xAAL: A Distributed Infrastructure for Heterogeneous Ambient Devices

Christophe Lohr, Philippe Tanguy, Jérôme Kerdreux

DE GRUYTER
Postprint version

J. Intell. Syst. 24(3) (2015), 321–331
DOI 10.1515/jisys-2014-0144

Christophe Lohr, Philippe Tanguy and Jérôme Kerdreux

xAAL: A Distributed Infrastructure for Heterogeneous Ambient Devices

Abstract: Ambient assisted living systems are based on sensors and actuators, with a diversity of network protocols and vendors. This commonly leads to the introduction of gateways or middlewares into the technical infrastructure in order to address interoperability issues. The xAAL framework presented in this paper aims to provide interoperability and to redesign such "gateways" into well-defined functional entities communicating with each other via a lightweight message bus over IP. Each entity may have multiple instances, may be shared between several boxes, and may be physically located in any box. Thanks to the distributed architecture of the system, each home automation vendor may peacefully provide its own xAAL box without revealing details of its technology. Also, several applications may be plugged together on the xAAL bus without getting bored with underlying heterogeneity. Moreover, the management of the dynamicity allows sensors or applications to enter and leave the system at any time.

Keywords: Home automation networks, interoperability, ambient assisted living.

2010 Mathematics Subject Classification: 68, 68M12, 68M14.

1 Introduction

Ambient assisted living (AAL) systems focus on improving quality of life by offering integrated products and services for an enhanced living experience. Technical platforms and proposed solutions may include mobile applications, tablets, connected TVs, plain old telephone systems, fellow robots, home automation devices, innovative physiological sensors, actigraphy, coaching services, serious games, social interactions, etc. However, to provide intelligent services, one needs to access physical devices, sensors, and actuators. As it is not realistic for a service provider to reinvent the needed sensors and actuators, it then has to face to a pragmatic issue: interoperability of underlying existing (and possibly already deployed) technologies.

For instance, an AAL system may propose actigraphy services based on the user activity monitored by existing home automation sensors. Many vendors offer home automation solutions. Small devices are plugged everywhere at home. They talk to each other using dedicated link layers and protocols (specific powerline channels, radio channels, data coding scheme, cryptography key, etc.). There are almost as many protocols as manufacturers (or alliances of manufacturers). The consequence is a strong issue about interoperability: how to make a device from vendor A (e.g., a switch) talk with a device from vendor B (e.g., a lamp), and then, how a smart service may collect data or activate a device of some vendor when it needs to. Let us assume that the user wants health-care services with innovative physiological sensors using yet another communication protocol. Should it be a new system separated from the previous one or could it be integrated with it, and how? These are usual practical questions that ALL systems have to tackle. This is not only a matter of technique; this is also a matter of concepts and design.

The rest of the paper is organized as follows. In Section 2, we investigate the main streams to enable interoperability in AAL systems. Section 3 presents our xAAL solution. Finally, Section 4 summarizes conclusions and outlines future work.

This project has received funding from the European Community's Seventh Framework Program for research, technological development and demonstration under grant agreement No.611366 (PRECIOUS).

2 Interoperability Issues in AAL Systems

2.1 Issues and Existing Solutions in AAL Systems

One of the main issues in home automation systems is the choice of a communication protocol. There are well known protocols (Z-Wave [32], KNX [19], EnOcean [11], etc.) and some of them are widely used; however, their interoperability is not well considered. Therefore, if someone wants to use a sensor using Z-Wave with an actuator using KNX, it is not feasible without a special gateway in charge of the interoperability between both protocols. In the past, it was not necessarily a big problem. Indeed, when a home automation system was installed, sensors/actuators using the same communication protocols were deployed.

The number of different protocols in home automation systems is the result of several factors. First of all, sensors and actuators do not have the same characteristics in terms of the following:

- *Throughput and bandwidth* – Usual central frequency: 433 MHz, 868/915 MHz (Eu/Usa), 2.4 GHz.
- *Range* – Wireless area network, body area network, etc. Devices handling Bluetooth or 802.15.4 do not have the same range.
- Problem of *spectral occupancy*, interference – Occurs especially in the ISM band at 2.4 GHz.
- *Energy consumption* – It is a critical topic that has a deep impact on the PHY and MAC layers. A lot of devices are battery powered, and it is not acceptable to change them every day, month, or year.

