Second Rapport INF6404A : Présentation du Middleware

Alexandre Mao
David Johannès
Philippe Troclet
Fabien Berquez
Département Génie Informatique et Génie Logiciel
École Polytechnique de Montréal, Québec, Canada
alexandre.mao@polymtl.ca
david.johannes@polymtl.ca
philippe.troclet@polymtl.ca
fabien.berquez@polymtl.ca

19 mai 2016

1 Introduction

Le monde d'IoT représente par définition un système "intégré", où les différents objets interagissent entre eux, sont inter-connectés, à travers l'échange d'informations et de commande (requête, demande). Ces objets ont une forte probabilité d'être hétérogènes en termes de niveau de sécurité, de privacité minimal garanti, de technologie, de protocole de communication, et de politique d'exécution. Le challenge est ainsi davantage lié au besoin d'avoir une structure horizontale capable de gérer les spécifications de sécurité et de privacité de manière unique et homogène. Ces spécifications auront besoin en effet d'être instanciées sur des "entités" et auront potentiellement des interfaces d'implémentation, de spécification et de communication différentes.

Les caractéristiques de IoT comprennent donc un réseau à très grande échelle des objets, une grande hétérogénéité au niveau des dispositifs et du réseau, et un grand nombre d'événements générés spontanément par ces objets. Malheureusement, toutes ces caractéristiques feront du développement des diverses applications et des services une tâche très difficile. En général, le middleware peut faciliter un processus de développement en intégrant des dispositifs informatiques et de communication hétérogènes, et en soutenant l'interopérabilité au sein des diverses applications et services.

En effet, un middleware fait abstraction de la complexité du système ou du matériel, permettant au développeur d'applications de concentrer tous ses efforts sur la tâche à résoudre, sans la distraction des préoccupations orthogonales au niveau du système ou du matériel. Ces complexités peuvent être liées à des préoccupations de communication ou au calcul plus généralement. Un middleware fournit une couche logicielle entre les applications, le système d'exploitation, les couches de communication réseau et les différents dispositifs du système, ce qui facilite et coordonne certains aspects du traitement coopératif. Du point de vue informatique, un middleware fournit une couche entre les logiciels d'application et des logiciels système. Dans l'IoT, l'hétérogénéité des dispositifs implique très souvent une hétérogénéité considérable dans les technologies de communication utilisées, ainsi que dans les technologies au niveau du système, c'est pourquoi un middleware devrait supporter ces deux types d'hétérogénéité. Nous avons donc besoin d'un middleware qui respecte des caractéristiques, que nous décrirons par des modules. Ces modules seront divisés en trois groupes : les modules fonctionnels, liés aux services et aux fonctions que notre middleware doit fournir ; les modules non-fonctionnels, liés à la Quality of Service (QoS), aux performances et aux différentes exigences que notre middleware devra prendre en compte ; et les modules architecturaux, liés à l'architecture de notre middleware.

Rappelons que notre sujet consiste à établir un système IoT dans le domaine Smart Health, et plus précisément dans les services de soins intensifs des hôpitaux, afin de résoudre le problèmes de congestion et de surveillance en continu dans ces services. La Figure 1 nous permet de visualiser les choix de dispositifs et de protocoles de communication et de réseau que nous avons établis dans le rapport précédent. Avant de commencer à décrire les différents modules dont nous auront besoin dans notre middleware, ce que nous feront dans les section 2, 3, et 4, il est important de préciser le choix que nous avons fait concernant l'architecture générale de notre middleware. En effet, nous avons décidé de diviser notre middleware en deux couches différentes, la première étant liée au moniteur qui centralise toutes les informations des divers dispositifs présents dans la chambre du patient, et qui donc va gérer l'hétérogénéité entre les dispositifs présents dans l'environnement du patient. La seconde couche middleware est liée au Gateway de notre système, qui s'occupe de centraliser les informations de tous les moniteurs (le problème d'hétérogénéité se pose moins ici), et qui va faire le lien entre la couche de stockage et celle de liaison à la couche physique. Cette seconde couche va être celle qui permettra d'identifier les différents groupements de dispositifs à travers la connaissance des différents moniteurs intelligents.

La Figure 2 nous permet de visualiser la structure de notre système en considérant seulement les couches dispositifs, réseaux et communication, middleware, et architecture. Ainsi, dans les trois prochaines sections, notre tâche ne sera pas seulement de décrire les différents modules dont nous avons besoin dans notre middleware, mais aussi de décrire dans quelle(s) couche(s) middleware nous en avons besoin (possiblement les deux).

