



جامعة محمد الخامس بالرباط  
Université Mohammed V de Rabat

# **L'internet des objets Matériel & Logiciel**

**Séance 3  
Master spécialisé IPS**

# Architecture générale de bout en bout

## Business

- Gérer l'ensemble des activités et services du système IoT
- Accompagnement des processus décisionnels

## Application

- Fournir les services demandés par les clients
- Outils de visualisation

## Middleware

- Donner accès aux services
- Traitement des données reçues et prise de décision

## Réseau

- Transfert des données collectées de la couche Dispositifs à la couche Middleware

## Dispositifs

- Objets physiques: capteurs et actionneurs
- Recueillir les événements détectés

# Quel est l'objectif de l'internet des objets ?

**L'objectif de l'internet des objets** est de poursuivre l'intégration du réseau Internet pour **permettre à autre chose que des ordinateurs d'échanger des données.**

Utiliser des protocoles développés pour des ordinateurs et maintenant des téléphones portables (plus puissants que les ordinateurs utilisés par l'internet à ses débuts) mais dans des environnements plus contraints.

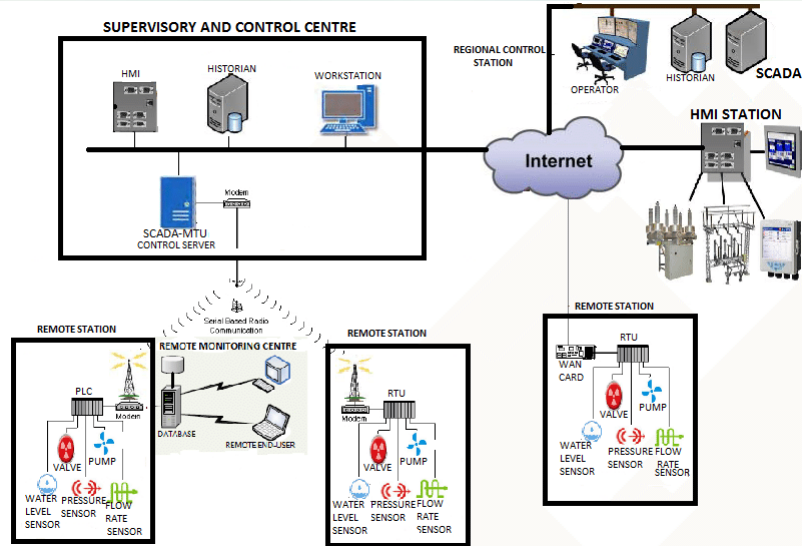


**Le but principal est d'optimiser les processus pour qu'ils soient plus efficaces pour économiser des ressources ou pour augmenter la productivité.**

L'IoT est une architecture globale permettant à des objets, d'interagir de manière autonome via internet. Cette interaction :

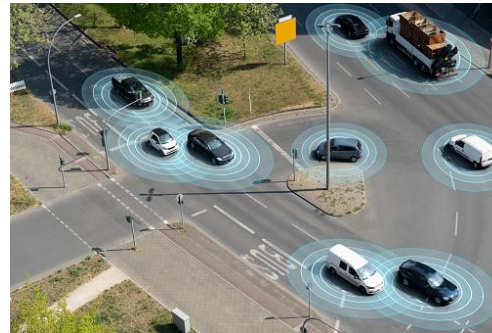
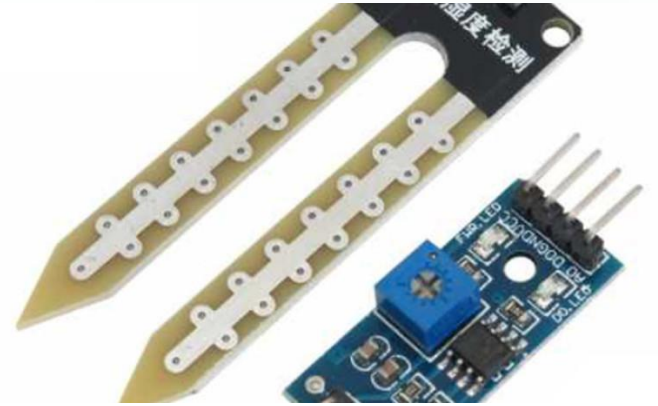
- est réalisée, par construction, au travers d'un réseau internet, ce qui implique généralement que les objets soient pourvus d'une adresse IP ;
- est relative à des commandes (opérations de contrôles ou appels de fonctions) ou des échanges de données ou d'informations.

# Quel est l'objectif de l'internet des objets ?




**Des capteurs dans une usine pour contrôler la production / maintenance préventive**

**La surveillance du degré d'humidité d'un champ pour réduire la consommation d'eau**



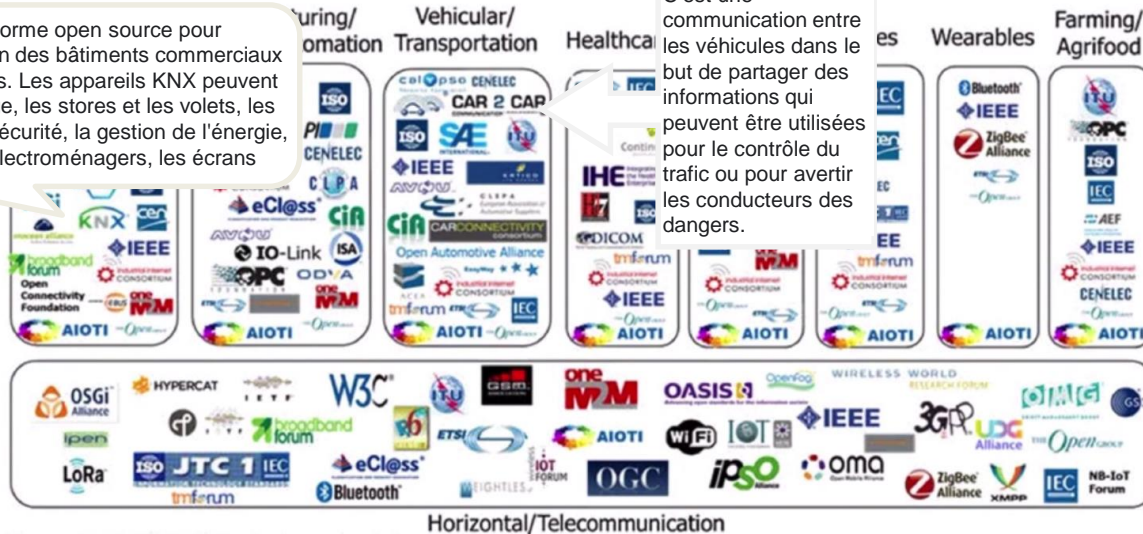
**Des voitures connectées qui vont dialoguer pour éviter les collisions**

# C'est quoi le problème?



KNX est une norme open source pour l'automatisation des bâtiments commerciaux et domestiques. Les appareils KNX peuvent gérer l'éclairage, les stores et les volets, les systèmes de sécurité, la gestion de l'énergie, les appareils électroménagers, les écrans

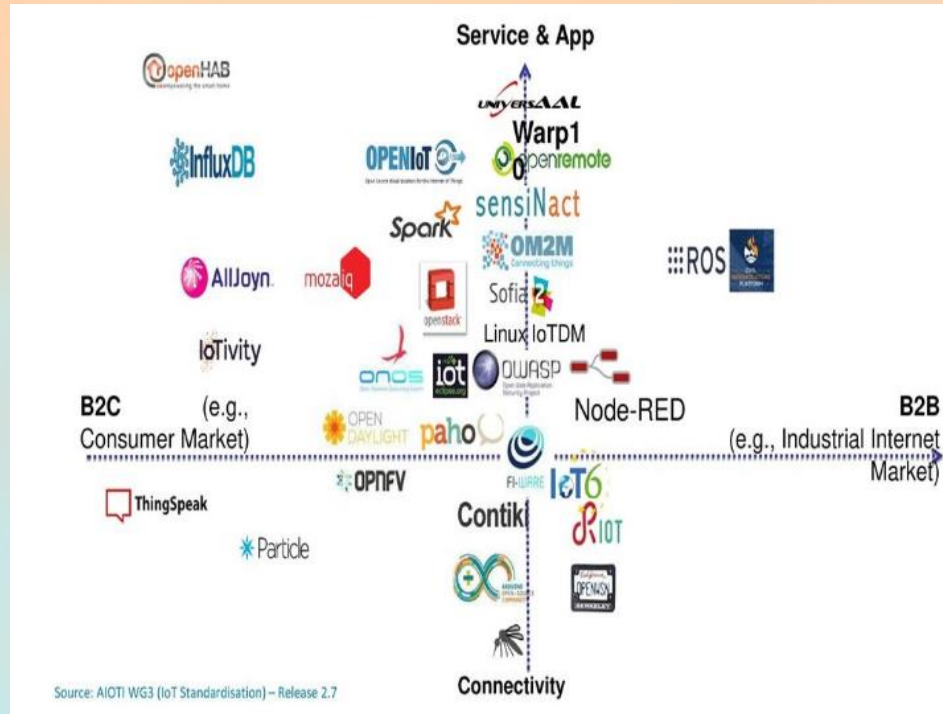
C'est une communication entre les véhicules dans le but de partager des informations qui peuvent être utilisées pour le contrôle du trafic ou pour avertir les conducteurs des dangers.



Source: AIOI WG3 (IoT Standardisation) – Release 2.6

L'internet des objets consiste à simplifier cette architecture, pouvoir généraliser la communication, acquérir l'interopérabilité « Capacité de systèmes, unités, matériels à opérer ensemble » pour pouvoir les commercialiser.

