Administration autonomique et décentralisée de flottes d'équipements de l'Internet des Objets

Neil Ayeb

THÈSE

Pour obtenir le grade de

DOCTEUR DE L'UNIVERSITE GRENOBLE ALPES

Spécialité : **Informatique**

Arrêté ministériel : 25 mai 2016

Présentée par

Neil AYEB

Thèse dirigée par **Eric RUTTEN, INRIA** et co-encadrée par **Sebastien BOLLE et Thierry COUPAYE, Orange Labs**

préparée au sein de l'**INRIA** et d'**Orange Labs** dans **l'École Doctorale Mathématiques, Sciences et technologies de l'information, Informatique**

Administration autonomique et décentralisée de flottes d'équipements de l'Internet des Objets.

Autonomic and decentralized device management for the Internet of Things.

Thèse soutenue publiquement le **25 novembre 2020**, devant le jury composé de :

Monsieur Thomas LEDOUX
Professeur, IMT Atlantique - Nantes, Rapporteur

Madame Laurence DUCHIEN
Professeure, Université de Lille, Rapportrice

Madame Françoise BAUDE
Professeure, Université de Nice, Examinatrice

Monsieur Thierry MONTEIL
Professeur, INSA Toulouse, Examinateur

Monsieur Noël DE PALMA
Professeur, Université Grenoble-Alpes, Président du Jury

Monsieur Eric RUTTEN
Chargé de Recherche, HDR, INRIA, Directeur de thèse

Monsieur Thierry COUPAYE
Directeur de Domaine de Recherche, HDR, Orange Labs, Co-encadrant

Monsieur Sébastien BOLLE
Responsable de Programme de Recherche, Orange Labs, Co-encadrant

Monsieur Marc DOUET
Responsable de Projet de Recherche, Orange Labs, Invité

Remerciements

Je souhaite tout d'abord remercier les membres du jury d'avoir accepté de rapporter et/ou examiner puis évaluer mes travaux. L'expression de ma gratitude va à Éric, qui a accepté de diriger ma thèse ainsi qu'à Sébastien et à Thierry pour l'avoir co-dirigé. Sans leur disponibilité, leur expertise pointue, leurs efforts et surtout leur patience, je n'aurais probablement pas pu mener à bien ces travaux. Je remercie Orange pour avoir financé ces travaux de recherche et aux équipes à Orange Meylan et à l'INRIA de Grenoble qui m'ont accueilli et offert un cadre de travail bienveillant sans lequel une thèse ne peut aboutir. Merci à Marc d'avoir guidé de près (p. ex. à trouvé debout en cinq minutes café à la main comment faire un kill avec grep pour tuer un processus déterminé) et pour ses conseils ainsi que sa sagesse qui m'ont fait progresser tant humainement que techniquement.

Dédicace à Julien, Ségo, Thibaud, pour nos concerts, sorties, pause cafés, randos, voyages, beuveries ainsi que nos moult repas dans l'écoute, le partage et la bonne humeur.

Kudos à Guillaume, Jules, Ouiame, Romain pour les festivals, soirées, before, afters, afters de l'after durant lesquels on a bien rit, trollé, et j'en passe.

Chapeau à Samuel, François, Thibault, Mayl, pour les instants de geekeries, les burgers, les concerts de musiques uniques en leurs genres, et pour certains les escapades diverses entre le nord (oui Paris c'est au nord de Grenoble non ?) et le sud de l'hexagone.

yare-yare[1] ... Alâa pour nos interminables discussions et débats enrichissants sur la science, l'informatique, les nouvelles technologies, les courants économiques et politiques.

Ma gratitude va à mes collègues et aux musicien(ne)s d'HARA KIWI (anciennement les BKA) à Orange Meylan que j'ai pu croiser autant au bureau qu'en dehors de travail (ping à Ioana pour les multiples balades à vélo). Entre randonnées, restaurants, 'afterworks' en centre-ville, fête de la musique et concerts, il y avait de quoi faire. De même que les fois où pour certain(e)s, je débarquais dans leurs bureaux sans crier gare ! Coucou, Frédéric, Philippe, Romain P., Hélène, Julien C., Pauline, David, Matthieu (à la GdV[2] avec Thibaud et au bureau), Anne, Jean-Roch, Jean-Didier (JD), Alexis, Severine, Émilie, Stéphane, Guilhem, Emna, Naji, Agnès et Nadia.

Merci à mes collègues de l'équipe CTRL-A à l'INRIA, Raphaël, Sophie, Lucie, Imma, qui m'ont accompagnés durant ma phase de rédaction, et qui ont contribué à la bonne ambiance dans les bureaux de MINATEC.

Par la même, j'ai une pensée pour les personnes qui m'ont occasionnellement soutenu de loin (parfois même en pleine mer ! n'est-ce pas, capitaine ;)), grâce au miracle du numérique, vous vous reconnaitrez ;)

Mes pensées vont à mon père Dr. Béchir Ayeb, à ma mère (a.k.a Mida), à ma grande sœur Sara qui est au pays des *caribous*, et ma petite sœur Lyna proche

1. si vous avez la référence, bravo !
2. Gym de Vincent

de la méditerranée, qui ont toutes et tous, foi en ma capacité à continuellement m'améliorer et les rendre fiers de ma personne.

Résumé

Avec l'avènement de l'internet des objets (IdO) qui se base sur des objets hétérogènes, dynamiques et de haute volumétries, des besoins en administration à distance sont requis pour un bon fonctionnement de ces objets. Il s'agit par exemple de mise à jour de logiciels, de configurations, de résolution de problèmes à distance ainsi que de récolte de données de fonctionnement. Ces opérations d'administration permettent d'assurer une bonne qualité de service et d'expérience pour les utilisateurs. Elles permettent en outre, le déploiement de nouvelles fonctionnalités, de correctifs logiciels, et de mise à jour de sécurité.

Les plateformes industrielles existantes d'administration montrent leurs limites avec des parcs formés d'objets statiques, en termes de capacités et d'environnements, comme les passerelles internet domestiques et décodeurs de flux TV. Ces plateformes sont opérées manuellement par des équipes d'administrateurs systèmes et requièrent une expertise conséquente.

Concernant les flottes de l'IdO, l'hétérogénéité se traduit en un ensemble d'équipements ayant des capacités différentes de calcul et de connectivité réseau. La dynamicité concerne les environnements de ces équipements qui varient en termes de services en cours d'exécution, de qualité du lien réseau, de capacité restante de calcul. La volumétrie des objets de d'IdO impose un besoin de passage à l'échelle afin de gérer des milliards d'équipements contrairement aux flottes composée de millions d'équipements aujourd'hui.

Par suite, l'administration de flottes de l'IdO requiert une adaptation constante de ces opérations en termes de nature, de vitesse et de cible. Les approches manuelles existantes ne permettent pas de réaliser ces opérations en prenant en compte les spécificités de l'IdO.

Afin d'adresser cette problématique, ce travail de thèse industrielle chez Orange, vise à appliquer le paradigme de l'informatique autonomique au pilotage et la distribution des plateformes d'administration. L'objectif est d'assurer que les besoins en administration des flottes de l'IdO soient automatiquement réalisés, et ce, avec une consommation optimale de ressources de calcul et de réseau, ainsi qu'avec un nombre le moins élevé possible, d'erreurs d'exécution.

Notre proposition s'appuie sur quatre boucles autonomiques coordonnées. Deux d'entre elles sont responsables de l'automatisation du maintien à jour de la flotte d'équipements tandis que les deux autres sont chargées de la régulation de l'utilisation des ressources assurant ainsi un passage à l'échelle vertical et horizontal.

Notre proposition est validée au travers de deux prototypes. Le premier sert de démonstrateur de l'utilisabilité de notre approche pour le pilotage d'une plateforme industrielle d'administration de l'IdO (Live Objects d'Orange) qui est utilisée en production. Le deuxième démontre les capacités de passage à l'échelle vertical de notre proposition. Il s'appuie sur des technologies à code source ouverts. Les résultats sont encourageants par rapport aux approches existantes (p. ex. Vitesse d'exécution multipliée par deux sans augmentation du taux d'équipements en dysfonctionnement).

Abstract

With the expansion of Internet of Things (IoT) that relies on heterogeneous; dynamic; and massively deployed devices; Device Management (DM), which consists of firmware update, configuration, troubleshooting and tracking, is required for proper quality of service and user experience, deployment of new functions, bug fixes and distribution of security patches.

Existing Home and IoT industrial DM platforms are already showing their limits with a few static home and IoT devices (e.g., routers, TV Decoders). Currently, these platforms are mainly manually operated by experts such as system administrators, and require extensive knowledge and skills. Heterogeneity implies that devices have diverse compute and network capabilities. Dynamicity translates to variation of devices environments (e.g., network quality, running services, nearby devices). The massive aspect is reflected in fleets composed of billions of devices as opposed to millions currently.

Therefore, IoT device administration requires launching administration operations that assure the well-functioning of device fleets. These operations are to be adapted in terms of nature, speed, target, accordingly to devices current service requirements, computing capabilities and network conditions. Existing manually operated approaches cannot be applied on these massive and diverse devices forming the IoT.

To tackle these issues, our work in an industrial research context, at Orange Labs, proposes applying autonomic computing to platform operation and distribution. It aims to ensure that administration requirements of a device fleet are automatically fulfilled using the optimal amount of resources and with the least amount of execution errors.

Specifically, our contribution relies on four coordinated autonomic loops. The first two loops are responsible for handling fleet variation and update operations dispatching, while the remaining two others focus on vertical and horizontal scalability. Our approach allows automatic administration platform operation, more accurate and faster error diagnosis, vertical and horizontal scaling along with simpler IoT DM platform administration.

For experimental validation, we developed two prototypes : one that demonstrates the usability of our approach with Orange's industrial IoT platform for its piloting, while the other one demonstrates vertical scalability using extended open-source remote administration software. Our prototypes show encouraging results, such as two times faster firmware upgrade operation execution speed, compared to existing legacy telecommunication operator approaches.

Table des figures

Liste des tableaux

Table des matières

Introduction

Contexte

Dans un monde où le besoin en capacité de traitement de données s'accroît avec les usages qui évoluent sans cesse et se multiplient (e.g., réalité virtuelle, flux vidéo ultra haute définition, voiture connectée autonome, industrie 4.0), les systèmes se voient suivre cette tendance en devenant de plus en plus complexes (i.e., hétérogénéité, dynamicité des capacités de calculs, multiplicité des composants). De ce fait, un besoin d'administration s'impose. En effet, les briques logicielles et matérielles qui composent ces systèmes requièrent des opérations de maintenance (e.g., configurations, mises à jour, patch de sécurité, procédures d'assistance) déployées à distance pour assurer leur bon fonctionnement. C'est ce qu'on nomme "Administration Système", pouvant aussi être connue sous le nom de Device Management (DM) dans le cadre d'administration de flottes d'équipements domestiques de télécommunication ou de l'Internet des Objets (IdO).

Une mise à jour logicielle présentant des bugs peut causer des dysfonctionnements entraînant de lourdes pertes pour les sociétés ou propriétaires d'équipements. Ces dysfonctionnements peuvent être dus à des causes externes (logiciel développé pour les équipements présentant des bug ou à des errerus durant le processus de mise à jour). À titre d'exemple, nous pouvons citer Google qui en octobre 2019 a installé à distance une mise à jour du logiciel de leurs assistants vocaux 'Google Home' et Google Home Mini'. Cette mise à jour était fonctionnelle en apparence mais enclenchait, une fois l'appareil redémarré, une boucle infinie qui rendait celui-ci inutilisable de manière permanente. Cette mise à jour problématique obligea l'entreprise à remplacer tous les appareils défectueux à sa propre charge. Avec un parc de plus de 52 millions d'unités fin 2018 [31], si 1% de la flotte a été impactée, cette mise à jour aura couté le prix de 520 000 assistants ainsi que leurs frais d'envoi. Un autre exemple du monde industriel est celui de 'HP-Enterprise' et 'Dell EMC' qui ont dû déployer un logiciel correctif en urgence sur une tranche de leur flotte de disques à états solides (SSD : Solid State Drive). En effet ces derniers ont été usinés et envoyés aux clients avec un logiciel présentant un bug qui entraînait une panne irréversible avec perte de données à partir de 2768 heures de fonctionnement [18], [13]. Sans moyens d'administration système, pousser ce correctif n'aurait pas pu être aussi accessible et aurait été installé de manière manuelle, sur chaque machine de manière individuelle.

L'administration système de flotte ne sert pas qu'à régler d'éventuels dysfonctionnements sur le parc d'équipements. Elle peut également servir à améliorer l'expérience client comme dans le cas de Tesla qui envoie des mises à jour logicielles pour leurs voitures afin d'en augmenter les performances moteur et en ajoutant de nouvelles fonctionnalités [16].

Pour un opérateur de télécommunication comme Orange, l'administration système de flottes d'équipement est un moyen d'assurer que les équipements domestiques qu'il fournit aux clients (p. ex. Passerelles internet, décodeurs TV) sont en bon état de fonctionnement et ne nécessitent pas de remplacements fréquents. De plus, déployer de nouvelles fonctionnalités via des mises à jour logicielles permet d'améliorer l'expérience client. L'ambition d'Orange est de proposer de nouveaux services aux particuliers et aux industriels, par exemple, des services d'industrie 4.0 ou de ville intelligente et autres solutions innovantes. Ces dernières s'appuient sur l'IdO. Les équipements de l'IdO ont autant besoin d'administration système à distance que les parcs actuellement déployés par l'opérateur.

Plusieurs types d'équipements sont actuellement administrés à l'aide de plateformes logicielles de gestion à distance. La première cible de ces dernières fut les stations de travail et serveurs au milieu des années 90. Peu de temps après, avec l'émergence des accès internet résidentiels, le consortium industriel et à but non lucratif nommé "Broadband Forum" [10] fut fondé par des compagnies de télécommunications et d'informatique dans l'optique de définir les standards régissant les accès de type large-bande, communément appelés "broadband" (e.g., lignes ADSL Asymmetric Digital Subscriber Line) qui avaient pour vocation de remplacer les connexions par ligne commutée (e.g., accès internet 33.6K et 56K). Ce consortium identifia comme équipement requérant de l'administration à distance les routeurs et décodeurs de flux vidéo (e.g., télévision chiffrée, vidéo à la demande) publiant pour cela les spécifications pour un serveur de configuration automatique (ACS : Auto-Configuration Server) et les agents d'administration qui sont déployés dans les équipements. Au cours des années 2000, la démocratisation de téléphones et assistants personnels dans les flottes d'entreprises imposa leur prise en charge par ces plateformes d'administration à distance. Un standard adapté pour la gestion de ce type d'équipement fut proposé par l'OMA (Open Mobile Alliance [27]), OMA Device Management (OMA DM [26]). Il permet d'activer et désactiver certaines fonctionnalités, mettre à jour le périphérique, ainsi que la configuration de ce dernier. Ces solutions ont, par la suite, intégré les téléphones intelligents (ou smartphones) dans leur cible.

Durant les dernières années, avec l'avènement de l'IdO, plusieurs nouveaux types d'équipements ont fait leur apparition. Nous parlerons d'objets dans ce manuscrit pour nommer les équipements de l'IdO. Ces objets sont déployés pour des usages tant domestiques qu'industriels. On peut citer, à titre d'exemple, les stations météo, les capteurs de température, pression atmosphérique, ou de mouvement, ou encore les serrures et ampoules connectés. Pour l'industrie, viennent s'ajouter à la liste les compteurs intelligents, les voitures connectées, les alarmes à incendies ou périphériques de surveillance de santé. Ces objets ont besoin d'administration à distance

pour un fonctionnement optimal et un risque atténué d'interruption ou perturbation de service.

Problématique

L'objectif global de l'administration système est d'assurer le fait d'avoir un parc d'équipements à jour en accord avec les besoins actuels en termes d'opérations d'administration. Il s'agit par exemple, d'installation de firmwares et configurations, d'exécution de commandes, ou de récolte de données de log. Celles-ci sont nécessaires pour assurer un bon fonctionnement et une bonne qualité de service. En effet, le long du cycle de vie d'un équipement la disponibilité de nouveaux logiciels internes (ou firmwares) permet d'installer des services ou de régler des bugs. De plus, au moyen du déploiement de configurations, il est possible d'activer de nouvelles fonctionnalités ou de changer le comportement d'un équipement pour régler la fréquence d'échantillonnage de données ou activer la connectivité sans-fil par exemple. La collecte de données de log permet de détecter d'éventuels problèmes et d'y palier au moyen d'exécution de commandes à distance sur les équipements concernés.

Les objets composant l'IdO ont des caractéristiques qui rendent leur administration plus complexe que celles de routeurs domestiques ou de stations de travail. Il s'agit d'hétérogénéité (c.a.d., capacité de calcul, alimentation, localisation), de dynamicité (condition de réseau, de charge de calcul qui varient constamment), et le besoin de passage à l'échelle pour permettre la gestion de quantités massives d'objets. En outre, un besoin d'interopérabilité s'impose vu la nature multi-service, multi-protocole et multi-opérateur de l'IdO.

L'hétérogénéité implique d'avoir une administration système capable de prendre en charge de multiples parcs d'équipements ayant des capacités de calcul et des besoins d'administration différents en termes de fréquence, de type d'opérations, de tolérance aux fautes. Par exemple, une flotte de traceurs de positions GPS de palettes de marchandise, a besoin d'être reconfigurée avec une adresse de plateforme de récolte de données de suivi différente à chaque utilisation. Des stations météo feront plus l'objet d'opérations de type "Mise à jour firmware", pour colmater une faille de sécurité ou corriger des bugs. Dans l'administration système actuelle, le type d'opération et la périodicité sont peu variables. Cela est dû à l'homogénéité des flottes et la stabilité des objets comme les routeurs domestiques ou stations de travail, qui ne changent pas fréquemment de propriétaires et/ou d'utilisation. Les règles déjà complexes de mise à jour pour un type d'équipement le sont donc encore plus. Sans formalisation et automatisation, il est compliqué pour un administrateur humain de gérer les flottes de l'IdO à cause de cette caractéristique.

La dynamicité (des conditions réseaux, charge de calcul, disponibilité, d'alimentation, et périphériques connectés) requiert de l'administration système qu'elle soit adaptée de façon continue à ces variations. Cette caractéristique a des conséquences sur le besoin de lancer des opérations d'administration et aussi sur la manière de gérer l'exécution. Le changement d'état (présence d'autres objets connectés à proxi-

mité, dégradations d'indicateurs de performances) d'un ou plusieurs périphériques de la flotte peut induire un besoin de changement de configuration, du logiciel interne ou d'exécution de commandes à distance. Par ailleurs, la vitesse de déploiement d'une mise à jour de firmware ou de configuration dépend de la capacité du parc à la prendre en compte et l'accepter. Une perturbation réseau ou une incompatibilité de l'opération avec le contexte (logiciel ou matériel) engendre des erreurs d'exécution et pannes non récupérables (c.a.d. objets devant être remplacés). Ces anomalies peuvent se produire après un certain temps après la mise à jour. La plateforme d'administration système doit donc observer de manière active les périphériques durant les opérations et a posteriori.

Le besoin de passage à l'échelle est dû aux 7.6 milliards d'objets connectés que le cabinet Transforma Insights affirme être en activité. Ce dernier estime que ce chiffre atteindra 24.1 milliards en 2030 [6]. À titre de comparaison, le nombre de passerelles internet Orange actuellement en circulation est de 12 Millions (si on prend le principe une passerelle par ligne fixe) [14].

Ce besoin en administration système fait face à plusieurs verrous. Les solutions existantes à Orange permettant d'administrer les flottes d'équipements domestiques de télécommunication (c.a.d Passerelles internet domestique et décodeurs TV) sont pilotées manuellement et ont un besoin de re-configuration constante afin de répondre aux nouveaux services et équipements à déployer. Vu les défis induits par l'IdO en termes de dynamicité, d'hétérogénéité, le pilotage manuel des solutions d'administration de flotte n'est pas une piste viable. À cette complexité s'ajoute le besoin de passage à l'échelle imposé par la volumétrie des flottes d'équipements de l'IdO. Les solutions existantes sont centralisées et conçues pour des millions d'équipements. Elles n'ont pas la possibilité de gérer des nombres plus importants dans leurs états actuels.

Ces limitations de l'existant motivent cette thèse CIFRE où le défi que représente la problématique industrielle de l'administration de flotte d'équipement de l'IdO est relevé en explorant et évaluant des approches issues de la recherche. Les travaux existants en recherche académique sont majoritairement axés sur le déploiement et les services et peu sur la maintenance. Cet aspect n'est pas identifié ni formalisé. Dans une perspective de durabilité des équipements de l'IdO, nous explorons et étudions l'adaptation de l'administration des dits objets dans un environnement distribué et une volumétrie qui se chiffre en milliards d'unités [6].

Pour résumer, le besoin en administration système de l'IdO est un problème comprenant plusieurs défis et qui a été identifié par l'industrie comme un besoin clé pour l'amélioration des performances des périphériques déployés et l'augmentation de leur durée de vie. ABIResearch estime ce marché à plus de 20.5 Millions de dollars américains en 2023 [3].

Approche et Contribution

Afin de relever ces défis, l'application à ces plateformes d'administration à distance d'une gestion autonomique, c'est-à-dire capable de s'adapter d'elle-même de manière continue à son environnement, est étudiée, conçue, réalisée, et évaluée. En effet, ce type de gestion permet d'adresser la complexité de pilotage pour les administrateurs systèmes tout en proposant une adaptation constante et rapide aux éventuels changements / perturbations pouvant subvenir durant le fonctionnement des plateformes d'administration système. L'approche retenue est la conception d'un système autonomique avec des gestionnaires autonomiques multiples et coordonnés.

La première contribution, théorique, de cette thèse est une architecture logicielle de gestion autonomique basée sur quatre gestionnaires autonomiques collaborant pour un objectif commun : garder à jour un parc d'équipements de l'IdO en utilisant les ressources d'administration système disponibles de manière optimisée. Deux types de gestionnaires sont identifiés, d'une part ceux traitant de l'automatisation du pilotage des plateformes d'administration de flottes d'équipements et du suivi d'exécution des opérations d'administration système, et d'autre part, ceux en charge de la régulation du fonctionnement du système autonomique en fonction des ressources disponibles. Ces derniers assurent une adaptation de la vitesse d'exécution aux capacités des équipements à recevoir les opérations (Scalabilité Verticale) et une adaptation du déploiement des composants du système autonomique dans l'infrastructure disponible (Scalabilité Horizontale). Un des gestionnaires autonomique est multi-instanciable ; il s'agit de celui en charge de l'exécution et du suivi des opérations. Ce choix est motivé par le fait que ce gestionnaire soit le plus consommateur en ressources de calcul et de bande passante réseau. Cette caractéristique (multi-instanciation) permet un passage à l'échelle horizontal réalisé par le gestionnaire en charge de ce type de passage à l'échelle. Cette prise de décision influe sur le nombre d'instances de gestionnaire d'exécution dans les différentes parties de l'infrastructure matérielle disponible.

La deuxième contribution est expérimentale et de validation. Il s'agit d'un ensemble de prototypes permettant une mise en œuvre et validation de l'approche proposée avec des composants utilisés dans l'industrie. D'une part, il s'agit d'un système de gestion autonomique de pilotage de plateforme d'administration, couplé à la plateforme industrielle d'administration d'équipements de l'IdO, Live Objects d'Orange [29]. L'aspect mis en valeur par ce prototype, est la capacité de prise de décision du système complet (gestion autonomique et plateforme d'administration) lors d'évènements courants dans le cycle de vie des objets (c.a.d. départ et arrivée d'objets dans la flotte et disponibilité de mises à jour). D'autre part, le deuxième prototype, est un système de gestion autonomique couplé à des modules d'administration d'équipements de l'IdO (c.a.d. entité logicielle qui est plus bas niveau qu'une plateforme, et par conséquent permettant un pilotage plus fin de son comportement). Le choix de modules d'administration pour ce deuxième prototype est motivé par le fait que les plateformes industrielles sont sous la forme de boites noires qui n'offrent pas la possibilité d'avoir un pilotage de granularité supérieure : par exemple, la ré-

gulation de la vitesse d'exécution, la préemption d'opérations en cours, ou la récolte de traces d'exécution. Ces possibilités de réglages fins sont nécessaires au vu de l'objectif de ce prototype. En effet, il vise à montrer l'utilité d'un pilotage plus bas niveau que celui offert par les plateformes d'administration existantes, pour détecter les erreurs d'exécution, et en éviter la généralisation. La vitesse d'exécution des opérations d'administration est régulée en fonction des erreurs d'exécution. Ce démonstrateur utilise un protocole standard d'administration des objets, Lightweight Machine to Machine (LWM2M) [28] ainsi que des implémentations à code source ouvert de ce protocole, le projet Leshan [15].

Organisation du manuscrit

Le manuscrit est composé de sept chapitres, décomposés en trois parties. La première partie comporte deux chapitres qui traitent l'état de l'art de l'administration système à distance de flottes d'équipements ainsi que de l'informatique autonomique et certains de ses cas d'usage.

La deuxième, composée de trois chapitres, détaille la problématique, l'approche ainsi que les contributions, en donnant les spécifications de l'architecture du système autonomique d'administration à distance de flottes d'équipements proposée durant ces travaux de thèse.

La troisième et dernière partie, comportant les deux chapitres restants, aborde les contributions de validation expérimentales, réalisées durant cette thèse.

Ce manuscrit se conclut par un rappel de la problématique et des principales contributions avant d'élaborer les perspectives envisagées de ce travail.

Première partie

État de l'art

Chapitre 1

Administration d'un parc d'équipements

Sommaire

1.1 Cible et motivation

L'administration système représente le fait d'assurer l'évolution du logiciel et le bon fonctionnement, pendant un cycle de vie complet, d'équipements (passerelle internet domestique, contrôleur d'ampoules connectées) au moyen d'opérations à distance.

L'administration système de ces flottes permet d'effectuer des opérations distantes différentes en fonction du type d'équipements du parc géré. À Orange, l'administration de flottes est identifiée comme un domaine à part entière, nommé le Device Management (DM). En dehors d'Orange, dans le monde académique, ce domaine n'est pas identifié. Ceci nous a mené à effectuer un état de l'art au sein des équipes DM chez Orange, plus particulièrement à partir de la documentation des solutions utilisées, des retours des développeurs et des administrateurs de flottes. Celui-ci montre qu'il est possible de regrouper les fonctionnalités du DM en quatre catégories comme suit :

— Maintenance : mises à jour du système d'exploitation (et du logiciel intégré pour les objets de l'IdO), (p. ex. changer la version d'Android d'un téléphone, mettre à jour la distribution linux d'un serveur ou d'une machine virtuelle, ou installer une nouvelle version du logiciel intégré d'une station météo).
— Provisionnement : configuration initiale et re-configurations d'un équipement (p. ex. changer la clé pour la connexion sans fil, les heures d'activation d'une alarme, l'identifiant du propriétaire).
— Surveillance : collecte de données de sondes, soit applicatives (p. ex. Statistiques d'activation d'un capteur de présence, logs de fonctionnement d'une application sur un téléphone), soit matérielles (p. ex. température des composants d'un équipement).
— Assistance : exécution de commandes (p. ex. redémarrage) pour résolution de problèmes sur des équipements montrant des signes de perturbations (p. ex. consommation électrique anormale, absence de réponse).

Plusieurs types d'équipements peuvent faire partie des flottes administrées. Historiquement, durant les années 90, il était question d'administration de stations de travail, serveurs (ou Mainframes) [1]. Ces solutions propriétaires permettaient d'installer les mises à jour du système d'exploitation, de gérer la configuration réseau et les applications ainsi qu'à faire l'inventaire du parc. Peu d'années plus tard, les accès internet de type large-bande commencent à se démocratiser et un consortium composé de grands acteurs des télécommunications et de la radiodiffusion [10] (c.a.d. BroadBand Forum, BBF), s'est formé afin de définir les standards régissant ces accès internet et les services qui s'appuient dessus (p. ex. téléphonie IP, flux de télévision, vidéo à la demande ou VoD). Un de ces standards, (CPE [1] WAN Management Protocol, CWMP) plus connu sous le nom de son rapport technique : TR-069, définit la manière d'administrer les équipements domestiques permettant l'accès à ces services [4]. Par la suite, les premiers téléphones intelligents furent identifiés comme équi-

1. CPE : Consummer Premises Equipment

pements à administrer (p. ex. Blackberry, Nokia Symbian). Ce besoin vient du fait que ces équipements sont connectés à internet et embarquent des fonctionnalités se rapprochant de celles des stations de travail [62] (courrier électronique, messagerie instantanée, applications autres que la téléphonie). Par conséquent, ils doivent être correctement maintenus (et sécurisés dans le cas d'une flotte d'entreprise) pour bien fonctionner et éviter les risques de pannes ou de fuites de données. Depuis le début des années 2010, les périphériques de l'IdO commencent à prendre leur place dans la transformation numérique de la société. En effet, on estime leur nombre en 2019 à plus de 7.6 Milliards d'objets. L'IdO est caractérisé par des services complexes proposés par plusieurs acteurs avec des objets nombreux et hétérogènes [77], cela rend le besoin en administration plus conséquent qu'avec les trois autres types de flottes identifiés, plus homogènes et moins massives (c.a.d. stations de travail, équipements Télécoms de la maison, téléphones intelligents).

L'administration système de flottes est nécessaire pour de multiples raisons.

— Les équipements et objets connectés n'ont plus vocation d'être à usage court et unique [25]. Il est intéressant d'un point de vue économique et environnemental pour les fournisseurs et revendeurs de maximiser la durée de vie de leurs équipements déployés sur le terrain. De plus, il est impossible vu les ressources limitées disponibles pour leur fabrication, de déployer sans fin des millions d'objets connectés, sans assurer leur possibilité de réutilisation et leur maintenance. De plus, le DM offre la possibilité d'incorporer de nouvelles fonctionnalités aux équipements en mettant à jour leur logiciel interne tout en adressant les failles de sécurité qui sont découvertes [45], [75], [40].

— Les fournisseurs de service sont soucieux de la qualité de service (QoS : Quality of Service) offerte à leurs clients (particuliers ou professionnels). Pour la maximiser, le DM permet de détecter (grâce à la récolte de données de sondes) d'éventuels problèmes sur une partie (ou la totalité) du parc d'équipements. Par suite, il est aussi possible de tenter d'exécuter des commandes à distance pour résoudre ces derniers. Par ailleurs, ces fonctionnalités du DM permettent d'économiser les coûts de déplacements de techniciens sur site et en réduire l'impact carbone.

1.2 Niveaux d'Administration

Afin de caractériser l'administration système des différents parcs d'objets existants, nous définissons des niveaux d'administration, en se basant sur les opérations effectuées dans des parcs variés (p. ex. stations de travail, capteurs de température, routeurs domestiques). Ces niveaux sont exposés dans la figure 1.1.

En effet, nous considérons la mise à jour du logiciel interne (firmware) (p. ex. système d'exploitation pour des machines virtuelles et des stations de travail, microcode/micrologiciel pour un capteur de température ou une caméra de surveillance) comme étant l'élément de base nécessaire à la présence de capacités d'administration d'équipements. Le niveau suivant d'administration concerne la gestion de la

configuration des équipements (p. ex. luminosité, paramètres de conservation de l'énergie, mot de passe pour les réseaux sans-fil). Le niveau d'administration qui suit est la gestion des applications installées sous forme de 'packages' stockés et installés à partir d'un dépôt. Les équipements interagissent avec ce dernier via un gestionnaire de paquets (p. ex. Aptitude en Linux [5], Windows Store [7]). Le niveau le plus élevé est l'envoi de commandes élémentaires (p. ex. redémarrage, activation, désactivation, mise-en-veille) permettant de réaliser des actions de maintenance ou d'assistance à la résolution de problèmes.

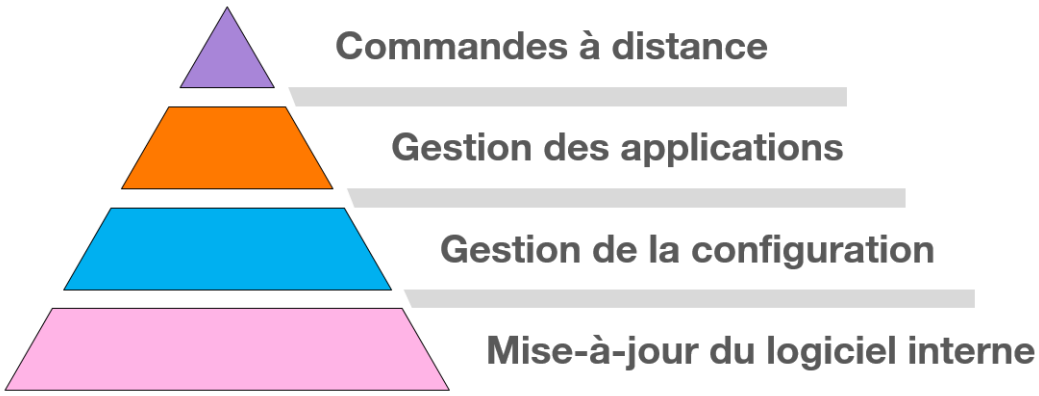

FIGURE 1.1 – Niveaux d'administration systèmes identifiés

Dans les prochaines sections, nous détaillons pour chaque type de flotte étudiée (c.a.d. équipements domestiques de télécommunications, objets de l'IdO, stations de travail, mobiles d'entreprises), les caractéristiques et les défis associés. Les flottes d'équipement domestiques de télécommunication et celles de l'IdO sont la cible principale de notre état de l'art car elles sont les flottes respectivement, existantes et visées, pour le futur d'Orange en termes d'administration de parc d'équipements dans le cadre de cette thèse CIFRE.

1.3 Administration d'équipements domestiques de télécommunications

Dans cette section, nous analysons l'administration système des équipements domestiques d'opérateurs de télécommunication. En effet, chez Orange, ce type d'administration système était le cœur de métier avant l'arrivée de l'IdO qui a diversifié les activités d'administration système de flottes. Les opérateurs de télécommunications, dans leurs branches grand public, administrent principalement des parcs d'équipements dits "de la maison" (Home Devices). Ce type inclut les passerelles internet domestiques (p. ex. LiveBox et FlyBox d'Orange, BBox de Bouygues Télécom), les décodeurs de flux TV (Set Top Box - STB, p. ex. décodeur TV d'Orange). La raison principale qui motive ce besoin d'administration est de limiter les frais de maintenance qui requièrent des interventions de techniciens tout en maximisant

la qualité de l'expérience utilisateur en réduisant les pannes et offrant de nouvelles fonctionnalités sans remplacer d'équipements (p. ex. au moyen de mises à jour logicielles). Ce type de parc est généralement administrable au troisième niveau : gestion des configurations. Certains périphériques font exception à cette règle quand ils utilisent des systèmes d'exploitation tiers (Décodeurs TV utilisant Android TV [12], et ayant donc accès aux capacités de gestion des applications de la plateforme de Google [30]).

— Les opérations de 'Maintenance' permettent de mettre à jour le logiciel interne des passerelles internet domestiques et des décodeurs TV. Cela permet d'activer de nouveaux services (p. ex. nouveau service Netflix ou YouTube), ou adresser des bugs considérés comme critiques (faille dans le noyau système entraînant un blocage puis redémarrage de l'appareil).
— Les opérations de 'Provisonnement' offrent la possibilité d'intégrer facilement de nouveaux équipements dans le parc (lors de la souscription en boutique les équipements sont identifiés et dès leur connexion au réseau chez le client, les paramètres et abonnements sont automatiquement télé-versés et activés). Ces opérations permettent aussi l'activation ou désactivation de services par l'opérateur (p. ex. souscription à des options payantes, participation à des phases de tests de nouveaux services).
— La 'Surveillance' permet aux opérateurs de récolter des données de fonctionnement qui d'un côté peuvent être source de détection de pannes réseau (p. ex. surcharge, impact de foudre sur une zone) ou d'équipements (p. ex. redémarrages fréquents de certains types de décodeurs), et de l'autre servir de source d'informations complémentaires pour le service client (p. ex. Savoir si tel équipement est connecté en Wi-Fi ou par liaison câblée RJ-45).
— L'assistance est un moyen pour le service client d'intervenir via des commandes à distance pour régler des problèmes chez les utilisateurs (p. ex. remettre le mot de passe Wi-Fi par défaut). Les administrateurs peuvent lancer des diagnostics sur une partie du parc pour en évaluer l'état.

1.3.1 Caractéristiques

Les flottes d'équipements des opérateurs de télécommunication sont majoritairement formées de passerelles domestiques et décodeurs TV. Chaque type d'équipement est géré par une plateforme séparée (comme indiqué dans la Figure 1.3). L'ensemble des équipements de ces flottes est accessible de manière permanente grâce au fait qu'ils soient alimentés en électricité et connectés de manière permanente au réseau Internet (sauf équipements physiquement éteints par les utilisateurs).