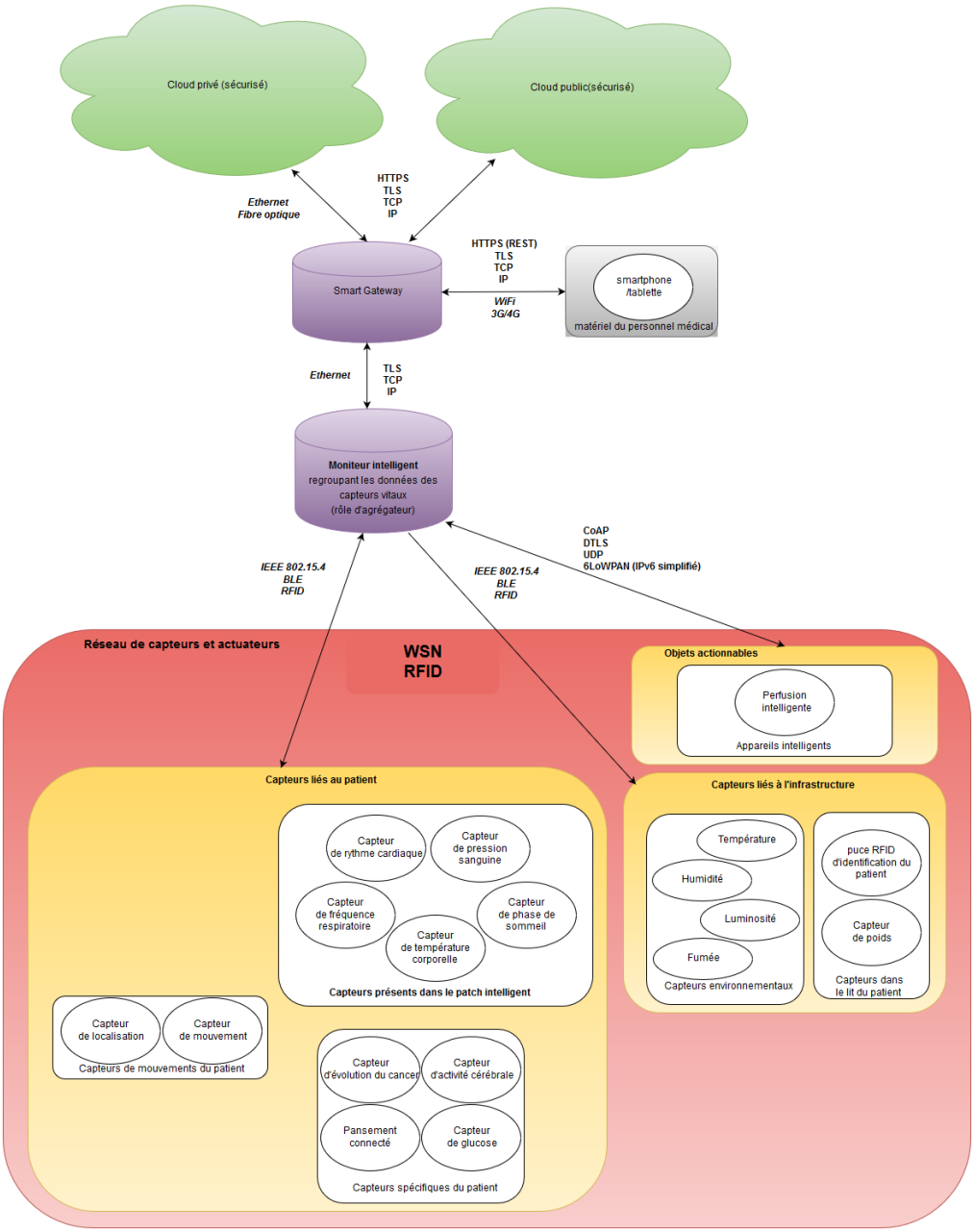

FIGURE 1 – Couches Dispositifs, et Protocoles de Réseau et de Communication de notre Système

