Communication Architectures and Experiences for Web-Connected Physical Smart Objects

Juan Ignacio Vazquez, Jonathan Ruiz-de-Garibay, Xabier Eguiluz, Iker Doamo
Intelligent Environments-DeustoTech
University of Deusto, Bilbao, Spain
{ivazquez, jonathan.garibay, xabier.eguiluz, idoamo}@deusto.es

Silvia Rentería, Ana Ayerbe
Infotech Unit
Robotiker-TECNALIA
Zamudio, Spain
{silvia,anayerbe}@robotiker.es

***Abstract*—We are witnessing a tremendous hype on the Internet of Things paradigm, with not only research projects, but also commercial products claiming to implement its fundamental mechanisms. Smart-connected-objects designers often have to face decisions on the global architecture of the service, since no single solution is valid for all the cases. In this paper, we explore the different criteria for designing architectures for Internet of Things solutions, along with illustrative examples of prototypes that implement these approaches.**

Keywords: Internet of Things, architecture, smart objects, networking

I. INTRODUCTION

The Internet of Things (IoT) is currently enjoying a *golden age* of interest, both from a research and a pre-market perspective. The hype started with the initial popularization of the term in the Scientific American article by Gershenfeld et al. [1] in 2004 and the ITU report in 2005 [2], and nowadays there are dozens of international events and conferences which include the topic among their coverage.

One of the main advantages of the IoT paradigm is that it promotes the creation of very cheap, yet awesome prototypes with little knowledge and experience. This can be done by taking advantage of rapid prototyping platforms such as Arduino, Propeller and Microchip PIC family, and their integration with Internet-ready communication mechanisms.

In fact, one of the main appealing characteristics of IoT prototyping for engineers is that it encourages a balance between interaction, hardware and software skills. Therefore, this is a field where different knowledge must be applied in a coherent and integrated way in order to produce a consistent IoT gadget. One of the several challenges that a designer must face in every new IoT prototype is which architectural alternative is going to apply depending on several determinant factors.

Smart connected objects are here to provide a service, which partially resides on the Internet, while the object itself is bounded to real world constrains such as battery, physical form or interaction mechanisms. While the Web browser provides more and more sophisticated and flexible ways of using Internet services by people, the above mentioned limitations compel to carefully select the best architectural solution for a concrete IoT service.

In this paper, we describe our experience during the last years designing Internet-connected objects based on different architectures depending on the service to provide and existing contextual constraints. The structure of the paper is the following: in section II we introduce the determinant factors that influence which architectural solution to adopt; in sections III to V, we describe the three main possible architectures for IoT services, along with a brief depiction of how we implemented them in some prototypes and the results; section VI presents a comparative analysis of the architectures based on the above factors for easy reference, and finally section VII provides a summary, conclusions and future research.

II. FACTORS INFLUENCING THE COMMUNICATION OF PHYSICAL OBJECTS WITH WEB SERVICES

Every time a designer tackles the challenge of creating a new concept of device or smart object connecting to the Internet, there are several factors that must be taken into consideration since they definitely influence the final solution. Some authors have proposed OSGI-based [3] or autonomic-computing [4] approaches for IoT architectures, while others pay special attention to the application of the successful Web mash-up model [5].

In our experience, for creating Internet of Things solutions, we must consider the following aspects and answer the associated questions:

- *Computing requirements*: How much "intelligence" needs to be embedded in the device? How much can be commissioned to the Internet, the "cloud" [6]? Which is the proper balance between local and remote processing?
- *Networking requirements*: Which bearer topology is appropriate for this solution? Is the energy-inefficient Wi-Fi alternative suitable for an electronic pet? So that the pet can access the Internet, roaming through different available networks. Is the non-IP native IEEE 802.15.4/ZigBee more appropriate? So that we save more energy for the actual user-oriented activities. Do we need to provide a privative RF solution for

978-1-4244-5328-3/10/$26.00 ©2010 IEEE

communicating our objects due to budget constraints? May 6LoWPAN be the solution?

