Journal of Network and Computer Applications ∎ (∎∎∎∎) ∎∎∎–∎∎∎

Contents lists available at ScienceDirect

Journal of Network and Computer Applications

journal homepage: www.elsevier.com/locate/jnca

Enabling end-to-end CoAP-based communications for the Web of Things

Miguel Castro[*], Antonio J. Jara, Antonio F. Skarmeta

Research Institute for Oriented Information and Communications Technologies (INTICO), Computer Sciences Faculty, University of Murcia Regional Campus of International Excellence "Campus Mare Nostrum", Murcia, Spain

A R T I C L E I N F O

Article history:
Received 5 December 2013
Received in revised form
27 May 2014
Accepted 23 September 2014

Keywords:
Internet of Things
Web of Things
CoAP
End-to-end
IPv6
WebServices

A B S T R A C T

Web of Things (WoT) plays a crucial role for the convergence of the Internet of Things (IoT). Web technology offers efficient support for global communications and wide access to services and information. WebServices-enabled devices allow the usage of different existing Web technologies for the development of applications in a more scalable and flexible way. For this purpose, homogenous and seamless end-to-end communication needs to be enabled independently of concrete solutions or specific Web-clients. WoT offers CoAP as a lightweight WebServices-based protocol for IoT devices. The main challenge is that CoAP has not been completely integrated into the existing Web clients without the need of intermediate application servers, proxies or gateways for the deployment of a fully end-to-end paradigm between end-clients (e.g., Web-browsers) and end-nodes (e.g., Sensors/Actuators). This paper proposes a flexible and scalable embedded CoAP solution for Web applications that offers full integration with browsers and Web-clients, without the need of intermediate application servers/ gateways, or proxies.

1. Introduction

The rapid evolution of communication technologies, in conjunction with the development of new devices such as wireless personal devices, embedded systems, smart objects and remote sensors have imposed an extension of the current Internet capabilities towards the Internet of all the objects and devices located around us, defining the so-called Internet of Things (IoT) (Atzori et al., 2010). These new capabilities are enabling a completely new and never imagined generation of devices, services and applications where IoT resources can be conceived as service end-points, i.e. Things or Resources as a Service (TaaS or RaaS). Moreover, technologies involved provide to devices the capabilities to expose their resources to the rest of the Internet services and applications.

On this regard, IPv6 over Low power Wireless Personal Area Networks (6LoWPAN), in conjunction with the integration of Wireless Sensor Networks (WSNs) and smart objects within the Internet, have enabled the utilization of IPv6 over the IEEE 802.15.4 wireless communication standard. At the same time, Web services based on Representational State Transfer (RESTful) have been enabled for IoT devices through the Constrained Application Protocol (CoAP), providing the required capacities and functions to integrate smart objects with the Web.

The use of Web services on top of IPv6-based smart objects reduces the complexity of application development and deployment by using *Things* as Web resources and enabling the integration of data through Web mashups.

However, a last mile still exists to be covered in order to enable real end-to-end connectivity between Web-services clients (Shelby 2010) and end-nodes without the need of additional proxies, gateways or servers (Kovatsch et al., 2012).

Even when CoAP follows up the REST style, and it could be seen as a lightweight HTTP protocol, this is not directly interoperable with the current Web Services-enabled clients such as browsers. For that reason, this work pursuits the integration of CoAP-support over end-clients in a transparent and scalable way, taking into account the end-to-end foundation of the Internet and the integration with the existing architectures and applications.

Our main contribution in this topic relies on the fully end-to-end connectivity achieved between clients (e.g., Web browsers) and the connected entities instead of having to trespass intermediate servers or application gateways as proposed in Colitti et al. (2011). Our approach is focused in having all the computation and network requests together running in a sandbox in the client side. Thereby, there is no need for extra processing infrastructure between clients and end-nodes.

The rest of the paper is organized as follows. In Section 2 is presented an analysis of different WoT integration approaches based

[*] Corresponding author.
E-mail addresses: miguel.castro@um.es (M. Castro), jara@um.es (A.J. Jara), skarmeta@um.es (A.F. Skarmeta).

http://dx.doi.org/10.1016/j.jnca.2014.09.019
1084-8045/© 2014 Elsevier Ltd. All rights reserved.

on current platforms and Web technologies offering end-to-end access to resources following the premises of the Future Internet (IPv6) and IoT integration such as defined in Jara and Skarmeta (2012). Section 3 explains the proposed architecture for designing CoAP based Wireless Sensor Networks, presenting an embedded CoAP library that enables scalable and flexible deployments of end-to-end CoAP and IPv6-ready Web applications. Finally, Sections 4 and 5 discuss the advantages of the end-to-end integration of WoT services over a real deployment of a Smart Lighting system where streetlights, sensors and actuators are exposed as Embedded Web Services based on Java and accessible through JavaScript in order to offer the mechanisms to interact with the functionality offered by constrained devices and discover new resources in a straightforward manner (Castro et al., 2012) (Table 1).