Those characteristics have a deep impact on the PHY and MAC layers. On the other hand, sensors and actuators are dependent on application domains. Indeed, for e-health (health care based on electronic communication means) and m-health (mobile health) applications, sensors will use WiFi or Bluetooth 4.0 because they are well deployed in smartphones. In the home automation domain, historical vendors have developed their own solution, e.g., X2D [9] (Delta Dore), to switch lights on and off, or to pull roller shutters up and down. Finally, the business model has a deep impact on the system architecture currently deployed in a house. The consumer is facing a captive market: few providers with solutions that cover every needs, with few chances to add equipment from others. That is why, the common architecture is composed by a network of sensors/actuators speaking the same proprietary protocol and a gateway gathering all the messages. The gateway also plays the role of manager with an administration interface and offers a connection with Internet to develop access via a mobile application.

Nowadays, new services appear, like actigraphy, e-health, preventive care, etc. Health and well-being applications are growing, and new devices are proposed on the market every month. They often use smartphones as access points to communicate first in Bluetooth or WiFi, then on Internet in 3G/4G or WiFi. Those new applications also need to use several data from different sensors using different communications protocols to quantify activity, sleeping quality, and environment to give a specialized feedback to the user in order to advice that person to have a healthy life. One of the ideas of preventive care, like in the PRECIOUS project [28], is to change user behavior or to advice the user to have a healthy life and reduce cardiovascular disease or type II diabetes. To achieve this goal, it appears that a model in charge of analyzing user behavior needs a lot of data from different sources like home sensors, social networks, physiological sensors, activity trackers (wearable devices), etc. In new AAL systems, it is necessary to have a seamless sensor/actuator layer to aggregate a huge quantity of data and to allow easy access to developers of new applications for AAL systems.

The main challenge is the interoperability between devices. Actually, there is a lack of standardization in the home automation systems. In the past 10 years, a lot of initiatives, alliances, or standards appeared, like ZigBee, EnOcean, or Z-Wave; however, they still do not promote interoperability with other alliances or standard. At the same time, the AAL domain has been developed in the research domain, and different solutions have been proposed about the interoperability between

heterogeneous devices. The next section provides a short review of research and solutions proposed to solve the problem of interoperability between heterogeneous devices used at home.

2.2 Proposed Solutions for Home Automation Interoperability

As shown in the previous section, a transparent sensor/actuator layer is needed to propose more ambitious AAL systems. In the literature, we focused our attention on existing solutions in the domains of personal health system architectures, smart home/home automation architectures, smart home, and health system architectures; communications protocol related to the e-health/m-health domain; and communications protocol related to the smart home domain.

Existing solutions have been classified in three categories: protocol of communication, middleware, and architecture/framework. For those three categories, common mechanisms (or functionalities) are used to achieve interoperability. Indeed, most of the solutions have similar components or functionalities that are essential for a system including heterogeneous devices:

- *Addressing* is an essential functionality to send/receive information to/from devices. Several techniques can be employed, e.g., UUID, IP address, etc.
- *Description* gives information about the functions available on a device.
- *Discovery* mechanism gives information to the system about the available devices.
- *Communication protocol* is the way a device shares its functionality or information.
- *Security and privacy* are now a necessity to protect user information.
- *Store/cache* data are essential for sensors that are not queryable, or to verify the state.

Those criteria are used to classify solutions for each of the previous three categories.

2.2.1 Communication Protocols

Standards, consortiums, or alliances are potential solutions to solve the problem of interoperability. Indeed, one can dream of a unique protocol interconnecting every smart devices. The reality is totally different in the smart home domain where there exist several initiatives (e.g., ZigBee, EnOcean, etc.). On the other hand, in the e-health/m-health domain, other protocols or standards are used, e.g., CEN ISO/IEEE 11073, ZigBee alliance [34] or Bluetooth. All of these initiatives are trying to solve the interoperability at a low level; however none of them have established themselves. Standards are important to allow a better acceptance of one technology by manufacturers, vendors, and users. Like that, interoperability issues can be reduced between smart devices in the same domain. However, in smart home or e-health domains, it is not feasible to have one unique protocol for gathering data from sensors. Business model, frequency, range, throughput, power consumption, physical size, and cost are reasons to explain the diversity of communication protocols.

Solutions about communications protocols can be separated into two subcategories: low level and high level. In the first one, we have a communication protocol defining the PHY and MAC layers, and is generally dedicated to devices with constrained resources. The most popular protocols in the smart home domain are KNX, EnOcean, Z-Wave, and ZigBee/IEEE 802.15.4. In the e-health and m-health domains, there are Bluetooth, ZigBee, ANT/ANT+, etc. In the second one, only communications protocols defining upper layers (e.g., MQTT, XMPP, CoAP, or UPnP) are considered. In Reference [29], Vasseur and Dunkels explain why interconnecting smart objects with IP is a good solution to develop interoperability. However, it is currently not feasible to deploy a full IP stack on small and smart devices. That is why a compromise has to be found between low-level networks focused on power consumption, physical size, and cost, and high-level protocols. Several solutions trying to solve interoperability issues with high-level protocols have been identified, as shown in Table 1. For example, xPL/xAP is widely used in home automation. It is based on a simple text-based protocol (JSON like) and a single transport (UDP broadcast). However, xAP mixes type and address, and both protocols use a non-standard text format. Reciprocally, UPnP

