Middleware For the IoT – TP1/2

1. Objective

The goal of this lab is to explore the capabilities of the MQTT protocol for IoT. To do that you will first make a little state of art about the main characteristics of MQTT, then you will install several software on your laptop to manipulate MQTT. Finally, you will develop a simple application with an IoT device (ESP8266) using the MQTT protocol to communicate with a server on your laptop.

2. MQTT

Based on web resources and the previous course respond to those questions:

- What is the typical architecture of an IoT system based on the MQTT protocol?
- What is the IP protocol under MQTT? What does it mean in terms of bandwidth usage, type of communication, etc ?
- What are the different versions of MQTT?
- What kind of security/authentication/encryption are used in MQTT?
- Suppose you have devices that include one button, one light and luminosity sensor. You would like to create a smart system for you house with this behavior:
  - you would like to be able to switch on the light manually with the button
  - the light is automatically switched on when the luminosity is under a certain value

  What different topics will be necessary to get this behavior and what will the connection be in terms of publishing or subscribing?

3. Install and test the broker

For this TP, we will use the mosquitto broker made by the eclipse opensource foundation (https://mosquitto.org). Mosquitto exists for many platforms: linux, windows, macOS, etc.

- Download and install this broker on your laptop.
- Run the mosquito broker
- Test publish and subscribe with the command shell programs: `mosquitto_pub and mosquitto_sub`

4. Creation of an IoT device with the nodeMCU board that uses MQTT communication

During the TP, we will use thenodeMCU board based on ESP8266.

a. Give the main characteristics of nodeMCU board in term of communication, programming language, Inputs/outputs capabilities

b. Install Arduino IDE on your laptop

c. Add the board NodeMCU 09 (ESP-12 module)

d. Add the library (and necessary dependency) ArduinoMqtt by Oleg Kovalenko

e. Based on examples in the library we will build our own application

- Open the file in the menu: exemples/arduinoMqtt/connectESP8266wificlient and have a look at the different parts of the code. Explain those different parts
- Modify the network characteristics to connect to the local wifi network (cisco38658 or you own wifi network with your smartphone)
- Modify the address of the MQTT server to be able to connect to the broker on your laptop (remember that your laptop should be on the same wifi network)
- Open the serial monitor on Arduino IDE and run your Arduino code, validate that the MQTT connection is done between you device and the broker

f. Add a publish/subscribe behavior in your device

- Open the file in menu: exemples/arduinoMqtt/PubSub
- Extract the specific part: publish and subscribe and necessary declaration and put that in the previous example
- Open the serial monitor on Arduino IDE and run your Arduino code, using the command mosquitto_pub and mosquitto_sub validate that your device publishes values and can receive the result of subscription

5. Creation of a simple application

By using what you did on MQTT and the necessary sensors and actuators, program the application's light management behavior through MQTT exchanges. Button publish state and light subscribe to button state

6. Creation of a complex application

Share a broker with another team and create an application with multiple ESP8266 boards.

7. Write a report

Respond in your report to questions: 2., 4.a, and the core part of 5. In term of publish and subscribe calls.

TP3 – Middleware for IoT Based on oneM2M standard

After the MOOC, you have learnt the different concepts of oneM2M and got information on a specific implementation, Eclipse OM2M. Because oneM2M is a standard, it exists different implementations of the standard that can work together. In those TP we will used another oneM2M stack called ACME (https://github.com/ankraft/ACME-oneM2M-CSE). The objective of this labs is to manipulate the service layer of oneM2M and different resources.

I - First tests with ACME

1) Installation and configuration of an IN-CSE on your laptop

   On the github of ACME project go on installation pages and install acme on your laptop. You will need to have at least python 3.8 installed before (best with the last version of python).

2) Run the IN-CSE
   Have a look on. The configuration pages and on running pages. Open a terminal and run the IN-CSE on your laptop. Test the command to see the resources tree, use also the web interface to see the resources. You can stop the IN-CSE.

