# W3C Workshop on the Web of Things

## Enablers and services for an open Web of Devices

25–26 June 2014, Berlin, Germany

**Position Paper by Kheira Bekara, and Chakib Bekara - Centre de de Dveloppement des Technologies Avances (CDTA) Cite 20 aout 1956 Baba Hassen, Algiers, Algeria**

## Our vision of services for open Web devices

As smart-phones, our environment elements (called also Things, to point out the large diversity) become *smart*, able to calculation, execution of applications, data storing and access to the wireless network. Then, the possibility to use their capacities to discover innovative uses, is open. It is also possible to extend our control with new functionalities, and even more imagine the original interactions between objects.

Services of ”augmented environment” will be enabled, like a car that retrieve easily a free parking place, the complete accommodation management, context aware personalized emergency services in case of disaster, assistance of old people, ...etc. Omnipresent, interactive with user, or with its objects, the ”pervasive” approach put informatics to the outskirts of the user awareness. The user’s attention is less and less mobilized thank to an integration effortlessly, transparent of the computing dimension and its environment, this what Mark Weiser define by ”Calm Computing” [1].

Today, we are one step closer to this vision due to recent advances in identification technologies, wireless networks (constrained environments), and Web services. The primary focus of the Web of Things (WoT) is to bridge the gap between the constrained environments and digital worlds over a common and widely used platform, which is the Web. This should be done on two levels, the network level and the application level.

For the network integration, the IPv6 over low-power wireless area network (6LoWPAN) standards [2], [3] are playing a key role in driving the Web of Things. This new standard now allows the use of IPv6 in Low power and Lossy Networks (LLNs), such, the popular IEEE 802.15.4 wireless standard, industrial ,scientific, and medical (ISM) band telemetry radios, over low-rate power-line communications (PLC), using very simple embedded microcontrollers. While retaining the principle of addressing provided by the network protocol, it will maintain a table of correspondence between the address of each embedded device (very short) and a long IPv6 address for the outside. Also, the direct link from end to end between the embedded devices and other types of networks is established. Hence, once of the major benefits is to enable the use of standard web service architectures without using application gateways.

## Mains issues of embedded Web services

Web services have proven to be indispensable in creating interoperable communications between machines on today’s Internet, but at the same time the overhead and complexity of web service technology such as SOAP, XML, and HTTP are too high for use in the constrained environments often found in machine-to machine (M2M) applications.

In fact, the interaction model of M2M applications differs greatly from how web services are used today. Exchanges are short-lived, battery-powered devices mostly sleep, multicast is commonly required in automation, and flow control is needed across the entire M2M network. The basic nature of M2M applications is often asynchronous rather than synchronous.

The technologies deployed for web services today are usually not suitable for constrained networks and devices. Although RESTful web services are simple in concept, the protocols and payload formats used to realize them today were not designed with these types of applications, networks, or devices in mind. Problems with these protocols include:

- Overhead: The ideal UDP payload over 6LoWPAN and IEEE 802.15.4 is just 60-80 bytes for reasonable performance (avoiding fragmentation). HTTP headers alone are easily larger than this. The use of XML payloads adds further overhead as it represents content inefficiently. The embedded web will require both highly efficient application transfer protocols and payload formats to meet the expected overhead.
- TCP binding: Web services today depend on TCP, which has performance problems over lossy links, sensitivity to mobility, no multicast support and high overhead for short-lived transactions. Furthermore the congestion and flow control model does not match M2M interaction models well [4].
- Pull model: In sensor networks wireless nodes are typically sleeping over 90 percent of the time, making the HTTP request/response pull model inappropriate. Other interaction models are needed such as eventing and publish/subscribe.
- Complexity: Although HTTP may in theory seem simple, when used by modern HTTP servers, clients, and proxies it is not. HTTP has evolved into a highly complex protocol as used between modern servers and browsers. A large number of features and optional headers may be employed, increasing embedded device complexity. The use of XML as a payload adds further parsing complexity.In RPC web services this is further compounded by the use of SOAP.

In order to fully realize the Web of Things, the RESTful web service paradigm needs to be extended into the constrained domain. To do this we need a fresh approach to both the transfer protocol used to convey REST semantics, and the payload formats exchanged between applications.

# Embedded Web services solutions

## Palyload compression and encoding

As XML is not dense and difficult to parse, reducing the XML overhead is a key requirement. This is challenging, as the complexity of creating and parsing content must be minimized for all of devices at the same time.

Although compression algorithms like gzip are suited for HTTP clients and servers, thay are not appropriate for embedded web services because of the high complexity and low performances for small payloads. An alternative to the XML compression is to encode xoth an alternative binary representation.