Une autre caractéristique clé de ce type de flotte est le fait que tous les équipements utilisent le même protocole d'administration à distance. Il s'agit de CWMP (plus connu par le nom de son document de spécification, TR-069 [4]). Le nombre d'équipements utilisant ce standard est aux alentours du milliard [9]. Ce protocole est centralisé et s'appuie sur l'architecture exposée dans la figure 1.2. Il s'agit d'un serveur de configuration (Auto-Configuration Server - ACS) qui est connecté au

système d'informations de l'opérateur (OSS/BSS/Policy et le centre d'appels clients dans la figure 1.2). Ce serveur, qui est configuré par des experts en DM, administre les équipements domestiques en passant par la passerelle installée par l'opérateur chez ses clients. Dans le cas où l'interface d'administration n'est pas hébergée dans l'équipement lui-même, on parle d'équipements "proxifiés" (cas d'une ampoule gérée par une passerelle internet). [2]

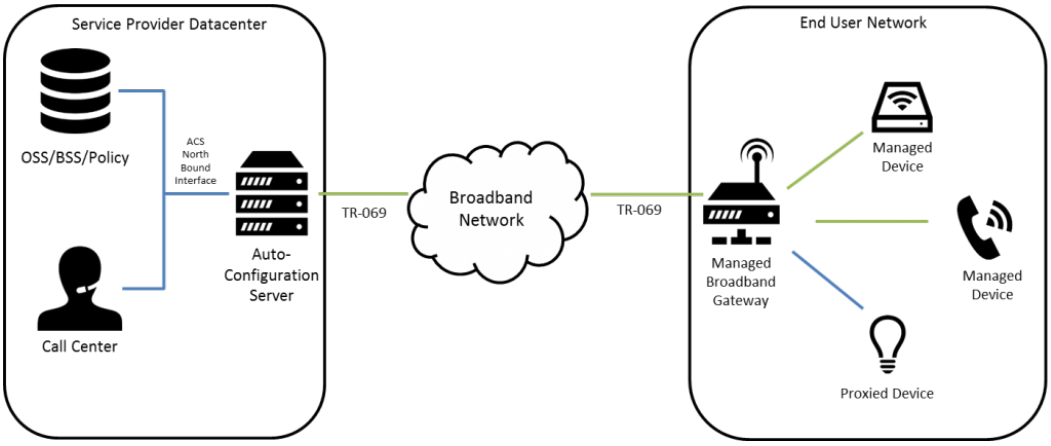

FIGURE 1.2 – Architecture d'administration avec le protocole CWMP (TR-069) [8]

1.3.2 Cas d'Orange

Du côté d'Orange, l'administration des équipements de ce type est réalisée à l'aide de multiples composants exposés dans la figure 1.3. La plateforme (Karma) d'administration est hébergée dans un datacenter privé. Cette dernière est connectée au système d'information (SI) d'Orange via un Intergiciel nommé Papyrus, qui se charge de traduire les "besoins métier" (p. ex. souscriptions d'abonnement ou d'options, remise de matériel, activation de services), en opérations d'administration système (p. ex. activer le module YouTube sur un décodeur TV déterminé). La plateforme se charge par la suite de traduire ces opérations en commandes (p. ex. changer la valeur du paramètre YouTubeActivated en 'Vrai' dans la configuration de l'équipement STB_202146) et de les exécuter sur les bons équipements.

2. des mots anglais Proxy et Proxied

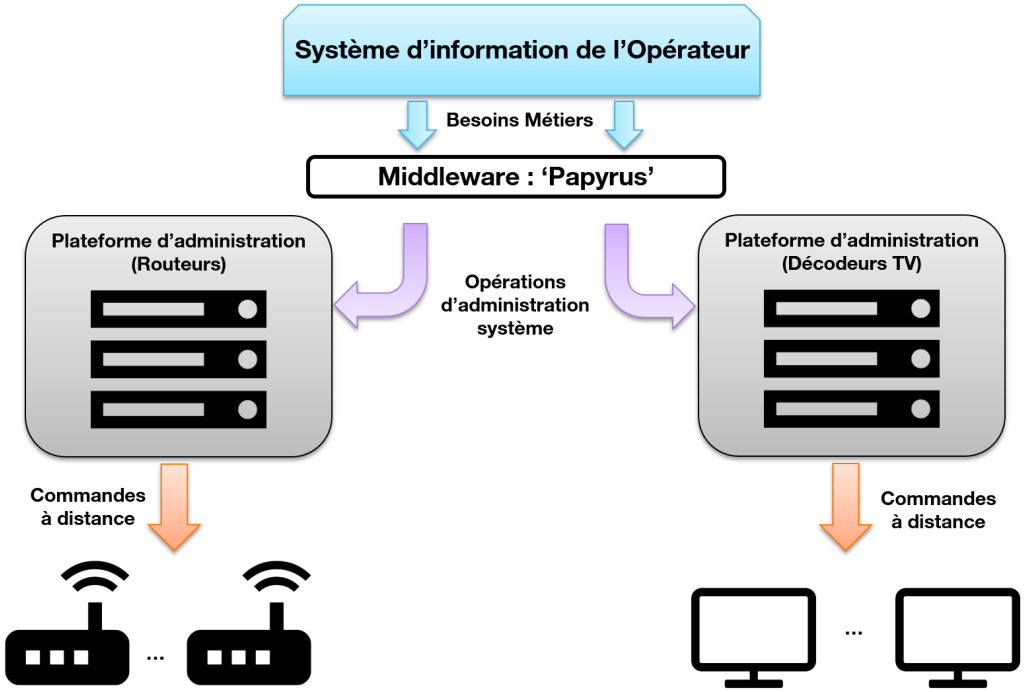

FIGURE 1.3 – Plateforme d'administration système des équipements de la maison (Cas d'Orange)

1.3.3 Défis

Même si le parc est homogène, un besoin de personnaliser les configurations en fonction des contextes clients-services des équipements se fait ressentir (p. ex. privilégier une vitesse de connexion maximale pour un client n'ayant pas souscrit au service TV, au lieu de viser un nombre plus bas d'erreurs de transfert, cela au détriment de la vitesse.) De plus, malgré des caractéristiques favorables à des opérations d'administration à faible taux d'erreur (c.a.d. alimentation secteur permanente, toujours accessible par le réseau), les parcs d'équipements domestiques de télécommunications restent exposés à des erreurs post-mise-à-jour (p. ex. logiciel ou configuration erronée entraînant des anomalies quelques heures après déploiement). Un défi est induit par l'utilisation du protocole CWMP, le contact entre l'équipement et le serveur d'administration système est toujours à l'initiative de l'équipement. En d'autres termes, les administrateurs ne peuvent faire des requêtes sur le parc. Le principe est que les équipements viennent périodiquement demander au serveur si des opérations sont à faire. Les solutions actuelles compensent cela en mettant une périodicité de contact basse, mais cela entraîne inévitablement une hausse de la charge pour le serveur qui doit gérer constamment les équipements qui le contactent pour des requêtes.

1.4 Administration d'équipements de l'internet des objets

L'internet des Objets est composé d'équipements qui ont des capacités de calcul différentes, des accès à l'énergie variables, des services multiples offerts par des opérateurs multiples, qui requièrent interopérabilité entre eux.

Les quatre fonctionnalités de base de l'administration de flottes identifiées en section 1.1 se traduisent comme suit dans le cas de l'administration de flottes d'objets l'IdO.

— Les opérations de 'Maintenance' permettent ici de mettre à jour le logiciel interne des équipements de l'IdO. Dans le cas d'objets ayant des capacités de calcul limitées, on procède à l'installation à distance d'un logiciel interne complet (quelques Kilo-octets) afin de changer la configuration (p. ex. couleur d'une Diode à Émissions Lumineuses), contrairement au cas nominal où on envoie simplement une nouvelle configuration (p. ex. code couleur en quelques octets).

— Les opérations de 'Provisonnement', comme pour le cas du parc d'opérateurs de télécommunications, offrent la possibilité d'intégrer facilement de nouveaux équipements dans le parc (p. ex. cas d'un nouveau capteur d'humidité déployé dans un champ qui sera affecté à la flotte déjà présente au moyen de sa géolocalisation). Une fois affecté, le système d'administration lui envoie la même configuration que les autres capteurs de la flotte (c.a.d. fréquence d'échantillonnage, adresse internet de la plateforme de remontée des données.)

— La 'Surveillance' dans ce cas, permet aux propriétaires de récolter des données (Logs) de fonctionnement (si l'équipement le permet), ou simplement des codes d'erreurs dans le cas d'équipements contraints (p. ex. 01 pour batterie épuisée, 02 pour dysfonctionnement physique).

— L'assistance pour ce type de parc représente principalement l'envoi de commandes de redémarrage pour tenter de remettre en marche un équipement en état d'erreur.

1.4.1 Caractéristiques

Le niveau d'administration possible avec ce type de flotte est variable en fonction des capacités de calcul de chaque équipement. En effet, un capteur de température n'est pas capable d'héberger un logiciel permettant son administration à distance contrairement à une station météo qui est assimilable à un mini ordinateur sous Linux. L'administration d'objets dits 'contraints' se fait au moyen de tierces parties (p. ex. équipement à proximité pouvant servir de mandataire pour la transmission des opérations d'administration).

Une caractéristique de ce type de parc est la fragmentation en termes de protocoles d'administration et de capacité de communication réseau. En outre, la sécurité et la fiabilité du processus de mise à jour logicielle sont rarement assurées pour ce

type de parc [76].

1.4.2 Défis

Les caractéristiques citées plus haut, rendent l'administration de ce genre de parc complexe.

1.4.2.1 L'hétérogénéité

L'hétérogénéité des équipements réside dans le fait que les flottes d'équipements sont composées d'objets interdépendants et variés, et donc n'ayant pas les mêmes besoins d'administration à un temps donné (contrairement à des flottes de décodeurs TV par exemple). A titre d'exemple, on peut considérer le cas d'une flotte d'équipements connectés dans un bâtiment de nouvelle génération comportant des équipements dits "intelligents" (p. ex. capteurs, ampoules, climatisation). Les capteurs d'humidité de certains entrepôts peuvent demander une reconfiguration pour en régler la sensibilité en fonction des risques sur les produits stockés. Les caméras n'ont pas à être reconfigurées fréquemment mais doivent être mises à jour au plus vite en cas de détection d'une vulnérabilité logicielle pouvant mettre en péril la sécurité du bâtiment. Une approche naïve serait de gérer chaque ensemble d'objets seul comme dans le cas des flottes homogènes d'opérateurs de télécommunications (c.a.d. décodeurs et routeurs domestiques), mais les dépendances entre services (donc entre objets) et la multiplications des flottes (qui requièrent chacune une expertise différente de la part des administrateurs systèmes), dans le cas de l'IdO, ne permettent pas d'appliquer ce type d'approche. Le besoin créé par l'hétérogénéité est celui d'une automatisation du pilotage de l'administration des multiples flottes qui prend en compte les spécificités de chacune.

1.4.2.2 La dynamicité

La dynamicité implique que les environnements et capacités de calcul varient durant le fonctionnement des équipements d'une flotte. Contrairement à des flottes qui sont toujours sur secteur et qui ne sont jamais en déplacement, la connectivité réseau varie en fonction de la géolocalisation (p. ex. présence en zones blanches). De plus à un temps donné, un service peut tourner sur un équipement en occupant des ressources de calcul pour une durée de temps limitée (p. ex. exécution d'un modèle de reconnaissance de déplacements lors de la présence d'humains dans une pièce). Cette variation de capacités de calcul et d'environnements requiert une adaptation constante de la manière dont on exécute les opérations d'administrations sur ces flottes (p. ex. vitesse d'exécution, Types d'opérations exécutables en fonction du risque d'échec, Ordonnancement des opérations en fonction de la priorité et de la criticité). En plus des implications impactant l'exécution des opérations d'administration, la dynamicité des équipements peut faire apparaître des anomalies post-exécution de la mise-à-jour (p. ex. redémarrages fréquents, équipements qui ne répondent plus) avec une certaine latence ou seulement en présence de conditions

bien déterminées (p. ex. fuite mémoire seulement si un service est en exécution, comportement anormal sur certains types de réseaux). Cette caractéristique implique un besoin constant d'observation et d'adaptation rapide à ces variations. Un administrateur système humain ne peut réaliser cela vu la complexité de la tâche.

1.4.2.3 Le besoin de passage à l'échelle

Le besoin de passage à l'échelle vient du fait que contrairement aux flottes existantes d'équipements, celles composées d'objets de l'IdO sont peuplées de milliards d'éléments. En effet, le cabinet Transforma Insights estime que ce chiffre atteindrait 24.1 Milliards en 2030 [6]. Cette volumétrie impose un besoin de passage à l'échelle étant donné que les flottes actuelles de passerelles internet domestiques et de décodeurs TV sont de l'ordre d'une dizaine de millions [14]. Les approches actuelles sont centralisées et commencent déjà à montrer leurs limites avec les flottes existantes. En effet, les travaux de recherche existants côté Orange trouvent leur origine de projets qui visent à traiter cette problématique dans le cas d'usage d'administration de flottes d'équipements domestiques.

Outre les défis cités plus haut, l'aspect sécurité est problématique concernant l'IdO. En effet, vu les contraintes matérielles de certains équipements [37], [73], le processus d'administration n'est pas aisément sécurisable. De plus, avec le nombre d'objets qui est en augmentation massive, la surface d'attaque croît en même temps [65].

1.4.3 Plateformes existantes d'administration de l'internet des objets

De multiples plateformes dites 'Plateformes IdO' (ou IoT Platforms) sont disponibles et exploitées par les entreprises. Ces dernières permettent de récolter et d'analyser les données remontées par une ou plusieurs flottes d'objets, ainsi que de lancer des opérations d'administration. On parle donc respectivement de 'Data Management' et de 'Device Management' (DM). Nous nous intéressons ici à la partie administration des objets connectés (DM) de ces plateformes.

Le tableau 1.1 résume les fonctionnalités des modules d'administration des plateformes de l'IdO. Nous avons opté pour les critères suivants :

- Capacité à mettre à jour le logiciel interne des équipements.
- Capacité à mettre à jour la configuration des équipements.
- Capacité à effectuer des opérations sur tout le parc (ou une partie conséquente) : par exemple, mise à jour du logiciel de toutes les ampoules connectées.
- Capacité à suivre les opérations en cours en termes de progression (nombre d'opérations effectuées par rapport au total), d'erreurs, d'application de politique de ré-essai.
- Protocoles d'administration à distance supportés.

Les plateformes IdO existantes ne représentent en effet que des briques logicielles

	Orange LiveObjects	AWS IoT	Azure Iot Hub	Bosch IoT	IBM Watson
Gestion du logiciel interne	✓	✓	✓	✓	✓
Gestion de la configuration	✓	✓	✓	✓	✓
Opérations massives	✓	✓	✓	✓	✓
Suivi des opérations	Progression	Progression	Progression	Progression	Progression
Protocoles supportés	Standard + Propri.	Propri.	Propri.	Multiples	Propri.

TABLE 1.1 – Comparaison des fonctionnalités des modules d'administration des plateformes de l'IdO

permettant d'effectuer des opérations d'administration système sur des flottes d'objets : gestion du logiciel et de la configuration et lancement d'opérations massives avec récolte de résultat d'exécution. Elles n'ont pas pour but de gérer la complexité de cette administration et ses contraintes. Elles permettent de lancer les opérations que l'administrateur a décidé de lancer sur la flotte d'équipements qu'il est chargé de maintenir. Le besoin d'adaptation du pilotage et de la configuration des plateformes d'administration système face aux contraintes et particularités induites par l'IdO n'est pas adressé par ces plateformes.

1.4.4 Plateforme du graphe des objets de l'IdO : Thing'In

Thing'in [32] est une plateforme de recherche du domaine de recherche Internet of Things d'Orange. Son objectif est d'aller au-delà des fonctionnalités des plateformes existantes citées dans la section précédente qui se focalisent sur le relayage des données des objets vers les applications métiers. Thing'In vise à fournir un graphe d'objets contenant une description complète de ces derniers, de leurs relations et de leurs environnements [61]. Les objets qui sont dans Thing'in ne sont pas systématiquement connectés et incluent des extincteurs ou salles de réunion par exemple.

L'administration système a connu ses débuts en ciblant les postes de travail et les flottes de téléphones mobiles. Dans les prochaines sections, nous détaillons les caractéristiques de chacune de ces flottes et les positionnons par rapport à l'administration de l'IdO cas d'usage visé par Orange aujourd'hui.

1.5 Administration de stations de travail

L'administration des stations de travail vise les flottes d'ordinateurs d'entreprises. Elle répond au besoin de maintenance et de bon fonctionnement de ces équipements. Il s'agit de surveiller le parc avec des métriques définies par l'équipe d'administration. De plus, il faut installer les mises-à-jour du système d'exploitation de ces machines et des applications. En outre, la politique de sécurité comme le droit d'accès à certains répertoires ou la possibilité d'installer des logiciels manuellement est gérée par l'administrateur de l'entreprise. Certaines solutions d'administration offrent une possibilité d'assistance à distance et permettent aux administrateurs de prendre la main sur l'équipement pour aider à la résolution de problèmes rencontrés par les utilisateurs.

1.5.1 Caractéristiques

L'administration de ce type de flotte est caractérisée par des mises-à-jour fréquentes (p. ex. un ensemble de correctifs pour Windows 10 chaque semaine [34]) d'une part, et un niveau d'administration élevé ('Gestion des applications' dans la Figure 1.1) d'autre part. Cela est possible grâce à la maturité des solutions existantes

depuis 1996 [1] et la puissance de calcul des stations de travail en constante augmentation grâce à la loi de Moore [64]. Un besoin de personnalisation des politiques d'administration en termes de fréquences de mises-à-jour, de restrictions et de configurations réseaux est imposé par les caractéristiques des terminaux composants ce parc. Ce besoin requiert une expertise des administrateurs systèmes et une configuration complexe des solutions d'administration. Cela implique un besoin d'abstraire les spécificités des utilisateurs de stations de travail en profil ou groupes et par la suite d'automatiser le pilotage de ces solutions afin de masquer cette complexité grandissante pour les administrateurs.

1.5.2 Positionnement par rapport aux objets de l'IdO

Vu qu'une partie conséquente des objets de l'IdO et en particulier les passerelles et équipements multi-fonctions fonctionnent avec un noyau Linux, il est possible de proposer d'assimiler l'administration d'un parc d'objets de l'IdO à celle d'une flotte de stations de travail à capacités de calcul très hétérogènes mais fonctionnant avec le même système d'exploitation : Unix. Cela serait possible si les équipements étaient statiques (p. ex. présents souvent dans le même environnement) et si le niveau d'administration était le même. En effet, pour le cas des flottes de stations de travail, il est au plus haut niveau, tandis que pour les objets de l'IdO, nous avons remarqué lors de l'étude de plusieurs périphériques et protocoles d'administration [28] qu'il s'agissait du niveau gestion de configuration, voire un niveau inférieur, 'Gestion du logiciel interne' pour les objets les plus contraints. Le noyau du système d'exploitation a beau être basé sur Unix, il s'agit souvent d'un micro-noyau adapté aux contraintes matérielles de calcul et de consommation énergétique de ce genre d'équipement.

1.6 Administration de mobiles d'entreprise

L'administration des téléphones d'entreprises vise les mobiles fournis (ou utilisés) par les employés d'une entreprise. Ce type de flotte est hétérogène d'un point de vue capacités matérielles, mais les disparités sont moins importantes qu'entre les objets formant l'IdO. En effet, suivant que le constructeur soit Apple ou un autre, le système d'exploitation sera iOS ou Android. Comme pour les stations de travail, les cas d'usages varient en fonction des utilisateurs.

Les téléphones grand-publics, bien que matériellement identiques à ceux des entreprises, ne sont pas gérés de la même manière que les flottes d'entreprises, c'est-à-dire en tant que parc. En effet, le rôle des constructeurs de téléphones est de fournir et d'assurer la disponibilité (et non pas le déploiement) du logiciel interne de ces téléphone (c.a.d. leur système d'exploitation Android ou iOS). Chaque utilisateur est libre de configurer ses politiques de sécurité, d'accès réseau, de mise à jour et d'installer les applications qu'il souhaite quelle qu'en soit la source.

1.6.1 Caractéristiques

L'administration de ces flottes de mobiles est caractérisée par un niveau d'administration aussi élevé que celui des stations de travail. Il s'agit du niveau "Gestion des applications". En effet, l'écosystème des téléphones mobiles (ou intelligents) est pourvu de ce qu'on appelle des magasins d'applications (Stores) qui offrent des possibilités d'ajout, de suppression et de mise à jour des applications installées sur ces terminaux. Les solutions d'administration existantes permettent la modification de la configuration des appareils, cela permet de mettre en place des droits d'accès à certaines fonctionnalités ou d'installer des applications ainsi que régler la politique de verrouillage de l'appareil. En outre, la configuration réseau est gérée par ce biais, comme le droit de se connecter à certains réseaux Wi-Fi avec des certificats ou l'utilisation de l'itinérance d'accès au réseau (utilisation des accès internet mobile à l'international).

Un des aspects cruciaux de ce type d'administration est la gestion de la sécurité. En effet, les parcs de mobiles sont un vecteur d'attaque pouvant avoir un sérieux impact sur la sécurité et la pérennité de l'activité des entreprises [71] [72], [52]. Une sécurité limitée peut entraîner l'utilisation du mobile comme point d'accès au réseau interne de la société, ou l'intégrer à un botnet. Il convient donc d'avoir une politique plus stricte concernant les codes de déverrouillages, les politiques d'installation d'applications, et le chiffrement de l'espace de stockage.

1.6.2 Positionnement par rapport aux objets de l'IdO

Comme les flottes de stations de travail, celles composées de mobiles ont la particularité d'avoir un niveau d'administration supérieur à celui des flottes d'objets de l'IdO, en l'occurrence celui de "Gestion d'applications". Les mobiles sont des terminaux qui sont presque constamment en ligne (moyennant connectivité aux réseaux télécoms) et joignables au contraire des objets de l'IdO qui dépendent de connectivités limitées et d'états de veille (pour la conservation d'énergie) qui viennent perturber la disponibilité de ces objets pour des opérations d'administration.

1.7 Comparaison des flottes d'équipements

Le tableau 1.2 résume les caractéristiques des différents types de flottes à administrer. Nous avons opté pour les critères suivants :

— L'interaction entre équipements représente le besoin de briques logicielles disposées dans des objets différents pour construire un service offert aux utilisateurs.
— La caractéristique 'Source d'énergie' représente la possibilité pour un équipement d'avoir soit une alimentation électrique infinie et constante (sous secteur) soit une source finie ou intermittente (sous batterie, parfois en charge).
— La 'fiabilité du réseau' concerne l'accès à un réseau fixe ou mobile offrant de hautes vitesses et un faible taux d'erreur.

	Domestiques	Mobiles	Stations de Travail	Serveurs	Objets de l'IdO
Interactions entre équipements	✗	✗	✗	✗	✗
Source d'énergie	Constante	Intermittente	Constante	Constante	Intermittente
Fiabilité du réseau	✓	Fréquent	✓	✓	✗
Homogénéité du parc	✓	✓	✓	✓	✗
Niveau d'administration	Commandes	Commandes	Applications	Commandes	Configuration*
Protocole de DM	Standard	Standard	Propriétaire	Aucun ou Propriétaire	Multiples et Variés

TABLE 1.2 – Caractéristiques de l'administration des différentes flottes d'équipements

— 'L'homogénéité du parc' représente la composition de la flotte à partir d'un ensemble d'objets de constructeurs, propriétaires et fonctionnalités différentes.
— Le 'niveau d'administration' permet de statuer sur les fonctionnalités d'administration de ce type de parc selon l'échelle que nous avons définie dans la Figure 1.1.
— Le 'protocole de DM' représente l'aspect standard et la multiplicité des protocoles utilisés pour l'administration de ce type de flottes.

Les flottes administrables sont différentes du point de vue complexité et nature des opérations à effectuer sur celles-ci. En effet, les interactions entre équipements sont quasi absentes contrairement aux flottes de l'IdO visées par ce travail de thèse, où des équipements appartenant à des propriétaires différents collaborent pour offrir un service aux utilisateurs.

Les flottes ayant des équipements sédentaires telles que celles des stations de travail ou équipements domestiques ont souvent un accès réseau fiable et une source d'énergie constante. Les flottes de l'IdO ne jouissent pas de cet avantage et les opérations d'administration sont donc plus exposées à des risques d'erreurs et un besoin d'adaptation à ces conditions.

L'homogénéité du parc en termes de type d'équipements est présente pour tous les types de flottes sauf celles de l'IdO. Les flottes de l'IdO, elles, sont très hétérogènes et requièrent prise en compte des particularités de chaque type d'équipements en termes de capacité de calcul et de connectivité.

Les niveaux d'administration sont variés en fonction du type d'équipements composant les flottes administrées. Les flottes domestiques (Passerelles Internet et Décodeurs TV) ont le niveau le plus élevé car les équipements sont la propriété de l'opérateur et sous son contrôle total. Dans le cas des mobiles d'entreprises, il est souvent question du niveau 'commandes' car le logiciel interne peut être mis à jour, de même que la configuration et les applications installées, et des commandes de redémarrage et de commande d'effacement à distance. Les stations de travail, quant à elles, sont au niveau commandes pour les mêmes raisons tout comme les serveurs dans un datacenter. Les équipements de l'IdO sont au mieux capables d'être gérés d'un point de vue configuration, voire seulement au niveau logiciel interne pour les objets les plus contraints. Par suite, pour exécuter un changement de configuration sur ces objets contraints, il faut parfois effectuer la mise à jour du logiciel interne en incorporant la nouvelle configuration (p. ex. cas de changement d'un texte sur un affichage à diodes lumineuses).

1.8 Synthèse

Les différents parcs d'équipements présentent des propriétés qui impactent la complexité de leur administration. En effet, tous les types de flottes ne présentent pas d'interaction entre équipements pour fournir un service commun au même niveau que les parcs de l'IdO. Une dépendance entre objets requiert des opérations

d'administration en fonction des services en cours d'exécution sur les équipements. Les sources d'énergie variables font que certains équipements ne sont pas toujours accessibles car physiquement éteints. Il faut faire la différence avec des équipements qui seraient dans un état d'erreur et ceux qui sont dans une phase de mise à en veille. La fiabilité du réseau peut induire des erreurs de transferts de commandes ou d'accusés de réceptions de celles-ci, cela impose de mettre en place un mécanisme de ré-essai performant et adapté au type d'équipement. Dans le cas de l'IdO, les parcs sont hétérogènes en termes de capacités et de protocoles d'administration et ont en plus un niveau d'administration fréquemment bas : Niveau 2 - Gestion de la configuration. Étant donné le contexte de la thèse (en entreprise dans une équipe de R&D en administration d'équipements domestiques de télécommunication), que les objets de l'IdO sont la cible des solutions d'administration de flotte d'Orange et les défis qu'apporte le paradigme de l'IdO, nous ciblons donc l'administration de ce type de flottes dans un contexte industriel.

L'équipe dans laquelle cette thèse a été réalisée est responsable du développement de la solution administrant les équipements domestiques (Niveau 4 : Commandes). Même avec ces capacités élevées d'administration des équipements, cette équipe rencontre un besoin constant d'adaptation de la solution existante et de son pilotage (et configuration).

Dans cette optique, nous nous sommes concentrés sur ces deux types de flottes : l'une servant d'existant (Équipements domestiques) venant progressivement à devenir 'Legacy', et l'autre (IdO) représentant le type de flotte d'équipements qu'il faudra administrer dans l'avenir, avec les défis d'hétérogénéité, de dynamicité et de volumétries impliquant un passage à l'échelle.

Les plateformes industrielles de l'IdO comportent des fonctionnalités d'administration système mais n'ont pas pour but d'adresser les défis présents pour l'administration de ce genre de flottes. Elles représentent une solution technique utilisée par les administrateurs qui décident de leurs opération et configuration de manière manuelle comme pour les flottes d'équipements domestiques de télécommunication. Vu les défis en termes d'hétérogénéité, de dynamicité et de de multiplicité des équipements, ce pilotage et cette configuration, tous deux effectués manuellement, n'est pas une approche viable.

Un besoin d'automatisation et d'adaptation continue émerge de notre analyse de d'état de l'art concernant les différents types d'administration système et en particulier celle concernant les flottes d'objets de l'IdO au vu des différents défis apportés par ce paradigme. Nous abordons dans le prochain chapitre une méthode d'automatisation : l'informatique autonomique.

Chapitre 2

Informatique autonomique

Sommaire

Dans ce chapitre, nous détaillons ce qu'est l'informatique autonomique, ses composants, son architecture et ses paradigmes, ainsi que les niveaux d'automatisation d'un système autonomique. La notion de gestionnaires autonomiques multiples est abordée ainsi que les modèles de coordination de ces derniers. Nous terminons ce chapitre en exposant certains cas d'usages de l'informatique autonomique.

2.1 Définition

Le concept d'informatique autonomique a été défini en 2001 par IBM [2], et détaillé par Kephart et al. en 2003 [49]. Il a pour but de rendre les systèmes capables de s'adapter à leur environnement, et ce pour :

- l'auto-configuration : capacité d'un système donné à reconfigurer de manière automatique ses composants des systèmes en accord avec des politiques de haut niveau.
- l'auto-protection : capacité du système à mitiger d'éventuelles pannes créant des incidents en cascade.
- l'auto-optimisation : capacité à continuellement tenter d'améliorer ses performances.
- l'auto-réparation : détection, diagnostic et mitigation de dysfonctionnements logiciels ou matériels.

Un système ayant les capacités citées ici est nommé système autonomique. Ce système est constitué des éléments suivants :

- Les éléments gérés : entités matérielles ou logicielles sur lesquelles il est possible d'effectuer des adaptations afin d'atteindre un ou plusieurs objectifs (p. ex. minimisation d'erreurs ou de consommation électrique, maximisation des indicateurs de performances).
- Les capteurs : entités matérielles ou logicielles permettant d'avoir des informations sur le système à gérer. Il peut s'agir par exemple d'un capteur de température ou de consommation énergétique d'un processeur ou d'un indicateur d'occupation disque.
- Les actionneurs : leviers de régulation sur les entités gérées. Par exemple un limiteur de consommation électrique d'un processeur ou une commande pour lancer des instances d'un logiciel.
- Le gestionnaire autonomique : entité responsable de l'adaptation continue des éléments gérés, au moyen des mesures et observation des capteurs. La mise en œuvre de ces décisions d'adaptation se fait via les actionneurs.

2.2 Origines

L'informatique autonomique trouve ses origines dans des travaux existants dans plusieurs domaines : il s'agit principalement de la biologie, la théorie du contrôle et l'intelligence artificielle.

2.2.1 La biologie

L'inspiration principale de l'informatique autonomique venant de la biologie est le système nerveux autonome qui être chargé de réguler les fonctions vitales dans le corps humain. Un individu ne contrôle pas volontairement ces fonctions pourtant essentielles (p. ex. glycémie, pression artérielle, digestion). Le principe est qu'en fonction d'information sur le corps et son environnement, le système nerveux s'adapte

aux éventuelles perturbations pour garder un état d'équilibre interne [60]. En informatique, les choix faits par un système autonomique sont plutôt des taches d'administration que les administrateurs ont choisi de lui confier.

2.2.2 La théorie du contrôle

Une autre source d'inspiration de l'informatique autonomique est la théorie du contrôle. La principale contribution tirée de cette source est la boucle de rétroaction exposée dans la figure 2.1. L'objectif de celle-ci est de réguler une valeur de sortie dans un intervalle de référence (ou consigne), et ce en fonction de perturbations physiques subies par le système. Une boucle de rétroaction est composée de trois éléments : le système à réguler, les sondes fournissant les éléments d'informations de la sortie et le contrôleur en charge de fournir la bonne entrée au système pour obtenir une sortie conforme (c.a.d. Figurant dans l'intervalle de référence).

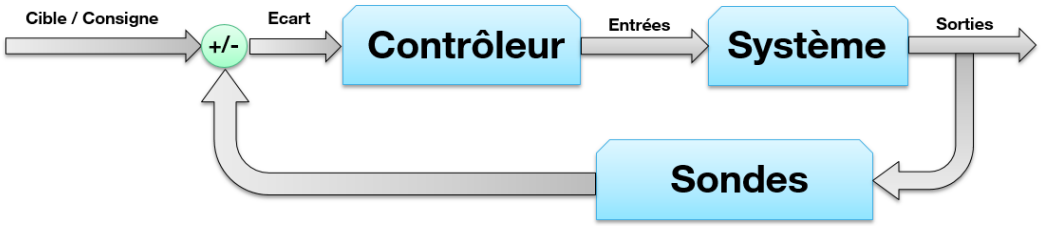

FIGURE 2.1 – Boucle de rétroaction

Cette boucle est reprise par l'informatique autonomique [49] sous la forme de la boucle MAPE-K définie dans la section 2.3.

2.2.3 L'intelligence artificielle

L'intelligence artificielle a pour objectif d'introduire des capacités de raisonnement afin de produire des systèmes ayant des capacités de raisonnement proche de celle des êtres vivants [42]. Cette discipline se rapproche donc de l'informatique autonomique sur cet aspect mais il est à noter que l'objectif de l'informatique autonomique est plus modéré dans le sens où il est question de donner à des systèmes des capacités d'auto-adaptation tout en restant sous la supervision d'un administrateur, là où l'intelligence artificielle vise à créer des systèmes 100% autonomes.

2.3 Gestionnaire Autonomique

Nous détaillons dans cette section le design de référence MAPE-K de boucles autonomiques et une possibilité de sa mise en œuvre avec du contrôle continu.

2.3.1 Boucle MAPE-K

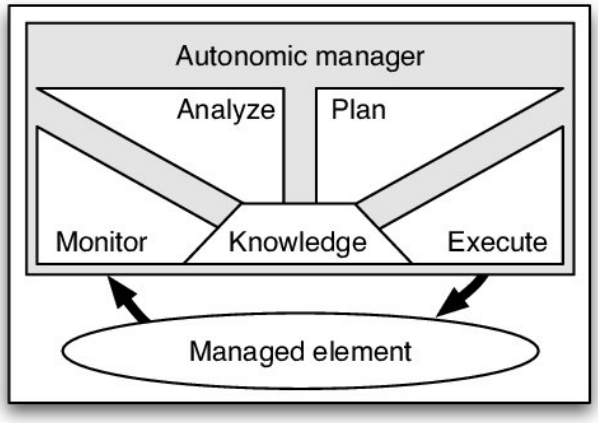

FIGURE 2.2 – Boucle autonomique MAPE-K [2]

Un gestionnaire autonomique peut être représenté sous la forme d'une boucle de rétroaction nommée MAPE-K [49]. Cet acronyme représente les phases par lesquelles le gestionnaire passe afin de réaliser une prise de décision concernant un éventuel besoin d'adaptation du système. Ces phases sont au nombre de quatre (4) : *Observation*, *Analyse*, *Planification* et *Exécution*, qui s'appuient sur une *Connaissance* concernant le système géré. Ces noms correspondent aux termes suivants en anglais (Monitor, Analyze, Plan, Execute, Knowledge Base - MAPE-K).

Un gestionnaire autonomique fonctionne en passant par les phases suivantes :

— (M) Observation : consiste en la collection, l'agrégation et le filtrage de données en provenance du système administré. Ces données servent d'entrée pour l'étape d'analyse.

— (A) Analyse : extrait des données d'observation un état du système et décide en fonction de ces dernières et éventuellement de l'historique, s'il y a besoin de re-configuration ou adaptation.

— (P) Planification : définit un plan d'action visant à faire transiter le système d'un état de perturbation détecté à un état stable souhaité selon les objectifs de haut niveau fixés par les administrateurs.

— (E) Exécution : exécute le plan d'action déterminé durant l'étape de planification.

Ces étapes se basent sur de la connaissance (K) pouvant être mise à jour durant n'importe quelle étape et sert de support de stockage et d'historisation des informations récoltés, décisions prises, ou encore ordres de régulation envoyés. Cette base contient les objectifs de haut niveau fixés par les administrateurs du système en question. Le gestionnaire peut fonctionner de manière continue ou répondre à des événements déclencheurs (fonctionnement événementiel). Les gestionnaires peuvent être multiples et partager des objectifs. Dans ce cas, il faut en assurer la coordination. Nous abordons ces notions dans la section 2.6.

2.3.2 Automatique pour l'informatique autonomique

L'architecture MAPE-K est une référence qui ne spécifie pas particulièrement de modèle ou de technique de prise de décision. L'automatique [63] [53], est un outil permettant de réguler les sorties du système en les gardant dans un intervalle en ajustant de manière continue l'entrée. Ce contrôle a pour particularité d'être précis et vérifiable formellement grâce à sa modélisation mathématique. Par contre, afin d'utiliser ce type de contrôle, il faut, au préalable avoir effectué une analyse comportementale très fine du système à contrôler afin de le modéliser sous-forme d'équations, de façon à concevoir une commande présentant des garanties formelles.

2.4 Niveaux d'automatisation d'un système autonomique

Selon le degré de maturité d'un système autonomique, il est possible de le classifier dans un des niveaux suivants définis par IBM [2]. Ces niveaux sont détaillés dans la figure 2.3.