# L'interopérabilité



# L'interopérabilité

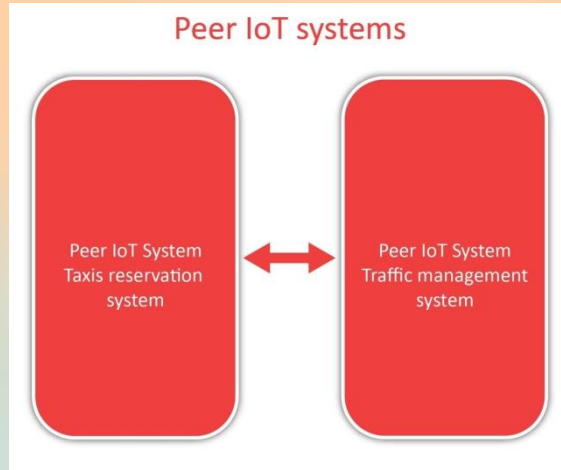


**Sigfox** est un opérateur de réseau mondial français qui construit des réseaux sans fil pour connecter des objets à faible consommation d'énergie tels que des compteurs d'électricité et des montres intelligentes, qui doivent être allumés en permanence et émettre de petites quantités de données.

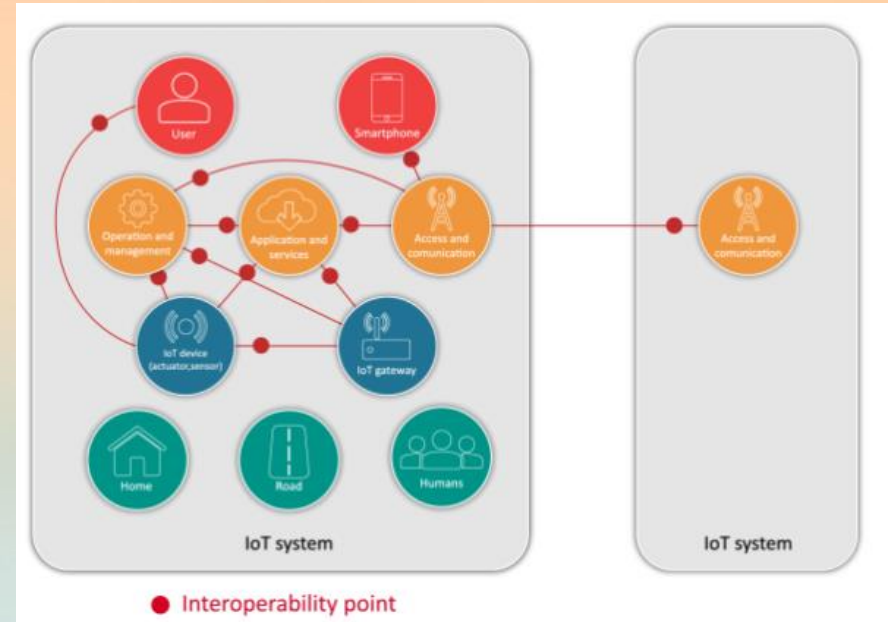
La plate-forme IoT Open est indépendante de la technologie, ce qui signifie qu'elle peut prendre en charge divers appareils IoT, quel que soit le fabricant ou le protocole de communication.



# Qu'est ce qu'est l'interopérabilité de l'IoT ?



Un système de réservation de taxi peut interagir avec un système de gestion de trafic.



**L'union internationale des télécommunications** : l'IoT est une infrastructure globale pour la société de l'information permettant la mise en place de services avancés grâce à une interconnexion (physique ou virtuelle) d'objets basés sur des technologies interopérables de communication et d'information.

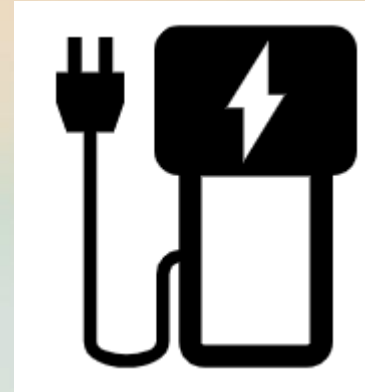


# L'évolution de l'IoT

En plus des technologies spécifiques, les protocoles de l'internet sont intégrés, mais en les adaptant aux contextes du secteur.



Nous sommes actuellement en train de vivre la convergence vers un ensemble réduit de protocoles, une standardisation de la représentation de la donnée, et son traitement sur des plateformes plus génériques.



# Protocoles IoT- partie 1

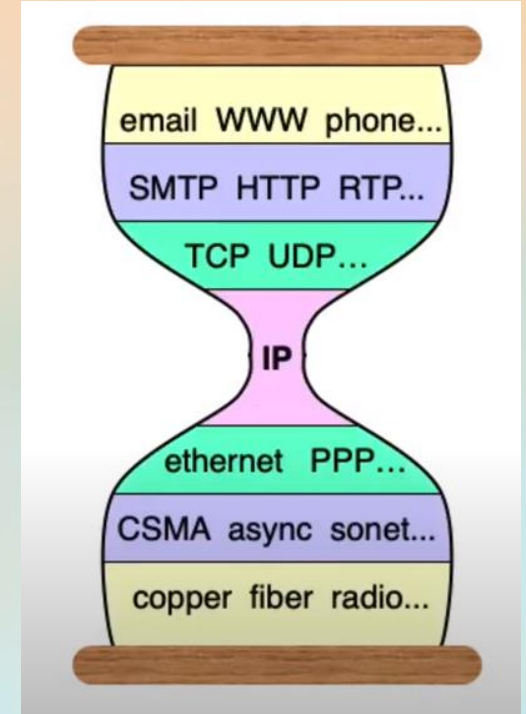
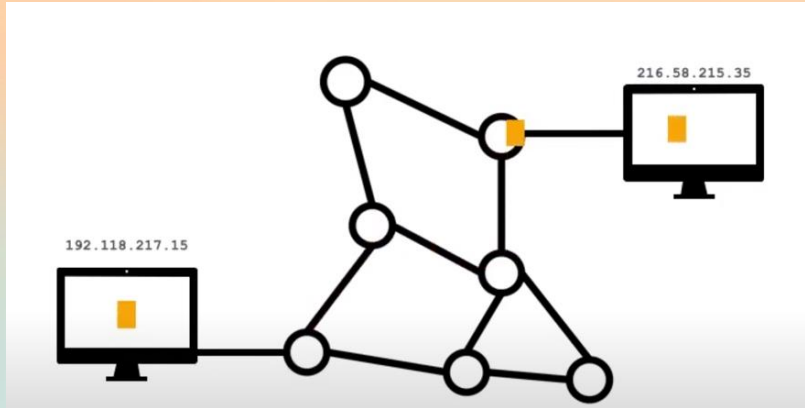
Application Protocol		DDS	CoAP	AMQP	MQTT	MQTT-NS	XMPP	HTTP REST
Service Discovery		mDNS				DNS-SD		
Infrastructure Protocols	Routing Protocol	RPL						
	Network Layer	6LoWPAN				IPv4/IPv6		
	Link Layer	IEEE 802.15.4						
	Physical/ Device Layer	LTE-A	EPCglobal		IEEE 802.15.4		Z-Wave	
Influential Protocols		IEEE 1888.3, IPSec				IEEE 1905.1		

# Rappel - Encapsulation

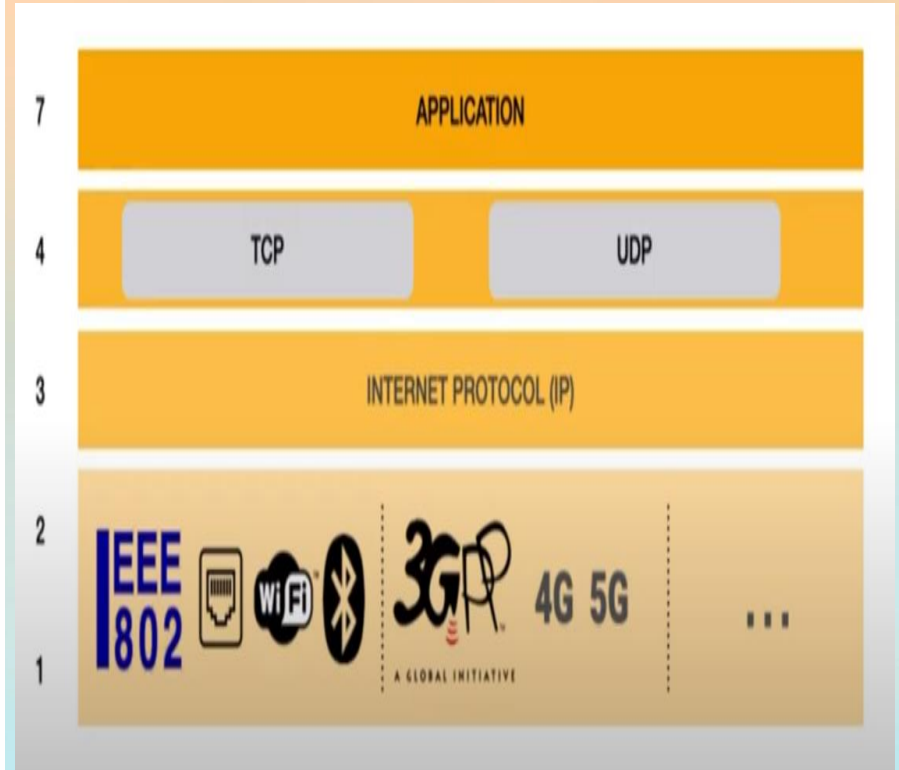
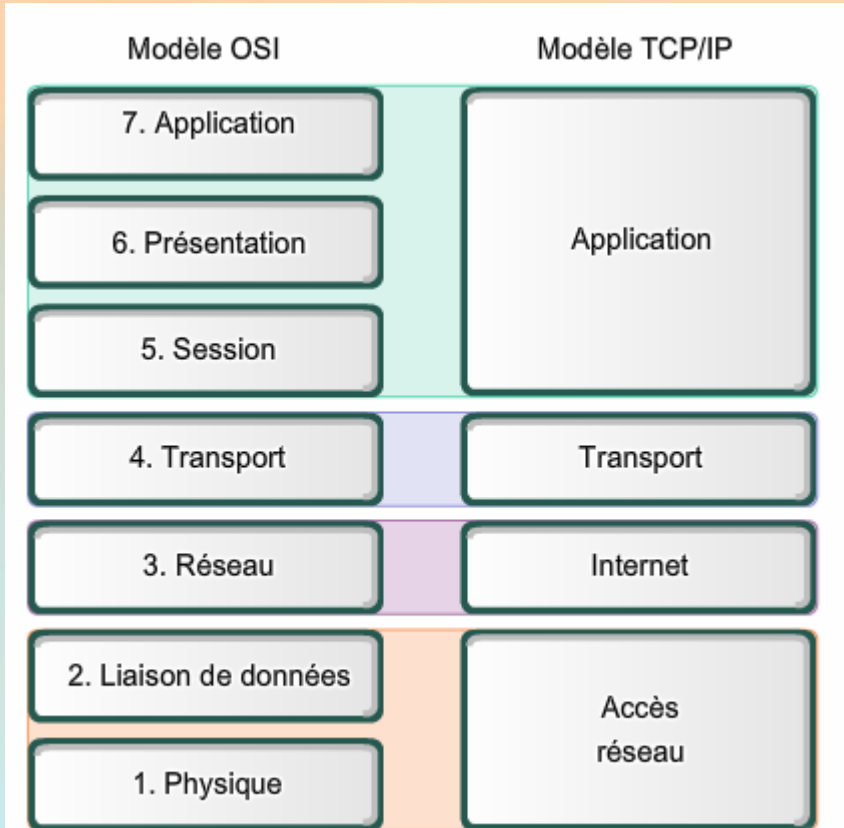
## Flux de données dans la pile de protocoles



