Programming Device Ensembles in the Web of Things
A Position Paper for the
W3C Workshop on the Web of Things

Matias Cuenca, Marcelo Da Cruz, Ricardo Morin
Intel Services Division (ISD), Software and Services Group (SSG), Intel Corporation

1. Introduction

The Internet of Things (IoT) is undeniably picking up steam – we are already witnessing the incorporation of an enormous variety and huge quantity of devices interacting with each other and with services in the cloud, to solve an endless number of problems in many domains such as energy, transportation, health and industrial, to name a few. It is forecasted that 7 trillion wireless devices will be serving 7 billion people by 2020[1]. We share W3C's vision of the Web Of Things (WoT), that is, Web technologies will be at the front and center of the IoT megatrend moving forward, thanks to their universal adoption and enduring scalability. However, while a great deal of attention has been devoted to data access, collection and processing, the definition of a standard programming model for "things" has not been established yet. We believe that to support the WoT, Web technologies need to evolve from device-centric applications, to a programmer-friendly model that extends the concept of an application to seamlessly encompass multiple heterogeneous devices (i.e., device ensembles) collaborating and sharing resources and computational capabilities, both locally and across the cloud.

2. Programming Device Ensembles

The current model for creating IoT applications deployed to 'things' is mostly based on an evolution of traditional embedded computing, with the addition of tiny Web servers implementing standard Web protocols, plus the incorporation of new lightweight protocols (e.g., MQTT[2], CoAP[3]) for severely constrained devices. In this model, applications are:

- *Device-centric* – Code is deployed to individual devices, mostly pre-provisioned at manufacturing time.

- *Statically partitioned functionality* – Application code logic is apportioned across devices and the cloud a priori by design, and deployed to devices in a pre-arranged and explicit way.

- *Statically constrained by device capabilities* – Code only makes use of computational resources available on device and leveraging of remote resources requires explicit partitioning and code distribution.

- *Single purpose* – Deployed code meets a predetermined set of requirements and it is not designed to accommodate dynamic addition of "apps" and arbitrary devices.

- *Siloed security* – Due to the lack of standards, security and privacy is typically rigid, custom-designed and not architected to deal with multiparty situations, thus discouraging productive and safe sharing of data and capabilities – it is important to note that security and privacy concerns have often been cited as major risks in the IoT[4].

We believe that the programming model for the WoT needs to evolve towards device ensembles – dynamic arrangement of "things" that collaborate towards fulfilling the business goals of applications. Contrasting with the traditional embedded approach, device ensemble applications are:

- *Multi-device* – Target devices for applications are not defined a priori, but determined dynamically (i.e., zero configuration) depending on multiple factors including availability, ownership, proximity, and network accessibility.

- *Transparent partitioning* – Programming model is such that actual distribution of workloads across processing elements is accomplished at run time depending on device ensemble configuration. This is important for flexibility because it allows applications to be developed without prior knowledge of target configuration capabilities.

- *Elastic* – Because computational resources are allocated at run time and computational workloads can be offloaded to the cloud dynamically, the programming model is flexible and adaptive.

- *Multipurpose* – Devices incorporate application runtimes that allow for after market deployment of value added "apps" by third parties.

- *Designed with security and privacy* – There is a consistent security and privacy model with interoperable protocols to enable secure sharing of resources and data across devices, users and multiple parties.

3. Requirements and opportunities for standardization

To realize the device ensembles vision outlined above, there are a number requirements and opportunities for standardization as follows:

- *Application runtime for "things"* – Defining a common application execution engine standard, utilizing Web programming languages such as ECMAScript[5], and a set of standardized APIs. A good example of this kind

of runtime is the Node.js framework[6], which has already being proposed for supporting WoT applications[7] [8]. The work in this area may include defining standard profiles to cover APIs for various device categories (e.g., bare sensors, constrained microcontrollers, edge gateways, anchor devices). The application runtime will also need to define a standardized security model as well as user interaction approaches for constrained devices (e.g., control panels, dashboards). In this area, there seems to be a high synergy potential with the Web Applications (WebApps) Working Group[9] that should be pursued.

- *"Things" as resources* – All devices and their capabilities (e.g., sensors, actuators and computational resources) should be accessible via standard Universal Resource Identifiers (URIs) using the RESTful model[10]. Applicable standards in the areas of subscriptions, push notifications, and real time communications via Web Sockets[11] should be reviewed to identify improvement opportunities for the WoT.

- *Code offloading* – Standard declarative mechanisms for remote execution of workloads to local devices and to the cloud. This approach would enable the distribution of computation and access to remote resources from highly constrained devices within a unified programming model. We believe that extending the Web Worker[12] model to encompass remote execution is feasible because of their loosely coupled design.

- *Discovery* – Standard mechanism for discovering services in the local area and in the cloud and the dynamic establishment of device ensembles at runtime. We believe this entails the review and adaptation of the proposed Network Service Discovery[13] specification and the consideration of additional discovery and registration mechanisms to maintain dynamic relationships between multiple devices.