- *Energy*: Is our device battery-powered most of the time or plugged to a power socket? Mobility of the object during its operation demands battery, which may affect decisions about computing and networking requirements (delegating processing to the Internet and energy-efficient bearers), while an unmoving object (such as a digital photoframe or an electronic plant pot) may be continuously plugged, unaware of energy constraints.
- *Interaction during operation*: Physical objects do not have keyboard, mouse or screen, but they do browse (part of) the Internet. Which are the right mappings between user (inter)actions and object behavior? How do we physically represent changes in the Internet data through built-in actuators (LEDs, buzzers, vibrators, mechanical parts)?
- *Configuration*: Sometimes configuring the Wi-Fi network in a computer is tricky. How can we do it in an object which, again, does not have keyboard, mouse or screen? Obviously we can connect the object to a computer through USB and perform the configuration process, but we must conceptualize pure Internet-connected objects that do not require computers to become fully operational.
- *Infrastructure requirements*: Does our object connect to the Internet more or less naturally without special environment preparation? Or, do we have to deploy an infrastructure to enable local coverage? Wireless sensor networks (WSN) are being widely used to report real-time information to the Internet [7] in popular websites such as Pachube [8] or Microsoft SensorMap [9], but a careful deployment of a fully functional WSN infrastructure with nodes acting as routers at the proper places is required.
- *Business models*: Considering all the previous requirements may lead to the design of an appropriate Internet of Things solution from a technical perspective, but lacking the features to become a successful commercial product. Sometimes, right technical solutions increase the cost to a degree that is not acceptable by customers. In other cases, the selected architecture may not be flexible enough to be easy adaptable/updatable by the service provider, thus resulting in an obsolete user experience. We must strongly remark that real objects are not as easy to upgrade as web services, especially from a user interface perspective since they are constrained by their physical form.

From our experience, all the above factors are of paramount importance for creating experiences connecting the real and the digital world. While designing several prototypes, we have applied different architectural styles that implement some of the previous features in diverse flavors. The next three sections describe these architectures, the prototypes and their implications.

III. Direct Communication Architecture

This architectural design represents the pure essence of physical things accessing and consuming services in the Internet without any intermediate entity. As depicted in Figure 1, the smart object is powered with all the required functionalities to fully operate and communicate with Internet services without any external aid.

The functionalities required are generally an embedded HTTP client (along with the TCP/IP stack), XML processing facilities (or any other form of processing structured information, including RDF/OWL if the services provide semantically annotated data), the processing logic, which represents the behavior of the object depending on the data and other local stimuli, and finally a user interface for configuration.

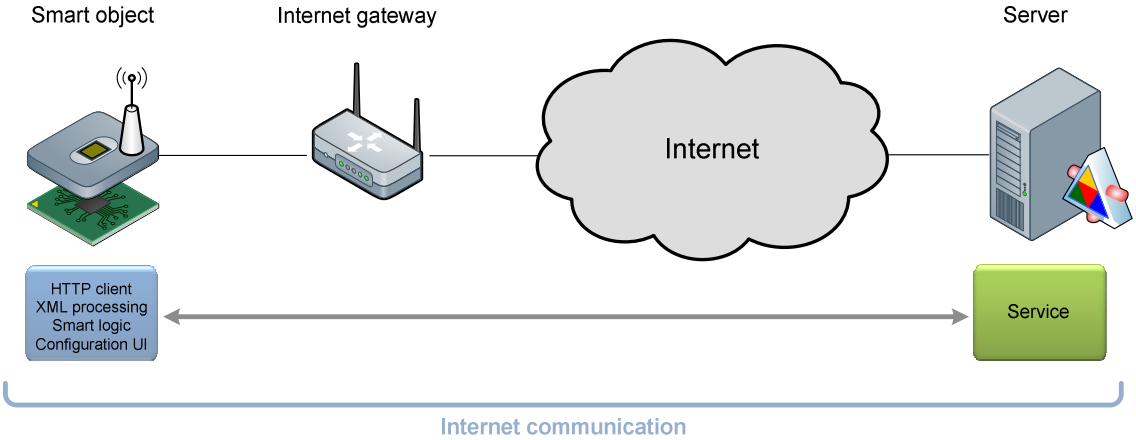

Figure 1. Direct communication between the smart object and the service in the Internet.

This approach is the most demanding in terms of physical object capabilities, especially computing power to process the data from the service, both at the data structure parsing phase and at the behavior implementation phase (smart logic). The most obvious implication of this is the energy demand by the device in order to operate, which may affect the lifetime of batteries in unplugged mode.