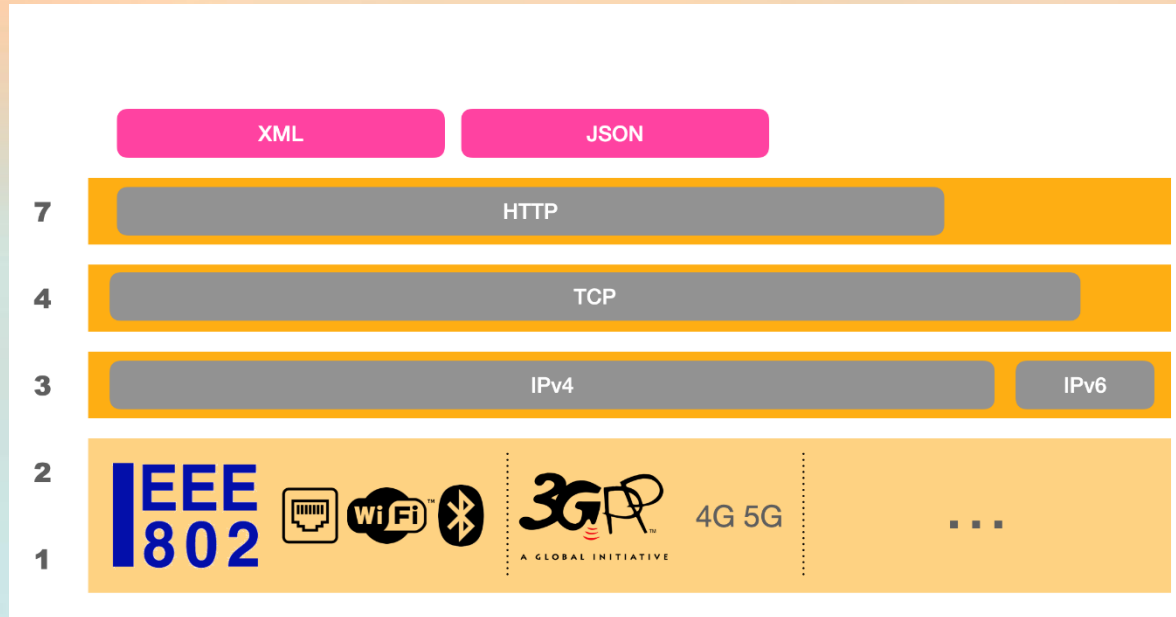
# Rappel – Service des couches



# Rappel –Modèle architecturale de l'Internet

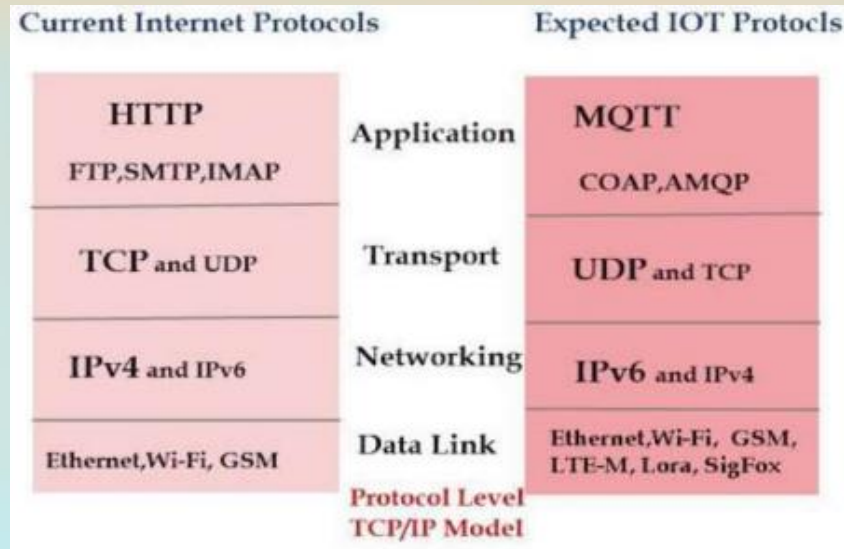


# Rappel – Protocoles de l'Internet



# IoT vs Internet Protocoles

L'IoT utilise des protocoles Internet existants et introduit d'autres qui sont nouveaux.





# Protocoles de la couche application

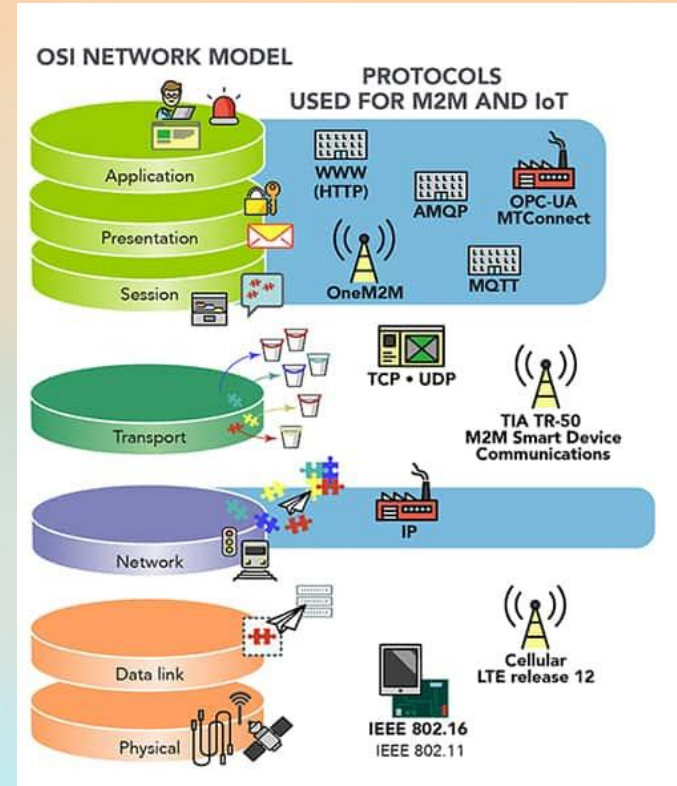
- Une application IoT permet aux objets connectés d'envoyer leurs données à un serveur Web Internet ou une plateforme Cloud.
- Les protocoles de la couche application permettent de transmettre des commandes depuis les applications utilisateurs aux actionneurs des objets connectés.
- L'infrastructure Web classique n'est pas adaptée à la majorité des applications IoT qui sont dotées d'équipements de faibles ressources : Petits microcontrôleurs, petites quantités de mémoire RAM, énergie limitée, etc.

# Protocoles de la couche application

Les protocoles applicatifs qui utilisent un nombre limité de messages de petites tailles sont utilisés pour les applications

IoT, et sont classés en 3 familles:

- Protocole de transfert web: Web REST, COAP
- Protocole de messagerie: MQTT, XMPP et AMQ.
- Protocole réseau: Websocket



# Protocole de transfert web : Services Web REST

Services web de type *Representational state transfer (REST)*

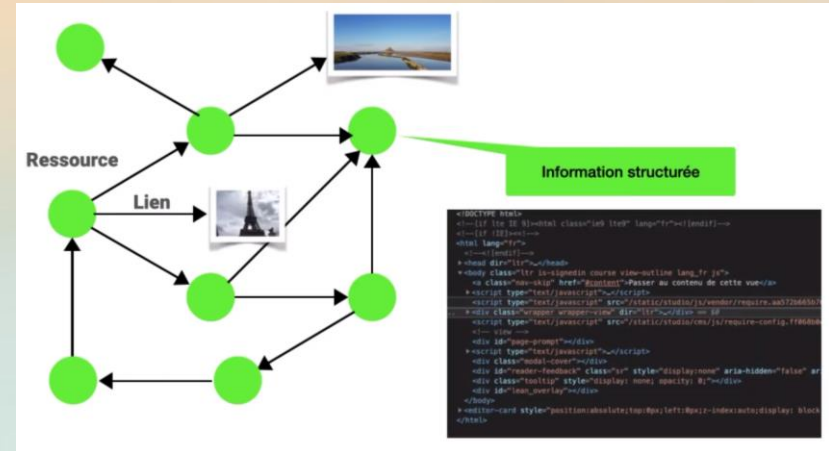
En principe, l'architecture REST ne dépend pas d'un protocole en particulier. Le concept central repose obligatoirement sur les propriétés suivantes selon Fielding (Roy Fielding en 2000) :

- ❖ **Adressabilité** : chaque ressource, par exemple, une commande, un produit ou un article doit pouvoir être identifié par un Unique Resource Identifier (URI).
- ❖ **Interface homogène** : chaque ressource doit pouvoir être facilement utilisée et de manière homogène, grâce à des méthodes standard. Par exemple avec les méthodes « GET », « POST » ou « PUT » en HTTP.
- ❖ **Structure client-serveur** : généralement, le principe de la structure client-serveur est appliqué lorsque le serveur met un service à disposition qui peut être demandée par un client.
- ❖ **Serveur sans état**
- ❖ **Différentes représentations des ressources** : chaque ressource peut afficher différentes représentations et différents langages ou formats comme HTML, JSON ou XML devront être utilisés par exemple.
- ❖ **Hypermédia** : la mise à disposition des ressources a lieu via Hypermedia, par exemple sous forme d'attributs « href » et « src » dans des documents HTML.