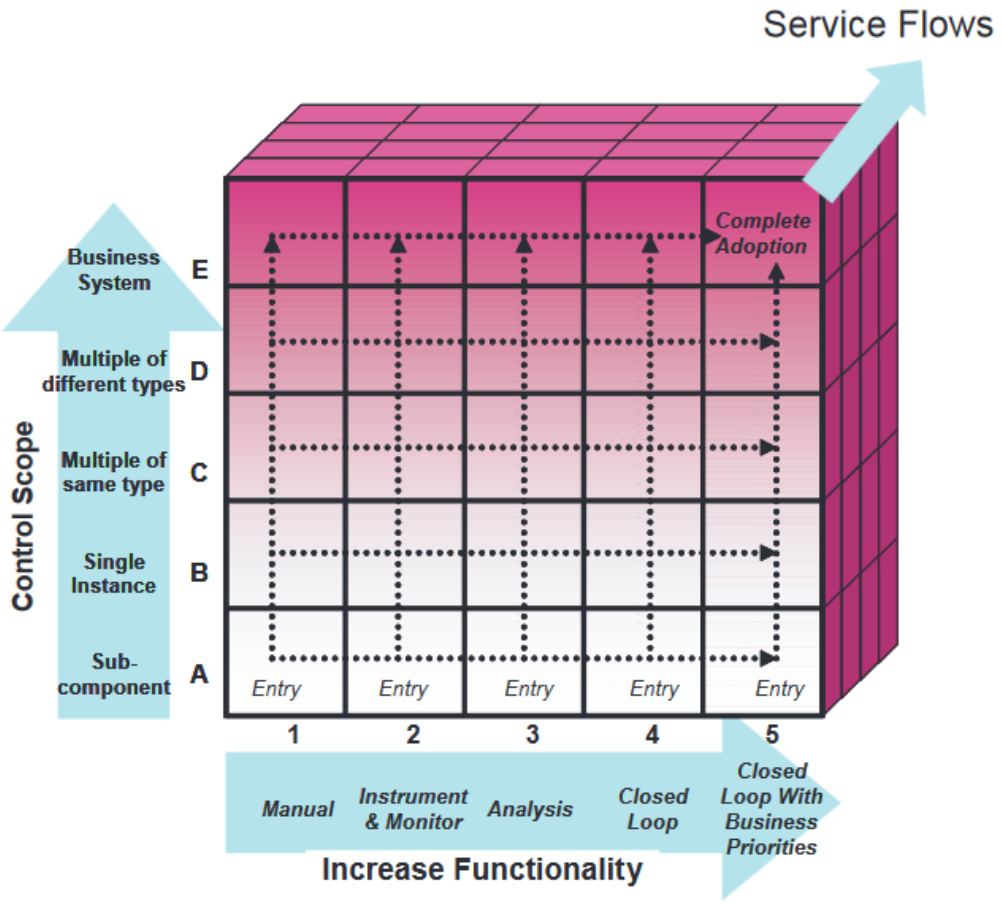

FIGURE 2.3 – Niveaux d'automatisation d'un système autonomique [2]

L'axe des abscisses traite du niveau de fonctionnalité assuré du contrôle. En effet, le premier niveau correspond à un contrôle manuel. Il est suivi du niveau qui correspond à un système intégrant des sondes remontant des données de fonctionnement qui sont agrégées et synthétisées pour analyse. Le niveau suivant concerne un système étant capable d'analyser et corréler les données collectées dans le but de proposer une configuration optimale. Le niveau d'après représente les systèmes en boucle fermée qui peuvent mettre en place des décisions d'adaptation de manière autonome. Le dernier niveau ajoute à ce qui précède la présence de règles métier de haut niveau qui régissent le processus de prise de décision. Ces règles sont élaborées par les administrateurs.

L'axe des ordonnées traite de l'échelle à laquelle l'automatisation est réalisée. Au premier niveau, il s'agit de sous-composants comme les applications d'un système d'exploitation d'un serveur. Le suivant concerne les ressources dites 'instances', qui représentent une application et son environnement logiciel et matériel. Le troisième niveau vise des instances multiples de ressources à administrer comme des serveurs de datacenters. Le niveau suivant concerne des instances multiples et hétérogènes. Le dernier a comme cible un système complet (logique métier ainsi que les applications et matériels nécessaires).

2.4.1 Cas d'Orange

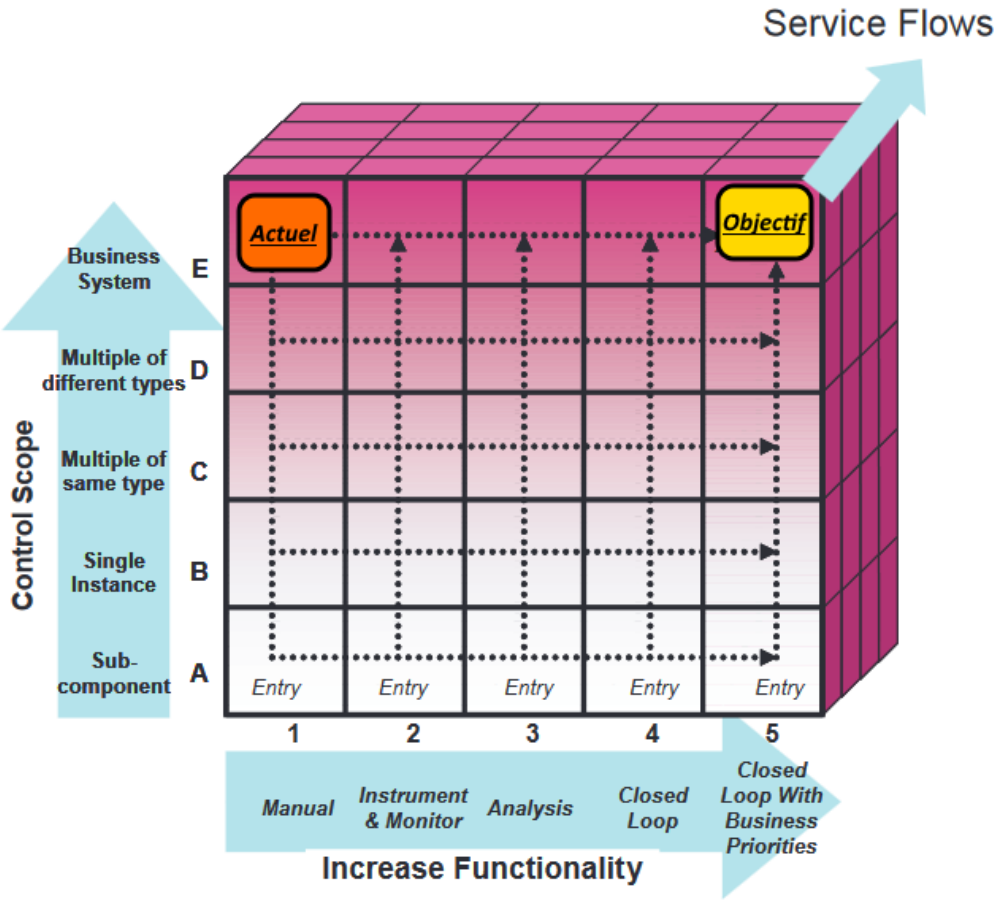

FIGURE 2.4 – Positionnement du système d'administration d'équipements d'Orange en termes de niveau d'automatisation [2]

Les solutions d'Orange pour l'administration de flottes d'équipements domestiques de télécommunication (citées en section 1.3), présentent un niveau de cible de pilotage (Control Scope) élevé (Niveau E dans la figure 2.4). Cela est dû à la maturité de la solution en termes de fonctionnalités, mais le pilotage reste manuel et opéré par une équipe ayant l'expertise de ces flottes d'équipements utilisées et déployées par l'opérateur.

Avec les défis induits par l'IdO d'un point de vue administration de flottes, en termes de dynamicité, d'hétérogénéité et de volumétrie, l'ambition d'Orange est d'aller vers un système en boucle fermée prenant en compte les spécificités métier (situé en haut à droite dans la figure 2.4). C'est dans cette optique que ces travaux de thèse ont été menés.

2.5 Architectures des systèmes autonomiques

Nous distinguons trois types d'architectures concernant les systèmes autonomiques : centralisées, distribuées, hiérarchiques.

Les architectures centralisées s'articulent autour d'un unique élément en charge d'administrer tous les composants du système et sur tous les aspects. Ce type d'architecture permet de limiter les incohérences dans le cas de gestionnaires autonomiques multiples, mais elles présentent une forte complexité de réalisation.

Les architectures distribuées [74] [36] sont basées sur une multitude de gestionnaires autonomiques ayant des objectifs élémentaires et relativement simples. Cela les rend faciles à mettre en œuvre et à maintenir. Ce type d'approche présente un point faible : le risque de comportements erratiques du système, dus à une erreur de coordination entre certains de ces multiples gestionnaires.

Les architectures hiérarchiques [74] [36] de la même manière que les architectures distribuées, sont construites au moyen de gestionnaires multiples. La différence se situe au niveau de l'organisation qui est pensée pour que les gestionnaires de haut niveau pilotent ceux de niveau inférieur. Ces architectures sont plus coûteuses en communications vu la nécessité de transmettre les ordres et de recevoir les résultats.

2.6 Coordination de gestionnaires multiples

Dans le cas d'architectures distribuées ou hiérarchiques, il est question d'automatiser la gestion d'un système au moyen de multiples gestionnaires autonomiques ayant des objectifs communs ou distincts. En administration système, le recours à plusieurs gestionnaires est dû à un besoin de passage à l'échelle [41], [47]. Ce type d'approche est basé soit sur une décomposition d'un seul et même gestionnaire en sous-gestionnaires multiples couvrant une ou plusieurs des phases MAPE(K) soit en instanciant le nombre de gestionnaires nécessaires, à un moment donné, pour administrer le système. Ces gestionnaires déployés se doivent d'être coordonnés afin d'assurer un bon fonctionnement du système autonomique.

2.6.1 Besoin de coordination

Les gestionnaires autonomiques sont des entités à part entière qui ont pour but d'adapter un ou plusieurs aspects du système géré. Certains objectifs poursuivis par des gestionnaires déployés au-dessus d'un même système peuvent être conflictuels. À titre d'exemple, on peut citer un besoin d'accélérer le calcul, qui implique l'augmentation de la fréquence effective d'un processeur. Cet objectif est en conflit avec celui de la minimisation de la consommation énergétique globale d'un système [70]. Pour ce faire, plusieurs modèles de coordination existent.

2.6.2 Méthodes de coordination de gestionnaires autonomiques

Les systèmes autonomiques étant souvent complexes et larges requièrent de multiples gestionnaires autonomiques devant être coordonnés. Une modélisation adé-

quate est nécessaire afin d'en assurer une gestion optimale. Dans le cadre de l'administration système, plusieurs approches de coordination de gestionnaires autonomiques multiples ont été proposées dans la littérature. Nous explorons dans ce qui suit la coordination à base de fonctions d'utilité, de règles, et de fonctions d'optimisation.

2.6.2.1 Coordination via une fonction d'utilité

[69], [43], [50], [51], [55], [44], et [58], proposent une coordination de boucles multiples à base de fonction d'utilité. Une fonction d'utilité est un modèle analytique qui permet d'associer une valeur générée à une quantité de ressources consommée. [69] compare deux approches d'allocation de ressources dans un datacenter, l'une basée sur la théorie des files d'attentes et l'autre sur du reinforcement learning. [43] adaptent de manière locale la gestion de flux dans un environnement distribué aux conditions réseaux variables et aux objectifs dits 'métiers' actuels. [51], [58] et [50] optimisent le ratio performance/watt dans un datacenter. [55] présentent deux boucles de gestion d'un cluster de serveurs. L'une estime le besoin en ressources en se basant sur les requêtes, tout en prenant en compte les Service Level Agreement (SLA) tandis que l'autre s'occupe de l'allocation des ressources à des nœuds de calcul. [44] se concentrent sur le refroidissement des datacenters en accord avec les objectifs de consommation énergétique, de températures et de performances.

2.6.2.2 Coordination à base de règles

De multiples approches de coordination sont basées sur des règles [59] [54]. [59] cherchent à gérer la consommation énergétique en prenant en compte les besoins en ressources des machines virtuelles. Les agents implantent des règles d'administration fixées par les administrateurs. Les décisions conflictuelles entre les agents sont résolues grâce à des priorités fixées de la même manière.

2.6.2.3 Coordination via une fonction d'optimisation

Les propositions de [48] et [68] se basent sur des fonctions d'optimisation composant un problème d'optimisation linéaire pour la régulation du système autonomique. [48] vise à stabiliser les boucles dans un réseau autonome. La synchronisation se fait comme suit : avant chaque prise de décision, chaque module consulte les autres pour validation. Par suite, chaque module de régulation est capable d'identifier les actions permettant d'améliorer la qualité de service générale en prenant en compte l'impact sur les autres modules de régulation. [68] adresse la stabilité dans un réseau autonome équipé de plusieurs boucles de contrôle. Il identifie trois problèmes : l'interaction de boucles de régulation, la résolution de conflits entre les boucles de contrôle et la synchronisation entre gestionnaires. La théorie des jeux est proposée comme approche pour la stabilité de la régulation autonome. La conception repose sur l'architecture GANA qui permet une instantiation hiérarchique des boucles de

contrôle. L'approche proposée permet également la résolution de conflits via un module, une fonction de synchronisation des actions [48].

2.6.3 Cas d'usages de coordination de gestionnaires autonomiques multiples

2.6.3.1 Approches génériques

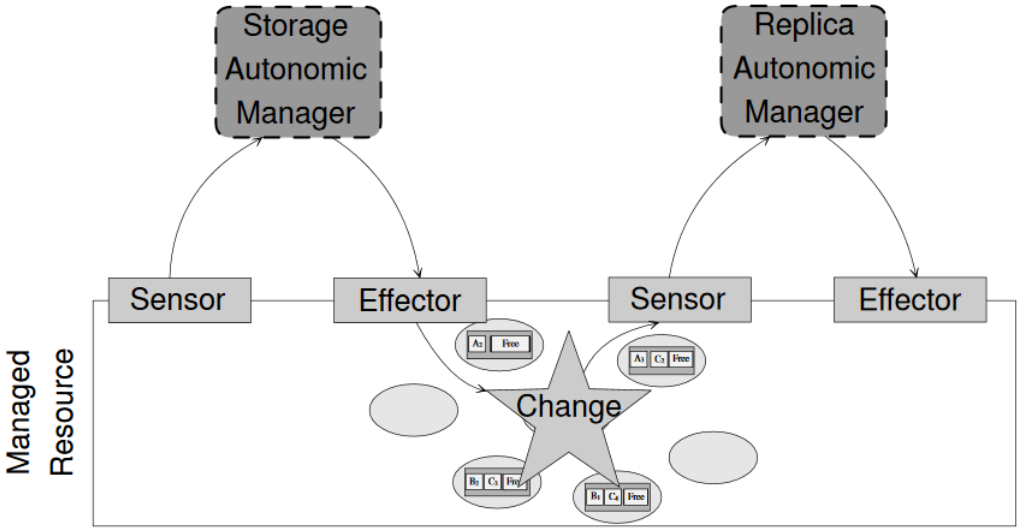

FIGURE 2.5 – Coordination basée sur une approche dite de 'Stigmergy' [36]

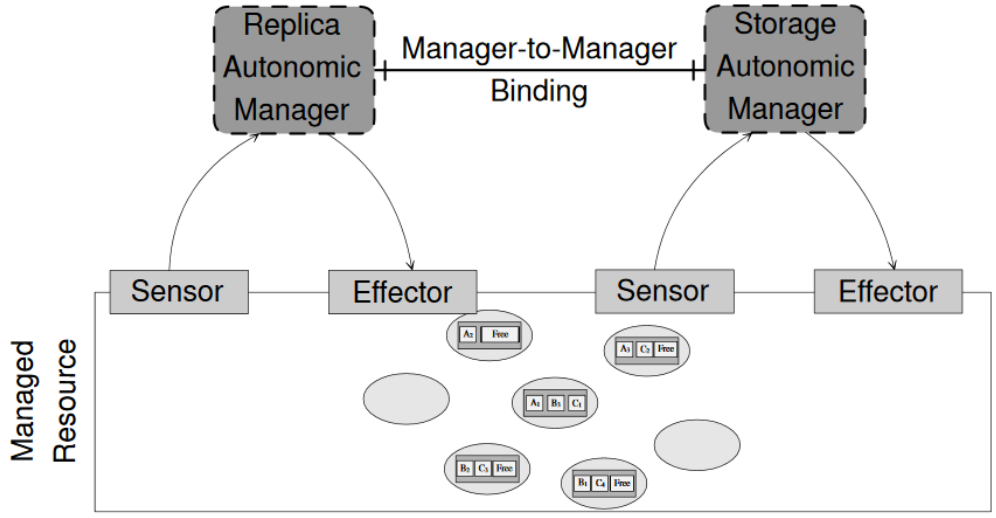

FIGURE 2.6 – Coordination basée sur une approche pair-à-pair [36]

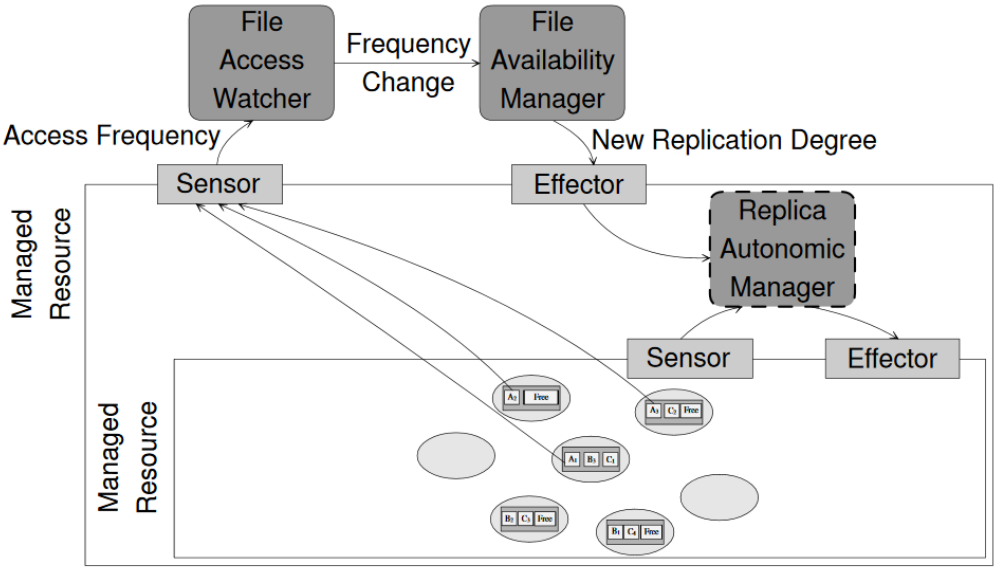

FIGURE 2.7 – Niveaux d'automatisation d'un système autonomique [36]

[74] et [36] décrivent la manière de décomposer les gestionnaires autonomiques et de les coordonner. [36] discute des possibilités de motifs de distribution de gestionnaires autonomiques. Ils proposent une librairie logicielle nommée DCMS permettant de développer des gestionnaires autonomiques, les déployer, et les gérer. Plusieurs méthodes de coordinations sont décrites dans ces travaux : la 'stigmergie', le pair à pair, et la gestion hiérarchique respectivement exposées dans les figures 2.5, 2.6 et 2.7. La première consiste en une méthode de synchronisation indirecte reposant sur le fait que les gestionnaires observent le même système et par la suite infèrent à partir des données d'observations les actions des autres gestionnaires. Le pair à pair (P2P) est une méthode de synchronisation basée sur les communications dites (1 à 1) entre les gestionnaires, tandis que la coordination hiérarchique se base sur une organisation pyramidale de ces gestionnaires autonomiques.

2.6.3.2 Gestion de ressources dans l'informatique nuagique

[46] et [38] se basent sur des gestionnaires autonomiques multiples afin de gérer les ressources d'un data-center. [38] adresse les conflits potentiels entre l'objectif d'assurance de qualité de service entre les fournisseurs d'infrastructure nuagique et les consommateurs et celui de consommation énergétique. Ils proposent un modèle de synchronisation de boucles autonomiques multiples traitant ce problème et appuient leur proposition avec une validation par simulation.

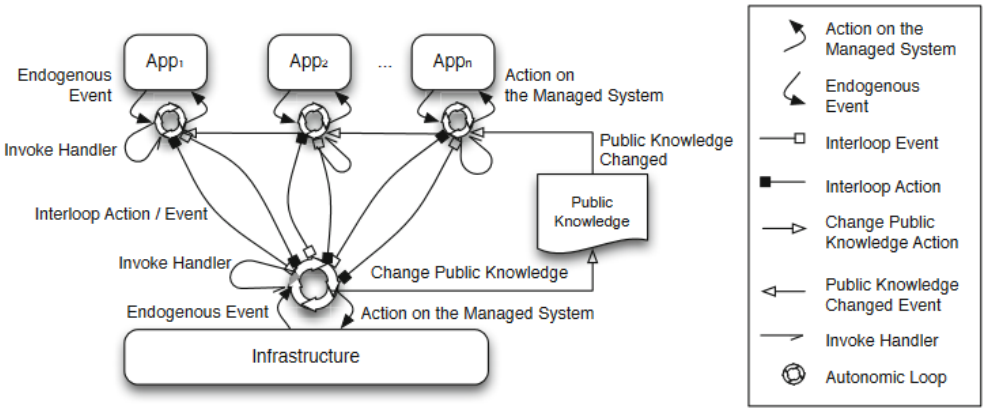

FIGURE 2.8 – Architecture autonomique à gestionnaires multiples coordonnés pour l'informatique nuagique [38]

La figure 2.8 détaille leur approche ; En effet, chaque changement demandé par une application est d'abord validé par la boucle autonomique en charge de l'application en question, qui se base sur une base de connaissance partagée avec le gestionnaire de l'infrastructure. Ce dernier prend en compte les contraintes énergétiques que les gestionnaires d'applications n'ont pas à gérer directement.

[46] utilise des techniques de contrôle pour la conception de contrôleurs de coordination. Celles-ci sont basées sur la programmation synchrone avec synthèse de contrôleurs discrets. Cette approche basée sur les méthodes formelles permet d'automatiser la construction et la validation des contrôleurs en question. Afin d'éviter la complexité liée à l'exploration de l'espace d'état pour la validation de ces contrôleurs, [46] suivent une approche à base de composant afin d'en réduire la complexité. Un des cas d'usage traité par [46] est le dimensionnement d'applications Web dans une infrastructure nuagique.

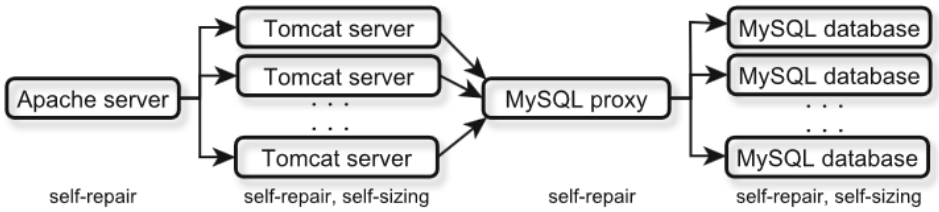

FIGURE 2.9 – Architecture autonomique d'une application web multi-tier [46]

La figure 2.9 détaille les objectifs pour chaque composant de cette application : La fonction d'auto-réparation est implantée pour assurer le bon fonctionnement du serveur d'hébergement (Apache) et le serveur mandataire (MySQL proxy), les serveurs d'applications (Tomcat) et de base de données (MySQL) sont répliquées en fonction de la charge mettant ainsi en œuvre l'auto-réparation et l'auto-

dimensionnement.

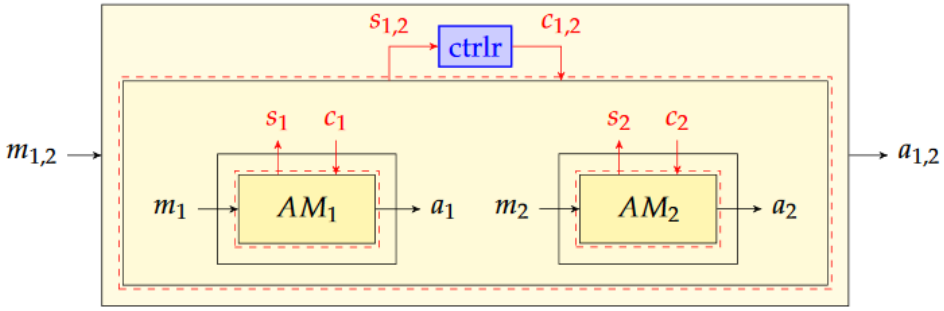

FIGURE 2.10 – Gestionnaires autonomiques coordonnés à l'aide d'un contrôleur discret [46]

La figure 2.10 détaille l'approche utilisée par [46] pour la coordination de gestionnaires multiples. Il s'agit de synthétiser un contrôleur discret (en bleu ctrlr) qui est responsable d'agir sur les interfaces de coordination disponibles au niveau des gestionnaires AM1 et AM2.

2.6.3.3 Gestion d'un bâtiment intelligent

Dans [66], [67], Sylla et al. proposent un intergiciel permettant la conception et le déploiement de système adaptatif fiable dans le cadre du bâtiment intelligent. Leur approche permet d'assurer un comportement excepté d'incohérences au moyen de mécanismes de transitions et d'exécutions fiables via un mécanisme de transaction. Leur approche supporte la théorie du contrôle continu ou discret, et la décision à base règles. Leur approche supporte aussi des gestionnaires autonomiques multiples coordonnés de manière parallèle ou hiérarchique.

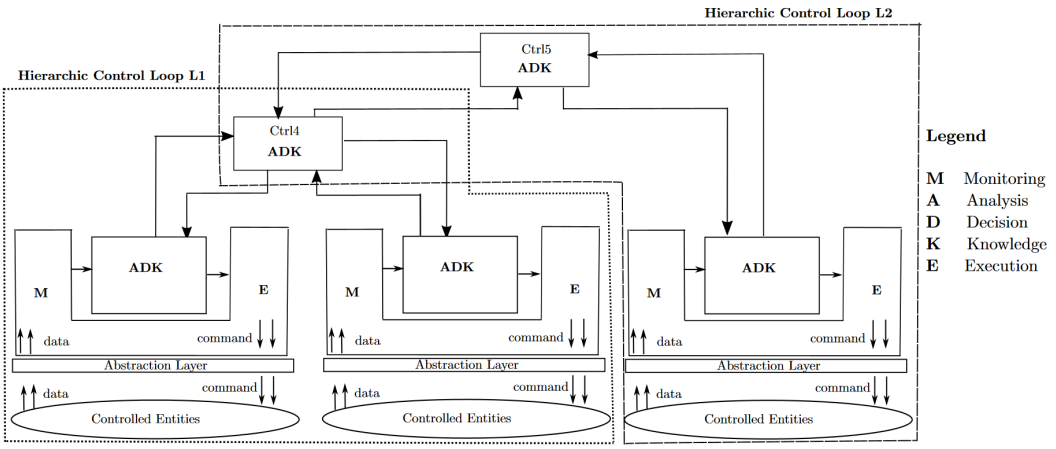

FIGURE 2.11 – Coordination de gestionnaires autonomiques parallèles [67]

La figure 2.11 détaille l'approche explorée par [67] pour la coordination de ges-

tionnaires autonomiques hiérarchiques. En effet, la prise de décision est prise au niveau des contrôleurs Ctrl4 et Ctrl5 qui englobent dans leurs éléments gérés une partie des gestionnaires autonomiques déployés dans le système.

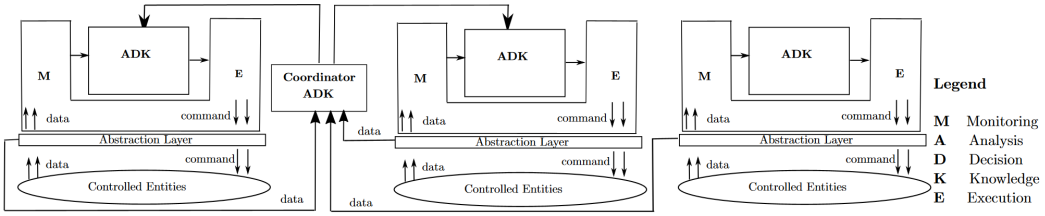

FIGURE 2.12 – Coordination de gestionnaires autonomiques hiérarchiques [67]

La figure 2.12 expose leur approche de coordination de gestionnaires autonomiques parallèles. En effet, un coordinateur prend en entrée des données d'observations et d'actions venant des gestionnaires et pousse une décision à ces derniers, afin d'éviter des prises de décisions conflictuelles et d'assurer la cohérence des actions décidées.

2.7 Conclusion

L'administration système est un sujet critique pour les entreprises qui assurent via ce procédé une bonne exploitation des ressources matérielles dont elles disposent en termes d'infrastructure, de flottes d'équipements, mobiles ou fixes, et cela représente un défi concernant les équipements de l'IdO vu la complexité amenée par ce paradigme (c.a.d. Hétérogénéité, besoin de passage à l'échelle, interopérabilité et dynamicité des environnements). Des besoins se font ressentir en termes d'adaptation continue et de régulation dans ce domaine. Ceci nous a mené vers l'exploration de la piste de l'informatique autonomique. Différents travaux de recherche abordent ce sujet et ses applications comme la régulation d'infrastructure nuagiques [46], [38] ou le contrôle de bâtiments intelligents [66], [67].

À notre connaissance, aucun des travaux de l'état de l'art ne propose d'automatiser l'administration de flottes d'équipements de l'IdO. C'est dans cette optique que nous proposons dans ce travail de thèse une architecture autonomique à base de gestionnaires multiples et coordonnés pour l'administration de flottes d'équipements de l'IdO.

Deuxième partie

Une architecture pour la gestion automatisée et distribuée de l'administration de flottes d'équipements

Suite à l'analyse de l'état de l'art, un besoin en automatisation et adaptation continue des plateformes d'administration système se fait ressentir. L'avènement de l'IdO rend cette discipline encore plus complexe à cause d'une hétérogénéité, d'une dynamicité, d'un besoin d'interopérabilité et d'une volumétrie importante. L'informatique autonomique est l'approche issue de la recherche identifiée pour adresser le problème industriel que représente l'administration de flottes d'objets de l'IdO. Cette partie comporte trois chapitres :

Le premier chapitre rappelle des définitions avant d'identifier les cas d'usages permettant d'établir la problématique visée. L'architecture générale de gestion automatisée et distribuée de flottes d'équipements est détaillée dans la dernière partie de ce chapitre.

Les deux chapitres suivants, détaillent respectivement les gestionnaires autonomiques en charge de l'automatisation de l'administration de flottes et ceux adressant l'aspect gestion des ressources.

Chapitre 3

Définitions, cas d'usage, problématique et architecture générale

Sommaire

Dans ce chapitre nous commençons par rappeler les définitions de base du domaine de l'administration de flotte, avant de détailler les cas d'usages que nous visons à traiter dans cette thèse. La problématique qui en découle est par la suite abordée ainsi que l'architecture générale de notre proposition. Cette dernière est basée sur quatre gestionnaires autonomiques coordonnés.

Dans la section suivante, nous définissons les concepts propres au domaine de l'administration de parc d'équipements utilisés dans notre proposition.

3.1 Définitions

Nous rappelons dans ce qui suit la définition du modèle de données et des équipements, dans le cadre de l'administration système. Nous détaillons aussi la représentation d'un équipement dans une plateforme d'administration : le jumeau numérique. D'un point de vue métier, il est nécessaire de définir les concepts d'opérations, de sous-opérations, de commandes qui sont explicités par la suite. Nous définissions, les erreurs pouvant arriver lors de l'exécution d'opérations d'administration d'une flotte et les notifications.

3.1.1 Équipement

Un équipement est un objet administrable à distance. Il est caractérisé par un ou plusieurs logiciels internes ('Firmwares') et sa configuration qui est matérialisée sous la forme d'une instance d'un modèle de données. Les logiciels internes des équipements sont généralement sous la forme d'un fichier binaire. Il s'agit dans notre cas d'usage, d'équipements connectés, de la maison (routeurs, décodeurs TV, contrôleurs d'ampoules connectées), ou autres objets de l'IdO comme les véhicules connectés, feux de circulation intelligents, traceurs de position GPS.

3.1.2 Flotte

Une flotte d'équipements ou parc d'équipements est un ensemble d'objets pouvant être administrés à distance par une équipe d'administration système. En fonction du niveau d'administration (Section 1.2) des objets, il est possible d'effectuer des mises à jour de configurations, de logiciels, d'applications ou l'exécution de commandes. Ces flottes peuvent être composées d'équipements du même type (passerelles internet domestiques), ou d'équipements appartenant à plusieurs types (Équipements de l'IdO : Ampoules connectées, traceurs de positions GPS, stations météo).

3.1.3 Modèle de données

Un modèle de données (Data-Model) est une représentation structurée des données concernant un équipement. Ces données comportent la configuration de ces équipements (p. ex. Wi-Fi activé) et des informations sur son état matériel (p. ex. Température ou charge CPU) et logiciel (p. ex. Services Activés). Dépendants du constructeur de l'équipement, les modèles de données peuvent être conformes à un standard ou propriétaires. Aujourd'hui un standard domine les équipements domestiques de télécommunications (Passerelles Internet, Décodeurs TV, Répéteurs Wi-Fi), il s'agit du TR-181 [33] définit par le Broadband Forum [10]. Ce standard fait partie de l'écosystème TR-069 [4] qui est fortement utilisé par les opérateurs de télécommunications pour l'administration de leurs parcs [9]. La figure 3.1 donne un exemple de la partie du modèle de données concernant un décodeur TV. Device :2.13 indique qu'il s'agit de la version "Issue 2 Amendement 13" du modèle de données. Les deux premiers rectangles à partir du haut de la figure "DeviceInfo" et "UserInterface" comportent des valeurs à mono instance "MemoryStatus" et d'autres à multi-instances "FirmwareImage".

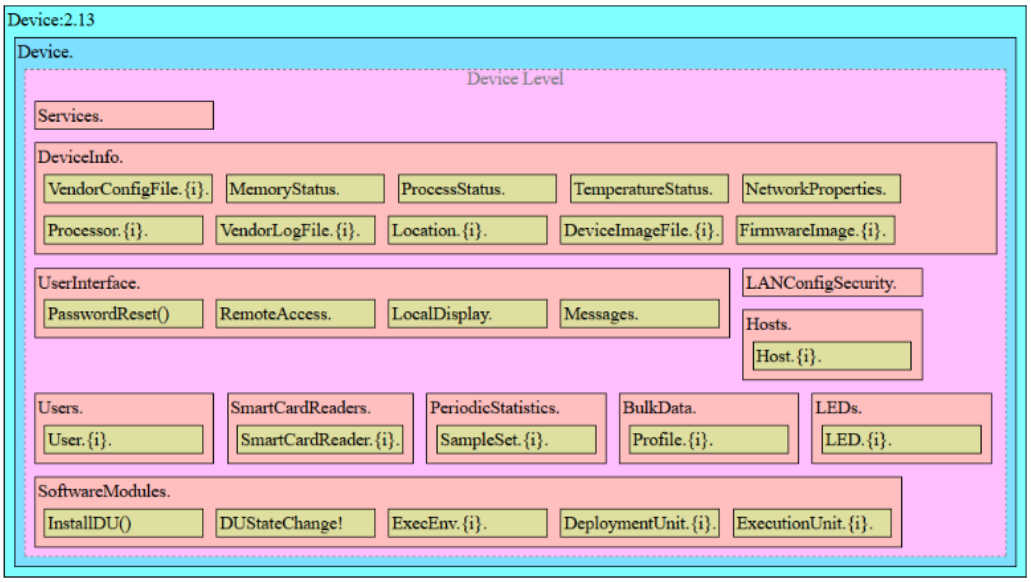

FIGURE 3.1 – Exemple de modèle de données TR-181 [33]

Concernant l'IdO, un standard d'administration d'objets est en développement depuis 2012 et est disponible depuis 2019 dans sa dernière version et en source libre. Il s'agit de celui de l'OMA (Open Mobile Alliance) - LWM2M (Lightweight Machine to Machine). Les spécifications de ce standard d'administration comportent un modèle de données et un modèle de description des objets nommé IPSO Smart Objects [20]. Grâce à cette description, il est possible de modéliser n'importe quel objet de l'IdO au moyen des spécifications.

Malgré l'existence de ce standard qui est capable d'être intégré à une multitude

d'objets de l'IdO, nous estimons que vu la multiplicité des fabricants d'objets et des services, qu'il est peu probable qu'un standard (LWM2M ou un autre) domine le marché. Notre approche doit être agnostique en termes de standards.