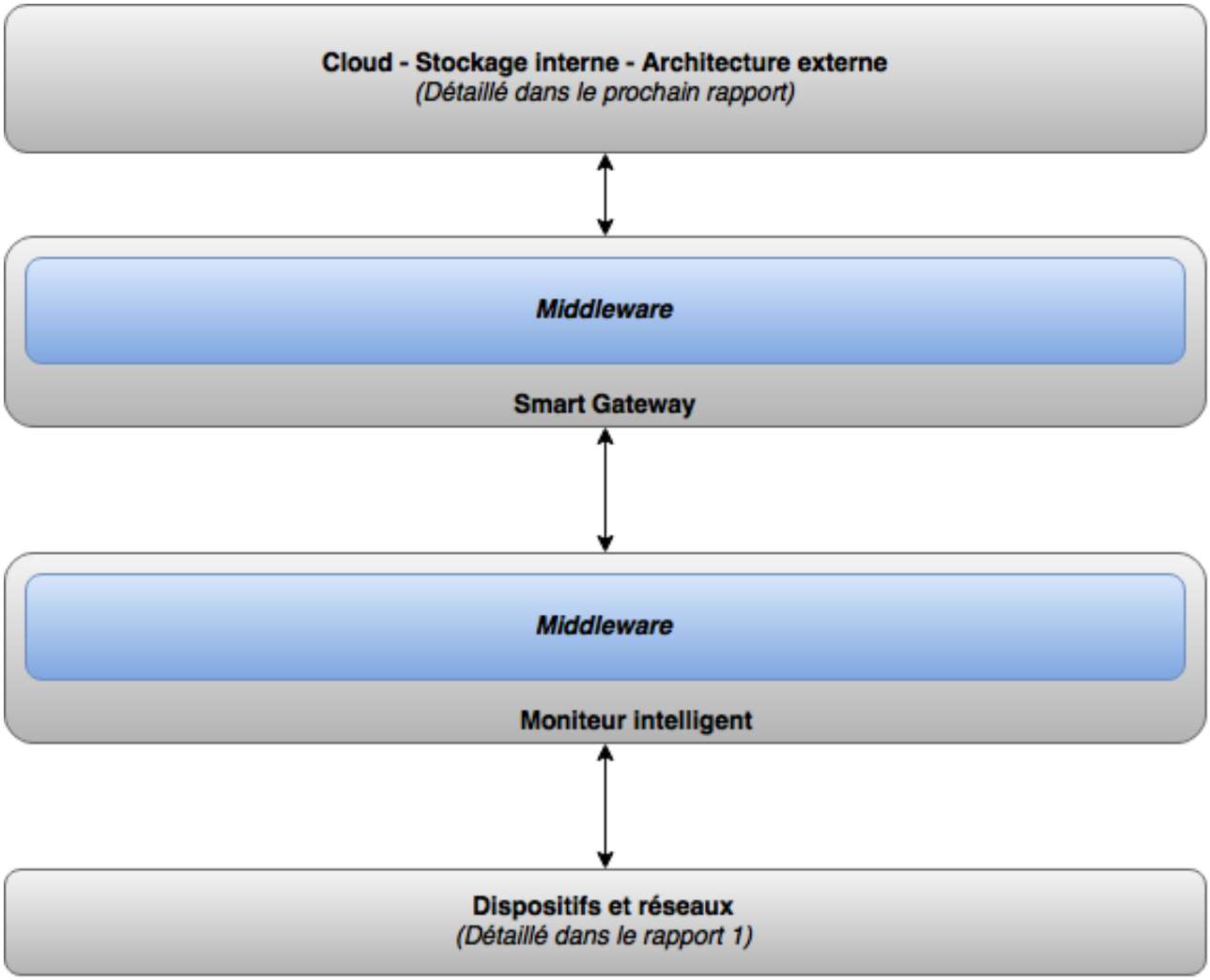

FIGURE 2 – Vue Globale de notre Middleware au sein de notre Système

2 Les Exigences du Middleware

Dans cette section, nous aborderons les différentes caractéristiques ou exigences nécessaires pour le bon fonctionnement d'un Middleware en général, et donc plus particulièrement pour notre Middleware. Ces exigences seront intégrées dans notre Middleware à travers les différents modules de celui-ci que nous détailleront dans la prochaine section.

La Figure 3 résume les exigences à avoir dans un Middleware en général (et donc dans le notre).

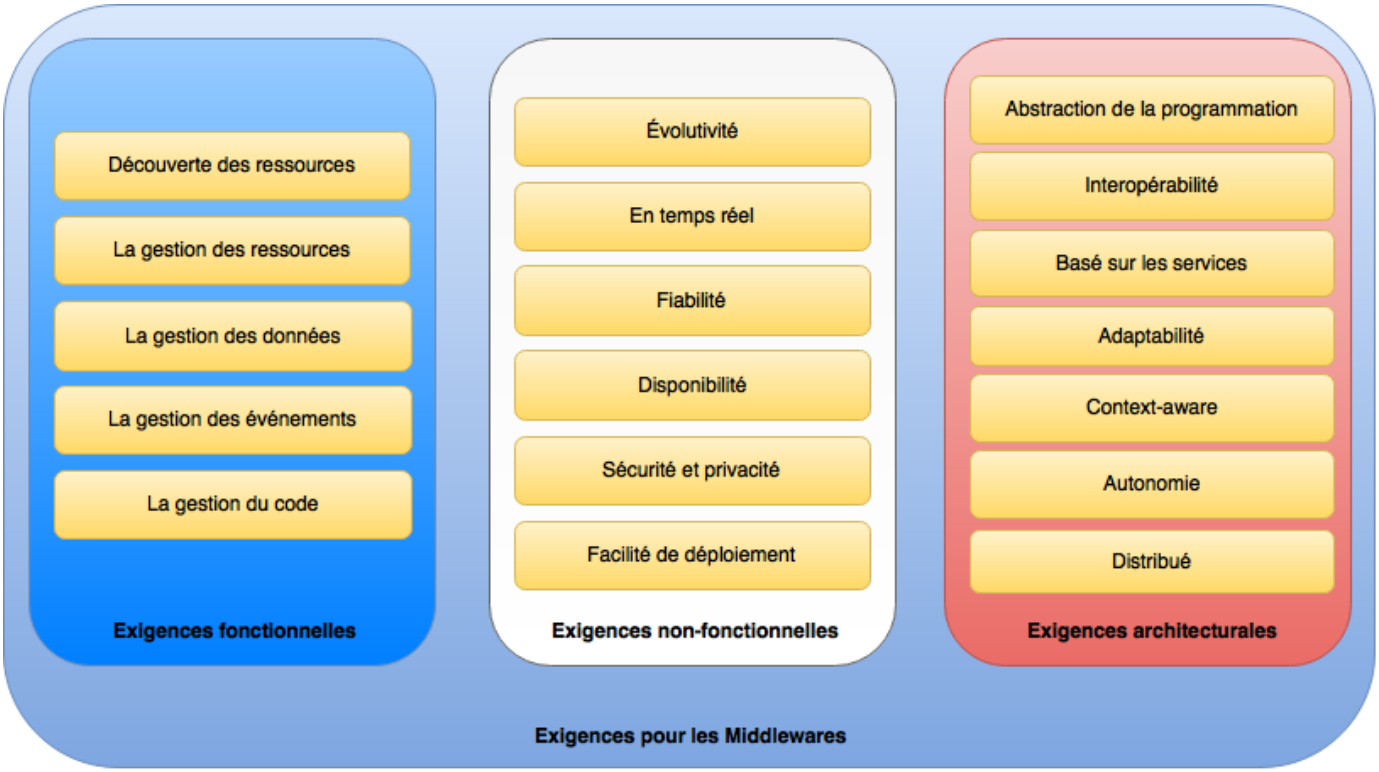

FIGURE 3 – Exigences nécessaires au sein d'un Middleware

2.1 Les Exigences Fonctionnelles

Les exigences fonctionnelles du Middleware correspondent aux services et aux fonctions qu'un middleware doit fournir, et sont donc représentées par la découverte des ressources, et toute forme de gestion (gestion des données, des ressources, des événements et du code).

