Interoperability layers for avatar-based WoT platforms

Deliverable 6 (version 1)

Lionel MÉDINI

31 December 2017

Project ASAWoO

Adaptive Supervision of Avatar / Object Links for the Web of Objects

Grant Agreement: ANR-13-INFR-0012-04

Abstract

This document merges deliverables initally intended to be 6.1 “specification of generic algorithms that allow an avatar to establish and modify a communication scheme with the object”, 6.2 “specification of the algorithm that perform protocol deduction from a functionality description” and 6.3 “implementation of these algorithms in the avatar”. It describes the work related to the interoperability layer on which the ASAWoO platform is to be built.

Contents

1 Introduction

In the IoT field, numerous platforms have been implemented. They differ by the types of connection and communication protocols they take into account, as well as by how they expose objects. They take into account concerns such as discovery and communications with objects, and some of them rely on existing standards such as exposing objects as RESTful resources. However, there is currently no standard solution to address concerns at the level of WoT software platforms. Hence, existing IoT solutions can be used as a lower layer to connect things, and WoT platforms implemented as "WoT application servers" on top of these solutions.

We studied this in a side-project of the ASAWoO project, called "Couche d'Interopérabilité Matériels-Applications" (CIMA), which started in fall 2013 and ended in 2016. The initial objective of CIMA was to gather several needs of the LIRIS laboratory regarding connecting objects to existing platforms, and to get funding to develop and maintain an IoT platform matching these needs. Unfortunately, no resource were available at that time and it was not funded. Hence, we restricted this project to the needs of the ASAWoO project and this side-project was led by Lionel Médini (LIRIS), as member of ASAWoO. The contents presented herein are the result of student works in first and second years master degree, as well as in institute of technology, as part of their mandatory projects.

In this deliverable, we address the following 3 questions :
1) How can an avatar-based WoT platform communicate with an object through an IoT platform?
We first present the requirements on which was based the CIMA project.
2) Which IoT platform to select as a base layer for our WoT platform?
We then present a comparison of IoT platforms made by first-year master degree students in early 2014.
3) How can an avatar inside a WoT platform acquire fine-grained control over the communications with its object?
We then go deeper into the API of the IoT platform and present the work done to make it suitable as an interoperability layer for the ASAWoO project.

2 Initial requirements

The initial objective was to provide LIRIS researchers with a common access point facilitating the connection and use of simple devices (sensors, microcontrollers) or complex devices (interface peripherals, robots) through different communication interfaces. (wired or wireless). The difficulty was to take into account the different needs related to the various types of research work (types of information reported, performance, communication interfaces ...). Figure 1 depicts a generic architecture that was issued after the needs gathering step.

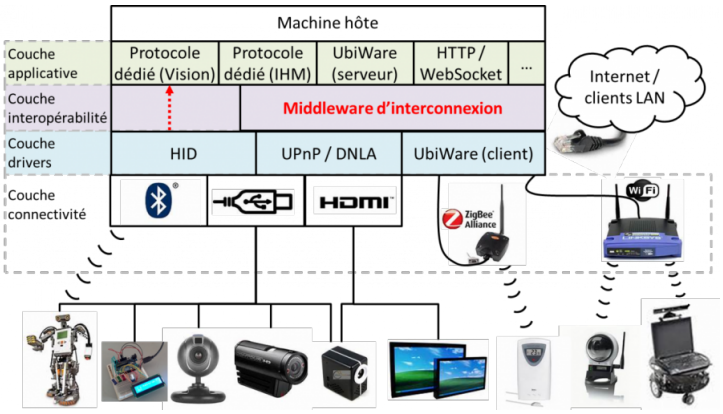

Figure 1: Preliminary architecture for the CIMA platform.

