ECOLE
POLYTECHNIQUE
DE BRUXELLES

ULB

UNIVERSITÉ LIBRE DE BRUXELLES

Deuxième ligne de titre du mémoire

Ligne du sous-titre du mémoire

Mémoire présenté en vue de l'obtention du diplôme
d'Ingénieur Civil en informatique à finalité spécialisée

João Marques Correia

Directeur
Professeur François Quitin

Co-Promoteur
Professeur Antoine Nonclercq

Superviseur
Quentin Delhaye

Service
BEAMS

Année académique
2019 - 2020

Contents

1 Introduction **4**
- 1.1 Contexte du mémoire . . . 4
- 1.2 Parties prenantes . . . 4
- 1.3 Projet précédent . . . 4
- 1.4 Structure du mémoire . . . 4

2 État de l'art **5**
- 2.1 Technologies de communication à distance . . . 5
- 2.2 Monitoring de l'état d'un serveur linux . . . 8
- 2.3 Plateformes de visualisation de données . . . 11

3 Cahier des charges **13**
- 3.1 Caractéristiques matérielles du dispositif . . . 14
- 3.2 Fonctionnalités à implémenter . . . 14
- 3.3 Priorité des tâches et méthodologie de travail . . . 15

4 Prototype **16**
- 4.1 Le premier prototype (2018-2019) . . . 16
  - 4.1.1 Problèmes détectés . . . 17
- 4.2 Nouveau Prototype . . . 18
  - 4.2.1 Solutions considérées . . . 18
  - 4.2.2 Thingstream . . . 18
  - 4.2.3 Résultat . . . 18
  - 4.2.4 Tests réalisés . . . 18

5 L'application **19**
- 5.1 Client . . . 19
  - 5.1.1 Architecture . . . 19
  - 5.1.2 Base de données . . . 19
  - 5.1.3 Implémentation des fonctionnalités . . . 19
  - 5.1.4 Sécurité . . . 19
- 5.2 Serveur . . . 20
  - 5.2.1 Architecture . . . 20
  - 5.2.2 Base de données . . . 20
  - 5.2.3 Implémentation des fonctionnalités . . . 20
  - 5.2.4 Sécurité . . . 20

6 Tests et résultats **21**

7 Conclusion **22**

A Tableau comparatif des réseaux **25**

Chapter 1

Introduction

1.1 Contexte du mémoire

1.2 Parties prenantes

1.3 Projet précédent

1.4 Structure du mémoire

Chapter 2

État de l'art

2.1 Technologies de communication à distance

Lors des dernières années, le nombre de dispositifs Internet des Objets (IoT) connaît une croissance fulgurante qui se maintiendra au cours des années qui suivent (voir figure 2.1). Ces objets sont capables d'acquérir des données sur leur environnement et/ou de prendre des actions sur celui-ci. Ils communiquent avec d'autres machines pour transmettre les informations acquises et recevoir des commandes lorsque cela est nécessaire. Les dispositifs IoT ont de diverses applications telles que les thermostats intelligents, les voitures connectées, suivi et monitorage d'actifs, et bien d'autres. Selon les conditions d'utilisation et son objectif, ces dispositifs doivent être capables d'envoyer des données sur des courtes ou des longues distances. En outre, leur consommation énergétique doit généralement rester faible, notamment lorsqu'ils sont alimentés par une batterie.

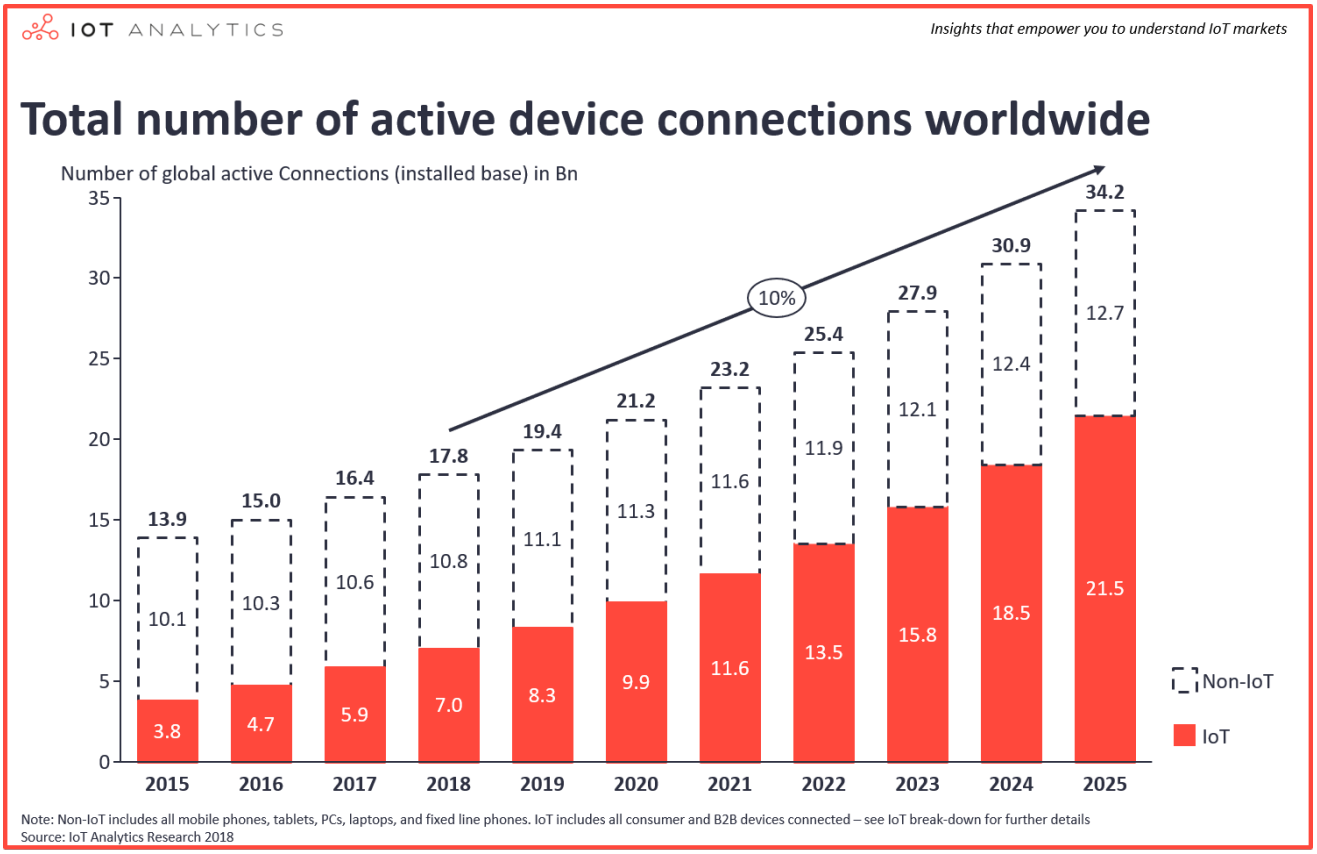

Figure 2.1: Nombre de dispositifs connectés à Internet [1]

Un réseau de communication unique et employable indépendamment du contexte du projet n'existe pas [2]. Toutefois, une multitude de réseaux avec des caractéristiques distinctes sont disponibles, tels que LoRaWAN et GSM. La figure 1 présente l'architecture simplifiée des réseaux utilisés par les dispositifs IoT pour échanger des informations avec des serveurs ou des utilisateurs qui sont connectés à Internet. Nous ne nous intéressons ici qu'aux technologies responsables de la transmission de données dans la phase 1 de la figure.