II – Manipulation of an acme CSE with a Jupitrer Notebook

1) Install the notebook on your laptop
   Clone the source code of https://github.com/ankraft/onem2m-jupyter-notebooks and then install and configure the Jupiter notebook.

2) Running the jupyter note book as explained on the web pages, for example:
   ```
   jupyter lab --NotebookApp.token='' __START__.ipynb
   ```

3) Start the CSE and notification server and test webUI.

III – Manipulation oneM2M resources

1) Practice the chapter 1 – introduction of the notebook to initialize and get information on CSEBase resource.
2) Practice the chapter 2 – basic resources with the creation of Application Entity, Container and Content Instance

For information the different values of resources type (ty field)

1 accessControlPolicy	12 mgmtCmd	23 subscription
2 AE	13 mgmtObj	10001 accessControlPolicyAnnc
3 container	14 node	10002 AEAnnc
4 contentInstance	15 pollingChannel	10003 containerAnnc
5 CSEBase	16 remoteCSE	10004 contentInstanceAnnc
6 delivery	17 request	10009 groupAnnc
7 eventConfig	18 schedule	10010 locationPolicyAnnc
8 execInstance	19 serviceSubscribedAppRule	10013 mgmtObjAnnc
9 group	20 serviceSubscribedNode	10014 nodeAnnc
10 locationPolicy	21 statsCollect	10016 remoteCSEAnnc
11 m2mServiceSubscriptionProfile	22 statsConfig	10018 scheduleAnnc

3) Write your own program to add a new content instance on the AE: Notebook-AE inside the container: Container. To do that use this python skeleton for example:

```
import sys
import requests
import json

def handleResponse(r):
    print (r.status_code)
    print (r.headers)
    print (r.text)
    return;

def createCIN():
    payload = '{ \
        "m2m:cin": { \
        --- payload ---- 
        } \
    }'
    _headers =   {--- header tags ---}
    json.dumps(json.loads(payload,strict=False), indent=4)
    r = requests.post(--resource URL----,data=payload,headers=_headers)
    handleResponse(r)
```

4) Define your own functions in python to create an application entity, a container and to get the last content instance.

IV – Discovery and group

1) Practice the chapter 3 – discovery, see the interest of label to dynamically discover resources with defined characteristics.
2) Practice the chapter 4 – groups

V – More advanced resources

1) Practice the chapter 5 – access control policy
2) Practice the chapter 6 – notifications

VI – Simulated device

Create an application that simulates the behavior of the device that you have created during the labs 1 and 2 on the ESP8266. You can write this program in python or any other language that allows to create easily REST requests.

VII - Report

1) Explain in a report how you have created your simulated device and give the code.

TP 4 - Fast application prototyping for IoT

Develop a high level application based on real IoT architecture thanks to *Node-RED, oneM2M CSE and MQTT*

1) Objectives

The aim of this last session is for you to fully integrate what you have done in the TP1, TP2 and TP3 and have a complete application that will interact with several real and virtual devices. You will do those activities:

- Deploy a high-level application
- Deploy a concrete architecture with real devices
- Learn to use NODE RED to develop the app faster

The idea here is to have a real life use case where you have devices connected to several technologies. Because you use several protocols and technologies, you have to develop specific interfaces for each device. If you have used the possibility of oneM2M standard and the concept of Interworking Proxy Entity, you will have only one type of protocol to understand and to manage. oneM2M in this case will manage the interoperability between all technologies at the middleware level.

2) Deploy the architecture

Use the work done at TP1/TP2 and TP3. If you haven't the physical device (ESP8266) used at TP1, you can simulate the publication of data on the MQTT broker with the shell command.

3) High level application

To develop a high-level application, you can do it two ways:

- implement everything with your HTTP/MQTT client developed in TP 1, 2 and TP 3, but it will be a bit time consuming.
- or use a tool to do so: e.g. NODE-RED.

