experiment, we used jmDNS project [20] in which an AndroidTest service is published.

To start a simple service, we first created ServiceInfo structure, then called jmdns:

```java
serviceInfo = ServiceInfo.create("_test._tcp.local.",
    "AndroidTest", 0,
    "test from android");
jmdns.registerService(serviceInfo);
```

```
...jmdns.unregisterAllServices();
jmdns.close();
```

The other side is the same VM used in the previous experiment phase. Both the VM and the Android phone were connected to the same wireless router. The Avahi Discovery browser application was started on the VM and the jmDNS application on the mobile phone. The VM started the mDNS query in the .local domain.

In Figure 15, The result of the mDNS query in the .local domain in which the service offered by Android phone is listed from the VM Avahi Discovery. Service browsing is possible also
from another mobile devices in the same .local domain.

![Figure 15: Result of mDNS query in .local domain from VM Avahi Discovery](image)

This experiment proved the possibility of the mDNS implementation for service discovery purposes in an IoT6 network which contains mobile devices in both directions (publishing/browsing from a mobile phone).

### 4.5. Implementation of mDNS when mobile phone acts as a mobile hotspot

The purpose of this experiment was to provide a proof of concept for the feasibility of implementing mDNS service publishing/browsing in mobile network, when a mobile phone serves as a mobile hotspot. From the group of several mobile phones with embedded sensors, one was selected to be the wireless router (tethering - functionality available in Smartphones [21]). All the other phones connect to the WLAN (Wireless Local Area Network) using the selected phone. The application for mDNS publishing/browsing is possible in both directions (from one of the phones in the group and from WLAN nodes).

The experiment setup is shown in Figure 16. We used an Android phone as a mobile hotspot. Another Android phone was connected to the mobile hotspot. Also, an Ubuntu [22] workstation advertising the service _busmobile._tcp and another PC with an Ubuntu VM advertising the _ecobus._tcp service were connected to the mobile hotspot.
D6.2 Ubiquitous access and mobile phone network interactions

From Lab2-VM Ubuntu we checked the publishing of the service _busmobile._tcp from the Ubuntu Workstation using the Avahi browser. The output of this application is shown in Figure 17.

Figure 17: Service browsing from Lab2

From an Android phone using the Avahi browser and the mDNS protocol, all available services in the .local domain were listed. When the experiment was setup, the Avahi browser
registered both services advertised from the Ubuntu Workstation (busmob) and from the VM virtual machine (ecobus). This is shown in Figure 18. The application Zeroconf browser for Android phone was used.

![Image]

*Figure 18: Service browsing from mobile phone*

This part of the experiment proved the transparency of the mDNS service publishing/browsing in the described scenario in mobile networks [23].

### 4.6. Testing of global publishing/browsing techniques

According to the mDNS specification [18], publishing/browsing is limited to the .local domain. For integration into a global network, techniques for global publishing/browsing should be used. In [3] a connector artifact (mDNS / DNS-SD Connector) was introduced, to be instantiated in the .local domain of the elements (resources, devices) that should be published globally.

The connector architecture is shown on Figure 19. The connector component is in charge of receiving mDNS requests and performing the discovery operation by searching through the “Information Infrastructure” to obtain the external services published by other connectors that match with the query, and finally sending the results to the requester client. “Information Infrastructure”, which is an overlay network, and DHT, initially built with Chord [10].
D6.2 Ubiquitous access and mobile phone network interactions

Figure 19: Overview of the mDNS / DNS-SD Connector, showing how to publish globally the resources and devices found on each local network

The next step in the experimental work should be the testing of global publishing/browsing techniques, starting from a mDNS/DNS-SD connector which should be integrated into a mobile network.

We intend to produce an update of this deliverable D6.2 document and add more tests performed on real IPv6 networks, in cooperation with UMU (until M30). These tests will be carried out after UMU finishes the implementation of Digcovery.
5 Conclusions and recommendations for further integration of mobile phones within the IoT6 platform

The objective of this deliverable was to address the aspects and possibilities of integration of LTE mobile devices with IoT6, while fulfilling the project defined requirements, i.e. to enable the mobile phones to provide ubiquitous access to the smart things connected to the developed architecture. Then to enable smart things or systems connected to the architecture to connect and send messages to the mobile phone and finally to enable the architecture to use mobile phones as mobile sensing tools, and to retrieve information from the mobile phones, such as temperature, motion, localization, etc.

The registration procedure and addressing of mobile devices have been analyzed, together with the analysis of mobile devices acting as Gateways and Half Gateways in the IoT6 architecture. The principles of integration of LTE devices with the IoT6 system have been given. In section 2, different ways of IP addressing LTE devices was discussed and it can be concluded that an optimal solution would be that operators define a dedicated APN, where additional specific parameters could be defined for IoT6 devices.

In the scope of the IoT6 architecture, a LTE network represents one of the enablers for communication with IoT6/M2M devices. A mobile phone can be an IoT6 device itself or one or more devices can be connected to the mobile phone in which case the mobile phone acts as a Gateway or a Half Gateway. A mobile device can have its own sensors (embedded) or different sensors can be connected wirelessly (for example via Bluetooth, Infrared, etc.) or using wires (for example via an USB connection).