Extensible XML Interchange (EXI) is a very compact representation of the XML Information Set that is intended to simultaneously optimize performance and the utilization of the computational resources. The EXI format uses a hybrid approach drawn from information and formal language theories for encoding XML information. Using a relatively simple algorithm, it reliably produces efficient encodings of XML event streams. The EXI specification allows for a simplified mode of operation called schema –informed mode. Here a schema is used to form a grammar and state-machine limited to that schema, enabling extremely efficient encoding. The result of the simple state machine is that even the most minimal embedded devices can work directly with the encoding without the need to work with a full XML parser.

Fast Infoset is an International Telecommunication Union — Telecommunication Standardization Sector (ITU-T) defined standard (X.891) that specifies a binary encoding for the XML Information Set based on ASN.1 encoding. Unlike XML compression schemes, the Fast Infoset standard has the dual benefits of both compression and performance, making it a good choice for moving large XML data between low bandwidth systems or for high performance systems such as servers utilizing web services.

Binary Extensible Markup Language (BXML) is a straightforward encoding for XML that is designed for efficient processing in general with an emphasis on dense numeric data, which is one of the weakest areas for textual XML. This encoding specification, designed by the Open Geospatial Consortium (OGC), is specified for GML compression.

XML is not the only format used for payload representation in RESTful web services. One popular alternative is the JavaScript Object Notation (JSON) [RFC4627].

## The Constrained Application Protocol (CoAP)

Since 2010, the CORE IETF group has defined an application transfer protocol that realizes a minimal subset of REST along with resource discovery, subscription/notification, and the use of appropriate security measures [5].

In order to realize the web service architecture while meeting the demanding requirements of M2M applications [6], CoAP has the following key features:

- Compact header: CoAP includes a compact binary header with extensible options. The protocol has a base header size of just 4 bytes, and a total header of 10-20 bytes for a typical request.
- Methods and URIs: In order for a client to access resources of a server, CoAP supports the request methods GET, PUT, POST, and DELETE. These methods are answered using a subset of HTTP compatible response codes. CoAP supports URIs, which is a key feature of the web architecture.
- Content types: As CoAP is a transport protocol for any number of different uses, it is possible to explicitly indicate the content type of the payload in the header. This is achieved by assigning codes to a subset of Internet media types.
- Simple caching: Caching for resource representations is supported to optimize for the performance of constrained environments.
- Transport binding: CoAP uses UDP for its transport, and provides a simple stop-and-wait reliability mechanism when needed by a request. The use of transport layer security is also supported through an optional binding to Datagram Transport Layer Security (DTLS).
- Resource discovery: In order to find and use CoAP resources autonomously, a built-in resource discovery format is supported (think of this as index.html for machines). This may be used to discover the list of resources offered by a device, or for a device to advertise or post its resources to a directory service. Resource discovery is discussed further.
- Subscription: One major shortcoming of HTTP is its reliance on a pull model. CoAP uses an asynchronous approach to support the push of information from servers to clients using subscriptions.

## Future challenges

The WoT vision which proposes to connect things within the Web, using well-defined RESTful interfaces based on the previous solutions is a significant step towards an affective embedded web services. Nevertheless, programming WoT applications remains difficult.

Our challenge is to design a system that encompasses constrained devices with a law foot print memory size running a Web server directly. Previous WoT solutions usually require either more powerful devices or the Web server being on a gateway.

Following we summarize how WoT applications can be developed:

- Virtual Machine (VM): the idea is to embed a VM in the device and deploy applications compiled as bytecode.
- Scripting languages.
- Macro-programming is another solution which aims for programming a network as a whole.

The first approach provides a high level of abstraction, a dynamic loading, software isolation, and minimizes the size of compiled applications. It has also the advantage to run the application logic on the device in a fully distributed manner.
The second approach provides also a high level of abstraction, providing the programmer with the ability to batch well defined basic device operations. The application logic is not fully distributed on devices, but increases productivity by making applications self-contained, focused on functionality.
The last approach can be combined with the first one providing in network processing between devices.

## References

[1] M. Weiser and J. Brown,The coming age of calm technology, beyond Calculation, Springer New York, 1997.

[2] Z. Shelby and C. Bormann,6LoWPAN: The Wireless Embedded Internet, Wiley, 2009.

[3] G. Montenegro et al,Transimission of IPv6 Packets over IEEE 802.15.4 Networks, IETF RFC 4944, 2007.

[4] L. Eggert, Congestion Control for the Constrained Application Protocol (CoAP), draft-eggert-core-congestion-control-00, 2010.

[5] Z. Shelby, B. Frank, and D. Sturek, Constrained Application Protocol (CoAP), draft-ietf-core-coap -02, 2010.

[6] Z. Shelby et al.,CoAP Requirements and Features, draftshelby-corecoap-req-00, , 2010.