Développement d'une architecture multistandard pour l'IoT

Malgré les enjeux économiques de l'IoT, de nombreuses entreprises peinent à s'y investir pleinement. Le manque d'interopérabilité explique partiellement pourquoi. Dans cet article, on décrit une architecture matérielle et logicielle dans l'optique de résoudre ce problème. Le projet CleverIoT est né comme une manière d'implémenter cette architecture et de pallier cette fragmentation via la mise en place d'une plateforme centrale capable de fédérer les divers standards de communications.

Despite the economic stakes being high, many companies are struggling to fully invest themselves in the IoT. The lack of interoperability partly explains why. This article describes a hardware and software architecture meant to solve this issue. The CleverIoT project was born as a way to implement this architecture and overcome the fragmentation by setting up a central platform capable of fulfilling a unifying role.

Ing. VETS E.
CeREF Technique,
HELHa Mons

1. Introduction

Bien que le marché de l'IoT n'ait pas connu la croissance explosive qu'on lui avait espérée, il n'en reste pas moins considérable. Les nouvelles perspectives de croissances lui décrivent d'ailleurs une taille globale de plus de mille-milliards d'ici 2026 [1]. Les raisons propres à expliquer un tel ralentissement sont nombreuses. Cependant, de celles-là, on distingue un facteur technique particulièrement crucial : l'interopérabilité.

En effet, l'IoT se décline en de multiples standards de communications souvent redondants (dans leurs fonctionnalités, leurs performances et la portée de réseau qu'ils adressent). Ainsi, dans l'état actuel des choses, créer et coordonner un réseau IoT requiert un panel de compétences assez vaste, et peu transposable d'un protocole à l'autre. D'un autre côté, quoique l'IoT ne soit plus exactement à ses balbutiements, le type de R&D qui lui est caractéristique reste spécifique et donc coûteux. En conséquence, les entreprises s'aventurant dans le monde de l'IoT justifient difficilement d'ouvrir leurs spécifications ou d'apposer un prix plus abordable à leurs solutions.

Somme toute, la fragmentation de l'IoT mène à un manque à gagner considérable. Malheureusement, cette situation existe pour une raison. La densité énergétique des piles « boutons » au lithium, très populaires dans ce genre d'applications, ne dépassent pas les 3000 J/cm³ [2]. Prenons un ordre de grandeur de consommation représentatif aux alentours de 100 mW (ou 20 dBm, « the maximum RF output power for adaptive Frequency Hopping equipment » [14]). De cette puissance, on tire une durée de vie sur pile d'approximativement 8.3 h/cm³. Ces performances n'étant pas acceptables, un compromis doit être trouvé. Pour une durée de vie fixe, il est, en général, question de choisir entre la bande passante, la complexité du protocole, la portée du signal ou, plus simplement, le volume du nœud. Ainsi, aucune solution ne pouvant répondre à tous les cahiers des charges, toute une constellation de standards a été mise au point afin de matérialiser un échantillon particulier dans cet espace de contraintes. L'EnOcean [3], par exemple, de sorte à pouvoir alimenter ses nœuds par *energy harvesting* a choisi de sacrifier la bande passante et la complexité de son protocole.

On comprendra alors qu'aucun protocole fédérateur ne résoudra la problématique de l'interopérabilité. A contrario, l'état des technologies actuelles force à embrasser la fragmentation de l'IoT afin de profiter de la capacité de chaque standard à résoudre les problèmes pour lesquels il a été conçu.

Dans cet article, nous comptons proposer une architecture pour résoudre ce problème d'interopérabilité. En outre, nous mentionnerons le projet CleverIoT, né du souci de la mettre en œuvre.

2. État de l'art

Dans ce qui suit, on décrit deux grandes méthodes pour résoudre ces problèmes de fragmentation. On distingue ainsi la méthode des *concentrateurs ad hoc*, qui unissent les réseaux sous une même plateforme matérielle et, d'un autre côté, celle des *passerelles à traduction de protocole*, qui les combinent par le biais d'un super-réseau.