In terms of interaction, the user has no means of configuring the object other than using the limited physical interface provided. For simple configuration operations, such as on/off or update-now, a couple of buttons could be enough, but as more operations are added in a device without display, interaction becomes difficult and confuse (touching, shaking). The obvious alternative is embedding a lightweight HTTP server in the object with configuration pages. While this option allows greater configuration flexibility, it also demands additional support mechanisms that must be designed (e.g. discovering the IP address of the object).

A. *Example: Aware-Umbrella*

The Aware-Umbrella is an example of prototype that implements the direct communication architecture. The umbrella obtains context information from the Internet through a "virtual software sensor", a small piece of code that connects to the Internet to get weather information about the location (provided by weather.com) and semantizes these data using a weather ontology we designed. The umbrella finally checks the state of a nearby door (provided by other objects in the local network) in case the user is leaving when raining in order to decide whether to issue a voice alert to the user ("Please, take me with you. It is going to rain.").

The computing platform used in this prototype was a Gumstix connex 400xm with a Wi-Fi interface, a semantic middleware layer, and the Festival Lite TTS engine for text-to-speech. The state of the door was provided by a proximity sensor (based on a magnetometer). The Gumstix connex 400xm, an Intel XScale 400 MHz Linux-based platform, provides enough computing power as to embed all the smart logic in the umbrella itself, which may roam freely though open Wi-Fi networks. The text-to-speech engine provided the required interface for communicating the weather forecast to the user, although an array of LEDs could provide similar functionality in a simpler way.

Obviously, the main problem with the real use of this object is energy consumption. Unplugged operation is a must for an umbrella, and the processing and communication load during normal operation results in two AA batteries lasting for about 2 hours.

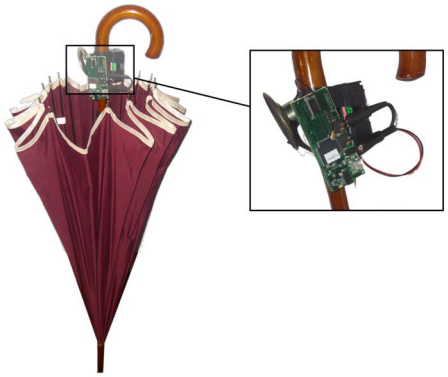

Figure 2. The Aware-Umbrella.

IV. Computer-Mediated Communication

This architectural style is implemented by several current commercial or pre-commercial products, such as TouchATag [10] or i-Buddy. The key factor here is creating simple objects with very limited capabilities, and delegating all the intelligent stuff to a locally connected computer (via USB, Bluetooth, Wi-Fi, Ethernet, or similar mechanisms).

The main advantage of this approach is the cost of the physical smart object, which can be very cheap since it basically acts as a peripheral. The direct connection to the local computer opens the possibilities for advanced user interfaces, *through the computer*, in order to operate or configure the object, which again, lowers its cost.

On the other hand, the most important drawback of this architectural model is the lack of autonomy of the smart object, which continually depends on the computer to do the activities. This approach is optimal for objects which are *always* in the range of a *known computer*.

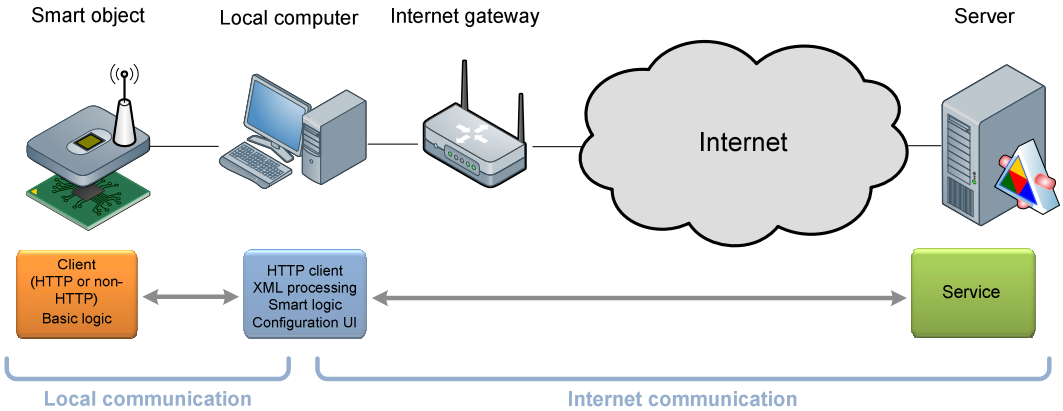

Figure 3. Computer-mediated communication between the smart object and the service in the Internet.

