Dioptase Overview

The Internet of Things (IoT) is a promising concept that may radically change the way people interact with the physical world. However, the IoT paradigm raises tremendous challenges. One such challenge is the continuous processing of data streams presented by Things, which must be investigated urgently because it affects the future data models of the IoT. This cross-cutting concern has been studied extensively in the context of Wireless Sensor Networks (WSN) given the focus on the acquisition of data from the physical world. However, the WSN technologies are heterogeneous and complex, which represents a hurdle to their wide deployment. There is thus a need for a middleware solution for data stream management that leverages existing WSN work, while extending it using the enabling technologies of the Internet, both from network and application perspectives, in order to improve the flexibility and the interoperability of the future IoT.

Toward the above goal, we introduce Dioptase, a Data Stream Management System for the Internet of Things, which aims to facilitate the integration of the Things and their streams into today's Web by presenting sensors and actuators as services. The middleware specifically provides a way to describe complex stream-based mashups and to deploy them dynamically, at any time, as task graphs, over available Things of the network, including tiny ones.

Dioptase Source Code

The source code of the prototype of Dioptase can be downloaded at:
https://www.dropbox.com/s/qmf4mpq9y0syhv8/dioptase.7z?dl=1

The 7-zip archive is protected. Please send an email to **benjamin (dot) billet (at) inria (dot) fr** and we will then send you a password.

Demonstrations

https://www.rocq.inria.fr/arles/attachments/article/248/dioptase_demos_jse.zip

The archive contains eight demos applications and their source code. The applications are built for Java SE 7 and include the relevant drivers. The drivers support: TCP/IP communication, memory/file storage and some bogus sensors (random values).

Each demo application can be launched using the corresponding .bat or .sh file according to your operating system. However, in case of failure, you can directly launch the applications using your OS's shell and the following instruction:

java -cp "apps_jse.jar:lib/" fr.inria.dioptase.examples.*

Note thar the class path separator for Windows is ";" and not ":". In addition, you can add the path to your JRE, if your CLASSPATH global variable is not defined.

The demos are the following (filmed versions are made available at the end of this page):

1. SimpleProducer: The demo creates a Producer from the bogus temperature sensor (sample rate = 1 reading every 2 seconds) and displays the acquired values.

2. SimpleProcessor: An interpreted task is locally deployed and computes an average value over the temperature stream.

3. DynamicOperator: Using the middleware services, a compiled task (COUNT) is deployed and continuously counts the number of produced temperature values.

4. DynamicInterpretedTask: Using the middleware services, an interpreted task is deployed and removes the temperature values that are less than 10°.

5. SimpleStorage: Using the middleware services, a memory storage is deployed and connected to the temperature producer for a fixed period of time. Three requests are then performed: get all values, get the five last temperature values, get the values that were created within the last two seconds.

6. MetadataServices : The demo invokes each metadata service and displays the available sensors/actuators description, the available operators and task instructions, and a schema of two chained processors (a COUNT task reads the output of an AVERAGE task that reads the output of the temperature producer).

7. DistributedComputation : The demo builds five instances of the middleware (port 5000 to 5004). Five components are then deployed: On the instance 5000, a temperature producer; On the instance 5001 and 5002, a COUNT task and a SUM task, which read the temperature producer through the remote connector; The instance 5003 computes an average temperature value from the outputs of 5001 and 5002 (SUM divided by COUNT); Finally, the instance 5004 removes the average temperature values that are less than five degrees.

8. Mashup : This demo deploys five producers from the bogus temperature sensor, the bogus light sensor, the CPU load sensor and the memory load sensor. A web page, called mashup.html contains some javascript code to access directly the Dioptase instance (you can directly launch it in your web browser). The web mashup acquires the list of running producers and enables users to get the values produced by these producers.

Dioptase Demo Videos

You may alternatively watch the videos of the above mashup demo at https://www.dropbox.co

m/s/z5cesyc3a00q1i5/demo_videos.zip

In addition, a complete video shows the Mashup demo with two SunSPOTs (deployment, running and sensing): https://www.dropbox.com/s/pdcra0bjfa3fu1f/complete_sunspot_demo.mp4?dl=1 (for a better text quality, please download the video instead of viewing it in the online dropbox player).