Table 1: Protocol of Communication Related to Smart Home Domain

Protocols	Description	Discovery	Com. sublayer	Event	Security
xPL/xAP [31]	Type based	Heartbeat broadcast	UDP broadcast	UDP broadcast	×
UPnP [12]	XML	SSDP	SOAP	GENA	TLS
DPWS [24]	WSDL	WS-Discovery	SOAP/HTTP, SOAP/UDP	WS-eventing	WS-Security
CoAP [17]	URI	Built-in	UDP, SMS, TCP (possible)	Asynchrous notification	DTLS
MQTT [20]	MQTT topic	Pub/sub	TCP/IP	Pub/sub	SSL
AgoControl [1]	Schema	Device announce	AMQP	✓	SSL (AMQP)

Table 2: Middleware Related to Smart Home

Architecture	Type	Description	Discovery	Communication
Amigo [3]	SOA, OSGi-based	✓	✓	✓
SM4ALL [26]	SOA, OSGi-based	✓	✓	✓
Hydra [16]	SOA	Device manager	Discovery manager	P2P, SOAP tunneling, JXTA backbone
EnTimid [21]	OSGi-based	Component model	×	Message oriented (Pub/Sub)
universAAL [27]	OSGi-based	✓	✓	UPnP, R-OSGi

is used in home automation products and use a standard implementation based on a strong and clear specification. However, it uses a lot of protocols, events are hard to tailor, it is generally too big to fit in embedded devices, and there are some problems concerning security.

Communications protocols defining upper layers than IP are an interesting solution to fight interoperability issues. They allow choosing the network topology, i.e., distributed or not. They allow flexibility, and the network can be extended easily. They do not impose one specific programming language. However, they need to develop gateways to integrate all existing sensor/actuator networks. Finally, a solution based on a communications protocol needs to be widely adopted to guarantee the success of the solution.

2.2.2 Middlewares

Middleware solutions are attractive because a software layer between hardware and applications offers dynamicity and flexibility. The middleware domain is wide with different kind of solutions like SOA-based [3] or model-driven [21] solutions. Several existing middlewares dedicated to AAL have been compared in Table 2. Another review of middleware solutions used for a smart home can be found in Reference [30]. The Amigo project [3] defined a middleware to provide a transparent layer related to heterogeneous devices. Four categories of devices have been identified and classified according to their resources. For example, the Legacy Amigo device cannot embed the Amigo middleware compared with Full Amigo home automation devices. Resourceconstrained devices in home automation with their own proprietary protocol are still an issue because it is not currently feasible to embed a middleware layer. Similarly, the SM4All project [26] defines a pervasive layer where functions of sensors/actuators and devices are exposed as services. Like in Amigo, small devices in terms of resources do not use the SM4ALL middleware, and the interoperability between small devices is realized in one gateway, thanks to the OSGi framework. In 2010, the Hydra project [16] proposed a middleware usable in three different vertical domains: home automation, e-health, and agriculture. Like Amigo, Hydra separates the devices in non-hydra-enabled devices, hydra-enabled devices, and gateways. As we can see, resource-constrained devices are not well adapted for middleware solution. Consequently, the interoperability between devices is often realized by one gateway using the OSGi framework and centralizing all the home automation networks.

More generally, middleware initiatives did not well consider the communication between instances on different nodes (devices). More recently, the universAAL project [27] proposed a middleware (based on the PERSONA middleware [4]) solution dedicated to new services and deployment solution in the AAL domain. The middleware allows communicating between several uAAL nodes,

Table 3: Architecture Related to Smart Home

Architecture	Type	Description	Discovery	Communication
Web-of-things [15]	web-based	Metadata model, REST+microformats	Local lookup and discovery service	web standards
openHAB [22]	OSGi-based	×	×	Non intrinsic
DOG gateway [5]	OSGi-based	Ontology	✓	API, XML-RPC
IoT-A [18]	Ref. model	×	×	Rules to build interoperable stacks
BUTLER [8]	OSGi-based, API	✓	REST/JSON	Java API (same HW device) or HTTP
Eclipse M2M project [10]	Framework	×	×	MQTT, M3DA

thanks to the peering component. Then, communication can be established between uAAL nodes using the same protocol stack like UPnP or R-OSGi in the so-called AAL space. However, the interoperability between home automation devices is still managed by OSGI in one gateway. Consequently, the distributed approach of UNIVERSAAL is lost for resourceconstrained devices.