A. *Example: Real-Widget*

Desktop-based operating systems are increasingly using some mini-applications called *widgets* or *gadgets* in order to provide small services or show concrete up-to-date information to users. Yahoo! Widgets, Apple Mac OS X Dashboard, Google Gadgets and Microsoft Windows Gadgets are the most popular flavors of this form of interaction. Often, these widgets connect to online services in the Internet in order to provide weather or traffic information, most-popular YouTube videos, latest news, and so forth.

With RealWidgets we wanted to embody the functional capabilities of these digital entities into real world tiny wireless displays, in order to have small "windows" deployed everywhere opened to information from the Internet.

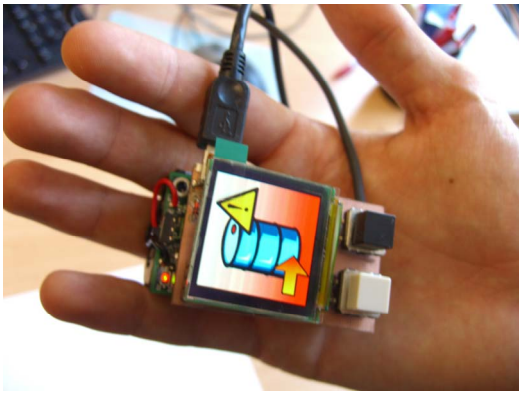

Figure 4. The Real-Widget.

A RealWidget is formed by an OLED display, with high resolution and contrast while small energy consumption, integrated with a Crossbow Mote2 wireless sensor network node. A local computer acted as a proxy between the Internet and the wireless sensor network, running a RealWidget Management Application that connected to the required sites on the Web, downloaded the information, analyzed it and finally sent the content to the appropriate widget as configured by the user.

Figure 4 depicts a RealWidget showing information about the liquid level in a remote chemical container that was monitored by a wireless sensor network over the Internet. Two buttons are provided for interacting with the RealWidget, basically for managing the energy by postponing the information for a later time, or immediately discarding it.

One of the most popular widgets in computer desktops are those related to weather information, so one of the earliest examples of the RealWidget prototype involved obtaining the data from a weather information site on the Internet. We chose weather.com due to the availability of an open API that provided the ability to download an XML document with the weather forecast for any location in the world.

The RealWidget Management Application could host different adapters that acted as processing gateways between the Internet service and the physical RealWidgets. In the previous example, we created an adapter that was able to connect to the weather.com website, retrieve the XML document with the weather forecast of the configured location, and transformed it into a notification message that was sent to the RealWidget. If any change in the weather information forecast occurred, subsequent notification messages would be sent to the RealWidget. The Crossbow Mica2 wireless node on the RealWidget received the message, processed it, and displayed an icon on the OLED display representing the weather conditions (sunny, cloudy, raining, and so forth).

The RealWidget embodied the advantages of the computer-mediated communication model. Since all the processing is carried out at the computer, the RealWidget performs very simple processing, saving energy for working during long periods.

V. SERVER-SUPPORTED COMMUNICATION

This approach provides a balance between autonomy and intelligence. The functional architecture looks similar to the computer-mediated alternative, with the smart object delegating all the processing to an external entity, thus saving energy. But the main difference is that now the physical smart object is independent from a local connection, enabling roaming and mobility.

Configuration and personalization can be performed through a web page on the support server, therefore providing advanced user interface mechanisms. The device can download the new configuration either triggered by a user action (e.g., pressing an "update" button) or through a periodic update process.

Moreover, since all the communication between the smart object and the service in the Internet goes through a single control point (the support server), it is easy to add new features, restrict access, and implement billing or other subscription mechanisms, depending on the desired business model. In fact, this approach provides the greatest flexibility to have an adaptable smart object as long as the support server evolves with additional functionalities. However, in this case the problem rests on how to design physical devices that are ready to integrate and embrace new remote capabilities that were not foreseen at the moment of the object conceptualization. The only solution that we can suggest is making the object so simple, logic-less, as to act as a "peripheral" of the remote service, although this approach may result in losing the "smart experience" of using the object.

