Synthèse Middleware for the IoT

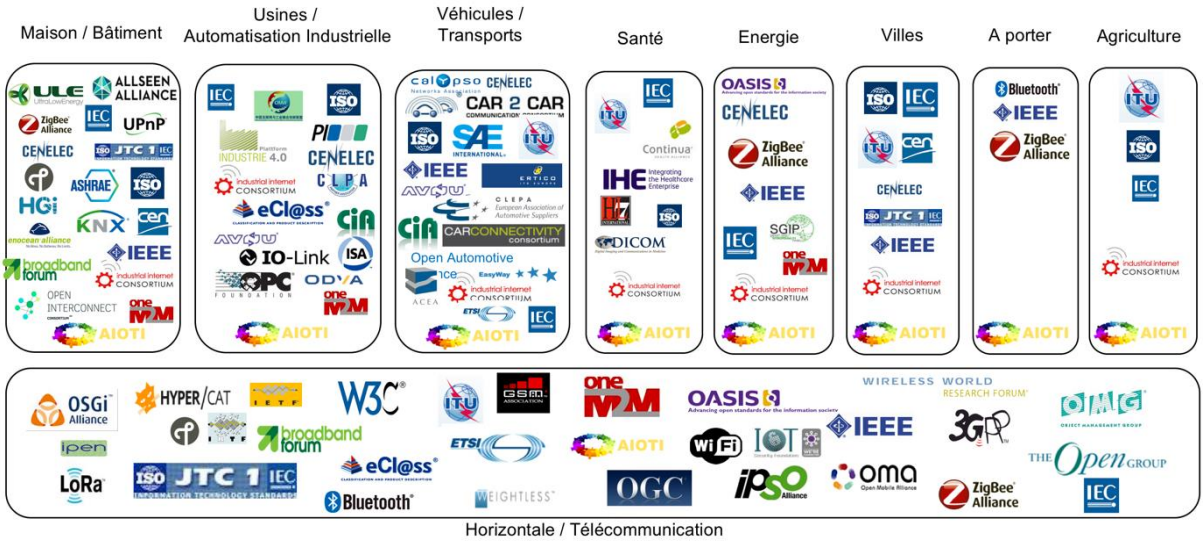
Berrada El Ghali
Bououlid Idrissi Ilias

Comprendre la problématique générée par le secteur IoT :

De nos jours, la majorité des secteurs (immobilier, transport, santé, énergie …) font appel à la puissance de l'Internet des Objets permettant de faire communiquer une multitude d'éléments et de tirer profit des données partagées sur les réseaux. De ce fait, depuis quelques, l'IoT connaît un essor fulgurant. Pour faire communiquer les différentes entités physiques et virtuelles de l'IoT :

- Plusieurs moyens de communications coexistent : le Bluetooth, le ZigBee & Zwave, le Wi-Fi, le cellulaire, Lora et Sigfox
- Plusieurs protocoles de communication coexistent : HTTP, MQTT, CoAP
- Plusieurs protocoles de gestion d'objets coexistent : TR-69, LWM2M
- Plusieurs alliances et consortiums coexistent : Thread, Homekit, Fiware

L'écosystème de l'Internet des Objets est donc très vaste, varié et peuplé de beaucoup de types de standard et de protocoles.
Voici une photo qui illustre l'éventail de l'IoT :



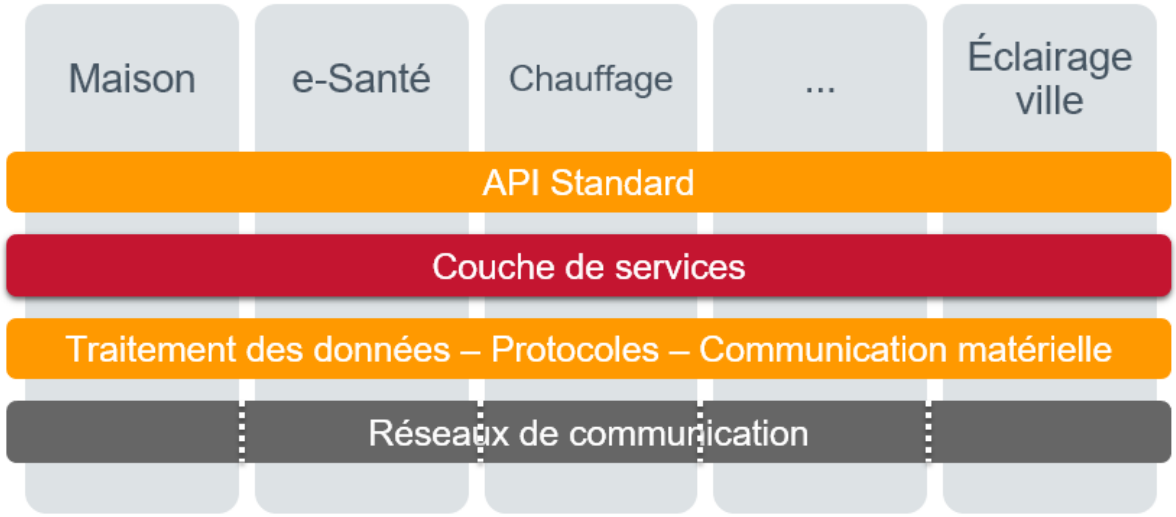
Source: AIOTI WG3 (IoT Standardisation) – Release 1.2