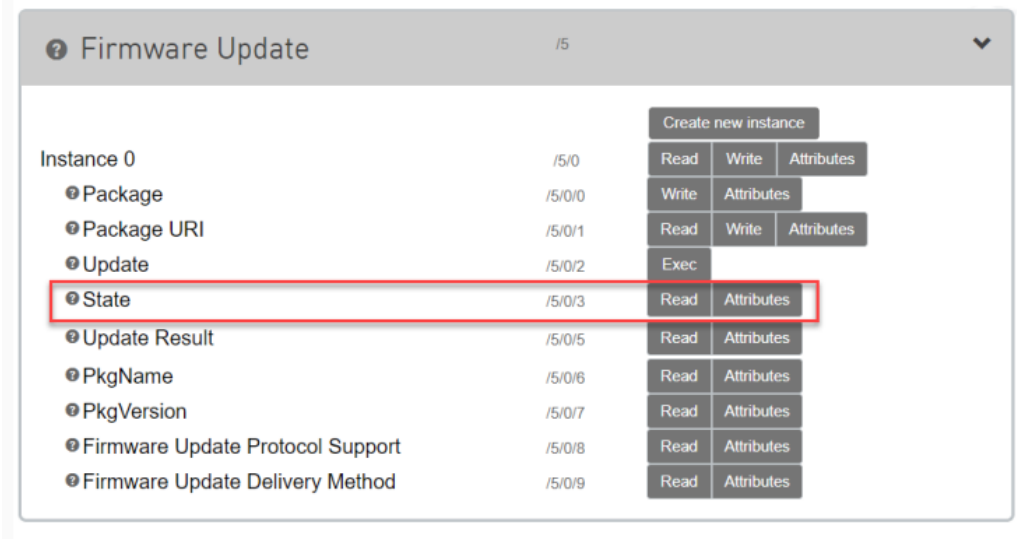

FIGURE 3.2 – Exemple de la partie "Firmware Update" du modèle de données LWM2M

La figure 3.2 montre une partie du modèle de données utilisé par le protocole LWM2M. Il s'agit ici de la partie concernant la mise à jour du logiciel interne (firmware update). Certains attributs composant ce modèle données sont de type informatifs : "PkgName", "PkgVersion", et on ne peut donc que faire de requêtes de lectures dessus. L'attribut "Package", représentant un logiciel interne à installer est accessible en lecture. L'attribut "Update" permet d'effectuer la mise à jour du logiciel interne vers celle écrite dans l'attribut "Package".

Nous intégrons le modèle de données dans une représentation d'un équipement, nommée jumeau numérique, détaillée dans ce qui suit.

3.1.4 Jumeau Numérique

Le jumeau numérique, dans le contexte de l'administration système, est une représentation de l'état d'un équipement, comprenant son modèle de données, fréquemment utilisée pour stocker l'état d'objets de l'IdO dans des plateformes de services. Microsoft dans sa plateforme de service IdO *IoT Hub* intègre le modèle de données 'Digital Twin' détaillé dans la figure 3.3 [23]. Amazon et IBM utilisent des concepts similaires. Il s'agit respectivement du *Device Shadow* et du *IBM Device Twin*.

La représentation utilisée par Microsoft est composée de deux parties : les informations invariables spécifiques à l'objet comme le nom, le fabricant, l'identifiant et les informations variables. Dans celles-ci en jaune, et gris dans la figure, sont

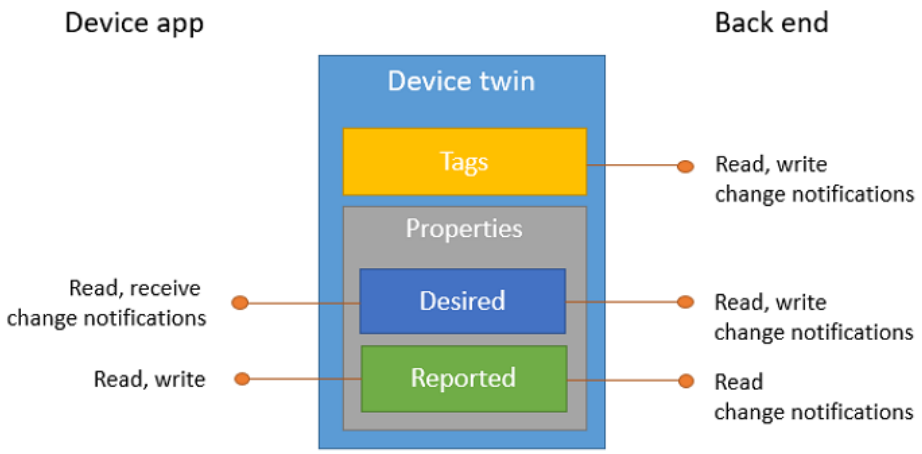

FIGURE 3.3 – Représentation Microsoft du jumeau numérique : Digital Twin [23]

détaillées les marqueurs (Tags) et les propriétés (Properties). Les marqueurs servent à affecter les objets à des groupes (Objets de Orange - Client M. Dupont) et les propriétés représentent l'instance de modèles de données de l'objet. La partie "propriétés" comme son nom l'indique, contient les propriétés demandées et celles qui sont émises par l'objet physique. Cette différenciation permet de séparer l'état réel de l'objet de celui que la plateforme d'administration s'attend à retrouver après avoir effectué des opérations sur celui-ci.

Nous nous basons sur la représentation de Microsoft qui est similaire à celle des plateformes d'Amazon et IBM pour modéliser un équipement de notre système d'administration de flottes d'équipements comme stipulé dans la figure 3.4.

3.1.5 Modélisation des traitements

Afin de modéliser les traitements effectués par une plateforme d'administration système sur les équipements d'une flotte, nous définissons les concepts suivants : opération, sous-opération et commande.

FIGURE 3.5 – Relations entre opérations, sous-opérations et commandes

La figure 3.5 détaille les relations entre les différentes entités que nous venons de définir. Une opération étant composée de multiples sous-opérations, la relation est donc de type 1-*. Celle-ci étant à son tour composée d'une liste de commandes, la relation est aussi de type 1-*. À chaque augmentation du niveau de granularité

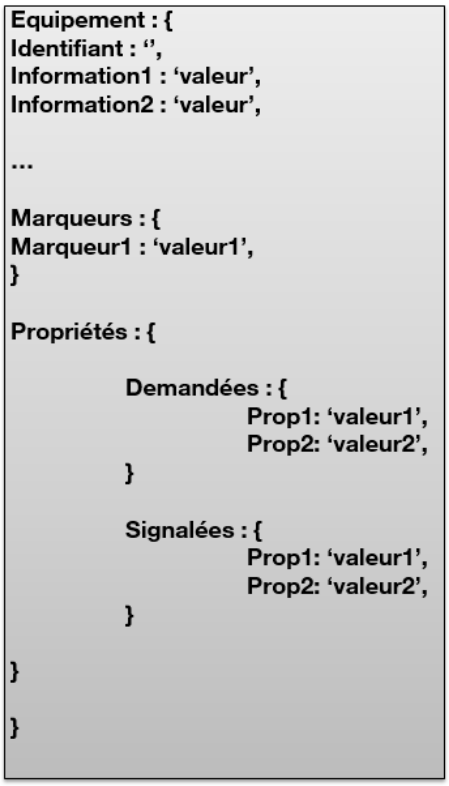

FIGURE 3.4 – Représentation d'un équipement dans le système autonomique

(opération, sous-opération puis commande), l'entité plus fine hérite des informations de l'entité mère tout en y ajoutant celles qui lui sont spécifiques.

3.1.5.1 Opération

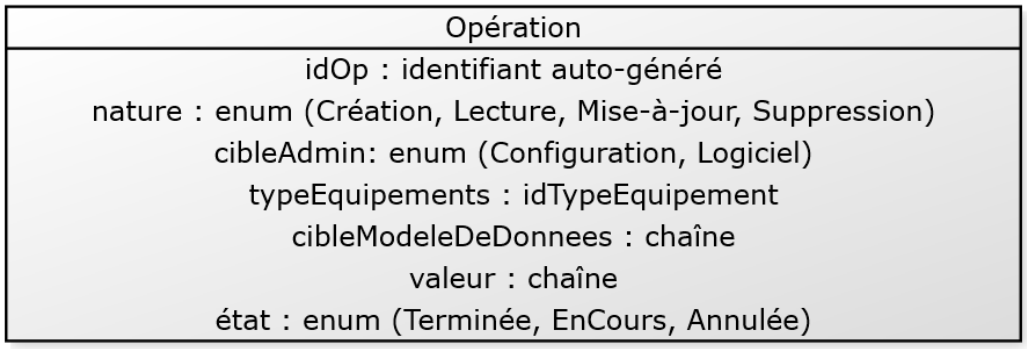

FIGURE 3.6 – Schéma de données d'une opération

Une opération est une lecture ou modification du logiciel interne ou de la configuration (exprimée sous la forme d'une instance de modèle de données). Comme indiqué dans la figure 3.6, une opération est caractérisée par son *identifiant* autogénéré, sa *nature* : une création, une lecture, une écriture ou une suppression. Elle est aussi caractérisée par sa *cible d'administration* : une partie du modèle de données (p. ex. Configuration du canal Wi-Fi) ou un logiciel interne (p. ex. Logiciel interne de secours), sa valeur à écrire (c.a.d. le fichier binaire dans le cas de mise à jour d'un logiciel interne ou la valeur du paramètre à écrire dans la configuration). Chaque opération vise un *type d'équipement* : par exemple, tous les contrôleurs d'ampoules Phillips Hue. Ce ciblage sur le type se fait en renseignant l'identifiant du type d'équipements visé.

Une opération ne cible qu'un type d'équipement. Elle a pour devenir d'être décomposée en un ensemble de sous-opérations qui elles sont caractérisées par une liste précise d'équipements comme cible.

Le champ 'cibleModeleDeDonnées' permet de préciser l'adresse dans le modèle de données dans la représentation de l'équipement où effectuer le traitement (p. ex. "Device/SoftwareModuleWirelessNetwork).

L'état de cette opération peut être soit terminée, en cours, ou annulée.

Par exemple : "Ecrire"*(nature)* dans "logiciel interne" *(cible d'administration)* **des contrôleurs d'ampoules Phillips Hue** *(type d'équipements)*, dans "Device/FirmwarePrimary" *(cible dans le modèle de données)*, "fichierMiseAJour.bin" *(valeur)*.

3.1.5.2 Sous-Opération

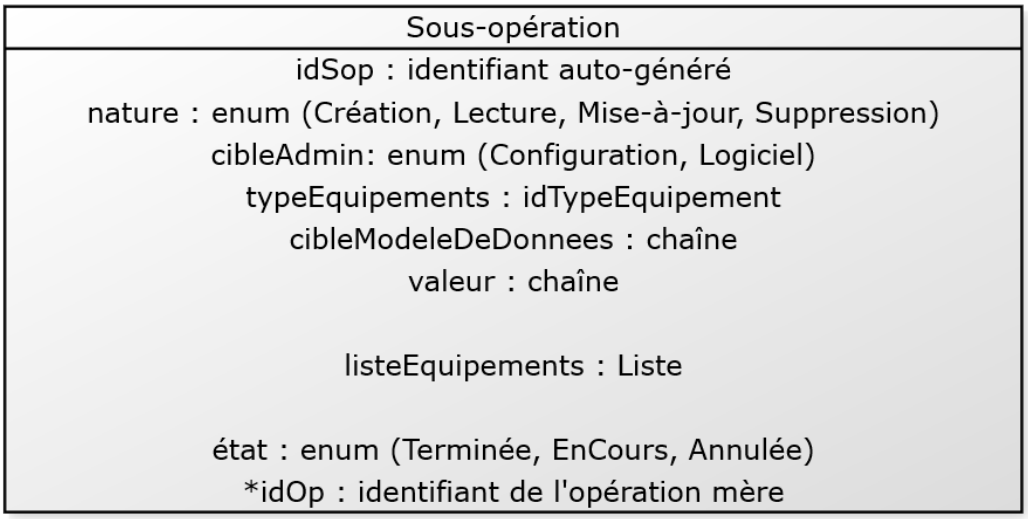

FIGURE 3.7 – Schéma de données d'une sous-opération

La figure 3.7 détaille le modèle de données d'une sous-opération. Elle cible une liste d'équipements déterminée contrairement à l'opération qui ne vise qu'un type d'équipement. Celle-ci est un fragment d'opération de la même nature, cible d'administration, cible dans le modèle de données et valeur que l'opération mère. Elle a pour devenir d'être traduite en un ensemble de commandes. Elle est aussi, caractérisée par un état (terminée, en cours ou annulée). Utiliser une liste d'équipements à ce niveau nous permets d'assurer ultérieurement la décomposition d'une sous-opération en un ensemble de commandes élémentaires (visant un seul équipement).

L'opération citée dans la section précédente serait décomposée en deux sous-opérations :

— "Ecrire"*(nature)* dans "logiciel interne" *(cible d'administration)* des contrôleurs d'ampoules Phillips Hue *(type d'équipements)*, dans "Device/Firmware-Primary" *(cible dans le modèle de données)*, "fichierMiseAJour.bin" *(valeur)* **sur Lampe1,Lampe2** ***(liste d'équipements)***.
— "Ecrire"*(nature)* dans "logiciel interne" *(cible d'administration)* des contrôleurs d'ampoules Phillips Hue *(type d'équipements)*, dans "Device/Firmware-Primary" *(cible dans le modèle de données)*, "fichierMiseAJour.bin" *(valeur)* **sur Lampe3,Lampe4** ***(liste d'équipements)***.

3.1.5.3 Commande

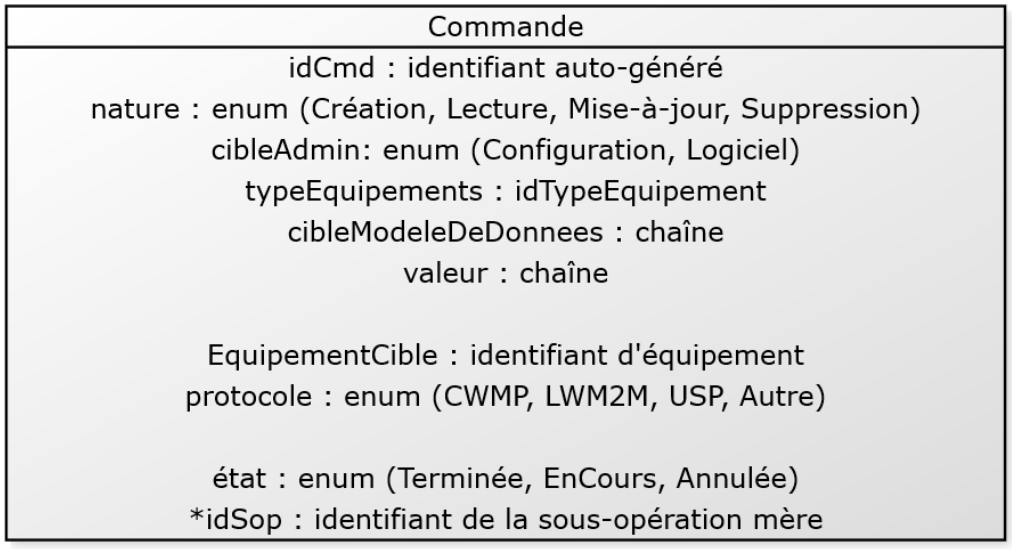

FIGURE 3.8 – Schéma de données d'une commande

Une commande est un composant d'une sous-opération qui cible un équipement déterminé, de la même *nature*, et *cible d'administration* que la sous-opération mère. Les *types d'équipements*, *cible dans le modèle de données* et la *valeur* sont conservés en concordance avec les données comprises dans la sous-opération mère correspondante. Elle est caractérisée par *l'identifiant de l'équipement ciblé* et son *protocole d'administration système* propriétaire ou standard. Ce protocole définit le format des messages échangés entre la plateforme d'administration système et l'équipement, le protocole de messagerie (p. ex. SOAP pour CWMP) ainsi que transport utilisé (p. ex. CoAP pour LWM2M, MQTT ou HTTP pour USP). Comme pour les opérations et sous-opérations, elle est caractérisée par son *état* (terminée, en cours ou annulée).

L'exemple cité dans Opérations et sous-opérations serait traduit en cet ensemble de quatre commandes :

— "Ecrire"*(nature)* dans "logiciel interne" *(cible d'administration)* des contrôleurs d'ampoules Phillips Hue *(type d'équipements)*, dans "Device/Firmware-Primary" *(cible dans le modèle de données)*, "fichierMiseAJour.bin" *(valeur)* **sur Lampe1** ***(équipement cible)*** **avec le** ***protocole*** **LWM2M**.

— "Ecrire"*(nature)* dans "logiciel interne" *(cible d'administration)* des contrôleurs d'ampoules Phillips Hue *(type d'équipements)*, dans "Device/Firmware-Primary" *(cible dans le modèle de données)*, "fichierMiseAJour.bin" *(valeur)* **sur Lampe2** ***(équipement cible)*** **avec le** ***protocole*** **LWM2M**.

— "Ecrire"*(nature)* dans "logiciel interne" *(cible d'administration)* des contrôleurs d'ampoules Phillips Hue *(type d'équipements)*, dans "Device/Firmware-Primary" *(cible dans le modèle de données)*, "fichierMiseAJour.bin" *(valeur)*

sur Lampe3 ***(équipement cible)*** **avec le** ***protocole*** **LWM2M**.

— "Ecrire"*(nature)* dans "logiciel interne" *(cible d'administration)* des contrôleurs d'ampoules Phillips Hue *(type d'équipements)*, dans "Device/Firmware-Primary" *(cible dans le modèle de données)*, "fichierMiseAJour.bin" *(valeur)* **sur Lampe4** ***(équipement cible)*** **avec le** ***protocole*** **LWM2M**.

Lors de l'exécution de commandes, deux types d'erreurs peuvent survenir : les erreurs d'exécution et les erreurs d'infrastructure. C'est dans la prochaine sous-section que nous les abordons.

3.1.6 Erreur d'exécution

Au niveau des plateformes d'administration et des équipements, deux types d'erreurs peuvent subvenir lors de l'exécution des commandes composant une sous-opération : des erreurs *critiques* et *non critiques*. Les erreurs critiques peuvent subvenir dans le cas où la commande serait erronée, c'est-à-dire comportant un logiciel interne ou une configuration entraînant un dysfonctionnement des équipements. Cette anomalie peut survenir instantanément lors de l'exécution ou avec une certaine latence. Les erreurs non critiques surviennent lorsque les équipements appliquent les changements attendus avec un temps de latence conséquent après réception des commandes d'administration système. Ça se traduit par un temps de réponse élevé de la part de ces équipements pouvant être symptôme d'anomalies.

Une erreur critique fait que la commande est définitivement considérée comme en échec tandis qu'une erreur non critique permet de retenter la commande. En cas de nombre trop élevé d'erreurs critiques ou non critiques une sous-opération est considérée comme ratée. De même concernant le taux de sous-opérations échouées : s'il est trop élevé, l'opération sera considérée comme en état d'échec et abandonnée.

Il convient d'éviter ces erreurs en surveillant automatiquement et consentement les équipements durant l'exécution des commandes et un laps de temps après pour détecter les erreurs latentes. Les erreurs d'exécution n'étant qu'un des deux types d'erreurs pouvant arriver durant l'administration d'un parc, nous abordons dans ce qui suit les autres erreurs concernant l'infrastructure hébergeant les plateformes d'administration de parcs d'équipements.

3.1.7 Erreur d'infrastructure

Deux types d'erreurs d'infrastructure peuvent apparaître au niveau de l'infrastructure hébergeant les plateformes d'administration :

— Erreurs applicatives : Plateformes d'administration système donnant des métriques de fonctionnement dégradées ou de plus en plus longues : par exemple le temps de réponse des API.

— Erreurs matérielles : Saturation du processeur, de la mémoire et de la charge réseau du matériel hébergeant les plateformes d'administration système.

Ces erreurs sont donc à éviter et à réduire au maximum en adaptant le nombre de sous-opérations (et par suite de commandes) exécutées à la capacité de l'applicatif

et de l'infrastructure.

3.1.8 Notifications

Comme stipulé au début de ce chapitre, les mises à jour de configurations ou de logiciels internes sont déclenchées automatiquement dues à une modification de la composition de la flotte ou à la disponibilité de nouvelles configurations et mise à jour, ou manuellement au travers des administrateurs systèmes. Dans le cas automatique, il est question de réception de *Notifications* de disponibilité de mises-à-jour.

Nous définissons par 'notification', l'alerte reçue par le système concernant la disponibilité d'une nouvelle configuration ou d'un nouveau logiciel interne. En fonction du type de notification, les données présentes sont différentes. La figure 3.9 détaille la structure de donnée utilisée dans les deux cas.

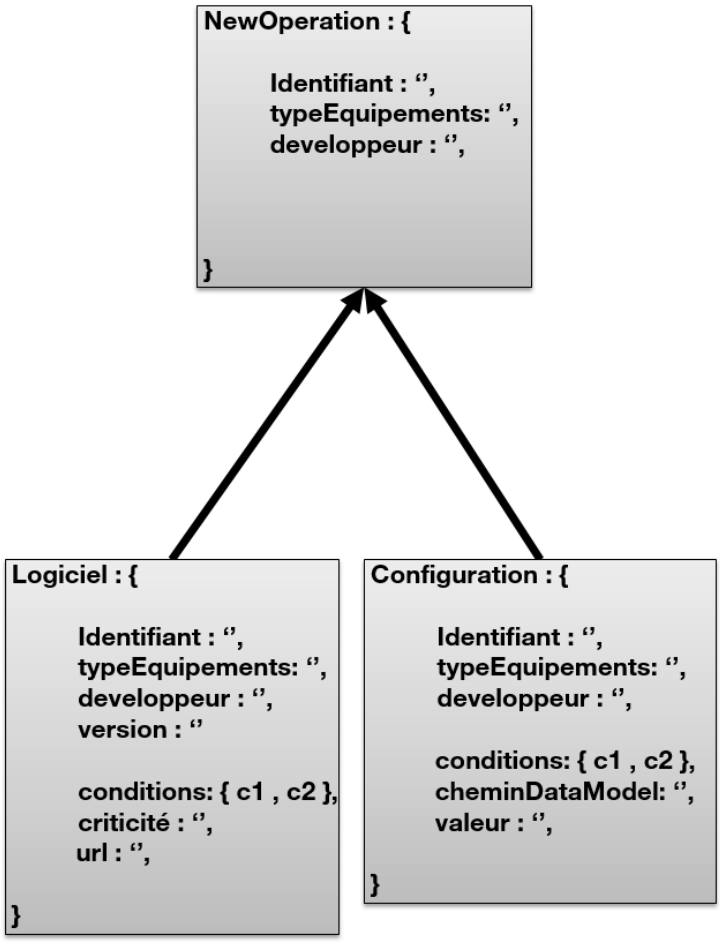

FIGURE 3.9 – Structure d'une notification

Le début de la structure dans la figure 3.9 est commun aux notifications de configuration et de logiciel interne et comprend les informations statiques concernant

cette mise à jour (p. ex. Version, développeur). Les deux structures ont des champs spécifiques en fonction de leur type. En effet, un logiciel est caractérisé par l'adresse où la plateforme peut récupérer son binaire [1], et une configuration est identifiée par le chemin permettant d'accéder à la partie du modèle de données qui sera lue, écrite, exécutée ou supprimée. Les deux types ont une structure nommée condition dans lesquels la plateforme peut traiter des spécificités concernant le modèle de données des équipements devant être ciblés par cette mise à jour. Par exemple tous les objets ayant une batterie supérieure à 40%. Nous avons recours à cette structure pour prendre en compte les spécificités des objets de l'IdO et de leurs environnements dynamiques.

Les concepts et définitions concernant l'administration de flottes d'équipements, maintenant détaillés, nous abordons dans la prochaine sous-section les cas d'usages puis la problématique avant de détailler l'architecture générale de notre approche.

3.2 Cas d'usage adressés

Une flotte d'équipements requiert des mises-à-jour logicielles qui concernent soit leur configuration, soit leur logiciel interne, ainsi que des commandes (p. ex. Redémarrage) et ce tout au long du cycle de vie des équipements qui la composent. Les mises-à-jour servent à augmenter la durée de vie des équipements en améliorant leurs performances, ajoutant et configurant des services ou en réglant des bugs. Les commandes permettent aux administrateurs systèmes de faire de l'assistance aux clients à distance pour la résolution de problèmes sur les équipements. Étant donné le contexte de cette thèse (industrielle chez Orange), l'expertise métier que nous avons récoltée de l'équipe d'administration de flotte est liée aux activités de production concernant les équipements domestiques de télécommunications actuellement en place. Il s'agit de décodeurs TV, de passerelles résidentielles et d'étendeurs de portée de réseaux sans-fils (répéteurs Wi-Fi). Nous détaillons dans ce qui suit, concernant les équipements domestiques de télécommunications et les flottes d'objets de l'IdO, les cas d'usages que nous souhaitons couvrir avec notre proposition :

— Les changements des configurations des équipements d'une flotte.
— Les variations de la composition de la flotte en termes d'équipements.
— Les campagnes de mise-à-jour de configuration et de logiciel.

Nous ne traitons pas dans cette thèse les commandes de résolution de problèmes à distance sur les équipements. Nous détaillons dans les prochaines sous-sections les cas d'usages visés par cette thèse et ce en commençant par les changements de configurations des équipements.

3.2.1 Changements de configurations des équipements d'une flotte

L'activation de service par les utilisateurs ou les propriétaires des équipements enclenche des changements de configuration sur certains équipements d'un parc.

1. Un logiciel interne est fréquemment sous la forme d'un fichier binaire

Un exemple d'illustration chez Orange est le cas d'un client qui active un bouquet de télévision optionnel. Cette activation déclenche dans le système d'informations d'Orange un ordre de changement de configuration qui est reçu par les plateformes d'administration système. Celui-ci est poussé vers le bon équipement (celui du client concerné) et dans un format intelligible par cet équipement. Cette traduction du format besoin métier (Activer Bouquet TV) vers le format compréhensible par les plateformes d'administration relève de la responsabilité du système d'information de l'opérateur. La procédure est la même pour la désactivation de service.

Les changements de configurations peuvent aussi être lancés manuellement par les administrateurs systèmes dans le but d'optimiser ces configurations par rapport à l'environnement de certains équipements. Dans le cas des parcs d'équipements domestiques de télécommunications, il s'agit d'adapter la configuration réseau (filaire et sans-fil) aux périphériques connectés. Concrètement, un exemple métier existant est de faire communiquer les équipements ayant besoin de beaucoup de bande passante (Décodeurs TV, Disque dur réseau) sur les réseaux sans-fils rapides et ceux qui en ont moins besoin (tablettes, téléphones) sur les réseaux sans-fils plus lents.

Dans le cadre de l'IdO, des changements de configurations du même type peuvent aussi être déclenchés par les clients et utilisateurs des objets tout comme les administrateurs systèmes. La multiplicité des services et de leur interdépendance fait que les solutions ad-hoc existantes seraient dépassées par la charge induite par cette multiplicité. Des critères additionnels sont aussi présents dans les parcs de l'IdO et requièrent que l'exécution de ces activations de services se fassent dans des conditions ou dans un ordre particulier. Il peut s'agir de besoin en énergie ou en capacité de calcul que des équipements dynamiques comme ceux de 'lIdO ne peuvent fournir de manière permanente contrairement aux parcs domestiques. Un besoin d'automatisation et d'adaptation continue à la charge et aux spécificités d'un parc de l'IdO est donc identifié pour ce cas d'usage.

Afin de gérer l'arrivée et le départ d'équipements dans une flotte, les plateformes d'adaptation doivent effectuer des opérations sur ces équipements sortants ou entrant afin de garder un parc bien configuré. Nous détaillons ce cas d'usage dans la sous-section suivante.

3.2.2 Variation de la composition de la flotte

La composition de la flotte change en fonction du retrait d'équipements, ou l'ajout de nouveaux équipements. En effet, ces variations sont dues, dans le cas de flottes d'équipements domestiques de télécommunications, à l'arrivée ou du départ de nouveaux clients ainsi qu'aux pannes non réparables sur des équipements. L'arrivée de nouveaux clients implique que des équipements sortis du carton, et donc ayant une configuration d'usine par défaut, s'ajoutent aux autres éléments existants d'une flotte. Ceux-ci requièrent que les plateformes d'administration leur fournissent les configurations et logiciels internes adéquats pour fonctionner. En cas de départ des équipements, il faut les enlever des bases de données du système et les traiter comme de nouveaux équipements lors de leur potentiel retour dans la flotte, car ils

sont affectés à un nouveau client. Si un équipement est absent trop longtemps du système d'administration, il continue à faire partie de la flotte tant qu'il n'a pas été déclaré comme dé-commissionné par le propriétaire de cet objet. Pour des raisons de sécurité, le délai d'absence maximal est plafonné et dépend de la politique de sécurité des entreprises administrant ces flottes.

Dans le cas de flottes d'objets de l'IdO, tout comme pour les équipements domestiques, de nouveaux objets intégrant la flotte requièrent configuration initiale et installation éventuelle d'un logiciel interne. La différence réside dans le fait que les dépendances sont plus complexes (contexte de l'IdO impliquant de multiples acteurs interagissant ensemble pour offrir des services). Des conditions spécifiques aux états des objets de l'IdO peuvent aussi être présentes (tout comme pour le cas des configurations de l'IdO cité plus haut). Ce fait est aussi vérifié durant le cycle de vie de ces objets. De plus, les modifications sont plus fréquentes dûes à l'aspect dynamique plus présent dans l'IdO que dans des parcs d'équipements domestiques de télécommunications. Ces particularités font que la gestion manuelle est inadéquate et confirme le besoin d'automatisation du pilotage et d'adaptation continue des plateformes d'administration système.

3.2.3 Campagnes de mise à jour

Mis à part les variations de la composition de la flotte d'objets, il y a un besoin continu de maintenance et d'amélioration de la qualité de service offerte par les équipements de la flotte. Pour ce faire, une opération de mise à jour (du logiciel ou de la configuration) des équipements est effectuée sur tout le parc. C'est ce qu'on nomme *"Campagne de mise à jour"* à Orange. Dans cette sous-section nous définissons ce qu'est une campagne de mise à jour ainsi que le moyen d'en déterminer la criticité.

3.2.3.1 Définition

Un autre cas d'usage présent dans l'administration de flottes d'équipements est ce qu'on nomme à Orange *campagnes de mise à jour*. Une campagne de mise à jour concerne une partie ou l'intégralité d'une flotte (soit des millions d'équipements) et est un cas d'usage fréquent des plateformes d'administration système chez Orange. Aujourd'hui, le choix est fait par Orange de simplifier l'administration qui est pilotée manuellement par des administrateurs systèmes. Cela se fait en opérant une seule campagne de mise à jour à la fois. Orange procède aujourd'hui de la manière suivante :

— La mise à jour est effectuée sur une faible population (p. ex. une dizaine d'équipements). Ces équipements seront surveillés par des équipes d'administration pendant une période fixe déterminée par ces équipes (deux semaines) par les administrateurs. En cas d'anomalie grave, la campagne est arrêtée.
— La mise à jour est poussée cette fois à un échantillon conséquent (quelques milliers). Cette fois, il y a comparaison des données de surveillance venant

de cet échantillon et un autre de taille équivalente qui n'a pas été mis à jour. Si de nombreux appels au service client ou une différence trop grande entre les données de sondes des deux échantillons est détectée, la campagne sera annulée.

— La mise à jour est installée par la suite sur tous les équipements restants.

L'approche existante s'appuie aujourd'hui sur des moyens humains et une surveillance manuelle à Orange. Cette surveillance manuelle n'est pas envisageable dans un parc d'objets de l'IdO vu la complexité, la taille et la dynamicité du parc. En fonction de la nature mise à jour, qu'elle soit un patch de sécurité, une mise à jour d'un service, ou un moyen de corriger un bug n'arrivant que rarement, il convient d'adapter la fréquence de surveillance en fonction de la criticité de la mise à jour. Afin de juger de cette criticité, nous avons identifié les niveaux que nous détaillons dans ce qui suit.

3.2.3.2 Criticité

Nous avons identifié quatre niveaux de criticité des mises-à-jour, les voici par ordre de la plus critique à la moins critique :

— Critique
— Majeure
— Mineure
— Corrective

Les mises à jour critiques concernent majoritairement des failles sécurité ou des modifications empêchant les dommages permanents sur un équipement. Les mises-à-jour majeures sont celles qui apportent de nouvelles fonctionnalités ou services. Les mises-à-jour mineures sont celles qui modifient les services existants ou enlèvent ceux qui n'ont plus lieu d'être ou qui ne sont plus maintenus. Les mises-à-jour correctives sont en charge de régler des bugs pouvant arriver dans certaines configurations et conditions d'utilisation. Nous ne nous proposons pas de classement de criticité des configurations car celles-ci sont fortement liées à un service et dépendent donc de la criticité du service associé.

Les cas d'usages visés par ce travail de thèse étant détaillés, nous en faisons une analyse visant à en extraire les principales limitations et besoins dans la sous-section suivante.

3.2.4 Synthèse

L'activation de service et les modifications de configurations aujourd'hui sont effectuées comme la gestion de variation de la composition de la flotte, c'est-à-dire au moyen de solutions ad-hoc. Les défis de l'internet des objets en termes d'interopérabilité, de dynamicité et scalabilité se traduisent en un nombre très élevé de services qui nécessitent activation et désactivation en fonction des besoins utilisateurs et des états des équipements. L'approche proposée se doit d'être automatisée et s'adapter de manière continue à ces besoins.

Concernant les campagnes de mise à jour, l'analyse des données de sondes (dites de log) n'est pas possible si plusieurs campagnes sont effectuées en parallèle (limites de la capacité humaine à en effectuer le suivi). De plus, la surveillance active pendant deux semaines d'un parc n'est pas viable (et probablement contournée) dans un cas de faille de sécurité critique. Dans le cadre de l'IdO avec une multiplicité du nombre et des types d'équipements, des services, et une dynamicité conséquente de leurs états et environnements, un besoin d'automatisation du pilotage et une adaptation continue se justifient. De plus, pouvoir faire une campagne seulement à la fois n'est pas viable pour l'IdO dû à la quantité d'objets à gérer et la volumétrie.

Somme toute, l'approche existante concernant les cas d'usages détaillés dans la section 3.2 comporte des limitations sévères la rendant non viable pour des parcs d'objets de l'IdO. Cela requiert régulièrement des mises-à-jour en termes de développement logiciel pour adapter la solution d'administration à distance à de nouveaux services et équipements et leur spécificités. Vu la multiplicité des objets de l'IdO et celle des services fonctionnant autour, ainsi que la présence de critères spécifiques aux types d'objets, il y a besoin d'une approche capable de s'adapter aux types d'objets administrés sans pour autant développer une nouvelle fonctionnalité ad-hoc dans la plateforme d'administration.

L'objectif de cette thèse est de proposer une architecture autonomique permettant d'administrer à distance tout type d'équipements (équipements domestique de télécommunications et objets de l'IdO), et ce en automatisant les traitements des trois cas d'usages ciblés mentionnés dans cette section en prenant en compte les éventuelles spécificités. Ces cas d'usages sont : les changements des configurations des équipements d'une flotte, de variation de la composition d'une flotte, et d'opération de campagnes de mise à jour.

3.3 Problématique

Les besoins en administration système ciblés dans la section 3.2 se modélisent par une nécessité fréquente de lancer des opérations d'administration, d'en identifier la bonne cible selon les spécificités des équipements et de la mise à jour. Ces besoins surviennent à chaque fois qu'un équipement rejoint ou quitte la flotte, et lorsqu'un nouveau logiciel interne ou configuration est disponible. Afin d'exécuter ces opérations de manière progressive et par suite d'éviter les erreurs éventuelles et laisser la possibilité au système d'y réagir, elles seront décomposées en sous-opérations de taille variable. La taille de ces opérations (en termes du nombre d'équipements visés) est calculée par le système en fonction des erreurs rencontrées lors de l'exécution. Les sous-opérations n'étant pas compréhensibles par les plateformes d'administration, chaque sous-opération sera traduite en un ensemble de commandes spécifiques à la plateforme d'administration et aux équipements administrés. Ces commandes sont exécutées sur les équipements du parc.

Une fois ces opérations lancées, il est obligatoire de surveiller si elles ont un effet négatif, en termes de perte de performances ou dommages sur les équipements. Il

est aussi nécessaire de surveiller l'infrastructure pour éviter la saturation si trop de commandes sont envoyées aux équipements dans un court laps de temps via les plateformes d'administration de flotte.

La problématique est donc composée des éléments suivants :

1. Générer les opérations nécessaires pour maintenir un parc à jour, visant les équipements en fonction de l'arrivée et du départ des équipements du parc ainsi que la disponibilité de nouveaux logiciels internes ou configurations.
2. Traduire les sous-opérations en liste de commandes à exécuter et suivre leurs taux de réussites et d'erreurs.
3. Surveiller les ressources des plateformes d'administration et le parc d'équipements durant les opérations d'administration pour réduire les fautes d'exécutions (non critiques : temps de réponses élevé des plateformes, et critiques : causant des dysfonctionnements sur les équipements).
4. Déployer les composants d'administration système sur l'infrastructure en évitant les éventuelles surcharges sur le processeur, la mémoire ou le réseau.

Ces problématiques à résoudre sont divisibles en deux catégories :

1. Objectifs fonctionnels (c.a.d. concernant le métier) d'automatisation de l'administration de parcs.
2. Objectifs de gestion de ressources des plateformes d'administration de parcs d'équipements.

Chaque catégorie est traitée dans un chapitre dans la suite de ce manuscrit. La section suivante détaille notre approche pour aborder cette problématique.