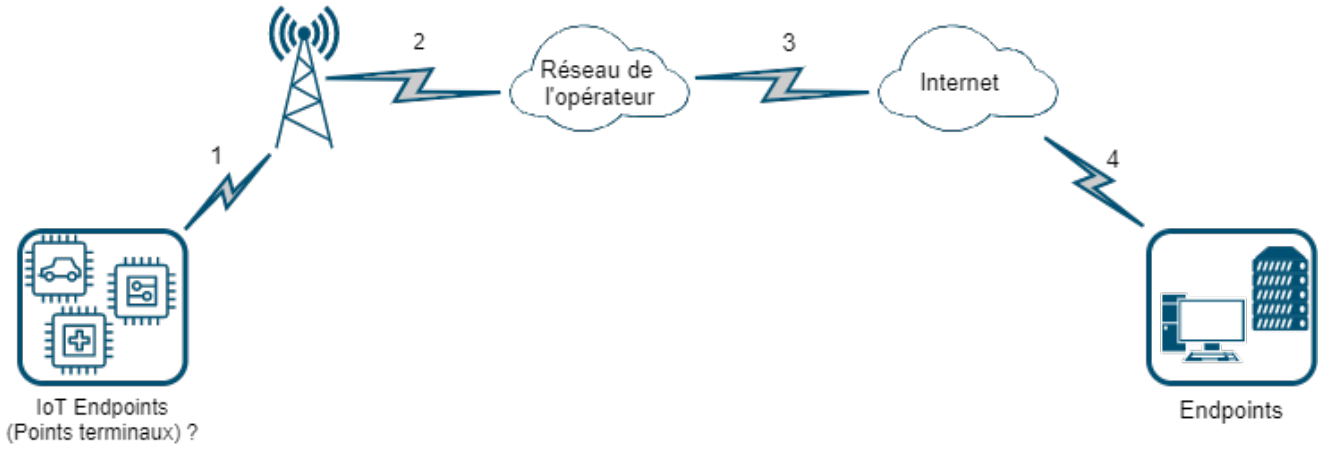

Figure 2.2: Architecture simplifiée d'un réseau avec dispositifs IoT (Basé sur [3, 4])

Actuellement, les principaux réseaux sans fil disponibles pour les dispositifs IoT sont :

- LoRaWAN
- Réseaux mobiles (2G/3G/4G/5G)
- Sigfox
- Satellite
- Wi-Fi
- Zigbee
- Bluetooth
- Réseaux mobiles IoT (NB-IoT et LTE-M)

Ces technologies utilisent toutes des ondes électromagnétiques pour transmettre les données, mais les fréquences sur lesquelles elles opèrent sont parfois très différentes. Certaines technologies utilisent une implémentation propriétaire, d'autres se basent sur des standards open source. [5] De façon similaire, certaines technologies exploitent la bande ISM[1], alors que d'autres exploitent des fréquences non réglementées. Ces dernières permettent de déployer son propre réseau privé, à condition d'acheter et configurer tout le matériel requis ce qui peut demander un investissement initial assez important. Ces différences accordent des propriétés distinctes aux réseaux en ce qui concerne le coût d'utilisation et déploiement, la bande passante, et la portée du signal. À

[1]Bande industrielle, scientifique et médicale

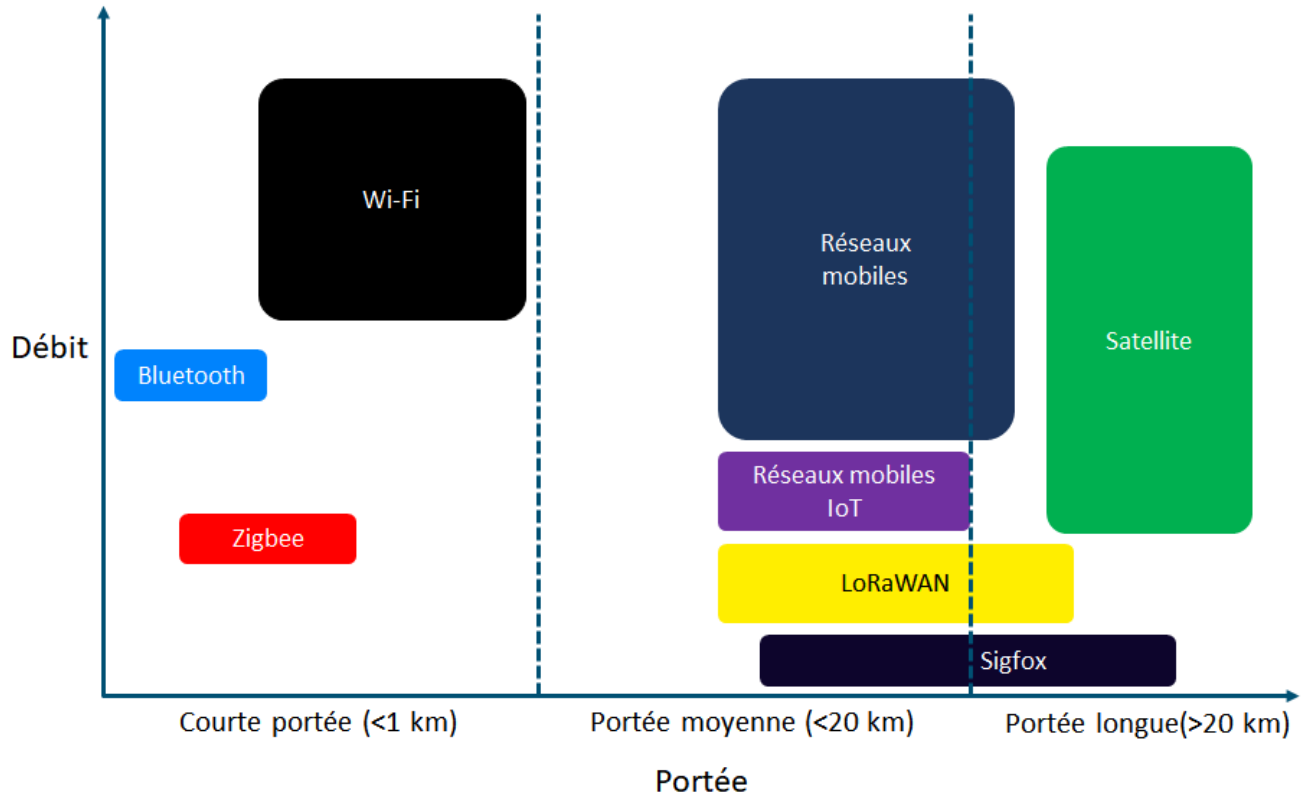

Figure 2.3: Classement des réseaux en fonction de leur portée et débit (Basé sur)

noter également, certains réseaux imposent une taille maximale sur chaque message, qu'il soit envoyé ou reçu. Le tableau comparatif de l'annexe A offre une vue d'ensemble sur les différentes technologies et leurs propriétés. Il faut cependant remarquer que la portée et le débit sont juste donnés à titre comparatif et ils sont à prendre avec des pincettes. En effet, ces valeurs varient beaucoup en fonction de certains paramètres tels que la géographie du milieu (urbain ou rural) et des éventuelles interférences. Néanmoins, le débit et la portée sont très importants pour classifier ces réseaux et, avec l'aide de la figure 2.3, ces technologies peuvent être classées de la façon suivante [6] :

- IoT haut débit : Dans cette catégorie se trouvent principalement les technologies comme les réseaux mobiles, le Wi-Fi et le satellite. Les débits proposés sont généralement supérieurs au Mb/s. En contrepartie, l'utilisation de ces technologies engendre une consommation énergétique plus élevée.
- IoT bas débit : Ici se trouvent les réseaux mobiles IoT, LoRaWAN, Sigfox, et Zigbee. En fonction du réseau, le débit peut varier entre quelques bits par seconde jusqu'à un Mb/s. Ces technologies offrent une consommation énergétique faible.
- IoT critique : C'est-à-dire lorsqu'il s'avère nécessaire de transmettre des données sur un intervalle de temps donné [6]. C'est une propriété des réseaux 5G.