2.1. Les concentrateurs ad hoc

La figure 1 illustre la méthode sous-jacente aux concentrateurs ad hoc. À chaque réseau correspond une couche d'adaptation à la fois matérielle et logicielle qui imprime un format commun au protocole de communication en question. Fort de cela, les nœuds au sein d'un réseau communiquent de manière transparente avec ceux des autres, au moyen de la couche applicative commune.

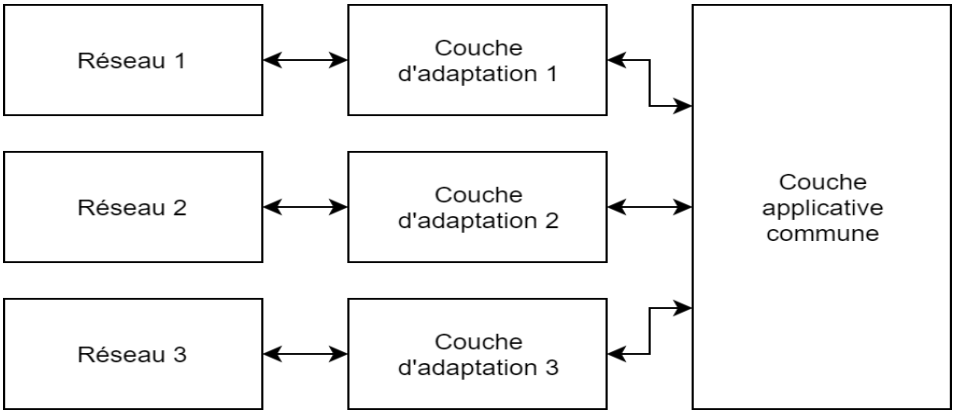

Figure 1 : Architecture ad hoc :
Les couches d'adaptations unifient les réseaux dans la couche applicative commune

De nombreuses tentatives [4][5][6] pour matérialiser cette architecture ont donné lieu à maintes suites logicielles. Leur mise en œuvre sont similaires. D'une part, elles sont écrites dans des langages de programmation haut-niveau. D'autre part, leur architecture logicielle évolue de manière itérative, modifiant certaines constructions selon les besoins du dernier standard avec lequel être compatible. Une telle approche est certes avantageuse puisque chaque standard vient s'y greffer « organiquement », adaptant la base logicielle en fonction des nouvelles exigences si besoin est. Mais, en contrepartie, la complexité de la partie commune grandit sans borne ou se limite à l'intersection réduite des protocoles adaptés. Par ailleurs, les solutions mises en place ne relèvent jamais d'un standard reconnu, prompt à favoriser une collaboration professionnelle à grande échelle. En conséquence, procéder selon cette architecture mène à une solution opérationnelle mais difficile à faire évoluer sur le long terme. De plus, elle décourage l'usage des fonctionnalités optimales mais moins communes des protocoles individuels. En conséquence, elle est potentiellement plus sujette à réduire les durées de vie sur batteries des nœuds dans le réseau.

Home Assistant [4] est un exemple d'un package Python suivant cette architecture. Il intègre en son sein un grand nombre de standard de communications et d'objets IoT spécifiques. Il souffre des problèmes mentionnés, à savoir d'une base logicielle de plus en plus difficile à gérer. Mentionnons également FHEH [5], écrit en Perl, ou

OpenHAB [6], écrit en Java, comme exemples supplémentaires, lesquels sont autant affectés par les obstacles décrits ci-dessus.

2.2. Passerelle à traduction de protocole

Les passerelles à traduction de protocole procèdent non plus en adaptant les standards de communication vers une implémentation logicielle de circonstances, mais plutôt vers un protocole central capable de rendre compte des particularités de tous les autres.

