Middleware for the IoT - TP 3

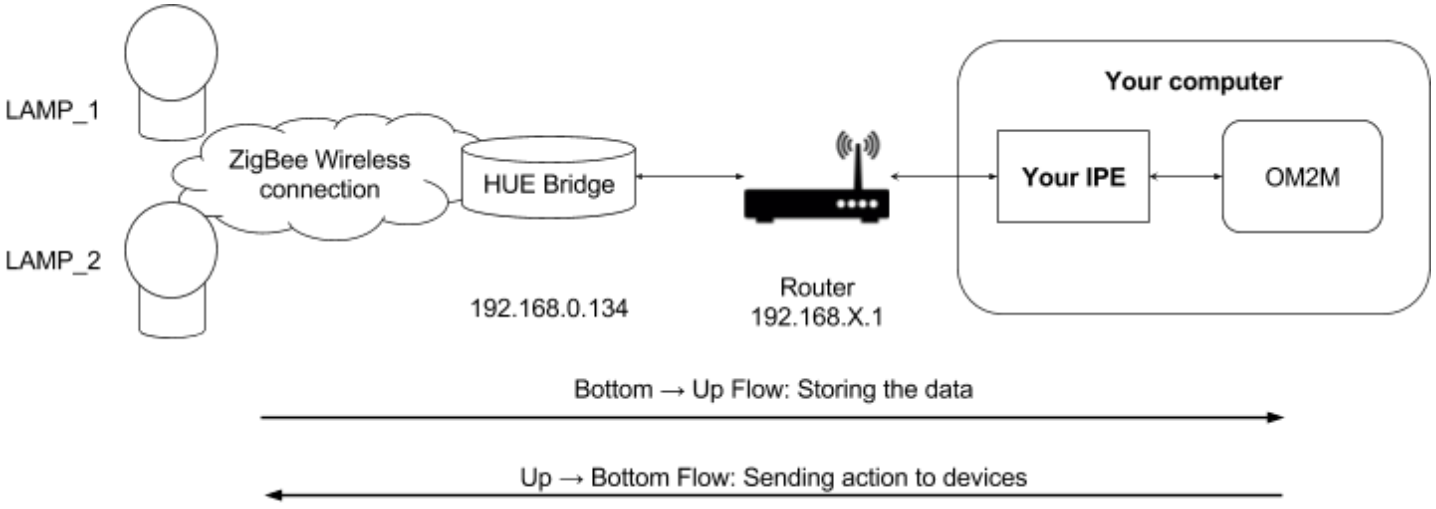
Interworking Proxy - Application to Philips HUE

Objectives

In this session you will learn how to **connect a new or already existing technology to a oneM2M system**. To do so, we will use some Philips HUE lamps that use the ZigBee technology between the bridge and the lamps and provides a **REST API** at the bridge level. Then you will use what you developed during the previous session to interact with OM2M (Client + monitor) in order to create an interworking proxy entity that will make the connection between **HUE** and **oneM2M**.

In the end you will create an ADN (Application Dedicated Node) that will interact with the middleware through the HTTP REST API.

Architecture to deploy



Information about the provided classes

Package HUE

HueProperties

- regroups a few properties you will store and use such as IP address of the bridge and user id (generated by the bridge and that you will have to store)

HueSdkListener

- Entity that will listen to the bridge and have some callbacks that will be called when a specific event occurs.

HueUtil

- Some samples of methods you will have to fill to go further

Package IPE

Controller

- This class will implement the method that will execute a received command. This command will be based on the query string provided at the creation of the DESCRIPTOR of the Application Entity.

Monitor

- This class is usually in charge of monitoring the targeted devices and push the new states values to the middleware. In the case of HUE lamps, a callback will be called when the cache is updated (onCacheUpdated in the SDKLIstener). You will just have to implement the method that pushes the values.

Model

- This class stores the object model you will use to represent your system. The Descriptor instance is incomplete, you will add new functionalities to the light as far as you go further in the implementation of your IPE.

IpeServer

- This is an already implemented server that will start and listen to a specific port and context (you have to configure this in the constants). Then it will send to your controller any received request to execute on the HUE system.

1. Learn to interact with the Philips HUE Lights

To help you develop your IPE, philips provide an SDK for Java. This SDK has been integrated in the project available in Moodle. Download it and import it. The aim is to integrate there what you have done during the TP2.
To find more documentation about HUE for developers go to: https://www.developers.meethue.com/ (you will have to register if you want to get the full documentation).
You will need to access to this specific page to have more detailed information about the SDK by Philips.
https://www.developers.meethue.com/documentation/java-sdk-getting-started
Use the provided methods to learn to use the HUE tools in Java.

Implementation

Firstly, try to understand what the **`HueSdkListener`** class is doing.

Then, to learn to use the HUE library you will have to fill the incompletes methods in the **`HueUtil`** class (hue package). Validate and test your methods with the IpeLauncher class.

- add the HueSdkListener to the Sdk
- connect to the HUE bridge (connection info will be provided by teacher (IP of the bridge)).
- discover the HUE devices: use the cache from the bridge to get the available Lights (Java objects) and know how many are connected.
- Get the last known state of a device.
- Try to change the state of a Light (switch on / off, change the color). Use the **`PHLightState`** to do so.