2.2.3 Architecture/Framework

Some solutions provide a whole architecture or a framework. Indeed, AAL systems providing more elaborate services cannot be only built around a protocol or a middleware. Most of the time, they define gateways, servers, services, and communication protocol between all components inside the architecture. Several architectures dedicated to AAL systems are compared in Table 3. According to the table, OSGi is often used to manage interoperability between heterogeneous devices. Home automation protocols are implemented in OSGI bundles, which allow the interoperability between devices in one gateway. The main drawback is that the network topology is fixed. Indeed, the gateway centralizes all the low-level protocols in the same physical place. Another approach, called Web-of-things, has been proposed by Guinard in Reference [15] and consists in using Web standards to manage the heterogeneity between Internet-of-thing devices. Therefore, the main idea is to bring wireless sensor networks to the Web and use Web standards together with a RESTful architecture. Indeed, Guinard argues that if Web standards have proven in the past that they can connect millions of heterogeneous computers, it may be a good idea to do the same with Internet-of-things.

An architecture allows horizontal interoperability between domains: smart home, e-health, m-health, smart transport, and smart city [8, 18]. To manage interoperability, several options are offered to a designer of AAL systems: a unique protocol, a middleware, or a hybrid solution. An architecture needs to deploy several software/hardware components. In that case, designers need to know several technologies (server, gateway, standards, etc.) and different protocol of communications. The deployment of those solutions is generally a task that needs several skills. Finally, complex architectures and systems involve problems in terms of maintenance and reliability.

2.3 Findings of the Survey

Solutions exist to address the interoperability issues. Almost all solutions play around the idea of a socalled *gateway* often implemented with middleware: a box (a small computer) is equipped with two or more modules, each module talking in a given home-automation protocol; then, a piece of software over those modules make it possible to forward messages between protocol A and protocol B. Moreover, those boxes embed additional functions to make all of these usable: HMI (via Web and/or smartphones), configuration facilities, connectors to extra services in the cloud, etc. Such solutions are partial from the interoperability point of view, as each box manages a limited number of low-level protocols, and boxes are hardly able to talk to each other.

There is also a key point about applications. Thanks to middleware approaches, those boxes are also designed to be the dedicated place to run applications. Boxes are not only the gateway for low-level physical devices, but they also act as a hub for applications and smart services. Everything that

should cover all needs is in the same box: home automation, alarm, telemedicine, actigraphy, etc. Thus, the box provider should provide everything by itself, or should ask (beg) third parties to offer components for its box, based on strong specifications of means to plug such add-ons. Brønsted et al. [7] noticed that "with an approach using a central server, it is more difficult to obtain a balanced business relationship between market actors since one actor is likely gain dominance over the server."

One should also consider sporadic devices. For instance, a newly connected TV may embed applications, which is an opportunity. A TV is not supposed to be mobile, but it may be on or off depending on the time of the day. Moreover, mobile devices such as smartphones or tablets are nowadays common. This may be interesting from the AAL point of view: they embed sensors relating to user activity, and they can run applications. However, such devices may enter or leave the system at any time. This poses questions about means to interconnect such mobile devices to the above-mentioned boxes.

Starting from those basements, we decided to reorganize and formalize the architectures of those boxes, which are in fact all composed of the same functionalities. Therefore, we propose to split functionalities into well-defined functional entities, communicating to each other via a message bus over IP. Each entity may have multiple instances, may be shared between several boxes, and may be physically located in any box. Thanks to the distributed architecture of the system, the xAAL solution can address interoperability. Each sensor's family, each vendor, may peacefully provide its own xAAL box without revealing details of its technology.

3 Descriptions of the xAAL Solution

xAAL is basically a distributed architecture. A set of functional entities is defined. Those entities may be present several times, be implemented separately and/or grouped into hardware components of various manners, and talk to each other via a message bus.

3.1 Functional Architecture

Figure 1 shows the general functional architecture of the xAAL system.

Native equipment: Some devices (sensors, actuators) can communicate natively using the xAAL protocol.

Gateways: In general, home automation devices only support their own proprietary communication protocol. Gateways translate messages between the manufacturer protocols and the xAAL protocol. This is the very classical function found in all other home box solutions. This part is very specific to manufacturers' technologies. It may be the role of vendors to provide it by themselves. All other functions may be shared.

Schema repository: Home automation devices are described by a schema (the nature of the equipment, how to interact with it). These schemas are listed in one or more schema repositories that can be queried by other entities. Those repositories may be fed with specific schemas describing the vendor's devices. Some usual schemas are generic.