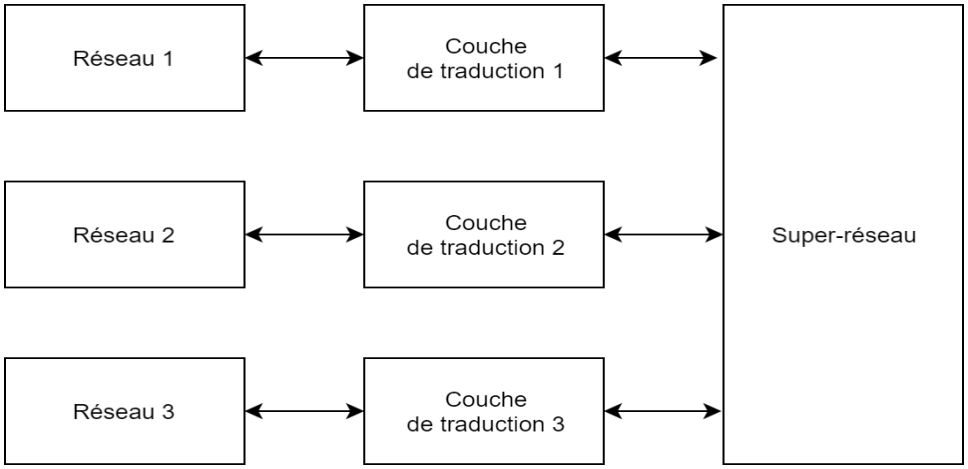

Figure 2 : Architecture à traduction :
Les couches de traduction unifient les réseaux dans un super-réseau

Les avantages liés à cette façon de faire sont multiples. D'abord, les ressources logicielles pour implémenter cette architecture ne seront plus développées « à l'aveugle » mais bien dans une direction précise : celle du protocole du super-réseau. Ensuite, accéder à la passerelle depuis l'Internet devient fortement simplifié : tous les objets sont disponibles de manière transparente par l'intermédiaire du super-réseau. Et ce dernier peut être mis à disposition sur l'Internet via un protocole adéquat, comme le MQTT. En outre, pour peu que le protocole central soit choisi avec discernement, diverses bibliothèques de fonctions matures sont disponibles

pour multiplier les capacités de la passerelle. Enfin, les compétences nécessaires pour maîtriser le dispositif ne sont plus arbitraires. En conséquence, le travail de développement s'en trouve plus appréciable.

Sous l'effet de cet inconvénient, peu de projets suivant cette architecture ont vu le jour. EMQX [7] opère similairement, en supportant des protocoles tels que le MQTT et le CoAP. Zhu, Qian, et al.[8], dans une optique cousine, à concilier le GPRS et le ZigBee. Et Fremantle Paul et al. [9] a investigué la possibilité de joindre un réseau IoT à une *web api*.

Mais, dans tous les cas, le même problème persiste : le nombre de protocoles gérés reste bas et, par là même, la surface balayée dans l'espace des fonctionnalités demeure modeste, réduisant ainsi l'intérêt de la plateforme.

3. Solution proposée : architecture hybride

L'architecture proposée ici est une forme hybride. En effet, il est question d'opérer via l'entremise d'un super-réseau mais également d'ajouter les protocoles de manière ad hoc. Cela permet de profiter de la facilité de développement de ce dernier tout en tirant parti des avantages structurels de l'autre. La figure 3 donne un aperçu de ce dont il est question.