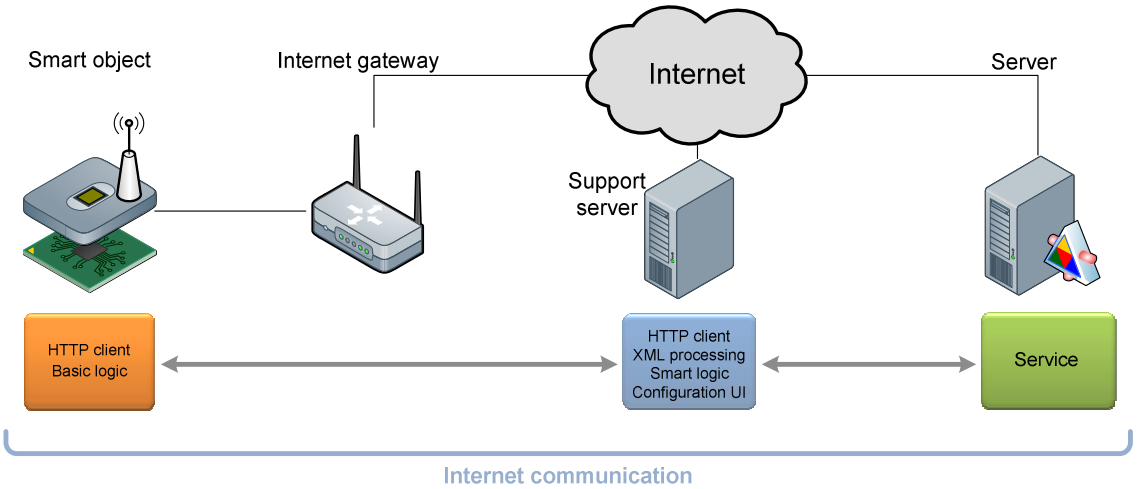

Figure 5. Server-supported communication between the smart object and the service in the Internet.

A. *Example: iCompass*

The iCompass is a device that acts like a compass, but instead of pointing North, it points to a zone of a background dial card that represents some concrete Internet information. For instance, inserting a Digg dial card with a gradient indicating popularity, triggers the iCompass to connect to the support-server, which searches for the popularity of certain preconfigured term in today's diggs (the term can be configured through the support-server webpage). The result is sent back to the iCompass, which moves the pointer to the appropriate part of the dial.

Figure 6. The iCompass and example weather dial cards.

Dials can represent aspects such as "connected friends in my social network", "weather forecast for my town" or "current ranking of my favorite football team", among others. The entire configuration is carried out through a web-based user interface at the support-server, while the only interaction that the user performs with the device is inserting or removing the dial card. The design of the iCompass is based on the integration of a Propeller microcontroller, a Wi-Fi module and an embedded RFID reader for identifying the dial cards.

The iCompass and similar gadgets are good representatives of the server-supported architecture, which is the model we are currently implementing in ongoing designs.

VI. Comparative Analysis

Based on the factors introduced in section II, we have carried out an analysis of the above architectures in order to help in the decision process about which one best fits the requirements of a particular scenario. The results are shown in Table I.

First, we have eliminated from the analysis the aspects related to "energy" and "interaction for operation", because, although they are important for any IoT solution, their values do not influence the selection of the architecture. In the case of "energy", the decision whether the object is battery-operated or needs to be continuously plugged depends on its intended use, not on the communications architecture. Similarly, the physical interaction methods for using the object do not affect the selection of the architecture.

On the other hand, computing and networking requirements may dictate how to distribute the logic of the experience among the object and the Internet service. Requirements related to the configuration flexibility may also determine if the configuration process can be embedded in the object, or provided by a more complex software system running on a computer or a server. For instance, configuring the Aware-Umbrella with the appropriate location may require displaying a dynamic interactive map to facilitate the process, which may not be possible with the limited computational and storage capabilities of the embedded web server.

