Les protocoles de communication pour Exploiter les données

Partie

1

Les protocoles

Qui et où sont-ils ?

Les protocoles

POURQUOI ?

- **visualiser,**
- **gérer le stockage,**
- **intégrer des données dans une application**

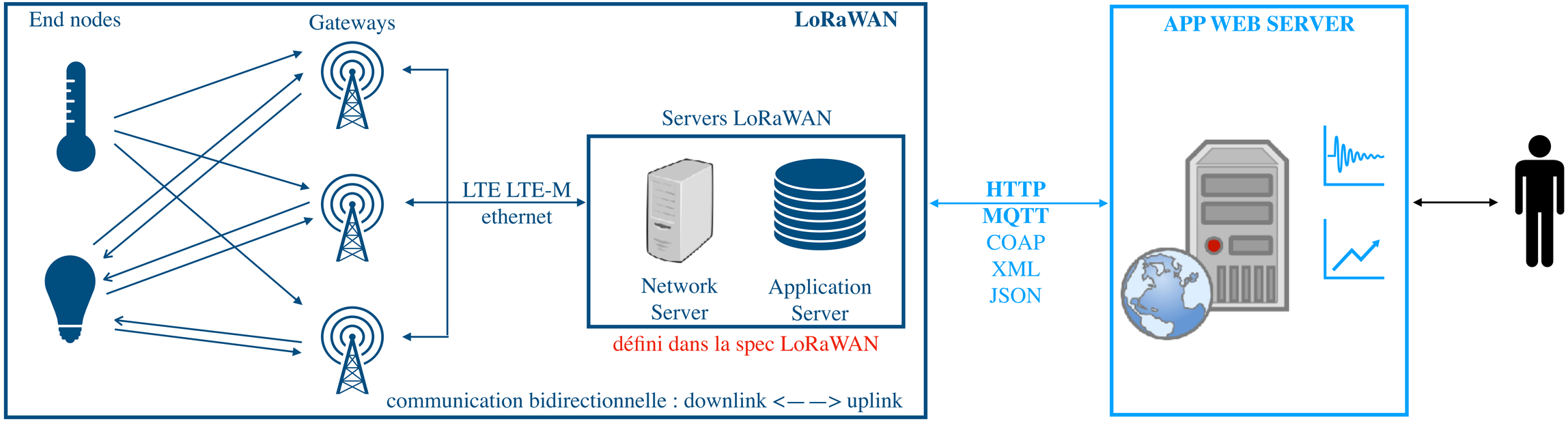
les + répandus

INTEGRATIONS

MQTT.ORG

HTTP

Les protocoles entre les serveurs LoRaWAN et d'APP WEB SERVER



Partie **LoRaWAN**

échange de données entre End Nodes et les serveurs LoRaWAN
en passant par les Gateway

Ces données ne sont pas disponible directement pour l'utilisateur, il faut :

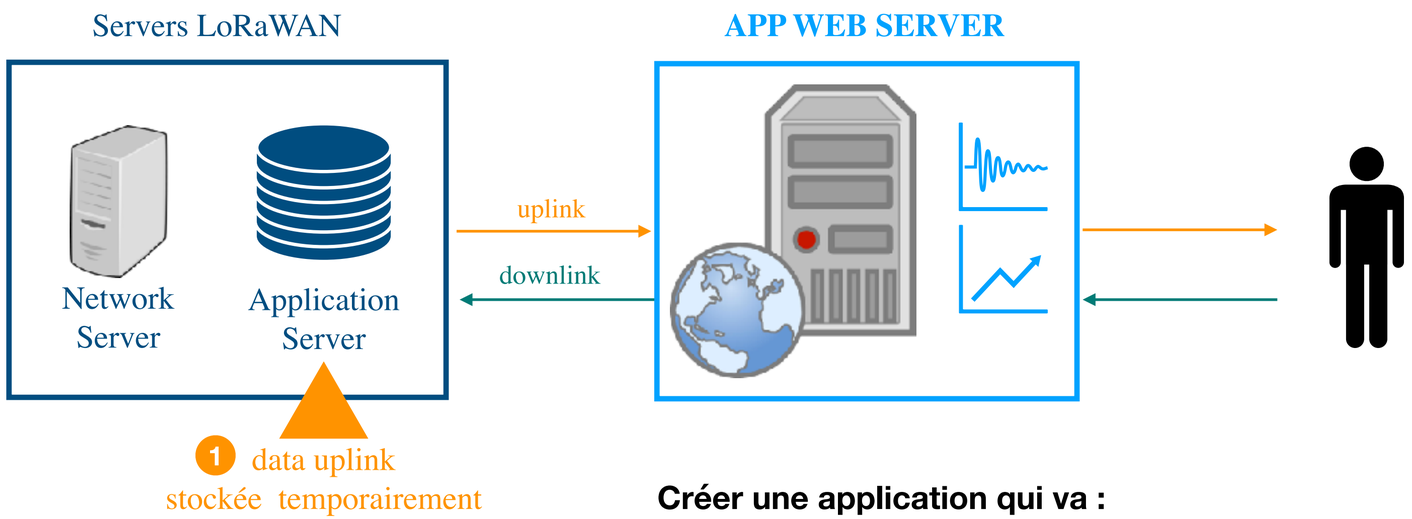
- les formater convenablement,
- les stocker dans des BD,
- les mettre à disposition via un service WEB
- …

Souvent **Application Server LoRaWAN** permet de gérer des Applications et des Services

Partie **APP**lication **WEB SERVER**

indépendante des protocoles LoRa + LoRaWAN
(peut être transposée aux autres protocoles liés aux IoT (sigfox, …)

Les protocoles entre les serveurs LoRaWAN et d'APP WEB SERVER



Créer une application qui va :

principe d'un système de monitoring 1 2 3 4 5

services uplink

- **récupérer les données** 2
- traiter et stocker en BD 3
- visualiser les données (graphique ou tableur) 4
- envoyer à l'extérieur (requête) 5