Un point également important est que ces technologies ont de différents niveaux de maturité. Au moins un réseau mobile est disponible dans chaque pays, mais les nouveaux réseaux comme Sigfox, LoRaWAN, et 5G sont en revanche toujours en cours de déploiement. De ce fait, ces dernières technologies ne sont disponibles que dans certaines régions possédant une infrastructure moderne. La couverture du réseau est un critère important à prendre en compte avant de faire un choix

afin d'éviter les mauvaises surprises. Finalement, toujours dues à ce différent degré de maturité, certaines technologies plus anciennes pourraient voir la fin de ses jours dans les années à venir. Par exemple, certains opérateurs vont décommissionner leur réseau 2G à partir de 2020 [7]. Les opérateurs restants devraient suivre la même tendance dans les années suivantes. De plus, le réseau 3G devrait suivre la même tendance comme affichée sur la figure 2.4.

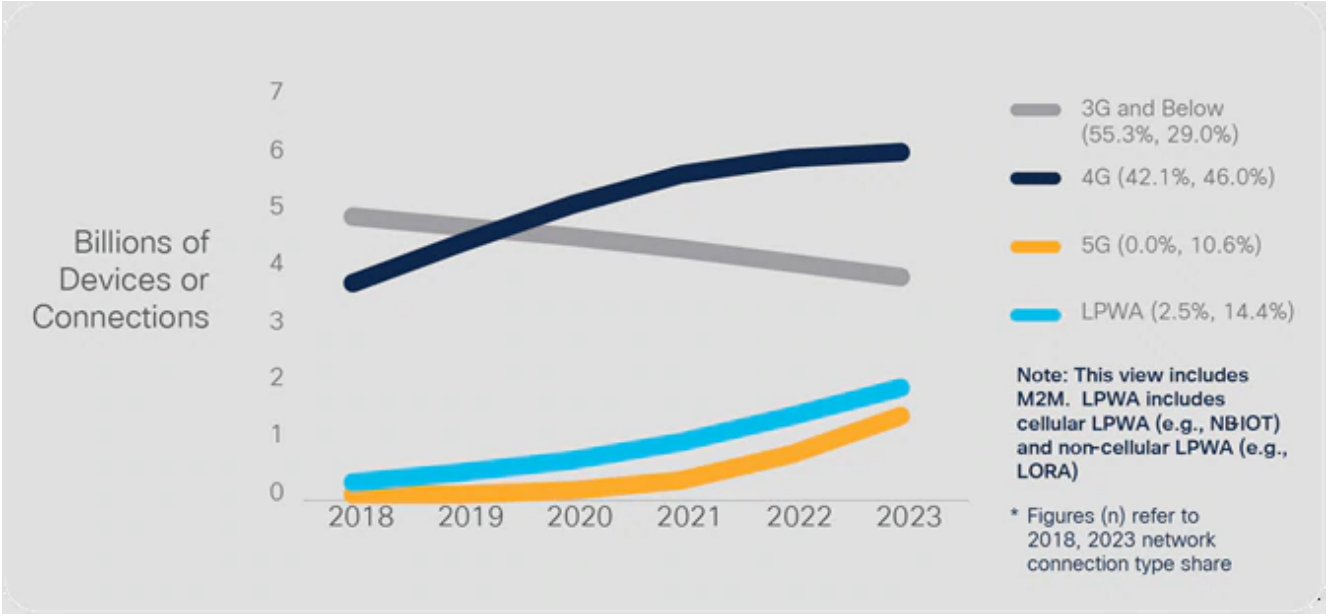

Figure 2.4: Évolution du nombre de dispositifs connectés aux différents réseaux [8]

2.2 Monitoring de l'état d'un serveur linux

La nécessité de surveiller toute l'infrastructure IT, et en particulier des serveurs tournant sous Linux, existe depuis un bon nombre d'années. Cependant, surveiller un système n'est pas trivial. Cela peut être accompli de différentes manières en fonction des évènements ou statistiques qui doivent être observées dans la machine hôte. Par exemple, l'utilisation de logs pour débugger une application ou voir l'historique de transactions d'une base de données est une technique de monitoring utilisée depuis de nombreuses années. D'autres techniques de monitoring existantes sont le profiling, le tracing et l'acquisition de métriques [9]. Dans cette section, seulement les technologies d'acquisition de métriques sont présentées puisqu'elles sont les plus utiles pour connaître l'état de la machine.

Créé à la fin des années 80, le protocole SNMP [10, 11] permet de monitorer et configurer des dispositifs différents connectés au réseau. Tout au long des années, des solutions de monitorage de métriques de plus haut niveau, ou tout-en-un, sont apparues sur le marché. Ces solutions utilisent différents protocoles, dont SNMP, pour surveiller les dispositifs sur le réseau et stockent toutes les informations recueillies sur une base de données. De plus, elles permettent de définir un ensemble de conditions à surveiller de plus près et si une de ces conditions est violée, une alerte est transmise vers l'équipe responsable de l'infrastructure. Deux exemples de solutions créées à la fin des années 90 et qui demeurent disponibles sont Nagios et Zabbix.

Actuellement, un grand nombre de solutions de monitoring sont disponibles sur le marché proposant différents modèles commerciaux. En effet, certaines solutions sont vendues comme un produit, mais d'autres préfèrent suivre un modèle open source. Dans le cas de ces dernières, les

entreprises responsables de l'application proposent souvent leurs services en matière d'assistance technique afin d'installer et maintenir la solution. Un autre modèle aussi appliqué est de proposer une version gratuite avec un nombre limité de fonctionnalités, et une version payante débloquant toutes les capacités de l'application. Avant d'entamer toute recherche d'une solution, il faut comprendre quelle formule s'adapte le mieux aux besoins de l'utilisateur. Comme il est affiché sur la figure 2.5, deux architectures existent pour les solutions de monitoring : *pull* et *push* [9, 12, 13, 12]. Chaque application est entièrement basée sur une de ces deux approches. Dans certains cas, une solution peut parfois supporter les deux architectures (voir exemple figure 2.6). Cependant, ces solutions détiennent toujours un modèle préféré, et l'autre possibilité sert juste à combler certaines lacunes du premier modèle puisque, en effet, chaque architecture possède ses avantages et faiblesses.

