DDS: Ubiquitous Data Sharing for the Web of Things

Angelo Corsaro, PhD
Chief Technology Officer
PrismTech
OMG Architectural Board
OMG DDS SIG Co-Chair
angelo.corsaro@prismtech.com

Web Standards and technologies have been evolving at an amazing pace over the past few years. Most notably, HTML5 [1] and related standards are providing web applications with unprecedented control over computational resources (e.g. Web Workers) and communication ressources. As an example, Web Sockets have brought light-weight and efficient push communication to web applications and upcoming standards such as the Raw Socket API [2] will provide further control over networking along potentially enabling broker-less peer-to-peer web-application communication.

At the same time, if we look at the standard landscape, no standard addresses yet a simple yet ubiquitous need of virtually any web of thing application: Data Sharing. Currently, most web applications rely on low level abstractions to create their own data sharing framework or leverage proprietary solutions.

Recently, an implementation of the Data Distribution Service (DDS) Standard [3] has been introduced that allows Web Applications share data efficiently and using high level abstractions. In addition, taking advantage of the fact that DDS is already supported over any kind of device, from embedded boards to high end servers, this allows effective and ubiquitous communication across web, mobile, embedded, cloud and enterprise applications.

As an example, below is reported the code snippet that would allow a temperature sensor to share its value and a web application to read it and display it.

```
// Suppose the sensor is connected to a RaspberryPI over
// which we run an embedded JVM. To keep the example
// compact our code is written in Scala but C++, Java,
// C#, etc.  are also possible.

// Define the Topic
val topic  = Topic[TempSensor](“TemperatureSensor”)

// Define a DataWriter for the temperature sensor
val dw = DataWriter[TempSensor](topic)

// Write a value
val currentValue = sensor.value()
dw.write(v)
```

Below is reported the snipped of code of a simple applications that just shows the current temperature as a text element on a web page.

```
// For convenience we show the code in CoffeeScript


// Create a runtime
runtime = new Runtime();

// Handle the on-connected event
runtime.onconnect = () ->
      // Define the topic
      topic  = Topic(“TemperatureSensor”,
                     “com.acme.sensors.TempSensor”)
      // Define the topic
      dr = DataReader(topic)
      dr.addListener(
        (d) -> $(“#tempElem”).html(formatTemperature(d))
       )


// Connect the runtime to either a PaaS implementation of DDS
// or to enable peer-to-peer communication if Raw Sockets are
// supported
runtime.connect(addr)
```

The conceptual simplicity and elegance of the code above should be compared with the effort that would be required to achieve the same goal using currently available web standards.

Thus the questions to ask is whether the time has come for some cross contamination between standards? Should DDS become a web standard too?

This presentation, will introduce the abstractions that DDS provides toward building Web of Things applications and will discuss the idea of adopting the DDS abstractions as an additional Web Standard for efficient and ubiquitous data sharing across just about anything.

[1] HTML5 http://www.w3.org/TR/html5/

[2] Raw Socket API http://www.w3.org/TR/raw-sockets/

[3] The Data Distribution Service http://www.omg.org/spec/DDS/Current