3.4 Architecture Générale

Dans cette section, l'architecture globale du gestionnaire autonomique est détaillée. L'approche est définie à un niveau macroscopique avant de détailler chacune des boucles de ce gestionnaire autonomique. Les boucles qui traitent de l'automatisation de l'administration du parc sont traitées en premier avant de passer à celles qui gèrent la gestion de ressources.

Notre approche est basée sur quatre gestionnaires autonomiques. Chaque sous-problématique citée dans la section précédente est adressée par une boucle. Ces gestionnaires collaborent pour un objectif commun. Assurer que les besoins d'administration d'un parc soient réalisés avec les ressources disponibles de manière continue et en fonction des spécificités des équipements du parc. Les boucles sont coordonnées de manière hiérarchiques [74].

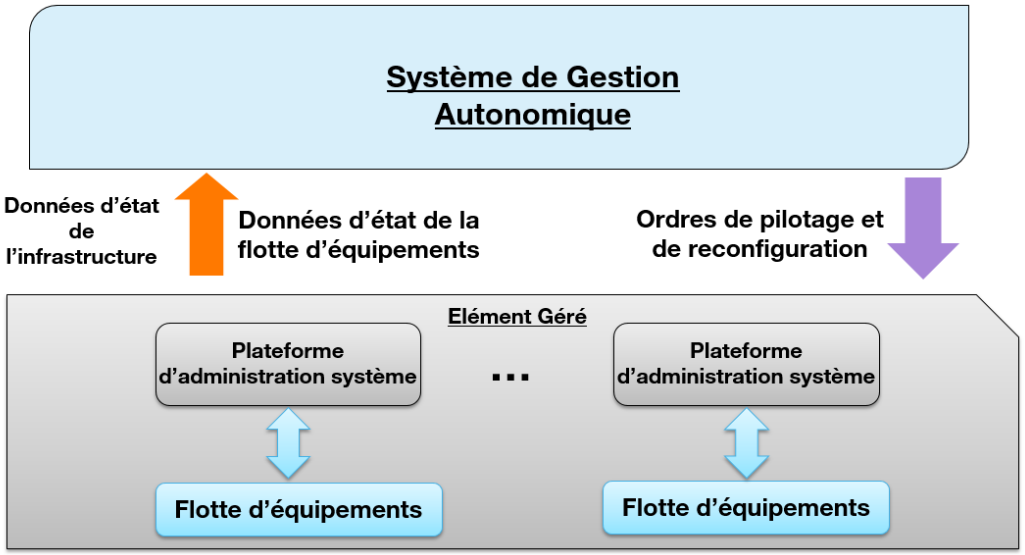

FIGURE 3.10 – Architecture globale

Afin d'adresser la problématique détaillée dans la section précédente, nous nous appuyons sur un système de gestion autonomique qui pilote les plateformes d'administration de flottes existantes. Le but est d'enrichir les capacités des dites plateformes en fonction de leur niveau d'administration. Ces plateformes servent de médiateurs entre les administrateurs et les équipements.

La figure 3.10 présente l'architecture globale de notre proposition à un niveau de détail macroscopique. Le système de gestion pousse des ordres de pilotage des plateformes et de re-configuration du déploiement des composants logiciels des dites plateformes, et ce, à partir des données d'observation récoltées : les données d'état de l'infrastructure et des flottes d'équipements.

Dans la suite de ce chapitre, nous détaillerons plus amplement cette architecture.

3.4.1 Boucles d'automatisation de l'administration d'un parc

La figure 3.11 détaille la partie qui traite de la problématique qui concerne l'automatisation du maintien à jour d'un parc d'équipements. Cela est réalisé en lançant des opérations d'administration quand une partie ou la totalité du parc n'est plus à jour. Les sous-problématiques abordées ici sont donc :

— Générer des opérations assurant ce maintien à jour de manière continue en fonction de l'arrivée et du départ des équipements du parc ainsi que la disponibilité de nouveaux logiciels internes ou configurations.

— Suivre l'exécution de ces opérations de mise à jour en traduisant les sous-opérations en liste de commandes à exécuter et à suivre (en extraire les taux de réussites et d'erreurs).

Chaque sous-problématique est gérée par une boucle d'automatisation.

La génération d'opération d'administration et leur décomposition en sous-opération

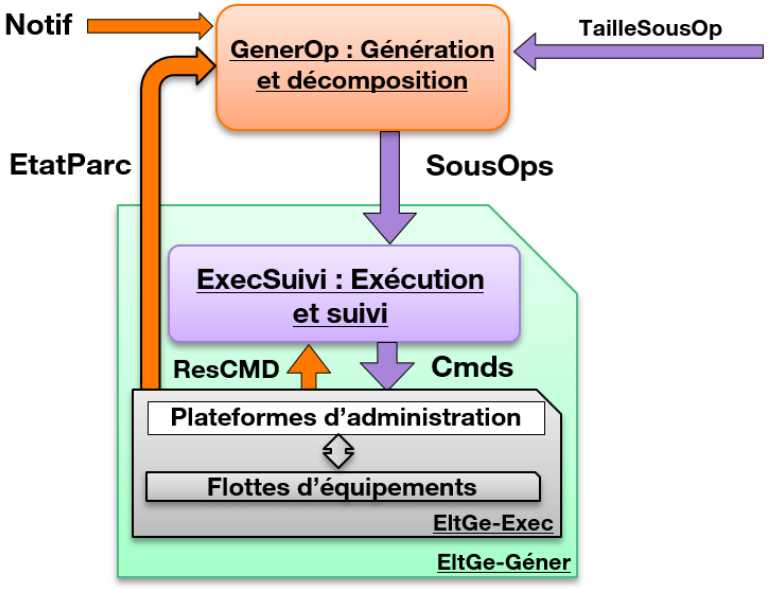

FIGURE 3.11 – Schéma simplifié des gestionnaires d'automatisation de l'administration d'un parc

est réalisée par le gestionnaire en orange dans la figure 3.11 : (GenerOp) : *Génération et Décomposition*. Il prend en entrée les données d'état du parc (EtatParc dans la figure) qu'il intègre dans sa base de connaissance. Ces données permettent de décider si le parc a accueilli ou perdu un équipement. Cette base de connaissance est mise à jour lors de la présence d'une nouvelle configuration ou un nouveau logiciel interne détectée par le gestionnaire sous la forme d'une *notification* reçue : N dans la figure. Si besoin d'administration il y a, le gestionnaire (GenerOp) identifie la cible et décompose l'opération à faire en sous-opérations visant une partie de la cible pour une exécution progressive. Ces sous-opérations sont envoyées au gestionnaire d'exécution et suivi (ExecSuivi). Cela permet de s'adapter à des possibles erreurs d'exécution et des pannes sur les équipements. La taille des sous-opérations est une entrée de cette boucle qui est définie par une autre boucle tierce détaillée dans la sous-section suivante.

L'exécution et le suivi des sous-opérations est assuré par une boucle à part (ExecSuivi), qui prend en entrée les sous-opérations venant de la boucle de génération et décomposition (GenerOp). Ces sous-opérations sont traduites en liste de commandes à exécuter sur la cible du parc visée par cette sous-opération. Une fois ces commandes exécutées, la boucle collecte les résultats avant d'en extraire des statistiques qui seront envoyés à une boucle responsable de la régulation de la vitesse de décomposition (RegulVitesse) que nous détaillons dans la sous-section suivante.

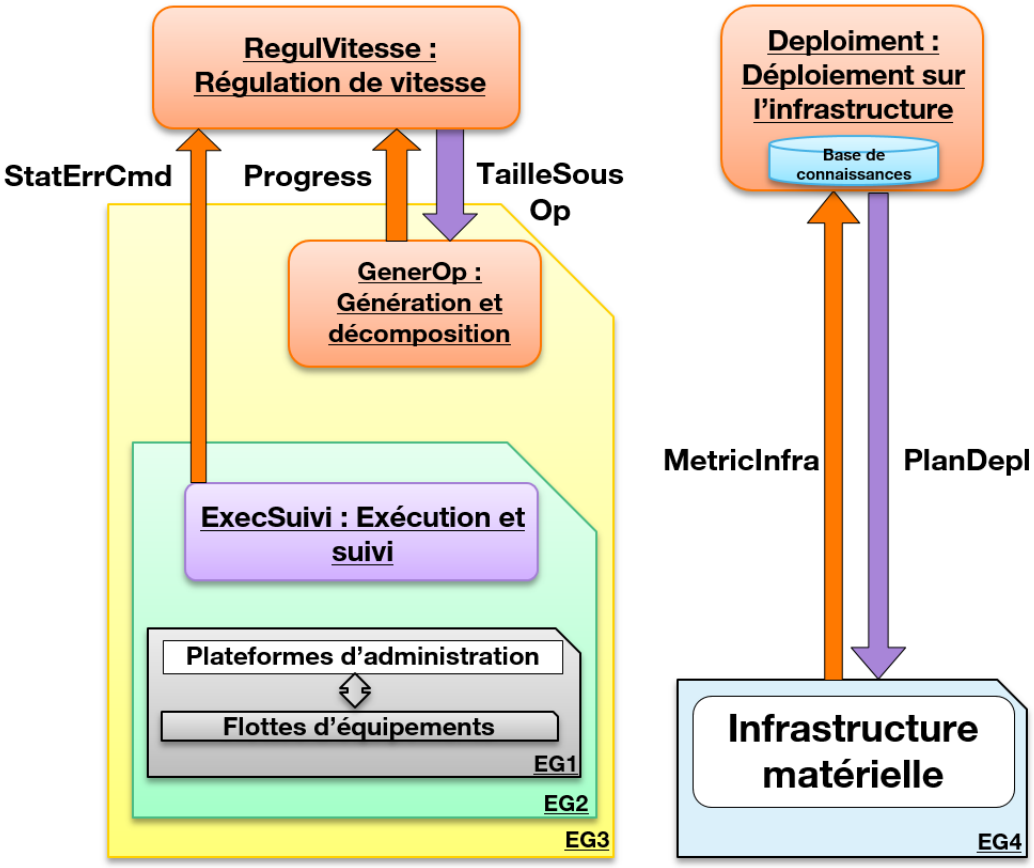

FIGURE 3.12 – Boucles d'automatisation de la gestion des ressources

3.4.2 Boucles d'automatisation de la gestion des ressources

Comme pour la gestion de l'automatisation de l'administration système d'un parc, chaque problématique liée à la gestion des ressources est gérée par une boucle.

Les constructeurs et administrateurs sont responsables du développement de configuration et logiciels internes et sont sensés en assurer le bon fonctionnement avant déploiement. La réalité du métier aujourd'hui à Orange concernant les équipements domestiques de télécommunication montre que le risque de configuration ou logiciel entraînant des erreurs est conséquent. La détection et la réaction aux erreurs et dommages dus à une configuration ou logiciel interne défectueux, est réalisée par ce gestionnaire.

Les plateformes s'exécutent sur des infrastructures qui hébergent d'autres applications, et ont pour vocation d'être potentiellement déployées dans n'importe lequel des nœuds de l'infrastructure d'un opérateur. Ce gestionnaire est responsable de la régulation de la vitesse d'exécution en fonction des ressources qui sont disponibles pour les plateformes.

C'est dans cette optique que nous avons conçu ce gestionnaire autonomique (RegulVitesse) : *Régulation de vitesse*. Celui-ci s'exécute lorsque des sous-opérations sont en cours d'exécution par la boucle d'exécution et suivi (ExecSuivi). En effet, son entrée provient de ce gestionnaire qui lui fournit des statistiques d'erreurs (StatErrCmd) de chaque type (critique et non critique) et les temps de réponse des plateformes d'administration. En fonction de la variation de ces statistiques et de la progression des opérations (Progress), une décision est prise et envoyée (TailleSousOp) concernant l'augmentation, la diminution ou la non-variation de la taille des sous-opérations générées (en termes de nombre d'équipements ciblés) à chaque pas, par le gestionnaire de génération et décomposition (GenerOp).

La surveillance de l'infrastructure est gérée par un gestionnaire autonomique dédié (Deploy). Il prend en entrée des métriques d'infrastructure (MetricInfra) (c.a.d. Charge processeur, mémoire et réseau) et en cas de saturation, génère une carte de déploiement (PlanDepl), orchestrant le redéploiement de certains composants de notre système autonomique : plateformes d'administration et gestionnaire d'exécution et de suivi (ExecSuivi).

En effet, celui-ci a été pensé pour être multi-instanciable contrairement à celui de génération et de décomposition (GenerOp) qui centralise la vision de l'état du parc en ayant un rôle de supervision. Ce dernier n'est pas soumis à la charge de l'exécution de commandes d'administration. L'idée est d'avoir un gestionnaire haut hiérarchiquement pour la prise de décision globale (GenerOp) et une multitude de gestionnaires (ExecSuivi) assurant l'exécution et le suivi au niveau local. L'expérience métier Orange montre que les modules d'exécution tendent à saturer beaucoup plus vite que ceux en charge du lancement d'opérations.

Les quatre gestionnaires autonomiques traitant des deux objectifs d'automatisation de l'administration du parc (ExecSuivi) et (GenerOp), et ceux de la gestion de ressources (RegulVitesse) et (Deploy), étant détaillés, nous présentons maintenant l'architecture globale détaillée de notre proposition.

3.4.3 Système d'administration autonomique d'administration de flottes d'équipements

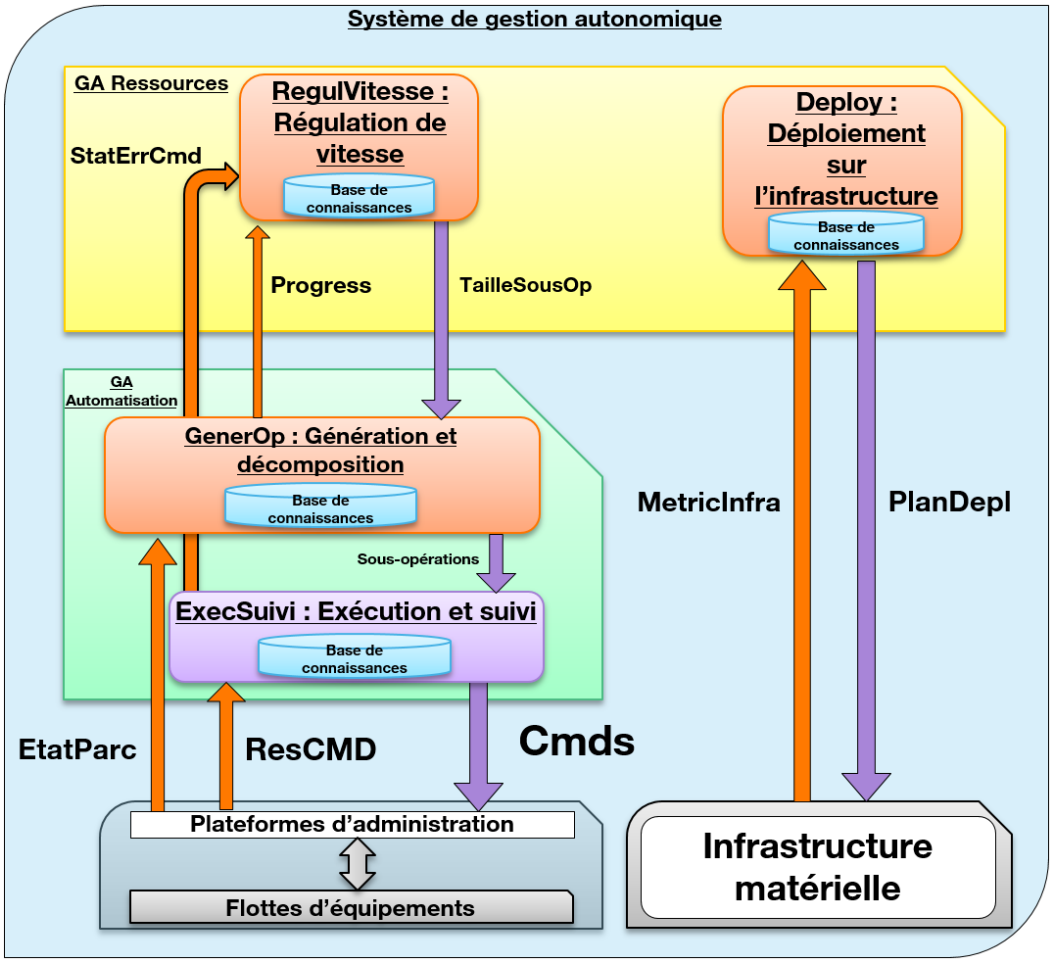

FIGURE 3.13 – Architecture globale du système autonomique

La figure 3.13 reprend les schémas des figures 3.11 et 3.12 en détaillant les interactions entre les différents gestionnaires et leurs éléments gérés.

Les gestionnaires sont organisés dans une optique de coordination hiérarchique [74]. Le gestionnaire GenerOp a pour éléments gérés d'un côté le gestionnaire ExecSuivi et de l'autre les éléments gérés d'ExecSuivi. Le principe est le même pour le gestionnaire RegulVitesse qui gère l'ensemble de GenerOp et ses propres éléments gérés.

3.4.4 Structure de présentation d'un gestionnaire autonomique

Pour chaque gestionnaire, dans ce qui suit, nous allons détailler les éléments suivants :

— Objectif : état de l'élément géré poursuivi par le gestionnaire

— Pas : fréquence à laquelle le gestionnaire autonomique passe par ses différentes phases Monitor, Analyze, Plan, Execute.
— Données d'observation : donnée en entrée provenant de l'élément géré représentant son état.
— Actionneurs : leviers d'ajustement permettant de modifier l'état de l'élément géré.
— Élément géré : système administré par le gestionnaire.
— Décision : processus durant lequel le gestionnaire identifie les besoins de reconfiguration de l'élément géré et la manière de faire ces changements.
— Base de connaissance : comporte la connaissance sur l'élément géré.
— Interactions : éventuels échanges avec d'autres gestionnaires autonomiques ou systèmes d'informations.

Dans la suite de ce manuscrit, nous traitons chaque type de gestionnaire dans un chapitre séparé. Ceux qui automatisent l'administration du parc sont détaillés dans le chapitre suivant et ceux qui automatisent la gestion de ressources sont détaillées dans celui d'après. Nous détaillons, chaque gestionnaire autonomique, selon la structure détaillée dans la section 3.4.4.

Chapitre 4

Automatisation de l'administration d'un parc

Sommaire

Nous abordons dans ce chapitre les gestionnaires autonomiques en charge de l'automatisation du pilotage des plateformes d'administration système. Ces gestionnaires sont au nombre de deux et relèvent de deux niveaux. Un gestionnaire dit 'haut niveau' (GenerOp) pensé pour une prise de décision centralisée ayant une vue globale sur la flotte, qui n'est instancié qu'une fois. L'autre gestionnaire (ExecSuivi) dit de 'bas niveau' est multi-instanciable. Ce choix est motivé par l'expertise métier en administration de flottes domestiques d'équipements de télécommunication

d'Orange qui tend à démontrer une saturation sur l'exécution avant saturation en termes de prise de décision.

Dans les sous-sections suivantes, nous commencerons par détailler le gestionnaire de niveau inférieur (ExecSuivi) avant de passer au supérieur (GenerOp) vu que celui de haut niveau incorpore le gestionnaire de bas niveau dans son élément géré. La multi-instanciation est régulée par un gestionnaire tierce, qui fait l'objet de la deuxième section du chapitre 6 car elle relève de la gestion des ressources.

Dans ce chapitre, les gestionnaires autonomiques sont exposés au format MAPE-K. Le nom des modules correspondant à une étape est nommé de la première lettre de l'étape (M, A, P, E) suivi d'un chiffre s'il y a plusieurs sous étapes. Par exemple la première étape du (Monitoring) sera nommé M1 dans les schémas.

4.1 Exécution & Suivi des commandes

Nous détaillons dans cette section les objectifs, le pas, les données d'observation, les actionneurs, les éléments gérés, le processus de prise de décision, la composition de la base de connaissance et les interactions de ce gestionnaire autonomique responsable de l'exécution et du suivi des commandes d'administration système.

Ce gestionnaire autonomique permet au système d'un côté de traduire les sous-opérations reçues du gestionnaire (GenerOp) de *génération et décomposition* en commandes et de l'autre d'agréger les données d'exécution des commandes d'administration afin d'en extraire des statistiques de réussite, d'erreurs critiques et non critiques (aussi nommées avertissements). Ces données sont envoyées au gestionnaire (RegulVitesse) de régulation de vitesse qui prendra la décision d'accélérer ou ralentir, voire abandonner les opérations qui ont un taux d'erreur trop élevé. Étant donné que les sous-opérations n'ont pas la même vitesse d'exécution et que certaines peuvent être abandonnées en cours d'exécution en fonction du nombre de ré-essai des commandes qui les composent, il y a ordonnancement (en plus de celui réalisé par le gestionnaire (GenerOp)) à chaque pas de ce gestionnaire. L'ordonnancement au niveau exécution (ExecSuivi) se focalise sur la politique de ré-essai des commandes en excluant les commandes n'aboutissant pas et privilégiant les opérations qui viennent d'arriver afin d'éviter un phénomène de famine.

4.1.1 Objectif

L'objectif est d'assurer le suivi de l'exécution des sous-opérations à réaliser sur le parc. Pour chaque sous-opération, il y a réalisation de sa traduction en commandes, l'exécution et l'envoi des statistiques d'erreur d'exécution au gestionnaire (RegulVitesse).

4.1.2 Pas

Le lancement initial de l'exécution de ce gestionnaire suit les envois de sous-opérations à traiter de la part du gestionnaire de génération et décomposition (Exec-

Suivi). Une fois lancée, l'exécution est périodique à fréquence fixe déterminée par les administrateurs. Ce gestionnaire ne s'arrête que lorsqu'il n'y a plus de sous-opérations en cours.

4.1.3 Données d'observation

Afin d'atteindre ses objectifs ce gestionnaire autonomique prend deux entrées.

— Un ensemble de sous-opérations à traiter, qui seront ordonnancées, traduites en commande et envoyées pour exécution.

— Les instances de modèles de données des équipements du parc administré. Ces dernières servent à détecter les erreurs d'exécution des commandes de configuration ou de mise à jour. Au travers de ces instances de modèles de données, les administrateurs peuvent mettre en place des alertes d'anomalies dans le comportement des objets. À titre d'exemples, nous pouvons citer le nombre de paquets envoyés sur le réseau, la charge CPU moyenne de l'équipement après sa mise à jour. Pour les équipements domestiques de télécommunication, l'indicateur de nombre d'appels au service après-vente est pris en compte par les administrateurs pilotant les plateformes d'administration, si le système d'informations permet d'avoir ces données, il est possible de les intégrer comme données d'observation.

4.1.4 Actionneurs

Ce gestionnaire génère deux sorties, des commandes à exécuter sur les équipements et les statistiques d'erreurs de ces commandes qui sont envoyées au gestionnaire de régulation de vitesse (RegulVitesse). En effet, à partir des données d'exécution (logs, taux de réussite), des taux d'erreurs et de réussite des opérations sont compilés et envoyés vers (RegulVitesse).

```
{
  "statistiques": {
  "idStat": "00256120",
  "idOperationMere": "512400_ORANGE",
  "idSousOperationMere": "512400_ORANGE-P1",
  "stats": {
      criticalErrrors": "1",
      "nonCriticalErrrors": "5",
      "TotalCommands": "25"
        }

    }
  }

}
```

Structure 4.1.4 : Données d'érreurs d'execution des commandes

La structure 4.1.4 détaille le format des statistiques envoyées par le gestionnaire (ExecSuivi) au gestionnaire (RegulVitesse). Ces statistiques sont caractérisées par leur identifiant propre, l'identifiant de la sous-opération qu'elles concernent et l'opération mère de cette dernière. Les statistiques sont sous le format : type d'erreur, nombre d'erreurs, nombre total de commandes.

Afin de permettre une régulation de vitesse de décomposition d'opérations en sous-opérations, le gestionnaire (ExecSuivi) transmet la progression des opérations au gestionnaire (RegulVitesse). L'objectif est de faire en sorte que la variation de la *taille* de sous-opération générée, en termes de nombre d'équipements traités, par *pas* de gestionnaire soit aussi calculée en fonction de l'avancement d'une opération. En effet, une opération en *phase* (Section 5.1.6) de 'Généralisation' qui rencontre des erreurs aura une baisse de *taille* moins forte que lors d'une *phase* de 'Test' qui aura un comportement plus prudent vis-à-vis de ces anomalies.

4.1.5 Élément Géré

Ce gestionnaire agit sur les équipements de la flotte au travers des plateformes d'administration qui en ont la charge.

4.1.6 Décision

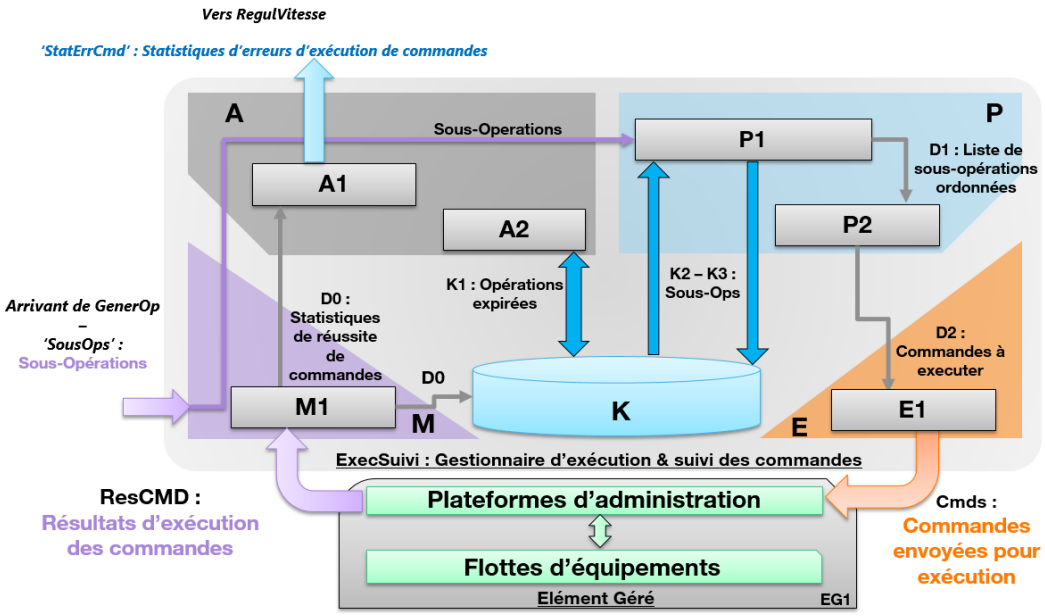

FIGURE 4.1 – Architecture MAPE-K du gestionnaire d'exécution & suivi des commandes

La figure 4.1 détaille les interactions avec les autres gestionnaires et son fonctionnement en s'appuyant sur l'architecture MAPE-K. Les variables utilisées sont D0,D1,D2 et K1,K2,K3, respectivement en référence aux données d'exécution et

données de connaissance du gestionnaire. Il possède deux entrées : 'SousOps' venant du gestionnaire de génération d'opération (GenerOp), et 'ResCmd' arrivant des plateformes d'administration. Il produit deux sorties, 'Cmds' destinée aux plateformes d'administration et 'StatErrCmd' envoyée au gestionnaire de régulation de vitesse (RegulVitesse). En voici les étapes :

— Les sous-opérations (SousOps) arrivant du gestionnaire (GenerOp) de *génération et décomposition des opérations* sont directement envoyées à un module d'ordonnancement (P1) afin d'écarter les moins urgentes de la file d'exécution. Les sous-opérations écartées sont stockées dans la base de connaissance (K2) avant d'être réinjectées (K3) dans l'étape d'ordonnancement (P1) lors du prochain tour de gestionnaire pour un lancement ultérieur durant un autre cycle de boucle.

— La liste des sous-opérations ordonnées (D1) est ensuite traduite (P2) en un ensemble de commandes (D2) qui sont envoyées pour exécution (Cmds) sur les équipements concernés via les plateformes d'administration système.

— Durant et après l'exécution de ces commandes, le gestionnaire surveille les équipements au travers de leurs modèles de données (E) pour récolter les données d'exécution des commandes d'administration. À part les erreurs survenant durant l'exécution, ce gestionnaire détecte d'éventuelles anomalies latentes qui seront incorporées aux statistiques (D0). Par exemple, une fuite mémoire qui mettrait du temps à déclencher des symptômes sur un routeur. À partir de ces données, des statistiques d'erreurs (critiques et non-critiques) (D0) sont construites (A1) et envoyées vers le gestionnaire de *régulation de vitesse d'exécution* (RegulVitesse) afin qu'en fonction du nombre d'erreurs, celui-ci baisse la vitesse de décomposition des opérations en sous-opérations dans le but que les erreurs ne se généralisent pas sur tout le parc. En cas de taux trop élevés, le gestionnaire de régulation de vitesse (RegulVitesse) annule complètement l'opération en tentant de récupérer l'état pré-mise-à-jour des éléments traités.

— À partir des statistiques de réussite des commandes (D0), ce gestionnaire applique une politique de ré-essai (A2) des commandes échouées. Les sous-opérations comportant des commandes réessayées un nombre de fois fixé par l'administrateur système, sans succès (K1) sont supprimées de la base de connaissance (K).

4.1.7 Base de connaissance

La base de connaissance de ce gestionnaire autonomique contient la liste des sous-opérations en cours ainsi que la liste des commandes associée à chacune d'entre elles. La figure 4.2 montre le schéma de données des entités composant cette base. Les sous-opérations sont identifiées par leur 'idSop', et des caractéristiques de leur opération mère : nature (Lecture, écriture, suppression, mise à jour), leur type (mise à jour de configuration ou de logiciel), de la valeur dans un cas d'écriture ou de mise à jour, leur type d'équipements ciblés, leur liste des équipements ciblés ainsi que

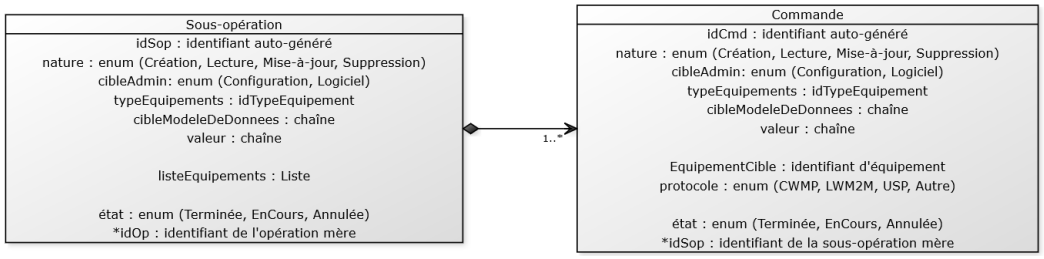

FIGURE 4.2 – Représentation des sous-opérations et commandes dans la base de connaissance

leur état (réussie, échouée, en cours).

Les commandes reprennent la même structure en ajoutant le nombre de tentatives et le protocole d'administration système (p. ex. LWM2M, USP).

La plateforme de recherche Thing'in [32] peut servir de source d'alimentation de cette base de connaissances. En effet, celle-ci contient les informations sur les équipements et par suite les interfaces d'administration systèmes disponibles, leurs protocoles et leurs modalités d'accès.

4.1.8 Interactions

Ce gestionnaire interagit avec les plateformes d'administration des équipements et avec deux autres gestionnaires : *Génération de décomposition d'opérations* (GenerOp) via l'entrée (SousOps) comportant la liste des sous-opérations à traiter et *régulation de vitesse d'exécution* (RegulVitesse) via la sortie (Cmds) comportant les statistiques d'erreurs.

Nous abordons dans ce qui suit le gestionnaire autonomique (GenerOp) dont l'élément géré est composé de la flotte d'équipements et leurs plateformes ainsi que du gestionnaire (ExecSuivi) d'exécution et suivi des commandes traité dans cette section.

4.2 Génération et décomposition d'opérations

Ce gestionnaire autonomique octroie au système d'administration des équipements de l'IdO des capacités d'auto-configuration. En effet, le système devient capable de se piloter lui-même afin de lancer les opérations d'administration nécessaires pour garder une flotte d'équipement 'à jour' selon les exigences métier actuelles. À titre d'exemple, on peut citer "une nouvelle version du logiciel interne doit être installée sur toute la flotte en moins de 48 heures". Au-delà du maintien à jour, cette boucle permet de lancer des changements de configuration et activations de services, en fonction des besoins actuels du parc. Par exemple dans le cas où un nouveau point de contrôle d'ampoules connectées serait détecté, il faut réaffecter les ampoules les plus proches à ce dernier. Un autre exemple est celui d'un répéteur Wi-Fi qui serait installé. Les équipements ayant une meilleure réception via ce dernier doivent être

reconfigurés pour l'utiliser comme passerelle afin de garantir une meilleure qualité de service, une moindre émission d'énergie, tout en réduisant le risque de pannes.

4.2.1 Objectif

Ce gestionnaire autonomique possède un objectif majeur : garder la flotte d'équipement à jour. Celui-ci se traduit en deux sous-objectifs :

— Garder un parc à jour en termes de logiciel interne ou de configurations selon la disponibilité de nouvelles mises à jour.
— Reconfigurer le parc de manière continue afin d'assurer que lors de l'arrivée ou du départ d'éléments de la flotte garde un état dit "à jour".

4.2.2 Pas

Ce gestionnaire a un pas périodique, fixé par les administrateurs systèmes, qui est enclenché quand il y a des *opérations* en cours d'exécution. Il peut aussi être activé de manière événementielle par deux déclencheurs : la réception d'une notification de nouvelle *notification* de disponibilité (configuration ou logiciel interne) ou la détection d'un changement de composition sur la flotte (c.a.d. Arrivée ou départ d'un équipement). Les figures 4.3 et 4.4 détaillent respectivement un exemple de notification possible concernant un nouveau logiciel interne et une nouvelle configuration.

```
NewFirmware : {

        identifiant : 'FirmwarePhillipsHue',
        typeEquipements: 'PhillipsHueBridge',
        developpeur : 'Phillips',
        version : '2.0'
        criticité : '1',
        url :
          'ftp://phillips.com/support/Bridge/FW/2.0/FW.bin',

}
```

FIGURE 4.3 – Exemple de structure d'une notification de disponibilité d'une mise à jour pour un pont de contrôle d'ampoules connectées

La figure 4.3 montre la structure d'une notification de nouveau logiciel disponible. Elle est composée d'un identifiant, du type d'équipement concerné, d'informations sur le développeur, la version du logiciel, et du niveau de criticité sur l'échelle que nous avons définie dans la section 3.2.3.2. L'URL représente l'adresse internet ou intranet permettant de récupérer le fichier binaire de la mise à jour.

La figure 4.4 montre la structure de la notification dans le cas d'une nouvelle configuration. Pour des raisons de complexité, les configurations ne sont pas classifiées par niveau de criticité dans notre approche, ce champ n'est pas présent. Il en est de même pour le champ URL car une configuration ne prend pas la forme d'un fichier binaire, il s'agit d'une lecture ou écriture sur le modèle de données d'un équipement. C'est pour cette raison qu'un champ 'cheminDataModel' indique la partie

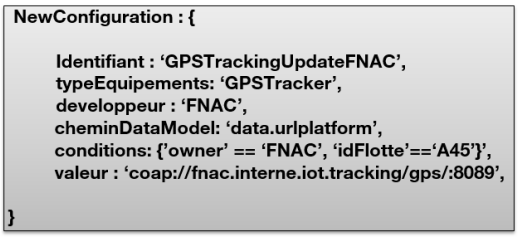

FIGURE 4.4 – Exemple de structure d'une notification de disponibilité d'une mise à jour de la configuration d'un traceur GPS

du modèle de données visée ainsi qu'un autre champ 'valeur' indique l'éventuelle valeur dans le cas d'une écriture. Un ensemble de conditions sont présentes afin de préciser si seuls certains équipements vérifiant ces conditions sur leurs modèles de données seront traités. Il peut s'agir par exemple d'une condition sur le propriétaire comme dans la figure 4.4 où la configuration ne vise que les objets appartenant à la FNAC, plus précisément la flotte A45.

4.2.3 Données d'observation

Pour ce faire, le gestionnaire prend en entrée les informations suivantes :

— Les états des équipements, en termes de configuration logicielle et matérielle. Ces états sont sous la forme d'instance de modèle de données. Ils alimentent sa base de connaissance détaillée dans la section 4.2.7.

— Des notifications de disponibilité de nouveaux logiciels internes ou configurations détaillées dans la section 3.1.8.

En outre, ce gestionnaire reçoit, de la part du gestionnaire de régulation de vitesse (RegulVitesse), des recommandations concernant la vitesse de décomposition des opérations en sous-opérations (taille de sous-opération générée par pas de gestionnaire).

4.2.4 Actionneurs

Ce gestionnaire génère des opérations d'administration système, puis les décompose en un ensemble de *sous-opérations* destinées au gestionnaire d'exécution et suivi (ExecSuivi) qui se charge de leur exécution sur les équipements de la flotte.

4.2.5 Élément géré

L'élément géré de ce gestionnaire est composé de deux entités :

— Les équipements de la flotte au travers de la plateforme d'administration qui les gère.