services downlink

mettre à dispo une interface IHM pour **transmettre les données** 1

traiter et formater la donnée 2

transmettre l'information via le protocole choisi (HTTP et MQTT) 3

Comment mettre en place tous ces services ? 🤔

Les protocoles de communication pour Exploiter les données

Partie

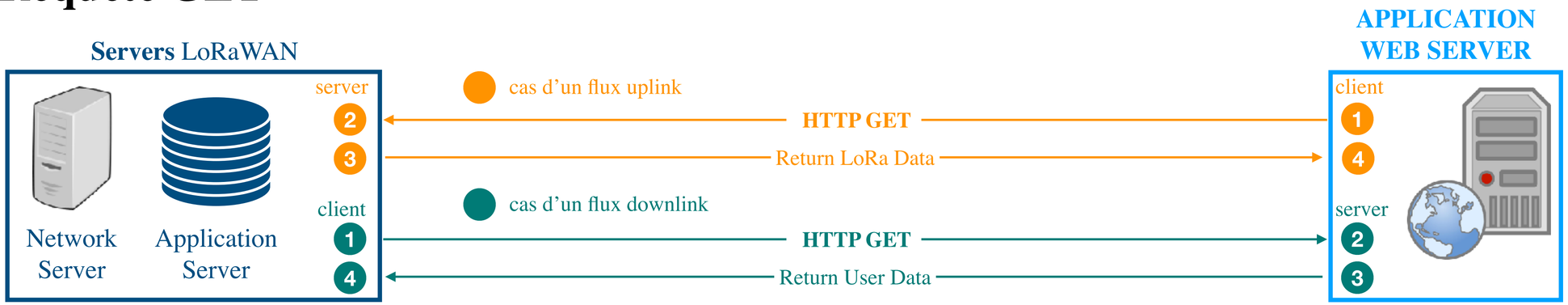
2 **Le protocole HTTP**
Exploitation

Le protocole

HTTP

Le protocole HTTP entre les serveurs LoRaWAN et d'APP WEB SERVER

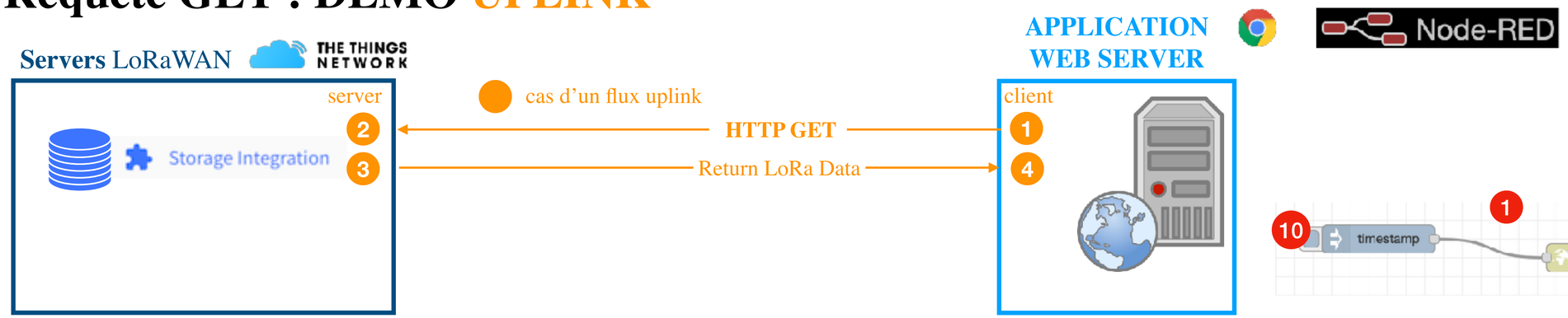
Requête GET

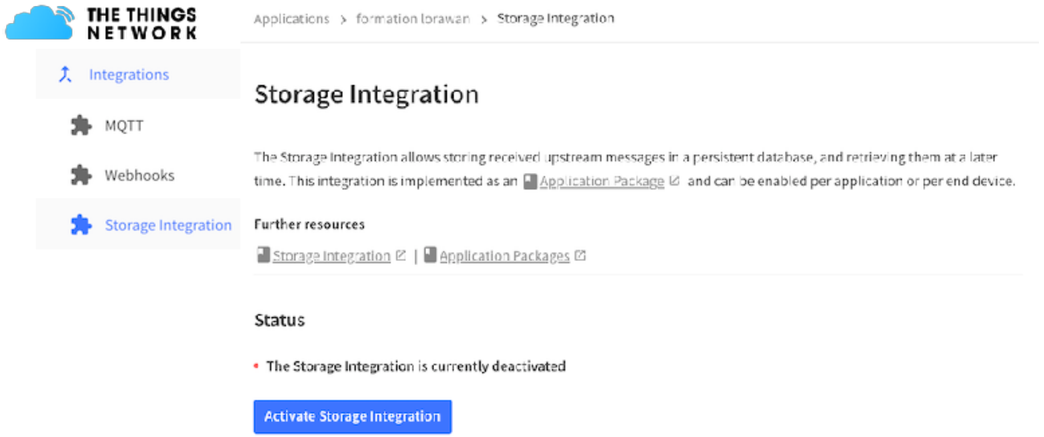


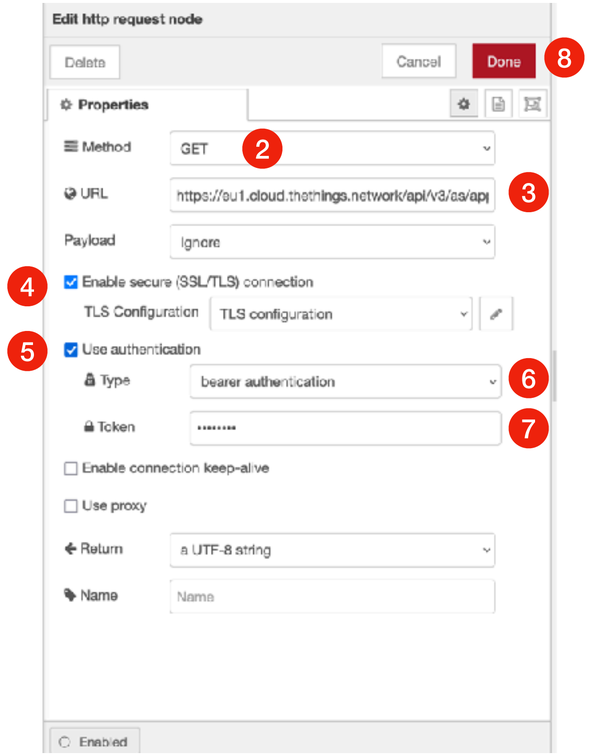
LA QUESTION A SE POSER :
qui est serveur et … qui est client ?

Le protocole HTTP entre les serveurs LoRaWAN et d'APP WEB SERVER

Requête GET : DEMO UPLINK







EndNode spécifique

```
GET https://eu1.cloud.thethings.network/api/v3/as/applications/formation-lorawan-1/packages/storage/{type}
GET https://eu1.cloud.thethings.network/api/v3/as/applications/formation-lorawan-1/devices/{device_id}/packages/storage/{type}
```
2

Récupère tous les messages uplink de mon objet :

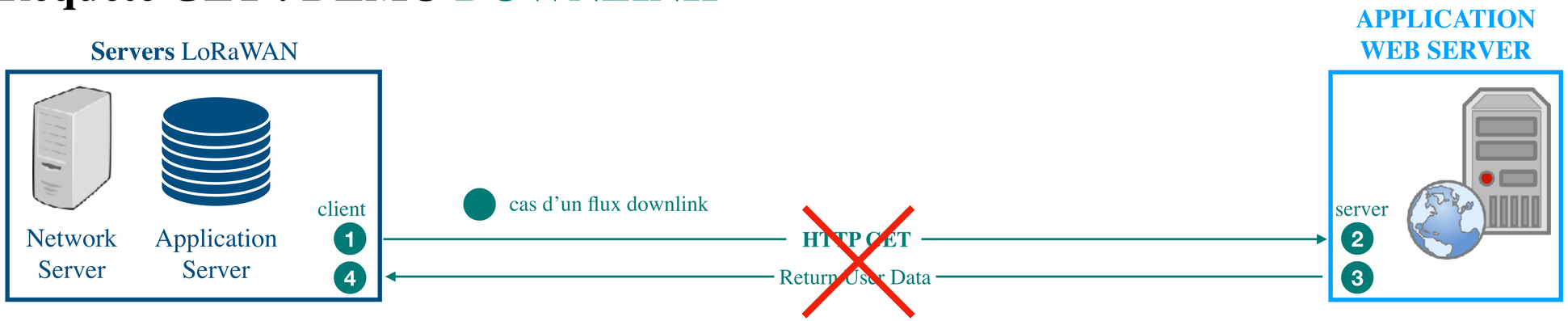
```
https://eu1.cloud.thethings.network/api/v3/as/applications/formation-lorawan-1/devices/{device_id}/packages/storage/uplink_message
```
3

