Rapport de conception Mycelium 3.0

CHEREL Valentin - DERRIEN Thomas
MAUGER Arthur - OUVRARD Pierre - POURCHER Pierrick

Encadrants : PARLAVANTZAS Nikos - PAROL-GUARINO Volodia

Avec la participation de :
MOUREAU Julien

2023-2024

Figure 1 – Mycélium : projet de suivi environnemental

Table des matières

1 Glossaire

— **API** : *Application Programming Interface*, ensemble de règles permettant à des logiciels de communiquer entre eux de manière standardisée, par exemple pour interagir avec les fonctionnalités ou les données d'une application, service ou système externe.

— **Broker MQTT** : Serveur intermédiaire qui facilite la communication asynchrone entre les dispositifs IoT en gérant les opérations de *"publish"* (publication) et *"subscribe"* (abonnement) pour l'échange de messages.

— **ChirpStack** : Logiciel open-source pour la gestion des appareils utilisant LoRaWAN.

— **Cloud computing** : Modèle informatique permettant l'accès à des ressources informatiques partagées, telles que des serveurs, des applications et des services, via Internet, permettant aux utilisateurs d'accéder et de stocker des données sans avoir besoin de posséder ou de gérer leur propre infrastructure.

— **FaaS** : *Function as a service*, modèle de déploiement de cloud computing en microservices serverless.

— **Flux RSS** : Format de données XML permettant de distribuer des mises à jour de contenu de manière structurée.

— **Fog computing** : Modèle informatique décentralisé qui étend le cloud computing en rapprochant le traitement des données vers le périphérique où elles sont générées plutôt que de les centraliser dans des centres de données distants.

— **Gateway** : Passerelle de communication entre deux réseaux distincts.

— **Grafana** : Plateforme open source de visualisation et d'analyse de données, utilisée pour créer des tableaux de bord interactifs et informatifs.

— **InfluxDB** : Base de données de séries temporelles open source conçue pour stocker, interroger et visualiser des données chronologiques.

— **IoT** : Internet des objets, décrit le réseau d'objets physiques constitués de capteurs, de logiciels et d'autres technologies dans le but de connecter et d'échanger des données avec d'autres appareils et systèmes sur Internet.

— **Kubernetes (K8S)** : Plateforme open source destinée à automatiser le déploiement, la mise à l'échelle et la gestion des applications conteneurisées. Kubernetes facilite la gestion d'applications distribuées et microservices sur un cluster de machines.

— **K3S** : Distribution légère de Kubernetes conçue pour les environnements avec des ressources limitées, tels que les systèmes embarqués, les dispositifs IoT ou

les clusters de développement.

— **LoRaWAN** : Protocole de communication basse consommation, longue portée et faible bande passante très utilisé pour l'IoT. Il se base sur le réseau LoRa (*Long range*).

— **Microservice** : Architecture logicielle basée sur le développement et le déploiement de petits services autonomes, indépendants et spécialisés, qui interagissent entre eux pour former une application complète.

— **MQTT** : *Message Queuing Telemetry Transport*, protocole de messagerie léger *publish-subscribe* basé sur le protocole TCP/IP.

— **OpenFaaS** : Plateforme open-source qui facilite le déploiement et la gestion de fonctions serverless, permettant ainsi l'exécution de code de manière événementielle sans se soucier de l'infrastructure sous-jacente.

— **Raspberry Pi** : Ordinateur peu coûteux de la taille d'une carte de crédit, pouvant effectuer toutes les actions d'un ordinateur.

— **Scalabilité** : Capacité d'un système à augmenter ou diminuer ses ressources en fonction de l'usage de ce système.

— **Serverless** : Modèle d'exécution pour les applications sans gestion de serveur par l'utilisateur, avec des coûts payés selon l'utilisation.

— **VPS** : *Virtual Private Server*, machine virtuelle, créée sur un serveur physique et employant ses ressources pour offrir à ses utilisateurs les mêmes fonctionnalités qu'un serveur dédié.

— **XML** : *eXtensible Markup Language*, langage de balisage utilisé pour organiser et structurer des données de manière lisible par les humains et les machines.

2 Introduction

La mise en place de la ligne B du métro rennais a laissé son empreinte sur l'environnement, engendrant des émissions de gaz à effet de serre et la disparition d'espaces verts. Face à ces enjeux, la ville de Rennes s'est engagée à contrebalancer ces impacts en initiant des projets de renaturation. C'est dans ce cadre que le projet Mycélium a pris forme en 2021, se concentrant sur le suivi de la renaturation du parc de la Croix Verte, localisé sur le campus de Rennes 1.

En partenariat avec le laboratoire des Géosciences de Rennes et sous la supervision de Laurent LONGUEVERGNE. Ce laboratoire fait partie de l'OSUR, (*Accueil Observatoire des sciences de l'univers de Rennes*). Ce projet vise à suivre et à évaluer la renaturation de La Croix Verte à l'aide de capteurs connectés.

Pour bien comprendre l'objectif de Mycélium, penchons-nous sur un exemple concret : lors de fortes pluies, le niveau d'eau d'une rivière peut augmenter brusquement. Dans ce cas, il est intéressant de notifier l'utilisateur lorsqu'il y a un risque d'inondation. Ce scénario de suivi de la montée des eaux sera prochainement étudié par une équipe de scientifiques rennais au Népal.

Dans notre cas, l'utilisateur peut par exemple être un scientifique chargé de la surveillance de la Croix Verte. Nous pouvons aussi envisager que les mesures effectuées avec les différents capteurs deviennent plus fréquentes lors de ces intempéries, afin d'évaluer avec plus de précision ces évènements soudains. Les données récoltées par les capteurs sont sauvegardées dans une base de données et peuvent être consultées via une interface utilisateur.

Entre 2021 et 2023, le projet Mycélium a connu deux versions, qui ont permis de mettre en place une architecture de collecte et de traitement des données basée sur le principe du fog computing. L'objectif de cette nouvelle version : Mycélium 3.0, est de simplifier l'architecture du projet afin de garantir de meilleures performances, une prise en main facilitée et une faible consommation énergétique.

Ce rapport de conception vise à décrire l'architecture interne du projet, sa modélisation et l'interfaçage des différents modules entre eux. Il commencera par présenter l'architecture globale de Mycélium 3.0 en détaillant succinctement chaque module qui le compose.