Finally, an interesting aspect is related to the adaptability of business models for web-connected objects. We are used to "beta version" web services that evolve over time from "free" to "freemium" business models based on findings about how to monetize the services provided. LinkedIn is a nice example of this approach, providing free professional networking services as well as more advanced paid services for premium users. While this approach works well in the traditional flexible software-based Internet, when it comes to real things is not so easy: the objects may not be designed for consuming the new services unless a firmware upgrade and/or a physical design upgrade are carried out.

TABLE I. COMPARATIVE ANALYSIS OF THE COMMUNICATION ARCHITECTURES

Factors (to be weighted)	**Architecture**		
	Direct communication	***Computer-mediated***	***Server-mediated***
Computing requirements	*High (3)* Smart logic at the object-side	*Low (1)* Distributed logic between the object and the computer	*Low (1)* Distributed logic between the object and the support-server
Networking requirements	*High (3)* Direct Internet connection required, possible problems with proxies and validations	*Low (1)* Indirect Internet connection enables several possibilities	*High (3)* Direct Internet connection required, possible problems with proxies and validations
Infrastructure requirements	*Low (1)* Only access point or similar required	*High (3)* Computer continuously required	*Low (1)* Only access point or similar required
Configuration	*Strict (3)* Embedded server with limited configuration options	*Very flexible (1)* Powerful configuration options through computer-based software, even firmware upgrades, diagnostics, …	*Flexible (2)* Web-based configuration
Adaptable business models	*Strict (3)* Software in the object cannot be easily and safely upgraded without a computer	*Flexible (1)* Computer software can be easily upgraded to adopt new models	*Flexible (1)* Server-side software can be easily upgraded to adopt new models

The above indications are only intended to be considered as a guide to help the designer in the process of selecting the most suitable communication architecture. Depending on the relative importance of the factors for a particular scenario different weights must be be assigned to obtain a final reference value, the lower the better.

VII. CONCLUSIONS

In this paper we have described the main factors that may influence the design of communication architectures for real objects accessing and consuming services from the Internet. We have introduced three architectures, characterizing them with their advantages and drawbacks, so that smart objects designers may consider which approach best suits a concrete Internet of Things concept. Along with the descriptions, we have provided examples of systems that use these architectures. Finally, a comparative analysis has been provided for easy reference when deciding which of the architectures to adopt depending of the relative importance of the factors for a concrete scenario.

We still think that some more work, both theoretical and experimental, has to be devoted to the study of all the implications that the different factors may have when designing user experiences based on the connection of the real and digital world through the Internet. Especially important for future work are factors such as energy and power harvesting, and the business models that these architectures may enable.

ACKNOWLEDGMENT

The authors would like to thank all the research staff of DeustoTech for their support during the development of these projects, and especially Iñigo Sedano for the embedded platform of the Aware-Umbrella.

REFERENCES

[1] N. Gershenfeld, R. Krikorian, and D. Cohen. The Internet of Things. Scientific American, 291(4):76--81, 2004.

[2] The Internet of Things. ITU Internet Reports, 2005.

[3] J. Rellermeyer, M. Duller, K. Gilmer, D. Maragkos, D. Papageorgiou, and G. Alonso. The Software Fabric for the Internet of Things. In Proceedings of the First International Conference on the Internet of Things, Zurich, Switzerland, March 2008.

[4] Guy Pujolle: An Autonomic-oriented Architecture for the Internet of Things. John Vincent Atanasoff Symposium 2006: 163-168.

[5] D. Guinard, V. Trifa, T. Pham, O. Liechti. Towards Physical Mashups in the Web of Things. Proceedings of INSS 2009 (IEEE Sixth International Conference on Networked Sensing Systems). Pittsburgh, USA, June 2009.

[6] B. Hayes. Cloud computing. Commun. ACM 51, 7 (Jul. 2008), 9-11, 2008.

[7] J. I. Vazquez, A. Almeida, I. Doamo, X. Laiseca and P. Orduña. Flexeo: an architecture for integrating Wireless Sensor Networks into the Internet of Things. Proceedings of UCAmI 2008 - 3rd Symposium of Ubiquitous Computing and Ambient Intelligence, Salamanca, Spain, 2008.

[8] http://www.pachube.com/

[9] http://atom.research.microsoft.com/sensewebv3/sensormap/

[10] http://www.touchatag.com/