Grant Agreement No.: 258378

FIGARO

Future Internet Gateway-based Architecture of Residential Networks

Instrument: Collaborative Project

Thematic Priority: THEME [ICT-2009.1.1] The Network of the Future

Architecture for service federation in residential networks

Due date of deliverable: 31.08.2011

Actual submission date: 31.08.2011

Start date of project: October 1st 2010 Duration: 36 months

Project Manager: Henrik Lundgren, Technicolor R&D Paris

Revision: v.1.0

Abstract

This document defines the preliminary version of the FIGARO architectural solution for federation within residential networks. The architecture is derived from use cases from the domains of e-health, energy management, domotics and social community services and thus supports requirements from each of these domains. This deliverable describes and validates the architecture and its main components.

Project co-funded by the European Commission in the 7th Framework Programme (2007-2013)		
Dissemination Level		
PU	Public	✓
PP	Restricted to other programme participants (including the Commission Services)	
RE	Restricted to a group specified by the consortium (including the Commission Services)	
CO	Confidential, only for members of the consortium (including the Commission Services)	

Document Revision History

Version	Date	Description of change	Editor	Authors
V1.0	31.08.2011	Version submitted to the EC	TNO	Philips, TNO, Home Automation Europe, Martel (formal review and quality control)

Table of Contents

LIST OF ACRONYMS

ACS	Automatic Configuration Server
AHD	Application Hosting Device
AN	Access Network
API	Application Programming Interface
Bonjour	Trade name for the implementation of a specific SDP
CE	Consumer Electronics
CE-HTML	Language for creating UI pages for CE
CN	Control Network
CPE	Customer Premises Equipment
CPU	Central Processing Unit
CWMP	CPE Wide Area Network (WAN) Management Protocol
DCP	Device Control Protocol
DHCP	Dynamic Host Configuration Protocol
DLNA	Digital Living Network Alliance
DM	Device Management
EUD	End-User device
GENA	General Event Notification Architecture
H.264	Standard for video compression
HDMI	High-Definition Multimedia Interface
HG	Home Gateway
HGI	Home Gateway Initiative
HL7	Standard of Health Level Seven
HN	Home Network
HNID	Home Network Infrastructure Device
HRN	Health Record Network
HTML	HyperText Markup Language
HTTP	HyperText Transfer Protocol
IEEE	Institute of Electrical and Electronics Engineers
IF	Interface
IP	Internet Protocol
ISO	International Organization for Standardization
JINI	Network architecture for the construction of distributed systems in the form of modular co-operating services
KNX	OSI-based network communications protocol for intelligent buildings
LAN	Local Area Network
NAS	Network Attached Storage
NP	Network Provider
OS	Operating System
OSGi	Open Services Gateway initiative
OSI	Open Systems Interconnection
PAN	Personal Area Network
PanID	Number for identifying the network in ZigBee
PD	Peripheral Device
PN	Peripheral Networks
POTS	Plain old elephone service
R-EUD	Retail EUD
RG	Residential Gateway
RPC	Remote Procedure Call
RUI	Remote User Interface

SDP	Service Discovery Protocol
SLP	Service Location Protocol
SOAP	Simple Object Access Protocol
SP	Service Provider
SP-EUD	Service Provider EUD
SSDP	Simple Service Discovery Protocol
TR-069	Technical Report nr 069 van Broadband Forum
UI	User Interface
UPnP	Universal Plug and Play
URI	Uniform Resource Identifier
URL	Uniform Resource Locator
USB	Universal Serial Bus
UUID	Universally Unique Identifier
VB	Virtual Backbone
VM	Virtual Machine
VoIP	Voice-over-IP
VPN	Virtual Private Network
WAN	Wide Area Network
WiFi	Standard for wirelessly connecting electronic devices
WPA	Wi-Fi Protected Access
X10	Industry standard for communication over wired power line or wireless used for home automation
XML	Extensible Markup Language
ZigBee	Specification for a suite of high level communication protocols using small, low-power digital radios based on an IEEE 802 standard for PANs

1 INTRODUCTION

The Future Internet will not be restricted to the IT, Telecom and media sectors. It will also include services and user-centric content from other sectors, such as energy management, e-health, and domotics. We foresee a rapid growth of Internet-based applications and services in these other sectors. Moreover, users will expect to access these services and their content in the same manner as any other Internet-based service. The Future Internet must therefore evolve to support not only the increasing demands in the IT/telecom/media sector, but also meet the new requirements from these emerging sectors.

Until recently, those sectors were used to develop their own communication infrastructure and system solutions. The intra-sector optimizations in such closed systems often lead to poorly standardized and non-scalable solutions. For instance, while the Internet Protocol (IP) clearly plays an important role in the aggregation of sectors, the services and applications in these sectors are most often build on non-IP technologies. Hence, there is a need for coordination across different sectors in order to interconnect them and ultimately enable collaboration among them.

In FIGARO, we define the concept of a "*federation of residential networks*". The residential gateway will undertake the role of federator, and connect two or more networks within a single residence facilitating cross-sector convergence. We call a residential federation *internal* when a single residence is considered. In contrast, we call a federation *external* when multiple gateways interconnect multiple independent residential networks. This document focuses on internal residential federation.

In Deliverable 5.1 [1] we described the state of the art of the e-health, energy management, domotics and community residential ICT services domains. A rich set of use cases provided insights into the domain specific requirements and led to a set of guiding principles for the design of the residential gateway. In this document we define a basic architecture for the internal federation of residential networks, based on the requirements in [1]. This document does not aim to be a final specification, but only a first step upon which further research and work in the project will be based. It will therefore be superseded by later Architectural Description documents from the FIGARO consortium.

This document is organised as follows. In Chapter 2 we define a number of concepts that form the basis of our architecture and we quickly summarize the general design principles as defined in [1]. In Chapter 3 we describe the architecture for the internal federation of residential networks. Chapter 4 describes implementation examples based on the architectural modules we have developed within the project so far. In Chapter 5 we reflect upon this architecture and analyze its applicability to the use cases and the general principles. We conclude this deliverable in Chapter 6 with some final remarks and discussion of future work.

2 ARCHITECTURE DESIGN

In this chapter we define two concepts that play an important role in our architecture. The first concept is a common service delivery infrastructure, which is an abstraction layer on top of the network layer, ultimately facilitating the integration of services from all domains. Secondly, we acknowledge that there will always be non-IP networks such as control networks that exist next to current IP networks. They need to be interconnected to the common service delivery infrastructure to form a communication infrastructure and enable collaboration among them. This chapter defines the necessary proxy functionality for this. Finally, we shortly revisit the general design principles defined in [1] that will provide guidance in making architectural choices. In the following, we first give a summary of the IEEE 1471 methodology we based our work on to define the FIGARO architecture.

2.1 Recommended practice for architecture description

In defining our architecture we largely follow the methodology as described in ISO/IEEE 1471-2000, "Recommended Practice for Architecture Description of Software-Intensive Systems" [2], here just called IEEE 1471. In IEEE 1471, architecture is defined as the fundamental organisation of a system, embodied in its components, their relationships to each other and the environment, and the principles governing its design and evolution. IEEE 1471 provides definitions and a meta-model for the description of an architecture. Figure 1 provides the key concepts in the context of an architecture for a particular system and an associated architectural description.

From IEEE 1471 follows that an architecture should address a system's stakeholders' concerns, and that architecture descriptions are inherently multi-view. A view is defined as a representation of a whole system from the perspective of a related set of concerns. In essence, IEEE 1471 states that no single view adequately captures all stakeholders' concerns. Therefore, before designing an architecture, the stakeholders and their concerns should be known first. This knowledge is best obtained by identifying the relevant use cases, as we have already done in [1].

Another important aspect of an architecture is the context, or the environment. The context determines the setting and circumstances that have influence on the system. The context can include other systems that interact with the system of interest, either directly via interfaces or indirectly in other ways. In this case the boundaries are given by the in-home environment and the residential networks.

Not all concepts in Figure 1 are as relevant for FIGARO. In this deliverable and D5.1 [1] we only discuss the concepts of "system", "environment", "architecture" "architectural description", "views", "stakeholders", and "concerns". For a more thorough understanding of the other concepts in Figure 1 ("mission", "viewpoint", "library viewpoint", "rationale", and "model") we refer to [2].

2.2 Common Service Delivery Infrastructure

The integration of services from new sectors, such as e-health, energy management and domotics, with the Future Internet poses challenges for how to interconnect different networks and systems that do not necessarily use IP technology. Current innovations in e-health services and remote energy management are still hindered by a tendency to create vertical "stove-pipe" ICT solutions (i.e. dedicated to a specific service), often leading to non-scalable, expensive and closed systems. By extending current intra-sector control and interface platforms, we aim to deliver new management and control modules for a *common service delivery infrastructure*, in which the residential gateway can play a key role.

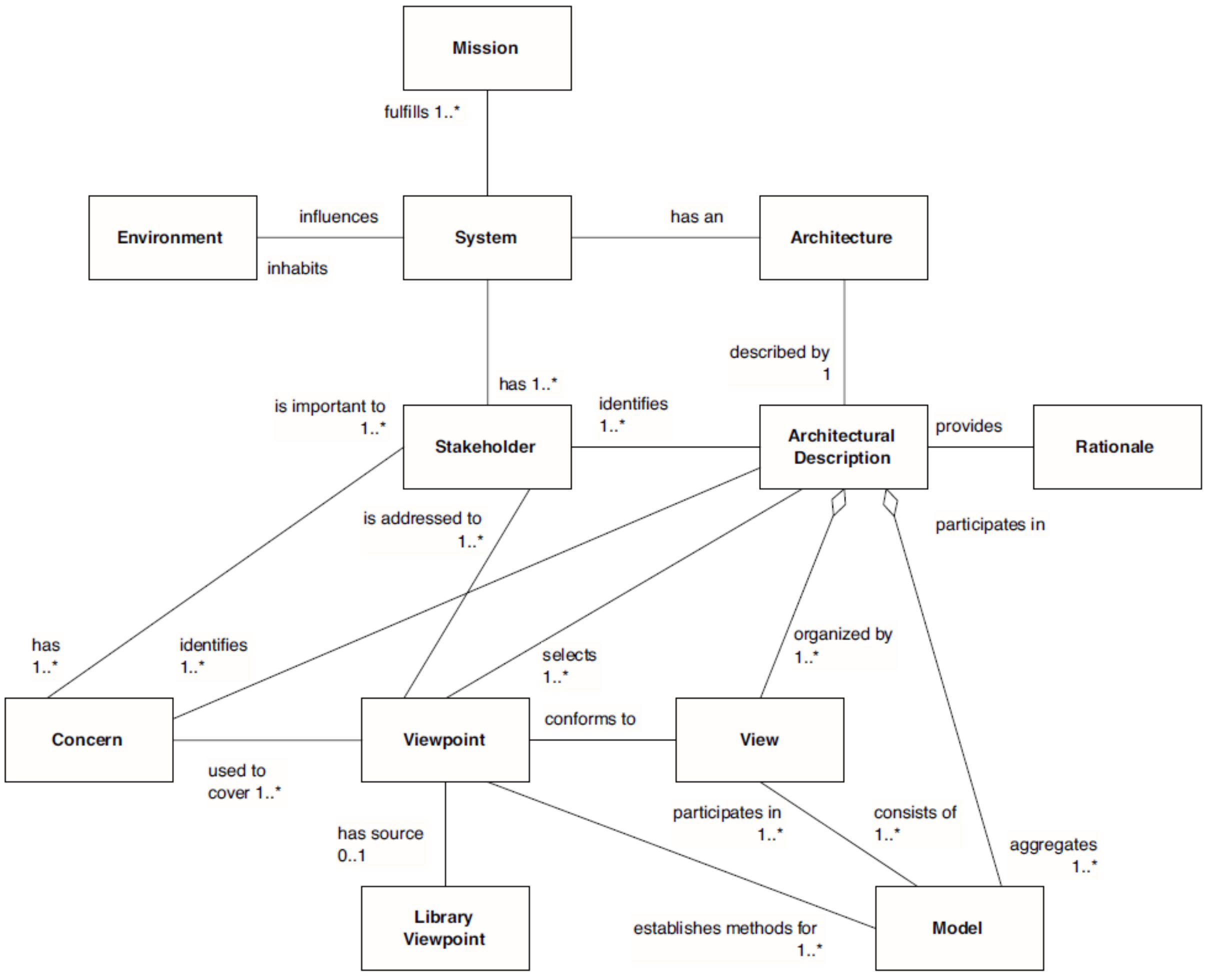

Figure 1. Overview of key concepts in defining an architecture of a system (from [2])

Figure 2 gives a rough architectural vision at what a common service delivery infrastructure should encompass. A common service delivery infrastructure lies on top of a common communication infrastructure which is mainly based on the IP family of protocols. Connected to the communication infrastructures are, what we call, function specific devices (such as smart meters, digital thermometers, set-top boxes and connected TVs), and more generic devices (such as PC, laptops, smartphones, tablet PCs). Besides a common IP interface to the common network layer, we foresee that these devices will also provide common application-layer interfaces to the common service delivery layer.

