Bridging Web Technologies with M2M Protocols

Charalampos Doukas, CREATE-NET

Juan Luis Pérez, Alvaro Villalba and David Carrera, BSC

COMPOSE Project

The Problem

The “Internet of Things” (IoT) is an emerging global information service architecture, which will likely be one of the most important technological advances of this century impacting a wide range of fields. IoT mainly consists of ‘things’ or smart objects that have the ability to communicate over the Internet with applications and services. The application and device management backbone needed to achieve inter-device and Internet communication can be provided by cloud computing, which facilitates scaling and provides support to billions of connected objects.

Interaction with smart objects and remote services requires the utilization of IoT technologies. Smart objects do not feature only sensing devices but can also integrate various actuators (switches, motors, relay circuits) that need to be communicated by the external services. Thus, bi-directional communication mechanisms are needed that take into account the limited resources of smart objects (low cost, low power hardware, etc.) and can also be deployed behind network firewalls, NATs, etc. In other terms, this is also refered to as Pub/Sub notion. A device, or a software component, can publish data (send) and subscribe to receiving data from a remote endpoint without the need to continisuly poll that endpoint for receiving new data.

While HTTP-based RESTful interfaces are commonly used and established, they do not provide appropriate means for bi-directional communication (i.e. sending notifications and commands back to the device) and developers often rely to vendor specifc solutions (e.g., Apple notification center). There are however dedicated Machine2Machine (M2M) protocols, like MQTT [3], CoAP [2]and WebSockets that address such issues.

The Message Queue Telemetry Transport (MQTT) is a lightweight Pub/Sub, open messaging protocol (http://www.mqtt.org). It can be used with low power sensors due its low protocol overhead. It enables the transfer of telemetry-style data in the form of messages from pervasive devices, or constrained networks, to a server or small message broker. Pervasive devices may range from sensors and actuators, to mobile phones or full-scale computers. This protocol addresses all issues of:

- Device identification and Network scalability (unlimited client devices can be added to the network of a broker)
- Message collection and forwarding between the broker and the client devices, and network status detection (in case of client failures the broker is automatically informed)
- Interoperability between services and devices (different brokers can communicate)

MQTT is a binary transport protocol, thus making it quite suitable for low-resource devices (e.g., low power microcontroller or connected devices with limited capabilities). Compared to aforementioned solutions (WebSockets and CoAP) it is more leigtweight

and is already established and used in commercial and other applications.

The most trivial interfaces for IoT applications are web or smartphone-based. While there are many client implementations for smartphone development platforms, there is no direct way to communicate MQTT with web-based applications.

The COMPOSE Solution

In COMPOSE Project (http://www.compose-project.eu) we have faced this challenge, since we are buiding a platform that provides RESTful APIs for sensor data streaming and processing, and at the same time we are integrating MQTT (as well as other M2M protocls) for maximising device interoperability.

While there are efforts to resolve this challenge at a protocol bridging level (see the proposed Eclipse Ponte project: https://projects.eclipse.org/projects/technology.ponte), there is no clear solution at thw application level. A REST API implements the notion of POST/PUT and GET for storing and reading sensor or device in general data. In MQTT how can a pubish request be transformed into a POST or GET request? An how a REST call can be transmitted to a specific device (or set of devices) using a subscription?

Within COMPOSE, and more specifically the ServIoTicy platform (http://www.servioticy.com), we resolve this issue through embeding the type of the request and the data itself inside a JSON structure within the MQTT message. In the following example:

```
{
   "meta": {
     "authorization": "API_TOKEN HERE",
     "method": "PUT",
     "url":
"/13944639303973f5530eab507412d88bef305089a7720/streams/weather"
   },
   "body": {
     "lastUpdate": 1199192932,
     "channels": {
       "location": {
         "current-value": "40.12,-71.34",
         "unit": "degrees"
       },
       "temperature": {
         "current-value": 33,
         "unit": "degrees"
       }
     }
   }
}
```

we store two sensor readings (location and temperature) for a specific device that

contains a weather sensor.

Despite having to construct a larger JSON message than in the case of sending the request through the REST API, the use of MQTT results in 6% less usage of program memory in the case of Arduino microcontroller platform (experiment conducted using an Arduino Uno, with Ethernet as network interface, based on the PubSub Library (https://github.com/knolleary/pubsubclient), and publishing the message above, versus constructing a PUT HTTP request for updating the same sensor data).

The most important feature though is the ability with the same code to receive data from the COMPOSE back-end without the need of polling the server. By defining the following action at the description of the device on the back-end:

```
{
    "actions": [
        {
            "name": "blink",
            "description": "Blink LED on Arduino"
        }
    ]
}
```

and by issuing a request to

http://api.servioticy.com/13944639303973f5530eab507412d88bef305089a7720/actuations/blinl
we can deliver a ‘blink’ command to an Arduino device that is connected to the ServIoTicy broker, subscribed to the topic “13944639303973f5530eab507412d88bef305089a7720”.

COMPOSE also provides open source libraries and SDKs for popular development platforms, including cross-domain smartphone platforms, and embedded device libaries. All components can be accessed at https://github.com/compose-eu

[1] Carlson, D.; Schrader, A., "Dynamix: An open plug-and-play context framework for android," Internet of Things (IOT), 2012 3rd International Conference on the , vol., no., pp.151,158, 24-26 Oct. 2012.

[2] Pereira, P.P.; Eliasson, J.; Kyusakov, R.; Delsing, J.; Raayatinezhad, A.; Johansson, M., "Enabling Cloud Connectivity for Mobile Internet of Things Applications," Service Oriented System Engineering (SOSE), 2013 IEEE 7th International Symposium on , vol., no., pp.518,526, 25-28 March 2013

[3] Hunkeler, U.; Hong Linh Truong; Stanford-Clark, A., "MQTT-S — A publish/subscribe protocol for Wireless Sensor Networks," Communication Systems Software and Middleware and Workshops, 2008. COMSWARE 2008. 3rd International Conference on , vol., no., pp.791,798, 6-10 Jan. 2008