Tout d'abord, une des exigences les plus significatives du Middleware est la découverte des ressources. Les ressources IoT comprennent les dispositifs hétérogènes de matériel (par exemple les étiquettes RFID, les capteurs, et les smartphones), la puissance et la mémoire des dispositifs, analogues à des dispositifs de conversion numérique (A/D), le module de communication disponible sur ces appareils, les informations au niveau de l'infrastructure ou du réseau (par exemple la topologie du réseau et des protocoles), et les services fournis par ces dispositifs. Les hypothèses relatives aux connaissances globales et déterministes de la disponibilité de ces ressources ne sont pas valides, car l'infrastructure et l'environnement de l'IoT est dynamique (les dispositifs, et leur nombre seront différents en fonction du patient qui est reçu). Ainsi, l'intervention humaine

pour la découverte de ressources est infaisable et, par conséquent, une condition importante pour la découverte de ressources est qu'elle doit être automatisée. Par ailleurs, lorsqu'il n'y a pas de réseau d'infrastructure, chaque appareil doit annoncer sa présence et les ressources qu'il offre.

La gestion des ressources est une exigence importante qui entre en jeu naturellement après la découverte des ressources du Middleware, car elle se réfère à la qualité de service (QoS ou Quality of Service). En effet, une QoS acceptable est attendue pour toutes les applications, et dans un environnement où les ressources ayant un impact sur la qualité de service sont limitées, comme l'IoT, il est important que les applications soient fournies avec un service qui gère ces ressources. Cela signifie que l'utilisation des ressources doit être surveillée, que les ressources allouées ou provisionnées le soient de manière équitable, et que les conflits de ressources soient résolus.

La gestion des données est également une caractéristique primordiale au sein du Middleware. Dans l'IoT, les données se réfèrent principalement aux données détectées ou toute autre information d'infrastructure de réseau d'intérêt pour les applications. Ainsi, un middleware IoT doit fournir des services de gestion des données pour les applications, y compris l'acquisition de données, le traitement des données (incluant le prétraitement des données), et le stockage de données. C'est une exigence importante à respecter dans notre cas car il faut récupérer les données des différents dispositifs, puis les traiter, et les interpréter, ce que nous décrirons par la suite à travers certains modules de notre Middleware.

Comme nous avons un système qui dépend des événements, la gestion de ceux-ci est aussi une exigence à considérer dans notre solution IoT. Il y a en effet potentiellement un grand nombre d'événements générés dans les applications IoT, qui devraient être gérés comme une partie intégrante d'un middleware IoT. La gestion des événements transforme les événements observés simples en des événements significatifs. C'est donc une exigence importante dans notre cas pour gérer les alertes, que ce soit au niveau des pannes des dispositifs, mais aussi au niveau des urgences (qui nécessitent l'intervention de personnel soignant).

Enfin, en temps qu'exigence fonctionnelle, il est nécessaire d'avoir une certaine gestion du code, car le déploiement du code dans un environnement IoT est difficile, et doit être directement pris en charge par le Middleware. En particulier, les services d'allocation de code et de migration de code sont nécessaires, et il est possible que les technologies en terme de dispositifs et de protocoles de communication et de réseau évoluent.

2.2 Les Exigences Non-Fonctionnelles

Les Middlewares ont un rôle important dans tout système d'IoT, et ce sont eux qui vont faire la transition entre les objets connectés et les couches supérieures, notamment celles de stockage. C'est pourquoi nous avons mis en évidence les différentes exigences non fonctionnelles du Middleware suivantes par rapport à notre système.

L'évolutivité correspond au fait qu'un middleware d'un système de l'IoT doit

pouvoir évoluer pour pouvoir répondre à l'expansion du réseau et à l'ajout d'applications et de services. Dans notre cas, l'évolutivité du middleware doit pouvoir prendre en compte l'ajout d'éventuels capteurs, et différents objets connectés. Il doit par ailleurs laisser la possibilité de fournir des services supplémentaires ainsi que des applications diverses en fonction des besoins.

Il faut permettre au Middleware de respecter l'exigence du temps réel. En effet, certains middlewares doivent pouvoir fournir des informations ou des services en temps réel lorsque la correction d'une opération qu'il prend en charge dépend non seulement de la correction logique, mais aussi du temps pendant lequel cette opération est réalisée. De nombreuses applications en temps réel vont utiliser les données fournies. L'envoi et la réception à temps des informations ou des services dans ces applications est alors essentiel. Certains capteurs que nous avons proposés ont pour but de surveiller des données vitales sur le patient, par conséquent ces informations doivent pouvoir être consultées en temps réel car des délais de transmission peuvent avoir des conséquences graves voire mortelles sur un patient.

La fiabilité d'un middleware est une exigence essentielle. Un middleware doit rester opérationnel pendant toute la durée d'une mission, même en présence de pannes. La fiabilité du middleware aide en dernier recours la réalisation de la fiabilité au niveau du système. Chaque composant ou service dans un middleware doit être fiable pour atteindre la fiabilité globale, ce qui inclut la fiabilité de la communication, des données, des technologies et des dispositifs de toutes les couches. Les signes vitaux qui sont surveillés imposent au middleware d'être capable de transmettre les données de façon fiable sans corruption de celles-ci, et cela toujours à cause des données qui sont surveillées.

Il y a la nécessité également d'introduire la disponibilité au sein de notre Middleware. En effet, même s'il y a une défaillance quelque part dans le système, le temps de récupération et la fréquence de défaillance de celui-ci doivent être suffisamment petits pour obtenir la disponibilité souhaitée. Les exigences en matière de fiabilité et de disponibilité doivent travailler ensemble pour assurer la plus haute tolérance aux pannes nécessaire depuis une application. Le middleware doit pouvoir être capable de détecter si l'un de nos capteurs est en panne, et pour cela, il va faire des requêtes à intervalles réguliers des signes vitaux et en cas d'absence de réponse pendant un temps prédéfini, une alerte sera envoyée au personnel médical pour vérifier l'origine de l'erreur. Dans le domaine de la santé, où la vie des patients est en jeu, nous ne pouvons pas nous permettre de laisser le patient sans surveillance médicale.