APIKEY Application

7 **NNSXS.TFCHJDHAFXF7FYSWZK4YMBMXTBINGHNZQH6MT5Y.KX5OG5XBZXO6VQCQEUHFYIJQWXHRNFHXWGDOQTFSL7WDICHCKFPA**

Le protocole HTTP entre les serveurs LoRaWAN et d'APP WEB SERVER

Requête GET : DEMO DOWNLINK



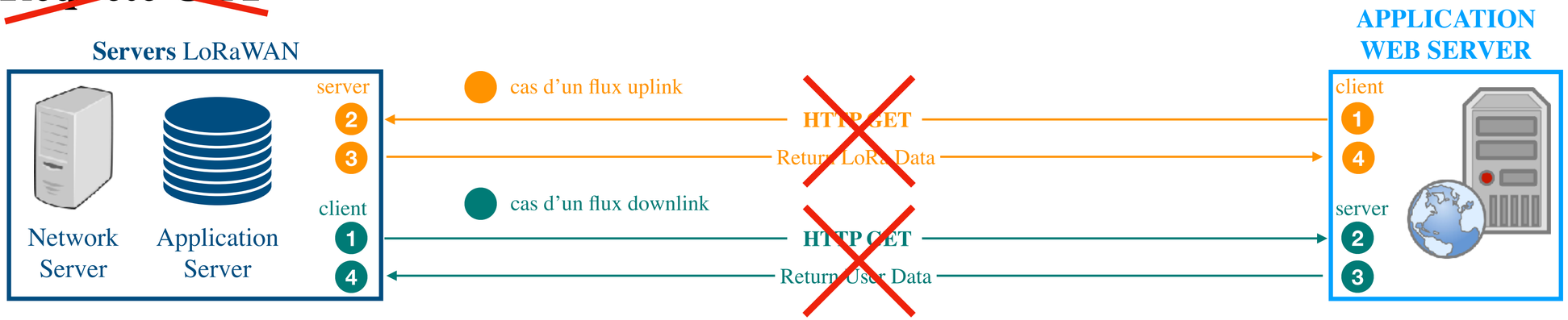
on a donc 2 PROBLEMES avec le HTTP GET pour un downlink

>>> on ne retiendra donc pas la requête HTTP GET

- **ni pour un downlink**
- **ni pour le uplink**

Le protocole HTTP entre les serveurs LoRaWAN et d'APP WEB SERVER

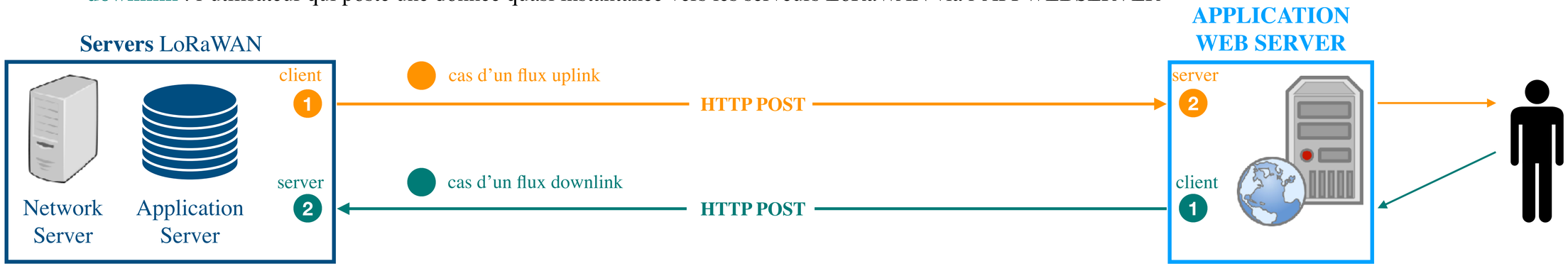
~~Requête GET~~



Requête POST

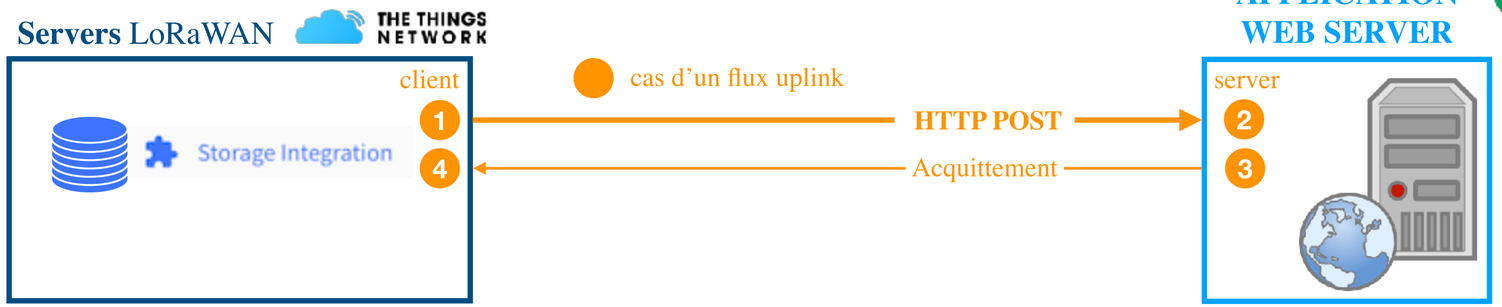
solution **uplink** : serveurs LoRaWAN qui postent les données sans trop de latence et sans que l'APPWEBSERVER en fasse la demande !!!

downlink : l'utilisateur qui poste une donnée quasi instantanée vers les serveurs LoRaWAN via l'APPWEBSERVER



Le protocole HTTP entre les serveurs LoRaWAN et d'APP WEB SERVER

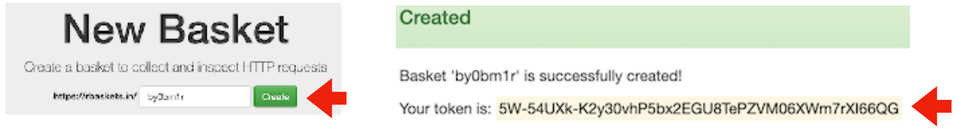
Requête POST : DEMO UPLINK



2 THE THINGS NETWORK applications > formation lorawan > integration > webhook

1 Configuration du serveur d'application (AWS)

https://rbaskets.in/web ← utilitaire en ligne



Création d'un EndPoint sur un réseau public avec token (authentification) créé permettra de revenir dessus protège un peu les données

https://rbaskets.in/by0bm1r ← il faudra envoyer les requêtes à cette url

Il n'y a plus qu'a attendre les données pour les voir dans l'interface présentée…

3 https://rbaskets.in/by0bm1r

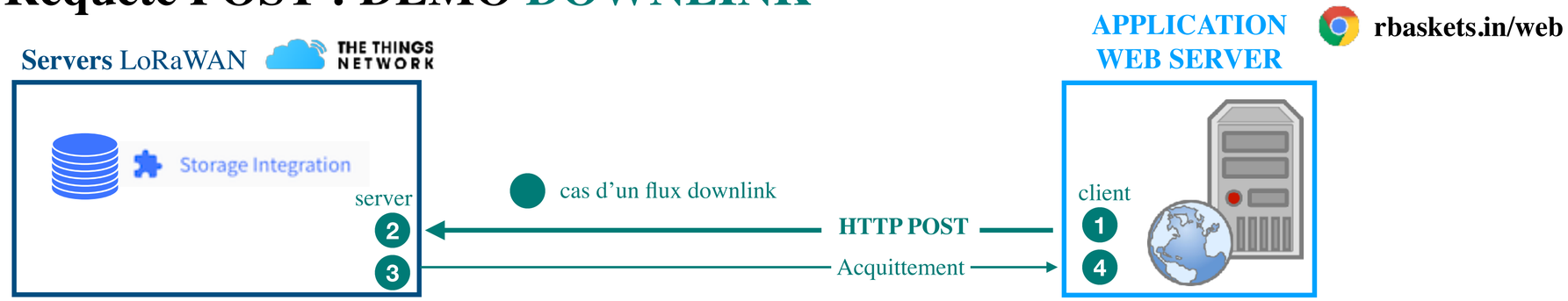
1 post reçu → [POST] 19:20:16 17/07/2021