Database of meta-data: Each automation device within a facility is likely to be associated with a piece of information: at least location information (e.g., declare that the equipment typed as a lamp and having address X is located in the kitchen) and possibly a symbolic name (e.g., device having address X is called "lamp#1"). This information is stored as a set of tags in one or more databases of meta-data. Those databases somehow contain the configuration of the home automation installed.

Cache: Some sensors can just send information in a sporadic way (e.g., a thermometer that sends the temperature data only if it changes). One or more caches store this information (with a timestamp) so that other entities can query them whenever necessary.

Automata of scenarios: Scenarios are advanced services like, for example, start a whole sequence of actions from a click of the user, or at scheduled times, or monitor sequences of events and then react, etc. To support it, one may have one or more entities of the type *automata of scenarios*. AAL services usually imply some actions to be triggered locally close to the user at home, and some

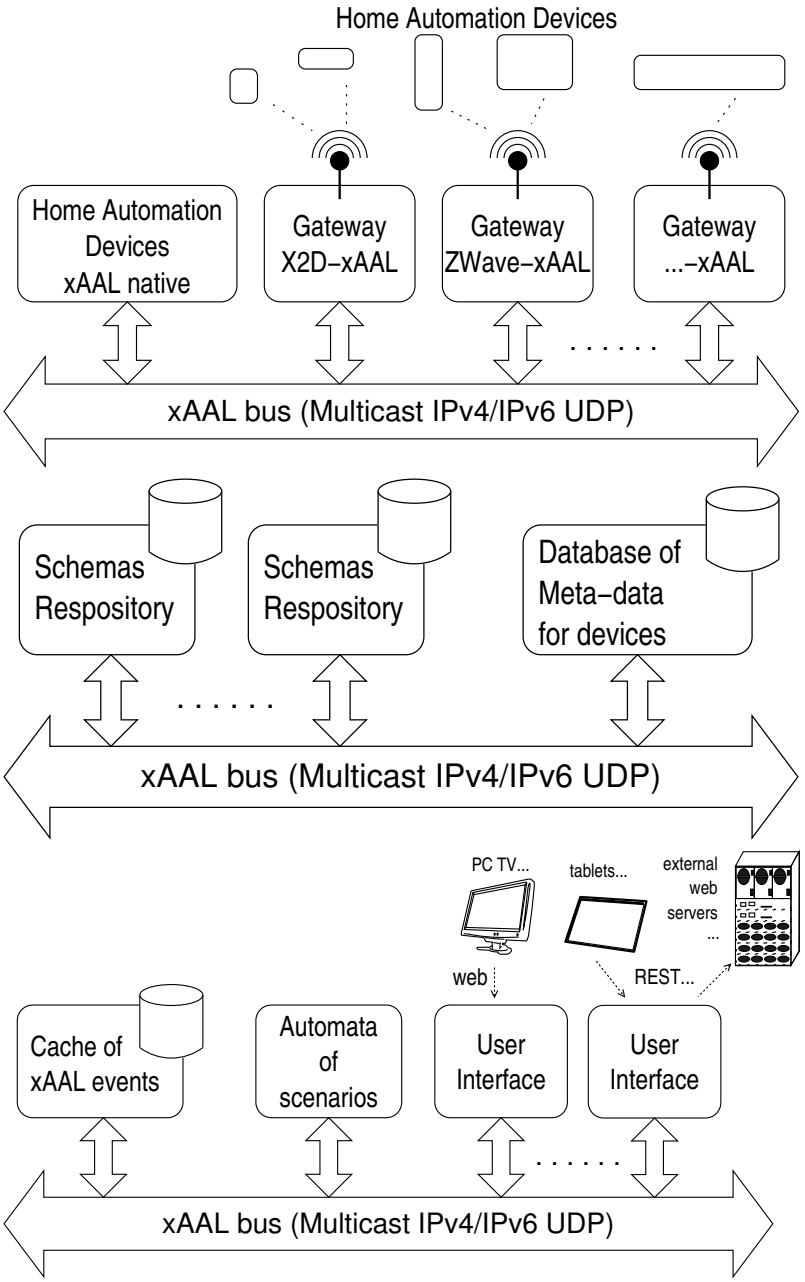

Figure 1: Functional Architecture of xAAL.

other actions to be computed somewhere in the cloud. Automata of scenarios are dedicated to the former.

These automata of scenarios are also the right place to implement virtual devices: for example, a scenario could aggregate and correlate a variety of events from real devices, and then synthesizes information such as "presence" and notify it on the bus, in order to be used by other entities. By proceeding in this way, this scenario should appear by itself as a device on the bus, with its address and its schema. This scenario is a kind of a virtual device.