# Protocole de transfert web : Services Web REST

## Services web de type *Representational state transfer (REST)*

- Le Web et ses extensions sont basés sur un modèle client-serveur. Les serveurs possèdent des ressources et les clients peuvent y accéder ou les modifier grâce à un protocole tel que HTTP.
- Les capteurs, les actionneurs et les systèmes de commande en général peuvent être représentés comme des ressources.



# Services Web REST

- Chaque ressource est définie par un unique URI (Uniform Resource Identifier).



# Services Web REST

Un URI commence par un [schéma](#) indiquant l'autorité de nommage, suivi d'une valeur d'autorité puis d'un chemin dans l'espace d'autorité. Des caractères comme les ":" ou les "/" sont utilisés pour améliorer la lisibilité de l'URI.

Par exemple :

- **mailto:**mduerst@ifi.unizh.ch
- **ssh:**//utilisateur@example.com
- **ftp:**//ftp.is.co.za/rfc/rfc1808.txt

# Services Web REST

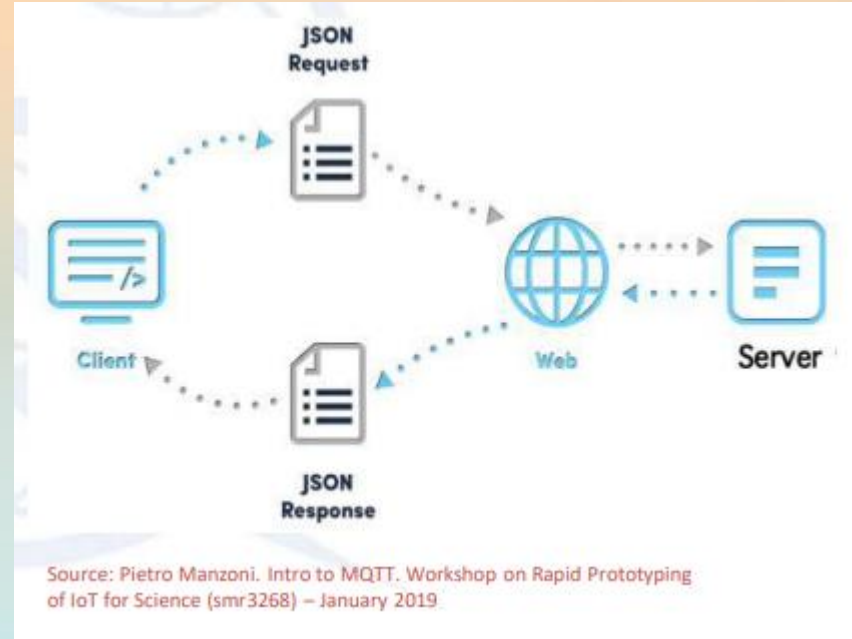
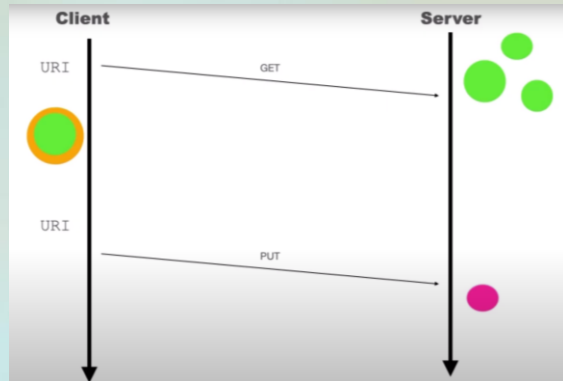
Qu'est ce qui est unique dans le monde?

- Un prénom
- Un nom de famille
- Un numéro de passeport
- Un numéro de téléphone portable avec son préfixe international
- Un numéro complet de compte en banque (IBAN)
- L'adresse IP de ma machine dans un réseau privé
- L'adresse IP d'un serveur google
- Le nom de domaine [www.um5.ac.ma](http://www.um5.ac.ma)
- le nom d'une ville



# Services Web REST

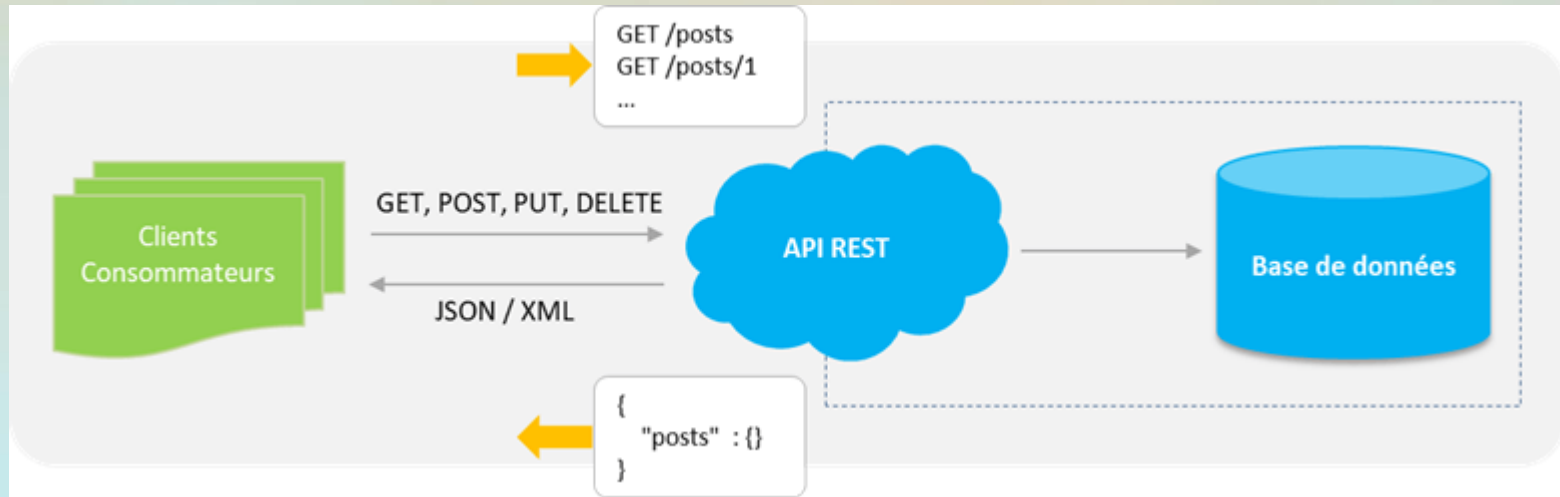
- REST utilise plusieurs formats pour représenter les ressources : Text, JSON, XML.
- JSON est le format le plus utilisé.



# Services Web REST

- Un serveur doit être sans état, ce qui signifie qu'il ne conserve pas d'information après avoir répondu à une demande d'un client. Cela permet de simplifier le traitement dans le serveur qui doit traiter les requêtes d'un grand nombre de clients.

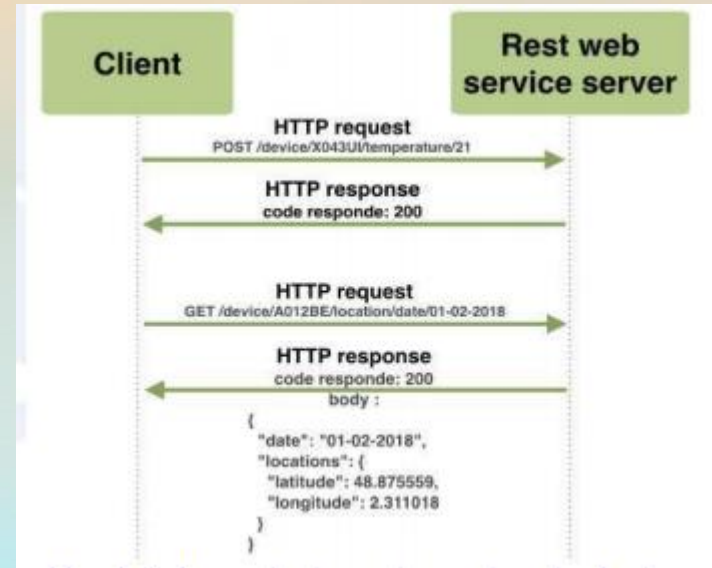
## L'architecture REST



# Services Web REST – Scénario IoT

URI	Méthode	Signification
/device/:device/temperature/:temperature	POST	Effectuer un POST en spécifiant, pour l'objet :device, une nouvelle valeur de température :temperature en °C
/device/:device/location/date/:date	GET	Effectuer un GET pour obtenir la position GPS d'un objet :device à une date donnée :date

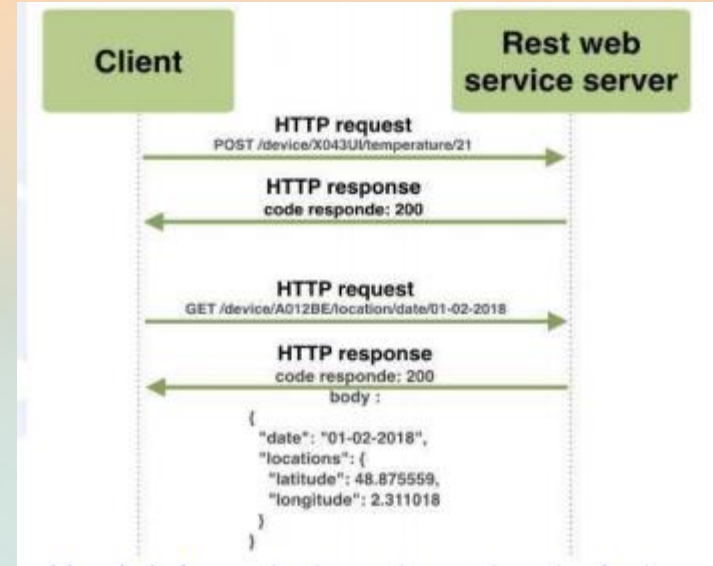
- Le client envoie une requête POST pour indiquer au serveur une nouvelle température de 21°C, pour l'objet X043UI.
- Le serveur lui répond avec un code de 200 pour indiquer que tout est OK.
- Le client envoie une requête GET pour demander la localisation de l'objet A012BE à la date du 01-02-2018.
- Le serveur répond en envoyant les coordonnées.