2. Enabling end-to-end Web of Things

The integration of smart objects into the Web is becoming an important aspect to cover in order to make them fully accessible and interoperable. For example, mashup applications are being developed to integrate data and services from real devices in conjunction with other virtual Web content and Web resources deployed all around the Internet.

While research has been mainly focused in network architectural concerns and communication issues, there has not been much research about how to cover the last mile between Web Clients (e.g., Web Browsers) and end-nodes avoiding the need for application servers, application gateways or other intermediate actors within the path between clients and deployed smart objects over a network.

To this end, our proposal mainly consists of a CoAP Library written in Java, embedded as an object in an HTML5 site that could be easily accessible through JavaScript, one of the most common scripting languages used today in the Web and for extension, in WoT.

The system deployment keeps completely integrated in the client maintaining the path between the browser and the end-node completely clean, without needing any intermediate delegate, just the JVM (Java Virtual Machine) running the CoAP Library already present in the client in order to open and manage network connections.

Several approaches have been analyzed and taken into account during this research work about how to create an Universal Datagram Protocol (UDP) socket from the client-side application. Since, this is the main requirement to integrate CoAP within the client applications.

Specifically, Cooper (Kovatsch 2011), a browser plugin for Firefox/Chrome to support CoAP browsing, Actinium, a server-based RESTful runtime container for scriptable IoT applications (Kovatsch et al., 2012), and gateway/proxy based deployments such as described in Colitti et al. (2011) have been analyzed. In addition, Instant Web P2P Peer architecture has been reviewed (Instant Web P2P) while at the same time parallel implementations have been developed in Node.js (Cantelon et al., 2013) and Java Network Launching Protocol (JNLP) in order to compare the required infrastructure, operation ways, and the main characteristics from the different approaches.

In the following lines, the considered approaches are described and their main design considerations are presented.

2.1. *Cooper – a CoAP user agent for firefox*

Cooper is a browser plugin for IoT resources based on CoAP. It consists on a user-friendly management tool deployed as a Firefox

Table 1
Approaches comparison for enabling end-to-end connectivity on the Web of Things.

Approach	Flexibility	Web integration	End-to-end	Security	Scalability	Lightweight
Cooper Kovatsch (2011)	**Low** Just offers a user interface for end nodes through a Firefox Plugin	**Low** Firefox Browser required. Depends on Mozilla foundation libraries	**Yes** Connections are open directly from client to end-node	**Low** Security model not defined	**No** Designed just for testing purposes	**Yes**
Actinium Kovatsch et al. (2012)	**High** Exposes scripts through a RESTful programming interface using CoAP	**High** Applications are accessed through JavaScript enabled browsers	**No** Delegates networking issues to Application Server	**High** Based on isolated sandboxes, policies and monitoring	**Low** Need for an Application Server for script execution	**Yes**
Gateway based Colitti et al. (2011)	**High** But needs for an application gateway reduces architecture flexibility	**High** Applications are accessed through JavaScript enabled browsers	**No** Delegates networking issues to Application Gateway	**Medium** Based on isolated sandboxes	**Low** Need for an Application Gateway for managing connections between clients and end-nodes	**Yes**
Node.js Instant Web P2P	**High** But needs for server running Node.js reduces architecture flexibility	**High** Applications are accessed through JavaScript enabled browsers	**No** Delegates application execution and networking issues to Node.js Server	**Medium** Security issues related to Server Side JavaScript injection	**Low** Need for an Application Server running Node.js	**Yes**
iWebPP Cantelon et al. (2013)	**High** USP Web services in Peer/P2P style	**High** Applications are accessed through JavaScript enabled browsers	**Yes** But it has to run a Node.js server for arbitrating peers	**Low** Security issues providing backdoors	**High** Could expand client central style web services transparently	**Yes**
JNLP	**High** Applications can be easily downloaded and executed	**Low** Download required and execution is performed outside the Web browser	**Yes** Considering that now the user interacts with the JNLP application, not the Browser	**High** Based on configurable policies and Access Control Context	**High** Applications are distributable in a straightforward manner without need of intermediate servers	**No**
Proposal	**High** CoAP Embedded Java Library permits directly access from JavaScript	**High** Applications can be executed by major browsers with JVM and JavaScript support.	**Yes** Connections are open directly from Web client to end-node	**High** Based on configurable policies and Access Control Context	**High**⁎ Applications are executed directly in the client side. ⁎JVM is needed	**Yes**