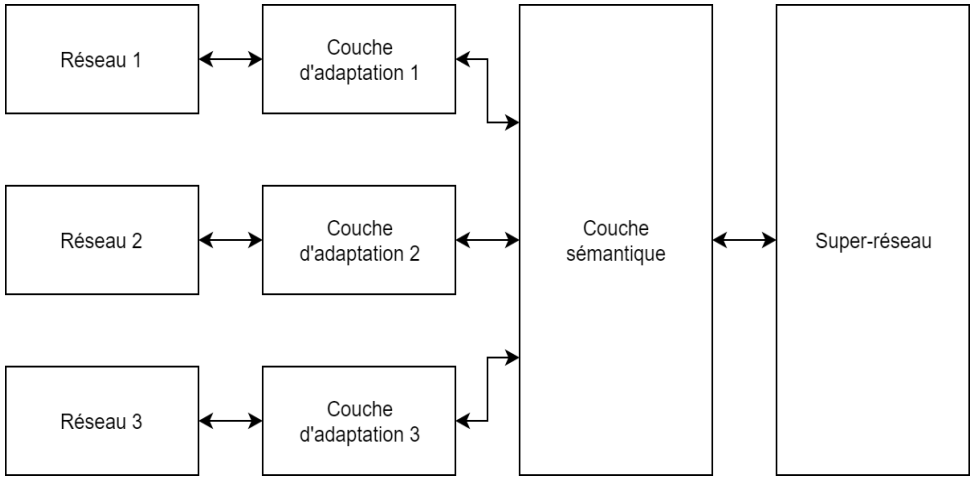

Figure 3 : Architecture hybride :
Les couches d'adaptation et la couche sémantique unifient les réseaux dans un super-réseau

En pratique, la couche applicative de chaque réseau est mise à disposition sur le super-réseau par l'intermédiaire d'une couche d'adaptation dédiée et de la couche sémantique.

Au sein de cette dernière, les ressources du nœud IoT seront représentées selon un format précis et universel. Le tout sera alors géré par un standard de communication central, capable de piloter les données.

3.1. Architecture matérielle

Du point de vue matériel, la stratégie des concentrateurs ad hoc ne connaît somme toute aucune alternative valable. Chaque réseau nécessite de maîtriser les prérequis de sa couche physique. En conséquence, s'il faut administrer N réseaux en parallèle, chacun (au pire) avec une couche physique différente, il faudra N plateformes matérielles dédiées. Cependant, avantage majeur de cette approche, rien ne force à manifester ces N plateformes par une architecture en étoile. En effet, le super-réseau est à même de prendre le rôle de super-passerelle de sorte à unir des sous-groupes via l'Internet. Ces plateformes prennent dès lors la forme d'un réseau distribué.

3.2. Architecture logicielle

Le point crucial de l'architecture logicielle porte sur la séparation des responsabilités entre *firmware* et *software*. En effet, les besoins en temps réels des protocoles sans fils imposent une contrainte matérielle qui se résout le plus simplement en séparant la charge de calculs entre une unité centrale et un coprocesseur. La question devient alors : où effectuer la scission dans la pile du protocole ? En pratique, plus la scission est proche de la couche physique, plus la bande passante entre le coprocesseur et son unité centrale devra être élevée et donc, potentiellement, la fiabilité de leur communication sera amoindrie. En conséquence, dès lors que la nécessité d'un coprocesseur est établie, il convient d'implémenter autant du standard que possible au sein de ce dernier. Accessoirement, l'industrie semble également se diriger dans cette direction, avec des entreprises internationales comme *Texas Instrument* ou *Silicon Labs* offrant des microcontrôleurs spécialisés pour cette application.

3.2. Besoin d'une sémantique générale

Au-delà des soucis des couches inférieures, les problématiques d'interopérabilité de l'IoT portent principalement sur les couches applicatives des standards de communication. Ainsi, les données d'un même capteur ou actuateur prendra une forme différente en fonction du protocole. Le Zigbee divisera les ressources en *profiles*, puis en *endpoints*, en *cluster* et, enfin en *commands* et *attributes*. Accéder à ces derniers requiert toute une procédure relativement complexe. À l'opposé de ceci, l'EnOcean encode les données des objets directement dans une trame extrêmement optimisée, sans hiérarchie. Dernier exemple, le MQTT représente les données dans une architecture *pub/sub* basée sur des abonnés, des publicateurs et des *topics*.

Fédérer cette population de standards nécessite que tous parlent un même langage, ou plus précisément une description systématique des capacités et des caractéristiques des nœuds. Pour ce faire, on fera usage de ce que la littérature appelle une « sémantique » ou « ontologie » pour l'IoT. Plusieurs candidats sont disponibles pour remplir ce rôle. Mentionnons d'abord IoT-lite [10] qui a l'avantage d'être dimensionné pour des plateformes pauvres en ressources. Comptons également IoT-

O [11], lequel offre une représentation permettant de prendre en compte la nature dynamique des nœuds de l'IoT et de leurs données.