# Services Web REST – Scénario IoT

Le serveur ajoute également un code réponse HTTP, à trois chiffres, afin d'indiquer l'état de la réponse dont la forme est comme suit :

- 2xx indique le succès du traitement de la requête du client (exemple : 200 pour OK)
- 3xx redirige le client vers un autre lien
- 4xx indique une faute dans la requête du client (exemple : 404 pour Not Found)
- 5xx indique une erreur de la part du serveur (exemple : 500 pour Internal Server Error)

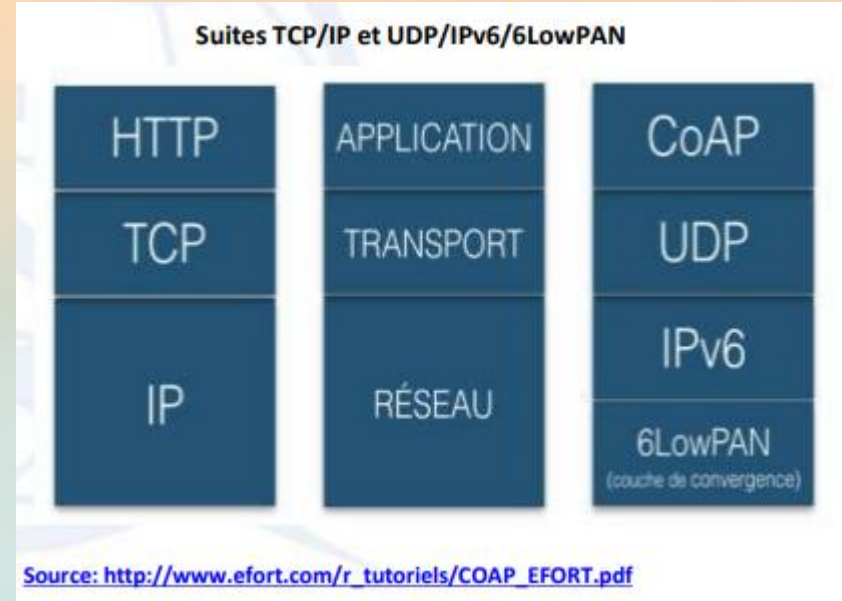


# Protocole de transfert web :

## CoAP ( Constrained Application Protocol )

CoAP (Constrained Application Protocol) est un protocole web basé sur une architecture client/serveur.

- CoAP est une version légère de REST conçu pour des communications UDP.
- CoAP est destiné à l'utilisation sur des appareils électroniques à faible consommation d'énergie.
- IETF CoAP utilise les URI pour identifier les ressources.
- HTTP est basé sur la suite TCP/IP alors que CoAP se base sur UDP/IPv6/6LoWPAN.



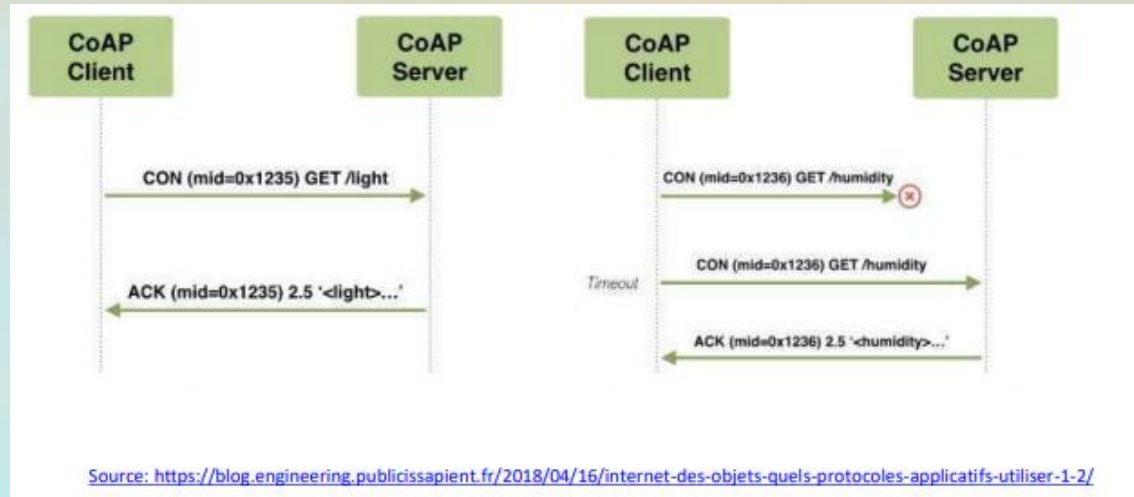
# CoAP ( Constrained Application Protocol )

Les requêtes CoAP sont équivalentes à celles de HTTP : un client envoie une requête à un serveur pour demander un service d'une ressource, identifiée par URI.

- CoAP utilise les méthodes HTTP {GET, PUT, POST, DELETE}.
- Les messages CoAP ont une taille (4 octets) allégée par rapport à celle des messages HTTP (variable).
- CoAP utilise quatre types de messages :
  - Confirmable (CON) : Message envoyé avec une demande d'accusé de réception.
  - Non-Confirmable (NON) : Message envoyé sans demande d'accusé de réception.
  - Acknowledgment (ACK) : Accusé de réception du message de type CON.
  - Reset (RST) : Accusé de réception d'un message qui n'est pas exploitable.

# CoAP ( Constrained Application Protocol )

Si la requête est du type CON alors le serveur retourne une réponse dans laquelle se trouve ; le type du message (ACK), le même mid que celui de la requête et un code réponse (2.xx, 4.xx ou 5.xx) et une représentation de la ressource.

