Metadata Management for the Web of Things: a Practical Perspective

Carolina Fortuna, Alexandra Moraru, Patricia Oniga, Zoltan Padrah and Mihael Mohorcic

http://sensorlab.ijs.si/
http://sensors.ijs.si/

Outline

• Motivation
• Physical data sources
• Architecture alternatives
  • Embedded approach
  • Middleware approach
• The metadata storage
• The metadata representation
• The metadata access
• Conclusions
Motivation

We identified two types of meta-data that can be useful in processing sensor measurements:

• Technical specifications of the sensor platforms
  • types of sensors, units of measurement for the observed phenomenon, range of sensors, performance in term of accuracy, stability of measurements, etc.

• Contextual specifications of the sensor platforms
  • environment where the platform is deployed (i.e. indoor or outdoor), placement conditions (i.e. outdoor under shelter, under water, exposed to turbulences), time of deployment, etc.
Motivation

• How do we know what a sensor measures?
• Typical platforms use abbreviations which are hard to recognize automatically:

```
"tmp",
"temp",
"temperature",
"t"
```

```
"location":
{
  "disposition":"fixed",
  "ele":"23.0",
  "name":"office",
  "lat":51.5235375648154,
  "exposure":"indoor",
  "lon":-0.0807666778564453,
  "domain":"physical"
}
```
Motivation

• Sometimes additional context is required

What does the “temperature” sensor measure?
  – Air, water or some other temperature?
  – Indoor, outdoor, in a container?

Consistent terminology is required for interoperability.
Physical data sources

• Using standard terminology
  • Open Geospatial Consortium’s Sensor Web Enablement standards
  • NOAA and other datasets published and annotated using these standards

• Non-standard based data sources
  • Existing and emerging middleware solutions such as GSN, Cosm, etc.
Standard based data sources

- Sensors push their observation results
  - to a database (which will be accessed by the SOS service)
  - or directly to the SOS service (Transactional-SOS in this case).

- Sensors are registered, using SensorML, at the catalog service (CAT).
- The user queries the catalog service
  - it responds with a list of SOS service instances that fulfill the requirements.
- The user binds the SOS and retrieves the observation data, encoded in O&M and SensorML.
W3C SSN

- W3C Semantic Sensor Network Ontology
  - Vocabulary that permits higher expressiveness
  - Compatible with SensorML
- Structured vocabularies have high overhead
  - Inconvenient for constrained devices

<table>
<thead>
<tr>
<th>Property</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>directType</td>
<td><a href="http://sensors.ijs.si/semsense/vocab/SensorType-1">http://sensors.ijs.si/semsense/vocab/SensorType-1</a></td>
</tr>
<tr>
<td>hasSubsystemOf</td>
<td><a href="http://sensors.ijs.si/semsense/resource/403AB8FC-1">http://sensors.ijs.si/semsense/resource/403AB8FC-1</a></td>
</tr>
<tr>
<td>isProducedByOf</td>
<td><a href="http://sensors.ijs.si/semsense/resource/LastSensorOutput-403AB8FC-1">http://sensors.ijs.si/semsense/resource/LastSensorOutput-403AB8FC-1</a></td>
</tr>
<tr>
<td>observedByOf</td>
<td><a href="http://sensors.ijs.si/semsense/resource/LastObservation-403AB8FC-1">http://sensors.ijs.si/semsense/resource/LastObservation-403AB8FC-1</a></td>
</tr>
<tr>
<td>observes</td>
<td><a href="http://sensors.ijs.si/semsense/vocab/vsn_state">http://sensors.ijs.si/semsense/vocab/vsn_state</a></td>
</tr>
<tr>
<td>type</td>
<td><a href="http://purl.oclc.org/NET/ssnx/ssn#SensingDevice">http://purl.oclc.org/NET/ssnx/ssn#SensingDevice</a></td>
</tr>
<tr>
<td></td>
<td><a href="http://sensors.ijs.si/semsense/vocab/SensorType-1">http://sensors.ijs.si/semsense/vocab/SensorType-1</a></td>
</tr>
<tr>
<td></td>
<td><a href="http://www.w3.org/2000/01/rdf-schema#Resource">http://www.w3.org/2000/01/rdf-schema#Resource</a></td>
</tr>
</tbody>
</table>
Architecture alternatives

We looked at two implementation options that would allow using semantic representation for sensor meta-data:

- Embedded approach
- Middleware approach

We compared them with respect to three aspects:

- Storage
- Representation
- Access
Embedded vs. Middleware based Architecture

Embedded approach
• assign an IP address to every device
• Sensors (things) are accessed directly from the web without any intermediates

Middleware approach
• involves using a gateway solution
• interface with the applications
• interface with a network of things
• applications communicate with a middleware that intermediates to the things
• the middleware is responsible to collect metadata from the things and respond to clients’ requests.
Sensor Metadata Storage

**Embedded**
- metadata has been added to the embedded software (firmware) of the devices
- facilitates the automatic gathering of information about the sensors
- stored in the flash memory of the devices

**Middleware**

**Relational Database**
- database schema reflects hardware configuration

**Knowledge base (triple stores)**
- Uses exiting ontologies for data model

<table>
<thead>
<tr>
<th>SN</th>
<th>ID</th>
<th>MAC</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>1</td>
<td>403AB8F9</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>ST</th>
<th>ID</th>
<th>phenomenon</th>
<th>uom</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>temperature</td>
<td>degC</td>
<td></td>
</tr>
<tr>
<td>2</td>
<td>pressure</td>
<td>mbar</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>SA</th>
<th>sn</th>
<th>st</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td></td>
<td>1</td>
<td>2</td>
</tr>
</tbody>
</table>