"frm_payload": "AA==", ← contient le payload de la donnée du End Node

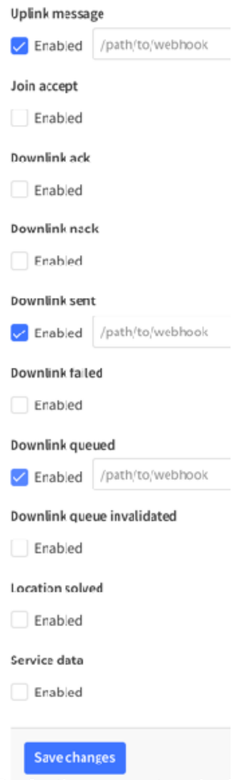
On a bien les données reçues, il faut ensuite :
les traiter, les mettre en forme, etc…

Le protocole HTTP entre les serveurs LoRaWAN et d'APP WEB SERVER

Requête POST : DEMO DOWNLINK



THE THINGS NETWORK applications > formation lorawan > integration > webhook



https://www.thethingsindustries.com/docs/integrations/webhooks/scheduling-downlinks/

```
/api/v3/as/applications/{application_id}/webhooks/{webhook_id}/devices/{device_id}/down/push
```

```
https://eu1.cloud.thethings.network/api/v3/as/applications/formation-lorawan-1/webhooks/
integration-http-post/devices/0004a30b001e0833/down/push
```

Même principe pour un downlink mais moins rapide à mettre en place donc pas vu ici !

Les protocoles de communication pour Exploiter les données

Partie

3 **Le protocole MQTT**
Définition

Le protocole

MQTT.ORG

protocole populaire qui s'est imposé de lui-même pour l'IoT

👌

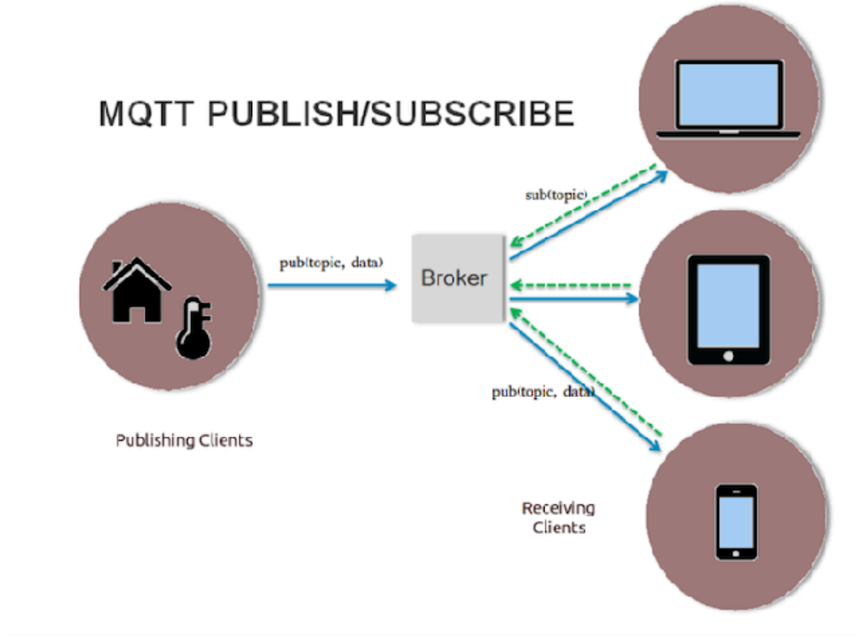
Protocole MQTT : Message Queuing Telemetry Transport

Protocole créé en 1999 par : - Dr **Andy Stanford-Clark** de IBM
- **Arlen Nipper** de Arcom (devenu Eurotech)

Objectif :
- **Protocole - implémenté au dessus de TCP/IP**
- **fiable et rentable pour relier les dispositifs de surveillance à des serveurs d'entreprise** distants (dans les secteurs du pétrole et du gaz)
- Conçu pour des clients réseaux dont les ressources sont limitées

MQTT est un protocole ouvert (tjs étroitement associé à IBM), et supervisé par OASIS (Organization for the Advancement of Structured Information Standards)
2 versions principales : MQTT 3.1.x et MQTT 5.0

Définitions Publisher - Broker - Suscriber



Publisher (éditeur) :
- publie un message sur un (ou plusieurs) sujet (**topics**) dans le **broker**

Subscriber (abonné) :
- s'abonne à un (ou plusieurs) sujet (**topics**) dans le **broker**
- reçoit tous les messages envoyés par le **publisher**

Broker (courtier) :
- route tous les messages des **publishers** vers les **subscribers**

Topic (sujet) :
- se compose d'un ou plusieurs niveaux séparés par un slash : **/**

Protocole MQTT : Publisher -> Broker <- Subscriber

EXPLICATION 1 → 8

1
- pas besoin de reconfigurer le **Broker** en cas d'envoie de données il doit les accepter sous réserve d'authentification
- idem en cas de récupération des données le transfère se fera si à nouveau le demandeur est authentifié

PUBLISHER
entité qui fourni des informations
(exemple capteur)

client publisher (1)
mesure de temp

client publisher (2)
mesure de temp + lux

2 données poussées vers le broker
répertoriées avec des info spécifiques
ex : un capteur Temp va publier dans
- le topic Temp

3 données poussées vers le broker
ex : un capteur Temp + Lux va publier dans
- le topic Temp
- le topic lux

4 topic temperature

BROKER centralise toutes les données

5 topic lux

SUBSCRIBER
(exemple actionneur)

6 il va s'intéresser au topic température et souscrire à celui ci

client subscriber (1)
chauffage

7 il va s'intéresser aux topics
- lux
- température

client subscriber (2)
volet roulant est subscriber des topics température + lux

X

8 **Subscribers** et **Publishers** ne se connaissent pas entre eux :
- il n'y a jamais de dialogue direct entre eux
- leurs intérêts communs sont uniquement les **Topics**

- un **Publisher** / **Subscriber** peut apparaitre / disparaitre sans reconfiguration du broker !!!
- les **Topics** peuvent être créés à l'infini par des **Publishers**
- les **Subscribers** pourraient souscrire à des **Topics** qui n'existent pas encore

Protocole MQTT : Topics

Un Broker MQTT est un serveur TCP, les Publishers / Subscribers sont des clients TCP

MQTT n'impose pas de format de donnée pour le PAYLOAD (charge utile)

La taille maximale autorisée est de 256 Mio par publication

Le format des données peut être binaire, JSON, XML, String, …

Les clients de souscription **MQTT (Subscribers)** s'enregistrent auprès du **Broker** via des **Topics** (ensemble de string séparées par des slashs)

Ils demandent ainsi à être notifiés lorsque quelqu'un publie sur ces topics

Exemple d'un **Topic** de température : **/sensor/bureau/temperature**

Pour souscrire à un ensemble de **Topics** on utilisera **# : qu'importe ce qu'il y a en dessous de l'arborescence**
ou **+ : qu'importe ce qui est à ce niveau de l'arborescence**

Exemples :

Si un client publie (**Publisher**) sur les **Topics** :
/sensor/bureau/temperature et
/sensor/bureau/humidity,
un autre client pourra écouter ces deux **Topics** à la fois via la syntaxe : /sensor/bureau/#

Si plusieurs clients publient leurs températures et humidités en intercalant les différents endroit sur leur **Topic**,
un autre client peut écouter toutes les températures ainsi :

/sensor/+/temperature <== Il recevra alors les températures du bureau (/sensor/bureau/temperature), du salon (/sensor/salon/temperature), etc.