3 Architecture de Mycélium 3.0

3.1 Architecture matérielle

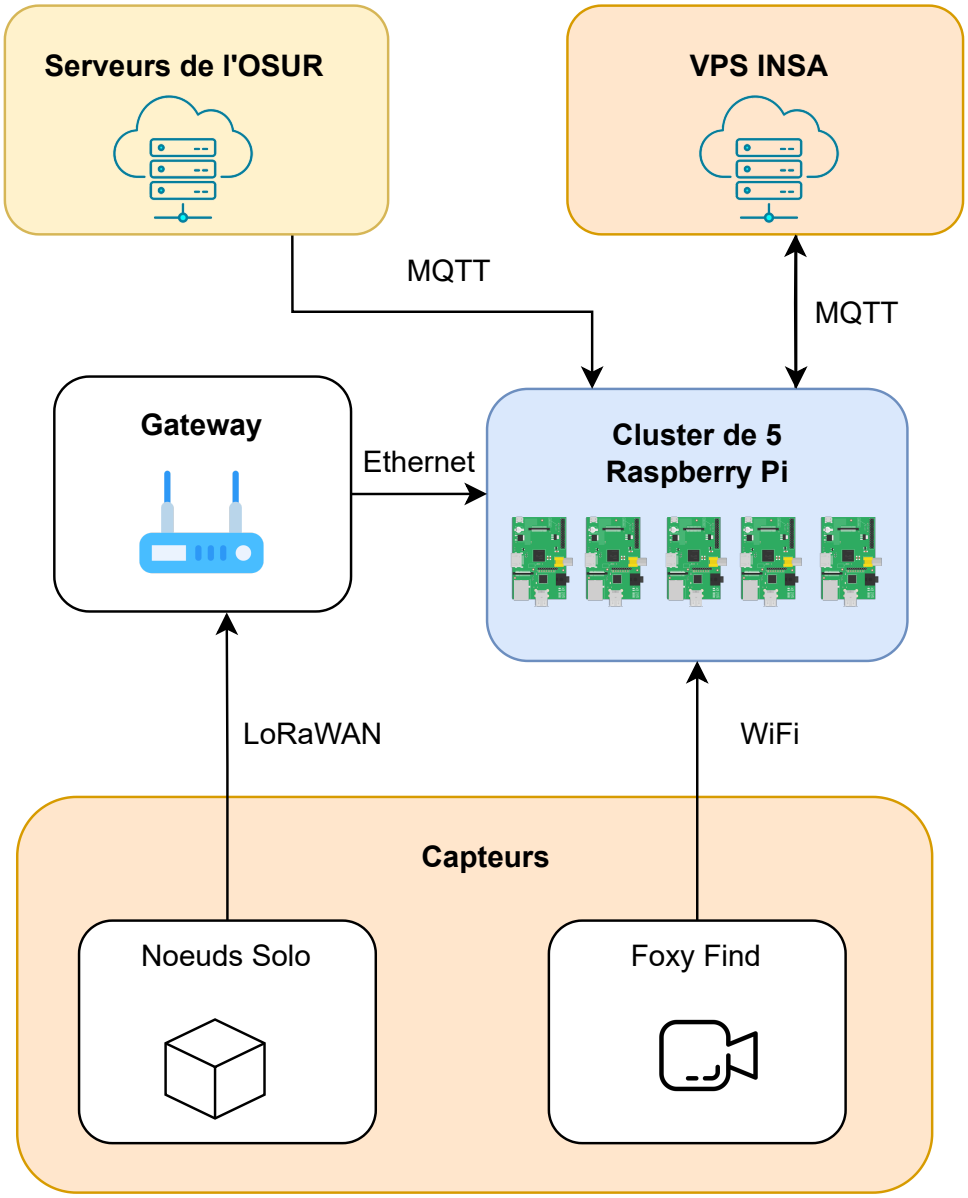


FIGURE 2 – Schéma de l'architecture matérielle de Mycelium 3.0

L'architecture matérielle de Mycélium, représenté en figure 2 intègre un réseau de capteurs déployés sur la Croix-Verte, conçu pour recueillir diverses données environnementales. Les capteurs, appelés nœuds SoLo[1], sont des dispositifs autonomes qui utilisent la technologie LoRaWAN[2] pour communiquer. LoRaWAN est un protocole de communication radio qui consiste à envoyer les données sur une fréquence de 868 MHz. Le réseau LoRaWAN assure une transmission fiable d'une faible quantité de données sur une portée de plusieurs kilomètres et une consommation énergétique très faible.

D'un point de vue matériel, chaque nœud SoLo est contenu dans un boitier étanche qui abrite plusieurs capteurs (température, humidité, luminosité, accéléromètre, pression, pluviomètre). Ces capteurs sont paramétrables à l'aide de fichiers au format JSON, permettant de configurer la fréquence d'envoi, la fréquence de prise de mesures, le réseau, etc.

En plus de ces nœuds SoLo, l'OSUR nous donne accès à ses capteurs répartis sur le campus : un port a été ouvert aux connexions entrantes effectuées avec les IP de l'INSA afin de recevoir les nouvelles mesures de ces capteurs. Le cluster est donc directement connecté aux serveurs de l'OSUR. Il existe également un module appelé Foxy Find. C'est une caméra qui permet de détecter le passage d'un renard. Elle est connectée au reste de Mycélium via WiFi

Dans le cadre du projet, ces capteurs ont pour mission de collecter des données environnementales et de les transmettre à une gateway installée dans les locaux de l'INSA. Une gateway LoRaWAN agit comme une passerelle de communication entre les nœuds SoLo et un réseau IP. Elle centralise les données provenant des capteurs, les recevant depuis les nœuds SoLo et les renvoyant via une connexion Ethernet vers le cluster de Raspberry Pi[3].

Le cluster, placé dans le bâtiment INFO, est composé de cinq Raspberry ayant 2 Go de RAM. Une des Raspberry est considérée comme un noeud superviseur tandis que les quatre autres sont des noeuds contrôleurs, recevant les ordres du premier cité. Ce cluster aura pour objectif d'effectuer tous les traitements et analyses voulus sur les données récoltées par les capteurs.

Cependant, l'une des contraintes handicapantes inhérentes à ce système fog est le peu de mémoire vive disponible. Dans le cas où les ressources viendraient à manquer, un VPS a été ajouté. Ce serveur, hébergé par l'INSA, permet ainsi de venir en aide au cluster et de l'aider à réaliser des calculs en cas de besoin. Au-delà d'un certain palier d'utilisation du cluster, les calculs sont effectués sur le serveur de l'INSA.

3.2 Architecture logicielle

Au niveau de l'architecture logicielle, présentée en figure 3 nous avons imaginé une architecture simple permettant de mener à bien les différents objectifs du projet. Au niveau des capteurs, nous disposons tout d'abord des noeuds SoLo. Ils communiquent via le protocole LoRaWAN avec une Gateway située dans les locaux de l'INSA. Cette Gateway communique avec le cluster, et plus particulièrement avec Chirpstack[4] via une connexion Ethernet. Une explication plus détaillée du fonctionnement de Chirpstack est proposée en partie 3.2.1.

Chirpstack renvoie ensuite les données via le protocole MQTT[5] vers une fonction de décodage, qui permet de rendre accessibles et lisibles les données enregistrées par le capteur. Il y a aussi une possibilité de récupérer les données directement depuis les serveurs de l'OSUR via MQTT.