— Le gestionnaire d'exécution et de suivi (ExecSuivi).

La figure 4.5 montre l'élément géré de ce gestionnaire et ses interactions.

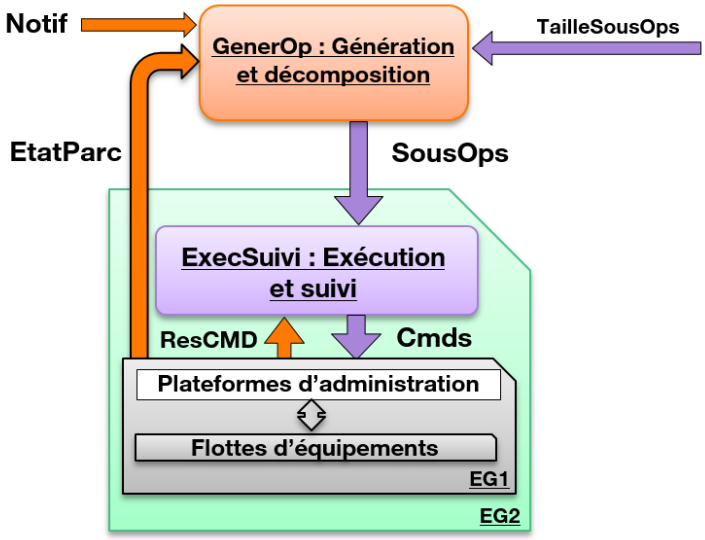

FIGURE 4.5 – Élément géré du gestionnaire de Génération et décomposition

4.2.6 Décision

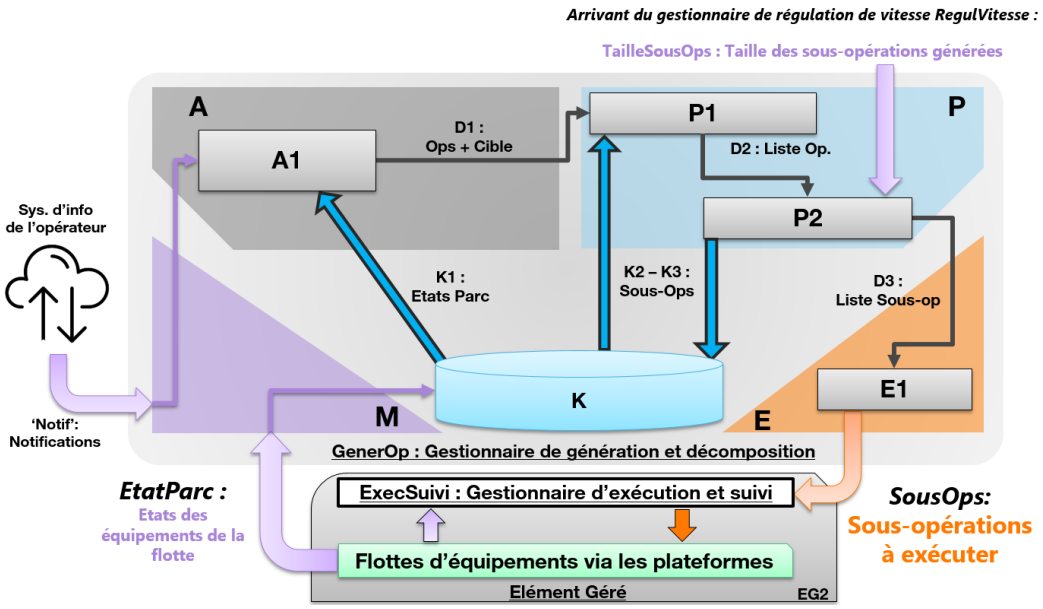

FIGURE 4.6 – Architecture MAPE-K du gestionnaire de génération et décomposition d'opérations (GenerOp)

La figure 4.6 détaille les interactions avec les autres gestionnaires et son fonctionnement en s'appuyant sur l'architecture MAPE-K. Les variables utilisées sont D1,D2,D3 et K1,K2,K3, respectivement en référence aux données d'exécution et

données de connaissance du gestionnaire. Il possède trois entrées : 'Notif' venant du système d'information de l'opérateur, 'EtatParc' arrivant des plateformes d'administration et 'TailleSousOp' provenant du gestionnaire de régulation de vitesse (RegulVitesse). Son unique sortie est 'SousOps' destinée au gestionnaire d'exécution et suivi (ExecSuivi). En voici les étapes :

- Si une *notification* de nouvelle configuration ou logiciel arrive, la boucle s'active. Ces notifications arrivent du système d'information de l'opérateur. Il s'agit de l'élément provoquant un déclenchement événementiel. En l'absence de notification le déclenchement est périodique. Chaque période de temps fixée par les administrateurs, le gestionnaire s'active pour vérifier s'il y a changement d'état dans le parc.
- Une étape d'analyse de l'état du parc est exécutée (A1). Dans le cas d'un déclenchement périodique, il est question d'identifier, soit un besoin d'opération de mise à jour sur un ou plusieurs équipements ayant un logiciel interne obsolète ou de retirer de l'inventaire un objet ne faisant plus partie de la flotte depuis un laps de temps fixé par les administrateurs. Le besoin de mise à jour peut soit provenir d'une notification de disponibilité de logiciel (ou configuration) pour plusieurs équipements du parc ou de la ré-apparition d'un équipement dans la flotte après une absence. La sortie de cette étape d'analyse est D1 : opération à réaliser sur quel type d'équipement. Dans le cas d'une arrivée d'un objet ayant un logiciel obsolète, l'opération d'administration ne vise que l'objet en question (ciblé avec son identifiant). Dans le cas d'une disponibilité de mise à jour de configuration ou de logiciel, D1 cible les équipements selon les critères présents dans la notification reçue. La cible est identifiée en fonction de contraintes figurant dans les conditions présentes dans la *notification*. Ces conditions sur les équipements peuvent concerner, par exemple, la version du logiciel déjà installée pour être apte à recevoir la nouvelle ou encore des éventuelles contraintes matérielles (compatibilité avec les objets datant d'au moins 2015 par exemple).
- Une fois la (ou les) opération(s) étant générée(s) et la cible identifiée, la criticité de la mise à jour (présente dans la notification, détaillée dans la section 3.1.2) est analysée pour ordonnancer les différentes opérations dans l'étape (P1). Les opérations ayant une criticité plus élevée seront mises en haut de la file d'exécution. Notre approche ne prend en compte que la criticité dans son état actuel pour l'ordonnancement, mais il est possible de prendre en compte des critères de qualité de service dans le cas où les équipements appartiendraient à plusieurs propriétaires et seraient administrés par des tiers. Avec cette approche il est possible de privilégier les équipements de certains propriétaires en fonction du contrat de qualité de service qui les lie avec l'entité en charge de l'administration de ces équipements.
- L'étape (P2) concerne la régulation de la taille de la liste des cibles des sous-opérations générées par ce gestionnaire autonomique à chaque pas. Cette régulation concerne donc la vitesse d'exécution des opérations. Chaque opération est décomposée ici en sous-opérations. La taille des sous-opérations

en termes de nombre d'équipements à mettre à jour, est calculée par le gestionnaire autonomique (RegulVitesse) *Régulation de la vitesse d'exécution*. Celle-ci est déterminée en fonction de la variation du nombre d'erreurs critiques et non critiques, provenant du résultat d'exécution des commandes sur les équipements déjà traités. Quand il s'agit de la première sous-opération, la cible est initialement fixée à 1% [1] du nombre total d'éléments de la flotte avant d'être ajustée durant l'exécution de l'opération. Cette liste de sous-opérations à réaliser est envoyée à l'étape E1 dans la variable D3.

— La dernière étape de la prise de décision concerne l'envoi (SousOps) de ces sous-opérations à exécuter au gestionnaire d'exécution et de suivi (Exec-Suivi).

4.2.7 Base de connaissance

Afin de prendre la décision de lancer des opérations d'administration système sur le parc géré, il est impératif pour ce gestionnaire d'avoir les informations sur l'état actuel des équipements qui le composent. Cependant, il est peu efficace de garder à jour cet ensemble d'informations d'état en effectuant des requêtes sur tout le parc de manière périodique. C'est dans cette optique que la base de connaissance intégrée dans ce gestionnaire contient ces informations en s'appuyant sur le paradigme du jumeau numérique défini dans la section 3.1.4. Ce paradigme permet dans notre cas d'usage de stocker l'état d'un objet sous la forme d'une représentation numérique de ce dernier selon les informations les plus récentes reçues de sa part. Toute modification de son état entraîne (p. ex. Géolocalisation, services activés) de la part de l'objet, l'envoi d'une notification mettant à jour l'état de son jumeau.

La figure 4.7 est un exemple de représentation d'une ampoule connectée dans la base de connaissance de notre système autonomique. Elle est divisée en trois parties : les informations sur l'objet, les marqueurs (Tags) comprenant la localisation dans ce cas, et les propriétés contenant les informations sur cette ampoule en termes de fréquence d'envoi de données ('5mins') ou de couleur ('purple').

La base de connaissance de ce gestionnaire contient l'état de tous les équipements du parc sous forme de jumeaux numériques.

La plateforme de recherche Thing'In [32] d'Orange est un cas réel de graphe d'objets, de leur représentation et leurs relations. Cette plateforme pourrait être utilisée comme support de stockage des états des équipements et comme base de connaissance permettant d'effectuer des inférences et raisonnements pour optimiser les opérations d'administration.

Des exemples de cet usage sont les suivants :

— Identifier comme "prévues" les coupures ou perturbations en cascades liées à une opération, redémarrer le routeur après mise à jour rend les équipements qui en dépendent pour la connectivité hors ligne pendant la durée de l'opération.

1. ou une autre valeur déterminée par les administrateurs systèmes

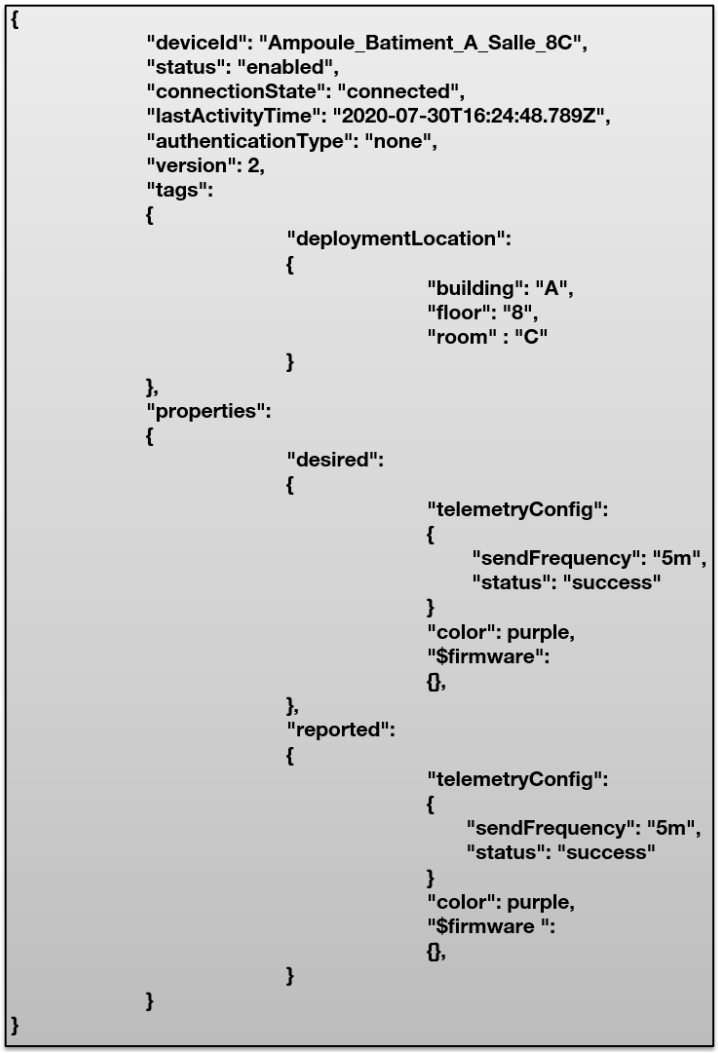

FIGURE 4.7 – Exemple de représentation d'un équipement dans la base de connaissance

— Cibler les équipements en priorisant certains en fonction de leur propriétaires déclarés dans Thing'In.

Elle contient aussi la liste des opérations en cours et la liste des sous-opérations générées en décomposant les opérations sélectionnées pour exécution. La figure 4.8 détaille le schéma de données de la base de connaissances du gestionnaire de génération et décomposition des opérations. Les opérations sont identifiées par leur 'idOp', leur nature (Lecture, écriture, suppression, mise à jour), leur type (mise à jour de configuration ou de logiciel), de la valeur dans un cas d'écriture ou de mise à jour, leur type d'équipement ciblé et leur état (réussie, échouée, en cours).

Les sous-opérations reprennent les mêmes données que la structure mère en ajoutant la liste des équipements ciblés pour cette sous-opération.

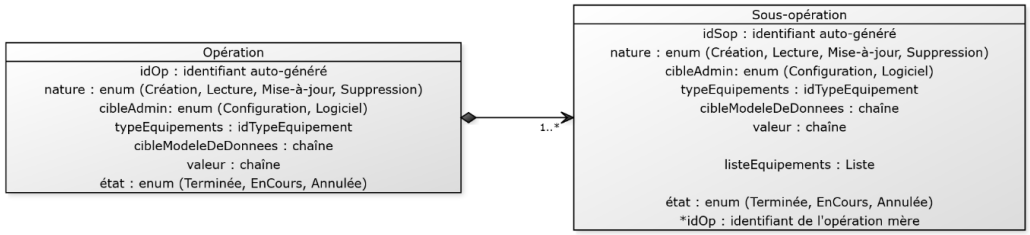

FIGURE 4.8 – Représentation des opérations et sous-opérations dans la base de connaissance

4.2.8 Interactions

Ce gestionnaire autonomique interagit avec les deux autres gestionnaires autonomiques (ExecSuivi) et (RegulVitesse). Cela se fait d'une part au travers de l'entrée EtatParc comprenant la taille de décomposition en sous-opérations de chaque opération en cours venant de (RegulVitesse) et d'autre part via la sortie S de ce gestionnaire allant vers celui d'exécution et suivi (ExecSuivi) indiqué dans la figure 4.6.

Grâce à cette coordination, en cas d'erreurs sur la configuration ou au niveau du logiciel interne, au travers de l'adaptation continue de la vitesse de décomposition, moins d'équipements sont impactés. Dans les approches existantes, la vitesse est constante et maximale et la surveillance est manuelle. Cela mène à un temps de réaction élevé et une complexité de la surveillance du parc. Cela fait que plus d'équipements sont susceptibles d'être en dysfonctionnement.

Les sous-opérations citées plus haut, sont envoyées aux gestionnaires de décomposition et suivi d'opérations pour être exécutées sur les équipements du parc. Cette séparation en sous-opérations permet de faire en sorte que la partie exécution et suivi des opérations d'administration de flotte passe à l'échelle horizontalement et ce en multi-instanciant le gestionnaire (ExecSuivi).

4.3 Conclusion

Nous avons détaillé dans ce chapitre les gestionnaires (ExecSuivi) et (GenerOp) gérant l'automatisation de l'administration d'un parc d'équipements. L'aspect métier ou fonctionnel de l'administration système étant traité, nous visons maintenant la problématique concernant la gestion optimisée des ressources. Ces ressources sont d'un côté, les objets administrés (assurer que la charge n'est pas supérieure à leur capacité à recevoir des opérations d'administration), et d'un autre le matériel composant l'infrastructure hébergeant la plateforme d'administration des équipements. C'est dans cette optique que notre approche inclut deux autres gestionnaires autonomiques chargés de la gestion des ressources : (RegulVitesse) et (Deploy). Ces derniers ont respectivement pour responsabilité, la régulation de la vitesse de décomposition des sous-opérations en fonction des erreurs d'exécution du parc d'équipements et de

la gestion optimisée des ressources de l'infrastructure matérielle.

Chapitre 5

Gestion des ressources

Sommaire

Dans ce chapitre, nous traitons deux types de ressources : les équipements administrés et l'infrastructure hébergeant la plateforme d'administration de ces équipements. Il s'agit ici de résoudre deux problèmes :

— Adapter la vitesse d'exécution des opérations générées par (GenerOp) et exécutées par (ExecSuivi) à la capacité de traitement disponible des plateformes d'administration des équipements.

— Déployer les plateformes d'administration et les gestionnaires d'exécution (ExecSuivi) dans les partiez de l'infrastructure ayant les ressources de calcul suffisantes pour les héberger, tout en restant au plus proche des équipements

(pour une consommation de bande passante réseau optimisée et une assurance de la protection des données privées accrue).

Chacun de ces problèmes est adressé par un gestionnaire autonomique. Nous abordons dans ce chapitre les deux gestionnaires autonomiques (RegulVitesse) et (Deploy) responsables respectivement du passage à l'échelle vertical et horizontal. Nous commençons par celui en charge de la régulation de la vitesse de décomposition d'opérations en sous-opérations avant de détailler les spécifications de celui qui gère l'infrastructure.

Dans ce chapitre, les gestionnaires autonomiques sont exposés au format MAPE-K. Le nom des modules correspondant à une étape est identifié par la première lettre de l'étape (M, A, P, E) suivi d'un chiffre s'il y a plusieurs sous-étapes. Par exemple la première étape du (Monitoring) sera nommé M1 dans les schémas.

5.1 Régulation de la vitesse de décomposition des sous-opérations

Ce gestionnaire autonomique, (RegulVitesse), confère au système une capacité d'auto-configuration. Ce dernier devient apte à réguler la vitesse d'exécution des opérations d'administration système.

Afin d'avoir une régulation précise, la vitesse de décomposition calculée dépend de l'avancement de l'opération, l'accroissement peut être linéaire, quadratique, exponentiel. Ces informations sont déterminées par les administrateurs systèmes dans un premier temps dans la base de connaissance de la boucle, puis peuvent être affinées et mises à jour au fur et à mesure de l'exécution au moyen des statistiques récoltées lors de l'exécution. En plus de l'avancement, cette régulation dépend de statistiques collectées par le gestionnaire (GenerOp) de *génération et décomposition d'opérations*.

Deux buts principaux motivent cette régulation continue. Le premier est d'éviter qu'une configuration erronée ou un logiciel interne défaillant ne soit déployé sur une partie conséquente du parc ou son intégralité entraînant des dysfonctionnements sur les équipements. Le deuxième concerne l'adaptation de la vitesse d'exécution par rapport aux capacités des plateformes à traiter les opérations de mise à jour.

5.1.1 Objectif

L'objectif de ce gestionnaire est de garder le nombre d'équipements en bon état de fonctionnement en nombre maximal et une bonne utilisation des ressources disponibles pour les plateformes d'administration à un moment donné. Pour ce faire, il adapte la vitesse de décomposition des opérations en sous-opérations (en termes de nombre d'équipements ciblés), au nombre d'erreurs sur les équipements mis à jour et des temps de réponses de la plateforme.

5.1.2 Pas

À chaque réception de nouvelles statistiques d'erreurs le gestionnaire fait un tour de boucle pour mettre à jour le nombre d'équipements ciblés par sous-opération générée par (GenerOp). Il s'agit d'un déclenchement événementiel.

5.1.3 Données d'observation

Ce gestionnaire prend en entrée les statistiques concernant les opérations en cours d'exécution au niveau du gestionnaire (ExecSuivi) *d'exécution et de suivi des commandes*. Ces statistiques contiennent celles qui concernent les équipements et celles concernant les plateformes.

Les données d'équipements sont le nombre d'erreurs par type (critique ou non critique). Dans notre modélisation, les erreurs critiques reflètent le nombre d'équipements en dysfonctionnement après mise à jour. Celles non critiques reflètent les objets présentant des métriques de fonctionnement en écart par rapport à la moyenne (p. ex. consommation mémoire, temps de démarrage).

La donnée de plateformes récoltée est le temps de réponse moyen des API. C'est l'information clé pour assurer le passage à l'échelle vertical de la plateforme d'administration car il reflète la capacité de leurs interfaces à recevoir ces commandes d'administration.

Le nombre d'erreurs tolérées ainsi que les écarts par rapport à la moyenne, sont définis par les administrateurs systèmes.

.

```
{
  "statistiques": {
    "idStat": "00256120",
    "idOperationMere": "512400_ORANGE",
    "idSousOperationMere": "512400_ORANGE-P1",
    "currentAvgAPIresponseTime" : "120ms",
    "stats": {
      "criticalErrrors": "1",
      "nonCriticalErrrors": "5",
      "TotalCommands": "25"
        }
    }
}
```

Structure 5.1 : Données d'erreurs d'execution des commandes

La structure 5.1.3 détaille le format des statistiques envoyées par le gestionnaire (ExecSuivi) au gestionnaire (RegulVitesse). Ces statistiques sont caractérisées par leur identifiant propre, l'identifiant de la sous-opération qu'elles concernent et l'opération mère de cette dernière. Les statistiques sont sous le format : nombre d'erreurs par type et nombre total de commandes.

5.1.4 Actionneurs

Ce gestionnaire autonomique agit sur la taille (en termes du nombre d'équipements ciblés) des sous-opérations générées par le gestionnaire (GenerOp) de génération et décomposition. Avec des techniques de contrôle continu et une modélisation adéquate, cette vitesse pourrait être ajustée dynamiquement durant l'exécution (section 2.3.2).

5.1.5 Élément Géré

Ce gestionnaire a pour élément géré les gestionnaires d'automatisation (ExecSuivi) et (GenerOp) ainsi que leurs éléments gérés respectifs, comme exposé dans la figure 5.1.

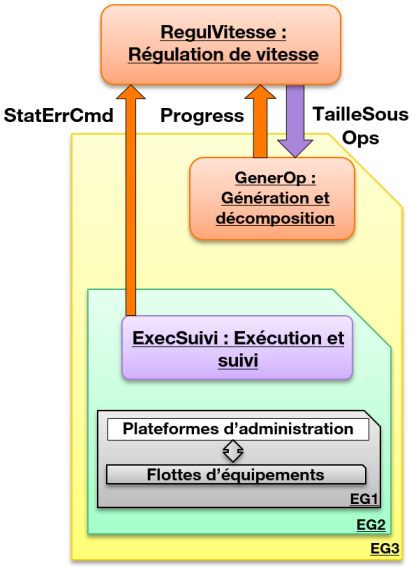

FIGURE 5.1 – Élément géré du gestionnaire de régulation de vitesse

5.1.6 Décision

La figure 5.2 détaille l'architecture (au format MAPE-K) du gestionnaire (RegulVitesse). Il possède deux entrées : 'StatErrCmd' en provenance du gestionnaire (ExecSuivi) et 'Progress' du gestionnaire (GenerOp). Son unique sortie est 'TailleSousOp' à destination du gestionnaire (GenerOp).

Dans le processus de décision, les opérations sont caractérisées par leur phase et leur état contenus dans la donnée d'entrée 'Progress'.

— État : caractérisant le fait qu'une opération soit en pause, en cours, terminée ou annulée.

— Phase : caractérisant la progression de ladite opération, sont dans l'ordre : *Test*, entre 0 et 5%, phase où la progression est lente et l'observation accrue.

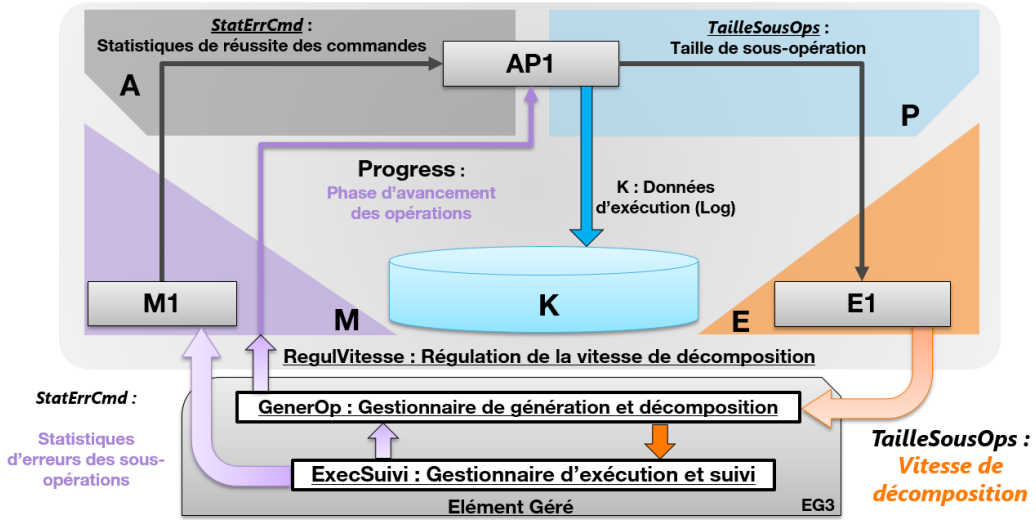

FIGURE 5.2 – Gestionnaire (RegulVitesse) de régulation de la vitesse de décomposition des opérations en sous-opérations

Prudente , entre 5 et 30%, ou la progression est légèrement plus rapide que la phase précédente. *Généralisation*, entre 30 et 100%, phase où le but est d'exécuter l'opération d'administration système sur le maximum d'équipements.

Le processus de prise de décision dans ce gestionnaire, se base sur les statistiques 'StatErrCmd' reçues en entrée et il est régi par l'algorithme suivant exécuté dans l'étape (AP1) pour chaque opération en cours :

1. Lire dans la base de connaissance la phase de l'opération traitée.
2. Si statistiques d'erreurs > taux (ou écart) autorisé : Annuler l'opération (c.a.d. envoyer la valeur -1).
3. Sinon si statistiques en amélioration : Augmenter la taille des sous-opérations (c.a.d. envoyer la nouvelle valeur).
4. Sinon si statistiques stables : Pas de variation (c.a.d. envoyer la valeur 0).
5. Sinon si statistiques en dégradation : Baisser la taille des sous-opérations (c.a.d. envoyer la nouvelle valeur).

La régulation se fait ici par seuil. En ayant recours à de la régulation à base de contrôle continu (section 2.3.2), nous pouvons envisager une régulation plus fine de cette taille de sous-opérations [63] [53].

En fonction de la phase dans laquelle est une opération donnée, l'accroissement sera plus ou moins prononcé. La variation dans les deux premières phases est plus maîtrisée afin de permettre aux anomalies latentes d'apparaître et de n'accélérer le déploiement qu'après une phase prudente. Cela est inspiré de l'approche existante mais effectuée manuellement par Orange dans le cadre de l'administration de flottes d'équipements domestiques de télécommunications.

Les statistiques sont stockées pour inventaire et historique dans notre approche (Données K sur une flèche unidirectionnelle dans la figure 5.2). Une piste d'amélioration serait d'utiliser les statistiques qui concernent des opérations similaires comme référence afin de calculer de manière plus précise les tailles de sous-opérations. Cela confèrerait à notre système autonomique des capacités d'auto-amélioration.

Une fois les tailles de sous-opérations calculées pour toutes les opérations en cours, elles sont envoyées vers le gestionnaire (GenerOp) dans la sortie sous la forme du vecteur :

TailleSousOps : < 'Operation' , 'TailleSousOp' >

5.1.7 Base de connaissance

La base de connaissance de ce gestionnaire contient deux types de données. La liste des opérations en cours et leur informations ainsi que la liste des statistiques reçues.

Pour chaque opération, les taux d'erreurs associés à chaque sous-opération sont ajoutées à la structure 'statistiques' qui cumule les statistiques reçues à chaque tour de boucle. Ce schéma de données est affiché dans la structure 5.1.7.

```
{
  "statistiques": {
    "idStat": "00256120",
    "idOperationMere": "512400_ORANGE",
    "idSousOperationMere": "512400_ORANGE-P1",
    "Progression" : "50,87%",
    "currentAvgAPIresponseTime" : "120ms",
    "cumulatedStats": {
      "criticalErrrors": "1",
      "nonCriticalErrrors": "5",
      "TotalCommands": "25"
      }
    }
}
```

Structure 5.1.7 : Contenu de la base de connaissance du gestionnaire RegulVitesse

La structure 5.1.7 détaille le format de la structure *statistiques* utilisée dans la base de connaissance de (RegulVitesse) afin de stocker les données envoyées par le gestionnaire (ExecSuivi). Ces statistiques sont caractérisées par leur identifiant propre, l'identifiant de la sous-opération qu'elles concernent et l'opération mère de cette dernière ainsi que la progression de l'opération mère. Cette progression permet d'ajuster la régulation en fonction de l'avancement de l'opération. Les statistiques sont sous le format : nombre d'erreurs par type et nombre total de commandes. Le

temps de réponse moyen mesuré des plateformes fait aussi partie de cette structure de données.

5.1.8 Interactions

Le gestionnaire autonomique (RegulVitesse), pour limiter la propagation de ces erreurs d'un côté et pour assurer un passage à l'échelle vertical d'un autre, adapte la vitesse de décomposition des opérations en sous-opération de manière continue. Pour ce faire, il interagit avec les deux gestionnaires contribuant à l'automatisation de l'administration du parc comme suit : Il fournit à celui en charge de la *génération de décomposition d'opérations* (GenerOp) une taille de sous-opérations à chaque fois qu'il reçoit des statistiques venant du gestionnaire (ExecSuivi) *d'exécution et de suivi des commandes* et ce en fonction des erreurs qui peuvent survenir sur les équipements et ce à cause de configurations ou logiciels internes défectueux.

Le passage à l'échelle vertical est adressé avec ce gestionnaire. C'est donc dans la prochaine section que nous abordons les spécifications du gestionnaire autonomique (Deploy) en charge du passage à l'échelle horizontal.

5.2 Gestion de l'infrastructure

Nous détaillons dans cette section le gestionnaire autonomique (Deploy) en charge d'adapter le plan de déploiement (en termes de nombre d'instances et de localisations) des gestionnaires d'exécution et suivi (ExecSuivi) et des plateformes d'administration sur les différents nœuds de calcul de notre infrastructure.

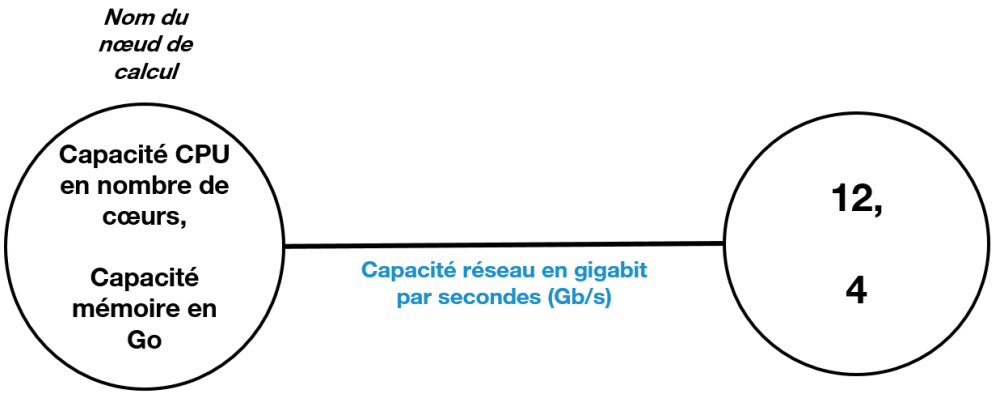

FIGURE 5.3 – Représentation d'un nœud de calcul dans notre modélisation

Dans ce qui suit, nous représentons l'infrastructure comme un graphe non orienté. Les nœuds sont représentés dans la figure 5.3. Chaque nœud de ce graphe comporte un couple <capacité de calcul processeur, capacité mémoire>. Les arêtes entre les nœuds représentent la capacité du lien réseau entre ces derniers. Les capacités pro-

cesseurs sont exprimées en nombre de cœurs. Les capacités mémoires en Gigaoctets et la bande passante réseau en gigabits par secondes.

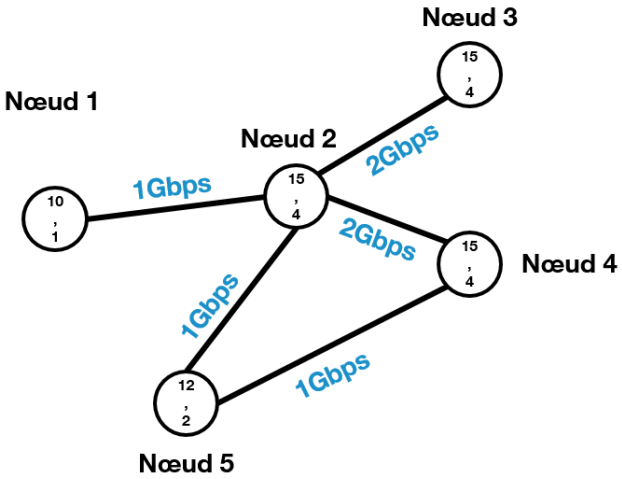

FIGURE 5.4 – Représentation du graphe de l'infrastructure dans notre modélisation

En assemblant les nœuds de calcul et les liens réseaux composant notre infrastructure matérielle disponible, on obtient la représentation de la figure 5.4.

Les applications : plateformes d'administration et les gestionnaires autonomiques sont représentés par un quadruplet
< consoCPU,consoMémoire,consoBandePassante,tempsDeRéponse >.

En voici un exemple pour une plateforme d'administration Orange, qui consomme 1,5 cœurs de processeur, 1 gigaoctet de mémoire, 256Mbit/s de bande passante réseau, et qui présente un temps de réponse de 200ms.

OrangeAdminInstance1 < "1.5","1","0.256","200ms" >

5.2.1 Objectif

L'objectif de ce gestionnaire autonomique est d'assurer que les indicateurs de performances se maintiennent à un niveau acceptable tout en minimisant l'utilisation de la bande passante réseau. Cela est réalisé en favorisant l'utilisation des ressources présentes au plus proche des équipements tant que la capacité des dits équipements le permet. Si la capacité (selon les seuils définis par les administrateurs systèmes) est dépassée ou que les objectifs de qualité de service n'est pas atteinte, les composants logiciels d'administration sont migrés vers le nœud le plus proche pouvant les accueillir. Les seuils en question dépendent du niveau de qualité de service négocié avec les utilisateurs de la plateformes d'administration système. On peut citer à titre d'exemple d'indicateurs, les temps de réponses, le nombre d'équipements

traités par heure ou encore les taux d'utilisation des différents nœuds.

5.2.2 Pas

Ce gestionnaire fonctionne de manière périodique et la période est fixée par l'administrateur système.

5.2.3 Données d'observation

Ce gestionnaire observe plusieurs métriques. Nous définissons deux types de métriques : Les métriques applicatives et les métriques d'infrastructure. Les premières servent à évaluer les performances de l'applicatif et sa capacité à pouvoir passer à l'échelle, les deuxièmes représentent l'utilisation de la capacité matérielle de calcul disponible ainsi que la bande passante réseau. Ce choix d'avoir deux types de métrique est motivé par deux raisons :

— La capacité à passer à l'échelle de la partie logicielle des boucles autonomiques et des plateformes d'administration système n'est pas assurée. Il faut donc observer des métriques applicatives appropriées : Nous avons sélectionné le temps de réponse des applications.

— Au-delà de l'aspect passage à l'échelle de l'applicatif, l'infrastructure peut être utilisée à 100% ou être sous-utilisée sans que ça soit observable via les métriques applicatives. Ce comportement est détecté au moyen des métriques, dites d'infrastructure : consommation CPU, mémoire et réseau.

```
{
  "Antenne5G_4580": {
    capacity :
      [6,4,1],
    applications : [
    {
      OrangeAdminInstance1 {"1.5","1","0.256","200ms"},
      OrangeAdminInstance2 {"1","2","0.512","250ms"},
      AmazonWSIoTInstance1 {"1","1.5","0.128","70ms"},
    }
    ]
  }
}
```

Données d'état de l'infrastructure reçues par le gestionnaire (Deploy)

La structure 5.2.3 comporte pour chaque nœud de calcul, sa capacité totale, les applications hébergées (avec leur consommation de ressources et leur temps de réponse).

Ces données permettent au gestionnaire (Deploy) de revoir, à la hausse ou à la baisse, le nombre d'instances des plateformes d'administration et des gestionnaires (ExecSuivi) ainsi que leur localisation dans les nœuds de l'infrastructure.

5.2.4 Actionneurs

Ce gestionnaire actionne des scripts écrits par les administrateurs systèmes qui agissent directement sur l'infrastructure et instancient les plateformes de DM et les boucles d'automatisation associées.

5.2.5 Élément Géré

Ce gestionnaire autonomique agit sur les API de contrôle de l'infrastructures :
— Le nombre d'instances de plateformes d'administration systèmes (et les boucles d'automatisation nécessaires à leur pilotage).
— Leur placement dans les différentes parties de l'infrastructure

```
{
  "Antenne5G_4580": {
    applications : [
      {
        OrangeAdminInstance1,
        OrangeAdminInstance2,
        AmazonWSIoTInstance1
      }
    ]
  }
}
```

Schéma de deploiement généré par la boucle (Deploy)

La structure 5.2.5 détaille le format de la sortie du gestionnaire (deploy). Cette structure représente le plan de déploiement des gestionnaires d'exécution et de suivi (ExecSuivi) et des plateformes d'administration. Chaque nœud (identifié par son identifiant unique) comprend la liste des identifiants des applications à héberger. Une fois un identifiant d'application déployé quelque part, son ancienne instance est détruite.