The common service delivery infrastructure itself provides ***standardized*** core components, such as message routing, service component registration and discovery. This is a fairly new approach to home networking, which is loosely based on the concepts of Service Oriented Architecture [3]. The idea is that not only a common network layer is needed for reaching truly cross-sector interoperability, scalability, and flexibility, but also a common middleware or service delivery layer, which converges and unifies middleware functionality just as IP converged and unified available network resources. On top of this generic service layer more service- or sector-specific requirements for services can be materialized to enable the actual services. Examples are service enablers in the sectors multimedia & communication, e-health and energy management.

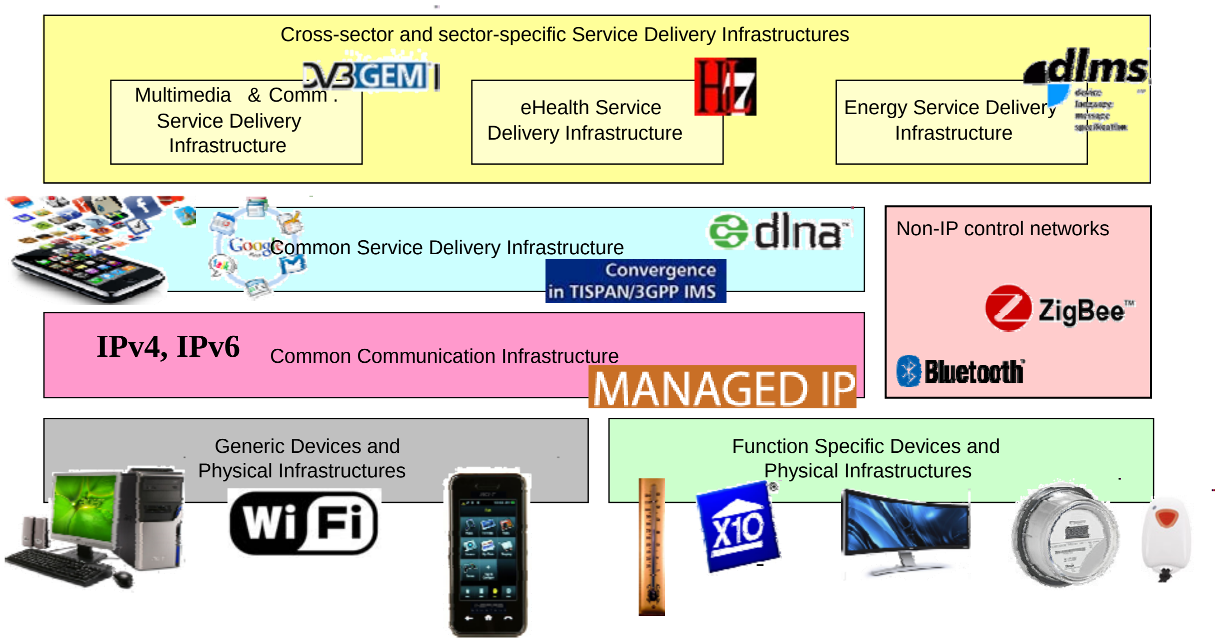
Cross-sector and sector-specific Service Delivery Infrastructures
Multimedia & Comm. Service Delivery Infrastructure
DVB
GEM
eHealth Service Delivery Infrastructure
HL7
Energy Service Delivery Infrastructure
dlms
Common Service Delivery Infrastructure
dlna
Convergence in TISPAN/3GPP IMS
Non-IP control networks
ZigBee
Bluetooth
IPv4, IPv6
Common Communication Infrastructure
MANAGED IP
Generic Devices and Physical Infrastructures
Wi Fi
Function Specific Devices and Physical Infrastructures
X10

Figure 2. Common service delivery infrastructure (the logos are indicative) [4]

An example where the concept of a common service delivery and common communication infrastructure becomes clearer is DLNA. DLNA [5] is a cross-industry collaboration delivering an interoperability framework of design guidelines in the multimedia sector. The interoperability guidelines define an architecture based on a single common communication infrastructure (IP) and a single common service delivery infrastructure (in this case fully compliant with UPnP [6,7]). On higher and lower layers, DLNA supports a much larger variety of protocols. Based on the requirements in [1] we will define an architecture in line with this philosophy. This focus is largely in line with the visions on the general Future Internet Architecture as laid out for instance by the European Future Internet Assembly [8].

Besides IP-based communication networks, other network technologies such as Zigbee and Bluetooth can also be present in a home. They provide their own, inseparable network and service delivery layer, and are usually only meant to interconnect function-specific devices with the home network. In the FIGARO project, a part of the architecture is designated to define proxy functionality between these non-IP networks and the common network and service layers.

2.3 General design principles

In [1] we defined a set of use cases and requirements specific for the domains of e-health, energy management, domotics and community services. From these requirements we derived a set of general design principles which are briefly summarized hereby:

- ***Loosely-coupled integrated control*** using information brokering; this means that failure of one of the providing or consuming sub-services, does not result in failure of all functionality.
- ***Distributed connection management for federated gateways;*** in this case the gateway aids in setting up connections between devices, but is not required for further communication. Such mechanism ensures that when the gateway fails, at least parts of the home system continue to operate.

- ***Security and privacy;*** this includes protecting the user against attacks on the home network, preventing others from getting access to the user’s personal data, and also prevents the user from tampering with information services.
- ***Graceful degradation of services;*** systems and services that are interconnected must be able to maintain an acceptable level of basic operation if the network fails. That is, if coordinating or governing super-systems are no longer available, the system should be able to run autonomously.
- ***Support for multi-vendor systems, multiple service provider, multiple gateways;*** consumers will use multiple services from multiple service providers, even though new services could be installed on a gateway the architecture must be able to support multiple gateways.

These general principles are used in the definition of the Internal Federation Architecture. They will provide guidance for making architectural choices. In the next chapter we describe the Internal Federation Architecture from a networking perspective and define the required components. In Chapter 4 we will evaluate to what extent the architectural choices fulfil the generic requirements and support the use cases.

3 RESIDENTIAL SERVICES FEDERATION ARCHITECTURE

In this chapter we define the architecture of the internal federation of residential networks. We start by identifying the home network architecture (OSI layers 1-3) and define control networks. Then we identify service management and control modules that should be present in the network to provide generic functionality in the common service delivery infrastructure.

3.1 Home network architecture

In this section we postulate the home network architecture, based on the SmartHouse Roadmap Reference Model, as defined in [9]. The reference model contains all generic building blocks needed to describe the smart house services and applications based on devices in the home, which result from the use cases defined in [1]. The model is shown in Figure 3. This model can be used to describe both applications which are completely contained within the home (such as central heating thermostat), as well as applications depending on external services (such as remote patient monitoring and automatic smart-meter reading).

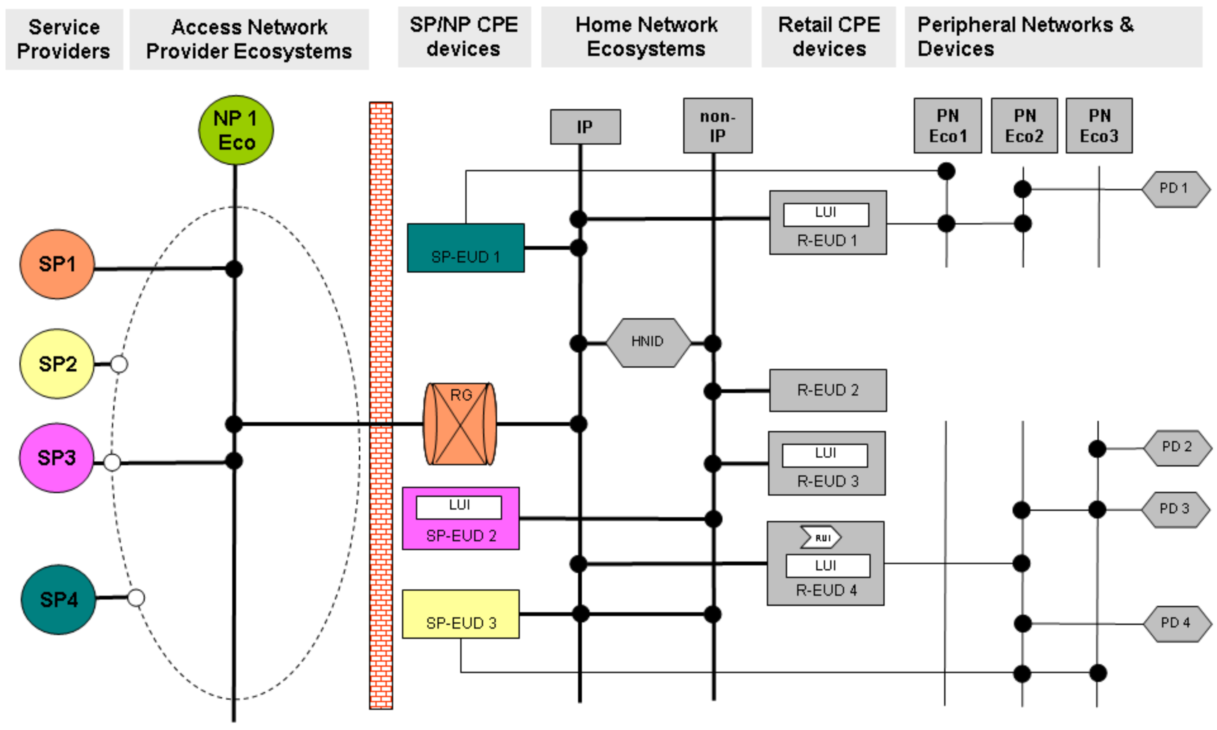

Figure 3 Reference model for home network architecture

Service providers (SP) provide a service (e.g. a Web service, ISP service, TV, voice, energy service, e-health service) typically from outside the home to end user devices (EUD) in the home. Access Network providers (NP) provide the use of their AN (e.g. ADSL, Cable, FttH). Other FIGARO reports focus on differentiation of access networks, but in this report we focus on federation within the home only. Therefore, for simplicity, we assume only one broadband access network entering the home. EUDs in the home may provide in-home services to other EUDs in the home (e.g. home automation applications) or to the Internet. Besides EUDs we also have Home Network Infrastructure Devices (HNIDs), Peripheral Devices (PDs) and a Residential Gateway (RG) in the home.

The following definitions are used to describe the infrastructure and networks.