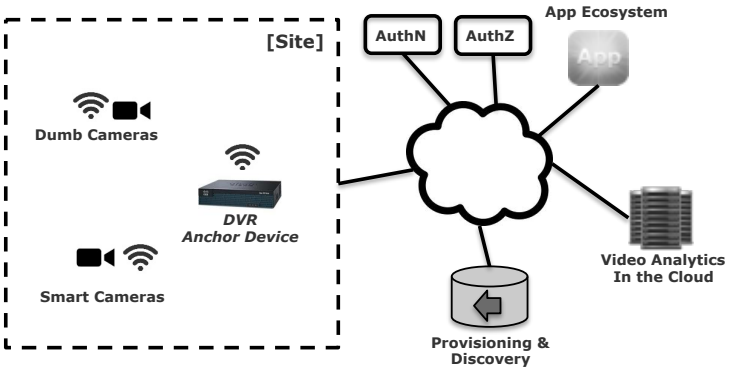
- *Application packaging and provisioning* – Definition of bundling and distribution of packaged applications to target devices and device ensembles. This area includes working on the applicable deployment models that need to be supported, as well as packaging standards and mechanisms for updating deployed applications. There is a good synergy here with the work of the Systems Applications Working Group[14] and the WebApps Working Group that should be leveraged for the WoT.

- *Access control and privacy* – A strong security and privacy model with standardized scalable protocols for sharing resources and data across device ensembles in complex multi-party situations. We believe that the proposed User Managed Access[15] (UMA) protocol provides a good starting point for this model because it is based on Web standards such as OAuth[16] and it is designed to deal with access control in multi-party situations and it encompasses policy-based implementations. We believe

mixed consumer, enterprise and Government actors engaged in WoT applications will be commonplace, which will require a robust privacy-by-design, policy-based approach such as UMA's.

4. Illustrative example

The diagram below depicts a video surveillance system, which consists of a mixture of smart and dumb cameras distributed on a site and a limited-processing DVR, which is an anchor device that acts as application "home" for the system. The anchor device and the smart cameras include an application runtime based on a framework such as Node.js, capable of hosting Javascript applications. Dumb cameras simply expose a RESTful interface for streaming images.



The initial provisioning process engages the UMA protocol to register ownership and establish access control. Applications are developed in Javascript as cohesive units (i.e., no explicit partitioning) without prior knowledge of the deployment configuration. CPU as well as data intensive algorithms are implemented using Web Workers. Upon UMA-authorized provisioning in the anchor device, the application discovers resources available and their capabilities (e.g., processing power, data locality). The runtime automatically dispatches remote Web Workers to authorized smart cameras throughout the site to perform motion detection and report only images of interest. While certain pattern recognition algorithms such as face detection can be performed in the anchor device using local Web Workers, since the DVR has limited CPU capacity, more complex video analytics algorithms (e.g., face recognition) are dispatched for remote execution in authorized cloud hosting services. Additional authorized 3rd party cameras, sensors, anchor devices and applications can be easily provisioned dynamically post-installation to enhance the capabilities of the system (e.g., fire detection, intrusion detection, object identification).

5. Conclusion

The WoT promises to be one of the most fascinating technology challenges of our time. The scale and overall implications of this nascent field are staggering. We believe that the standardization effort will be crucial to its success and the ability to scale it to its full potential. There are many areas for standardization in the WoT, and we are very eager to participate and contribute to this workshop. We believe that one of the critical domains to cover is the definition of a standard development, deployment and management model for WoT applications, including the formal definition of device ensembles and the associated privacy aware access control mechanisms.

References

[1] K. David, S. Dixit, N. Jefferies, "2020 Vision, The Wireless World Forum Looks to the Future," IEEE Vehicular Technology Magazine, Sep 2010, pp 22-29.

[2] MQ Telemetry Transport (MQTT), http://mqtt.org/

[3] Constrained Application Protocol (CoAP), https://tools.ietf.org/html/draft-ietf-core-coap-18

[4] R. Roman, P. Najera, and J. Lopez, "Securing the Internet of Things," Computer, vol. 44, no. 9, Sep. 2011, pp. 51-58.

[5] ECMAScript Programming Language, http://www.ecmascript.org/

[6] Node.js framework, http://nodejs.org/

[7] T. Grønli, G. Ghinea., M. Younas, "A Lightweight Architecture for the Web of Things," 10th International Conference, MobiWIS 2013, Paphos, Cyprus, Aug 26-29, Proceedings, pp 248-259.

[8] Node-RED, A visual tools for wiring the Internet of Things, http://nodered.org/

[9] Web Applications (WebApps) Working Group, http://www.w3.org/2012/webapps/charter/

[10] R. Fielding, R. Taylor, "Principled Design of Modern Web Architecture," ACM Transactions on Internet Technology, vol 2, no. 2, May 2002, pp 115-150

[11] The WebSocket API, http://www.w3.org/TR/websockets/

[12] Web Workers, http://www.w3.org/TR/workers/

[13] Network Service Discovery, http://www.w3.org/TR/discovery-api/

[14] System Applications Working Group, http://www.w3.org/2012/sysapps/

[15] User Managed Access (UMA) Home, https://kantarainitiative.org/confluence/display/uma/Home

[16] The OAuth 2.0 Authorization Framework, http://tools.ietf.org/html/rfc6749