User interfaces: One or many user interfaces are provided by specific entities connected to the xAAL bus. This can be a real hardware device with a screen and buttons, or a software component

that generates Web pages (for instance) to be used by a browser on a PC, a connected TV, or other devices; or software that provides a REST API for mobile applications (tablets, smartphones), to an external server on the cloud for advanced services, to an MQTT server, or to offer features for service composition, etc.

The typical workflow of an HMI could be as follows: (i) query the bus to be aware of the present devices; (ii) query the schema repository for the type of these devices, the type of data from the sensors/actuators, the possible actions, and URLs to retrieve icons; (iii) query the database of meta-data about devices to get the configuration of the installation (a friendly name to display, location, and other tags associated with each device); (iv) query the cache for the latest known states of such equipment; and, finally, (v) dynamically build display screens and control interfaces for these devices.

3.2 Devices and Schemas

Devices are identified by a unique address on the bus. This is a UUID (RFC 4122): the textual representation of a 128-bit random number. Address 0 is reserved for broadcast and refers to all devices. Device address may be assigned in the factory, or autogenerated (random) at installation time. However, it is recommended that this address remains persistent, e.g., after power breaks.

A device has a type, i.e., a schema name. This name is a string consisting of a pair of words separated by a dot. The first word refers to a class of device type (e.g., lamp, heating, multimedia, etc.). The second word refers to a specific type in a given class (e.g., within the lamp class, one may have lamp.basic, lamp.trigger, lamp.dimmer, etc.) The second word may also refer to a schema extension by a manufacturer. The identifier "any.any" is reserved and refers to all types of all classes.

The schema associated with a device provides semantics to a device and describes its capabilities: a list of possible methods on this device (mechanism of request-response), a list of notifications that it sends spontaneously, and a list of attributes (a device announces any change of values on the bus). Those schemas are inspired by the UPnP approach [12].

There is a notion of inheritance between schemas. A schema can extend an existing schema. We define the first three levels of this genealogy: (i) a generic schema common to all existing devices worldwide that everyone has to implement; (ii) a specific schema for every class (e.g., lamp, thermometer, switch, etc.) – such specific schemas thus inherit from the generic schema; and (iii) a specific schema for each type that inherits from a class schema, which indicates the level of functionality (e.g., a lamp type can be on/off, some are of a trigger type, and others with dimmer, etc.). Thereafter, manufacturers of home automation equipment will naturally define their own schemas among their product range that will inherit of schemas of level three.

3.3 Messages

Messages are transported by an IP multicast bus (IPv4 and/or IPv6). As proposed by UPnP [12], a bus allows the discovery: when a new component appears in the installation, it announces itself. All other entities can then take it into account. Similarly, when a new component enters, it can query the bus to discover the other components already present. This greatly facilitates the configuration and allows dynamicity. However, UPnP use the multicast bus only for discovery, not for transmission of data or control/command that therefore requires a large set of point-to-point TCP communications between UPnP devices (plenty of call-backs to get notifications).

We choose JSON [13] as the exchange format on the bus. Many libraries are available in many development environments. A message consists of a header and a body. The header is mandatory, while the body is optional. Figure 2 gives an example of xAAL messages. The header of a message includes the source and destination addresses of the message. A destination address of 0 means a broadcast message: everybody is invited to filter the relevance of the message based on the device type (name of the schema). Note that device type mentioned by the requester may contain the keyword "any" to widen the broadcast request. The body contains parameters and values for

```
{
 "header":
  {
   "version": "0.2",
   "source": "bd081741...9657",
   "target": "83c84e87...65a0",
   "devType": "thermometer.queryable",
   "msgType": "request",
   "action": "getAttributes",
   "cipher": "none",
   "signature": ""
  }
}
```

```
{
 "header":
  {
   "version": "0.2",
   "source": "83c84e87...65a0",
   "target": "bd081741...9657",
   "msgType": "reply",
   "devType": "thermometer.queryable",
   "action": "getAttributes",
   "cipher": "none",
   "signature": ""
  },
 "body":
  {
    "temperature": 9.3
  }
}
```


Figure 2: Example of xAAL messages

queries, responses, and notifications, if needed. The body is optional: it depends on what has been defined in the schema for the mentioned action.

In the above example, there is no encryption ("cipher:none"). One keeps the possibility of communicating without encryption. However, one may have "cipher:md5" with the signature of the message with a shared key; to avoid replay attacks, a timestamp parameter is added in the message before signing.