- ***Access Networks (AN):*** the last hop of the service delivery pipe (indicated via thick line and black dots) that physically connects an End-User device (EUD) or a Home Network Infrastructure Device (HNID) to a service delivery network outside the home.
- ***EUDs**:* devices operated by the end user and offering a User Interface (UI) to the end user through which he can access services. EUDs can be managed and owned by
  - The service provider: SP-EUD, e.g. a Set-Top Box, smart meter or a Residential Gateway). SP-EUDs are usually leased or given away for free by service providers as part of the service agreement.
  - The consumer: R-EUD (retail EUD). e.g. a TV, PC or electronic weighing scale. The consumer buys these devices independent from the SP in the retail shop.
- ***HNIDs:*** EUDs that do not offer a UI to the end-user to access the service itself e.g. network terminators, bridges, switches, routers, gateways. HNIDs carry and/or manage the data of the service delivery pipe. HNIDs may provide a UI to the end-user to configure the functions of the HNID or these devices may be managed remotely by the network provider (NP) or service provider (SP). HNIDs can also provide the interconnection between multiple (home) networks. The Residential Gateway typically contains HNID functionality.
- ***Residential Gateway (RG):*** Typically one device (but can be more devices) that connects one or more access networks to one or more home networks and delivers services to the home environment.
- ***Home Networks (HN):*** physically interconnect (indicated via thick line and black dots) EUDs and HNIDs inside the home, and can be interconnected by HNIDs as well. We distinguish IP and non-IP (or control) networks.
- ***Virtual Backbone (VB):*** virtually connects (indicated via dotted oval, thick line and white dots) an AN to a global infrastructure like the Internet or POTS or IMS backbone connecting NPs and SPs. How this is done is out of scope for the Reference Model. It may involve complete (inter-)networks running business or management applications between SPs and NPs.
- ***Peripheral Networks (PN):*** physically connects (indicated by thin line and black dots) peripheral devices (PD) to a master device, e.g. USB, HDMI, and Bluetooth. PDs typically connect to one master and do not exist standalone. The master may aggregate PD device functionality and make it accessible over the HN. Some PN technologies can in principle also be used for non-IP HN control networks, e.g. Bluetooth, and IEEE 1394. EUDs and HNIDs can have both PN and HN interfaces.
- ***Local User Interface (LUI):*** user interface on the device itself (buttons, display, etc.). In the remainder of the document we only discuss Remote User Interfaces (RUIs), which are exported to other devices.

The colours in Figure 3 designate who has the ownership/control over a network or device. Thus, the colours relate to the business models we assume and the stakeholders whose concerns we intend to cover. Every different colour indicates a different business role. A given business party may cover one or more business roles. In Figure 3 we, for instance, assume that SP1 (e.g. an ISP) owns, controls and manages the RG, but does not own, control, and manage the public access network (which in turn is own, controlled, and managed by NP1). The colour grey means that the device or network is owned, controlled and managed by the end user. These devices and networks are typically bought in retail shops. The end user may decide to outsource the management and control of some of his devices to other stakeholders, for instance a service provider.

It is also important to realize that in practice the devices RG, EUD, HNID, etc. may be combined in a single physical device in any combination. For instance, a Network Attached Storage (NAS) may also contain an Ethernet switch.

3.2 Control Networks

Internal federation requires seamless interworking of multiple types of networks in the home, both IP- and non-IP-based. Currently, various types of new networking technologies are appearing on the market. Compared to Ethernet these technologies offer much lower data transmission rates. However, their strength is in low energy consumption and low cost. Typically, these networks support several hundred kbit/s, whereas Wi-Fi offers several hundred Mbit/s. Energy consumption is low and most of these devices need to send messages only every now and then.

This allows devices to run on a small battery for years. In the remainder of this document we refer to these low-cost, low-speed and low-energy networks as "*Control Networks*", and refer to the Ethernet and Wi-Fi based home network as the LAN. The LAN is considered IP-based, whereas the control networks are mainly non-IP. The availability of control networks and interconnection to other networks creates new opportunities, which can be exploited using a common service delivery infrastructure.

The possibility to add wireless connectivity to many types of devices in these control networks creates opportunities for innovation. Moreover, bridging or proxying these networks to the Ethernet-based home network (LAN) really enables a whole new range of possibilities. As a result, basically any device can become connected to the Internet at a very low cost. For example, a device for measuring blood pressure can transmit information to a cloud storage server, the central heating thermostat can be reached using a mobile phone, and the energy consumption can be read remotely by a service provider.

3.3 FIGARO functionality of the common service delivery framework

To support the use cases from the domains of e-health, energy management, domotics and community services the common service delivery infrastructure needs to provide sufficient functionality. This includes uniquely identifying network nodes, discovering nodes and services offered by nodes, a description mechanism for the services that are offered, remote management, and user interfacing. Each of these topics will be discussed in the following sections.

3.3.1 Addressing

In a network it must be possible to address each node in a unique way. On OSI-layers 2 and 3, a node is a network interface card. We assume that network nodes are assigned addresses in the common ways. A network address is then used to send a message to a certain node, and to detect from which node a message originated. In the LAN this is accomplished by IPv4 or IPv6 addresses.

Control networks also uniquely identify network nodes. However, in most cases the required address range is limited. In a Zigbee network the PanID and short address are used to uniquely identify nodes. The PanID is a 16 bit number that is generated by the coordinator. When nodes join a network this PanID is used to identify the network. In a Zigbee Pro network, a Zigbee Coordinator forms a network by selecting a PanID, then the coordinator sends a message to detect if other networks in the neighborhood might be using the same PanID. Once a PanID is chosen the coordinator can allow other nodes to join the network. Any end device wishing to join a network sends a message to discover which networks are out there. Once a suitable network is found a message is sent to request joining the network.

A theoretically simple way to achieve interoperability between IP- and non-IP networks is by discarding any layer above OSI layer 2 in the non-IP network and encapsulate IP packets in the native layer-2 frames of the non-IP network. This is equivalent to creating an IP network from the non-IP network. On paper this would solve all our concerns, but in practice this is not possible, e.g. because of the limited bandwidth of the non-IP network. Federating IP- with non-IP networks must therefore be achieved on the higher layers, i.e. by creating proxy functionality. On these higher layers the concerns are more about device and service instance identification than about network interfaces addresses. Said otherwise, on higher layer "nodes" are devices and service instances rather than network interface cards. For this, identification and addressing schemes such as port numbers, Uniform Resource Identifiers (URIs), and Universally Unique IDentifiers (UUIDs, such as used in UPnP) become more relevant.

3.3.2 Discovery

When a node has a unique address it does not mean that other nodes in the network can discover a certain node. In the LAN a node does not always need to have a mechanism to make itself easily discoverable. For example, a mobile phone may connect to a home network via Wi-Fi, but it does not have to respond to a ping. However, when nodes offer services that are used by other nodes in the network a devices and service discovery mechanism is required. For broadband IP home networks, many discovery mechanisms exist such as Bonjour, Service Location Protocol (SLP), JINI and Simple Service Discovery Protocol (SSDP). Most discovery protocols meant for use in home networks are based on a broadcast or multicast methods.

To address all devices in a Control Network, these networks usually include a discovery mechanism that is inherent to the network solution. In Zigbee RF4CE [10], controller nodes connect to a target node. Thus creating a star network in which the target node exactly knows which nodes are present in the network. When bridging Control Networks to an Ethernet based LAN, the nodes in the control network should be discoverable from the LAN. A discovery message from the LAN could be sent to every node in the control network, and the nodes in the control networks could reply with a response message. Since most control networks have implemented their own discovery mechanisms, a better solution is to map the LAN discovery protocol to the control network discovery protocol. This removes the need for nodes to implement a new discovery mechanism that allows them to be discovered from the LAN. The proxy device that is responsible for the forwarding of messages from the control network to the LAN (and vice versa) can easily provide this mapping of discovery protocols. This also facilitates discovery of nodes in a control network that are in sleep mode. In control networks where low energy consumption is important (for battery operation) nodes usually are in sleep mode, and only wake-up to send a message. A thermometer may wake up only once per 10 minutes to send a temperature reading. A switch might only wake up the moment the user presses it. Still from the LAN side it would be good to be able to detect that a switch is present in the control network.

3.3.3 Description mechanisms for devices and device capabilities

When a networked device offers services to other devices in the network, these other devices need to be able to find out what service is actually offered. Is the device a networked printer, or is it a device which can playback an H.264 video file? Similarly in control networks we need to know whether a device is a temperature sensor, a blood pressure meter, a switch, a dimmable light, etc. Depending on the application and the device a certain amount of additional information is needed. Within the context of a control network it is often sufficient to use an enumeration type value to indicate the device type. Within the applicable standard all devices can be listed. In many cases it is useful to provide the device manufacturer, the version of the device, and the device's serial number. In more complex cases it is necessary to provide a complete listing of device capabilities and functions which the device can execute.

When including device and service discovery and control functionality in a common service delivery layer, the proper standardization of the message semantics is crucial. When a network device offers a service, the meaning of a message or a function call needs to be defined. A standardization document describes exactly which messages and functions must be available, and what the meaning of a message and what the result of a function call is. UPnP, for instance, defines which functions must be implemented for each service offered by a device, and which functions are optional. Similarly the IEEE11073 standard [11] defines the semantics of messages sent by e-health devices, and the Zigbee cluster library defines which commands and messages are implemented for certain types of devices.

3.3.4 Remote management

From a telecommunications management point of view, devices in the home network are nothing more or less than a network element in the telecommunications network. Ideally, network elements do not need to be managed, because they always work perfectly and configure themselves automatically. In practice however, the configuration of the elements and the errors they exhibit must be managed. To save on truck rolls, network elements must be managed remotely as much as possible.

For remote management of end-user devices, web-services based management has become increasingly popular. One of these web-services based protocols is the CPE Wide Area Network (WAN) Management Protocol (CWMP). It is defined in TR-069 of the Broadband Forum [1] and developed for the sole purpose of HG management. Its primary capabilities are secure auto-configuration and dynamic service provisioning, software/firmware image management, status and performance monitoring, and diagnostics.

TR-069 defines Remote Procedure Calls (RPCs) and, importantly, also standardizes the data model of the RG and various other end-user devices such as Voice-over-IP phones (VoIP), set-top boxes, and network-attached storage devices. In TR-069, the remote management server is called Auto-Configuration Server (ACS). TR-069 is currently gaining wide acceptance with service providers. Other end-device management is often based on vendor-specific web services and a pull model: the end device must initiate the management session.

3.3.5 User interfacing

We distinguish devices that have a user interface to export (controlled devices) and devices to which the user interfaces are exported (controllers). Three fundamentally different paradigms can be distinguished for exporting user interfaces to other devices [9]:

1) Functional models. This is the most common model deployed by current systems. Devices and the services they offer are modelled as object command sets that require (sometimes detailed) knowledge in the controller of the functionality of the device to be controlled and its services. Interoperability scenarios are based on gateways mapping command sets from different protocols.

2) Remote UI models. This model exports the control UI of the device and its services to the controller instead of the control commands themselves. The controller only needs to create and present a UI to the user from the UI description it receives and sends back the UI responses from the user via the Remote UI protocol. In this model the controller does not need to be aware of the functionality of the device and services it controls. This model requires a

standard protocol to export UIs that should work with the common control devices in the home such as PC, TV and mobile phone. The best known example of a Remote UI protocol are web pages stored in devices that are displayed by a browser on the PC or TV (i.e. HTML over HTTP). Also DLNA and UPnP are based on this model. Other examples are thin client varieties such as the Remote Desktop Protocol: UIs are exported as graphic primitives instead of as a web page.

3) Hosting models. In this model the controller acts as a host that uploads and executes a control program from the controlled device that communicates with and controls the object on the controlled device. Here, most intelligence is with the controller. Browsers running Javascript are good examples of this model. For FIGARO, this model seems to be less relevant than the other two. FIGARO instead opts mostly for model two, also for non-IP systems which now still mostly follow model 1.