Le manque de standardisation globale dans le domaine de l'IoT génère alors un manque d'interopérabilité.

Toutefois, en 2013, a été formé le consortium oneM2M porté par des organismes internationaux dont le but était de créer un standard global pour l'IoT. C'est un standard ouvert pour plateforme horizontal.

Comprendre l'architecture OneM2M et son intérêt

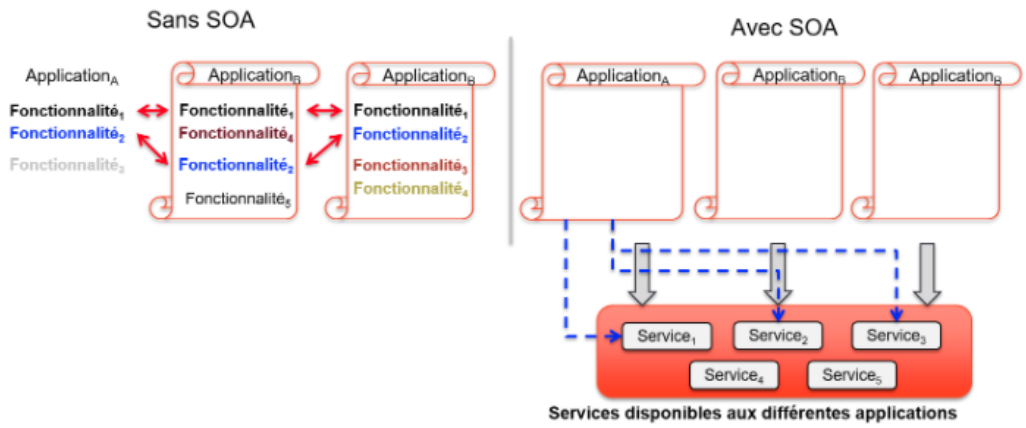
La force du standard OneM2M c'est son horizontalité. Contrairement à certains standards qui facilitent la communication dans un domaine précis (comme Homekit pour la domotique), OneM2M permet la communication entre tout type d'entités quelque soit leur domaine d'application. Ceci permet de garantir une homogénéité des systèmes mais aussi une forme d'interopérabilité.
Voilà comment on peut illustrer ces aspects :



L'architecture OneM2M repose alors sur 3 couches : une couche applicative (avec les AE application entities), une couche services (avec les CSE common services entities auxquels s'enregistrent les applications pour bénéficier de services divers) et une couche transport.
Les systèmes réels basés sur OneM2M s'articulent autour de noeuds : ADN (Application Dedicated Nodes) pour les applications s'exécutant sur les objets, MD (Middle Nodes) pour les gateways, et IN (Infrastructure Nodes)pour les serveur s'exécutant sur le cloud. Les AE représentant les objets, les passerelles ou encore la partie applicative des serveurs, s'enregistrent au CSE généralement présent sur les passerelles et les serveurs. C'est à travers les CSE que les AE pourront ensuite communiquer. s
Le standard OneM2M peut être implémenté sur la plateforme OM2M, c'est ce que l'on verra par la suite.