4. Projet CleverIoT

4.1 Objectifs du projet

CleverIoT est un projet FIRST Haute-École développé par le Cerisic pour, en essence, concrétiser l'architecture décrite dans la section n°3. Son objectif central consiste en la « Mise en œuvre d'un système de communication multistandard dans l'internet des objets ».

Le déroulement de la recherche s'est organisé en deux étapes. D'abord, il a été question de formuler et de motiver l'architecture. Fort de cela, un travail significatif de développement a ensuite permis de mettre en place certains réseaux de l'IoT et de donner le jour à une maquette permettant d'en démontrer la faisabilité.

4.2 Intégration de la couche matérielle

L'influence principale qui a donné forme à la couche matérielle réside dans les problématiques de gestion de temps réels des couches physiques des standard de communications. En effet, bien que les protocoles de l'IoT soient ordinairement à bas taux de transmission, les contraintes en termes de timing n'en sont pas moins cruciales. Et puisqu'il est question de gérer plusieurs protocoles sur une même plateforme, on en déduit la nécessité de déporter du moins une partie de leur code sur un processeur dédié : un NCP (Network Co-Processor). Avec l'architecture générale à l'esprit, le projet CleverIoT a procédé de la manière illustrée dans la figure 4.

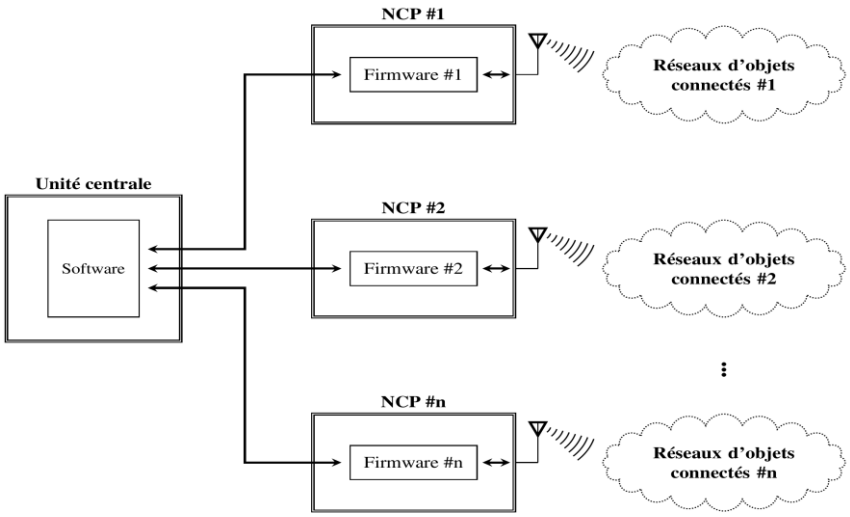

Figure 4 : Architecture matérielle :
L'unité centrale accède aux réseaux via un coprocesseur.

4.3 Intégration de la couche logicielle

Ordinairement, un protocole est implémenté via un groupe de logiciels appelé « pile ». Étant donné le choix architectural d'un NCP, il reste donc à préciser la portion du stack à y implémenter. L'intégration logicielle porte d'abord à s'interroger sur la partie du protocole qui existera comme « software » sur l'unité centrale et celle qui existera comme « firmware » sur le NCP. Afin de minimiser les besoins en bande passante, on a choisi de déporter autant que possible le protocole dans le coprocesseur. Sur le NCP, cela permet de s'appuyer sur les ressources logicielles mises à disposition par les grands fabricants de matériels pour l'IoT, comme Silicon Labs et son implémentation du ZigBee: EmberZNet. Et du côté unité centrale, pour chaque protocole, un contrôleur a été développé capable de gérer le coprocesseur et de coordonner le réseau adéquatement. Chacun de ces contrôleurs a été écrit en Python de manière à profiter de ses capacités de métaprogrammation et de mise en place de processus asynchrones parallèles.