Eric DUVIEILBOURG for ANF IoT Perfectionnement : eric.duvieilbourg@univ-brest.fr

Les protocoles de communication pour Exploiter les données

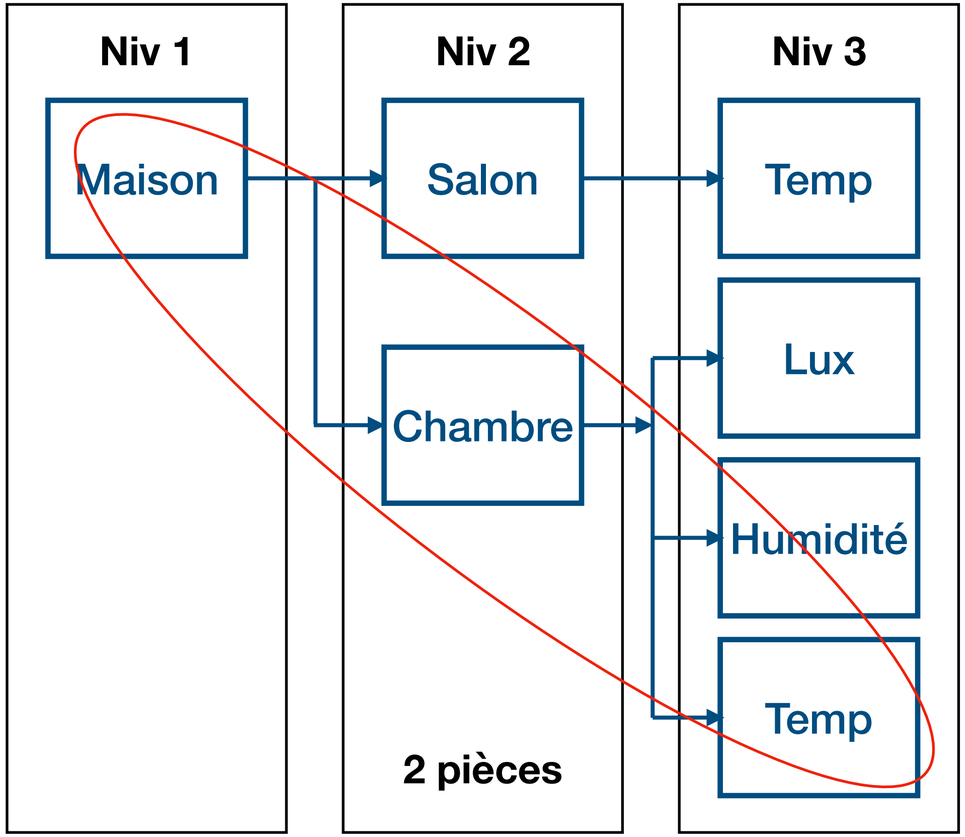
Partie

3

Le protocole MQTT

Exercice

Protocole MQTT : Exercice d'une maison pour expliquer les Topics



3 types de capteurs

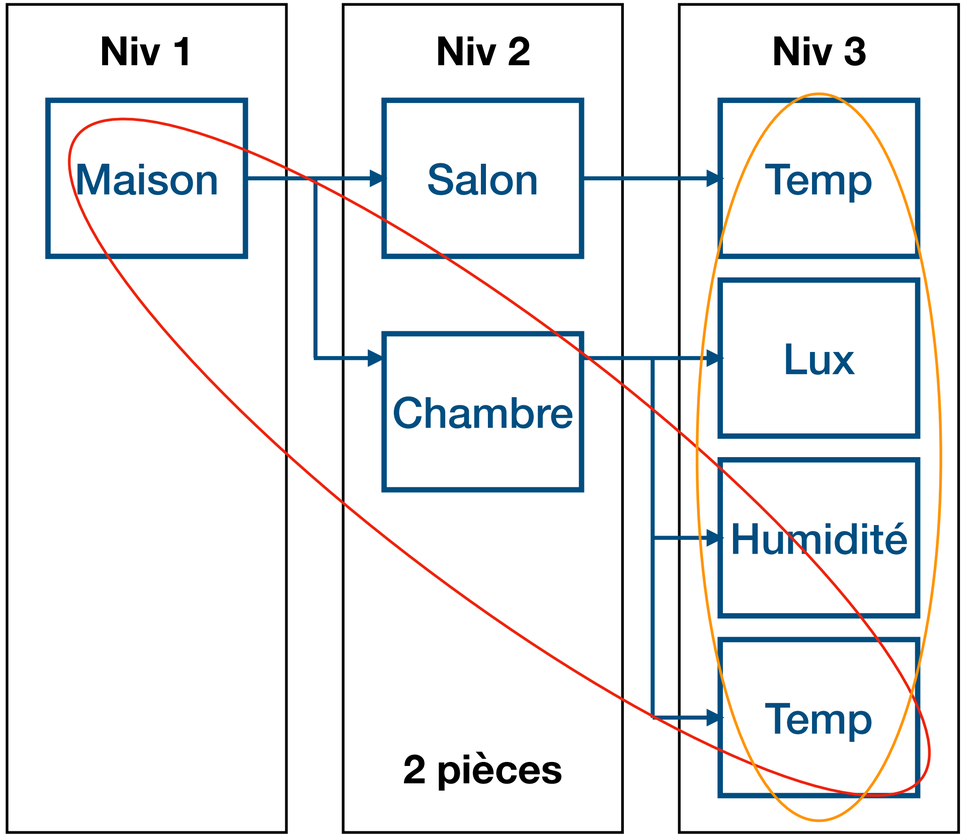
maison avec 2 pièces 3 types de capteurs

un **Topic** est défini par une **string** représentant l'ensemble de la hiérarchie

des caractères spéciaux peuvent désigner plusieurs topics en même temps

Détail	Nom (chaine de caractère)
température de la chambre de la maison	maison/chambre/temp
température du salon de la maison	
température l'humidité et le lux de la chambre de la maison	
température de toutes les pièces de la maison	

Protocole MQTT : Exercice d'une maison pour expliquer les Topics



3 types de capteurs

maison avec 2 pièces 3 types de capteurs

un topic est défini par une str représentant l'ensemble de la hiérarchie

des caractères spéciaux peuvent désigner plusieurs topics en même temps

Détail	Nom (chaine de caractère)
température de la chambre de la maison	maison/chambre/temp
température du salon de la maison	
température l'humidité et le lux de la maison	
température de toutes les pièces de la maison	