La sécurité et la privacité sont des exigences à ne pas oublier dans un tel système qu'est le notre. La sécurité doit être pris en compte dans tous les blocs fonctionnels et non fonctionnels. Nous avons différentes couches de sécurité dans les parties de notre architecture. Tout d'abord au niveau de l'identification des appareils connectés, nous proposons un système d'identification des appareils auprès du moniteur intelligent, ce qui évitera la connexion de capteurs non souhaités, et nous pourrons aussi dans l'autre sens connecter un capteur qu'à un seul moniteur. Au niveau de nos moniteurs intelligents, il y aura la mise en place d'un système de chiffrement des données pour l'envoi à la couche supérieure

pour préserver la confidentialité des données envoyées du patient.

La facilité de déploiement est également une exigence que nous visons au sein de ntore Middleware, car le déploiement ne devrait pas exiger des connaissances spécialisées, et les procédures d'installation et de configuration compliquées devraient être évitées. Dans notre cas l'ajout de capteur se fera à travers une interface simple sur le moniteur intelligent qui identifiera cet appareil au réseau mère. Et ainsi l'ajout de nouveaux dispositifs se fera de manière aisée.

2.3 Les Exigences Architecturales

Les exigences de Middleware abordées ici sont universelles, car applicables à n'importe quel Middleware, et concerne les exigences architecturales des Middlewares.

L'architecture de notre Middleware doit tout d'abord présenter une abstraction de la programmation, c'est-à-dire fournir une API (Interface de Programmation Applicative) pour les développeurs d'application. C'est une exigence fonctionnelle importante pour tout middleware. Pour le développeur de l'application ou du service, des interfaces de programmation de haut niveau ont besoin d'isoler le développement des applications ou des services provenant des opérations prévues par les infrastructures IoT hétérogènes et sous-jacentes. Le niveau d'abstraction, le paradigme de programmation, et le type d'interface doivent tous être pris en considération lors de la définition d'une API.

L'architecture de notre Middleware doit aussi être interopérable. Un middleware devrait en effet fonctionner avec des appareils, des technologies, ou des applications hétérogènes sans effort supplémentaire de la part du développeur de l'application ou du service. Les composants hétérogènes doivent être en mesure d'échanger des données et des services. L'interopérabilité dans un middleware peut être observée à partir du réseau, des perspectives sémantiques et syntaxiques, et chacun doit être pris en charge dans un IoT.

Notre Middleware doit être basé sur les services. Une architecture de middleware devrait être basée sur les services afin d'offrir une grande flexibilité lorsque les fonctions nouvelles et avancées doivent être ajoutées au middleware IoT. Un middleware basé sur les services fournit des abstractions pour le matériel sous-jacent complexe à travers un ensemble de services (par exemple la gestion des données, la fiabilité, la sécurité) nécessaires pour les applications.

L'architecture de notre Middleware doit être adaptable. En effet, un middleware doit être adapté afin qu'il puisse évoluer pour s'accorder à des changements dans son environnement. Dans l'IoT, le réseau et son environnement sont susceptibles de changer fréquemment. En outre, les demandes ou le contexte au niveau de l'application sont également susceptibles de changer fréquemment. Afin d'assurer la satisfaction des utilisateurs et l'efficacité de l'IoT, un middleware doit s'adapter dynamiquement ou s'ajuster à toutes ces variations.

Un middleware doit être context-aware (attentif au contexte), c'est-à-dire conscient du contexte des utilisateurs, des dispositifs, et de l'environnement et

utiliser ce contexte pour des offres de services efficaces et essentiels pour les utilisateurs.

Notre middleware doit bien évidemment présenter une certaine autonomie. Les appareils, les technologies, ou les applications sont des participants actifs dans les processus de l'IoT et ils devraient pouvoir interagir et communiquer entre eux sans intervention humaine directe. L'utilisation de l'intelligence, y compris des agents autonomes, de l'intelligence embarquée, et des approches prédictives et proactives dans le middleware peut satisfaire cette exigence.

Enfin, comme nous le montre en partie la Figue 2, notre Middleware doit être distribué. En effet, les applications, les dispositifs, et les utilisateurs sont susceptibles d'être distribués géographiquement, et donc une mise en oeuvre d'une vue ou d'un middleware centralisé ne sera pas suffisant pour supporter de nombreux services ou applications distribuées. Une implémentation d'un middleware doit prendre en charge les fonctions qui sont distribuées à travers l'infrastructure physique de l'IoT.

3 Intergiciel - Middleware

3.1 Middleware niveau moniteur

Dans cette partie, nous allons nous intéresser au middleware présent au niveau du moniteur. Afin de contextualiser, nous allons rappeler les fonctionnalités principales de ce dernier. Premièrement, il a un rôle d'agrégateur. Il doit donc rassembler les données issues de différents capteurs et les envoyer vers la passerelle intelligente via un unique message.

Ce requis impose de pouvoir comprendre différents protocoles, tant au niveau physique que applicatif, ainsi que différents formats. Il est en effet peu probable que tous les capteurs utilisent le même protocole pour leurs données sous un même format. Il est également nécessaire de permettre l'évolutivité au sein du système afin que de futurs capteurs plus performants, mais utilisant une nouvelle façon de communiquer, puissent être intégrés au système.