.

3.3.6 Device virtualization

With the device and capability description functional component described in Section 3.3.3 it is possible to discover and use devices and their capabilities or services they offer in a standardized way. To enable other devices not only to use a remote device's capabilities and services, but also to install new capabilities and services on that device, the devices need to be virtualized. That means it should run a Virtual Machine (VM) of some kind. A VM is a software implementation of a device that executes programs just like the actual device. VMs are separated into two major categories, based on their use and degree of correspondence to any real devices.

A system VM provides a system platform which supports the execution of a complete operating system (OS). In contrast, a process VM is designed to run a single program, which means that it supports a single process. Process VMs are the most relevant to the construction of a FIGARO common service layer. JAVA VM is a well-known example of a process VM. OSGi [1,12] is an extension to JAVA VM, adding resource management, life cycle management, and many standardized services. Another example of a process VM is the Dalvik VM. It is strongly related to JAVA VM and is used by the Android OS for device virtualization. The industry is working on creating a version of OSGi that can also run on Android.

3.4 FIGARO Common Service Delivery Framework Architecture for residential networks

The FIGARO Common Service Delivery Framework Architecture for home networks is basically stating that devices in the home network (see Figure 3) should support one or more functions as described in Section 3.3 of this report. This requirement is realistic as long as we do not require that full interoperability can only be achieved by having these functions implemented with the same technologies. In reality, home networks are and to our opinion will remain technologically heterogeneous, also on the common service delivery layer. This is a direct outcome of the business model we assume in Figure 3. Therefore the home network should also contain proxy functionalities wherever needed. In addition to the devices named in Figure 3, we therefore also define a proxy device, which has the sole responsibility to proxy service delivery implementations. The proxy may be a separate device or run as a function on any of the devices mentioned in Figure 3.

It is not realistic to expect that every device implements all functionality as given in Section 3.3. An RG may very well contain all functions, but a simple small sensor may only support addressing and discovery. Figure 4 provides an Architectural Description of our common service delivery framework functionality and how this functionality be distributed over the various types of devices. A box with a border drawn with a full line means that the function is mandatory. A dashed line means that the function is optional. We do not make any distinction yet regarding to the implementation model chosen. For a client-server type architecture, for instance, the functionality as described in Figure 4 may be implemented as a server or a client, or both, depending on the use cases to be supported.

The RG arguably should support all functionality as described in this deliverable. From both the WAN as the LAN side it should be addressable, discoverable, manageable, etc. On the LAN side it should also be able to address, discover, manage, etc. other devices and services/capabilities in the home network. It should contain various proxy functionalities, for LAN-LAN proxying as well as WAN-LAN proxying. It may also include an IP/non-IP proxy, but in Figure 4 such a proxy is drawn only externally.

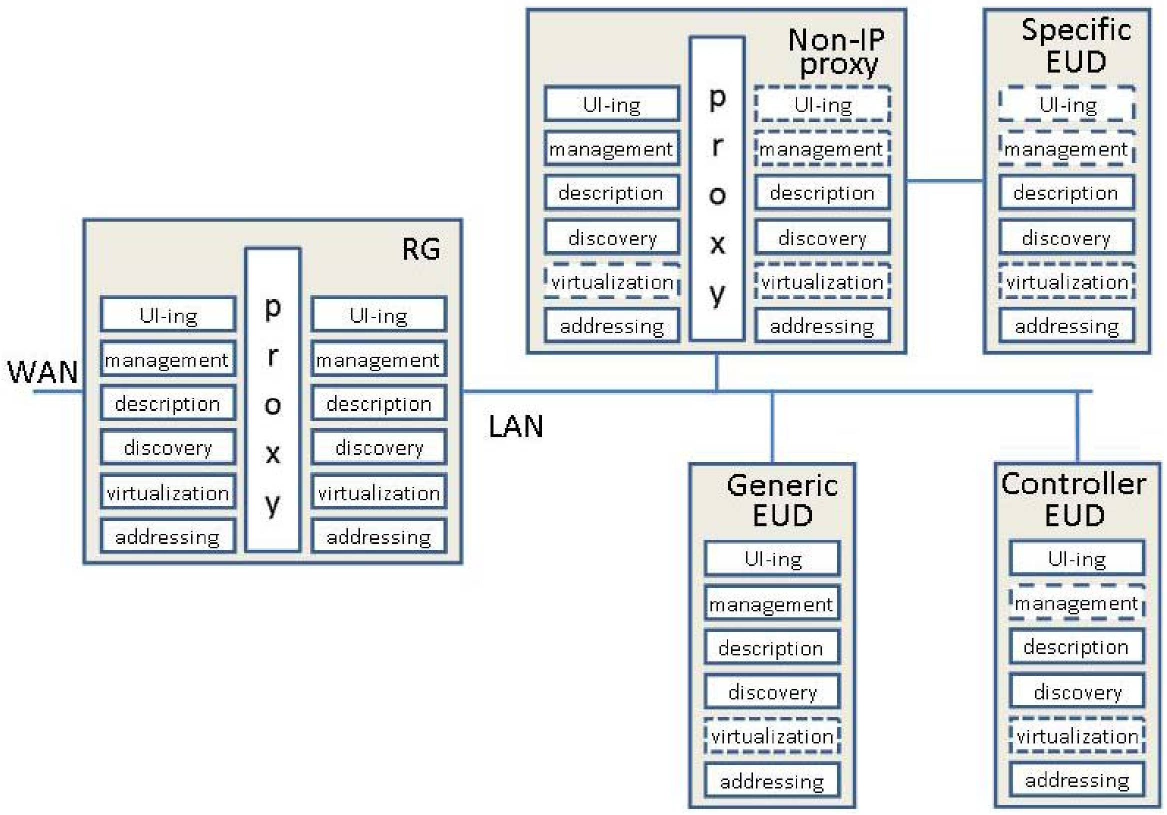

Figure 4. Preliminary architectural description of the FIGARO common service delivery infrastructure in the home network

An IP/non-IP proxy (in the figure just called “Non-IP proxy“) may contain less functionality than the RG, but to support the FIGARO use cases we demand the proxy as well as any other device and service in the home network to be at least addressable, discoverable, and remotely accessible (which is achieved by the device supporting the description functionality). On the IP side of the proxy we demand it to be remotely manageable as well, and it should present a user interface to controller devices.

In Figure 4 we assume Generic EUDs and Controller EUDs to be IP devices and Specific EUDs to be not. This is of course not necessarily the case. These devices may turn up in any part of the home network, IP or not-IP, but we expect most realistic implementation to be as described in Figure 4.

The only functionality that is not mandated for any device except the RG is virtualization. Virtualization is especially useful for generic devices that need to support services from different service providers and are strongly subject to frequent upgrading. That is for sure the case for the RG, and somewhat less so for proxies and generic EUDs in general. But a smart phone, for instance, can be seen as a combination of a Generic EUD and a Controller, to which the requirements of multiple service provider support and frequent upgrading definitely apply. The commercial success of Android smart phones is therefore partly due to their support of a process VM.

It is still debatable if the architecture should also support a federation controller functionality. The idea is that the federation of the home network services may have to be administered and managed. This may be done centrally in the RG or in a distributed way. However, the definition and necessity of such a function needs further discussion before being possibly included in a next version of the architectural description.

4 IMPLEMENTATION CHOICES

Various technologies exist with which the architecture can be implemented at least partly. Most of them have been described in [1]. The challenge is then to integrate these technologies seamlessly into a working system.

In this chapter we describe the choices we have made and indicate the gaps in the architecture for which no technologies exist yet. We also indicate which software components the authors of this deliverable (the FIGARO team dealing with federation in the home network), have already developed at this stage of the project. The latter concerns:

- A UPnP/Zigbee proxy and UPnP Control Point specific for Zigbee network control
- A UPnP/Bluetooth proxy
- A TR-069/UPnP proxy
- Home network services control and management display

4.1 UPnP

In this section we analyse how UPnP (Universal Plug and Play) can be used for addressing, discovery and description in the IP domain, and to connect non-IP Control Networks (low-speed, low-power, low-cost networks) to the Ethernet and Wi-Fi infrastructure (LAN) in the home. UPnP is the central service delivery framework protocol of DLNA [5]. As already introduced in Section 2.2, DLNA is an industry alliance which uses standards-based technology to make it easier for consumers to use, share and enjoy their digital photos, music and videos DLNA defines guidelines which provide strict rules on the use of UPnP to guarantee interoperability.

4.1.1 UPnP for addressing, discovery and description in the IP domain

UPnP is a set of networking protocols for residential networks that permits networked devices, such as personal computers, printers, Internet gateways, Wi-Fi access points and mobile devices to seamlessly discover each other's presence on the network and establish functional network services for data sharing, communications, and entertainment [1,7]. UPnP defines devices and control points. Devices announce themselves in the network, and offer services. Control points detect the devices and can invoke functions. Note that a UPnP device does not need to correspond with a physical device. A tablet PC could implement, for instance an UPnP Media Renderer device and also an UPnP Media Server device. The same tablet could also implement a control point for browsing content on other Media Servers. Since control points invoke functions on a UPnP device, different types of control points exist. Mobile smart phones may implement control point functions for browsing content on a Media Server, as well as functions for instructing a Media Renderer to playback a certain URL. In Figure 3, the UI functionalities of the controller can be implemented by an UPnP Control Point. In FIGARO, we also require controllers to make themselves known as such in the network. Controllers then are aware of each others presence in the network, and also of each others possibly conflicting behaviour [13]. They also have to be discoverable if they need to be remotely manageable. In UPnP a Control Point does not announce itself in the network. It is only required to listen to announcements and to be able to control devices. Therefore a new UPnP Device needs to be defined that runs next to Control Point software on a controller.

UPnP requires that the network supports either AutoIP (RFC3927) or DHCP for IPv4 addressing, and link-local address auto-configuration for IPv6 addressing. For discovering devices and services, a mechanism is described in UPnP which is called SSDP (Simple Service Discovery Protocol). A control point can send an SSDP search message, which is a multicast message that uses HTTP. Each

UPnP device responds with a SSDP message containing a URL that refers to a device description document. Additionally, each UPnP device sends an SSDP message to the network at regular intervals. These “Alive” messages are used to verify that a UPnP device is still present in the network. Absence of “Alive” message could indicate that a device crashed or is unreachable due to a network problem (a normal device shutdown requires it to send a bye-bye message).

When a control point has received an SSDP message it can use the URL to access the device description document. This document is an XML document that provides the device type. This way the control point can determine if it can control the device. A control point that is capable of accessing an Internet Gateway Device and open or close ports, would not be interested in media servers that could be running in the home. If a control point discovered a UPnP device it is interested in, the device description document can be used to get all information needed to control it. The device description contains amongst others a friendly name, the manufacturer, a list of services (basically just a logical grouping of functions), and a complete set of functions implemented by the devices.

Using the list of services and actions (functions) a control point can invoke a function. In UPnP this is implemented by means of a remote procedure call mechanism based on SOAP (Simple Object Access Protocol). SOAP is essentially a mechanism to send XML messages over HTTP. The XML message describes the function name, its parameters and their values to invoke an action on a device. Devices in UPnP can also transmit events. The event mechanism is called GENA (General Event Notification Architecture). GENA events are based on sending XML messages using HTTP headers. To receive events a control point first subscribes to the events of a device. If, for example, a server device has new content available, or the sound volume of a renderer was changed, devices can send an event to all subscribed control points.

4.1.2 Defining an IP/non-IP UPnP proxy gateway

To connect non-IP control networks to the LAN, some device in the network needs to have both an interface to a control network and an interface to the LAN. This device would receive messages from the control network, and forward them to the LAN. Messages from the LAN which are intended for the control network must be sent to the correct node in the control network.