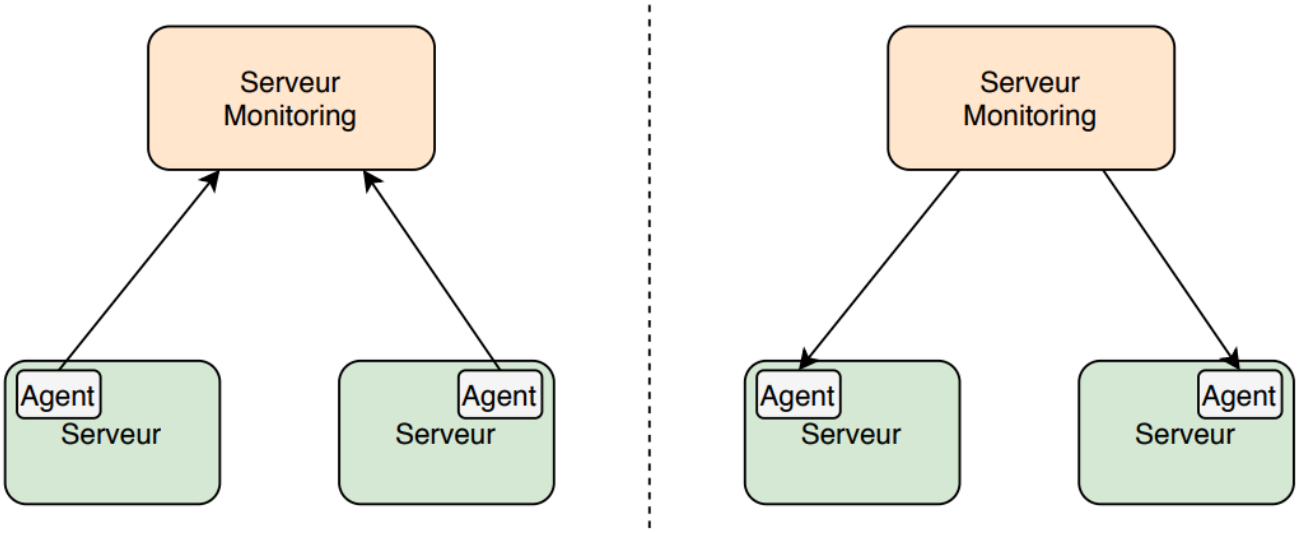

Figure 2.5: **Gauche :** Solution de monitoring avec architecture *push*. **Droite :** Solution de monitoring avec architecture *pull*.

Dans le modèle de type *push*, les agents dans la machine hôte envoient un ensemble de métriques ou des événements au serveur de monitoring. Cette technique permet notamment de définir des conditions d'alerte dans l'agent, et c'est celui-ci qui se charge de vérifier ces conditions. Lorsque le test des conditions passe, l'agent transmet un événement au serveur de monitoring. De plus, seul le système de push est capable de correctement acquérir des données à propos de jobs Batch de courte durée. [13, 14] En effet, comme affiché sur la figure 2.7, le modèle *pull* peut rater la fin du job ce qui se traduit par une perte d'informations.

Dans le modèle de type *pull*, les agents dans la machine hôte collectent et exposent les métriques, mais c'est au serveur de monitoring d'activement aller retrouver ces données. Par conséquent, les événements et alertes au niveau des agents ne sont pas pris en charge, dans la mesure où ces derniers ne savent pas quand le système de monitoring récupérera les informations. En revanche, cette méthode permet de mieux identifier si un problème s'est produit sur la machine hôte puisqu'elle ne répondra plus aux requêtes lorsque cela a lieu.[15] Ceci rend l'approche *pull* légèrement plus fiable que la technique *push*. Les deux méthodes possèdent donc leurs avantages et inconvénients qu'il faut tenir en compte lors du choix de la technologie. Cependant, dans la majorité des cas, les deux architectures offrent ces capacités très similaires [16].

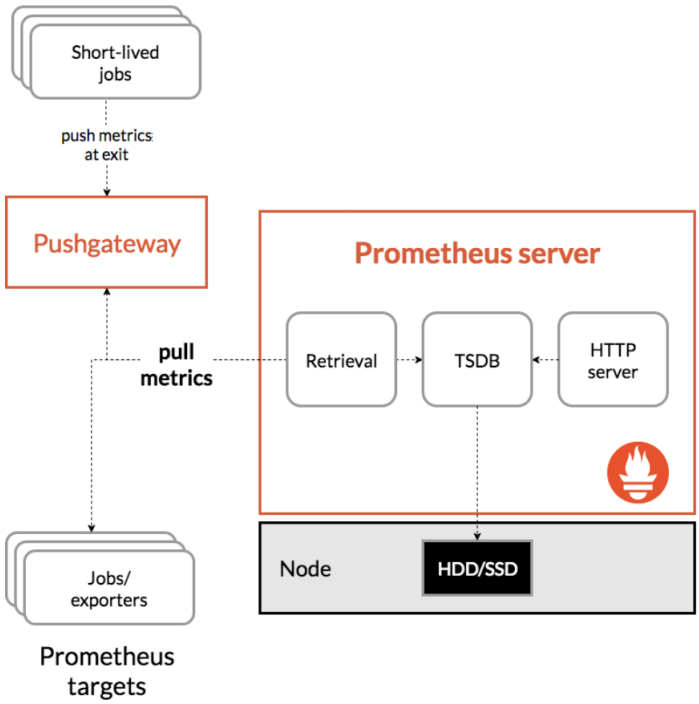

Figure 2.6: Architecture *pull* de Prometheus. Le modèle *push* est supporté grâce au Pushgateway.

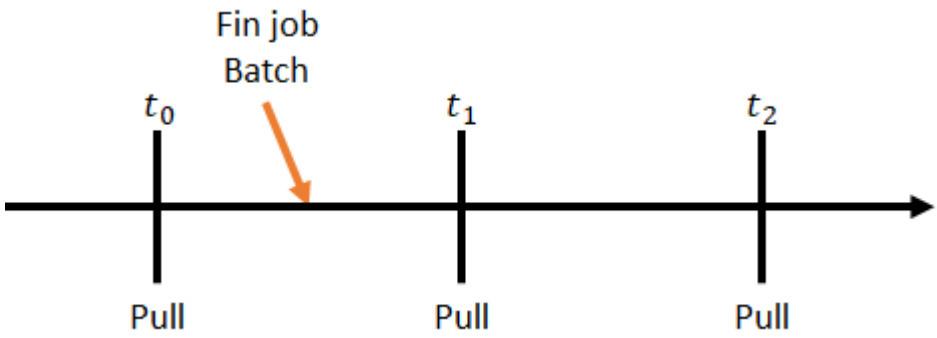

Figure 2.7: Faiblesse du modèle *pull*. Le système de monitoring a raté les informations de la fin du job

Indépendamment de l'architecture utilisée, toutes les informations recueillies ou reçues par le serveur de monitoring doivent être stockées sur une base de données. Les solutions plus anciennes comme Nagios et Zabbix, utilisent principalement des bases de données relationnelles [17, 18]. Ceci n'est plus le cas pour les solutions plus récentes telles que Prometheus et le TIG Stack[2] qui utilisent des bases de données orientées séries temporelles pour la persistance des données. Cette sorte de base de données, comme son nom indique, est mieux optimisée pour des valeurs horodatées [19]. En effet, ces bases de données possèdent les propriétés suivantes [20]:

- Elles supportent des écritures simultanées et avec de grands débits. Typiquement, dans ces bases de données se produisent énormément d'écritures, mais pas beaucoup d'accès.
- Ces bases de données doivent stocker d'énormes quantités de données temporelles. Elles utilisent des algorithmes de compression spécifiques pour stocker les données de manière efficace. [21]

[2]Le TIG stack est la combinaison de Telegraf, InfluxDB et Grafana pour créer un outil de monitoring

- Toutes les requêtes sont effectuées sur base d'un intervalle dans le temps
- Les données ont une durée de rétention. À partir d'une certaine date, la base de données élimine automatiquement les informations car elles sont jugées non utiles.