[Diagram of sensor metadata relationships]
Sensor Metadata Storage

**Embedded**
- JSON-LD in our implementation
- The message occupies 15KB out of the 512KB of flash memory (~3%).
- For comparison
  - the CoAP application code takes about 25 KB and the full system image
  - the operating system with networking stack and application takes 216KB without optimization and roughly 150KB with compiler optimizations.
- Therefore the metadata uses 10% of the entire system image

**Middleware**
- we implemented it as relational database and as triple store
- if additional metadata is needed at a later time
  - the extension or even redesign of the database
  - in the triple based representation predicate and object types need to be specified later in an ontology
Sensor Metadata Representation

Structured representation of metadata following standardized format

- Solutions typically use a publicly shared schema, vocabulary or ontology
  - SensorML, ZigBee SmartEnergy, RDF (plus domain ontology – e.g. SSN)
- Most of them follow XML syntax
  - Encoding techniques are needed for efficient storage and transmission from the embedded devices
  - Efficient XML Interchange (EXI), Binary XML (BXML), Binary JSON (BSON)

**Embedded**

Structured representation of metadata

- Disadvantage of occupying a notable proportion of the device’s memory resources,
- It is required for integration purposes

**Middleware**

Structured or custom representation

- Structured -> fast integration with different storage systems
- Custom -> good when there are memory and energy constraints; required specialized parsers and wrappers
Sensor Metadata Representation

- Implementations
  - Custom representation (a)
  - JSON-LD representation (b)

A sensor node that has two types of sensors attached to it, measuring temperature and pressure in degree Celsius and millibars respectively.
Sensor Metadata Representation

Semantic Vocabulary

- SSN ontology
- Other Vocabularies/Ontologies: Basic GeoWGS84, GeoNames, Measurement Units Ontology
- Subset of concepts and relationships from SSN
Sensor Metadata Representation

Embedded
• a structured representation
  • disadvantage of occupying a notable proportion of the device’s memory
  • it supports immediate integration into WoT
  • for comparison, the custom format equivalent to the full 15KB JSON-LD message takes only 1KB

Middleware
• the transformation of metadata can be straightforward if properly represented in a structured manner
• permits users to send SPARQL queries and receive the corresponding answer,
  – this is currently not supported in the embedded approach (but some efforts in this direction are underway)
Sensor Metadata Access

**Embedded**
- a RESTful API to the resource containing metadata which returns a structured JSON-LD message
  - The application which generated the call can parse the message using a JSON-LD processor
  - metadata access on the node is done through the resource and not by querying the sensor node URI

**Middleware**
- SPARQL end-points of the triple stores
  - Relational Database combined with D2R server
  - Triple store -> Sesame
Sensor Metadata Access

VESNA Sensor Node running Contiki 2.5 OS

ARM Cortex-M3
clock up to 72 MHz,
1 MHz 12-bit ADC,
1 MB flash, 96 kB SRAM,
128 kB non-volatile MRAM,
SD or micro SD card slot
USB 2.0 and RS-232 interface

uIPv6/6LowPAN/CoAP

2.05 Content (Blockwise)

```
metadata_start {
"SensorNode": 
{"sn_uid": "403AB8FC", "contact_info": "Jozef Stefan Institute", "purpose": "environmental monitoring", "project": "Collaboration with Miren - Kostanjevica", "HW_con" 
["VSN_v1.2.1", "VNS_v1.2", "sw_config": 
"os": "Contiki", "repository_url": "http://xpack.ija.si 
svn/rensontiki...", "version": "2.5", "data_message_dec" 
{"message_format": "key-value 
pair", "pair_separator": ":", "key_represents": "sensorId"}, 
"version": "0.1"}, "operation 
mode": "sensormode", "deploymentStartDate": "2011-08-25T16 
,deploymentEndDate": "2011-08-25T17:10-02:00", "Sensors 
["sensor_id": "c1", "sensor_type": "shht1_temperature", "si" 
um": "air", "observed_property": "temperature", "samplingMo" 
and": "output", 
{"data_type": "TYPE_DOUBLE", "uom": "degC"}, "MeasuringCapa 
{"minValue": "-40", "maxValue": "128.3", "resolution": "0.01", "acq 
0.3", "precision_..."}, "isaqma"
{"sensor_type": "virtual_sensor", "sensor_id": "s10", "samp 
battery", "observed_property": "voltage", "samplingMode 
measurement_interval_unit": "seconds", "measurement_into 
{"sensor_type": "virtual_sensor", "sensor_id": "s12", "samp 
node", "observed_property": "ID", "samplingMode": "alway 
measurement_interval_unit": "seconds", "measurement_interval": "15 
on": 
"location_id": "lala", "lat": "20.456", "lon": "32.45", "al" 
locationType": "fixed", "description": "under the bridge"} 
metadata_stop
```
Sensor Metadata Access

Embedded

- Constrained communications links
  - High packet loss rate (5-10%)
  - low datarate (10s kb/s)
- lack of or slow response from the thing
- inconvenient for the web application developer
- Retrieval of the JSON-LD message containing the metadata of the device took 12 seconds
  - approximately 15 KB in over 200 chunks
  - 100 kb/s link
- another inconvenience related to constrained devices is the inability to process multiple requests at the same time.
Conclusions

- Both the embedded and the middleware solutions can be prototyped.

- The current state of the art in sensor devices limits the amount of stored data:
  - XML-like syntax is not well suited for storing.
  - Constraints are imposed by the communication links.

- The middleware approach is more convenient from the perspective of web application developer:
  - Technological developments empowering the embedded approach may be catching up soon.
Lab:  http://sensorlab.ijs.si/
WoT Mash-up:  http://sensors.ijs.si/
Contact:  carolina.fortuna@ijs.si