In principle it would be possible for each node in the control network to implement the IP stack. Each node would get an IP-address, and message forwarding becomes trivial. However, a discovery mechanism that describes what the node has to offer would still be needed. A drawback of this solution is that each node would need to run these protocols. Considering examples like a switch or a thermometer this is a large amount of overhead compared to just sending a simple message. A thermometer probably uses a very small and cheap microcontroller as its CPU. The residential gateway (or any other device used for proxying control networks and the LAN) has a CPU that is much more powerful. Between 10 and 100 times more processing power is not unlikely. For this reason we investigate a solution which offloads most of the protocol implementations to the IP/non-IP proxy gateway.

The technical overview of UPnP provided in the previous section makes clear that the UPnP network middleware covers all our requirements concerning addressing, discovery and description. If we offload most protocol efforts to the proxy gateway we can still use UPnP as a middleware solution. We could implement the proxy gateway in such a way that it represents each node in the control network as a UPnP device in the LAN. In such a solution each light and each switch would show up as a separate UPnP device. In most use cases, however, such fine granularity is not needed. Besides, Zigbee networks might contain hundreds of nodes and a large granularity may then introduce a scalability problem. We choose to let the proxy gateway represent a control network as a single UPnP device. The new UPnP device (running on the proxy gateway) represents one control network. The

UPnP device will present itself on the LAN as a UPnP Control Network proxy. A control point can discover this UPnP Control Network Proxy device. If the gateway has a Zigbee radio, it is discovered in the network as one UPnP Control Network Proxy. If the gateway would also implement a KNX proxy, the control point discovers two UPnP Control Network proxy devices.

The UPnP Control Network (CN) proxy is automatically discovered by control points that are capable of controlling the new device. Similar to other device types, the device description contains information about the device. Besides friendly name, and manufacturer, this may include an indication of the network type. The UPnP CN proxy device implements an API that can be used to retrieve information about the network, and control its attached devices by sending messages to a node identifier (e.g. PanID and short address). Even if a node in the control network is in sleep mode, the proxy device can still provide information about the node. A function can be added to allow control over which devices are visible from the LAN. To receive messages it will be more convenient to use an event-like mechanism than to poll for messages using a call function. UPnP uses GENA for events. Any control points can subscribe to events from a certain device. A control point that subscribes to the GENA events of a CN proxy would receive all messages from the control network.

This is also the approach we have taken in proxying Bluetooth to UPnP [14]. We have implemented such a proxy, and its architecture is shown in Figure 5. Besides the UPnP and Bluetooth protocol stacks (white), the core architecture (grey) contains a device and service discoverer, a converter, a device and service announcer, a UPnP adapter, a Bluetooth adapter, and a database consisting of various tables. They are explained in more detail in [14]. Figure 6 shows how the devices are translated into services and vice versa. The top part of the figure depicts the discovery of UPnP devices and services by a Bluetooth Service Discovery Protocol (SDP) client in the Bluetooth network. We assume that the proxy has only one Bluetooth hardware interface. Therefore the Bluetooth client device in the Bluetooth network ("piconet") will discover it as a single Bluetooth device, containing Bluetooth services 1A-6B that represent properties of the UPnP services 1-6 as well as the properties of the UPnP devices A and B that contain the UPnP services. The bottom part of Figure 6 shows the discovery of Bluetooth devices and services by a CP in the UPnP network. We assume that the proxy has only one IP address. Therefore a CP will discover it as a single UPnP root device. Because the proxy contains an SDP client, it can discover Bluetooth services actively by performing a search at set times. There are two options for representing the discovered Bluetooth devices and services in this root device. The most obvious one seems to have a UPnP embedded device representing a Bluetooth device, and its UPnP services to mirror the Bluetooth services. However, Bluetooth service records often contain much more information than UPnP service descriptions, whereas a Bluetooth device is identified with not much more than its hardware address. From the details of the descriptions in Bluetooth and UPnP, it can be concluded that it is easier to represent a Bluetooth service as a UPnP embedded device containing one or more UPnP services. The downside of this approach is that a CP does not know which Bluetooth device it addresses when invoking a service.

In case a Zigbee network is attached to the gateway, the gateway will run a UPnP/Zigbee CN proxy device. A control point on any device in the home network can detect the presence of this device. Checking its device description will reveal it as a Zigbee network. The CN proxy provides a function to check which devices are present in the Zigbee network. The control point could, for example, be an android application that presents and stores blood pressure information. If the control point discovers that the Zigbee network contains a device that measures blood pressure, it can subscribe to the GENA events and process the received messages.

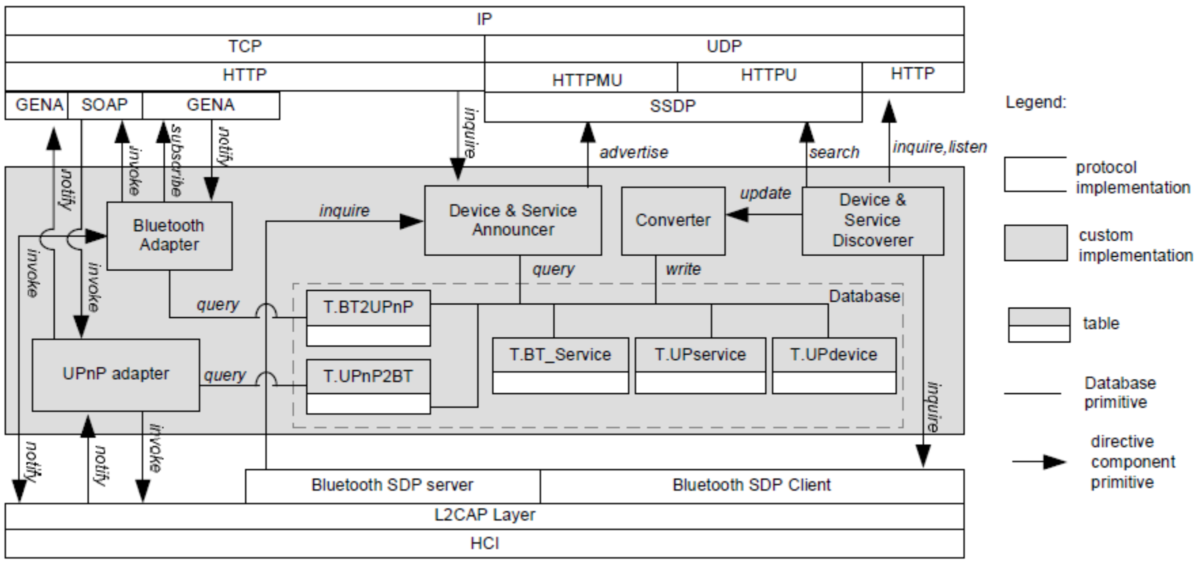

Figure 5. Architecture of a bi-directional UPnP/Bluetooth proxy [14].

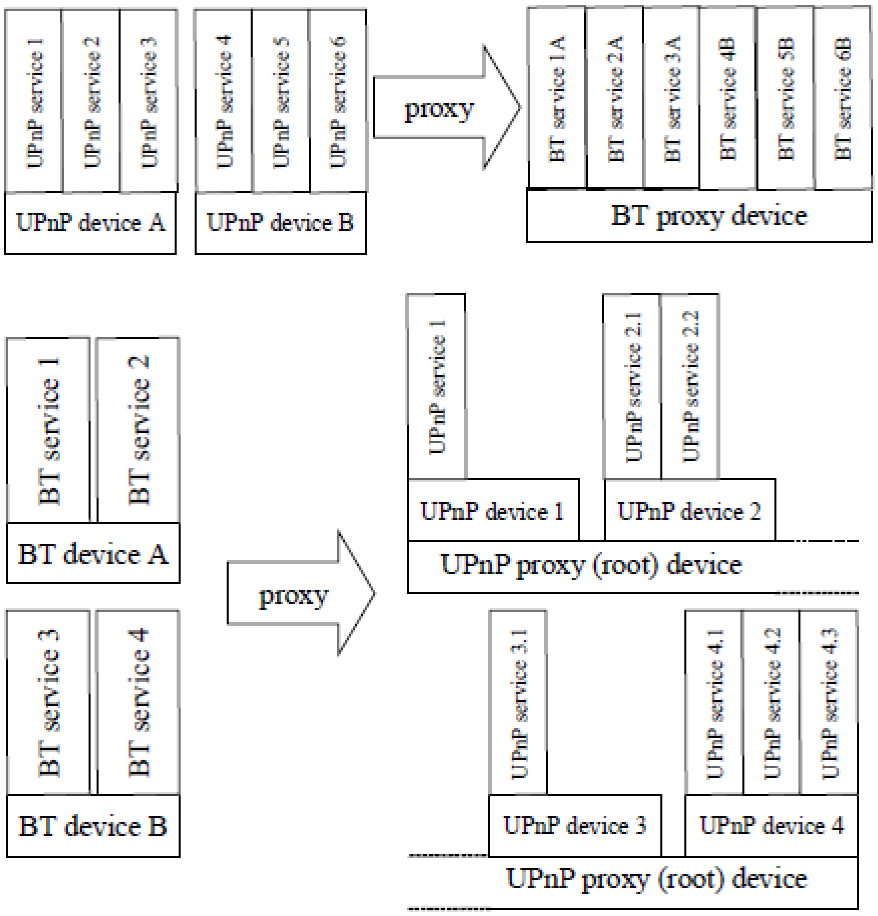

Figure 6. The proxy represents the UPnP network as a single device to the Bluetooth network and vice versa. The different devices are represented as different services of the proxy gateway.

4.2 Continua

A large part of the architectural requirements for e-health applications and services can be implemented by using the Continua Design Guidelines. The Continua Health Alliance [15] aims to establish an interoperable ecosystem for personal health systems by selecting standards and specifications and define these in guidelines. In this section we discuss the scope of the Continua Design Guidelines, after which the integration into the common service delivery infrastructure is presented.

4.2.1 Continua Design Guidelines scope

Continua aims at establishing a system of interoperable personal e-health devices and services, and considers the complete e-health end-to-end chain from measurement devices in the home to the clinical information systems such as remote monitoring services and other network-based services. While this leads to interoperability in the e-health domain, opportunities leading to trans-sector applications and services are not considered. An example would be to share measurement data in the home and access them by multimedia devices, instead of sending them directly to medical service provider. The scope of Continua is shown in Figure 7.

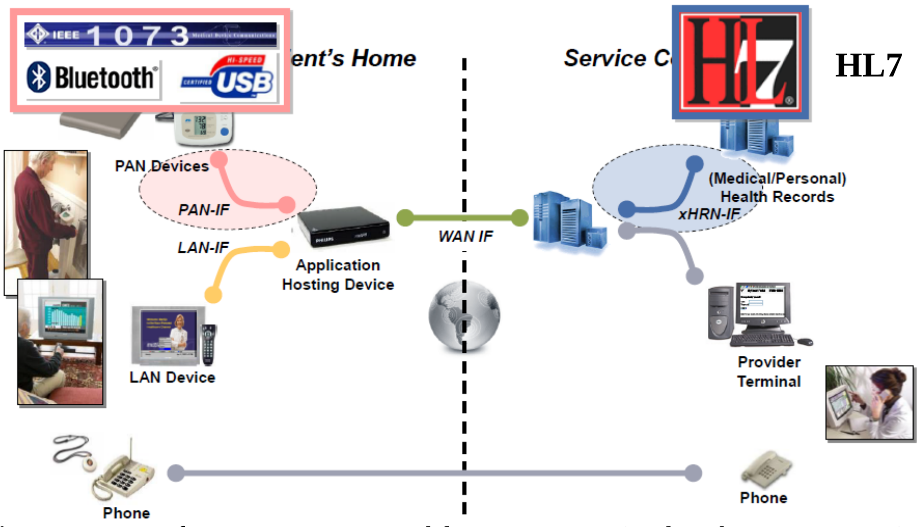

Figure 7. Scope of Continua Design Guidelines v1. IEEE 1073 later became IEEE11073.