Lors de la re-configuration du schéma de déploiement, les plateformes d'administration de flotte ne changent pas d'adresse IP, les re-configurations sont donc transparentes pour les équipements.

5.2.6 Décision

Le processus de prise de décision de ce gestionnaire est le suivant :
— Dans le cas où aucun couple plateforme de DM et boucles de régulation n'est présent, le gestionnaire autonomique instancie par défaut ; un de ces couples dans la partie de l'infrastructure la plus proche des équipements gérés. Par exemple, déployer ces composants logiciels dans un nœud de calcul présent dans un site d'une usine, pour la gestion des caméras IP de ce site.
— Vérification des indicateurs de performance applicatifs. Si ces mesures augmentent (p. ex. temps de réponse qui augmente fortement, ou code d'erreurs

indiquant une saturation), il est nécessaire d'augmenter le nombre d'instances d'éléments gérés dans les parties de l'infrastructure où de la capacité (calcul, stockage, mémoire) est encore disponible.

— Vérification des indicateurs de performances au niveau de l'infrastructure et détecter une éventuelle violation des **SLA**. Il convient alors de migrer les éléments gérés qui sont devenus trop gros pour le matériel qui les héberge vers l'élément de l'infrastructure le plus proche ayant une capacité supérieure disponible à ce moment.

Cette prise de décision comme pour le gestionnaire RegulVitesse, est prise par seuils définis par les administrateurs. Afin de concevoir un régulateur à base de contrôle continu (section 2.3.2), il est nécessaire de modéliser le système visé et d'identifier les métriques ainsi que les variables d'ajustement. Le comportement du système doit aussi être exploré et décrit. C'est cette étape qui a été réalisée durant ce travail de thèse, préparant ainsi le terrain pour la suite des travaux visant à aller vers un contrôle plus fin et plus précis.

5.2.7 Interactions

Ce gestionnaire autonomique (Deploy) est indépendant des gestionnaires d'automatisation. Il a pour but de gérer les instances de plateformes d'administration système et des gestionnaires d'automatisation qui leurs sont associés. Ces derniers n'interagissent pas avec le gestionnaire de distribution et n'ont pas conscience de son existence.

5.2.8 Base de connaissance

La base de connaissance de ce gestionnaire autonomique contient une représentation de l'infrastructure sous forme de graphe. Le graphe comprend la capacité restante sur les noeuds. Cette capacité est actualisée en fonction de la taille des applications (Gestionnaires autonomiques d'exécution et plateforme d'administration) qui y fonctionnent. Ces données sont fournies par un service de l'infrastructure. La figure 5.5 donne la représentation de ce graphe. Elle représente une architecture formée de plusieurs nœuds de calculs réparties entre les ressources qui sont les plus proches des équipements que nous nommons "Edge", et le "Cloud" représentant les ressources de calcul et de réseau disponibles chez un opérateur. Nous considérons le continuum entre le "Cloud" et "l'Edge" comme étant le "Fog" qui comporte les ressources de tailles petites et moyennes (c.a.d. quelques cœurs de calculs et quelques giga-octets de mémoire vive).

5.3 Conclusion

Dans ce chapitre nous avons détaillé les gestionnaires en charge de la gestion des ressources. Chacun d'entre eux est conçu pour adresser un des sous-problèmes que

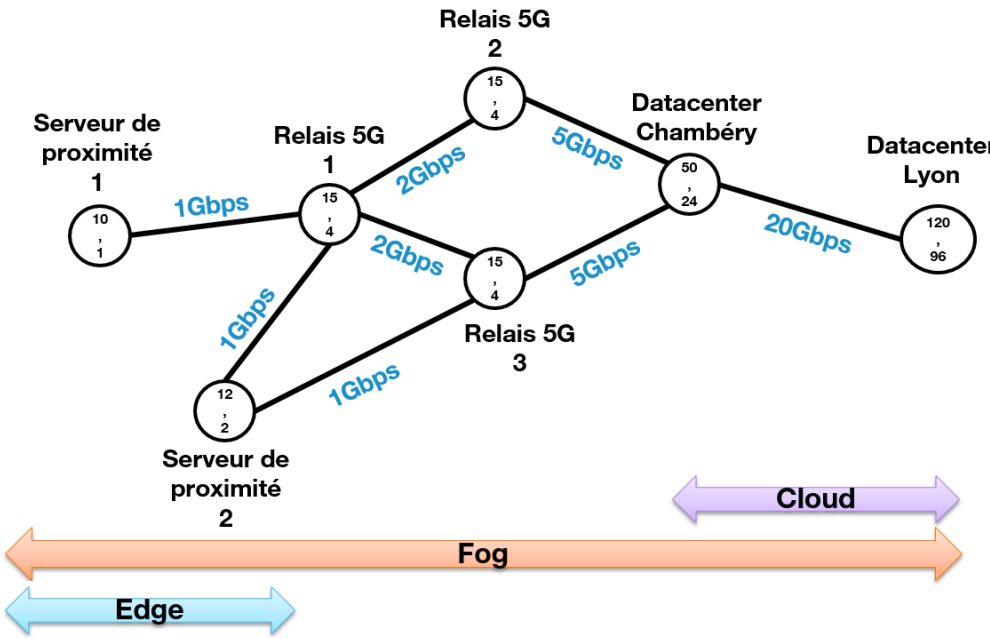

FIGURE 5.5 – Représentation de l'infrastructure dans la base de connaissance sous forme de graphe

nous avons identifiés. L'adaptation de la vitesse d'exécution des opérations d'administration à la capacité des équipements à recevoir ces ordres d'un côté, ainsi que l'adaptation du déploiement des plateformes et gestionnaires autonomiques d'exécution et suivi aux ressources disponibles de l'infrastructure de l'autre.

D'autre critères possibles n'ont pas étés pris en compte comme ceux de la confidentialité des données qui feraient que certains nœuds ne soient pas utilisables concernant certaines flottes en fonction de contraintes légales par exemple.

Troisième partie

Validation Expérimentale

Dans cette partie, nous détaillons les dispositifs mis en place pour la validation de notre approche qui est détaillée dans la deuxième partie de ce manuscrit. Il est question de deux expérimentations, l'une démontrant l'automatisation de la gestion des départs et arrivées d'équipements dans des flottes au moyen d'un gestionnaire autonomique pilotant la plateforme d'IdO d'Orange Live Objects utilisée en production et l'autre démontrant la réactivité du système d'administration autonomique à des erreurs provenant de la flotte. Ce démonstrateur est mis en œuvre avec trois gestionnaires autonomiques interfacés avec des composants logiciels ayant pour rôle de simuler des objets de l'IdO et leur plateforme d'administration système.

Chapitre 6

Maintien à jour d'un parc d'objets connectés avec Live Objects

Sommaire

Ce chapitre détaille le démonstrateur d'automatisation du maintien à jour en termes de configurations et de logiciel interne, d'une flotte d'équipements subissant

des variations. Ces variations sont des départs ou arrivées d'équipements. Nous commençons par déterminer l'objectif de ce démonstrateur, les motivations en termes de choix technologiques, l'architecture et le protocole de test. Nous terminons ce chapitre par l'analyse des résultats et en citant les limitations du protocole expérimental utilisé.

6.1 Objectif de la validation

Ce démonstrateur a pour but de démontrer la capacité à gérer les variations de l'état d'un parc en termes d'arrivées et départs d'équipements ainsi que de disponibilité d'un logiciel interne ou configuration d'un parc d'équipements. Ces équipements sont administrés par une plateforme industrielle d'administration à distance d'objets de l'IdO.

6.2 Technologies existantes et choix

Cette section détaille les choix que nous avons réalisés en termes d'architecture et de choix de langages pour le développement.

6.2.1 Plateforme d'administration

Certaines des plateformes industrielles d'IdO existantes telles que IBM Watson [19], Amazon IoT [11] et Microsoft Azure IoT [22], comportent des capacités d'administration système. Ces capacités concernent le déploiement de configurations et logiciels internes ainsi que le lancement d'opérations visant de multiples équipements : campagnes de mise à jour. La plateforme d'Orange : Live Objects [29] a pour objectif d'augmenter et perfectionner les fonctionnalités d'administration système de flottes. Nous nous sommes orientés vers l'utilisation de cette plateforme pour nos démonstrations et preuves de concept.

En fonction du niveau de fonctionnalités (c.f. Sections 2 et 6 du chapitre 1) et de faisabilité technique, il est possible d'automatiser à différents degrés les capacités d'administration de flottes de plateformes existantes au moyen de notre proposition d'architecture autonomique.

Étant données les capacités d'administration de la plateforme Live Objects, qui couvrent l'exécution d'opérations, la traduction de ces opérations en commandes et les campagnes de mise à jour, nous nous sommes fixés d'y rattacher un gestionnaire autonomique ayant pour objectif le maintien à jour du parc d'objets via la génération d'opérations d'administration. Il s'agit donc de mettre en œuvre le gestionnaire autonomique (GenerOp), chargé d'automatiser la génération d'opérations, capacité qui n'est pas dans les fonctionnalités offertes par Live Objects au moment de l'écriture de ce manuscrit.

Un autre aspect qui a motivé le choix de cette plateforme est le fait d'avoir une expertise interne Orange sur cette plateforme d'IdO. Dans le cadre de cette thèse CIFRE, l'objectif est d'exploiter les résultats des travaux de recherche dans

des démonstrateurs afin d'en démontrer la possibilité d'intégration des dits travaux dans des plateformes utilisées actuellement en production. L'objectif, à terme, est de faire évoluer, sur la base de ces travaux, les capacités d'administration de flottes de la plateforme Live Objects [29]. Dans ce qui suit, nous détaillons les fonctionnalités de Live Objects (ainsi que les autres plateformes de l'IdO) concernant l'administration de flottes.

6.2.2 Plateforme IdO Live Objects

6.2.2.1 Présentation

FIGURE 6.1 – Plateforme IdO d'Orange : Live Objects

Live Objects d'Orange est une plateforme cloud destinée aux professionnels permettant de collecter et stocker les données venant des objets de l'IdO. Elle intègre aussi des capacités d'administration de ces objets. Elle sert de pont entre les applications dites métier (p. ex. Application de suivi de colis) et objets de l'IdO (p. ex. Traceurs de position géographique).

6.2.2.2 Fonctionnalités

Live Objects permet d'inventorier les objets, gérer les identités, assurer le stockage et transfert des données émanant de ces objets. En termes d'administration d'équipements, la plateforme permet l'envoi de configurations et logiciels aux équipements ainsi que les *campagnes de mise à jour*. Live Objects fonctionne de la manière suivante : Les objets sont configurés pour envoyer leurs données vers la plateforme. Ils effectuent leurs transferts de manière périodique. Live Objects est configuré, par les utilisateurs de la plateforme, pour traiter et envoyer ces données reçues des objets vers leur système d'information. Si besoin, les utilisateurs peuvent lancer manuellement des opérations de configuration et de mise à jour sur leurs objets.

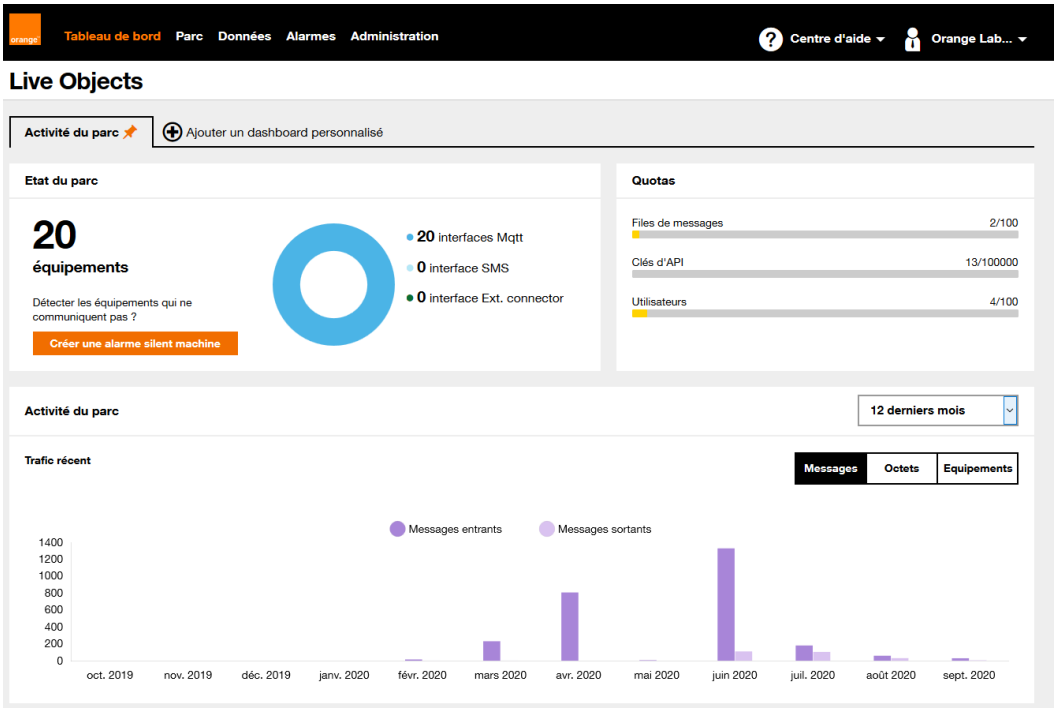

FIGURE 6.2 – Interface d'accueil de la plateforme IdO d'Orange : Live Objects

L'interface d'accueil de la plateforme est exposée dans la figure 6.2. Celle-ci affiche un résumé de l'état du parc en termes de nombre et de présence en ligne. Elle donne aussi accès aux fonctionnalités de gestion des données reçues par les équipements, et aux interfaces d'administration comme celle des figures 6.3 et 6.4.

L'interface de la figure 6.3 détaille la procédure de lancement de campagnes de mise à jour. En effet il faut sélectionner manuellement les bons paramètres dans cette interface comme par exemple : le type de connectivité à la plateforme, la liste des équipements ciblée ou une requête permettant de filtrer et cibler les équipements connectés (p. ex. Tous les équipements de marque Philips), le choix du type de campagne. Le type de campagne peut être statique ou dynamique. Les développeurs entendent par dynamique une campagne qui durant son exécution voit sa cible modifiée si des équipements ne répondent plus, ou se mettent à répondre, aux critères de la requête de ciblage.

Dans notre approche autonomique, la détermination de ces paramètres et le lancement d'une campagne est automatiquement effectué par le gestionnaire (GenerOp). C'est dans cette optique que nous avons implanté une version de ce dernier pour s'interfacer avec Live Objects.

L'interface de la figure 6.4 détaille l'interface d'historique de campagnes de mise à jour de la plateformes. En effet, après chaque exécution de campagne, les taux de succès sont enregistrés dans la base de données interne de la plateforme. Ces traces d'exécutions ne sont pas exploitées à ce jour dans la plateforme. En s'appuyant sur notre approche autonomique (plus précisément au moyen des gestionnaires d'auto-

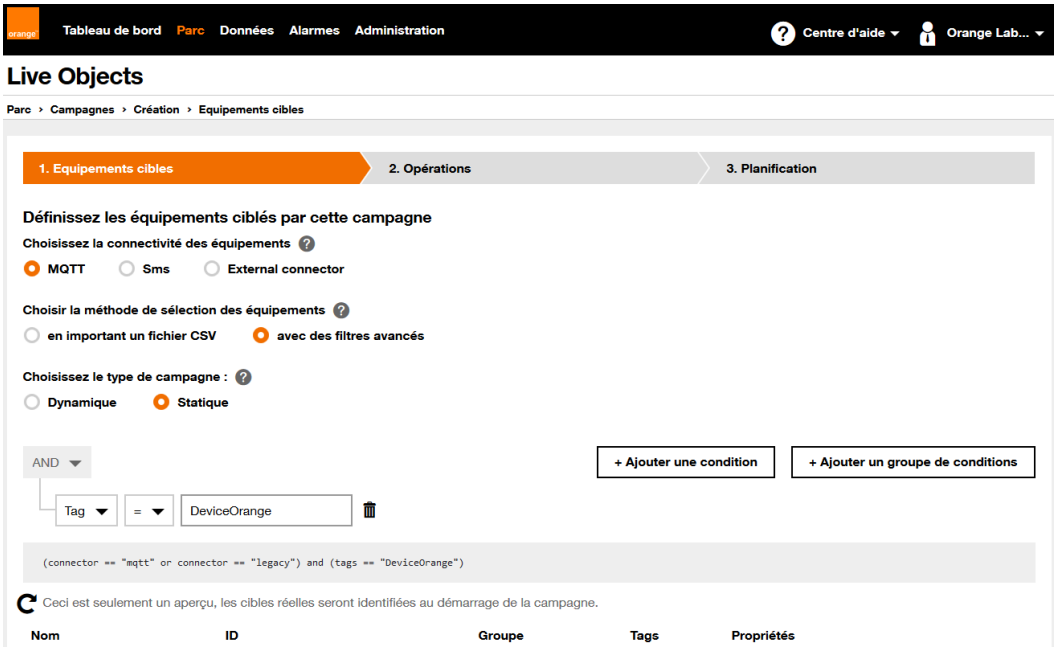

FIGURE 6.3 – Interface de création de campagnes de mise à jour de Live Objects

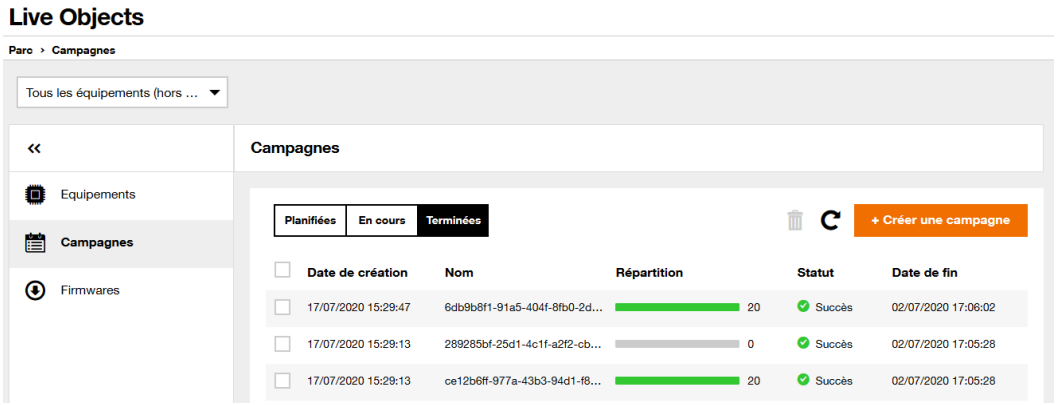

FIGURE 6.4 – Interface d'historique de campagnes de mise à jour de Live Objects

matisation GenerOp et ExecSuivi détaillés en 4.1 et 4.2) il serait possible d'exploiter ces données pour effectuer un suivi d'exécution de campagne en temps réel et détecter les fautes d'exécution et les opérations entraînant des dysfonctionnements sur des équipements du parc au moyen du gestionnaire (RegulVitesse).

6.2.3 Simulateur d'équipements

Afin d'accélérer le développement en évitant d'utiliser des équipements physiques complexes à mettre en œuvre, nous avons développé un simulateur d'équipement à partir d'un prototype existant chez Orange. Le prototype existant n'avait que la capacité de se connecter à la plateforme Live Objects sans déclarer autre chose qu'un

identifiant. Nous avons implanté la capacité à mettre à jour son logiciel interne, et celle de déclarer un modèle de données plus abouti formé de l'identifiant, la marque, les informations concernant la configuration et le logiciel interne de l'objet simulé. Nous avons aussi adapté le code à un lancement en ligne de commandes avec les bons paramètres contrairement à son état initial où cela n'était pas possible. Ce simulateur, dans sa version aboutie, possède donc la capacité de se connecter à la plateforme Live Objects ainsi que d'appliquer des commandes (pour rappel : Lecture, écriture, exécution, suppression) sur son modèle de données et son logiciel interne. Ce simulateur est développé en Python.

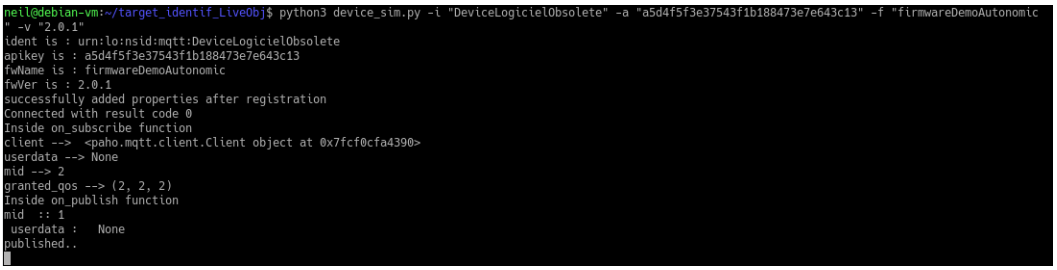

FIGURE 6.5 – Lancement du simulateur d'objet de l'IdO

La figure 6.5 montre la commande de lancement du simulateur et les traces d'exécution. Les paramètres nécessaires sont les suivants :

— i : identifiant de l'objet - "DeviceLogicielObsolete" dans l'exemple ;
— a : la clé d'accès à la plateforme - unique pour chaque objet ;
— f : nom du logiciel interne à utiliser - firmwareDemoAutonomic dans l'exemple ;
— v : version du logiciel interne - "2.0.1" dans l'exemple.

Une fois exécuté, le simulateur affiche les données qu'il a fournies à Live Objects durant son enregistrement dans la plateforme et le résultat de l'opération : 0 pour un succès, -1 pour un échec.

6.2.4 Gestionnaires autonomiques

Les gestionnaires autonomiques ont été développés dans le cadre de ces travaux de thèse à partir de zéro. En effet, à notre connaissance, aucune bibliothèque ou outil permettant d'automatiser des opérations d'administration système n'existe. Pour ce faire, nous nous sommes servis du langage Python qui offre des possibilités de prototypage rapide et des possibilités d'évolution du démonstrateur (p. ex. Interface Web, micro-services) comparé à ceux des langages similaires comme Shell ou PowerShell.

6.3 Architecture

La figure 6.6 détaille l'architecture utilisée pour ce démonstrateur. Le premier composant est une implantation du gestionnaire du maintien à jour du parc (GenerOp) qui génère des opérations de mise à jour lors de la disponibilité de nouveaux

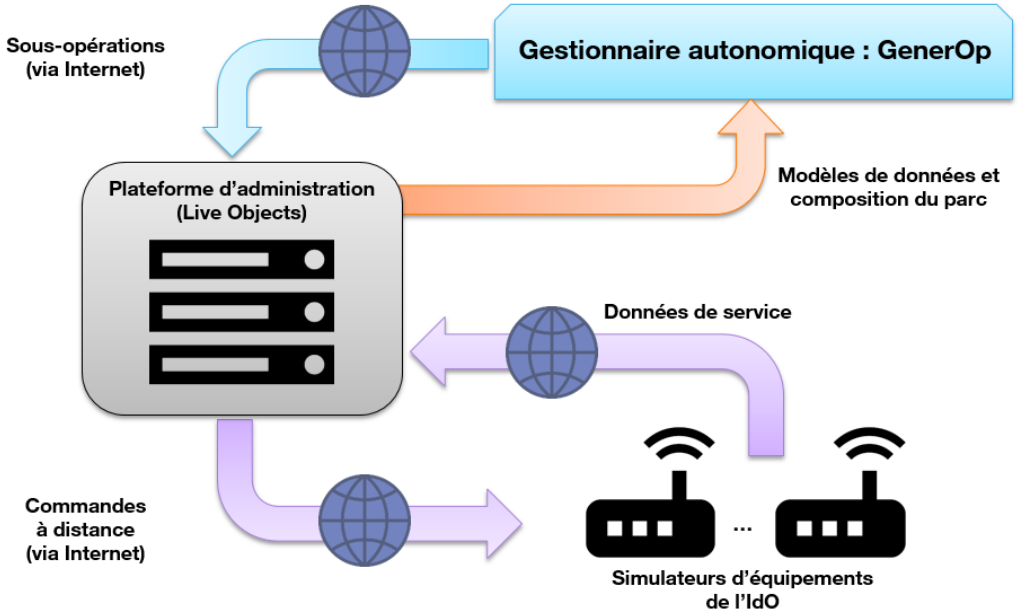

FIGURE 6.6 – Architecture du démonstrateur d'automatisation du maintien à jour d'un parc d'équipements de l'IdO

logiciels internes ou lors de l'arrivée d'objets dans la flotte qui n'ont pas la version déployée actuellement. Les fonctions de décomposition d'opérations en sous-opérations et de régulation de vitesse de décomposition n'ont pas été implantées dans ce prototype car trop complexes à mettre en œuvre sans avoir une instance de test de la plateforme, ce qui n'est pas notre cas car nous utilisons la version publique en production. Le deuxième composant est la plateforme IdO d'Orange dont nous utilisons les capacités d'inventaire d'objets et d'administration de flottes et plus particulièrement la capacité à traduire les opérations en commandes d'administration vers les équipements connectés. Le troisième et dernier composant est l'ensemble des simulateurs d'équipements que nous avons développés (détaillé en section 6.2.2). Ces derniers ont le même comportement que des objets physiques en termes de réaction à une commande de mise à jour logicielle.

6.4 Protocole de test

Le protocole de test à été défini pour couvrir deux des cas de perturbations possibles qui sont régulés par le gestionnaire (GenerOp)

- Modification de la composition de la flotte : Arrivée d'un nouvel équipement dans le cas de ce démonstrateur.
- Disponibilité d'un nouveau logiciel interne.

Le protocole de test est le suivant :

1. Lancement d'un ensemble de simulateurs de périphériques qui s'enregistre à

la plateforme ;

2. Injection d'une notification de nouveau logiciel interne disponible pour chaque type d'équipements dans le parc ;
3. En fonction de leur type, chaque équipement reçoit le logiciel adéquat ;
4. Un équipement nouveau se connecte à la plateforme, avec un logiciel interne obsolète, et déclenche un tour de gestionnaire autonomique ;
5. Le nouvel équipement reçoit le bon logiciel interne.

6.5 Résultats

Dans cette section nous montrons, pour chaque étape du protocole de test la capture d'écran correspondante venant de la plateforme Live Objects avant d'élaborer une synthèse sur les résultats extraits de ce démonstrateur.

6.5.1 Exécution

L'exécution du scénario de test est réalisée en cinq phases, chacune d'entre elles est détaillée dans ce qui suit.

6.5.1.1 Connexion des simulateurs à la plateforme Live Objects

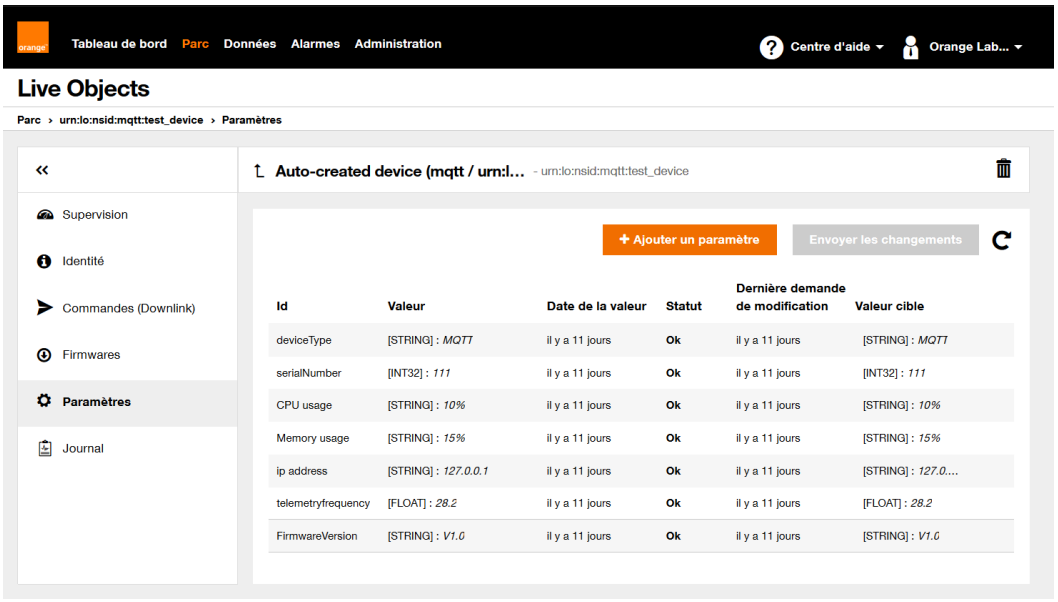

FIGURE 6.7 – État d'un objet une fois enregistré dans Live Objects

La figure 6.7 montre l'état d'un objet une fois enregistré dans Live Objects. Cet état comporte les informations de son modèle de données, ici : son type, son numéro de série, sa charge processeur, sa mémoire libre, son adresse IP, sa fréquence d'envoi de données et sa version de logiciel interne.

6.5.1.2 Arrivée d'une notification de disponibilité d'un nouveau logiciel interne

```python
import json, pika, uuid, datetime, time


connection = pika.BlockingConnection(pika.ConnectionParameters('localhost'))
channel = connection.channel()

newOperationID= uuid.uuid4()
timeNow = datetime.datetime.now()
timeS = timeNow.isoformat(timespec="seconds") + 'Z'

data = {
        "id": str(newOperationID),
        "developper": "Schneider",
        "binaryURL": "https://internal.orangeintranet/repoFW/test3_0-R0Schneider",
        "version":"3.0r0",
        "conditions" : {
            "firmwareOrigin": "2.0.1",
            "batteryMin": "70",
            "manufacturer": "Schneider"
            },
        "type": "minor",
        "timestamp": timeS

}
message = json.dumps(data)

channel.basic_publish(exchange='',routing_key='newFW',body=message)

connection.close()
```

FIGURE 6.8 – Envoi d'une notification de disponibilité de logiciel interne pour équipements Schneider

Les figures 6.8 et 6.9 exposent la structure de la notification (Section 2.1.8), que nous envoyons au système autonomique pour l'informer de la disponibilité de deux nouveaux logiciels internes : l'un visant les équipements Schneider, l'autre ceux fabriqués par Orange (champ 'manufacturer' dans les conditions). La structure de cette notification comporte un identifiant, le développeur du logiciel (ici Orange), l'adresse ou le binaire est disponible, la version (ici 3.0r0), le type de mise à jour suivant l'échelle proposée dans la section 3.1.3 (ici mineur), et les conditions de sélection des cibles parmi les équipements. Cette sélection se fait dans notre expérimentation avec la version du logiciel déjà installée (2.0.1), le niveau de batterie (minimum 70%) et le fabriquant : Schneider, comme dans la figure, ou Orange.

6.5.1.3 Lancement et exécution automatique de deux campagnes ciblées de mise à jour

La figure 6.10 montre les statistiques de réussite d'une campagne de mise à jour ciblant deux types d'objets dans Live Objects : Ceux fabriqués par Orange et ceux de Schneider. Chaque type reçoit une mise à jour différente. Dans l'existant, il est

target_identif_LiveObj > init.py > ...

```python
import json, pika, uuid, datetime, time


connection = pika.BlockingConnection(pika.ConnectionParameters('localhost'))
channel = connection.channel()

newOperationID= uuid.uuid4()
timeNow = datetime.datetime.now()
timeS = timeNow.isoformat(timespec="seconds") + 'Z'

data = {
        "id": str(newOperationID),
        "developper": "Orange",
        "binaryURL": "https://internal.orangeintranet/repoFW/test3_0-R0",
        "version":"3.0r0",
        "conditions" : {
            "firmwareOrigin": "2.0.1",
            "batteryMin": "70",
            "manufacturer": "Orange"
            },
        "type": "minor",
        "timestamp": timeS

}
message = json.dumps(data)

channel.basic_publish(exchange='',routing_key='newFW',body=message)

connection.close()
```

FIGURE 6.9 – Envoi d'une notification de disponibilité de logiciel interne pour équipements Orange

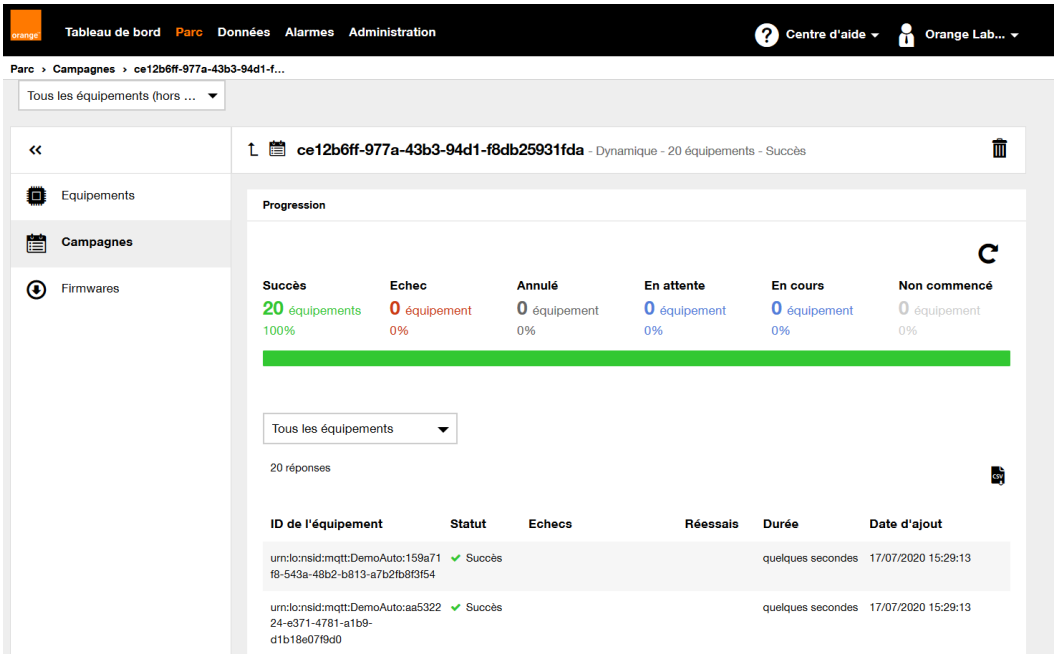

FIGURE 6.10 – Statistiques de réussite de deux campagnes de mise à jour ciblant les bons éléments du parc selon leur fabricant

du ressort de l'administrateur de configurer ce comportement, mais avec le système autonomique, cela est réalisé automatiquement au moyen des informations figurant dans la notification détaillée dans la figure 6.8 (c.a.d. Condition : manufacturer = Schneider).

Les statistiques affichées dans la figure comportent les taux de réussite, d'échecs et d'équipements en attente et sont rafraichies au fur et à mesure de l'avancement de la campagne de mise à jour. La plateforme stocke les statistiques d'exécution des campagnes dans sa base de données interne. Au moment de la rédaction de ce manuscrit, ces données ne sont pas utilisées pour autre chose que de l'historique. En intégrant les capacités du gestionnaire ExecSuivi de notre approche autonomique, il serait possible d'exploiter ces données en analysant et détectant les anomalies au fur et à mesure de l'avancement de la campagne, pour contrôler le flux de commandes envoyés aux équipements. Cela permettrait d'éviter de déployer un logiciel comprenant un bug sur toute la flotte.

6.5.1.4 Arrivée d'un nouvel équipement dans la flotte

La figure 6.11 montre les informations sur le logiciel interne de l'objet (c.a.d. Identifiant du logiciel, version et date de rafraichissement des informations) dont l'identifiant est "DeviceLogicielObsolete". Le logiciel interne installé sur cet objet est à la version "2.0.1" contrairement aux autres équipements de la flotte qui sont en '3.0-Schneider' s'ils appartiennent à Schneider et '3.0' si à Orange. Le comportement de migration logicielle visé est détaillé dans la figure 6.12 L'apparition de cet objet

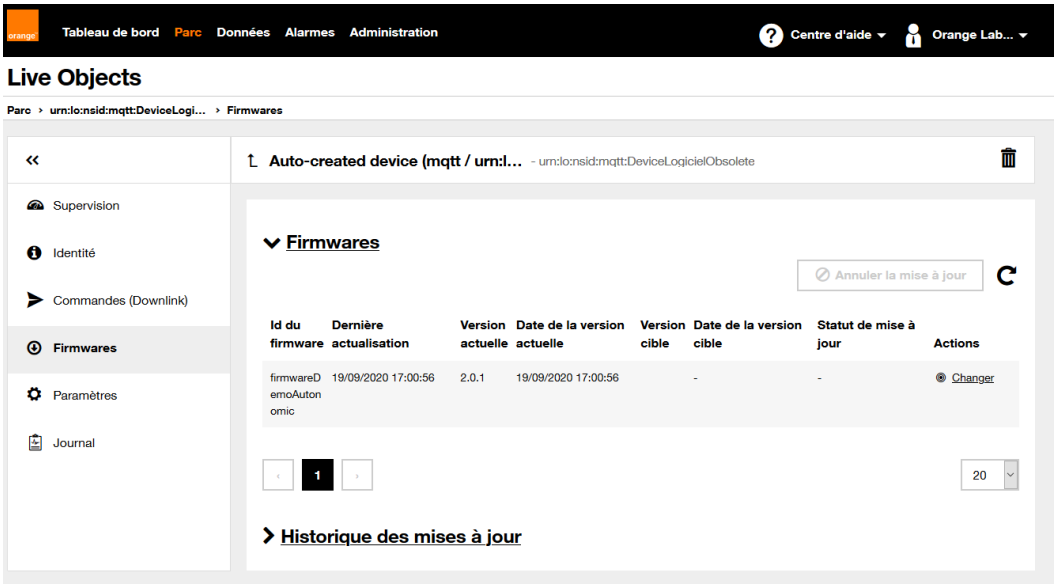

FIGURE 6.11 – État du logiciel interne de l'objet rejoignant la flotte

dans la flotte est détectée par le gestionnaire autonomique (GenerOp) qui lance automatiquement une opération de mise à jour avec le bon logiciel en fonction du fabricant de l'objet.