De ce fait, les bases de données orientées séries temporelles offrent un stockage plus efficace pour les métriques surveillées. Un dernier aspect très important d'une solution de monitoring est la visualisation de données. La majorité des solutions offrent soit un outil de visualisation propriétaire (Nagios ou Zabbix), soit elles exploitent des outils existants tels que Grafana (Prometheus et le TIG Stack).

2.3 Plateformes de visualisation de données

Cette section n'est pas encore terminée.

La visualisation de données consiste à représenter plusieurs formats de données sous un format visuel tel que les graphiques. Le système visuel humain est capable d'identifier des tendances ou des aberrations. Le but de la visualisation des données consiste alors dans l'aide de la compréhension des données en exploitant cette capacité. [22]

Les outils de visualisations de données sont alors des solutions software qui aident à accomplir cet objectif. [23] Ces outils fournissent une plateforme très puissante pour communiquer de l'information. Cependant, le concept de données est trop abstrait. En effet, elles peuvent être des coordonnées spatiales, les mesures de température d'une pièce, le nombre d'occurrences d'un évènement, le classement d'une course automobile, etc. Pour chacun de ces différents types de données, la méthode de visualisation à utiliser est différente. Par exemple, une carte serait utilisée pour afficher les coordonnées, mais une courbe serait préférée pour afficher l'évolution de la température. L'article [22] propose différentes méthodes et graphiques pour observer les différents types de données existants.

Comme présenté dans [source], la visualisation de données doit respecter les trois principes suivants : Thrustworthy, accessible, and elegant

Avec l'arrivée de l'ère du Big Data, les technologies de visualisation sont devenues d'autant plus importantes. En effet, la quantité de données est devenue tellement importante au point qu'un humain n'est plus capable d'analyser toutes les données. Ceci coïncide avec l'apparition de nouvelles technologies capables de mieux guider l'utilisateur à trouver des tendances dans les données. Actuellement, trois grandes catégories d'outils de visualisations de données sont disponibles :

- Les librairies software, telles que Matplotlib ou D3.js, qui aident les utilisateurs à créer de différentes visualisations de données. Ces outils demandent des compétences en programmation.
- Ensuite, il y a les outils purement pour la visualisation de données tels que Grafana ou Google Data Studio. À partir d'une interface web, ces outils permettent de facilement créer des graphiques en utilisant seulement des langages de requête comme SQL. Certains de ces outils sont capables d'auxilier l'utilisateur à formuler des requêtes. (Query Builders)

- Finalement, il y a les outils de Business Intelligence. Ils ont les mêmes fonctionnalités qu'un outil de visualisation de données. Cependant, ils permettent en plus de cela de faire un traitement des données et des prédictions sur celles-ci. Un exemple d'un tel outil est Tableau ou Power BI.

Chapter 3

Cahier des charges

Le but de ce mémoire consiste à développer un prototype capable de surveiller l'infrastructure de CERHIS et s'assurer du bon fonctionnement de celle-ci. De plus, ce système de monitoring doit pouvoir avertir un technicien lorsqu'un problème est détecté. Bien évidemment, pour accomplir cet objectif, le système devra acquérir des données sur le fonctionnement des différents dispositifs de CERHIS. Ces données devront être sauvegardées localement, mais aussi sur un serveur distant. Une interface web devra également être présente pour faciliter la visualisation de ces données, que cela soit dans le périmètre du centre hospitalier ou partout dans le monde. La figure 3.1 reprend l'architecture simplifiée de ce système de monitoring.

Pour simplifier la division du travail, la création du dispositif a été scindée en deux parties. Une première partie majoritairement *hardware* comportant le choix et l'assemblage de tous les composants nécessaires au développement du prototype. Cela englobe l'ordinateur requis pour tourner le logiciel, l'alimentation, la solution de communication et le software qui permet aux différents composants de s'interfacer. Ensuite, une deuxième partie comprenant juste le *software* nécessaire pour effectuer les tâches de monitoring. Cela comprend les fonctionnalités requises pour l'acquisition, traitement, stockage et visualisation des données. La présentation des spécificités de chaque partie se fera dans les sections suivantes (sections 3.1 et 3.2).

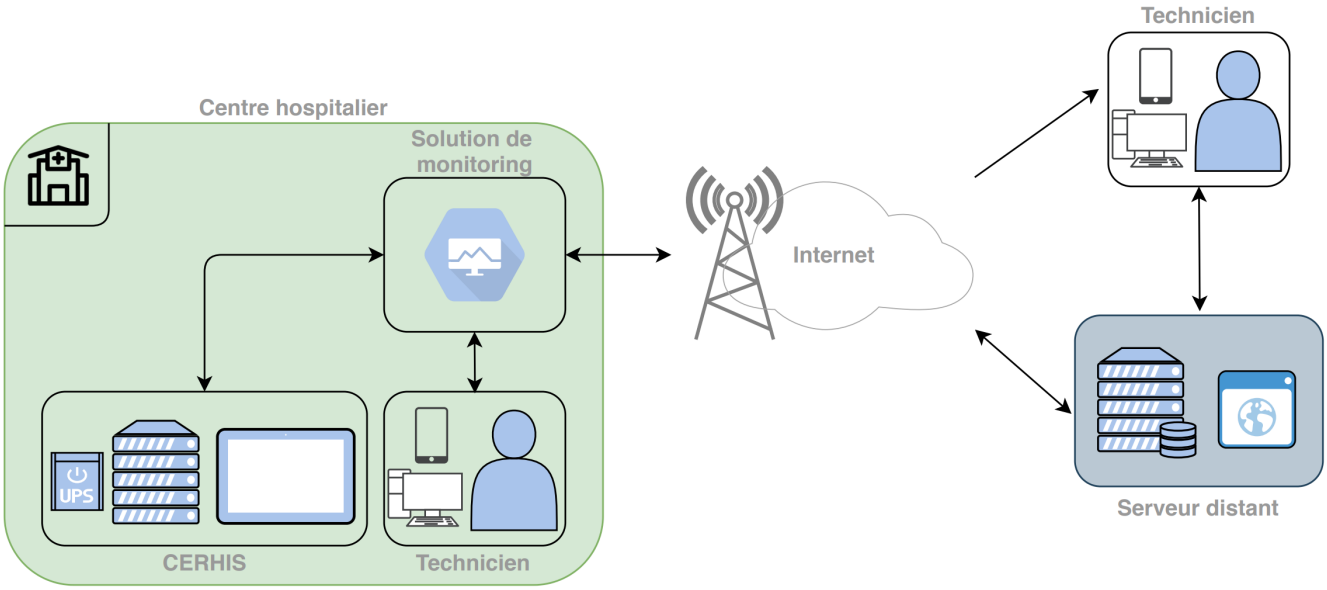

Figure 3.1: Architecture simplifiée de la solution de monitoring

3.1 Caractéristiques matérielles du dispositif

Étant donné que le dispositif de monitoring sera utilisé dans des centres hospitaliers en Afrique centrale, il doit être capable de résister à certaines caractéristiques propres du milieu telles que la température. Voici la liste exhaustive de toutes les caractéristiques matérielles que le produit doit posséder :

- Le prototype doit être robuste. Mes composants électroniques doivent notamment être capables de supporter des températures ambiantes supérieures à 30 °C pendant de longues périodes. [24]
- Le dispositif doit être capable de communiquer à distance, même lorsqu'une connexion à internet filaire n'est pas disponible. De ce fait, le prototype doit être capable de tirer parti d'une des technologies présentées dans la section 2 pour pouvoir envoyer des informations et des alertes.
- Le dispositif doit être d'apparence discrète, évitant d'attirer l'attention sur sa valeur.
- Le coût doit rester abordable et réaliste.
- Le dispositif doit posséder un système d'alimentation mixte capable d'alimenter le prototype pendant un ou deux jours en cas de passe du service électrique.
- Le boîtier doit être fermé hermétiquement pour empêcher des insectes d'y faire son nid.