On the left-hand side of the figure is the patient's home in which the measurement devices are used. One can think of blood pressure meters, thermometers, etc. The measurement data is collected at the Application Hosting Device (AHD), a data concentrator, at which point the data is converted from IEEE11073 [11] to HL7 [16]. From the AHD the data is sent to the medical backend systems of the service centre on the right-hand side of the figure. To ensure interoperability, Continua specifies four interfaces (IF) that are present in this scope: the Personal Area Network (PAN), the Local Area Network (LAN), Wide Area Network (WAN) and Health Record Network (HRN) interface. The interfaces are shown in Figure 8. Note that the Continua terminology differs from our terminology. PAN and LAN networks in Continua refer to Bluetooth and Zigbee networks respectively. Both these networks we refer to as *control networks*. The Ethernet and Wi-Fi based home network which we refer to as the LAN is absent in Continua.

In the PAN domain there are devices that support USB or Bluetooth, and are intended to address information exchange *around a person*. In the LAN domain are devices that support ZigBee, and are intended to exchange information *at a location*. LAN and PAN devices always connect to an

Application Hosting Device (AHD), of which there can be multiple in the home. For ZigBee the AHD is referred to as ZigBee Aggregation Device. The AHD sends data to the Wide Area Network (WAN) and exchanges information with personal telehealth service providers. Via WAN devices information can be exchanged with Healthcare Enterprises and Electronic Health Records. Note that an AHD does not specify a physical device, but specifies interfaces which must be satisfied. Therefore, different kind of devices, like a smartphone, PC, set-top box, or residential gateway can all have the function of AHD.

Figure 9 shows the protocol stack of the AHD in detail. It can be clearly seen how the WAN side supports IP and the messages are translated on the application layer from IEEE 11073 to HL7.

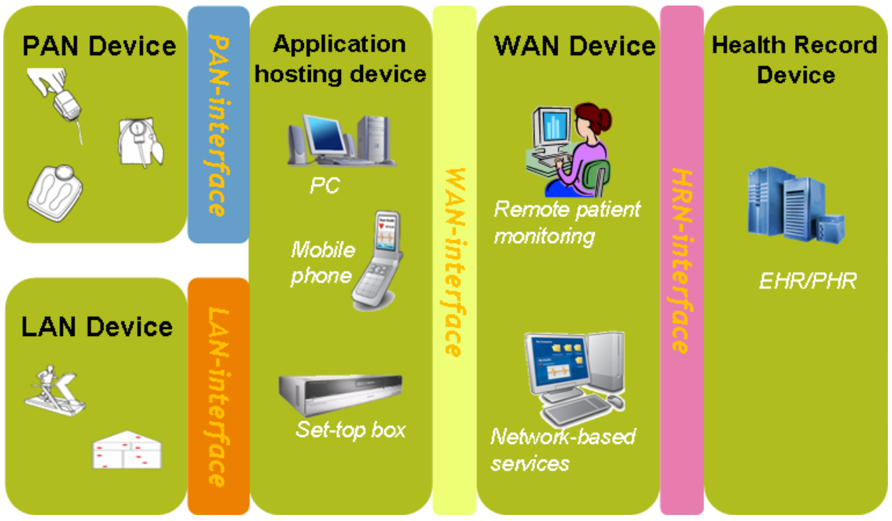

Figure 8. Continua interfaces and domains.

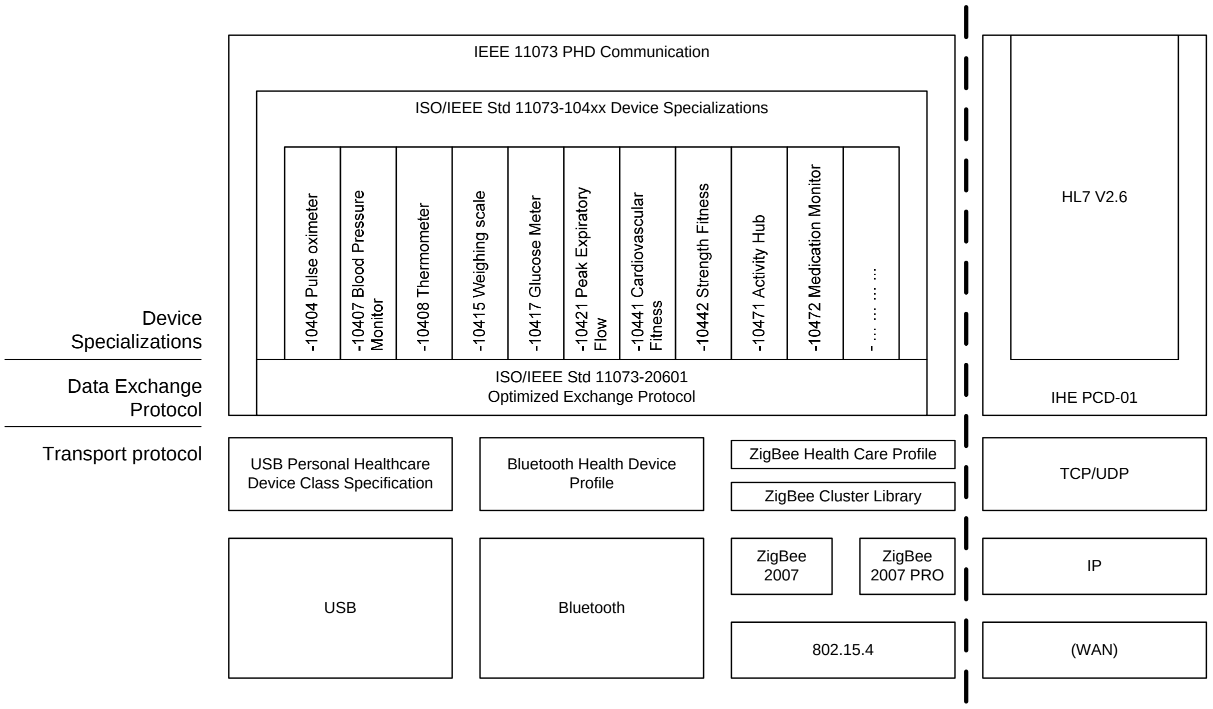
IEEE 11073 PHD Communication
ISO/IEEE Std 11073-104xx Device Specializations
-10404 Pulse oximeter
-10407 Blood Pressure Monitor
-10408 Thermometer
-10415 Weighing scale
-10417 Glucose Meter
-10421 Peak Expiratory Flow
-10441 Cardiovascular Fitness
-10442 Strength Fitness
-10471 Activity Hub
-10472 Medication Monitor
- … … … …
ISO/IEEE Std 11073-20601 Optimized Exchange Protocol
HL7 V2.6
IHE PCD-01
Device Specializations
Data Exchange Protocol
Transport protocol
USB Personal Healthcare Device Class Specification
Bluetooth Health Device Profile
ZigBee Health Care Profile
ZigBee Cluster Library
TCP/UDP
USB
Bluetooth
ZigBee 2007
ZigBee 2007 PRO
802.15.4
IP
(WAN)

Figure 9. Protocols in Continua v2010.

4.2.2 Integrating Continua into the common service delivery infrastructure

Integrating the Continua end-to-end chain into the common service delivery infrastructure requires the introduction of an interface between the Continua network and the common home network. There are three main points at which interaction between the Continua network and the home network can take place, which are shown in Figure 10 (again, with LAN we mean the non-IP control network, as used in Continua):

1) Direct interaction with the sensor to obtain the measurement data;
2) Interaction with the data concentrator;
3) Interaction with the medical backend services, such as an Electronic Health Record.

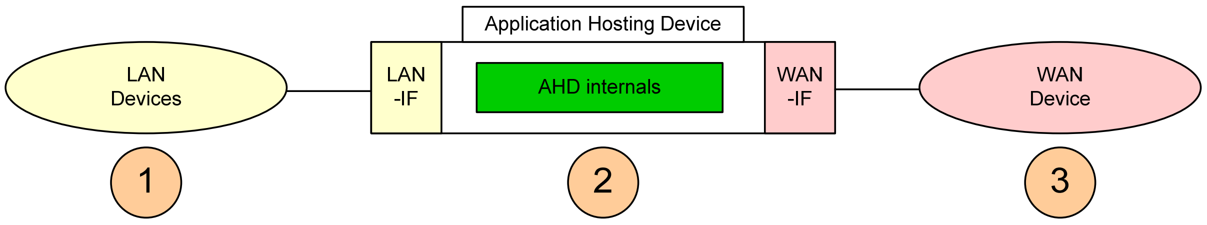

Figure 10. Block diagram of the Continua Application Hosting Device (AHD) and its interfaces

There are several aspects to consider in deciding at which point to interact with the Continua service chain. The aim is to make the measurement data available from LAN/PAN devices to other networks, applications, users, etc. Many measurement devices can only connect to a single AHD. Therefore, by letting a separate entity connect to the sensor, the existing chain would be interrupted. Thus, it is unwise to interact at point 1.

The most logical point is to interact at the next closest point to the measurement devices, which is point 2. This is also a point where IP is supported, since the data sent out to the Continua medical backend uses IP. We assume that the devices used to interact with the medical devices (i.e. collect data or control the device) in the home network are typical generic EUD devices which support IP, such as a PC, laptop, tablet, connected TV, or a smart phone connected via Wi-Fi. Interacting at point 3 is not interesting because it makes the system dependent on the offered medical services by the backend systems.

Similar to the proxying of Bluetooth networks with UPnP as described in the previous section, we now can also proxy the Continua network to UPnP by adding the UPnP stack to the AHD and defining a proxy functionality. This is shown in Figure 11. The WAN side of an AHD should be extended with a UPnP stack, and the IEEE 11073 device specializations which the AHD discovered and registered from the non-IP network should be converted into UPnP services. We intend to elaborate and validate this architecture further and contribute the results to the Continua Alliance as well as the UPnP Forum.

4.3 TR-069 and UPnP DM

For remote management we follow and use the developments made in Broadband Forum, HGI, and UPnP Forum. The work done in these forums has been initiated by the FIGARO partners TNO and Philips, and already largely follows our design principles. For remote management of the residential gateway we choose TR-069. For remote management of other devices in the home we choose either TR-069 (typically set-top boxes and NASs) or proxy TR-069 to other management protocols in the home network. One of those other management protocols is actually a UPnP Device Control Protocol

(DCP), namely UPnP Device Management (DM). We recently designed and developed a TR-069 / UPnP DM proxy [17]. Its architecture is shown in Figure 12.