Various ways to integrate an IPv6-based IoT into mobile phone networks enabling mobile devices to provide access to smart objects as well as to use mobile devices as sensors/actuators were explored. In this document, experiments set-ups have been explained, and results presented and analyzed for selected use cases. All experiments and measurement setups were based on the IoT6 architecture.

Although an IPv6 addressing system is not supported by every ISP and not every device is IPv6 enabled, for the IoT it is crucial to develop and explore the boundaries. This addressing system enables every IoT device to have a unique address which facilitates implementation by avoiding NAT-ing or port forwarding. In this document some implementations were presented in order to show the importance of the IPv6 for IoT and potential possibilities that IPv6 offers.

It was very challenging to work with IPv6 especially because in Serbia ISPs do not support IPv6. In order to overcome this problem, a tunnelling mechanism was used. The Freenet6
D6.2 Ubiquitous access and mobile phone network interactions

Tunnel Broker enabled carrying IPv6 addresses through the IPv4 network, which enables local networks IPv6 connectivity. To avoid having the tunneling mechanism for every device, a Raspberry Pi device was provided with one and it was configured to work as a Border Router for IPv6. In this way, any device could have IPv6 connectivity using WiFi through the Raspberry Pi. Also, every device connected through this system has a unique IPv6 address. In section 4.1, the communication between two end devices was presented using the IPv6. The first end device was a Raspberry Pi that controlled the IoT LED Lamp. It has a CoAP server that could communicate using the IPv6. After boot, this CoAP server sends the service descriptions to the Digcovery system and in that way enables the possibility to discover them.

On the other side of the communication was an IPv6 enabled Android phone as a second end device. This phone had an application that was able to discover the services registered at the Digcovery system and, by doing so, it was able to send a request to the IoT LED Lamp through the Raspberry Pi.

Beside this implementation, the ability to discover the available sensors on the mobile phone and get the data from them is presented. To make this possible, an Android phone was provided with the unique IPv6 address and CoAP server. On the first boot CoAP server on the phone, registers its services to the Digcovery which enables any other system or device to discover them. Any CoAP client with CoAP 13 support was enabled to get the service description from the Digcovery and, by doing so, it was able to send a request for the data from the phone. In this implementation an IPv6 CoAP client on the laptop was used.

With the same system, this document describes the option for non-IP devices to be discovered and contacted. To do this, an Android phone was connected to the MindWave device using a Bluetooth connection. An application with the CoAP server on the phone was able to read the data from the MindWave device. The CoAP server first registers the services that the MindWave device has on the Digcovery. An IPv6 CoAP client on the laptop reads the description gathered from the Digcovery and it is able to send a request to the CoAP server on the phone and in that way get the data from the Bluetooth device, MindWave.

Resource and service discovery mechanisms were also discussed. A test was executed that compared a CoAP RD system that uses IPv4 with the Digcovery system that uses IPv6. The results showed that IPv6 networking is still work in progress and that, at the moment, tunnelling mechanisms must be used to provide help for the case when an ISP does not support IPv6, but as a consequence additional delay is introduced in the communication.

One of the main project goals is to use the properties of the IPv6 protocol, like mDNS and DNS-SD that can be used for resource and service discovery functionalities instead of a standard approach. For that purpose few test cases were performed.

In one test case initial exploration of the feasibility of implementing mDNS on an Android based mobile phone was performed. An appropriate experiment was setup and executed. It was proved that services offered by an IPv6 client can be browsed using an Android mobile phone, and vice versa, that services offered by Android mobile phone can be browsed from
another phone or workstation.

In another experiment a proof of concept for the feasibility of implementing the mDNS service publishing/browsing in mobile network, when a mobile phone serves as a mobile hotspot was performed. From the group of several mobile phones with embedded sensors, one was selected to be the wireless router while the other phones were connected to the WLAN using the selected phone. The application for mDNS publishing/browsing is possible in both directions (from one of the phones in the group and from the WLAN nodes). The experiment proved the transparency of the mDNS service publishing/browsing in a described scenario in a mobile network.

A feasibility study of integrating a LTE mobile network and Digcovery will be explored. We intend to update this deliverable D6.2, adding more tests performed on real IPv6 networks in cooperation with UMU (until M30). This will be done after UMU finishes the implementation of Digcovery.
6 References

[1] IoT6 D1.2. First version of IoT6 architecture & SOA specifications
[2] IoT6 D1.3 Updated version of IoT6 architecture & SOA specifications
[3] IoT6 D3.1. Look up and discovery, context-awareness and resource
[12] Nenad Gligoric, Tomislav Dimcic, Srdjan Krco, Vladislav Dimcic, Jelena Vaskovic, Iva Vojinovic, ”Internet of Things enabled LED Lamp controlled by satisfaction of students in a classroom” , IPSI, transactions on Internet Research July 2014 | Volume 10 | Number 2 | ISSN 1820 - 4503 (under review)
[17] http://www.androidhive.info/2012/10/android-push-notifications-using-google-
cloud-messaging-gcm-php-and-mysql/


[19] www.ekobus.rs