Les protocoles de communication pour Exploiter les données

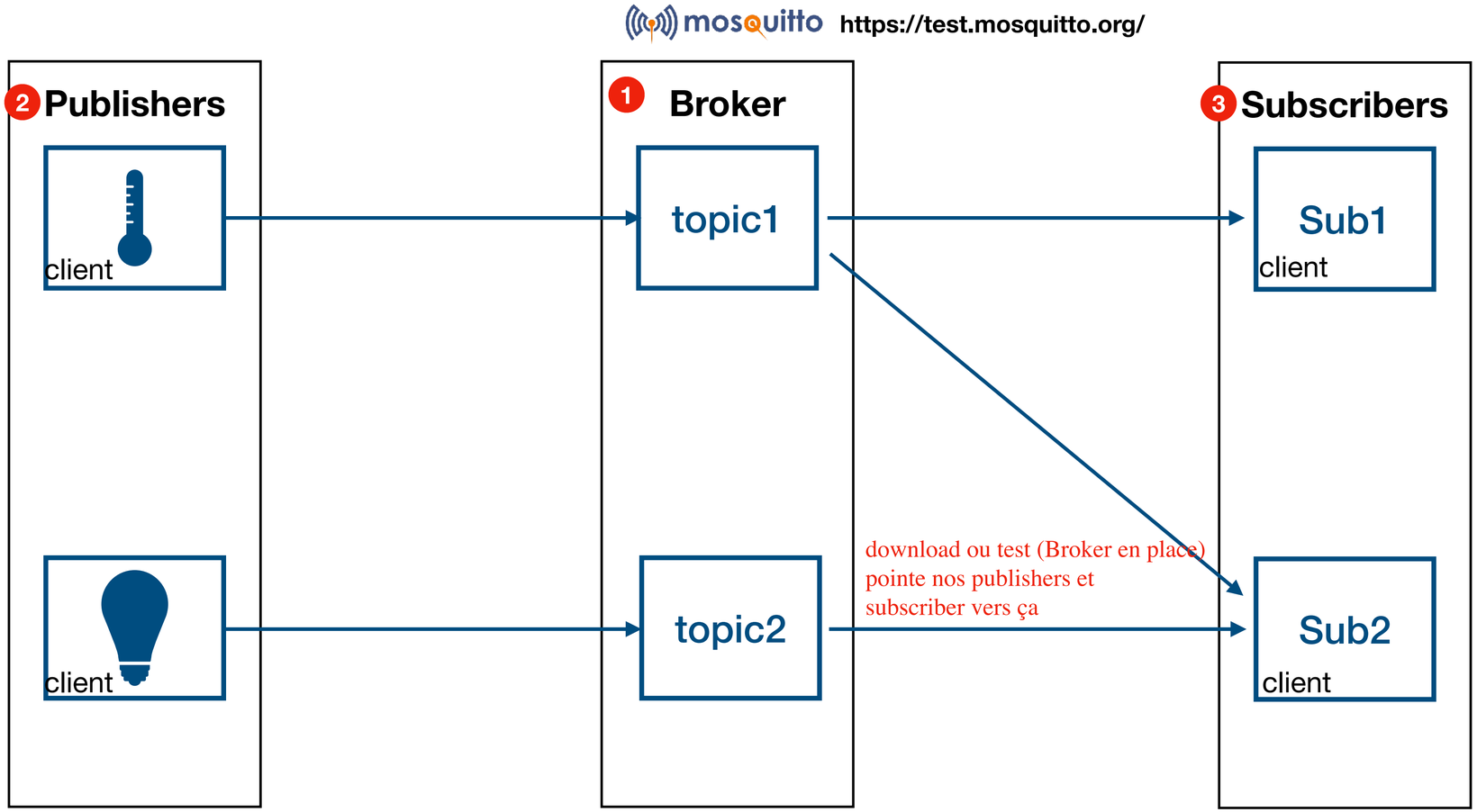
Partie

3 **Le protocole MQTT**

Démo / TP

Protocole MQTT : spécifier un Broker, démo avec mosquitto

An open source MQTT **broker**



dès qu'un client publie sur un Topic, il est automatiquement créé

Protocole MQTT : spécifier un Broker, démo avec mosquitto

An open source MQTT **broker**

1 mosquitto https://test.mosquitto.org/

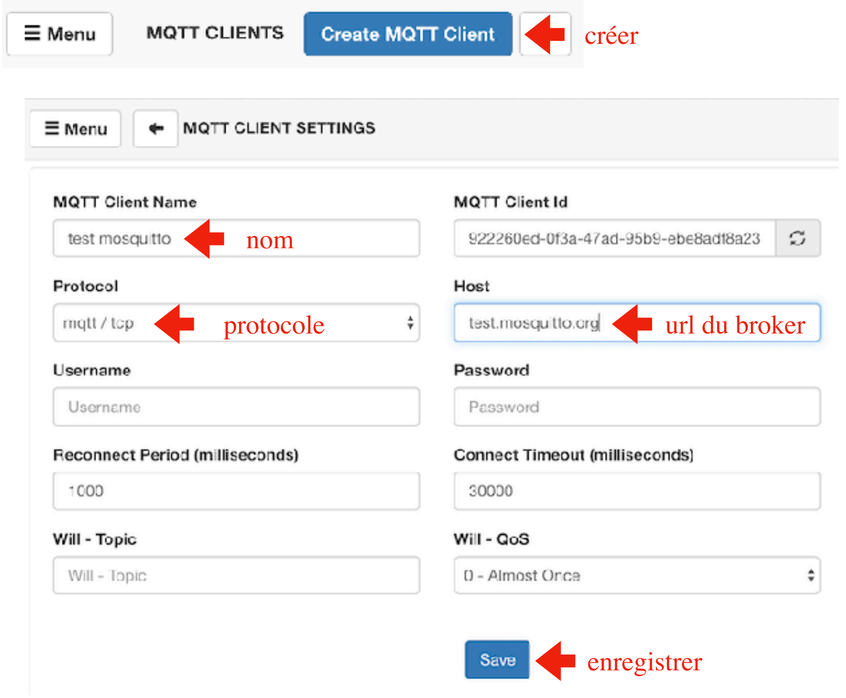
Test

You can have your own instance of Mosquitto running in minutes, but to make testing even easier, the Mosquitto Project runs a test server at test.mosquitto.org where you can test your clients in a variety of ways: plain MQTT, MQTT over TLS, MQTT over TLS (with client certificate), MQTT over WebSockets and MQTT over WebSockets with TLS.

Broker déjà configuré ici

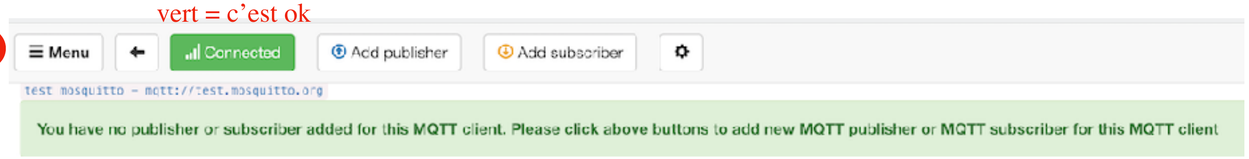
2 https://chrome.google.com/webstore/detail/mqttbox/kaajoficamnjijhkeomgfljpicifbkaf/related

Publishers (MQTTBOX)

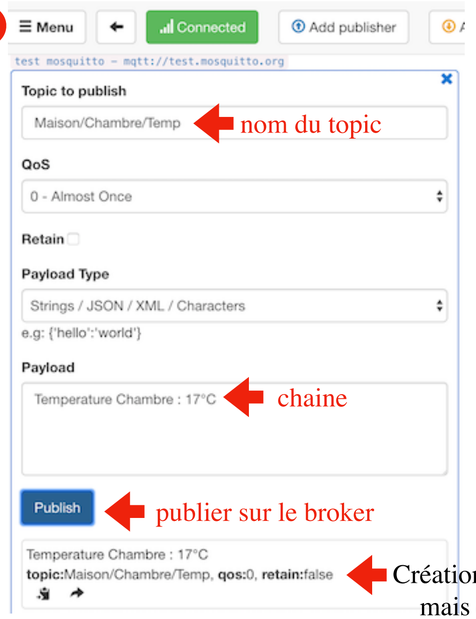


vert = c'est ok

3



4



Création d'un **Topic** Maison/Chambre/Temp
mais comme pas de **Subscriber**
il sera ***discard (effacé du Broker)***

Protocole MQTT : spécifier un Broker, démo avec mosquitto

An open source MQTT **broker**

1 mosquitto https://test.mosquitto.org/

Test

You can have your own instance of Mosquitto running in minutes, but to make testing even easier, the Mosquitto Project runs a test server at test.mosquitto.org where you can test your clients in a variety of ways: plain MQTT, MQTT over TLS, MQTT over TLS (with client certificate), MQTT over WebSockets and MQTT over WebSockets with TLS.