Table 1: Comparison of four main IoT platforms

Platform	Community size	Patners	Documentation	Native protocol failover	Client-side cross platform	Server-side cross platform	Open sourced
AllJoyn	++	+++	+++	Yes	+ (6 languages)	++ (6 languages)	Yes
DeviceHive	+	-	++	No	++ (9 languages)	+ (4 languages)	Yes
Open Remote	-	-	+	No	+++ (∼30 languages)	N/A	Partial
OM2M	-	-	+++	Limited	+++ (Standard ETSI protocol)	+++ (Web standards)	Yes

3 Base platform choice

3.1 First version

A first round of state of the art took place in early 2014, comparing several available IoT platforms. Among those, we pre-selected four of them that were suitable for our needs (AllJoyn, DeviceHive, OpenRemote and OM2M), and discarded others (Kalimucho, Koneki, Mihini, Ponte, RealTimeLogic).

- **AllJoyn** is an open source framework developed by Qualcomm Innovation Center to simplify the development of object-oriented web applications. The Framework has many services such as device discovery, peer-to-peer Peer-to-Peer service coupling, and message transport with or without reliability.
- **DeviceHive** is an open source framework developed by DataArt, it transforms all connected devices into a WoT object. DeviceHive provides various services such as the communication layer between connected objects, the object control layer, and multiplatform libraries to more easily integrate DeviceHive into an environment.
- **OpenRemote** is a middleware (software used as intermediary of communication between several applications) oriented towards the WoT. It allows to integrate almost any object to a network, and to create specialized interfaces for the control of these objects.
- **OM2M** is an IoT platform developed at LAAS laboratory in Toulouse. It is research-oriented and aims at connected different types of objects, and gives access to them to client algorithms. It followed the deprecated Eclipse M2M specification and later became compliant with ETSI standards. However, at the time of our first study, the OM2M project had just started: it was limited to objects connected to an IP gateway, and was not followed by a large community.

Table 1 summarizes the results of the study of characteristics of these three platforms. This study led us to choose, at spring 2014, the AllJoyn platform as best candidate for the ASAWoO interoperability layer. We implemented an interoperability layer based on AllJoyn, and demonstrated it at the InnoRobo exposition in Lyon. This demonstration is available on the ASAWoO Youtube channel.

3.2 Current version

In fall 2014, some of the students who had worked on CIMA during their first year master degree continued their work on their second year. At that time, we had activated our contacts with LAAS and the OM2M project had grown in size. Moreover, underlying protocol of AllJoyn was proprietary and did not cope with the DTN tasks of the ASAWoO project. We then chose to re-implement CIMA using the OM2M platform. OM2M is based on the ETSI M2M communications Functional architecture, depicted in figure 2.

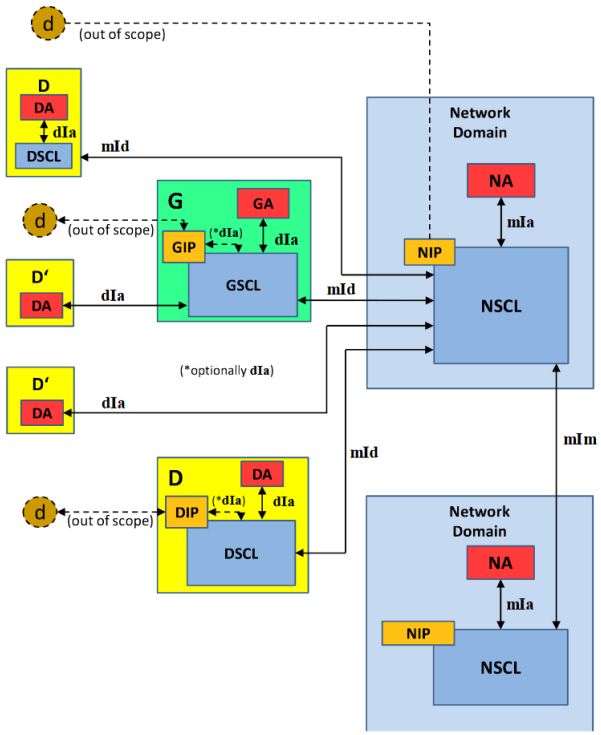

Figure 2: ETSI M2M reference Functional architecture (source: ETSI).