Now that you know how to interact with oneM2M through the HTTP REST interface and with MQTT with publish/subscribe mechanisms, you will learn to use a tool to develop high level applications faster.
Use the provided documentation on Node-RED to configure the tool, then use it to create your high-level application that will implement the proposed scenario.

4) Configure and use Node RED

Node-RED is a programming tool for wiring hardware devices, APIs and online services in new and interesting ways.

It provides a browser-based editor that makes it easy to wire flows together using a wide range of nodes in the palette that can be deployed to its runtime with a single-click.

a) Installation

Before installing Node-RED, you must have a working installation of Node.js. We

recommend you to use the latest version of Node.js. https://nodejs.org/en/download/

Once installation of Node.js has finished, we are going to proceed to the installation of Node-RED. The easiest way is to use the node package manager, npm, that comes with Node.js. To use it as a global module, add the command *node-red* to your system path.

npm install -g --unsafe-perm node-red

b) Integration of oneM2M nodes in Node-RED:

Install the LOM2M-Node-red from the gitlab: https://gitlab.irit.fr/sepia-pub/lightom2m. The source of nodes dedicated to oneM2M architecture are in the directory:

…. `/lightom2m/src/node-red`

These nodes are useful to interact with LOM2M but also with any CSE using JSON. The documentation is available on https://gitlab.irit.fr/sepia-pub/lightom2m/-/wikis/LOM2M/Node-RED

Install the node-red nodes in the Node-RED environment with these commands:

go to your node-red directory. Usually it is ~/.node-red.

cd ~/.node-red

npm install "path-to"/lightom2m/src/node-red

After the installation of Node-RED and the integration of LOM2M nodes, you can run Node-RED using the command:

node-red

You can then access the Node-RED editor by accessing this webpage through your web browser: http://localhost:1880.

c) Information about oneM2M node:

Go through the provided documentation on Moodle for the gitlab of lightOM2M : https://gitlab.irit.fr/sepia-pub/lightom2m/-/wikis/LOM2M/Node-RED

You can also use other pre-existing nodes in Node-RED like:

Switch: routes messages to specific outputs based on received data. It makes it possible to compare the latter with certain values introduced by the user to activate an output or block it.

inject: to allow the user to launch his application: on demand, on a specific date and at specific time intervals.

Debug: to display the results returned by the created user application.

boolan_logic: to perform operations of Boolean logic.

node-red-dash-board : to easily create dashboard

node-red-node-twitter : to interact with your social network (your laptop need to be connected to wifi shared with boards and INSA network at the same time)

node-red-node-email : to send email (your laptop need to be connected to wifi shared with boards and INSA network at the same time)

etc.

5) Applications

a) Simple test: get sensors values and display them

Retrieve the last data instance produced by the simulated sensors in TP3 with oneM2M node and Light sensor with MQTT node . For that, use the nodes: Named Sensor Data and Content extractor, MQTT subscribe and Debug.

Extract the value from received data and display them.

b) Sensors and activators

Perform a simple test between luminosity value and a threshold (choose an arbitrary value) using **SimpleCondition** node.

Depending on the previous result you can turn Off /On your lamps on the ESP82666 or simulated devices.

c) Dashboard

Create a dashboard that plots values of sensors with different graphs and buttons to switch on/off leds.

d) Imagine

Use other nodes in Node-Red to connect to your mail, social network, etc.

What you learned

- deploy a concrete architecture with heterogeneous devices

- deploy several oneM2M nodes (IN, MN)
- deploy MQTT nodes
- interconnect heterogeneous devices at the application level
- develop a high level application thanks to node RED

6) Report

a) Export the Node-Red flows that you have created for those applications (menu > export): Sensors and activators, dashboard and Imagine sub-questions and attach them to your report.
b) What are the benefits and drawbacks to build an application with Node-Red ?