De plus, afin d'optimiser le temps du personnel médical, il est souhaitable que le moniteur et donc le middleware facilite la découverte de capteurs ainsi que la récupération de leur contexte. Une fois les capteurs installés, le lien entre les données reçues et l'identité du patient devrait être fait automatiquement (ou presque). Ces mêmes capteurs devraient être accessibles depuis l'extérieur. En effet, pour certaines données relevées rarement, il est possible qu'un médecin souhaite les rafraîchir plutôt que d'attendre la prochaine mise à jour. Il se peut également que le médecin désir donner un ordre à un des objets intelligents. Pour cela, il faut que ces objets aient un nom, ou du moins soient adressables. En particulier, un nom du type *idPatient@typeDeCapteur* serait particulièrement pratique à utiliser.

Enfin, il est impératif, dans la mesure du possible, de sécuriser les communications au sein du système. Cela passe par un contrôle d'accès au niveau du moniteur avec un mécanisme d'authentification. Mais aussi par une détection des pannes. En effet, il est souhaitable que, de part l'importance des données fournies par ces capteurs, ils doivent être remplacés dès le premier indice de dysfonctionnement. De manière générale, il est important de garantir une bonne disponibilité ainsi qu'une certaine fiabilité.

Modules	Découverte ressources	Scalabilité	Interopéra--bilité	Adapta--bilité	Fiabilité	Disponi--bilité	Temps réel	Basé sur les services	Context aware	Autonome
Découverte des ressources	✓	✓		✓						
Détection panne					✓	✓	✓	✓		
Décision		✓		✓	✓	✓				✓
Gestion du contexte										✓
Émission d'événements			✓	✓		✓			✓	
Gestion des différents protocoles physiques		✓								
Nommage des capteurs		✓	✓	✓						
Agrégation de données et centralisation		✓	✓	✓						
Gestion des différents protocoles applicatifs		✓	✓	✓						

Table 1 – Utilisation des services du middleware par les différents modules

Les tableaux **??** et **??** présentent les modules sur la première colonne avec les exigences du middleware sur la première ligne.

Par ailleurs, il serait aussi nécessaire de présenter une API simple d'utilisation au niveau du middleware afin de faciliter la programmation de nouvelles fonctionalité.

Modules	Gestion ressources	Gestion données	Gestion événements	Gestion du code
Découverte des ressources	✓			
Détection panne	✓			
Décision				
Gestion du contexte		✓	✓	
Émission d'événements		✓	✓	
Gestion des différents protocoles physiques				✓
Nommage des capteurs	✓			
Agrégation de données et centralisation		✓		
Gestion des différents protocoles applicatifs				✓

TABLE 2 – Utilisation des services de gestion du middleware par les différents modules

3.2 Middleware niveau gateway

La smart Gateway présente un rôle central dans notre architecture, c'est elle qui va faire l'intermédiaire entre les parties plus bas niveau comprenant les dispositifs et le middleware s'occupant de gérer ces dispositifs, et la partie supérieure qui correspond au reste de l'application avec le stockage des données, et les différentes applications. Dans notre cas, elle va avoir différents modules pour répondre aux exigences du système. Elle aura un rôle primordial dans la transmission en temps réel des alertes depuis les couches inférieures, pour cela, un module autonome de gestion et d'émission des alertes sera présent pour pouvoir recevoir et réémettre aux bons destinataires ou aux bonnes applications les alertes, ce module pourra être couplé avec un module de localisation du personnel médical et du patient, pour que l'alerte soit déclenchée dans la zone proche du patient et qu'elle atteigne le personnel médicale compétent pour agir sur cette urgence tout en leur signalant la localisation de l'urgence. Ce module aura besoin d'être fiable et disponible dans la gestion des événements qu'elle reçoit car ceux-ci peuvent être des alertes vitales sur des patients.

L'identification des patients, et leur localisation dans l'hôpital est une nécessité, la présence d'un module permettant d'ajouter au réseau les nouveaux moniteurs connectés est indispensable. Dans l'architecture que nous proposons, nous avons un module dédié à la recherche des appareils, qui dans ce middleware sera la recherche des moniteurs intelligents. Les appareils de plus bas niveau vont se déclarer à ceux de plus haut niveau pour que ces derniers puissent les identifier de manière unique et leur associer les bonnes informations. Dans notre cas par exemple, lors de l'installation d'un nouveau patient, le moniteur va rechercher ou recevoir l'identifiant du patient, et une fois connectée au réseau, il se déclarera auprès de la passerelle intelligente la plus proche pour que celle-ci puisse mettre en mémoire la correspondance entre le moniteur et le patient. Une fois l'élément entré dans la table de correspondance de la passerelle intelligente, toutes les informations fournies par le moniteur pourront être transmises de manière cohérente auprès de la passerelle qui fera la redirection pour

le stockage ou l'interprétation des données. Et la passerelle intelligente, informera ensuite le serveur IoT et les autres passerelles de la présence de ce nouvel élément. La méthode de recherche des dispositifs que nous proposons permettra une évolutivité et une adaptabilité aisées du système.

Une fois qu'une passerelle intelligente a pris connaissances des moniteurs connectés dans sa zone de couverture, elle va avoir pour rôle de transmettre les différentes informations de ces moniteurs pour leur stockage. Elle va donc s'occuper de gérer les données reçues avec l'aide d'un module de communication Cloud, qui s'occupera de faire l'intermédiaire entre les couches de stockage et les couches supérieures, pour cela elle doit être capable de communiquer avec les moyens de stockages que nous proposons sous forme de Cloud. Ce module s'occupera donc de la gestion des données, et sera donc couplé avec un module de stockage interne, qui servira de table de correspondances entre les différents moniteurs intelligents et le patient auquel il réfère. Le module de stockage interne servira à avoir le contexte des données qui sont fournies.