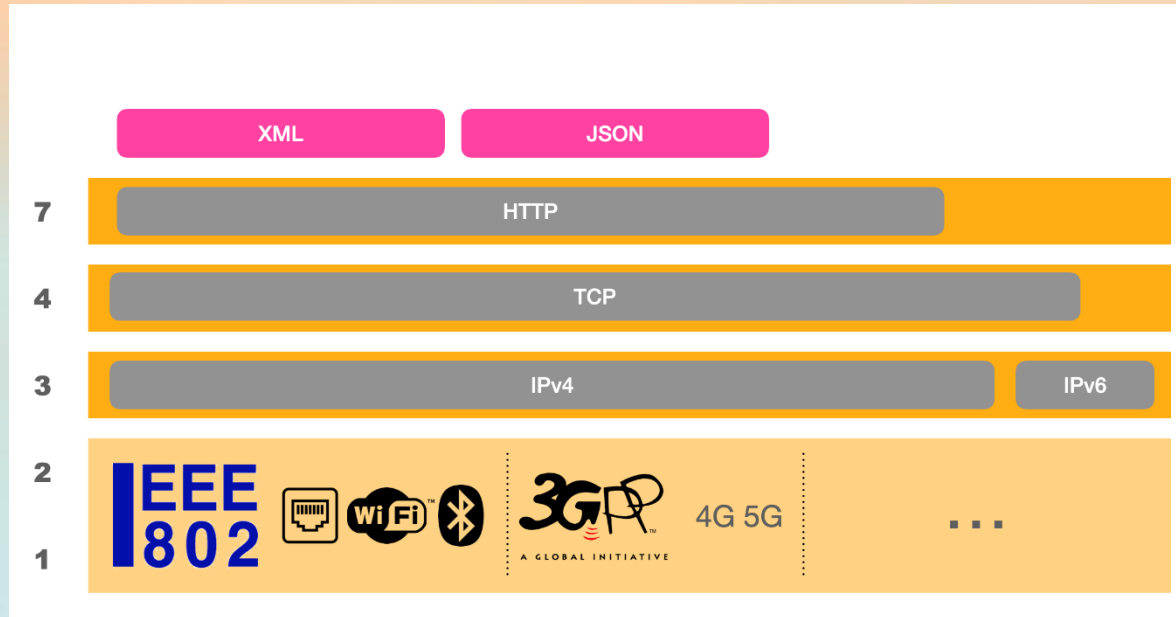




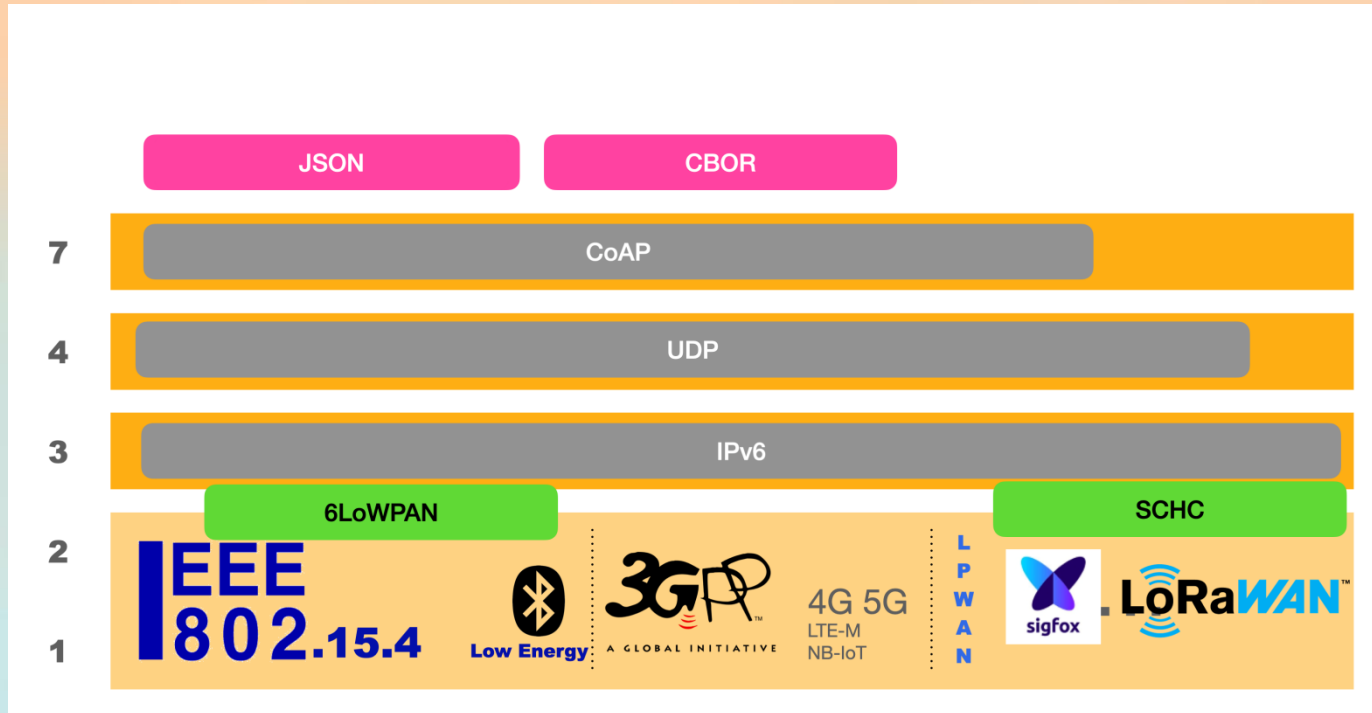
# CoAP ( Constrained Application Protocol )

- La signification du code réponse est la suivante :
  - 2.xx signifie que la requête a été correctement reçue et traitée
  - 4.xx signifie qu'une erreur a été rencontrée par le client
  - 5.xx signifie que le serveur n'est pas capable de traiter la requête

# Modification de l'architecture de l'Internet

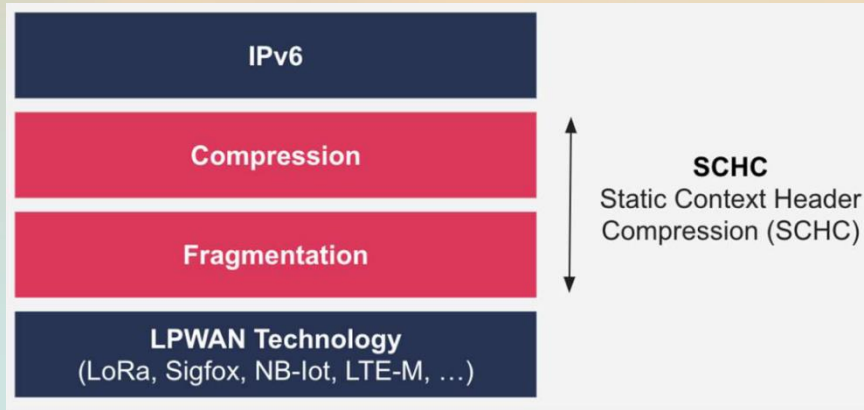


# Architecture de l'IoT

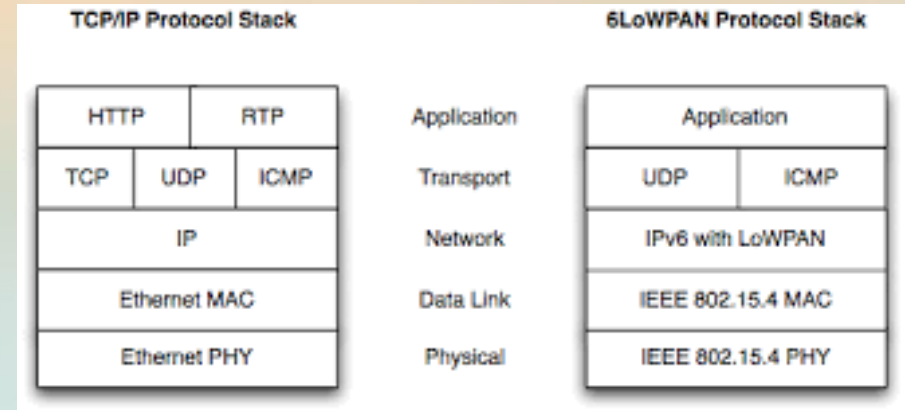


# Architecture de l'IoT

Le protocole d'application CoAP étend les services Web HTTP aux environnements contraints. Il offre notamment un ensemble de services adaptés à l'IoT dont des paramètres de communication asynchrones et asymétriques, la découverte automatique de dispositifs connectés et la multidiffusion. Il s'appuie sur le protocole UDP/IPv6 pour la communication.



La norme IETF "**SCHC** for CoAP" concilie  
interopérabilité, sécurité et efficacité  
énergétique  
RFC 8724



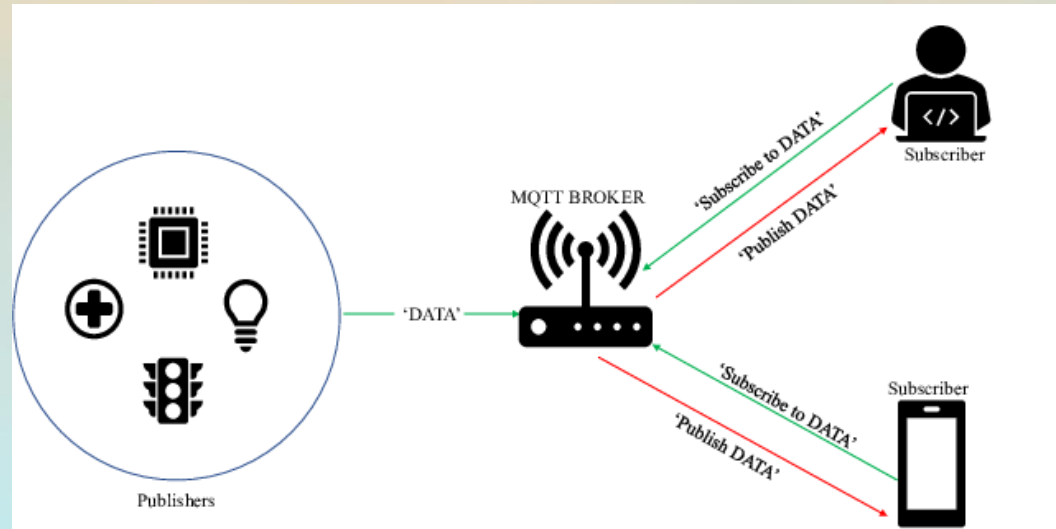
**6LoWPAN** : IPv6 Low power Wireless Personal Area N  
Défini les mécanismes d'encapsulation et de  
compression d'entêtes permettant aux paquets IPv6  
d'être envoyés ou reçus via le protocole IEEE 802.15.4

# Protocole de messagerie : MQTT

MQTT est un protocole de messagerie basé sur la publish/subscribe, qui peut être utilisé par-dessus le protocole TCP / IP.

Le modèle de messagerie publish-subscribe nécessite un broker (courtier) de messages.

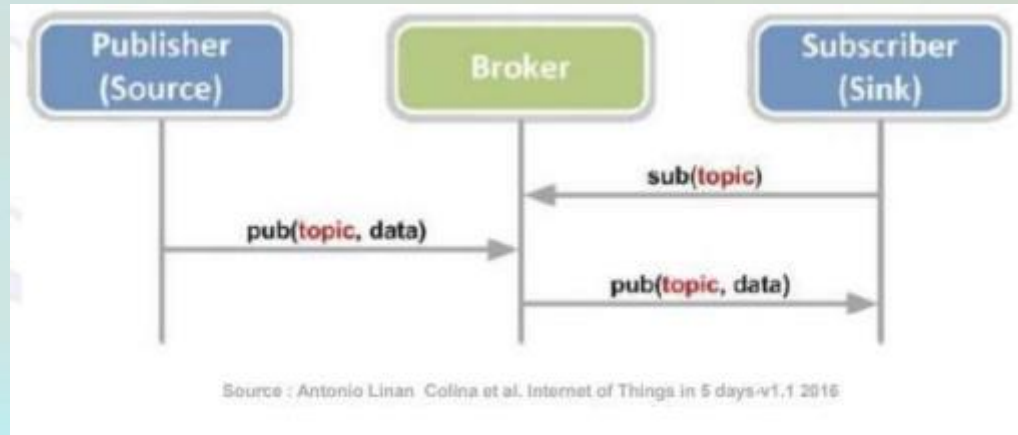
➤ Le broker est responsable de la distribution des messages aux clients intéressés en fonction du sujet d'un message.