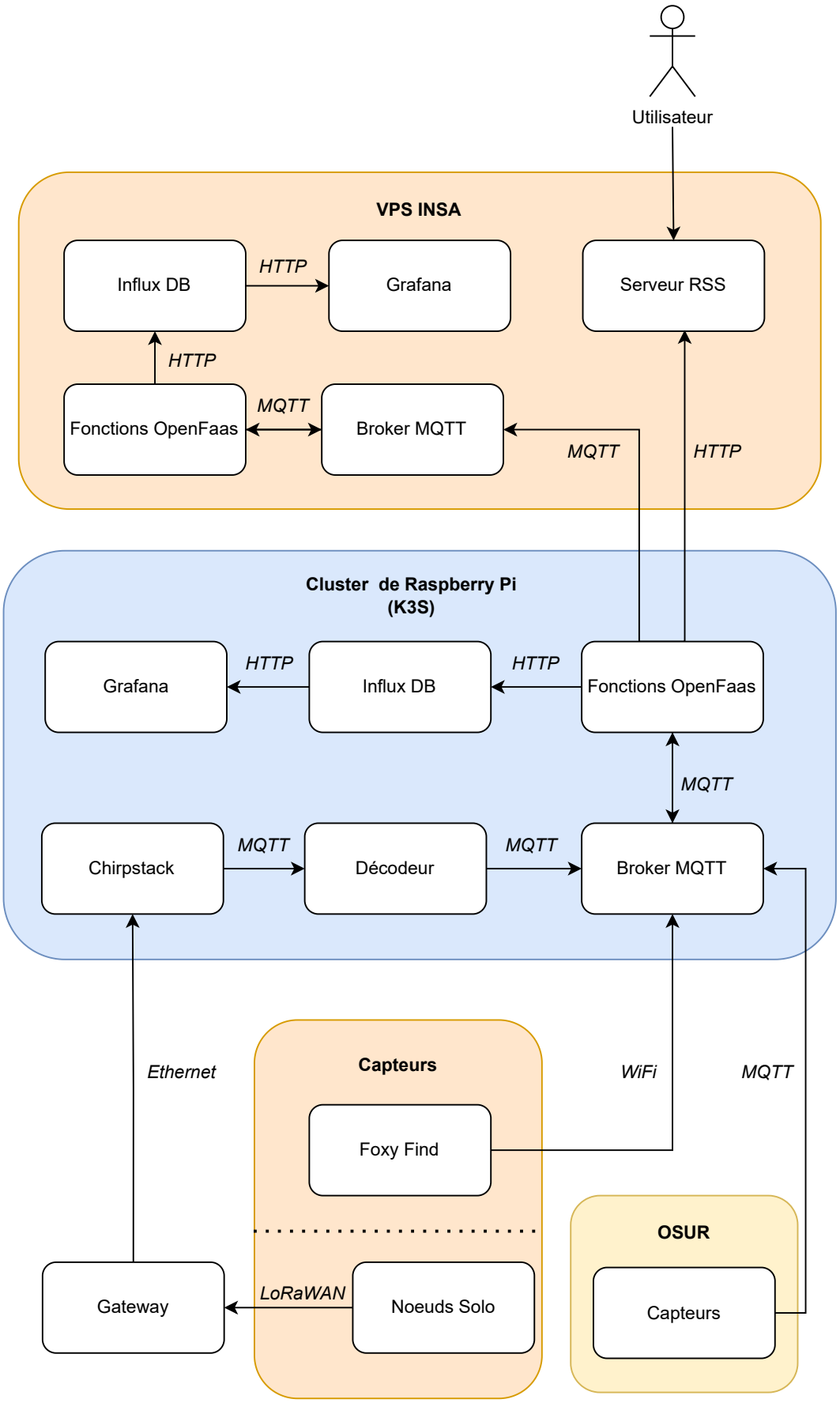


FIGURE 3 – Schéma de l'architecture logicielle de Mycelium 3.0

Les données sont alors transmises vers un broker MQTT, qui représente le cœur du cluster K3S[6]. Ce broker permet de faire transiter les données entre différentes fonctions OpenFaas[7], permettant d'effectuer les différentes fonctionnalités voulues. Les données sont par exemple sauvegardées sur InfluxDB[8], et peuvent être visualisées en utilisant Grafana[9]. Dans le cluster, les données présentes dans la base de données InfluxDB seront des données concernant l'état du cluster (taux d'utilisation du CPU, taux d'utilisation de la mémoire). Les fonctions présentes dans le cluster permettent de traiter les scénarios. Lorsqu'ils sont déclenchés, ceux-ci peuvent envoyer des notifications qui seront disponibles à l'utilisateur sous la forme d'un flux RSS.

Comme expliqué précédemment, le cluster communique avec le cloud de l'INSA, à l'aide de MQTT. Une architecture équivalente à celle du cluster est déployée dans le VPS et permet un traitement similaire. Celui-ci est utilisé pour de multiples raisons. Tout d'abord, les données de mesures des capteurs sont stockées sur la base de données InfluxDB présente dans le cloud. Pour éviter trop de communications inutiles entre le cluster, nous n'envoyons pas toutes les données relevées sur le capteur vers le cloud. Les données sont d'abord récoltées sur le cluster. Une fois par jour, un résumé des données sera envoyé vers le cloud afin d'être stocké.

Une autre utilité du VPS est de pouvoir aider le cluster lorsque la charge de calcul est trop importante. Au-delà d'un certain palier d'utilisation des capacités du cluster, les traitements sont effectués sur le cloud. Enfin, le serveur RSS permettant de renvoyer les notifications des scénarios à l'utilisateur sera déployé sur le VPS. C'est donc au cloud de l'INSA que l'utilisateur final se connectera pour visualiser les données ainsi que les notifications.

3.2.1 Chirpstack

Notre projet intègre ChirpStack, un ensemble de logiciels open source permettant la gestion et la configuration d'appareils utilisant le protocole LoRaWAN. Dans le cluster, Chirpstack va réceptionner les données provenant de la Gateway et va ensuite les transférer vers les fonctions de traitement du cluster à l'aide d'un broker MQTT.