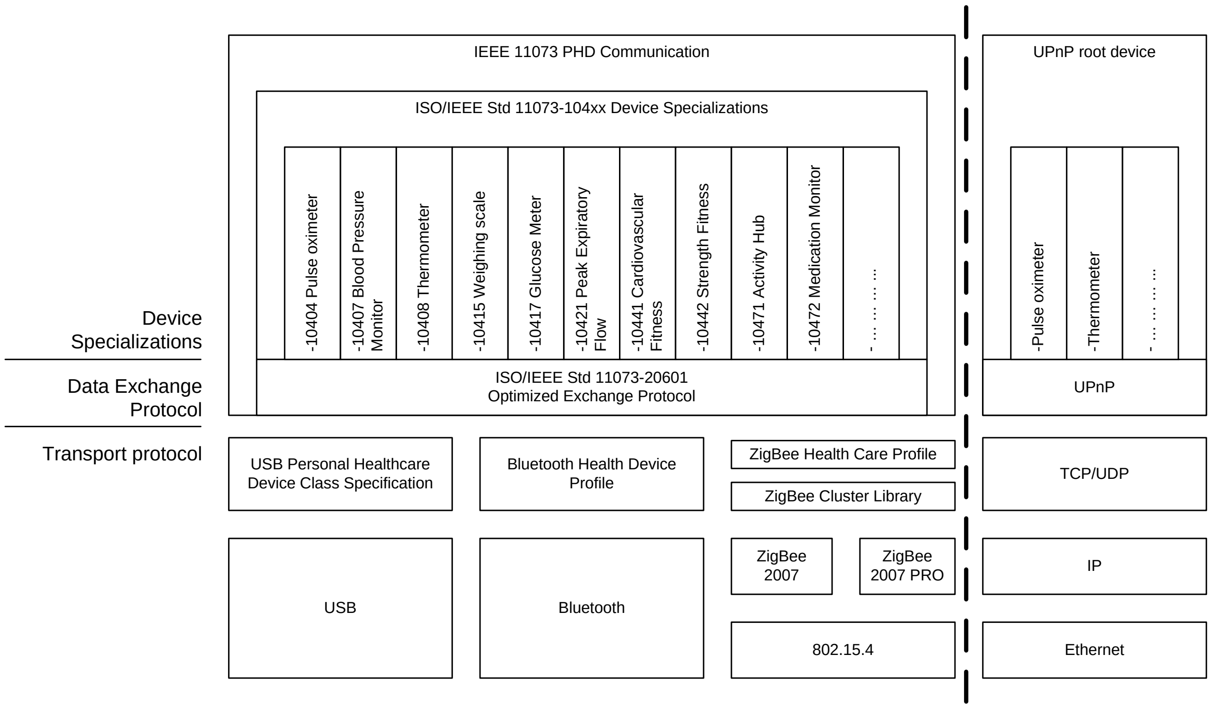
IEEE 11073 PHD Communication
ISO/IEEE Std 11073-104xx Device Specializations
-10404 Pulse oximeter
-10407 Blood Pressure Monitor
-10408 Thermometer
-10415 Weighing scale
-10417 Glucose Meter
-10421 Peak Expiratory Flow
-10441 Cardiovascular Fitness
-10442 Strength Fitness
-10471 Activity Hub
-10472 Medication Monitor
- … … … …
ISO/IEEE Std 11073-20601 Optimized Exchange Protocol
UPnP root device
-Pulse oximeter
-Thermometer
- … … … …
UPnP
Device Specializations
Data Exchange Protocol
Transport protocol
USB Personal Healthcare Device Class Specification
Bluetooth Health Device Profile
ZigBee Health Care Profile
ZigBee Cluster Library
TCP/UDP
USB
Bluetooth
ZigBee 2007
ZigBee 2007 PRO
802.15.4
IP
Ethernet

Figure 11. Protocols involved in the integration of a Continua network into the common service delivery infrastructure

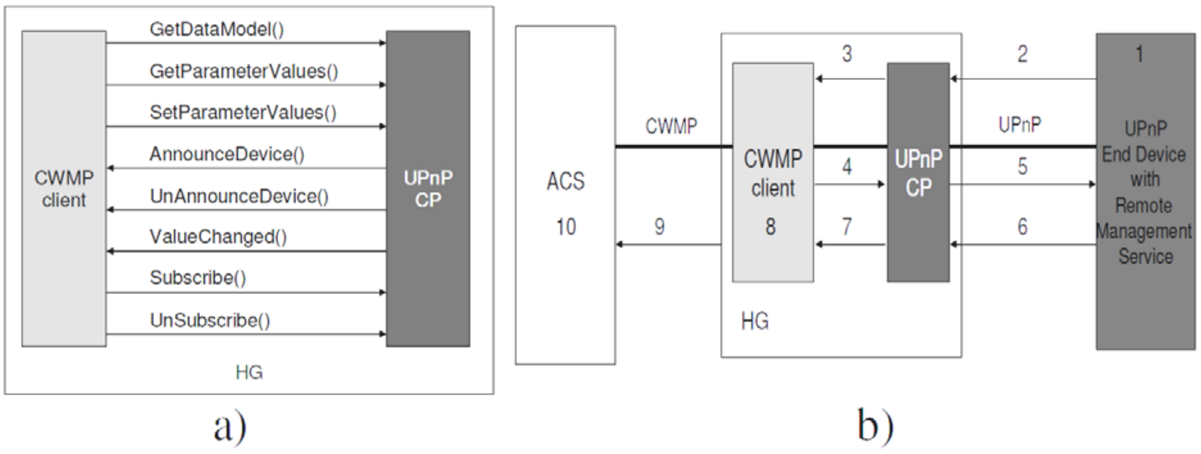

Figure 12. a) API between the residential gateway's UPnP Control Point and the TR-069 CWMP (Customer premises equipment Wide area network Management Protocol) client in our TR-069 / UPnP DM proxy. b) Turning the 2-step UPnP discovery and description process into a single CWMP description [17].

4.4 OSGI, Android, and HGI-RD008-R3 virtualization

For virtualization of the residential gateway we selected OSGi [12], implemented as suggested by HGI in HGI-RD008-R3 [18]. Other devices in the home may be virtualized with OSGi also, but the use of Android is more likely here. Many architectures of OSGi abstraction of home networks are available in the literature, such as for UPnP [19] and Zigbee [20].

4.5 DLNA, Android, and HTML Remote User Interface.

DLNA defines a standard for implementing remote user interfaces (RUIs). In this case a device is implemented that describes which user interfaces are available and what their capabilities are. A control point can discover this device, find a suitable user interface, and set up a connection. In the UPnP standard the actual RUI protocol is out of bounds. This means that UPnP sets up a connection, but does define which remote UI protocols to use (and video file formats are not specified). DLNA however defines the use of CE-HTML (CEA-2014).

A proxy gateway can contain an HTML server to export user interfaces to control nodes in the control network. In this way the devices in the control network can remain very simple. A thermometer just sends messages containing the current temperature. The application on the proxy gateway parses the messages and generates an HTML page. To make sure that the user does not need to enter the URL, the gateway can run a DLNA RUI device which can be automatically discovered by clients in the network.

On a controller that uses or controls devices in the IP home network and non-IP control networks, an Android app must implement a UPnP Control Network (CN) control point. This control point discovers the presence of one or more control networks in the home. The control point can invoke functions calls (actions) on the UPnP CN proxy devices to send messages to nodes in the control network, and it can subscribe to GENA events to receive messages from these nodes. In this way an Android app that can display the temperature in the room, should first detect control networks, query which devices are in these networks, request to receive updates from a temperature sensor, and display the results in the user interface.

If the gateway has an HTML server to export web pages that allow controlling nodes, any web browser can be used to display the results. To show the current temperature all that is needed is the URL of the temperature page. To prevent the user from having to enter a URL, the client can automatically discover the RUI device, and select the appropriate user interface from a list.

4.6 Overview of a typical implementation of a common service delivery framework for residential networks

Figure 13 shows an overview of the preferred FIGARO implementation of the common service delivery framework for residential networks. All functionality shown is described in previous sections, except for DLNA Remote Access.

DLNA Remote Access describes a standardised way to access DLNA devices from outside the home. It is based on setting up a VPN tunnel. The gateway allows a VPN connection to be made from outside the home, and runs a Remote Access server. This server is responsible for the communication over the tunnel, and offers control over which devices are visible outside home. This is implemented by sending a list of accessible devices over the tunnel. At the client side the list of devices is used to generate SSDP messages. This allows UPnP control points in the client network to discover the devices in the other network. Control points can then interact with the remote devices as if they were local. Since SSDP messages are not sent over the tunnel, only the selected devices are visible, which offers a convenient filtering mechanism. However from a security perspective this adds little extra protection after the VPN tunnel has been set up.

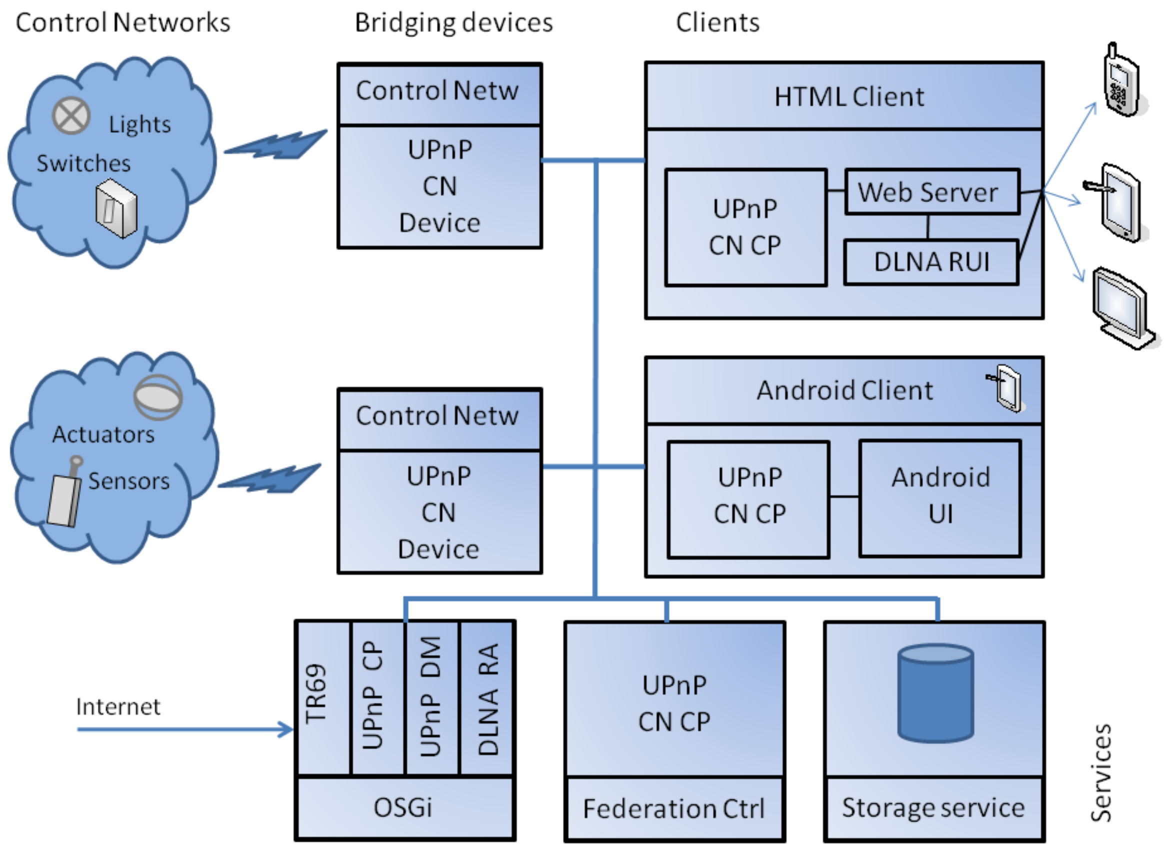

Figure 13. Overview of the preferred FIGARO implementation of the common service delivery framework for residential networks.

5 EVALUATION

In Chapter 2 we presented general principles to guide the definition of the architecture, and introduced the concepts of a common service delivery infrastructure and control networks. In Chapter 3 we defined an architecture based on six main building blocks: addressing, discovery, description, remote management, virtualization and user interfacing. In Chapter 4 we presented our preliminary implementation choices, and showed that relevant gaps in technology exist in, amongst others, UPnP based control network proxying and the interfacing with Continua e-health devices.

In this chapter we describe to what extent the architecture complies with the general design principles, i.e. the stakeholders’ concerns, in terms of what is defined in IEEE 1471.

5.1 Loosely-coupled integrated control using information brokering.

Based on OSGi, a system as described by the architecture of Figure 4 offers a service-based approach in which components can be installed and services discovered. In such a system the application tries to detect the remote presence of a certain service. If the service is available it can be used. If the service is not available, the application needs to be designed in such a way that most of its functionality is still available to the user.