Broker déjà configuré ici

2 https://chrome.google.com/webstore/detail/mqttbox/kaajoficamnjijhkeomgfljpicifbkaf/related

Publishers (MQTTBOX)

Menu | MQTT CLIENTS | Create MQTT Client — créer

Menu | MQTT CLIENT SETTINGS

MQTT Client Name: test mosquitto — nom

MQTT Client Id: 922260ed-0f3a-47ad-95b9-ebe8adf8a23

Protocol: mqtt / tcp — protocole

Host: test.mosquitto.org — url du broker

Username: Username

Password: Password

Reconnect Period (milliseconds): 1000

Connect Timeout (milliseconds): 30000

Will - Topic: Will - Topic

Will - QoS: 0 - Almost Once

Save — enregistrer

vert = c'est ok

3 Menu | Connected | Add publisher | Add subscriber

test mosquitto – mqtt://test.mosquitto.org

You have no publisher or subscriber added for this MQTT client. Please click above buttons to add new MQTT publisher or MQTT subscriber for this MQTT client

4 Menu | Connected | Add publisher

test mosquitto – mqtt://test.mosquitto.org

Topic to publish: Maison/Chambre/Temp — nom du topic

QoS: 0 - Almost Once

Retain

Payload Type: Strings / JSON / XML / Characters

e.g: {'hello':'world'}

Payload: Temperature Chambre : 17°C — chaine

Publish — publier sur le broker

Temperature Chambre : 17°C
topic:Maison/Chambre/Temp, qos:0, retain:false

5 Création d'un **Subscriber**

Topic to subscribe: Maison/Chambre/Temp — même nom du topic

QoS: 0 - Almost Once

Subscribe — publier sur le broker

Maison/Chambre/Temp — attente de publi.

Maison/Chambre/Temp

Temperature Chambre : 17°C — valeur

qos : 0, **retain** : false, **cmd** : publish, **dup** : false, **topic** : Maison/Chambre/Temp, **messageId** : , **length** : 48, **Raw payload** : 841011091121011149 711611711410132671049710998114101325832495519417667 — nom du topic

Protocole MQTT : spécifier un Broker, démo avec mosquitto

An open source MQTT **broker**

1 mosquitto https://test.mosquitto.org/

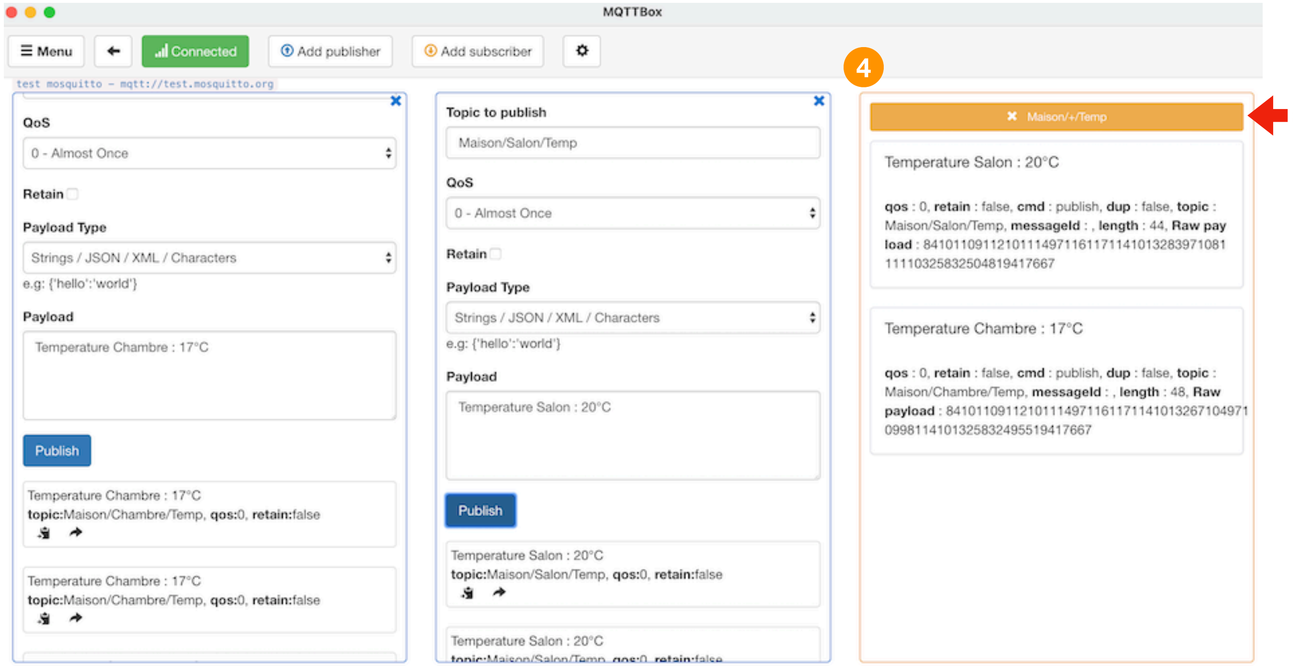
2 Publishers (MQTTBOX)

3 Topic to subscribe

#

Broker public : tous les Topics publiés sont visibles par tout le monde !!!!

si on met que # dans le topic to subscribe (**durant un temps très court**) : on va récupérer tous les **Topics** de tout le monde !!!



subscriber Maison/+/Temp

On a mis les **publishers** et les **subscribers** sur la même machine (localhost) pour améliorer la qualité de service sur machine différentes on modifira le Qos :

QoS 0 : ''Au plus une fois'' on tolère la perte

PUBLISH >

Message supprimé après envoi

QoS 1 : ''Au moins une fois'' les données doivent arriver doublons possibles

PUBLISH > < PUBLISH Ack

Message enregistré

Survie à une perte de connexion

Duplications possibles

QoS 2 : ''Exactement une fois'' les données doivent arriver exclusivement une fois

PUBLISH > < PUBLISH receive

PUBLISH release > < PUBLISH Complete

Message enregistré

Survie à une perte de connexion

Duplications impossibles

Dans la suite, Username et Password à renseigner du server LoRaWAN

Eric DUVIEILBOURG for ANF IoT Perfectionnement : eric.duvieilbourg@univ-brest.fr

Les protocoles de communication pour Exploiter les données

Partie

4

Le protocole MQTT

Application avec TTN et notre capteur !!!

Protocole MQTT : servers LoRaWAN + MQTTBox (démo)

cf doc de ttn

THE THINGS NETWORK

Le Broker est déjà existant

1 c'est en principe vrai pour tous les servers LoRaWAN

2 s'activera automatiquement dès qu'un **publisher** ou **subscriver** sera connecté dessus

3 **topics** seront figés à l'intérieur du **broker**

Network Server

Application Server

Pub(Devices) → 1 topic uplink = topic à lire

4 data envoyées **par** End Node

Sub(Devices) ← 2 topic downlink = topic à écrire

6 data à envoyer **au** End Node

MqttBox

https://www.thethingsindustries.com/docs/integrations/mqtt/

client — topic subscriber

5 v3/<username>/devices/<DevID>/up

ex : *v3/formation-lorawan-1@ttn/devices/0004a30b001e7b33/up*

v3/+/devices/+/up si tous les devices de toutes les applications

client — topic publisher

7 `v3/{application id}@{tenant id}/devices/{device id}/down/push` `replace`

ex : *v3/formation-lorawan-1@ttn/devices/0004a30b001e7b33/down/push*

\+ renseignez le payload : 8

Applications › formation lorawan › MQTT

MQTT

The Application Server exposes an MQTT server to work with streaming events. In order to use the MQTT server you need to create a new API key, which will function as connection password. You can also use an existing API key, as long as it has the necessary rights granted. Use the connection information below to connect.