Comprendre les architectures orientées services SOA et orientées ressources ROA :

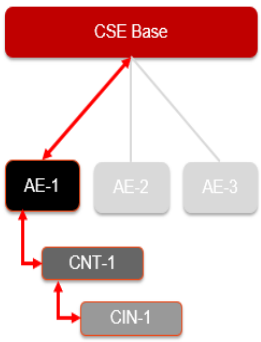
Les applications développées aujourd'hui sont de plus en plus complexes et évoluent de plus en plus vite. Cependant, lors de son développement, des fonctionnalités déjà conçues pour certaines applications sont très souvent requises dans le développement d'autres applications. Il est donc judicieux de favoriser une réutilisation pour un gain de temps et d'argent. De plus, des problèmes d'intégration peuvent apparaître entre plusieurs applications, notamment si elles sont codées avec des langages différents. L'architecture orientée service (SOA) propose d'encapsuler les fonctionnalités d'une application sous forme de couches de service.



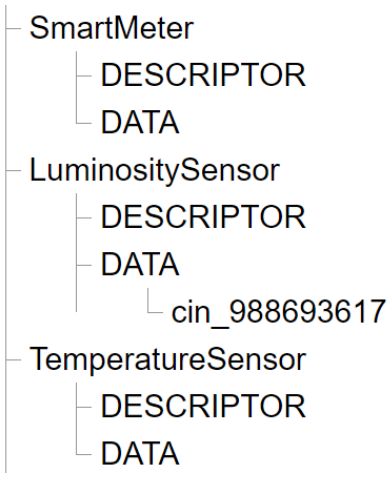
Chaque fonctionnalité est indépendante des autres et peut être utilisée par plusieurs applications. Pour pouvoir accéder à ces services, un service Web sert d'interface et fournit des données. Ce dernier regroupe trois standards :
- WSDL (Web Service Description Language) : Basé sur XML, décrit l'interface d'un service Web (opérations offertes, type de données supportées, protocoles et adresse du service).
- SOAP (Simple Object Access Protocol) : standard de communication qui propose un certain format de message basé sur XML.
- Le protocole http se charge de transporter les messages.
Le soucis est que l'architecture SOA implique l'ajout d'une couche supplémentaire pour la communication à travers le standard SOAP. Non seulement, cela est problématique lorsque la bande passante est restreinte, mais en plus SOAP est basé sur XML qui est particulièrement verbeux.
L'architecture REST, quant à elle, est orientée ressources. Elle se base uniquement sur le protocole http et permet d'interagir avec des services, qui sont en fait des ressources, à travers 4 opérations (GET, POST, PUT et DELETE). Le format des données échangées peut ensuite être du XML, du JSON ou du XHTML.
OneM2M est d'ailleurs basé sur une architecture REST. On y retrouve une hiérarchie de ressources sur laquelle on peut naviguer de manière bidirectionnelle.

Interagir avec les ressources OneM2M :

Comme présenté précédemment, l'architecture OneM2M est basée sur des ressources. Les différents types de ressources sont les CSE (ressources parents), les AE représentant les applications, les Containers CNT permettant de structurer le stockage des données liées à chaque AE, et les Content Instance CIN qui sont des ressources stockant des valeurs.



Nous avons d'ailleurs mis en évidence cette arborescence dans le TP3 à travers l'utilisation de la plateforme OM2M. Nous avons, en effet, implémenté un système composé de capteurs enregistrés sur un CSE, possédant des containers de DATA et de Descriptor (pour décrire le capteurs), ainsi que des CIN de données. En voici l'arborescence :



La question qui se pose c'est comment interagir avec cette arborescence : ajouter, enlever des ressources ou tout simplement préciser le contenu de chaque ressource
Comme précisé précédemment, l'architecture OneM2M repose sur une architecture REST. Ainsi pour interagir avec les ressources OneM2M, il faut utiliser un client REST permettant de soumettre des requêtes HTTP destinées à un CSE (ou une ressource plus spécifique).
Dans le TP3, nous avons eu recours au client REST API Postman, pour construire notre arbre de ressource sur OM2M.