The UPnP approach allows components to be implemented on multiple UPnP devices. Applications detect the presence of these devices and applications can register themselves to receive events from multiple devices. Multiple networks are represented as multiple UPnP control network proxy devices. Each network allows its nodes to communicate within the network. However, when messages from nodes are used outside the scope of the control network, a proxy gateway is responsible for the coupling. The gateway sets up a connection between multiple networks, provides a control mechanism, and acts as an information broker: home services may publish events or data, other services may use the data. Such mechanism can be used to verify if a node is entitled to receive messages, and to make nodes discoverable if they are in a sleep mode. The latter has already been applied in the UPnP Low Power DCP [6].

5.2 Distributed connection management for federated gateways

This principle ensures that the gateway does not become a single point of failure. When using a proxy gateway to set up a connection between nodes within a Zigbee network and an IP network, these nodes should be able to provide their basic functionality even when the gateway is not available anymore. When messages must be passed from the control network to the IP network, the device that hosts both physical interfaces obviously is required. If this device fails or is powered off, messages cannot be passed from one network to the other.

The current architecture offers flexibility in deciding which devices are used to implement the common service delivery functionality. The proxy gateway could, for example, be implemented in the residential gateway, but also as a dedicated proxy device. A still to be defined federation controller could then automatically discover this device and sets up connections as needed. A mixed approach is also possible. For example, a Zigbee interface can be implemented on the residential gateway, while another Zigbee or Bluetooth network is represented by another dedicated proxy device in the home network. If there is a redundancy of proxies in the network, a still to be defined federation controller may pass the proxy role from a failing device to another one, using some selection criterion.

5.3 Security and privacy

Security and privacy need to be taken into account when defining the architecture from the very beginning to ensure that solutions can be better integrated from scratch. The use cases show a wide range of security and privacy aspects to be taken into account, ranging from critical service provider requirements to convenience for the end user.

How certain is the service provider that information obtained from a device is genuine, and commands sent to a device are actually carried out? For a service provider it will be easier if the smart meter or the Continua AHD is part of a dedicated Zigbee network, were messages are encrypted and only passed to the service providers’ server. In a prepaid energy scenario, for instance, it is clear that tampering with the data needs to be avoided at all costs. In this case a design is needed where the prepaid account information and payment details are on the service providers’ server, and not in the user’s devices. Energy consumption data from the meter needs to be protected and signed, to prevent changing the information that is read.

In many cases content protection and security mechanisms conflict with ease of use. For the user it makes sense that the smart meter can also be read from an app on a tablet, to present a graphical overview of the energy consumption using a nice-looking energy app. For products with a Zigbee interface, often a button needs to be pushed on the network proxy device. This instructs the Zigbee proxy device to allow any device to be added to the Zigbee network for a short period of time. While this is a simple mechanism to assure that the new device is not accidently connected to a neighbour’s Zigbee network, users already consider this a hassle while from a security aspect the network is still somewhat vulnerable. If the user has bought a connected alarm clock it should connect to the network automatically, and the user should not worried about a virus changing the wakeup time.

Currently the home networks main protections are WPA encryption for Wi-Fi, and a firewall on the gateway. UPnP has defined a security system. However to date all DLNA media renderers on the market are completely open. A DLNA media renderer device announces itself in the network, and will play any content passed to it completely unprotected. Most people consider the protection by a firewall on the residential gateway sufficient. However once a hacker is in the home network most devices are completely openly accessible. While setting a new alarm time on an alarm clock or turning lights on or off would be annoying, the stealing of health and energy data have much more serious implications. In the future, when even more devices in the home are connected, an additional layer of protection will be needed. This could be achieved by adding a separate security component to our common service delivery infrastructure architecture. This component and its functionality should then be implemented on most devices in the home network, and should be easily configurable to set up the security needs such that an acceptable balance between user friendliness and protection is reached. This will be different from use case to use case and from service provider to service provider. FIGARO does not focus on further development of this functionality, but is dedicated to designing the common service delivery layer such that addition of the security functionality should be straightforward.

Privacy is another aspect. It seems many users are willing to consider measures to protect their privacy, while others trust that their information will not stand out in between information from millions of other users. Some users are concerned about their personal data when considering storage in the cloud. Especially for e-health information and controlling important devices in the home a high security level is required. Precise laws/legislation may be in place that may differ from country to country and that render the overall scenario even more complex to be dealt with. In other some cases a simple logging may provide enough confidence to the user. The user may want the option to check, for instance, if a smart meter that is supposed to be read by the service provider on a monthly basis is not actually read once per hour. Such functionality may be added to the common service delivery framework, but is once again not within the focus of FIGARO.

5.4 Graceful degradation of services

The “graceful degradation of services” requirement is strongly related to what is presented in Sections 5.1 and 5.2, i.e. loosely-coupled integrated control using information brokering and distributed connection management for federated gateways. The distributed approach of our architecture should be implemented in such a way that the basic functionality of devices in the home, like for instance the basic control of heaters by thermostats, is still available even if they are not connected because of a communication failure.

The functionality that connectivity adds to a device or service should always be in addition to the device’s or service’s basic function, and not inextricably mixed up with the basic function. A connected bed-side clock should still work as a basic, non-connected bed side clock if the connectivity is absent. Connectivity with the home network and the Internet may add functionality such as automatic clock synchronisation, streaming downloaded mp3 music to wake up with, etc. Another example is a weighing scale that is connected to a Personal Health Record (PHR). If the network connection fails, the user should still be able to read his or weight from the scale. Locally collected data can then be uploaded to the PHR at a later time, when the network connection is restored. The requirement of graceful degradation implies, for instance, that a device manufacturer cannot save money by leaving away a basic user interface (e.g. configuration buttons) with the argument that the device will always be connected and thus remotely controllable.

5.5 Support for multi-vendor systems

To support multi-vendor systems is the main idea behind designing a distributed common service delivery architectural layer in the first place: to ensure broad uptake and exploitation of new devices and services. Provided that the functional modules in such a layer are well standardized and their interfaces open, the common service delivery infrastructure makes it possible that many different service providers reuse the modules and thus share the home network successfully, even if it contains many different devices from many different vendors. Not only does this enable many new services, it also provides the resident with flexibility to choose a service provider to his liking. Our architecture and implementation choices illustrate the strong demands that multi-vendor support puts on standardization.

6 CONCLUSIONS AND FUTURE WORK

In this deliverable we defined a common service delivery layer for residential networks, also called home networks. We described a preliminary architecture for this layer containing six standardized functions, namely addressing, discovery, description, virtualization, remote management and user interfacing. The first three functions are deemed to be mandatory for every device in the home network, and the other functions are considered as optional, depending on the specific device and use case. We also defined proxy functionality between the parts of the service delivery layer that are built on IP and the parts that rely on non-IP control networks. We have shown that such architecture fulfils the generic guiding principles that follow from our use cases as described in D5.1 [1]. Finally we investigated in which ways the service delivery layer can already be implemented with current technologies, and defined a number of gaps where the needed technology is not yet available. We have also introduced some of the solutions that we designed to overcome these gaps. They are mainly in the field of IP/non-IP proxying of service delivery functionalities.

We have not yet matched the architecture with the requirements set forward by the individual use cases as described in D5.1 [1]. This is planned as future work. Especially the energy management use cases need additional work and resources. We expect that also for these use cases and related control networks, proxy functionality is needed as described in Chapter 4. Subsequently we will select a use case or a combination of use cases to be implemented, such that the strengths and weaknesses of our architecture are most clearly demonstrated.

Other future work lies in the area of interconnecting the home network with the external service delivery frameworks in the public sphere. Special attention should be given to the integration of the work described in this deliverable and the work done elsewhere in the FIGARO project on external federation of residential gateways and content management. This may lead to extra functional blocks in our architecture. We also intend to look in more depth at the interfacing with service provider’s platforms. For instance, the current Continua architecture only expects traffic to be tunnelled directly to the Health Record Network. This means that this stream cannot interact with other possibly useful common service delivery blocks present in the cloud. For instance, with the current architecture, a Continua AHD can only be remotely managed with a management server in the domain of the health care provider. This work cannot be outsourced to for instance a network operator using TR-069.

Finally we expect to dedicate significant efforts in further detailing and developing new UPnP and TR-069 functionality as described in this document, and contribute through this work to the relevant standardization bodies [1]. This does not only concern the UPnP Forum and the Broadband Forum, but also HGI, DLNA, the Continua Alliance, and energy management forums. As planned, a final FIGARO architecture for service federation in residential networks will therefore be presented in subsequent deliverables.

REFERENCES

[1] A. Delphinanto, A.M.J. Koonen, F.T.H. den Hartog, “Improving Quality of Experience by adding Device Resource Reservation to Service Discovery Protocols”, Proc. of 2008 IEEE International Conference on Communications (ICC 2008), Beijing.

[2] FIGARO project, “Deliverable D5.1: State-of-the-art of energy management, e-health and community-service requirements on common service delivery frameworks”, March 2011.

[3] ANSI/IEEE Std 1471-2000, “Recommended Practice for Architectural Description of Software-Intensive Systems”.

[4] Thomas Erl, “Service-Oriented Architecture: Concepts, Technology, and Design”, Prentice Hall, ISBN 0131858580 (2005).

[5] Nico Baken, Freek Bomhof, Erik Fledderus, Frank den Hartog, Annemieke de Korte, Jan Wester, “The art of smart living”, published by TNO (2009).

[6] DLNA: http://www.dlna.org/

[7] UPnP Forum: http://www.upnp.org/

[8] http://en.wikipedia.org/wiki/Universal_Plug_and_Play, 31 August 2011.

[9] Dimitri Papadimitriou et al, “Fundamental Limitations of current Internet and the path to Future Internet”, Whitepaper of the EC FIArch Group (2011).

[10] Frank den Hartog, Tom Suters, John Parsons, Josef Faller, “SA/CLC/ENTR/000/2008-20: Production of a Roadmap for an integrated set of standards for SmartHouse and systems related to it and an Open Event: Final Report”, CENELEC Project Report Smart House Roadmap (2011).

[11] Zigbee Alliance: http://www.zigbee.org

[12] CEN ISO/IEEE 11073 “Health informatics - Medical / health device communication standards”.

[13] OSGi: http://www.osgi.org

[14] A. Delphinanto, A.M.J. Koonen, F.T.H. den Hartog, “Improving Quality of Experience by adding Device Resource Reservation to Service Discovery Protocols”, Proc. of 2008 IEEE International Conference on Communications (ICC 2008), Beijing.

[15] A. Delphinanto et al, “Architecture of a bi-directional Bluetooth-UPnP proxy”, in Proc. of the 4th Annual IEEE Consumer Communications and Networking Conference (CCNC 2007), Las Vegas.

[16] Continua Health Alliance: http://www.continuaalliance.org/

[17] HL7: http://www.hl7.org

[18] A. Delphinanto, B.A.G. Hillen, I. Passchier, B.H.A. van Schoonhoven, F.T.H. den Hartog, “Remote discovery and management of end-user devices in heterogeneous private networks”, Proc. of the 6th Annual IEEE Consumer Communications and Networking Conference (CCNC 2009), Las Vegas.

[19] HGI-RD008-R3 “HG requirements for software execution environment”.

[20] P. Dobrev, D. Famolari, C. Kurzke, B.A. Miller, "Device and service discovery in home networks with OSGi," IEEE Communications Magazine, vol.40, no.8, pp. 86- 92 (2002).

[21] Young-Guk Ha, "Dynamic integration of zigbee home networks into home gateways using OSGI service registry", IEEE Transactions on Consumer Electronics, vol.55, no.2, pp.470-476 (2009).