4.4 Intégration de la couche sémantique

La partie sémantique de l'architecture est d'une importance capitale. C'est par son entremise que la problématique de l'interopérabilité est à proprement parler résolue. Malheureusement aucun consensus n'existe dans ce contexte dans la littérature. En conséquence, à défaut de pouvoir dépendre d'un standard établi, le projet CleverIoT, à titre provisoire, a choisi d'implémenter l'approche décrite dans [13]. Celle-ci se résume à étendre le standard RDF (Resource Description Framework) à l'IoT.

5. Conclusion

Résoudre la problématique d'interopérabilité de l'IoT requiert un travail de longue haleine. Et puisque l'état actuel de la technologie ne permet pas de mettre au point un standard de communication unique répondant à toutes les situations, on en conclut la nécessité de produire une solution multi-protocolaire. Dans cet article, on a décrit une architecture capable de l'accomplir. En outre, le projet CleverIoT se présente comme un premier pas dans cette direction. Mais, en conséquence de l'ampleur de la tâche, beaucoup reste à faire afin d'obtenir une solution rivalisant en stabilité et maturité avec l'infrastructure plus « traditionnelle » de l'Internet.

Sources

1. Fortune Business Insights, *Internet of Things (IoT) Market 2019: Global Industry Analysis, Size, Share, Trends, Market Demand, Growth, Opportunities and Forecast 2026,* *https://www.fortunebusinessinsights.com/industry-reports/internet-of-things-iot-market-100307*
2. TUNA, G., GUNGOR, V.C., *Energy harvesting and battery technologies for powering wireless sensor networks,* in Industrial Wireless Sensor Networks, 2016,
doi: 10.1016/B978-1-78242-230-3.00002-7

3. Technical Specifications, EnOcean Alliance, https://www.enocean-alliance.org/what-is-enocean/specifications/
4. Home Assistant, https://home-assistant.io
5. FHEM, https://fhem.de
6. OpenHAB, https://openhab.org
7. EMQX, https://emqx.io
8. ZHU, QIAN, et al. "Iot gateway: Bridging wireless sensor networks into internet of things." *2010 IEEE/IFIP International Conference on Embedded and Ubiquitous Computing*. Ieee, 2010, doi: 10.1109/EUC.2010.58
9. FREMANTLE, Paul, KOPECKY, Jacek, and AZIZ, Benjamin. "Web api management meets the internet of things." *European Semantic Web Conference*. Springer, Cham, 2015, doi: 10.1007/978-3-319-25639-9_49
10. BERMUDEZ-EDO, Maria, et al. "IoT-Lite: a lightweight semantic model for the Internet of Things." *2016 Intl IEEE Conferences on Ubiquitous Intelligence & Computing, Advanced and Trusted Computing, Scalable Computing and Communications, Cloud and Big Data Computing, Internet of People, and Smart World Congress (UIC/ATC/ScalCom/CBDCom/IoP/SmartWorld)*. IEEE, 2016, doi: 10.1109/UIC-ATC-ScalCom-CBDCom-IoP-SmartWorld.2016.0035
11. SEYDOUX, Nicolas, et al. "IoT-O, a core-domain IoT ontology to represent connected devices networks." *European Knowledge Acquisition Workshop*. Springer, Cham, 2016, doi: 10.1007/978-3-319-49004-5_36
12. CERISIC, projet CleverIoT, https://www.cerisic.be/technique/projet-cerisic/mise-en-oeuvre-dun-systeme-de-communication-multi-standard-dans-linternet-des-objets/
13. ABBAS, Abdullah, and PRIVAT, Gilles. "Bridging Property Graphs and RDF for IoT Information Management." *SSWS@ ISWC*. 2018.
14. ETSI, EN. "300 328 V1. 9.1." *Electromagnetic compatibility and Radio spectrum Matters (ERM)* (2015).