Afin de formuler une requête sur Postman, plusieurs informations sont toutefois nécessaires. Tout d'abord, il faut préciser le type de requête (GET, POST, PUT, ou DELETE) et à qui s'adresse cette requête en utilisant l'URI de la ressource.

```
GET <URI-CIN-1>
```

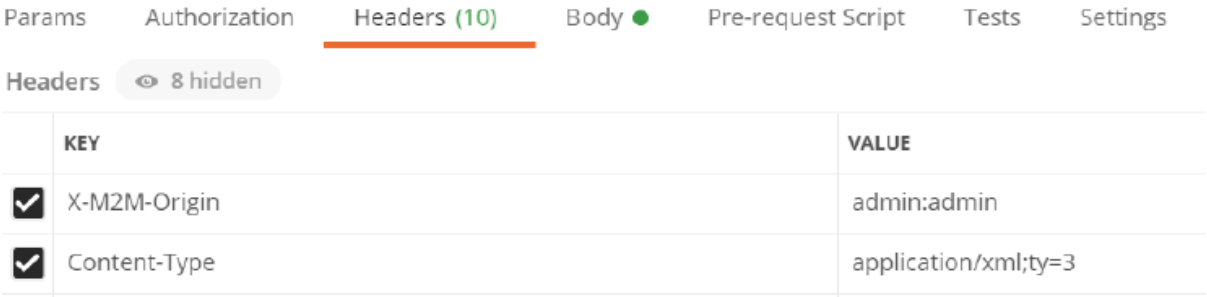
Cet URI est construite de la sorte :

```
/~/<cse-id>/<cse-name>/<resource-name>
```

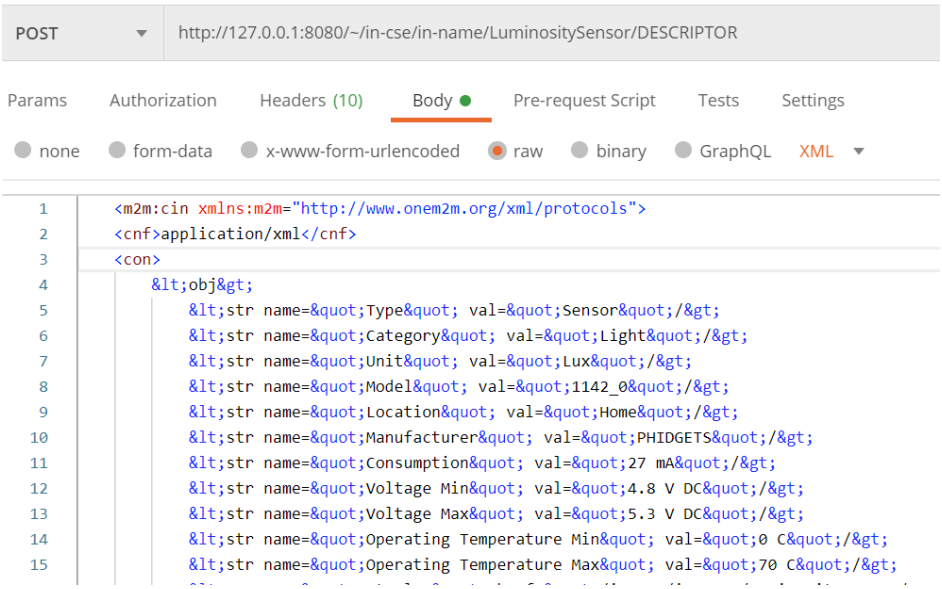
On y trouve le tilde propre à OneM2M, le CSE-id identifiant du CSE auquel on s'adresse, CSE-name nom de ce CSE et resource-name le nom de la,ressource donnée de manière hiérarchique en utilisant des slashs.

Voilà, par exemple, dans le TP3, comment nous avons formulé une requête POST visant à ajouter une ressource fille à la ressource LuminositySensor :

Vous remarquerez que la requête HTTP s'adresse d'abord à une adresse IP avec un numéro de port sur laquelle on a configuré notre CSE.
Sur Postman, on précise aussi dans la partie headers d'où provient cette requête et le type de ressource que l'on veut ajouter ou enlever…



Ensuite, pour préciser le contenu de la ressource que l'on souhaite ajouter par exemple, on doit passer l'onglet Body de Postman. Celui-ci permet d'écrire dans différents langages (XML ou JSON par exemple) le détail de la ressource que l'on ajoute et son contenu.
Par exemple, dans le TP3, on souhaitait ajouter au CNT Descriptor du LuminositySensor un CIN précisant les détails du capteur comme suit :