Connection credentials

Public address	eu1.cloud.thethings.network:1883	← url
Public TLS address	eu1.cloud.thethings.network:8883	
Username	formation-lorawan-1@ttn	← username
Password	•••	← password

MQTT Client Name: TTN V3 ← name	MQTT Client Id: 4acc7624-f97d-4ec1-878b-c14afcad
Protocol: mqtt / tcp ← protocole	Host: eu1.cloud.thethings.network:1883 ← url
Username: formation-lorawan-1@ttn ← username	Password: •••••••••••••••••••••••••••••••••• ← password
Reconnect Period (milliseconds): 1000	Connect Timeout (milliseconds): 30000
Will - Topic: Will - Topic	Will - QoS: 0 - Almost Once

Save

☰ Menu | ← | Connected | Add publisher | Add subscriber

TTN V3 - mqtt://eu1.cloud.thethings.network:1883

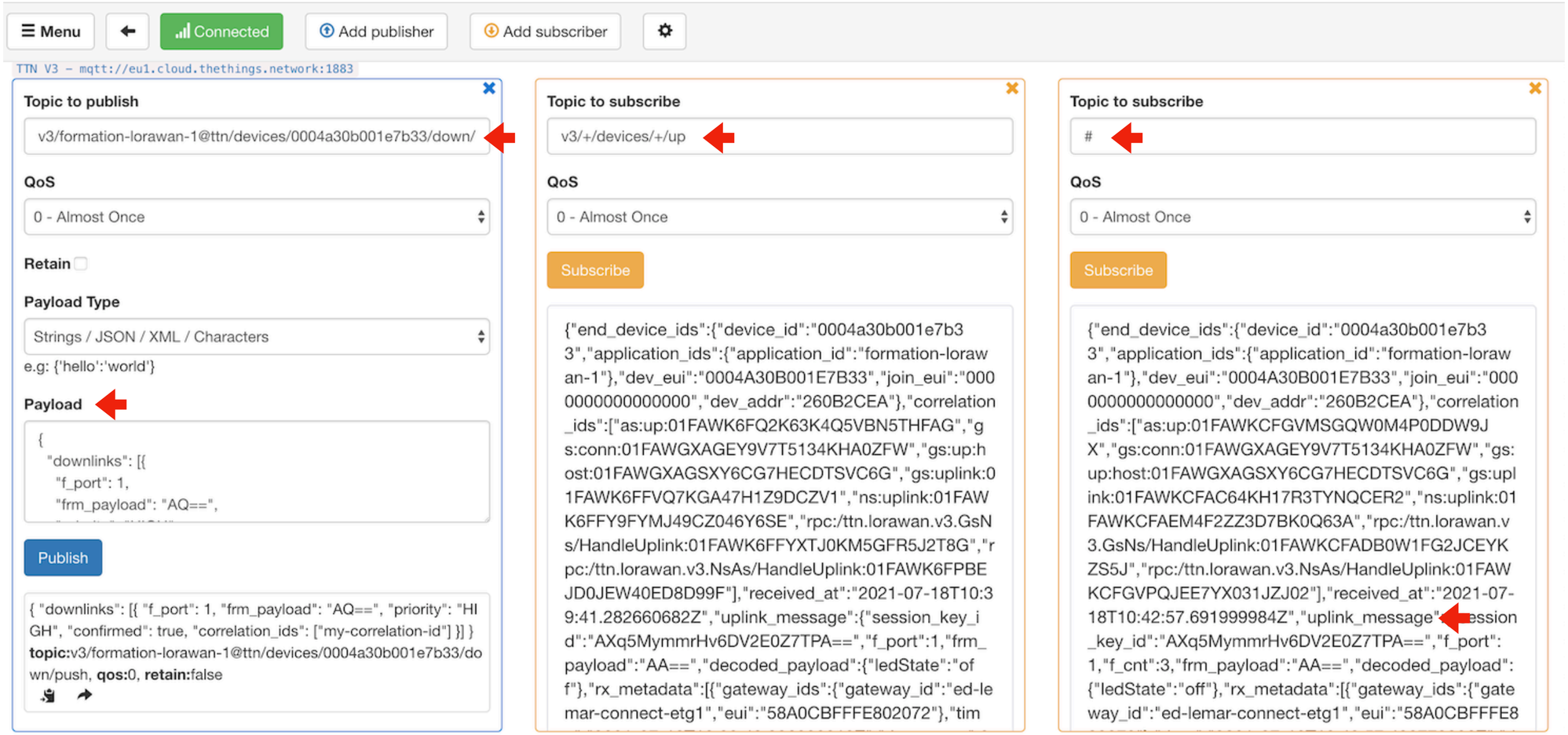
8 dans topic publisher : payload

```
{
  "downlinks": [{
    "f_port": 1,
    "frm_payload": "AQ==",
    "priority": "NORMAL"
  }]
}

puis :

{
  "downlinks": [{
    "f_port": 1,
    "frm_payload": "AA==",
    "priority": "HIGH",
    "confirmed": true,
    "correlation_ids": ["my-correlation-id"]
  }]
}
```

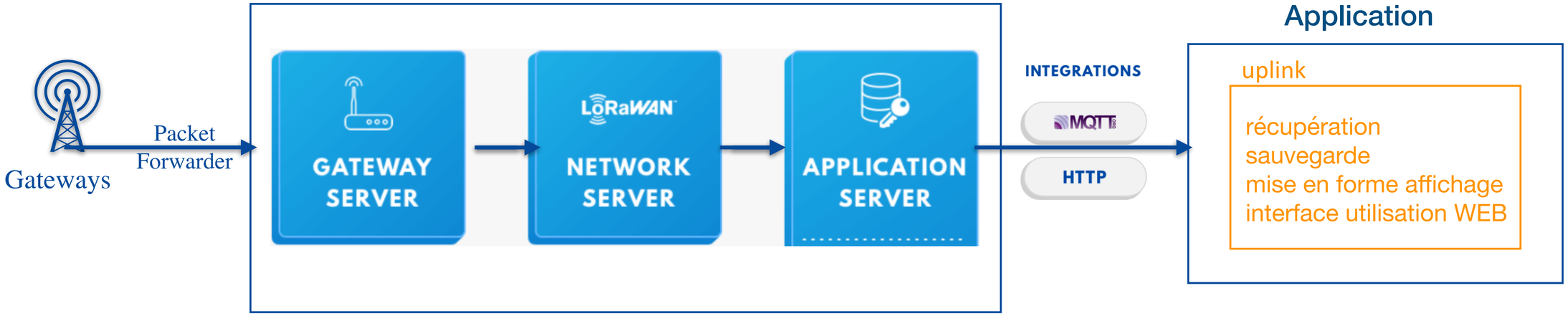
Protocole MQTT : servers LoRaWAN + MQTTBox (démo)



tous les événements sont donc représentés ici
plus besoin de se connecter au ttn live data
on pourra ainsi exploiter ces données dans une application complète

Réaliser une Application utilisateur

recevoir, stocker, traiter, et afficher ces données sur une page Web



Réaliser une Application utilisateur

recevoir, stocker, traiter, et afficher ces données sur une page Web

Service à rendre	Solutions génériques	Choix technologiques possibles			
Service web : *Mise à disposition de contenu WEB*	*Serveur WEB* *Interface utilisateur*		Grafana	kibana	NGINX, APACHE
Gestion de l'affichage des données : Courbes, histogrammes, tableaux, jauges, etc...	*Solution de monitoring complète, librairies pour Dashboard...*	Node-RED	chronograf		plotly, Dash, HIGHCHARTS
Sauvegarde des données	*Base de données* *Fichier texte*		influxdb	elasticsearch	SQLite, MySQL, Prometheus
Récupération des données	*Endpoint HTTP* *Subscriber ou Publisher MQTT*		telegraf	logstash	python HTTP / MQTT

TIC

MERCI !

Nouveau

+

à suivre plus tard dans la formation…