Ontology-based Virtual IoT Devices for Edge Computing

The 8th International Conference on the Internet of Things (IoT 2018)
Santa Barbara, California, USA
October 15–18, 2018

Kristina Sahlmann, Thomas Schwotzer
Heterogeneity of network devices

Source: https://goo.gl/images/tyaQo1
Idea

• aggregate self-descriptive devices at the edge of the network
• furthermore
  • derive new services
  • delegate requests/responses
  • collect and pre-process sensor data
  • generate events
Definition: Virtual Device

• no actuators and no sensors onboard
• deployed at the edge of the network
• hides real devices (information hiding principle)
• aggregate capabilities of real devices
• offer aggregated capabilities
System Architecture of a Virtual Device

Device Description: oneM2M Ontology

Virtual Device

Device 1

Device 2

NETCONF-MQTT bridge

Application Clients

Topics for Publish/Subscribe
Topology of Virtual Devices

Mobile Client

Client

Virtual Device 1

Virtual Device Building

Virtual Device 2

Floor 1

Floor 2

Ontology-based Virtual IoT Devices for Edge Computing
Operations and Tasks of Virtual Devices

• Scenario 1: Semantic Deployment
  • collect and aggregate device capabilities from device descriptions
  • infer new system capabilities from device descriptions
  • generate a virtual device description
  • publish this description e.g. to the MQTT broker

• e.g. define a new aggregated capability
  “switch off all lights in the 1st floor”
Operations and Tasks of Virtual Devices

• Scenario 2: Controlling and Measuring Services
  • map aggregated functionalities to the single device functionality
  • delegate controlling and measuring functionalities

• e.g. execute a controlling service
  “switch off all lights in the 1st floor”
Operations and Tasks of Virtual Devices

- Scenario 3: Aggregation of Sensor Data
  - collect and pre-process sensor data
  - aggregate sensor data
  - calculate average value

- e.g. calculate an average temperature in the 1st floor per hour
Operations and Tasks of Virtual Devices

• Scenario 4: Alarm/Event Trigger
  • monitor sensor data
  • prove conditions
  • create push events
  • trigger alarms if necessary

• e.g. raise alarm when temperature is over 30° Celcius
Software components of a virtual device

Virtual Device

- Semantic Deployment
- Service Delegation
- Aggregation of Sensor Data
- Event & Alarm Generation

MQTT Client

Input → Output
Implementation Scope

• Scenario 1: Semantic Deployment
• Scenario 2: Controlling and Measuring Services
Gap between the generic oneM2M and specific Device Ontology

oneM2M Ontology

Device Description
Extended oneM2M Ontology
1st level
Extended oneM2M Ontology 2nd level
Inference possibilities

• oneM2M ontology is defined in OWL 2 DL flavour
• OWL constructs:
  <owl:Restriction>, <owl:someValuesFrom>,
  <owl:sameAs>, <owl:equivalentClass>, etc.
• SWRL rules
• SPARQL requests
OWL Abstract Class for switchOff Functionality

```xml
<owl:Class rdf:about="service#AbstractSwitchOffFunctionality">
  <rdfs:subClassOf rdf:resource="onem2m#ControllingFunctionality"/>
  <rdfs:subClassOf>
    <owl:Restriction>
      <owl:onProperty rdf:resource="ext-onem2m#hasControllingFunctionality"/>
      <owl:someValuesFrom>
        <owl:Class>
          <owl:oneOf rdf:parseType="Collection">
            <onem2m#ControllingFunctionality rdf:about="dev#switchOff"/>
            <onem2m#ControllingFunctionality rdf:about="dev#turnOff"/>
          </owl:oneOf>
        </owl:Class>
      </owl:someValuesFrom>
    </owl:Restriction>
  </rdfs:subClassOf>
</owl:Class>
```
SWRL Rule for switchOff Functionality

\[
\text{onem2m:Device(?d)}
\]

\[
\wedge \text{service:hasControllingFunctionality(?d, ?f)}
\]

\[
\wedge \text{swrlb:containsIgnoreCase(?f, "switchOff")}
\]

\[\rightarrow \text{AbstractSwitchOffFunctionality(?f)}\]

\[
\text{onem2m:Device(?d)}
\]

\[
\wedge \text{service:hasControllingFunctionality(?d, ?f)}
\]

\[
\wedge \text{swrlb:containsIgnoreCase(?f, "turnOff")}
\]

\[\rightarrow \text{AbstractSwitchOffFunctionality(?f)}\]
SPARQL Request for switchOff Functionality

PREFIX onem2m: <http://www.onem2m.org/ontology/Base_Ontology/base_ontology#>

SELECT ?functionality ?type 
WHERE { 
  ?functionality rdf:type owl:NamedIndividual.
  ?functionality rdf:type onem2m:ControllingFunctionality.
  FILTER REGEX (?functionality, "off", "i"). 
}
Implementation

• Python, RDFLib incl. SPARQL
  • No Python Implementation of SWRL
• Virtual Device on Raspberry Pi 3B
Evaluation on IoT Devices

- CC2538 Development Kit from Texas Instruments
  - Light Sensor
  - Led
  - 6LoWPAN
- Arduino Yún Rev 2
  - Light Sensor
  - Humidity Sensor
  - Relay
  - Wi-Fi
Conclusion on Virtual Device

• information hiding principle
• reduce complexity
  • aggregate capabilities of real devices
• scalable architecture
• model-driven
Conclusion on Ontology

• oneM2M Base ontology must be extended
• only core vocabulary/schema is not enough
• describe the capabilities not only things (how vs. what)
• further classification of things and functionalities needed, e.g. subclasses, abstract classes, patterns
• but leave space on customization
Thank you!
Any questions?
Kristina Sahlmann
sahlmann@uni-potsdam.de
sahlmann@htw-berlin.de
An ontology-based NETCONF-MQTT bridge

Device Description: oneM2M Ontology

Introduced at InterOSS-IoT 2018, Bilbao, Spain