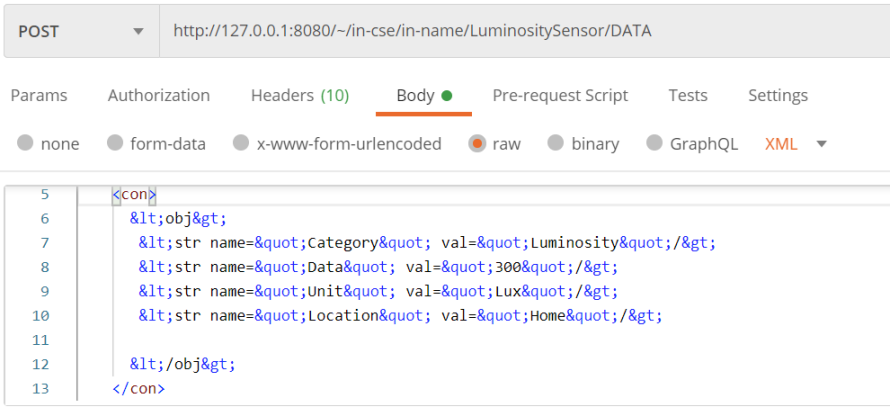


Voilà ce qu'on observe sur OM2M en conséquence :

con

Attribute	Value
Type	Sensor
Category	Lenght
Unit	Meter
Model	1008_0
Location	Home
Manufacturer	PHIDGETS
Consumption	40 mA
Voltage Min	4.8 V DC
Voltage Max	5.3 V DC
Operating Temperature Min	0 C
Operating Temperature Max	80 C
getValue	/in-cse/in-name/SmartMeter/DATA/la

Il en va de même pour créer une instance de data :



con

Attribute	Value
Category	Luminosity
Data	300
Unit	Lux
Location	Home

Comprendre l’intérêt du protocole MQTT :

Dans un réseau d’objets connectés, ces derniers doivent communiquer via différents moyens de communication ou réseaux : Bluetooth, le Wifi, ZigBee, Lora, Sigfox etc.. Les objets communiquant doivent par ailleurs utiliser des standards et des protocoles mis au point par des organismes de standardisation et des consortiums industriels pour communiquer. Nous allons expliciter les trois principaux protocoles de communication :

1) HTTP (Hyper Text Transfer Protocol) : Protocole utilisé pour naviguer sur le Web. Il se base sur le protocole TCP, qui est un protocole avec connexion. Les opérations possibles sont représentées par des requêtes http et peuvent être de type GET, POST,

PUT et DELETE. Une adresse URI permet d'identifier la ressource concernée par la requête.

2) CoAP (Constraint Application Protocol) : Protocole basé sur UDP, donc sans connexion. Son principal avantage est qu'il permet de ne pas surcharger le réseau, très utile pour des équipements en environnement contraint où la bande passante est réduite. Les requêtes sont similaires à celle d'HTTP et il s'agit d'un protocole qui repose sur un modèle client-serveur.
3) MQTT : Protocole qui se base sur un système publier-souscrire. Il se charge de transmettre les données aux clients qui se sont abonnés à un certain topic. Un broker gère la souscription aux topic et la publication de données sur ces topics. Ce protocole est adapté à la mobilité pour des objets à connexion changeante et des adresses IP variables.
   Durant le TP 1, nous avons mis en pratique ce protocole à travers la plateforme open source IOT NodeMCU basée sur le microcontrôleur ESP8266 qui fonctionne en réseau Wifi. Ce dernier a joué le rôle de device. Nous avons tout d'abord installé un broker open source sur l'ordinateur : mosquitto ainsi qu'Arduino IDE qui nous sert d'interface entre le broker et le device. Nous avons ensuite connecté le microcontrôleur au réseau Wifi grâce à un petit programme ainsi que l'ordinateur hébergeant le broker sur le même réseau Wifi. Lorsque la connexion MQTT est réalisée entre le broker et le device en réalisant les réglages correspondants, on peut alors tester le système publier-souscrire :

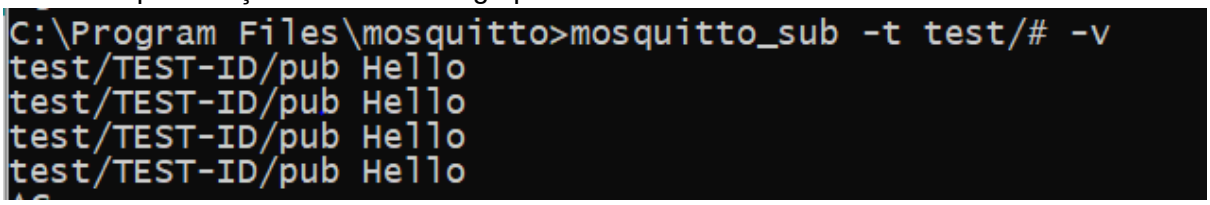
NodeMCU souscrit alors à un certains topic grâce au code suivant :

```
{
  MqttClient::Error::type rc = mqtt->subscribe(
    MQTT_TOPIC_SUB, MqttClient::QOS0, processMessage
  );
  if (rc != MqttClient::Error::SUCCESS) {
    LOG_PRINTFLN("Subscribe error: %i", rc);
    LOG_PRINTFLN("Drop connection");
    mqtt->disconnect();
    return;
  }
}
```