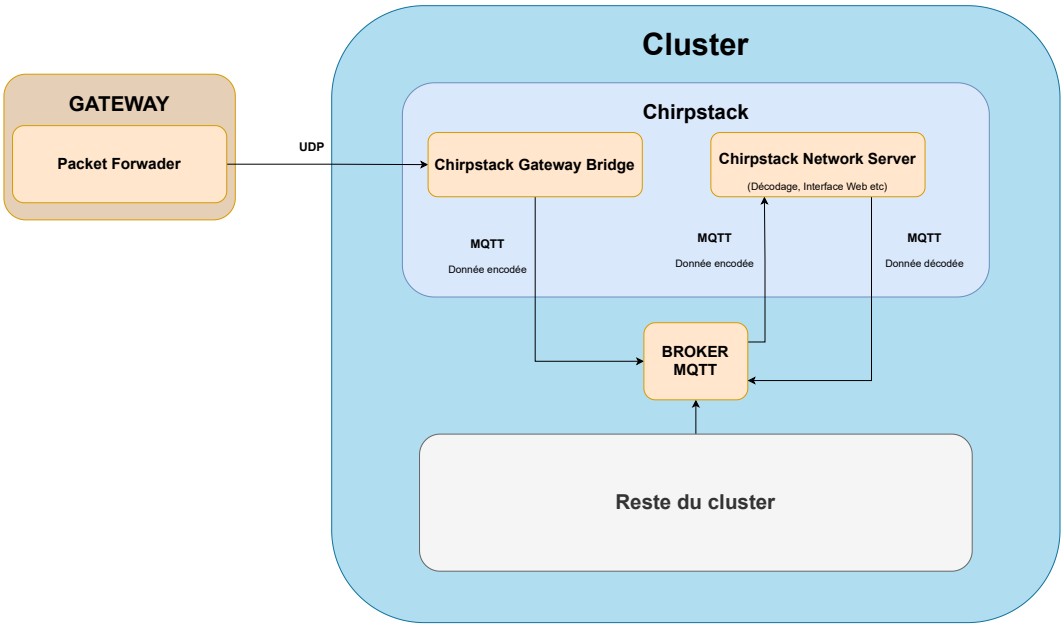


FIGURE 4 – Schéma de l'architecture de Chirpstack au sein de Mycelium 3.0

Comme nous pouvons le voir sur le schéma en figure 4, la Gateway envoie des paquets UDP vers le cluster. Les données sont ensuite réceptionnées par Chirpstack Gateway Bridge. ChirpStack Gateway Bridge agit comme un intermédiaire entre la gateway LoRaWAN et ChirpStack Network Server. Il reçoit les données encodées provenant des passerelles, les transforme et les publie sur un topic MQTT pour qu'elles soient utilisées par Chirpstack Network Server.

La partie Chirpstack Network Server permet ensuite de gérer les capteurs et la Gateway. Cette partie permet l'ajout de nouveaux capteurs ainsi que la configuration des Gateways et des capteurs. Pour cela, une interface Web est mise à disposition sur le port 8080. Elle permet aussi de décoder les données à l'aide de scripts écrits en Python et JavaScript. Les données décodées sont ensuite publiées sur le topic MQTT *"traiter_données"*.

Afin d'effectuer des traitements sur les données, il suffit de s'abonner au topic *"traiter_données"* et déclencher des fonctions lors de l'arrivée des données. Pour cela, nous utiliserons OpenFaaS que nous détaillons dans la partie suivante.

3.3 Fonctions OpenFaas et traitement automatique des données

Le traitement des données est rendu possible grâce à une organisation du code en fonctions selon le principe de Function as a Service (FaaS). En d'autres termes, chaque service est une fonction qui remplit une unique tâche. Les fonctions se servent de l'event broker pour communiquer. OpenFaaS sera utilisé afin d'implémenter cette architecture, présentée en figure 5

3.3.1 Organisation des services

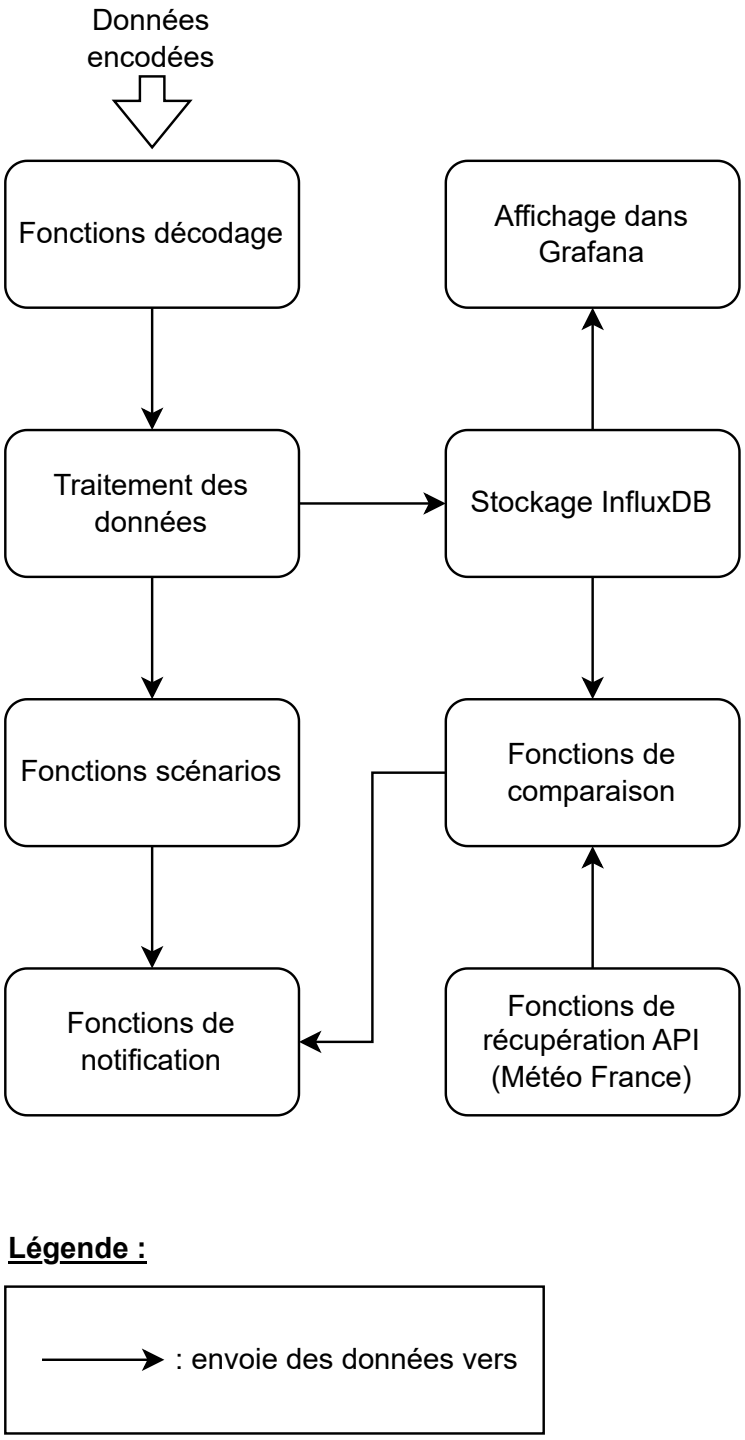


FIGURE 5 – Schéma de l'organisation des différentes fonctions OpenFaas

Les données reçues du broker MQTT sont encodées, une première fonction primordiale est donc la fonction décodage. Une fois les données en clair, nous pouvons commencer à les utiliser. D'une part, nous voulons les visualiser : pour cela, nous les stockons dans une base de données InfluxDB pour générer des graphiques et des tableaux de bord par la suite avec Grafana. D'autre part, nous les traitons en exécutant

des fonctions correspondant à des scénarios.