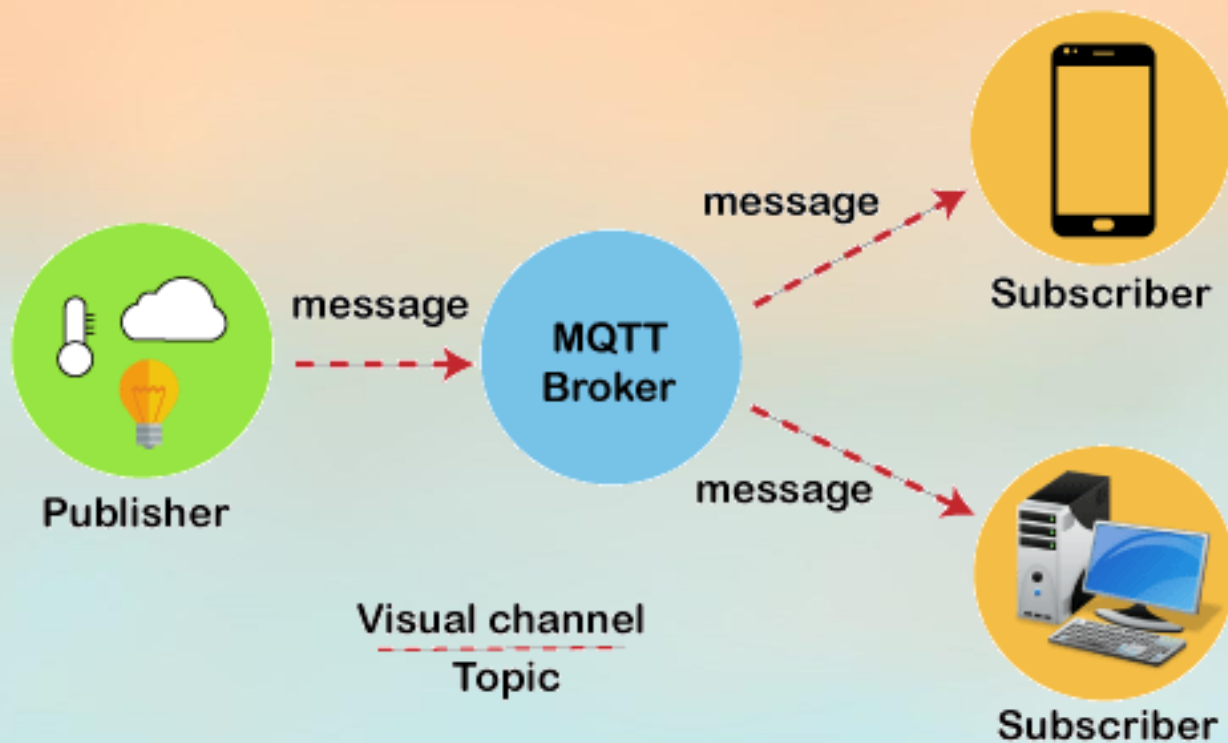
# MQTT

L'approche publish/subscribe classifie les messages par catégories (topics) auxquelles les destinataires s'abonnent (subscribe).

- Le client qui envoie un message (topic) est nommé publisher, celui qui reçoit le message est nommé subscriber.
- Un élément du réseau appelé broker, connu par le publisher et le subscriber, filtre les messages reçus et les distribue.



# MQTT Architecture





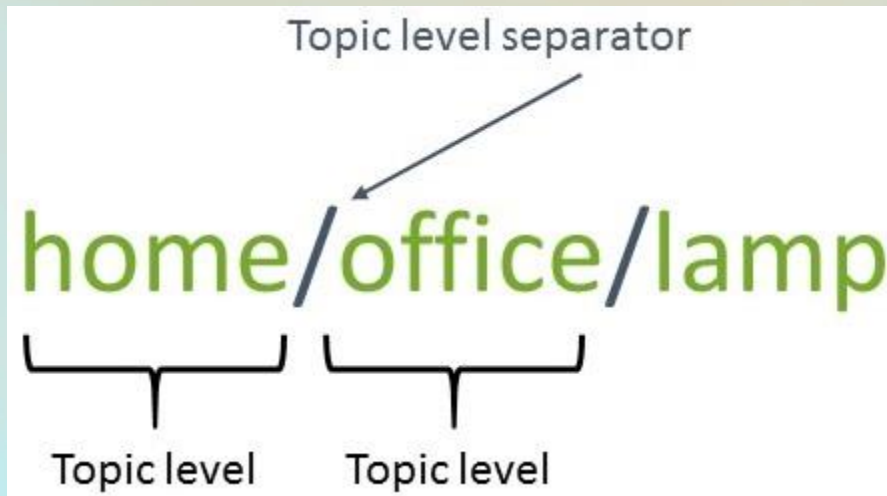
# MQTT

MQTT est caractérisé par :

- faible consommation d'énergie
- Entêtes compressées MQTT topics sont structurées d'une façon hiérarchique.
- Les topics sont sensibles à la casse, codées en UTF-8 et doivent comporter au moins un caractère.

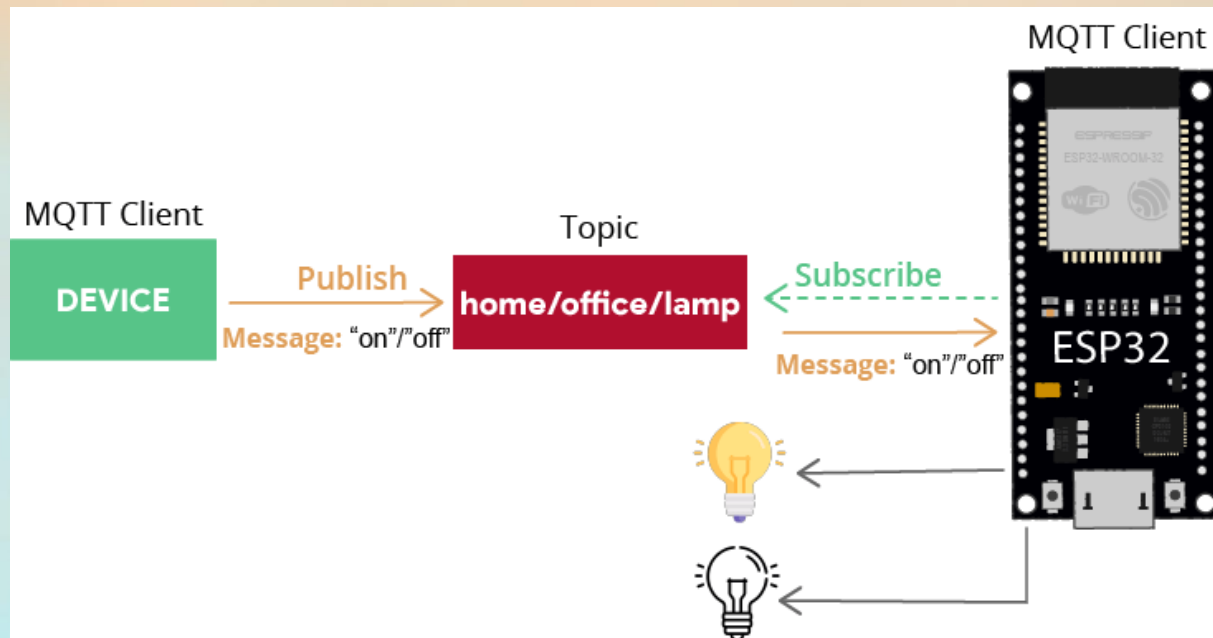
# MQTT

Les topics sont représentées par des chaînes séparées par une barre oblique. Chaque barre oblique indique un niveau de sujet. Voici un exemple de la façon dont vous créeriez un sujet pour une lampe (lamp) dans votre bureau à domicile (home office) :



# MQTT

Si vous souhaitez allumer une lampe dans votre bureau à domicile à l'aide de MQTT, vous pouvez imaginer le scénario suivant :



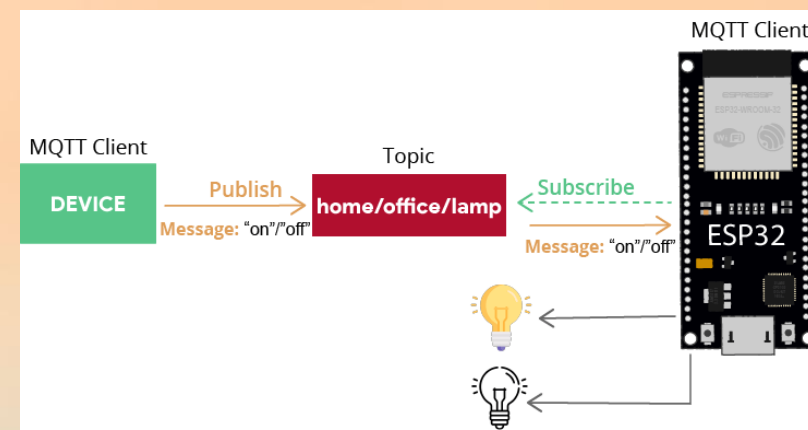
# MQTT : scénario

1- Un appareil publie des messages "marche" et "arrêt" sur le sujet maison/bureau/lampe.

2- Vous avez un appareil qui contrôle une lampe (il peut s'agir d'un ESP32, d'un ESP8266 ou de toute autre carte ou appareil). L'ESP32 qui contrôle votre lampe est abonné à ce même sujet : maison/bureau/lampe.

3- Ainsi, lorsqu'un nouveau message est publié sur ce sujet, l'ESP32 reçoit les messages "on" ou "off" et allume ou éteint la lampe.

- L'appareil qui publie les messages peut être un ESP32, un ESP8266 ou une plate-forme de contrôleur domotique avec prise en charge MQTT comme Node-RED, Home Assistant ou OpenHAB, par exemple.