3.4 Experiments

Nowadays, a prototype of the xAAL solution is developed in C and in Python. Both implementations [6] are proposed to the open-source community, which is very active on home automation solutions. In the meantime, several experiments were carried out in the form of student projects with the collaboration of some of our industrial partners (mostly small to medium enterprises and startups).

xAAL is deployed in our Experiment'HAAL living'lab. It can manage X2D, KNX, and Z-Wave devices: lights, shutters, door-opening sensors, heating, water flow sensors, pressure sensors, etc. Ad-hoc xAAL-native devices are prototyped using GrovePi hardware platform, which allows connecting Grove sensors/actuators with a Raspberry Pi [14, 23].

We drove experiments to interface a Nao robot (by Aldebaran Robotics [2]) to the xAAL bus of our living'lab. The robot acted like a remote control for the home automation system thanks to its voice recognition facility, and as an alerting device with its gestures and voice. For instance, the user says "Nao, morning scenario please!", then Nao makes a gesture to the direction of the windows and says "I raise shutters." while sending the corresponding xAAL query; then, it does the same to switch off the lights of the bedroom, switch on lights of the bathroom, etc. If the tap water remains open after the user leaves the bathroom, the xAAL system detects it and sends a dedicated notification to Nao that puts its hands on his head and says "Oups! I think the tap water is still open." The idea is to bring a kind of presence near lonely people. The xAAL ecosystem eases developing such smart services.

4 Conclusion

AAL systems are usually based on a set of subsystems that are vendor dependent or technology specific. This leads to interoperability issues. To handle this, several middlewares or gateways have

been proposed, at the cost of extra overhead and constraints between actors involved in an AAL system.

This article has presented xAAL, which provides the same functionalities as existing solutions for interoperability. However, the xAAL solution is more lightweight (only small JSON messages), and, thanks to its distributed architecture, many components may be shared between actors of the market. The use of communication bus over IP allows vendors to freely distribute their products (with an IP socket) without entering within a middleware or a box handled by a competitor. More comprehensive documentation and codes are provided on the project site [6].

Some points of the xAAL proposal have to be further explored. For instance, xAAL schemas are based on the JSON schema dialect [13]. Unfortunately, the corresponding RFC is not yet stabilized; this is still a draft.

A second point is about end-user services. Our first experiments with xAAL are based on classic home automation scenarios. One needs extra experiments involving pure AAL scenarios (actigraphy, health care, etc.). One then may enhance xAAL automation components to become platforms for hosting applications. Approaches extending the white-board paradigm, such as the CSTBox [33], seem promising.

The last point is about the use of assistive robots in the context of AAL services. We drove experiments to interface a Nao robot to the xAAL bus of our living'lab, with success. However, for a better interconnexion between the home and robots in general, xAAL could be interfaced to ROS (robot operating system), as proposed by Roalter et al. [25].

Bibliography

[1] Agocontrol, *Agocontrol Project*, `http://www.agocontrol.com/`, Accessed 20 February, 2015.

[2] Aldebaran, *Nao*, `https://www.aldebaran.com/en/humanoid-robot/nao-robot`, Accessed 20 February, 2015.

[3] AMIGO, *AMIGO European project*, `http://www.hitech-projects.com/euprojects/amigo/index.htm`, Accessed 20 February, 2015.

[4] Michele Amoretti, Giorgia Copelli, Guido Matrella, Ferdinando Grossi and Stefania Baratta, The PERSONA AAL Platform: Deployment in the Italian Pilot Site of Bardi, in: *AALIANCE conference*, Malaga, Spain, 2010.

[5] Dario Bonino, Emiliano Castellina and Fulvio Corno, The DOG gateway: enabling ontology-based intelligent domotic environments, *Consumer Electronics, IEEE Transactions on* **54** (2008), 1656–1664.

[6] IHSEV Telecom Bretagne, *xAAL – Home-Automation Interoperability*, `http://recherche.telecom-bretagne.eu/xaal/`, Accessed 20 February, 2015.

[7] Jeppe Brønsted, Per Printz Madsen, Arne Skou and Rune Torbesen, The Homeport System, in: *Proceedings of the 7th IEEE Conference on Consumer Communications and Networking Conference*, CCNC'10, pp. 754–758, IEEE Press, Piscataway, NJ, USA, 2010.

[8] BUTLER, *BUTLER European project*, `http://www.iot-butler.eu/`, Accessed 20 February, 2015, 2011.

[9] Delta Dore, *Delta Dore website*, `http://www.deltadore.com/`, Accessed 20 February, 2015.

[10] Eclipse, *IoT Eclipse - Services & Frameworks*, `http://iot.eclipse.org/frameworks.html`, Accessed 20 February, 2015.

[11] EnOcean, *EnOcean alliance*, `http://www.enocean-alliance.org/en/home/`, Accessed 20 February, 2015.

[12] UPnP Forum, *UPnP Device Architecture 1.1*, October 2008, `http://upnp.org/specs/arch/UPnP-arch-DeviceArchitecture-v1.1.pdf`, Accessed 20 February, 2015.