Par exemple, lorsque la température tombe dans les négatifs et que l'humidité est élevée, nous déclenchons une fonction de notification, qui alerte l'utilisateur d'une potentielle présence de neige. Ces informations sont consultables via un flux RSS. À l'avenir, nous aimerions pouvoir envoyer un message en direction du capteur, afin d'augmenter la fréquence de mesure des capteurs de température, humidité et pluviométrie, car c'est un moment intéressant à surveiller. Cependant, ce n'est pas encore possible, cette fonction n'est pas encore implémentée au niveau des capteurs. Cette implémentation est complexe, car elle nécessite de modifier dynamiquement la configuration des capteurs. Un projet parallèle est en cours pour essayer de résoudre ce problème.

Grâce à l'API de Météo France disponible gratuitement, nous avons accès à un immense et précieux ensemble de données. C'est très avantageux, car nous pouvons alors anticiper et moduler le comportement des capteurs en fonction des informations reçues de l'API. Nous pouvons ensuite choisir de faire varier la précision et la fréquence de nos prises de mesure de la manière décrite au paragraphe précédent. Dans des cas particuliers où les mesures provenant des deux sources ne coïncident pas (par exemple un écart de température supérieur à 15°C), nous pouvons en déduire qu'il y a une défaillance de la part de nos capteurs. Grâce à une fonction de notification, l'utilisateur peut être informé du problème.

Nous allons maintenant montrer l'objectif des différents scénarios. Nous allons ensuite expliquer le fonctionnement d'un des scénarios dans le contexte d'un système de type *publish/subscribe*, autour du broker MQTT.

3.3.2 Scénarios

- **Montée des eaux** : Vérifie l'évolution du niveau de l'eau.
- **Luminosité anormale** : Vérifie que la luminosité diminue lorsqu'il pleut.
- **Luminosité anormale selon l'heure** : Vérifie que la luminosité est cohérente avec l'heure (luminosité faible la nuit, plus élevée le jour).
- **Neige** : Vérifie s'il fait moins de 0°C et qu'il pleut en même temps.
- **Températures extrêmes** : Vérifie si les températures sont extrêmement basses ou extrêmement élevées (inférieur à -10°C ou supérieur à 50°C)
- **Températures anormales selon l'historique de températures** : Vérifie si les températures n'augmentent pas ou ne diminuent pas trop vite (écart de 5°C par rapport à la dernière mesure).
- **Vol** : Vérifie s'il y a un mouvement rapide du capteur, qui pourrait signifier un vol.
- **Capteurs endommagés** : Vérifie si les données mesurées sont aberrantes
- **Comparaison avec Météo France** : Vérifie si les données n'ont pas un écart supérieur à 15°C par rapport à la station météo située à Rennes. Pour cela, nous utilisons l'API gratuite mis à disposition par Météo France.

Le scénario que nous allons présenter est un scénario que nous avons créé. Un capteur permettant de mesurer le niveau de l'eau d'une rivière est bientôt disponible. Nous avons donc développé un scénario permettant de prévenir l'utilisateur en cas de montée soudaine du niveau de la rivière. Le fonctionnement de ce scénario est présenté dans la figure 6.

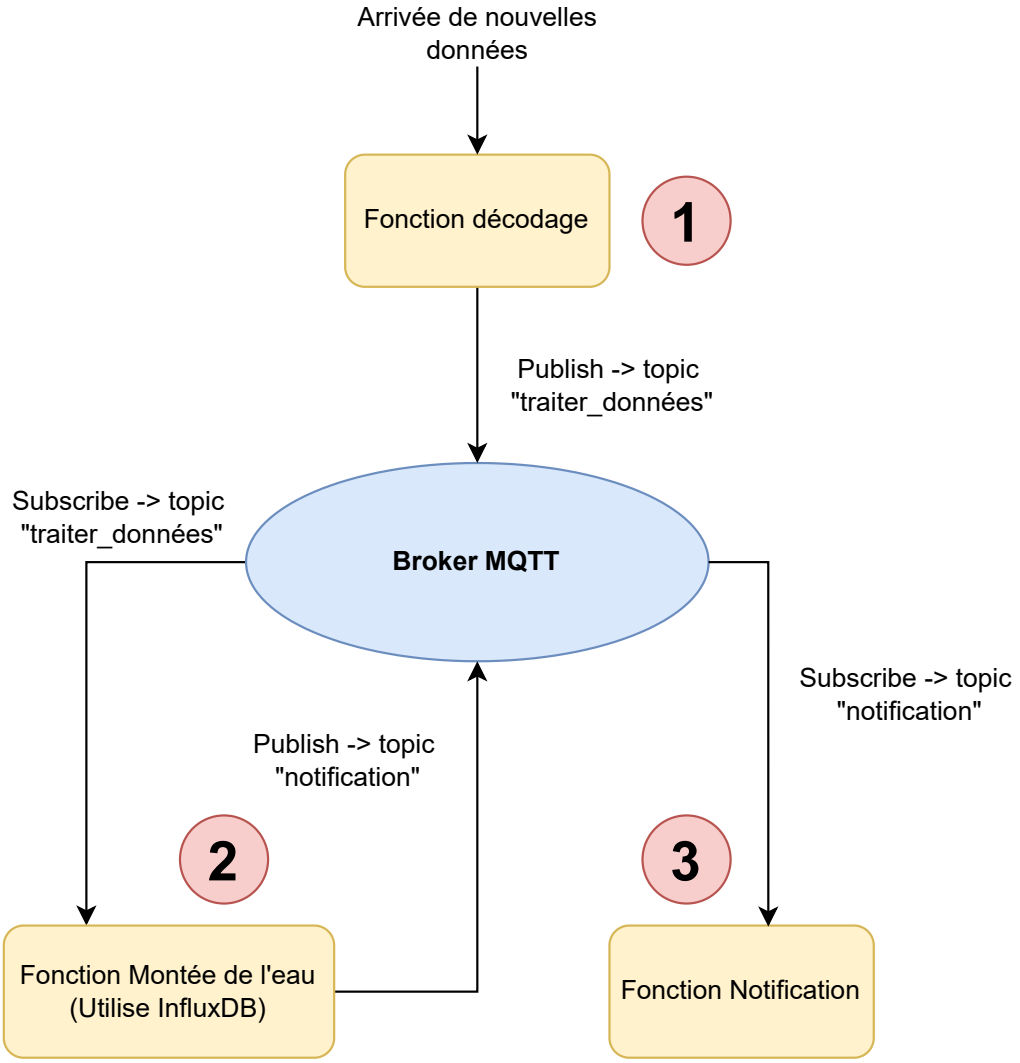


FIGURE 6 – Schéma du fonctionnement du scénario montée des eaux

Les données reçues par le capteur sont d'abord décodées par la fonction décodage. Une fois cette action effectuée, elles sont publiées sur un topic du broker MQTT, le topic *"traiter_ données"*. En parallèle, des fonctions sont abonnées au broker et attendent des données envoyées sur ce topic. Dans cet exemple, la fonction *Montée de l'eau* attend des données publiées sur le topic *"traiter_ données"*, les récupèrent quand elles arrivent, les traitent et publient le message de notification sur le topic *"notification"* si le niveau de l'eau a augmenté de plus de 5cm depuis la dernière mesure. Pour savoir cela, la fonction ira chercher la dernière mesure dans InfluxDB afin de la comparer.