Une publication est effectuée sur le broker sur message « Hello » sur ce même topic :

```
{
  const char* buf = "Hello";
  MqttClient::Message message;
  message.qos = MqttClient::QOS0;
  message.retained = false;
  message.dup = false;
  message.payload = (void*) buf;
  message.payloadLen = strlen(buf);
  mqtt->publish(MQTT_TOPIC_PUB, message);
}
```

Le souscripteur reçoit bien le message publié :



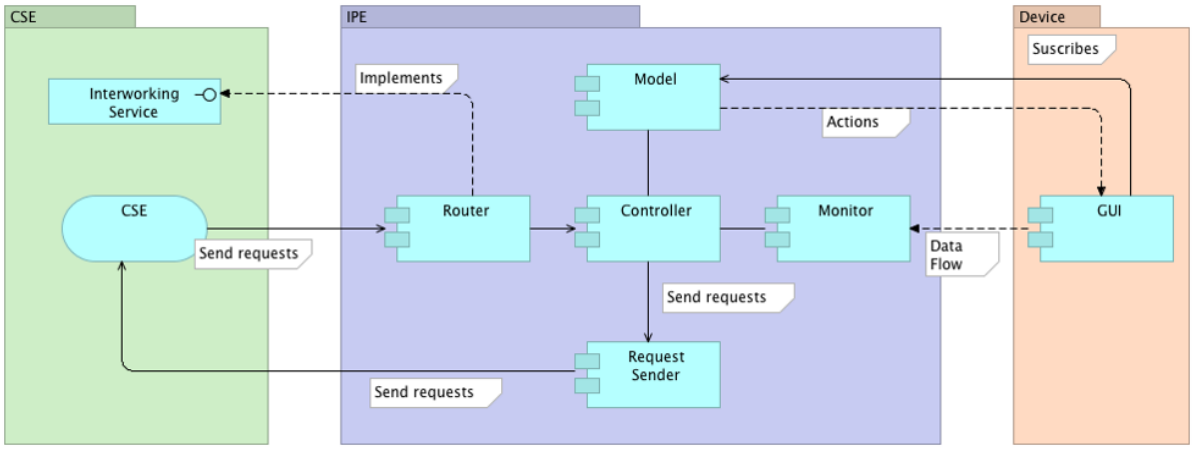
Nous pouvons également citer d'autres protocoles : LWM2M, Fiware.

Appréhender la notion de sécurité avec la ressource ACP :

Dans un réseau IoT, la sécurité est primordiale et peut être gérer à plusieurs niveaux. Tout d'abord, au niveau du canal de communication qui relie le client à la plateforme OneM2M, où l'utilisation d'une version sécurisée du protocole http, HTTPS, est recommandée. Cette protection est cependant limitée car elle n'empêche pas un attaquant de se faire passer pour le client lorsqu'il possède les identifiants nécessaires. Pour résoudre ce problème, un système de clés partagées entre les deux parties peut être mis en place.
Un autre moyen de sécuriser la communication peut se faire au niveau de l'authentification. Dans le cas de OneM2M, le client s'authentifie grâce à un identifiant, nommé *originator* pour envoyer une requête. Selon les droits du client, la plateforme établi si le client a le droit ou non de réaliser la requête. Dans notre cas, avec le protocole http, l'*originator* est précisé dans le header d'envoi : X-M2M-Origin. Pour mettre en place ces droits, une gestion du contrôle d'accès aux ressources peut être programmée par une ressource. Il s'agit de l'Access Control Policy (ACP). Ses champs définissent : acr pour une règle donnée , acor pour l'identifiant du client, acop pour établir les opérations autorisées. Le champs pvs permet enfin de gérer les droits d'accès à l'ACP lui-même.