2. Interworking Proxy Entity

The role of the Interworking proxy entity is to map the values coming from the devices and to enable the execution of operations on the device directly from a oneM2M system. To do so you will have to implement several elements

2.1 Create oneM2M resources representing the devices

Now that you know how to interact with the HUE ligths, you will have to create their representation in the oneM2M system.
The first steps are:

- Create an AE for each LAMP
- Create a DESCRIPTOR Container
- Create a Description content Instance (use the Model)
- Create a DATA container

The descriptor provided in the Model class is still incomplete. You will add more operations to it once you succeeded to map the first one correctly and at your convenience.
You will have to put in the POA of your AEs, the point of access of your IPE server that will receive the requests later.

2.2 Store an instance of DATA (bottom to up)

Now that you created the AE corresponding to your devices, implement the missing link between the onCacheUpdated method in the hue listener class using the monitor.
You will have to implement the following method in the Monitor to do so
public void pushState(String appId, String state, String color, int hue) ;
Get the last known state of each lamp and use the **requestSender** and the model to send the new instances of data.

2.3 Change the state of a device (up to bottom)

Now that you are able to send the values coming from the lights to the middleware, you need to implement the up to down part. To do so, you will have to implement the **controller**. The controller will receive the elements you indicate in the descriptor when you create your AE representing your device (via the queryStrings).

If you want to add an operation, you can add it to the descriptor. You will just have to implement it in the controller to be effective when clicking on the button in the Web interface of OM2M! (such as add another color)

Use the previously implemented method in the **HueUtil** class.

3 Integration in the IPE Launcher

Now all components has been developed, the aim is to finalize the java program to integrate both communication with HUE and listening server part.
This final launcher has to integrate:

- Connect to HUE Bridge
- Create OM2M resources representing the connected lamps
- Start the Server

Test your implementation by looking at OM2M resources with the web interface and send command to the LAMP via the same interface.

What you learned

- interact with the Philips HUE Lights in Java with the Philips SDK
- Integrate the Philips HUE technology to a oneM2M platform - OM2M with a REST client.
- Implement a oneM2M IPE - Interworking Proxy Entity

More information on IPE development

Integrate your IPE directly as an OM2M plugin (advanced development)

Now that you manipulated oneM2M Applications that interacted with OM2M, you can learn to integrate your app / IPE / whatever directly in the software. To do so, you can find more detailed information following this advanced tutorial: https://wiki.eclipse.org/OM2M/one/Developer

OM2M is based on the OSGi framework. This framework enables the platform to be structured as a set of plugins (bundles) that can be installed / started / stopped at running time. The idea is that OM2M will have a set of mandatory plugins to run and then run different plugins regarding to what is needed at a specific moment with a specific context.

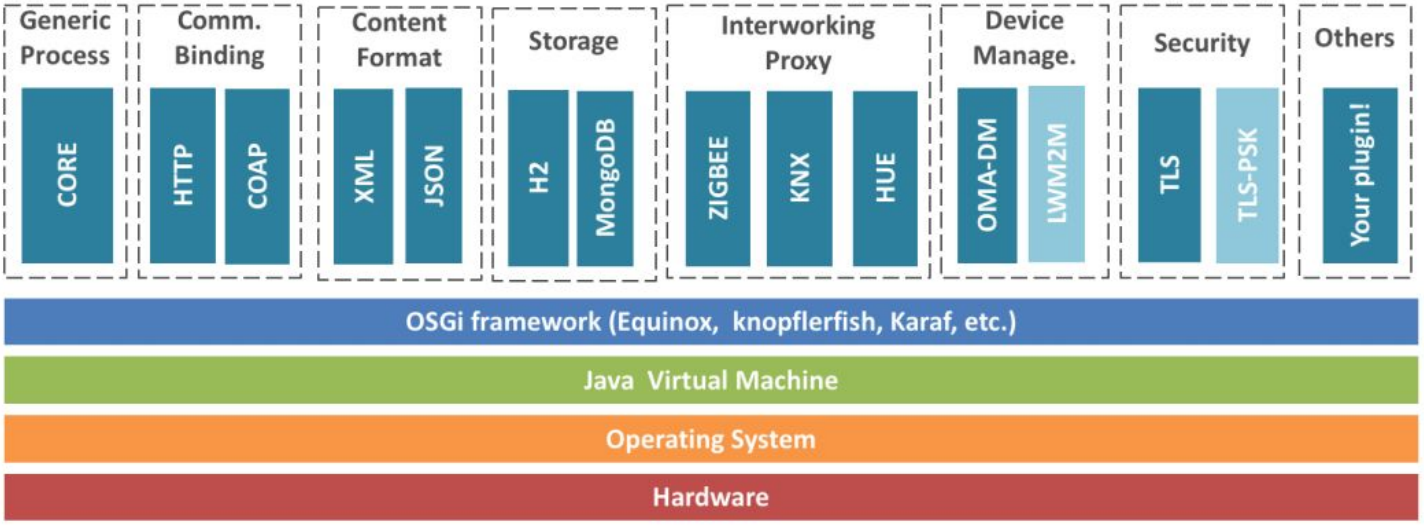


Figure : Diagram of OM2M components with Interworking Proxy Examples

If you are interested to go further into details with this part, ask teacher during the session and / or do not hesitate to contact us!