This architecture is based on three Service Capability Layers (xSCL), according to the following terminology:

Network (N): represents the "cloud" layer (in the ASAWoO terminology), hosting the reference services clients interact with.

Gateway (G): provides M2M Service Capabilities (GSCL) that communicates to the NSCL using the mId reference point and to DA or GA using the dIa reference point.

Device (D): on this figure:

- device D provides M2M Service Capability (DSCL) that communicates directly to an NSCL (this will not be the case for ASAWoO devices)
- device D' communicates to a GSCL (in ASAWoO) or to NSCL using embedded code
- additionally a non-ETSI M2M compliant device ('d') can connect to an SCL; in ASAWoO, it is then to the gateway to provide the Network layer with its capabilities

More details on this architecture are available in the complete specification.

4 Architecture of the CIMA platform

CIMA can be seen as an interoperability layer, as all physical devices exposed by CIMA are described using semantically annotated capabilities (in the ASAWoO sense). CIMA not only aims at supporting the ASAWoO platform, but in general, research algorithms that require interacting with connected objects. Among those, it targets several works in the fields of video stream processing and human-machine interaction.

As said above, we chose to implement CIMA on top of the OM2M framework. We added to this framework the following features, depicted in 3:

- On this figure, "Device" represent the objects connected under various protocols (IP, HTTP, USB...) allowed by the OM2M platform. In order to perform discovery and update the list of devices currently connected to the GSCL, **CIMA permanently scans the list of available devices**, and stores the updates in the database. It also tries to send a standard request to each newly discovered object (see below).
- So that capability descriptions emerge from the layer responsible of connecting and managing things, **CIMA implements the notion of capability**. Physical devices that are capable of performing introspection embed a *capability description service* to communicate their capabilities to the upper layers. Capability semantic descriptions are sent using the Hydra and ASAWoO ontologies, and can be transferred to CIMA clients, such as the ASAWoO platform.
- As all objects cannot perform introspection, **CIMA provides a configuration interface**, in which users can easily configure their devices, save and restore device profiles.
- As OM2M internal object-oriented database was about to be deprecated, **CIMA relies on a separate persistence solution**, instead of relying on the existing one. This solution is based on the MongoDB database and RestHeart interface that provides a REST API used to persist connected devices and their capabilities.
- Once discovered and described, clients no longer need to go through the complex OM2M stack to access the devices capabilities. **CIMA provides its client algorithms with a so-called "port-forwarding" mechanism** that allows bypassing the complex platform infrastructure and directly (although securely) access objects through dedicated ports of the gateway. Hence, the platform can be used for discovering and accessing objects, and performance-intensive clients (such as image-processing algorithms) can directly interact with these objects.