[13] F. Galiegue, K. Zyp and G. Court, *JSON Schema: core definitions and terminology draft-zyp-json-schema-04*, January 2013, `http://json-schema.org/`, Accessed 20 February, 2015.

[14] GrovePi, *GrovePi project*, `http://www.dexterindustries.com/GrovePi/`, Accessed 20 February, 2015.

[15] Dominique Guinard, *A Web of Things Application Architecture - Integrating the Real-World into the Web*, Ph.d., ETH Zurich, 2011.

[16] Hydra, *Hydra project*, `http://www.hydramiddleware.eu/news.php`, Accessed 20 February, 2015.

[17] IETF, *Coap: draft ietf core coap*, IETF, Report, 2013.

[18] IOT-A, *IOT-A European project*, `http://www.iot-a.eu/public`, Accessed 20 February, 2015, 2009.

[19] KNX, *KNX Association*, `http://www.knx.org/knx-en/index.php`, Accessed 20 February, 2015.

[20] MQTT, *MQTT Protocol*, `https://www.oasis-open.org/`, Accessed 20 February, 2015.

[21] Grégory Nain, *EnTiMid : Un modèle de composants pour intégrer des objets communicants dans des applications à base de services*, Ph.D. thesis, 2012.

[22] openHAB, *openHAB project*, `http://www.openhab.org/`, Accessed 20 February, 2015.

[23] Raspberry Pi, *Raspberry pi foundation*, `http://www.raspberrypi.org/`, Accessed 20 February, 2015.

[24] Stefan Reichhard, *UPnP and DPWS*, Bachelor thesis, 2013.

[25] Luis Roalter, Matthias Kranz and Andreas Möller, *A Middleware for Intelligent Environments and the Internet of Things*, Ubiquitous Intelligence and Computing, Lecture Notes in Computer Science 6406, Springer Berlin / Heidelberg, 2010, 10.1007/978-3-642-16355-523, pp. 267–281.

[26] SM4ALL, *SM4ALL European Project*, `http://www.sm4all-project.eu/`, Accessed 20 February, 2015.

[27] UniversAAL, *universAAL European project*, `http://universaal.org/index.php/en/`, Accessed 20 February, 2015.

[28] Aalto University, University of Helsinki, Firstbeat, University of Vienna, Telecom Bretagne, Hospital Universitari Vall d'Hebron, Campden BRI and EuroFIR, *The PRECIOUS Project*, `http://www.thepreciousproject.eu/`, Accessed 20 February, 2015.

[29] Jean-Philippe Vasseur and Adam Dunkels, *Interconnecting Smart Objects with IP: The Next Internet*, Morgan Kaufmann Publishers Inc., 2010.

[30] Ehsan Ullah Warriach, State of the Art: Embedded Middleware Platform for A Smart Home, *International Journal of Smart Home* **7** (2013), 275–294.

[31] xAP, *xAP protocol*, `http://www.xapautomation.org/`, Accessed 20 February, 2015.

[32] Z-Wave, *Z-Wave Alliance*, `http://www.z-wavealliance.org/`, Accessed 20 February, 2015.

[33] A. Zarli, E. Pascual and D. Cheung, *CSTBox*, `http://cstbox.github.io/`, Accessed 20 February, 2015, 2014.

[34] ZigBee, *ZigBee Alliance*, `http://www.zigbee.org/`, Accessed 20 February, 2015.

Citation Information

Christophe Lohr, Philippe Tanguy and Jérôme Kerdreux, xAAL: A Distributed Infrastructure for Heterogeneous Ambient Devices, in *Journal of Intelligent Systems*, Volume 24, Issue 3, Pages 321–331, ISSN (Online) 2191-026X, ISSN (Print) 0334-1860, DOI: 10.1515/jisys-2014-0144, March 2015, `http://www.degruyter.com/view/j/jisys.2015.24.issue-3/jisys-2014-0144/jisys-2014-0144.xml`

Author information

Christophe Lohr, Telecom Bretagne, Technopôle Brest-Iroise, CS 83818, 29238 Brest Cedex 3, France.
E-mail: `christophe.lohr@telecom-bretagne.eu`

Philippe Tanguy, Telecom Bretagne, Technopôle Brest-Iroise, CS 83818, 29238 Brest Cedex 3, France.
E-mail: `philippe.tanguy1@telecom-bretagne.eu`

Jérôme Kerdreux, Telecom Bretagne, Technopôle Brest-Iroise, CS 83818, 29238 Brest Cedex 3, France.
E-mail: `Jerome.Kerdreux@telecom-bretagne.eu`