Il y a enfin une fonction *Notification*, utilisée afin d'envoyer l'alerte sur un flux RSS. Cela se fait via un *POST* sur le serveur RSS présent sur le VPS. Cette fonction sera la même pour tous les scénarios, mais elle ne sera pas appelée avec les mêmes données.

Cette organisation est la même pour les autres scénarios. Seul le fonctionnement de la fonction dans la partie 2 est modifié.

4 Facilitation de la prise en main du projet

4.1 Réalisation d'une documentation

Au cours de la reprise du projet Mycélium, nous avons été confrontés à de nombreuses difficultés. Cela montre un problème important concernant la transmission de celui-ci dû à une structure complexe et un manque de documentation. Nous avons décidé qu'il était nécessaire de produire un guide facilitant la prise en main du projet, que ce soit pour les programmeurs ou pour les utilisateurs finaux.

Nous avions originellement pour objectif de produire un guide unique regroupant les informations que nous estimions nécessaires à la reprise du projet. Cependant, il est rapidement apparu qu'un tel document serait lourd et peut-être désagréable à utiliser. Nous avons ainsi décidé, par souci de clarté, de diviser ce document en un ensemble de guides et documents informatifs séparés en deux catégories :

La première catégorie correspond à la liste ci-dessous des documents que nous produisons dans l'objectif de donner une vision globale du projet :

— **Guide de bienvenue / Quickstart** : Le guide de bienvenue nous semble essentiel pour garantir une bonne appropriation du projet par les prochaines équipes, et de pouvoir les lancer dans le projet rapidement. Pour nous familiariser avec le projet, nous avons assisté à une présentation de la part de nos encadrants qui nous a permis de comprendre globalement le projet, mais qui n'était pas assez précise pour nous apprendre comment concrètement le faire fonctionner. De même, la lecture des rapports des années précédentes nous a été utile, mais était longue et manquait également de précision. Ainsi ce guide sera une synthèse de la présentation et des documents, mais devra également avoir un aspect pratique en indiquant où retrouver des informations pertinentes concernant chaque élément de la structure globale.

— **Liste de l'équipement matériel** : La liste du matériel permettra de pouvoir facilement avoir une idée du matériel à notre disposition et donc de se projeter quant à l'utilisation que nous pourrions en faire. Nous ne nous contenterons pas de dire le type de matériel, mais donnerons précisément le modèle afin de faciliter d'éventuelles recherches ou discussions sur le sujet.

— **Liste des technologies utilisées** : La liste des technologies utilisées permettra de centraliser l'information concernant le nombre et la nature des technologies utilisées. Elle donnera notamment les versions utilisées afin de minimiser les risques d'incompatibilités.

— **Schéma de synthèse de l'architecture** : Le schéma de synthèse de l'architecture devra être le plus complet possible. Il servira à comprendre la structure globale du projet et les interactions entre les différents éléments. Il sera accompagné d'explications complémentaires aidant à le comprendre.

La seconde catégorie correspond aux documents rattachés à chaque élément de la structure du projet. Les éléments sont les différentes technologies utilisées au sein du projet. Il y aura par exemple *"OpenFaas"*, *"MQTT"* ou *"Chirpstack"*. Ces documents sont :

— **Guide d'utilisation** : Un guide d'utilisation de l'élément contiendra des informations concrètes sur les actions à effectuer pour le faire fonctionner, ses fonctionnalités utiles ainsi qu'une description de son utilisation normale dans le cadre du projet.

— **Guide d'installation** (si pertinent) : Un guide d'installation sera créé pour les technologies nécessitant une installation. Il devra être complet, fin, préciser les versions à utiliser et contiendra éventuellement des images d'illustrations.

Nous organiserons nos guides dans des répertoires à la manière représentée dans la figure 7 :

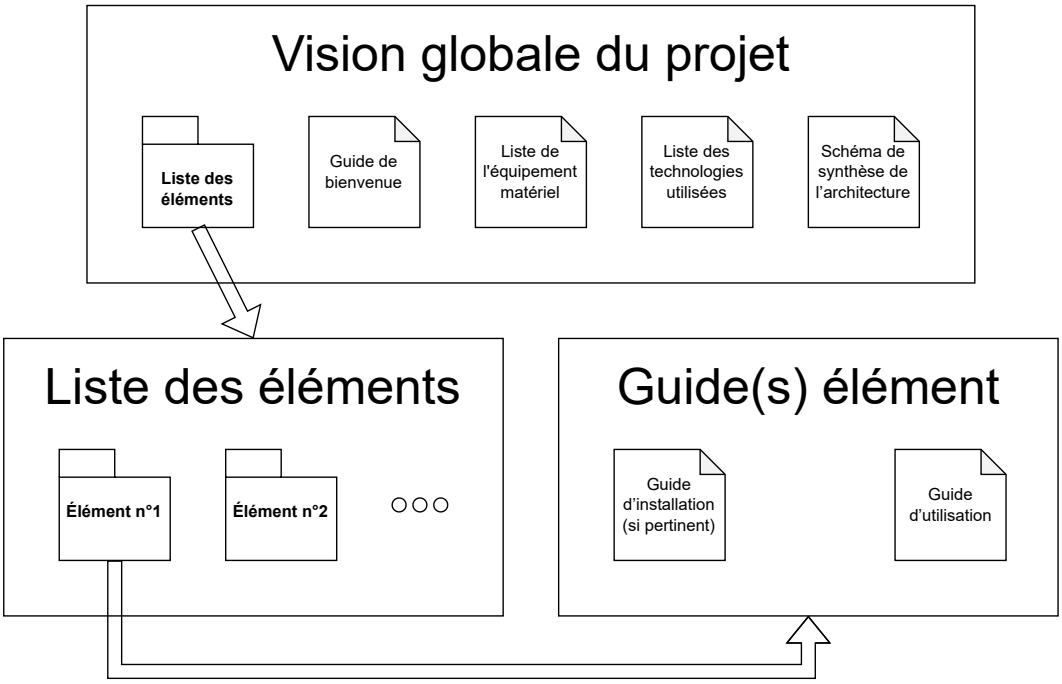


FIGURE 7 – Schéma de l'organisation hiérarchique de notre documentation

Comme vous pouvez le constater, la structuration en différents documents et leur organisation en fonction de leur nature rend plus lisible la répartition des informations.