# MQTT

Les caractéristiques du protocole MQTT en font un protocole adapté aux réseaux IoT car il répond aux besoins suivants :

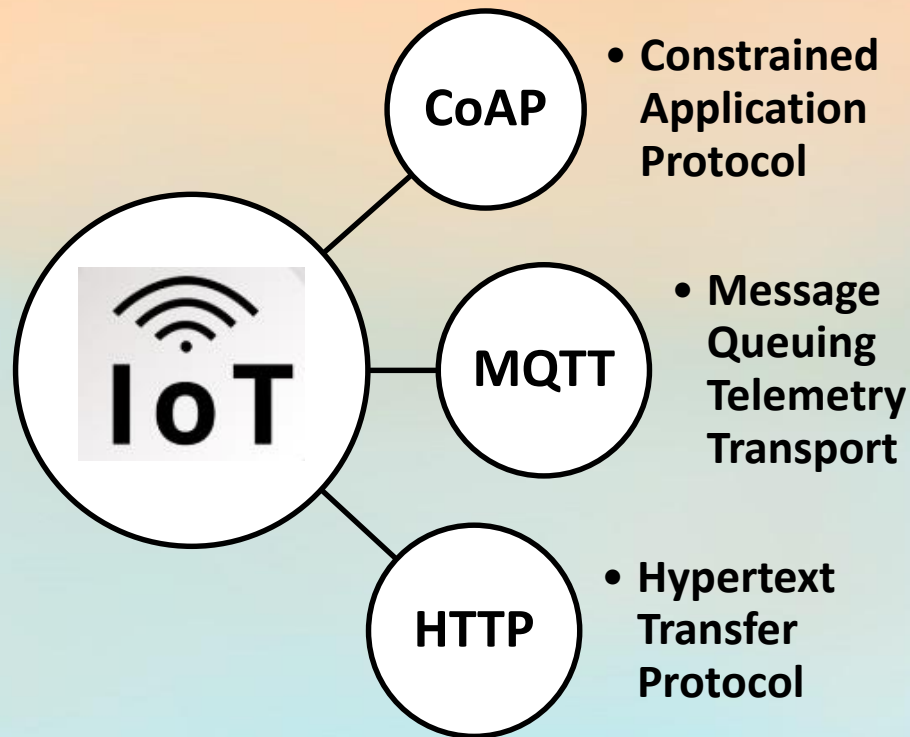
- Adapté aux réseaux à faible bande passante
- Idéal pour l'utilisation sur les réseaux sans fils grâce notamment à un nombre limité de messages de petite taille
- Faible consommation en énergie car la publication et la consommation des messages est rapide
- Nécessite peu de ressources de calculs et de mémoires
- Transmet un message à plusieurs entités en une seule connexion TCP

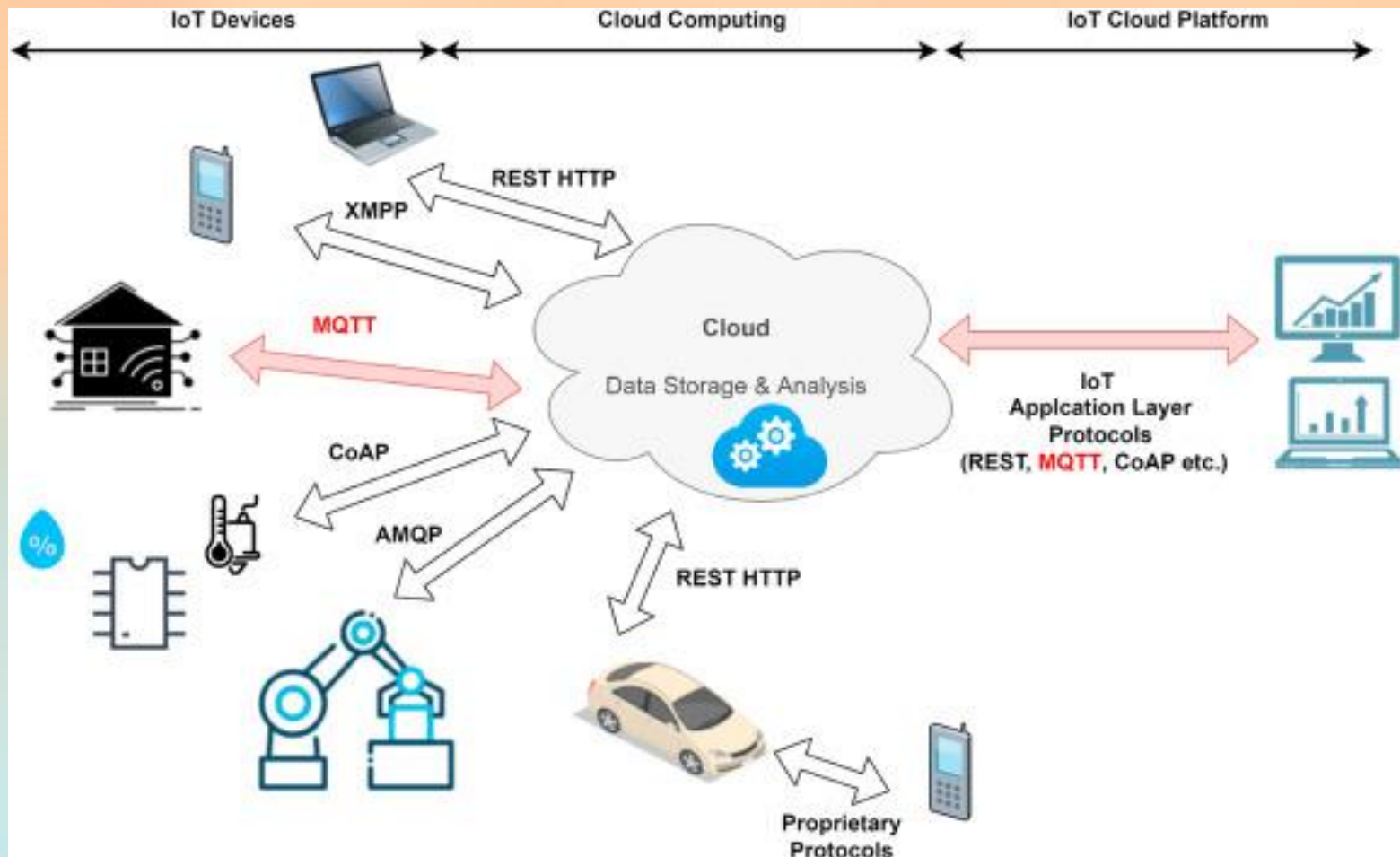
# Client/Serveur versus *Publish/Subscribe*

Les principaux avantages du paradigme *publish-subscribe* , sont les suivants :

- Faible couplage entre émetteur et récepteur, le *broker* sert d'intermédiaire et stocke les informations ;
  - Passage à l'échelle Les données provenant d'une source ne sont émises qu'une fois par la source. Le *broker* les recopie vers tous les abonnés.
- ***Dans un mode client/serveur, les données doivent être émises par le serveur autant de fois que les clients le demandent.***

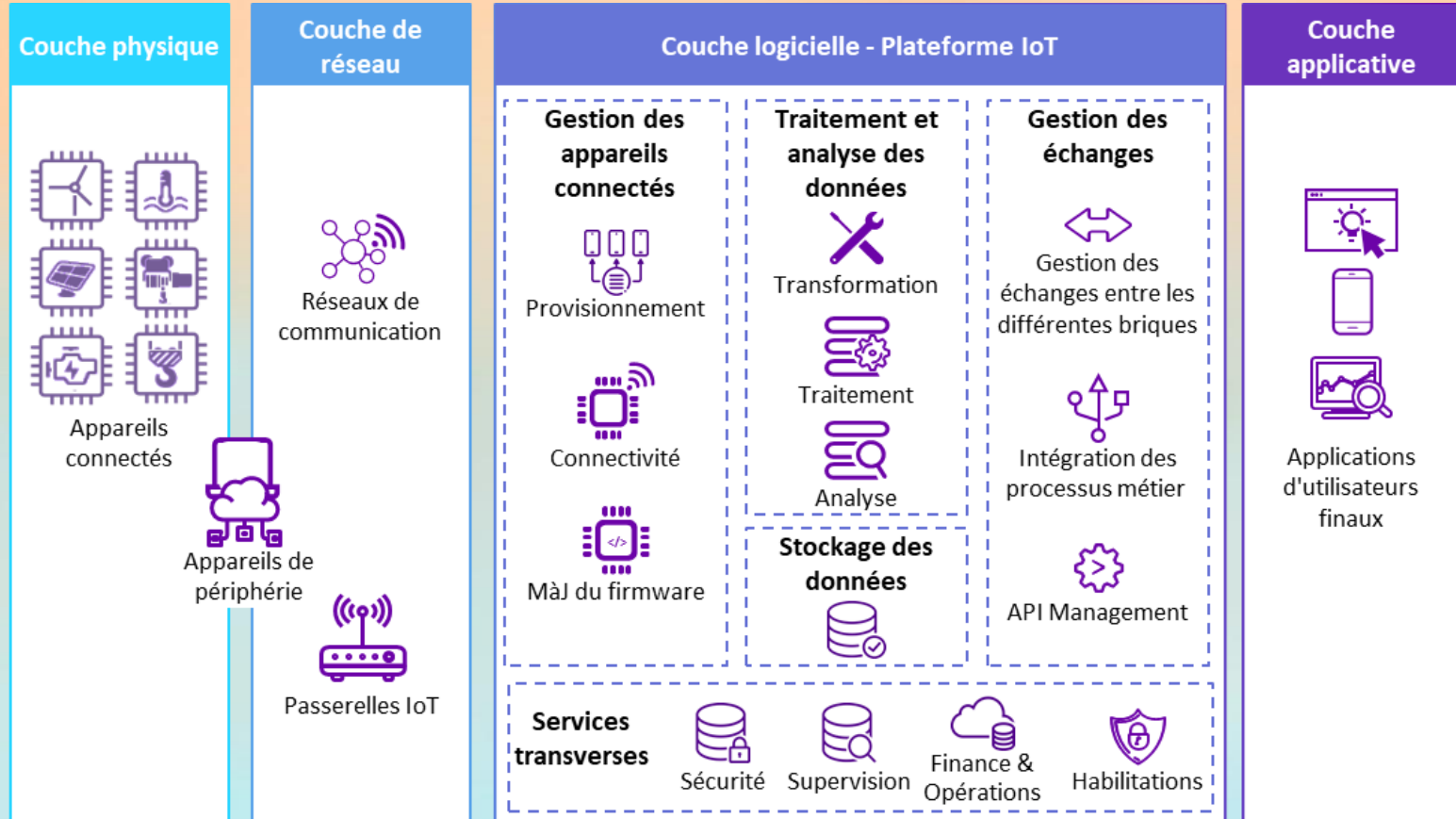
# Les protocoles IOT







# Architecture d'une solution IoT



Fin de la séance