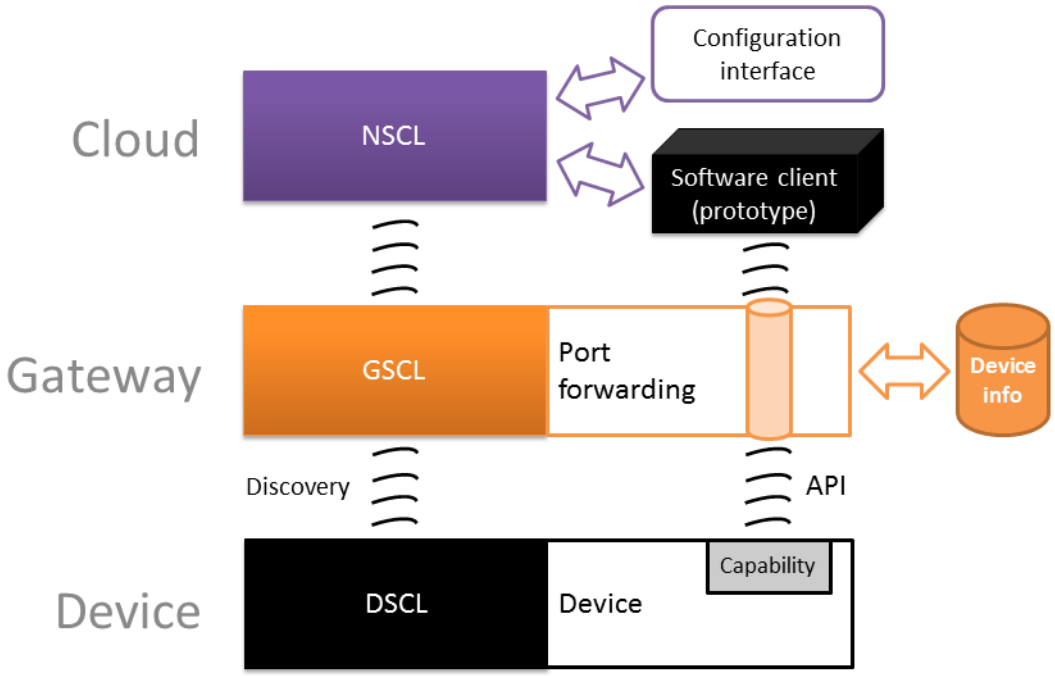

Figure 3: A schematic overview of the CIMA platform elements. (D|G|N)SCL are the different layers of the the OM2M framework.

5 Documentation

More details are available in the following documents, concerning:

- Technical and architectural platform description
- User documentation
- Port forwarding documentation
- Introspection mechanisms

6 Produced software

The code of the last version of the CIMA platform is available at its GitHub repository.

Software modules have also been issued for:

- the first version of the platform (not publicly available to avoid confusion)
- several versions of the code to be deployed on objects (performing introspection); the code developed for Lego EV3 under LeJOS (Java) is available at this GitHub repository

7 License

CIMA follows the conditions of OM2M's Eclipse Public License (EPL).

8 Conclusion

The CIMA platform has been developed in two years, according to the original specifications of the ASAWoO project. It was also designed to match to the requirements of other research prototypes that include connected objects. Even though, for compatibility reasons with the DTN work, it has not been linked the final ASAWoO platform, it has been developed as a side-project of the ASAWoO project and would permit to connect object with various network interfaces. However, its API has been reused in the ASAWoO JavaScript prototype, which comprises an interoperability layer.

CIMA has completely been developed by students using resources from the Lyon 1 University, computer science department, at no cost for the ANR and is ready to be used for further projects. It conducted us to strengthen our collaborations with the research community in the Toulouse area. Moreover, it participated in the dissemination of the ASAWoO project, as it was presented at two InnoRobo exhibitions, in 2014 and 2015. Videos presenting and demonstrating this work are publicly available on the ASAWoO Youtube channel.

9 Contributors

CIMA is a platform project developed at LIRIS lab in 2014 and 2015 and led by Lionel Médini.
It has been developed (and documented) by numerous master degree students of the Lyon 1 university:

- Alexandre Boukhlif
- Hady Mamadou Diallo
- Rémy Desmargez
- Maxime Baudin
- Farah Chetouane

- Harold Ngaleu Toumeni
- Maxime Guyaux
- Hicham Bouchikhi
- Malik Abdelkader
- Quan-Khanh Lu
- Ricardo Yamamotu
- Frédéric Da Silva
- Jordan Martin
- Christophe Pinelli

Thanks to them all!

Special thanks as well to the members of the LIRIS lab who also contributed:

- Hervé Saladin, from the LIRIS platform team
- Christian Wolf, head of the LIRIS-VISION platform
- Eric Lombardi, Olivier Georgeon, Erwan Guillou, Aurélien Tabard (and others) for their time during the use case study
- Mehdi Terdjimi, for his help during design and development stages