3.2 Fonctionnalités à implémenter

Pour la section *software* du dispositif, les fonctionnalités à implémenter peuvent encore être divisées en deux sous-sections : la solution de monitoring tournant sur le site, et le serveur distant capable de recevoir, stocker et afficher des données. Les fonctionnalités présentées ci-dessous seront alors divisées en suivant ces deux sous-sections.

- **Solution de monitoring dans le centre hospitalier**
  - Effectuer un mappage du réseau local de façon à trouver tous les appareils qui y sont connectés.
  - Monitorer la présence des appareils connectés au réseau local. Si un appareil se déconnecte du réseau, pouvoir indiquer combien de temps est passé depuis sa dernière connexion.
  - Monitorer l'état de fonctionnement du serveur de CERHIS.
  - Envoi d'alertes par SMS.
  - Envoie de données vers un serveur distant.
  - Interface utilisateur permettant de facilement visualiser toutes les données.
  - Vérifier le niveau de la batterie de CERHIS et l'état de charge des tablettes.
  - Créer un historique de connexion des tablettes au serveur de CERHIS
- **Serveur distant**
  - Serveur capable de stocker les données envoyées depuis plusieurs centres hospitaliers.
  - Interface utilisateur permettant de facilement visualiser toutes les données disponibles sur le serveur distant.

3.3 Priorité des tâches et méthodologie de travail

Les listes des fonctionnalités et caractéristiques présentes ci-dessus reprennent tout ce qui serait nécessaire pour avoir un produit fini. L'objectif de ce mémoire est bien évidemment d'essayer d'implémenter le maximum de fonctionnalités possibles. Cependant, le temps est limité et des imprévus peuvent survenir au long du chemin. De ce fait, les diverses fonctionnalités ont été classées par ordre de priorité afin de garantir que le prototype rendu à la fin de ce mémoire possède au moins un certain nombre de fonctionnalités.

En premier lieu, c'était la sélection du matériel requis pour implémenter les fonctionnalités. Cela reprend, le choix de la plateforme, l'alimentation et la solution de communication. Cette tâche reprend aussi des tests sur la plateforme du projet de l'année précédente afin de vérifier si des changements étaient nécessaires.

Deuxièmement, c'était l'implémentation de tout le code nécessaire pour lier les divers composants hardware. En particulier, l'implémentation du code essentiel pour mettre en marche la solution de communication. Ici s'ajoutent aussi les premiers essais afin de prouver le bon fonctionnement de la plateforme.

Une fois la plateforme préliminaire choisie et testée, la possibilité de commencer à développer les différentes fonctionnalités du logiciel s'est ouverte. À nouveau, en raison du grand nombre de fonctionnalités requises, les plus cruciales ont été sélectionnées lors de réunions avec les parties prenantes. La liste ci-dessous présente les fonctionnalités triées par ordre décroissant de priorité :

- Monitorer l'état du serveur, en particulier surveiller l'uptime de celui-ci et avertir un technicien en cas de modification anormale de la valeur.
- Mapper le réseau local.
- Surveiller la présence des dispositifs de CERHIS.
- Monitorer l'état de certains processus dans le serveur.
- Envoi des données vers un serveur distant.
- Interface web pour la visualisation des données

Finalement, les dernières tâches reprennent la finalisation d'un boîtier permettant de loger les différents composants électroniques et des tests approfondis pour s'assurer du bon fonctionnement et la fiabilité de la solution de monitoring.

Pour essayer d'être le plus efficace possible, la méthodologie de travail consistait à écrire toutes les tâches liées à ce projet sur des notes Post-it. Les Post-its orange représentaient les missions liées au software, et les jaunes correspondaient aux missions liées au hardware. Chaque note possédait aussi un numéro coïncidant avec la priorité de la tâche. Le critère de sélection était en premier lieu la couleur, orange prioritaire sur le jaune, et ensuite la priorité (lorsque les Post-its avaient tous la même couleur).

Chapter 4

Prototype

4.1 Le premier prototype (2018-2019)

Dans le cadre du projet d'année de la première année du Master ingénieur civil en informatique, un prototype capable de surveiller l'infrastructure de CERHIS a été réalisé. Ce dispositif s'inspirait sur d'autres prototypes plus anciens créés par des étudiants de l'école polytechnique de Bruxelles. L'architecture du dispositif produit l'année dernière est présentée sur la figure 4.1.