Comprendre la notion d'interopérabilité dans l'IoT :

Interopérabilité et IPE : Dans un réseau IoT, des objets et des systèmes d'objets sont amenés à communiquer avec d'autres systèmes. L'interopérabilité définit la capacité de ces deux systèmes à communiquer des données entre eux malgré les différences de programmation et de construction. Cette communication avec d'autre systèmes et l'intégration de ceux-ci dans plusieurs applications est un des objectifs de OneM2M : l'intégration horizontale.
Un déploiement IoT est constitué d'un ensemble d'objets qui communiquent avec un ensemble d'applications via un intermédiaire : une plateforme. La technologie du Web (Web des objets) est la communication essentiellement utilisée pour relier les applications et la plateforme. Des protocoles dédiés (Internet des objets) relie alors la plateforme et les objets. La plateforme se charge de faire la traduction entre les deux moyens de communication.
Dans le cas de OneM2M, l'intégration de nouvelles technologies se fait en divisant la plateforme en plusieurs modules, appelées plugins. C'est le plugin Interworking Proxy Entities (IPE) qui s'occupe de cette traduction d'une technologie (ZigBee, Hue etc..) au standard OneM2M. Son fonctionnement est décrit sur le schéma suivant :



***Enrichissement* :** Dans l'IoT, ce ne sont que des machines qui communiquent entre elles, il n'y a pas l'intervention d'un humain pour traduire ou remettre en contexte les données. Ainsi, les objets produisent des données qui ne peuvent être interprétées que par une application dans un contexte donné. Si l'on sort de ce contexte, ces données n'ont aucun sens logique. Les applications ont alors besoin du contexte, qui sont également des données en plus de la donnée primaire. Ces données « secondaires » ou métadonnées sont appelées des connaissances. Le problème est que ces données sont lourdes et inadaptées aux contraintes des réseaux de l'Iot à bande passante limitée et à vocation de dépense énergétique minimisée.

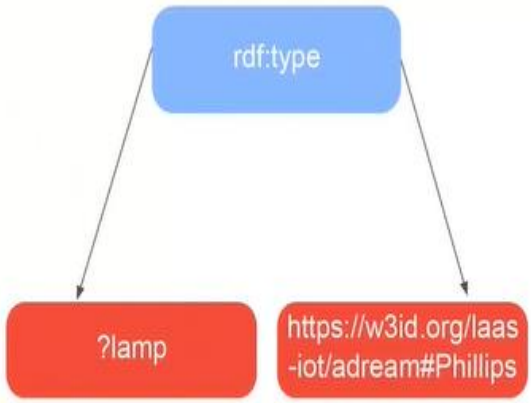
L'enrichissement des données entre l'objet et l'application est la solution à ce problème. Il est faisable par le biais de passerelles qui adaptent le type de données aux objets qui les manipulent et transforment un message dans un format plus expressif.

Ontologie : Deux machines doivent parler la même langue mais en plus de cela avoir le même vocabulaire pour décrire la connaissance partagée vu précédemment. L'ontologie définit alors un ensemble de concepts issus d'un domaine connu (ici le domaine des objets connectés), des relations entre ces concepts et d'axiomes logiques pour la description des connaissances. Pour que les données échangées entre deux objets deviennent interopérables, il faut les annoter avec du vocabulaire fourni par l'ontologie. On définit alors les données comme étant la partie assertionnelle et le vocabulaire comme étant la partie terminologique. L'ensemble des données et du vocabulaire constitue la base de connaissance.

Web sémantique : Pour représenter cet ensemble, on utilise un langage : le RDF (ressource description framework). Celui-ci définit un graphe de ressources et son modèle de donnée est construit sous forme d'un triplet : le sujet (entité de la déclaration), la propriété (la caractéristique qu'on donne au sujet) et l'objet (valeur de la propriété). Chaque élément du triplet est identifié par une URI.

Pour interroger les graphes RDF, on utilise le langage SPARQL qui permet d'exprimer des requêtes.

Exemple de la lampe :



?lamp = https://w3id.org/laas-iot/adream-appartment#PHL_LMP_01

Sujet : lampe , Propriété : type, Objet : Type en question. On retrouve alors en réponse à notre requête SPARQL : ?lamp, une solution possible qui est marquée ci-dessus.

Raisonneur : Pour finir sur ce cours, on va aborder le raisonneur. Son rôle est d'ajouter de nouvelles connaissances à la base de connaissances à partir de règles définies qui guident son raisonnement. Il existe des règles préétablies et d'autres qui sont ajoutées en fonction du besoin.