Ne souhaitant pas reproduire la complexité des répertoires Gitlab dont nous avons hérité des années précédentes, nous restons sur une architecture simple à trois niveaux. Il est évident que, par souci de préservation de leur pertinence et de leur utilité, nos documents devront être mis à jour en cas de changement de structure du projet, d'équipement ou de logiciel, nous ferons cependant en sorte de conserver leur simplicité d'organisation et demanderons aux équipes suivantes de faire de même.

Nous allons maintenant voir comment garantir que cette documentation soit utile, et ce qui peut être considéré comme une avancée par rapport aux années précédentes.

Tout d'abord, les guides sont directement issus de notre propre expérience lors

de la prise en main du projet, nous les avons réalisés au fur et à mesure de nos avancées. Nous avons fait en sorte qu'ils soient les plus clairs possibles, et doivent correspondre à ce que nous aurions souhaité avoir au commencement de Mycélium v3.

N'avançant pas tous à la même vitesse et sur les mêmes sujets nous avons déjà eu l'occasion d'utiliser les guides au sein de notre groupe ce qui a permis de les tester même si ça n'est sans doute pas suffisant, car nous étions déjà assez avancés dans le projet et avions la possibilité de communiquer directement avec d'autres personnes plus avancées.

Nous avons donc décidé de tester nos guides "en situation réelle", c'est-à-dire sur des personnes extérieures au projet, mais ayant tout de même des connaissances de base en informatique. Nous demanderons à des camarades de classe de tenter des installations et utilisations à l'aide de nos guides et récupèrerons ensuite leurs avis afin d'évaluer leur utilité et éventuellement de les améliorer.

Concernant les listes de matériels et de technologies utilisés, nous considérons que de par leur nature il est inutile de les tester. Nous testerons notre schéma de synthèse de l'architecture en présentant notre projet à des camarades de classe et en voyant s'ils le considèrent suffisamment clair.

4.2 Mise en place d'une Machine Virtuelle (VM)

Le cluster de Raspberry Pi que nous utilisons pour traiter les données récoltées est au centre de notre projet, cependant il est difficile de travailler directement dessus pour plusieurs raisons : déjà parce qu'il nécessite un accès direct au matériel et donc d'être physiquement présent au bâtiment informatique de l'INSA Rennes et d'avoir la clé de la salle où il est stocké, mais également, car cela crée un risque de retour en arrière de ses fonctionnalités voir de rendre le rendre inopérant en cas de mauvaise manoeuvre. Ainsi par souci de sécurité et de flexibilité nous travaillons d'abord sur une émulation avant d'implémenter des changements sur le cluster. Pour cela, nous utilisons le logiciel de virtualisation open source QEMU[10] (Quick EMUlator).

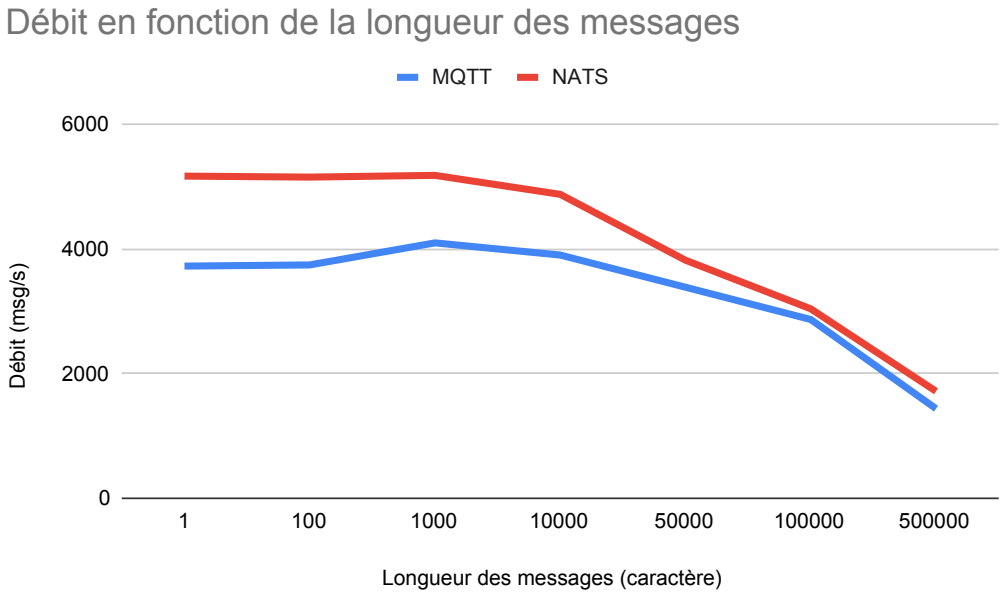
Nous mettons en place un guide visant à faciliter l'acquisition d'un cluster de machines virtuelles adapté afin de pouvoir simuler le cluster de Raspberry Pi. Le problème principal est celui de la compatibilité des versions, que ce soit entre les groupes d'une année à l'autre où au sein même d'un groupe, ce problème est rendu d'autant plus complexe par le fait que pour fonctionner le cluster demande l'installation d'un certain nombre de technologies qui peuvent, elles aussi, être disponibles sous plusieurs versions.

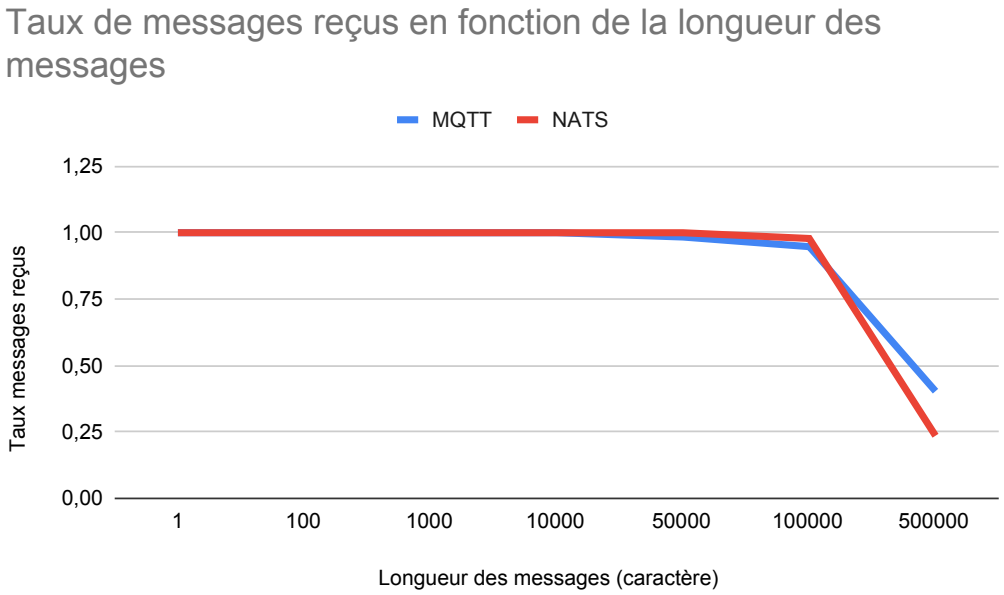
Ainsi, pour répondre à cette problématique de compatibilité, nous avons décidé de mettre à disposition des futures équipes un fichier contenant l'image disque du cluster de machines virtuelles que nous utilisons. Cela permettra à la nouvelle équipe de simplement cloner le nôtre, sans passer par toute une phase de paramétrage. Cela pourra également servir de sauvegarde en cas de problèmes majeurs sur leurs versions modifiées. Cette même technique de partage de mémoire pourra être utilisée afin de partager les avancées sur le projet au sein d'une même équipe, vérifiant ainsi que chaque membre puisse travailler séparément et simultanément sur un cluster de machines virtuelles strictement identique.