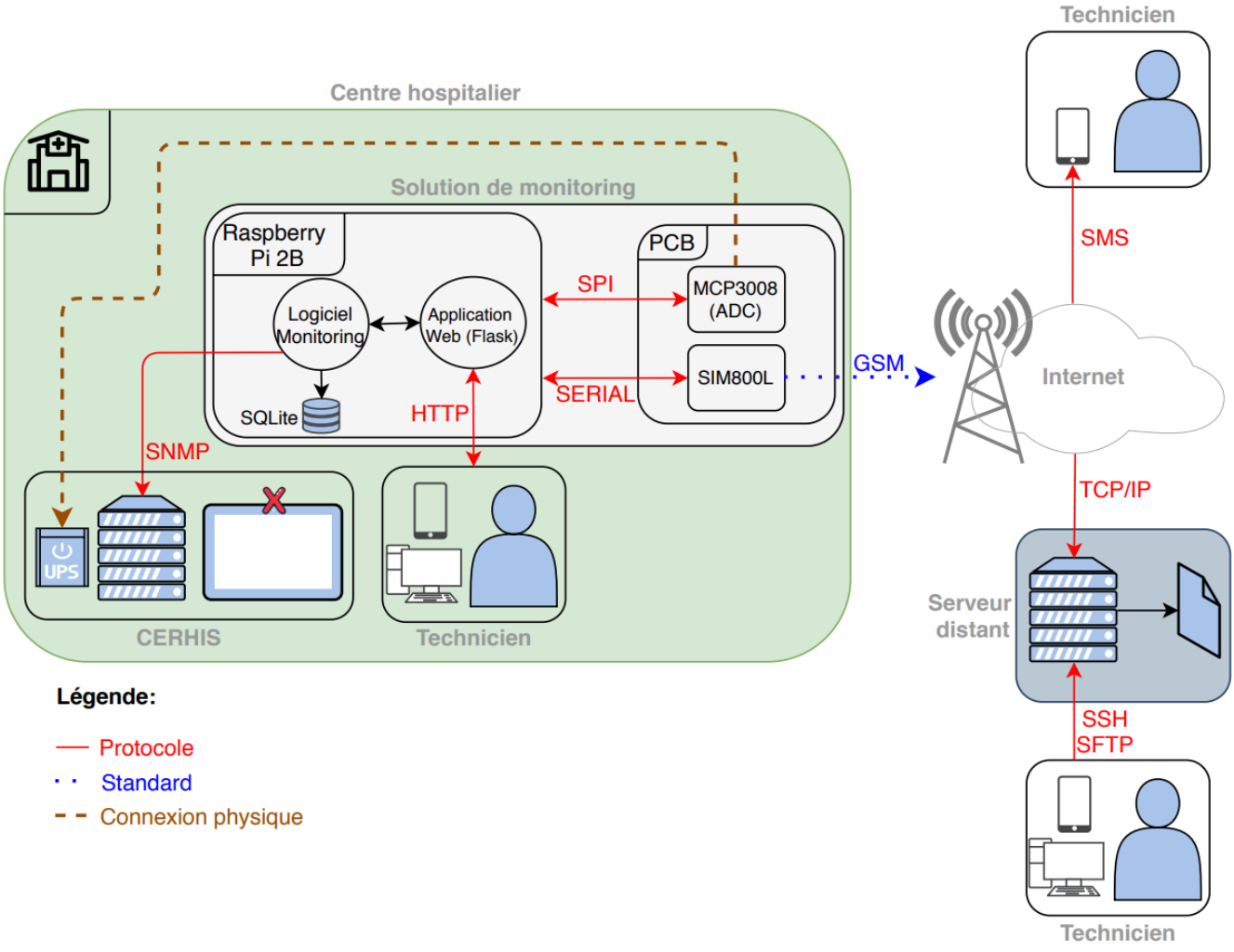

Figure 4.1: Architecture de la solution de monitoring développée lors de l'année académique 2018-2019

Un Raspberry Pi 2B était utilisé pour tourner le logiciel de monitoring ainsi que l'interface web permettant la configuration de celui-ci. Un circuit imprimé (PCB) avait été réalisé dans le but de loger des composants électroniques différents tels que le module de communication GSM SIM800L et le convertisseur analogique numérique MCP3008. Ce même circuit imprimé s'insérait sur les ports GPIO du Raspberry Pi afin que ce dernier puisse interfacer avec les différents composants présents sur le PCB. Grâce au SIM800L, le réseau GSM d'un opérateur mobile local pouvait être utilisé pour envoyer des SMS vers un technicien, ou des informations quelques conques vers un serveur distant. Dans ce premier essai, ce serveur était juste capable de recevoir les données envoyées par le prototype et de les stocker sur un fichier *.txt*. Aucun traitement ou visualisation des données n'était offert aux techniciens. De plus, le monitoring des tablettes n'a pas pu être implémenté par manque de temps.

De façon à alimenter le dispositif, un système d'alimentation mixte avait été mis en place. Comme le montre la figure 4.2, ce système était basé sur une batterie acide-plomb de 12V, un régulateur de charge permettant de recharger la batterie et un convertisseur step-down transformant la tension de 12 à 5 volts.

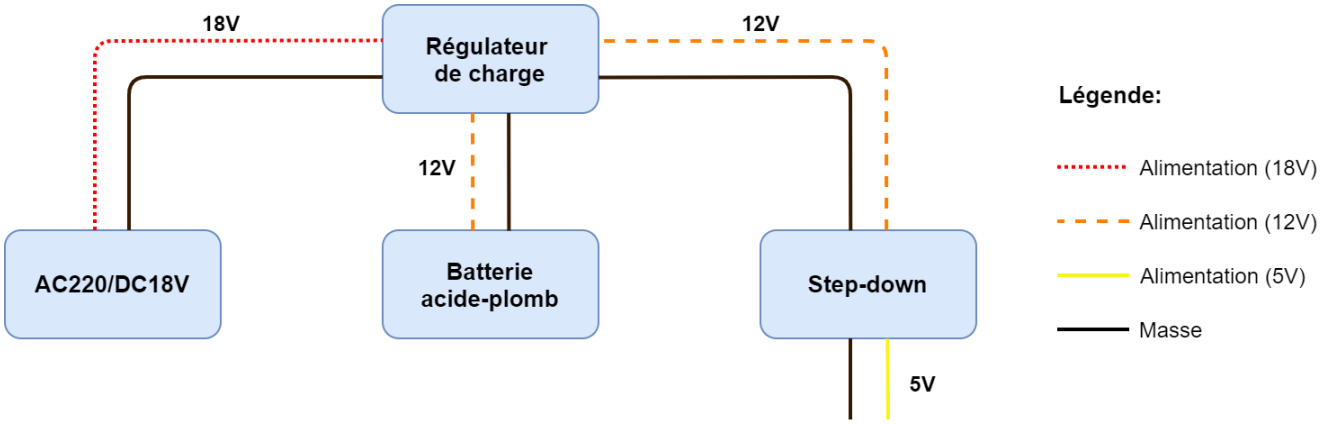

Figure 4.2: Système d'alimentation de la solution de monitoring développée lors de l'année académique 2018-2019

Ce prototype a été testé pendant plusieurs semaines[1] à Goma par un étudiant de l'ISIG[2]. Lors de cette période de tests, plusieurs problèmes ont été identifiés. Ils seront présentés dans la section suivante.

4.1.1 Problèmes détectés

Plusieurs protocoles de tests avaient été mis en place l'année dernière dans le but d'évaluer ce premier dispositif. Les soucis qui ont été trouvés lors de ces tests peuvent être classés selon trois grandes catégories : communication, alimentation et température. Tous les problèmes sont documentés ci-dessous.

- **Communication :**
  - Corruption de certaines données échangées entre le SIM800L et le Raspberry Pi. Des bits semblaient être modifiés lors de la transmission des messages entre les deux composants. Cela provoquait une perte d'informations. De plus, cela rendait plus difficile

[1] entre le 14 mars et le 10 avril 2019
[2] Institut supérieur d'informatique et de gestion de Goma

l'utilisation du SIM800L, car ce dernier n'était pas capable de traiter toutes les commandes qui lui étaient envoyées.

  - Implémenter toutes les fonctionnalités de communication requises avec le SIM800L était très laborieux. En effet, ce module peut être vu comme un automate fini déterministe. Des commandes AT sont utilisées pour contrôler le module, c'est-à-dire qu'elles sont employées pour changer l'état de celui-ci. Une succession correcte de multiples commandes est nécessaire afin d'accomplir une simple tâche telle que l'envoi d'un SMS. Ceci rendait le procès d'implémentation assez long, en particulier l'implémentation de la gestion des erreurs. Le problème mentionné ci-dessus a encore plus aggravé la complexité de la manipulation de ce module.

- **Alimentation :**

  - Le régulateur de charge initialement utilisé est tombé en panne lors de la réalisation des tests. Un simple remplacement de cette pièce semble avoir résolu le problème.
  - Le transformateur step-down est tombé en panne lors des tests. Malheureusement, aucun remplacement pour cette pièce a été trouvé à Goma. Ce souci est possiblement lié au problème de température qui sera présenté ci-dessous. Cependant, n'ayant pas assez d'information pour prouver l'existence d'un lien de causalité, chaque problème doit être considéré séparément.

- **Température :**

  - D'après les personnes en charge de tester le prototype, celui-ci semblait atteindre de hautes températures. Des températures élevées peuvent avoir un impact considérable sur la durée de vie des différents composants électroniques.