Comme nous avons pu le voir jusqu'à maintenant, le middleware présent au niveau du smart gateway à un rôle primordial dans le traitement et la redirection des informations, il doit être capable de faire ces différentes tâches en prenant en compte la sécurité des données qui transitent. La smart gateway va avoir un module d'authentification et de contrôle d'accès pour restreindre l'accès à ces données et éviter le vol de données privées sur les patients. Les applications dans le domaine de la santé, recueillent des informations confidentielles sur les patients, l'exposition de ces informations peut avoir des conséquences sur la vie privée et professionnelle de ces personnes, c'est pourquoi nous incorporons différents modules de sécurité dans les différentes couches de notre système. La smartGateway va ainsi posséder un module de chiffrement des communications pour garder la confidentialité des données qui sont transmises. Le couplage de ce module avec un module d'authentification et de contrôle d'accès va nous permettre d'avoir des mesures de sécurité quant aux informations des patients.

Enfin pour l'évolutivité du système proposé ainsi que l'ajout de différentes fonctionnalités, nous proposons un module API service qui contiendra les différentes API nécessaire à la transcription de méthodes et d'application depuis d'autres plateformes, il sera ainsi facile d'ajouter des fonctionnalités à la smart gateway comme par exemple l'introduction de nouveaux capteurs, et nous aurons donc une abstraction du code qui permettra une intégration plus facile de nouveaux éléments.

3.3 La sécurité dans le middleware

La vie privée et la sécurité sont des préoccupations majeures dans les systèmes IoT, et en particulier dans notre système centré autour de la santé. En effet, les données, en cas de fuite ou de corruption par un agent extérieur, sont susceptibles d'avoir des conséquences graves sur la vie du patient. Notre middleware doit donc prendre en compte cette problématique et répondre à ces défis. Comme la sécurité est un problème transverse, n'étant pas limité à l'un des maillons du système, nous avons choisi de l'évoquer dans cette partie séparée.

Après l'étude des risques potentiels, nous avons défini les modules suivants pour offrir des services et des garanties concernant la sécurité :

- ***Authentification :*** Il s'agit d'un module destiné à s'assurer et attester de l'identité d'un service, d'une application ou d'un utilisateur faisant une requête ou donnant un ordre à notre système, et particulièrement à notre middleware. L'objectif derrière ces considérations est de protéger le reste du système de requêtes provenant d'agents non-identifiés, tout en pouvant garantir l'origine des ordres, c'est à dire s'assurer qu'il n'est pas possible, pour une personne ou un service ayant fait une requête ou ayant transmis un ordre, de répudier l'avoir fait. Ceci est nécessaire à la fois pour la sécurité, mais aussi parfois pour des raisons de législation et de responsabilité devant la loi.
- ***Contrôle d'accès :*** Ce module, qui s'appuie tout d'abord sur le module d'*authentification* intervient à l'étape suivante. Une fois le requérant identifié, il convient de s'assurer qu'il a le droit d'accéder, de visionner, de modifier ou de supprimer les données visées, qu'elles soient dans le Cloud ou dans le stockage local mis en place dans notre architecture. Parallèlement, ce module s'assure également qu'un ordre passé ne soit transmis que si l'utilisateur ou l'application demandeuse est autorisée à effectuer cette action. Ce module répond aux considérations à la fois de sécurité, mais aussi de privacité, en s'assurant que les données ne soit transmises qu'à des personnes ou des applications identifiées, autorisées et légitimes.
- ***Chiffrement des communications :*** Ce module a pour rôle de chiffrer les communications en utilisant les techniques de chiffrage appropriés afin de s'assurer qu'en cas de détournement des paquets, ou d'attaques de type Man In The Middle, les données sensibles comme le profil des patients, leur historique, leur dossier, les éléments venant de la base de données interne, ou du Cloud ou encore les informations de localisation des médecins ou des patients ne puissent être utilisés ou même consultés. Ce module agit pleinement pour favoriser la sécurité, mais plus encore la privacité. Il s'agit d'éviter ici les fuites de données personnelles, l'un des risques majeurs de l'IoT, renforcé par le caractère sensible des données transitant.
- ***Sécurité et intégrité des communications :*** Ce module est conçu pour s'assurer de la sécurité des communications, particulièrement celles orientées vers et depuis l'extérieur (par exemple le Cloud), mais aussi en interne entre les moniteurs et la Smart Gateway, ainsi que de l'intégrité des données reçues. En effet, une fois les données des capteurs agrégées (notamment avec le contexte) sur le moniteur, celles ci sont transmises à la Smart Gateway qui doit s'assurer de leur intégrité afin de ne pas enregistrer de fausses entrées. De même, certaines alertes étant générées par les moniteurs intelligents et transmises à la Smart Gateway pour leur transmission finale vers l'équipe médicale, il est important qu'aucun renseignement ne soit perdu au cours de ces transmissions. Ce module se

concentre donc sur la sécurité, plus que la privacité des communications, car celle ci est renforcée par le module de chiffrement.

Ainsi, nous avons conçu cette partie transverse du middleware afin de s'assurer d'une sécurité et d'une protection de la vie privée maximale dans la limite des possibilités techniques. Les problématiques liées à la Qualité de Service (QoS) sont également pris en compte par la vérification de l'intégrité, donc de la fiabilité des données transmises, afin d'éviter des situations potentiellement risquées.

3.4 Récapitulatif

Figure 7.