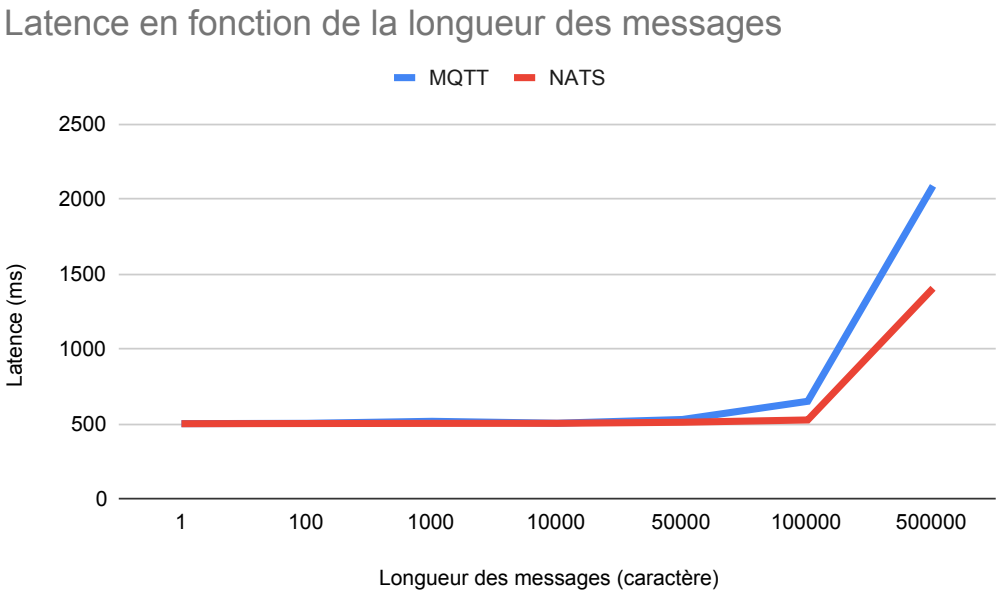
4.3 Améliorations diverses

4.3.1 Utilisation de MQTT

Lors du rapport de spécification, nous avons expliqué que nous allions remplacer le protocole MQTT par NATS[11], pour gagner en efficacité. Cependant, après une étude comparative plus poussée, nous nous sommes rendu compte que la technologie apportait un gain en efficacité quasiment négligeable. Cette étude comparative a necessité la création d'un benchmark. Deux programmes identiques à l'exception du protocole utilisé (MQTT ou NATS) ont été créé pour envoyer et recevoir des messages. Le débit moyen, la latence et le taux de messages reçus a ensuite pu être mesuré. Les résultats sont les suivants :







Concernant le débit en Nous voyons donc que les résultats ne montrent pas une différence significative entre les deux protocoles dans le cadre de notre projet.

MQTT étant déjà présent dans certaines technologies que nous utilisons (Chirpstack par exemple), et MQTT étant le protocole de référence utilisé dans ce domaine, nous avons décidé de rester sur ce protocole pour effectuer nos communications entre les fonctions au sein du cluster.

4.3.2 Tailscale

Dans le cadre de notre projet, nous avons déployé Tailscale[12] sur le nœud 0 du cluster afin de simplifier l'accès à distance via SSH. Tailscale crée un réseau privé virtuel (VPN) entre les appareils, permettant ainsi une connexion sécurisée et transparente, même à travers des réseaux non sécurisés. Grâce à cette configuration, nous pouvons accéder à notre cluster depuis n'importe où sans avoir à configurer des règles complexes de pare-feu ou de routage.

5 Conclusion

Ce document a présenté plus concrètement notre travail au cours de cette année dans le cadre du projet Mycélium v3. Nous avons présenté en détail l'architecture globale du projet ainsi que les ajouts que l'équipe de cette année apporte au projet.

Comme vous aurez pu le constater, l'expérimentation nous a fait revenir sur certains de nos objectifs, par exemple la forme du guide que nous voulions réaliser, ou le fait d'utiliser MQTT plutôt que NATS. Concernant la planification, nous avançons à peu près au rythme souhaité, en rencontrant régulièrement des phases de blocage, mais qui sont caractéristiques de tout projet informatique.

Nous essayons également de réfléchir, avec nos partenaires de l'OSUR, à de potentielles améliorations à apporter au projet à plus long terme, par exemple le fait d'utiliser l'API de Météo-France afin de récupérer leurs données et les utiliser pour les comparer aux nôtres ou encore trouver un moyen d'envoyer des messages aux capteurs, afin que notre analyse automatique puisse enclencher chez eux un changement de comportement. Une des applications pratiques de cette action pourrait être de prendre plus de relevés en cas d'évènement particulier.

Table des figures

6 Bibliographie

Références

[1] I-Site Clermont Auvergne, CNRS. SoLo : nœud communicant LoRaWAN https://www.google.com/url?sa=t&rct=j&q=&esrc=s&source=web&cd=&ved=2ahUKEwjg_qWMvL-CAxWgcKQEHQtrDp0QFnoECBYQAQ&url=https%3A%2F%2Findico.in2p3.fr%2Fevent%2F23490%2Fcontributions%2F91608%2Fattachments%2F62480%2F85614%2FPr%25C3%25A9sentation_logiciel_SoLo_fev21.pptx&usg=AOvVaw0tkdzpCRGQGBHEvQ2WC1do&opi=89978449

[2] L. Vangelista, "Frequency Shift Chirp Modulation : The LoRa Modulation," in IEEE Signal Processing Letters, vol. 24, no. 12, pp. 1818-1821, Dec. 2017, doi : 10.1109/LSP.2017.2762960.

[3] Site de Raspberry Pi. https://www.raspberrypi.com/

[4] Site de Chirpstack. https://www.chirpstack.io/

[5] Site de MQTT. https://mqtt.org/

[6] Site de K3S. https://k3s.io/

[7] Site d'OpenFaaS. https://www.openfaas.com/

[8] Site d'InfluxDB. https://www.influxdata.com/

[9] Site de Grafana. https://grafana.com/

[10] Site de QEMU. https://www.qemu.org/

[11] Site de NATS. https://nats.io/

[12] Site de Tailscale. https://tailscale.com/