4.2 Nouveau Prototype

4.2.1 Solutions considérées

4.2.2 Thingstream

4.2.3 Résultat

4.2.4 Tests réalisés

Chapter 5

L'application

5.1 Client

5.1.1 Architecture

5.1.2 Base de données

5.1.3 Implémentation des fonctionnalités

Mappage du réseau local

Surveiller la présence des divers composants

Monitoring du serveur

Envoie des données

5.1.4 Sécurité

5.2 Serveur

5.2.1 Architecture

5.2.2 Base de données

5.2.3 Implémentation des fonctionnalités

5.2.4 Sécurité

Chapter 6

Tests et résultats

Chapter 7

Conclusion

Bibliography

[1] Knud Lasse Lueth. State of the iot 2018: Number of iot devices now at 7b – market accelerating. https://iot-analytics.com/state-of-the-iot-update-q1-q2-2018-number-of-iot-devices-now-7b/, Aug 2018. [Accédé le 28-Avril-2020].

[2] Frederic Vannieuwenborg, Sofie Verbrugge, and Didier Colle. Choosing iot-connectivity? a guiding methodology based on functional characteristics and economic considerations. *Transactions on Emerging Telecommunications Technologies*, 29, 10 2017.

[3] Ramon Sanchez-Iborra and Maria-Dolores Cano. State of the art in lp-wan solutions for industrial iot services. *Sensors*, 16(5):708, 2016.

[4] Kais Mekki, Eddy Bajic, Frederic Chaxel, and Fernand Meyer. Overview of cellular lpwan technologies for iot deployment: Sigfox, lorawan, and nb-iot. In *2018 IEEE International Conference on Pervasive Computing and Communications Workshops (PerCom Workshops)*, pages 197–202. IEEE, 2018.

[5] Brandon Foubert and Nathalie Mitton. Long-range wireless radio technologies: A survey. *Future Internet*, 12:13, 01 2020.

[6] David Feugey. Comment la 5g part à l'assaut de l'internet des objets. https://www.orange-business.com/fr/blogs/usages-dentreprise/mobilite/comment-la-5g-part-l-assaut-de-l-internet-des-objets, Dec 2016. [Accédé le 28-Avril-2020].

[7] Renouvellement du réseau mobile. https://www.swisscom.ch/fr/about/entreprise/portrait/reseau/remplacement-2g.html. [Accédé le 28-Avril-2020].

[8] Cisco. Cisco annual internet report (2018–2023). Technical report, Cisco, 03 2020.

[9] B. Brazil. *Prometheus - Up and Running: Infrastructure and Application Performance Monitoring*. O'Reilly Media, 2018.

[10] Jeffrey D. Case, Mark Fedor, Martin Lee Schoffstall, and Chuck Davin. Simple network management protocol (snmp). RFC 1098, RFC Editor, April 1989. http://www.rfc-editor.org/rfc/rfc1098.txt.

[11] Jeffrey D. Case, Mark Fedor, Martin Lee Schoffstall, and James R. Davin. Simple network management protocol (snmp). STD 15, RFC Editor, May 1990. http://www.rfc-editor.org/rfc/rfc1157.txt.

[12] Tech hub guides - monitoring at scale - monitoring architecture. https://developer.lightbend.com/guides/monitoring-at-scale/monitoring-architecture/architecture.html. [Accédé le 29-Avril-2020].

[13] Giedrius Statkevičius. Push vs. pull in monitoring systems. https://giedrius.blog/2019/05/11/push-vs-pull-in-monitoring-systems/, May 2019. [Accédé le 29-Avril-2020].

[14] schkn. Prometheus monitoring : The definitive guide in 2019. https://devconnected.com/the-definitive-guide-to-prometheus-in-2019/, May 2019. [Accédé le 29-Avril-2020].

[15] Faq: Why do you pull rather than push? https://prometheus.io/docs/introduction/faq/#why-do-you-pull-rather-than-push? [Accédé le 29-Avril-2020].

[16] Kiran Oliver. Prometheus and the debate over 'push' versus 'pull' monitoring. https://thenewstack.io/exploring-prometheus-use-cases-brian-brazil/, 2019. [Accédé le 29-Avril-2020].

[17] Ethan Galsta. Ndoutils documentation version 2.x. https://assets.nagios.com/downloads/nagioscore/docs/ndoutils/NDOUtils.pdf, Mar 2017. [Accédé le 29-Avril-2020].

[18] Zabbix documentation 4.4: Database creation. https://www.zabbix.com/documentation/current/manual/appendix/install/db_scripts. [Accédé le 29-Avril-2020].

[19] Bastien L. Time series database : qu'est-ce que c'est, à quoi ça sert ? https://www.lebigdata.fr/time-series-database-definition, May 2018. [Accédé le 29-Avril-2020].

[20] Zhaofeng Zhou. Key concepts and features of time series databases. https://www.alibabacloud.com/blog/key-concepts-and-features-of-time-series-databases_594734, Apr 2019. [Accédé le 29-Avril-2020].

[21] Sheng Di, Hai Jin, Shengli Li, Jing Tie, and Ling Chen. Efficient time series data classification and compression in distributed monitoring. In *Pacific-Asia Conference on Knowledge Discovery and Data Mining*, pages 389–400. Springer, 2007.

[22] Jeffrey Heer, Michael Bostock, and Vadim Ogievetsky. A tour through the visualization zoo. *ACM Queue*, 8:20, 01 2010.

[23] Nikos Bikakis. Big data visualization tools. *arXiv preprint arXiv:1801.08336*, 2018.

[24] Monthly weather forecast and climate kinshasa, democratic republic of congo. https://www.weather-atlas.com/en/democratic-republic-of-congo/kinshasa-climate. [Accédé le 2-Mai-2020].

Appendix A

Tableau comparatif des réseaux

	LoRaWAN	**Réseaux mobiles (2G/3G/4G/5G)**	**Sigfox**	**Satellite**	**Wi-Fi**	**Zigbee**	**Bluetooth**	**Réseaux mobiles IoT (NB-IoT, LTE-M)**
Débit	0.3 - 50kbps	EDGE: 384 Kbps HSPA+: 42 Mbps LTE-A: 1 Gbps 5G: 10 Gbps	100 ou 600 bps	Quelques kbps jusqu'à plusieurs Gbps. Cela dépend de la fréquence de transmission.	Wi-Fi 6: 9.6 Gbps	250 Kbps	5.0 : 5 Mbps	NB-IoT: 200 Kbps LTE-M: 1 Mbps
Bande de fréquences	868 - 915 MHz	400 - 3000 Mhz + 25 - 39 Ghz (5G)	862 - 928 Mhz	1 - 40 GHz	2.4 GHz / 5 GHz	868/915/ 2400 MHz	2400 Mhz	800 - 2200 MHz
Canal de communication	Half-duplex	Full-duplex	Half-duplex	Full-duplex	Half-duplex	Full-duplex	Full-duplex	Half-duplex
Consommation énergétique	Très Basse	Haute	Très Basse	Haute	Moyenne	Très Basse	Très Basse	Basse
Portée	5 - 20 km	2 km - 35 km	10 - 40 km	Mondiale	15 - 300 m	30 - 100 m	3 - 30 m	NB-IoT : 1 - 15 km LTE-M : 1 - 11 km
Couverture		Mondiale	Europe de l'ouest principalement	Mondiale	Réseau privé	Réseau privé	Réseau privé	
Sécurisé	Oui (AES 128)	Exploits possibles dans le réseau GSM. Nouvelles versions sont plus sécurisées. (Possible aussi d'ajouter dans la couche application)	Non (Possible dans la couche application)	Dépend du standard utilisé	Oui, mais des exploits existent	Oui (AES 128)	Oui (propriétaire)	Oui (propriétaire)