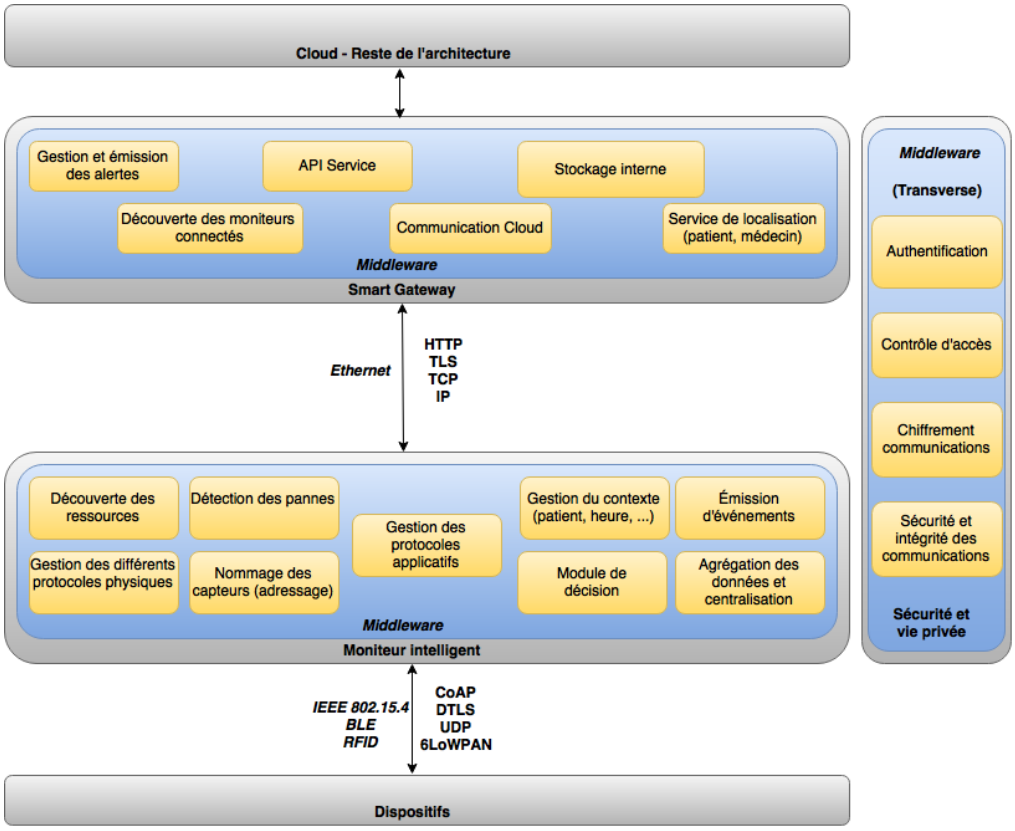

FIGURE 4 – Le Middleware de notre Système et ses Modules

Modules	**Découverte des ressources**	**Gestion des ressources**	**Gestion des données**	**Gestion des événements**	**Gestion du code**
Gestion et émission des alertes				✓	
Communication Cloud			✓		
Service de localisation (patient, médecin)		✓			
Découverte des moniteurs connectés	✓				
Stockage interne			✓		
API service					
Découverte des ressources	✓				
Détection des pannes		✓			
Décision					
Gestion du contexte			✓	✓	
Emission d'événements			✓	✓	
Gestion des différents protocoles physiques					✓
Nommage des capteurs		✓			
Agrégation des données et centralisation			✓		
Gestion des protocoles applicatifs					✓

Table 3 – Modules de notre Middleware et Exigences Fonctionnelles

Modules	**Évolutivité**	**Temps réel**	**Fiabilité**	**Disponibilité**	**Facilité de déploiement**
Gestion et émission des alertes		✓	✓	✓	
Communication Cloud				✓	
Service de localisation (patient, médecin)			✓	✓	
Découverte des moniteurs connectés	✓				✓
Stockage interne				✓	
API service					
Découverte des ressources	✓				✓
Détection des pannes		✓	✓	✓	
Décision	✓		✓	✓	✓
Gestion du contexte					
Emission d'événements		✓	✓	✓	
Gestion des différents protocoles physiques	✓				✓
Nommage des capteurs	✓				✓
Agrégation des données et centralisation	✓				✓
Gestion des protocoles applicatifs	✓				✓

Table 4 – Modules de notre Middleware et Exigences Non-Fonctionnelles

4 Conclusion

Nous avons présenté dans ce premier rapport notre solution IoT orientée Smart Health, afin de résoudre le problème de congestion des hôpitaux dans les

Modules	**Abstraction de la programmation**	**Interopéra-bilité**	**Basé sur les services**	**Adaptabilité**	**Context aware**	**Autonomie**
Gestion et émission des alertes						✓
Communication Cloud			✓			
Service de localisation (patient, médecin)			✓		✓	
Découverte des moniteurs connectés				✓		
Stockage interne			✓			
API service	✓		✓			
Découverte des ressources				✓		
Détection des pannes			✓			
Décision				✓		
Gestion du contexte					✓	
Emission d'événements			✓			✓
Gestion des différents protocoles physiques		✓		✓		
Nommage des capteurs		✓		✓		
Agrégation des données et centralisation		✓		✓	✓	
Gestion des protocoles applicatifs		✓		✓		

TABLE 5 – Modules de notre Middleware et Exigences Architecturales

services de soins intensifs. Plus précisément, nous avons proposé les dispositifs et les protocoles de communication et de réseau de notre solution. Nous avons aussi abordé les problématiques liées à notre solution, principalement celles liées au QoS, à la privacité et à la sécurité. Ceci nous permettra ainsi pour nos prochains rapports de nous concentrer autour de ces problématiques afin de permettre à notre solution de respecter le plus possible ces paramètres.

Dans notre prochain rapport, nous aborderons ainsi les détails de conception concernant la couche située au dessus de celles présentées dans ce rapport, à savoir la couche Middleware. Nous verrons alors comment nous allons gérer le stockage des données et comment nous allons essayer de résoudre les problématiques liées à notre solution notamment avec la mise en place d'un environnement de stockage approprié. Nous détaillerons tout cela dans le deuxième rapport.