La figure 6.13 montre la planification campagne de mise à jour logicielle visant l'objet décrit dans la figure 6.11.

6.5.1.5 Mise à jour de cet équipement avec le logiciel en vigueur dans la flotte

La figure 6.14 expose les détails concernant la campagne de mise à jour visant à restaurer l'état 'à jour' de la flotte. Cette campagne a pour objectif d'installer le logiciel 3.0-Schneider sur l'équipement qui vient récemment de se connecter à

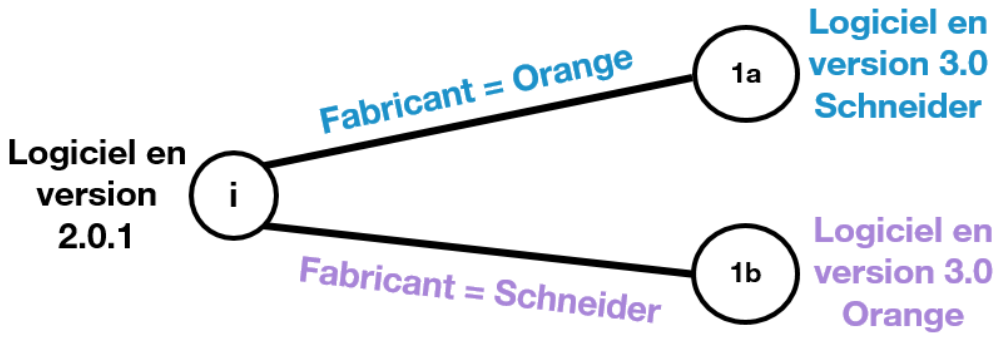

FIGURE 6.12 – Schéma de migration de version du logiciel interne visé

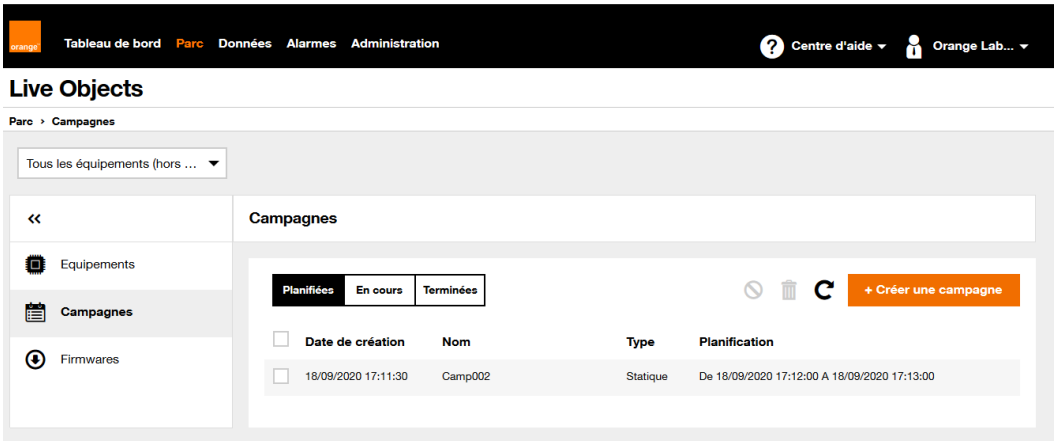

FIGURE 6.13 – Lancement de la campagne de mise à jour ciblant l'équipement obsolète

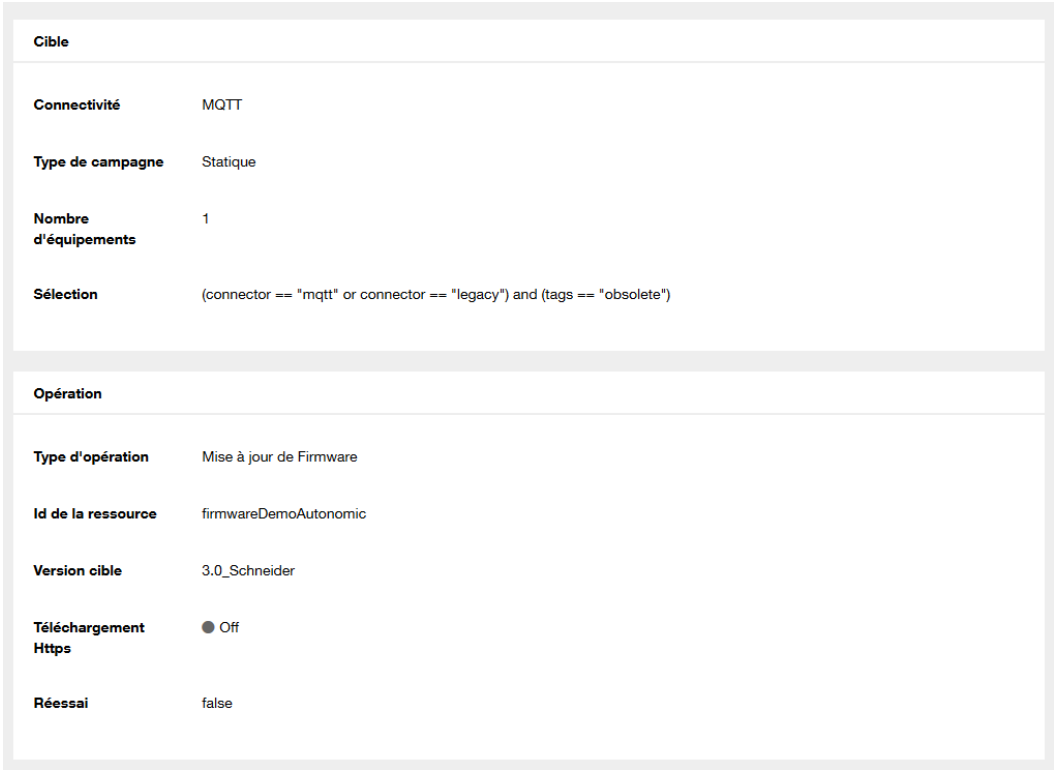

FIGURE 6.14 – Détails sur la campagne de mise à jour ciblant l'équipement obsolète

la flotte avec un logiciel interne obsolète. Cet objet possède un marqueur (tag) "obsolete" qui lui a été attribué par le gestionnaire autonomique. Ce dernier a lancé une campagne de mise à jour visant tous les équipements possédant ce marqueur.

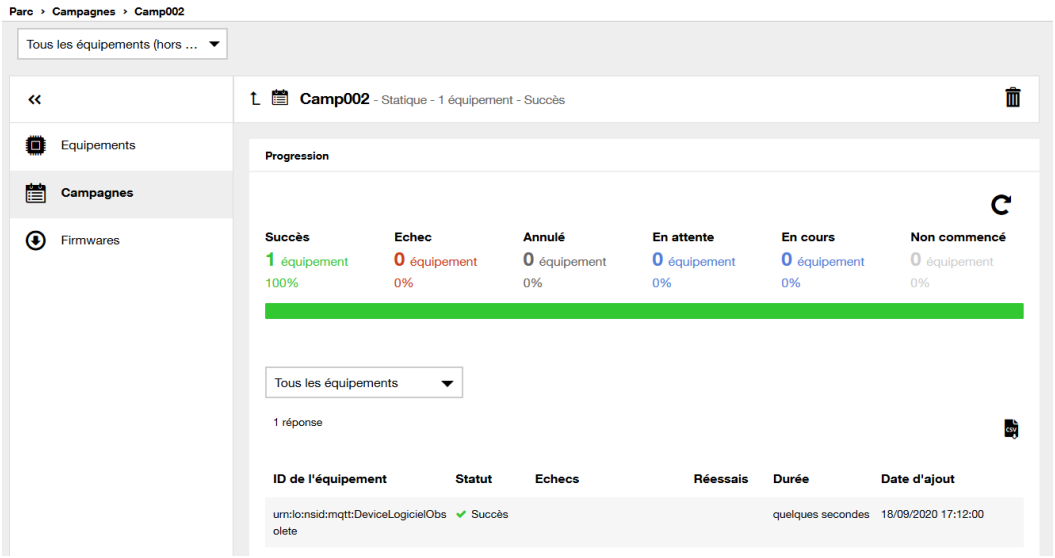

FIGURE 6.15 – Statistiques de réussite de la campagne de mise à jour ciblant l'équipement obsolète

Le résultat de la campagne "Camp002" est détaillé dans la figure 6.15. Il comporte un taux de réussite de 100% et vise un équipement : l'objet "DeviceLogicielObsolete" qui vient d'être mis à jour. La flotte étant à nouveau "à jour" le gestionnaire autonomique n'a plus besoin d'effectuer de régulation sur le parc.

6.5.2 Synthèse

Ce démonstrateur permet d'appuyer deux idées :

— La possibilité pour notre approche proposée dans la partie 2, section 4.1, de ce manuscrit de s'appliquer à des plateformes d'administration de différents niveaux de maturité : dans le cas présent, une plateforme incorporant la partie traduction et suivi (couvertes par le gestionnaire autonomique ExecSuivi dans notre approche).

— La mise en œuvre d'un gestionnaire autonomique de maintien à jour d'un parc d'objets (GenerOp) détaillé dans la section 4.1, sur une plateforme industrielle comportant des capacités d'administration d'objets de l'IdO. Cette plateforme Live Objects [29] est une plateforme industrielle relayant et analysant les données de service émanant d'objets de l'IdO. Elle comporte par ailleurs des capacités d'administration de flottes. L'objectif à terme, est d'incorporer ces travaux de thèse dans des versions futures de cette plateforme.

6.6 Limitations du protocole expérimental utilisé

Les principales limitations de ce démonstrateur sont les suivantes : l'absence de tests de passage à l'échelle et de gestion de ressources ainsi que l'absence de mise en œuvre avec plusieurs plateformes d'administration avec des capacités en administration de flottes différentes. En effet, l'IdO est caractérisé par une volumétrie importante d'objets hétérogènes, et donc administrés par des plateformes hétérogènes, qui n'auront pas systématiquement le même niveau d'administration. La volumétrie implique aussi des tests de capacité à s'adapter d'un système d'administration, au moyen de l'utilisation des gestionnaires autonomique en charge de la gestion des ressources de notre proposition d'architecture.

La multiplicité des plateformes d'IdO et des protocoles disponibles fait qu'il y a besoin d'un moyen d'interagir avec ceux-ci. C'est dans cette optique que des travaux sont en cours côté Orange afin de proposer une API standard faisant abstraction des capacités d'administration de flotte des plateformes existantes et qui sont, par conséquent, agnostiques en termes de protocoles d'administration [35].

6.7 Conclusion

Nous avons abordé dans ce chapitre le démonstrateur appuyant l'aptitude de notre approche à donner à des plateformes d'IdO utilisées en production (Live Objects), la capacité de s'adapter aux variations de l'état de la flotte en termes de composition d'équipements et de disponibilité de logiciels internes pour cette dernière.

Chapitre 7

Régulation de la taille des sous-opérations en fonction de la capacité des objets

Sommaire

Ce chapitre détaille le démonstrateur de régulation de la vitesse de décomposition d'opérations en sous-opérations sur une flotte d'équipements rencontrant des erreurs. Nous commençons par déterminer l'objectif de ce démonstrateur, les choix technologiques, l'architecture et le protocole de test que nous avons défini. Nous terminons ce chapitre par l'analyse des résultats et en énumérant les limitations du protocole expérimental utilisé.

7.1 Objectif du démonstrateur

Le gestionnaire (SuiviExec) de notre proposition a pour mission de réagir à l'arrivée de sous-opération du gestionnaire (GenerOp), en les exécutant et en assurant

le suivi en récoltant les statistiques de réussite et d'échec de manière continue. Pour rappel, les sous-opérations sont des fragments d'opérations qui sont générés par (GenerOp). La taille de ces dernières (en termes de nombres d'équipements ciblés) est déterminée par le gestionnaire de régulation de la vitesse de décomposition d'opérations en sous-opérations (RegulVitesse). Ce démonstrateur a pour but de mettre en œuvre ces trois composants, implantés par nos soins, avec une plateforme d'administration de flotte, afin de démontrer la capacité d'un système d'administration de flottes autonomique à réagir à des perturbations d'exécution : erreurs critiques et non critiques. Les erreurs *non critiques*, pour rappel, correspondent à des augmentations fortes du temps de réponse des équipements par rapport à la moyenne actuelle du système et les erreurs critiques correspondent à des équipements en dysfonctionnement.

7.2 Technologies existantes et choix

Cette section détaille les choix que nous avons réalisés en termes d'architecture et de choix de langages pour le développement. Les logiciels et briques tournent sur une station de travail ayant quatre cœurs physiques et huit unités logiques de calcul de type Intel x86-64 à architecture Kaby Lake, épaulés de 32Go de mémoire vive DDR4 à 2133MHz. Cette configuration permet de simuler quelques centaines d'objets et leur plateforme d'administration avant de montrer des limitations matérielles.

7.2.1 Plateforme d'administration

Les plateformes industrielles existantes ne permettent pas d'agir sur la partie exécution : vitesse et collecte des données de suivi. Pour palier à cette absence de solutions techniques dans l'état de l'art, nous avons exploré la piste des implantations partielles [15] du protocole LightWeight Machine2Machine LWM2M [28] développés par l'Eclipse Foundation en code source ouvert nommées Leshan. Ces implantations constituent une brique de base en Java implantant un client et un serveur d'administration à distance qui communiquent avec le standard d'administration IdO, LWM2M. Nous nous sommes basés sur la brique serveur pour simuler la fonctionnalité "mise à jour logicielle" d'une plateforme d'administration de l'IdO.

La figure 7.1 montre l'interface web du serveur d'administration. Cette interface affiche tous les objets actuellement connectés à la plateforme et leurs informations : identifiant, date d'enregistrement, et dernière date d'actualisation de l'enregistrement. La brique Leshan server n'a pas de capacité d'inventaire : si un objet n'actualise pas son enregistrement auprès du serveur périodiquement, il est dé-inventorié. Cette limitation technique n'est pas importante car durant notre protocole de test, les objets restent connectés de manière permanente.

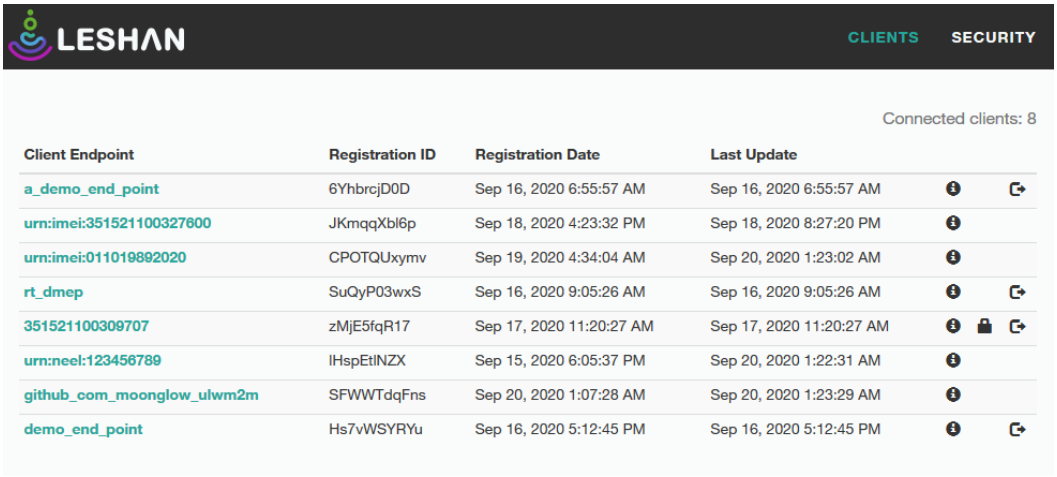

FIGURE 7.1 – Interface du serveur d'administration : Leshan Server

7.2.2 Simulateur d'équipements

Nous avons utilisé la partie client du projet Leshan [15] pour simuler des objets de l'IdO à laquelle nous avons ajouté la possibilité de choisir le nom, la marque, la version logicielle affichée par l'objet. Nous y avons aussi implanté la capacité de mise à jour du logiciel interne car cette fonction n'est pas développée dans la version disponible en ligne. Ces clients simulent des objets virtuels et sont implantés en Java comme pour la partie serveur. Nous avons aussi codé la capacité du client à avoir des temps de réponses variables en fonction du choix de l'utilisateur afin de pouvoir tester la capacité de la plateforme d'administration autonomique à s'y adapter.

La figure 7.2 montre les informations communiquées au serveur par le simulateur d'objet. Parmi ces informations nous pouvons trouver des données propres à l'objet lui-même : fabricant, modèle, numéro de série, version du logiciel, état de la batterie. Les autres sections comportent des données de services comme la localisation géographique ou la température pour un capteur de température. Ce format se rapproche de celui utilisé par Microsoft dans leur plateforme Azure [22] (section 3.1.4).

7.2.3 Gestionnaires autonomiques

Les trois gestionnaires autonomiques (GenerOp), (SuiviExec), et (RegulVitesse), mis en œuvre pour ce démonstrateur ont été développés de zéro en Python. Le choix de Python est motivé par le fait que ce langage possède une vaste collection de librairies et est nativement compatible avec le format de donnée JSON privilégié par d'autres composants du démonstrateur (base de données et bus de messagerie). Le gestionnaire GenerOp est différent de celui développé pour le démonstrateur fonctionnant avec Live Objects. Les technologies et protocoles utilisés n'étant pas les mêmes, l'implantation diffère partiellement de celle mise en œuvre dans le chapitre précédent. Ces gestionnaires communiquent via un bus de messagerie commun Rab-

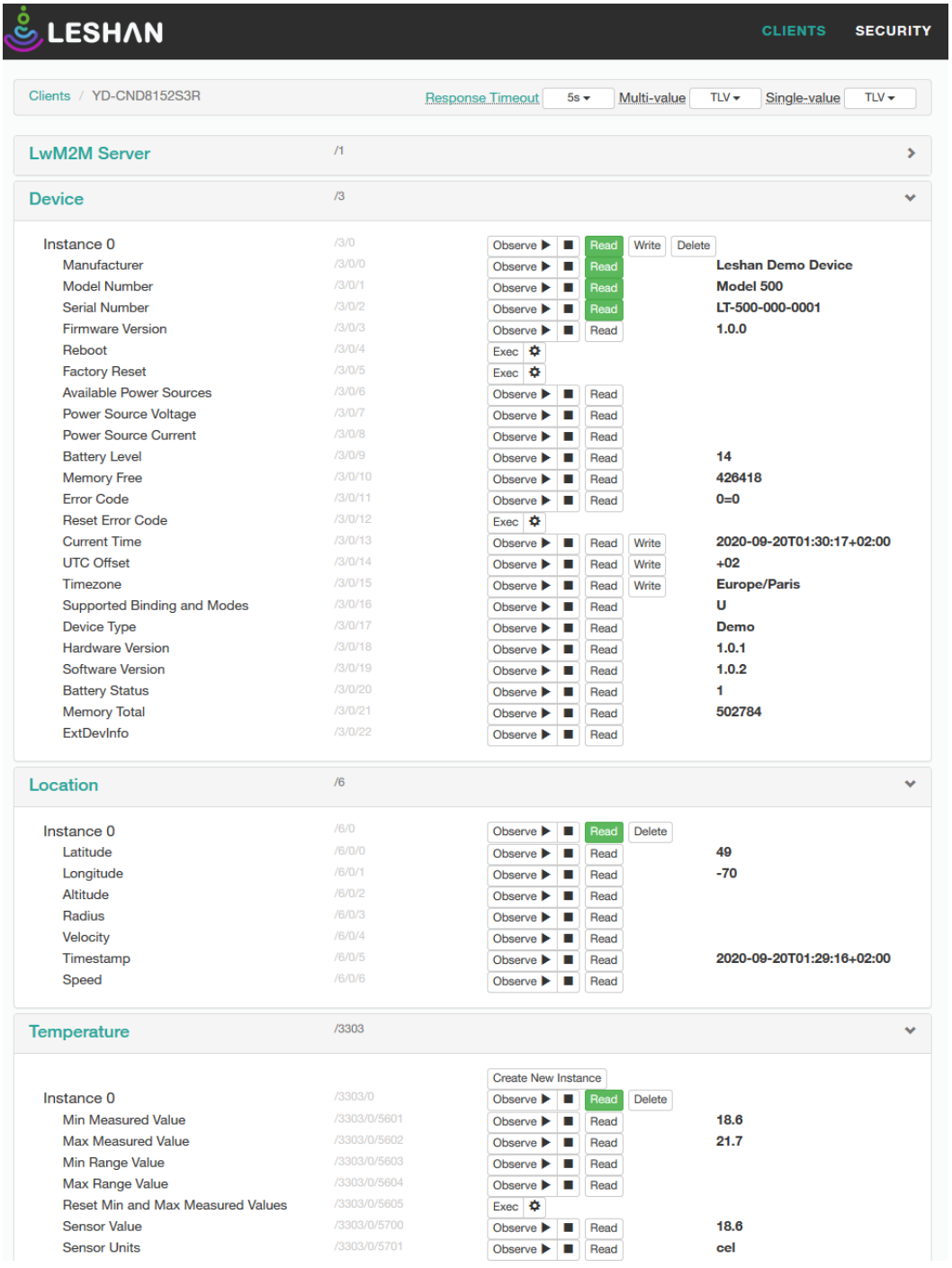

FIGURE 7.2 – Interface montrant l'état d'un objet enregistré auprès du serveur Leshan

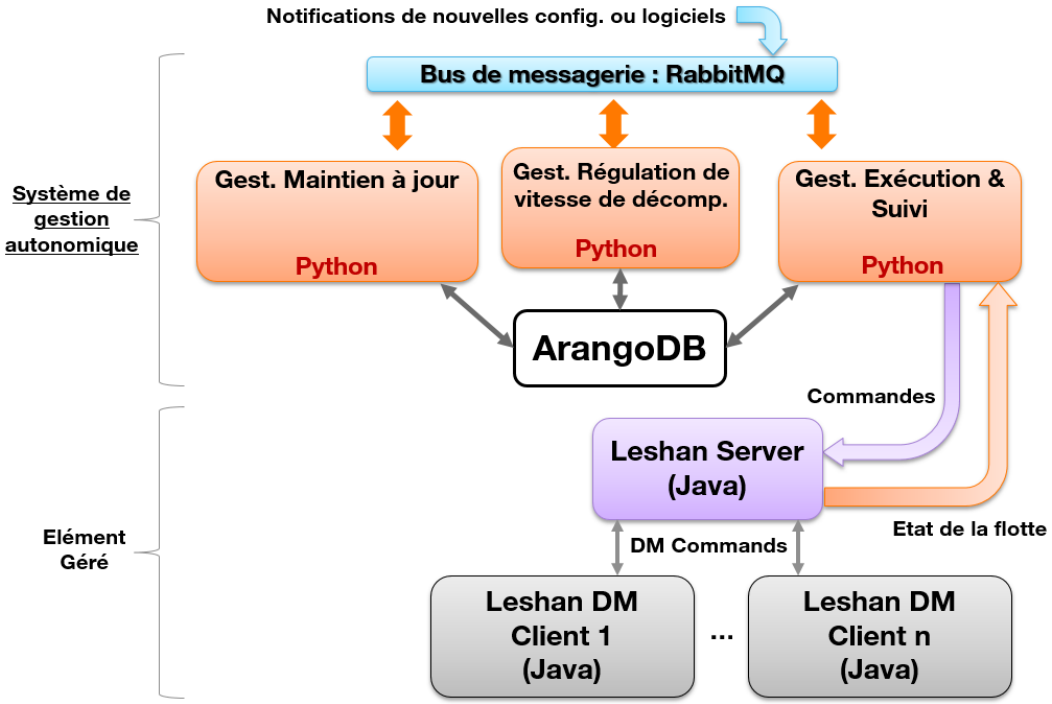

FIGURE 7.3 – Architecture technique du démonstrateur

bitMQ [21]. Les bases de connaissances des gestionnaires sont sous la forme d'une base de données multi-modèles : ArangoDB [17] qui offre des performances accrues par rapport aux bases de données relationnelles dans notre cas d'usage (tables de données indépendantes qui contiennent l'état de toutes les opérations en cours, et terminées). Le choix de ces deux technologies s'est fait de part leur documentation détaillée et leur maintenance assurée par leurs équipes de développement. Le bus RabbitMQ permet une communication via Message Queuing Telemetry Transport : MQTT [24] qui est un protocole de messagerie très utilisé dans le cadre de l'IdO et compatible avec Python.

7.3 Architecture

La figure 7.3 détaille les composants du démonstrateur. Les trois gestionnaires autonomiques fonctionnent et utilisent ArangoDB comme base de connaissance. La base de connaissance est partagée et chaque gestionnaire y accède avec un mécanisme de transaction pour éviter les incohérences. Le gestionnaire de maintien à jour y stocke les sous-opérations associées aux opérations en cours. Le gestionnaire d'exécution et suivi stocke les informations de suivi des sous-opérations en termes de réussite ou d'échec. Le gestionnaire de régulation de vitesse de décomposition d'opérations en sous-opérations n'accède pas à la base de données car il reçoit les statistiques par opération directement du gestionnaire d'exécution. Une fois la décision prise, la taille de sous-opération générée (en termes de nombre d'équipements

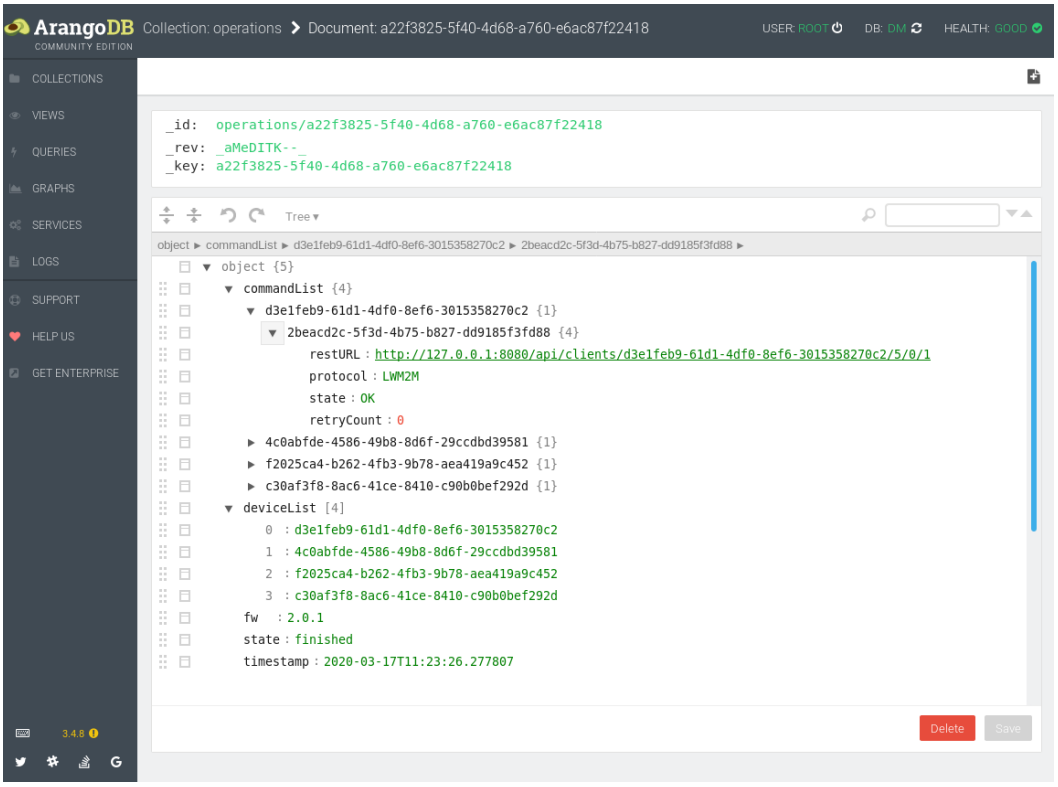

FIGURE 7.4 – Contenu de la base de données

ciblés) est envoyée directement au gestionnaire de maintien à jour (GenerOp).

La figure 7.4 montre le contenu de la base de données. Il s'agit d'une liste d'opérations qui comprend pour chaque sous-opération une liste de commandes (commandList dans la figure), chaque commande est caractérisée par son URL, son protocole d'administration (LWM2M dans la figure), le nombre de tentatives supplémentaires et son état (OK ou Not OK). Chaque commande vise un équipement. En plus de la liste de commandes, les sous-opérations contiennent la liste des équipements ciblés, la version du logiciel à installer, l'état et la date de début.

7.4 Protocole de test

Le protocole de test que nous avons défini vise à lancer une notification de disponibilité de mise à jour qui déclenche une opération de mise à jour du parc. Les simulateurs simulent un temps de réponse soit en amélioration constante ou en détérioration constante. L'objectif est de montrer la réaction du système face à ces temps de réponses des objets et par suite démontrer les capacités de scalabilité verticale d'un système autonomique d'administration d'une flotte d'équipements.

— Lancement d'un ensemble de simulateur d'objets : Leshan Client
— Envoi d'une notification (définie dans chapitre 3) de disponibilité d'un nouveau logiciel interne

— Déclenchement par le système autonomique d'une opération de mise à jour du parc.
— Observation des réactions du système en termes de taille de sous-opérations en fonction des variations du temps de réponse des objets.

Nous avons récupéré les traces d'exécution de ce protocole de test afin de déterminer la taille de sous-opération utilisée par notre système autonomique d'administration d'équipements à distance. Cette taille d'opération est tracée en fonction de l'avancement d'une opération dans les figures 7.6 et 7.7. L'axe des abscisses représente la progression totale de l'opération en minutes. L'axe des ordonnées comporte quantité d'équipements traitée par minutes. Les flèches en pointillés bleus représentent des perturbations du système autonomique. Il peut s'agir d'une amélioration du temps de réponse comme dans la figure 7.6, dans quel cas la vitesse augmente tant que les métriques sont bonnes. L'autre possibilité est la détérioration du temps de réponse qui mène à une baisse temporaire de la vitesse jusqu'à amélioration de celui-ci dans la figure 7.7.

À titre de point de référence, la vitesse utilisée aujourd'hui par Orange pour les équipements domestiques de télécommunication est représentée en orange dans les figures 7.6 et 7.7 (6.25% de la flotte totale par minute, soit à peu près 8 équipements par minutes).

7.5 Résultats

7.5.1 Exécution

La stratégie d'Orange cible toujours le même nombre d'équipements pour des raisons de prudence et afin de ne pas surcharger les équipes de surveillance de métriques. Par conséquent, la méthode actuelle se fait devancer par l'approche autonomique qui prend l'initiative d'accroître constamment la vitesse de traitement des équipements tant que les équipements ne montrent pas de signes d'anomalies. Cela mène à un temps d'exécution presque divisé de moitié pour l'approche autonomique tout en n'ayant pas surchargé les plateformes d'administration.

Dans la suite de cette sous-section nous allons comparer l'approche existante d'Orange à l'approche autonomique de cette thèse en termes d'accroissement de la vitesse d'exécution en fonction des temps de réponses des plateformes d'administration. Il s'agit de tester la capacité à s'adapter à deux types de perturbations :

— Des améliorations temporaires du temps de réponse des plateformes ;
— Des détériorations temporaires du temps de réponse des plateformes.

Dans les figures 7.6 et 7.7, nous affichons la variation de la vitesse d'exécution des opérations en nombre d'équipements mis à jour par minute de l'approche Orange (en orange) et l'approche autonomique que nous proposons (en rouge).

Dans la figure 7.6, au début, la vitesse d'exécution croit de manière linéaire stable comme dans le cas où il n y a aucune perturbation détectée sur les temps de réponses des plateformes. À la minute 3, une amélioration du temps de réponse est mesurée par le système autonomique (signe de disponibilité de nouvelles ressources). Celui-ci

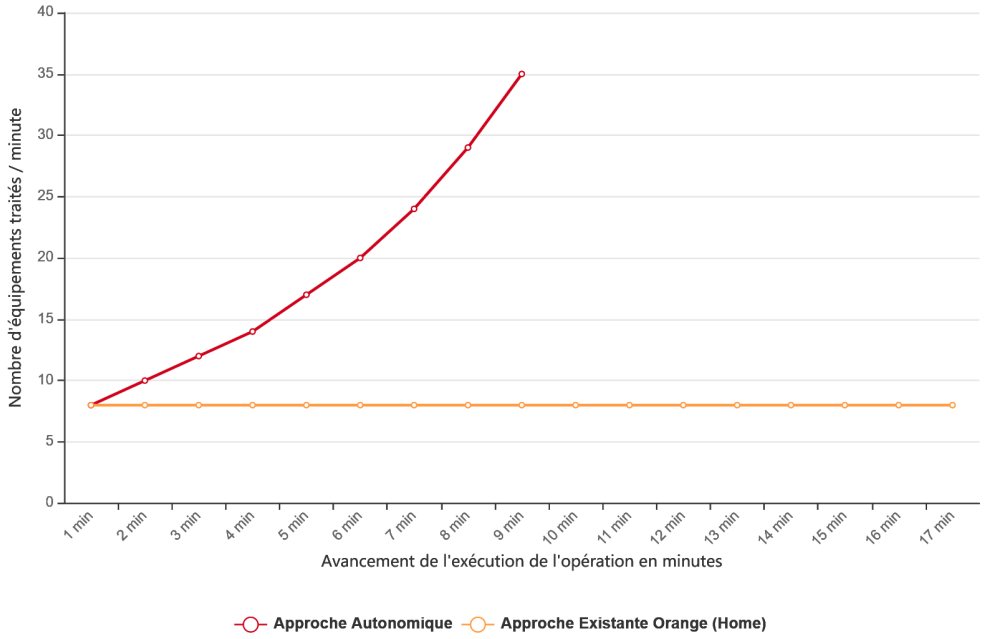

FIGURE 7.5 – Vitesse d'exécution en termes de nombres d'équipements ciblés : Approche existante Orange & Approche autonomique

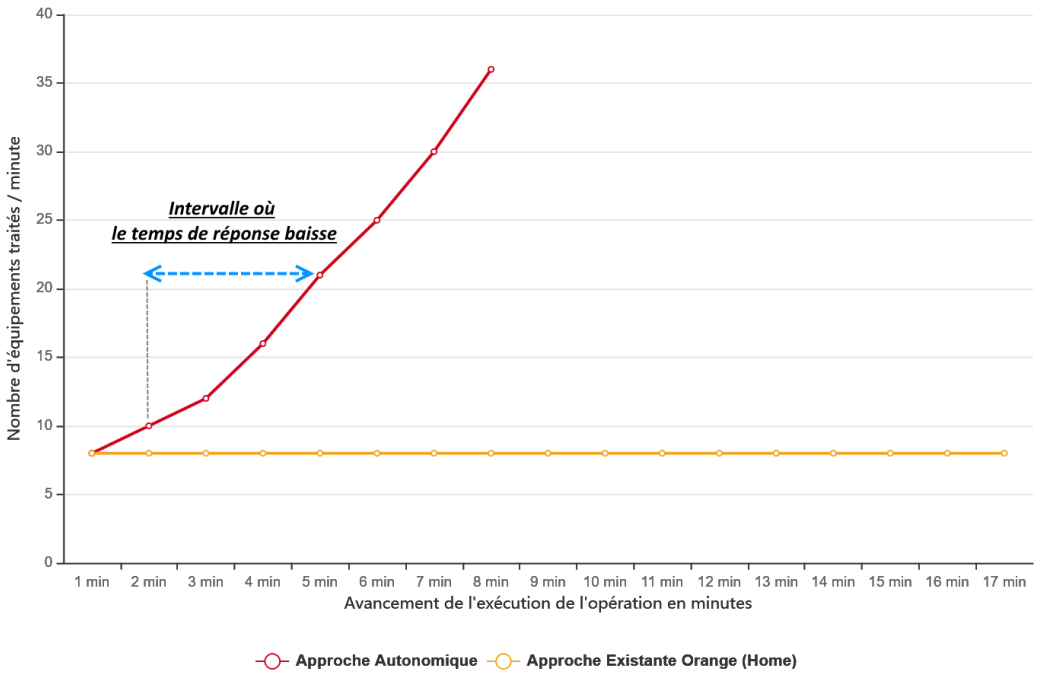

FIGURE 7.6 – Variation de la vitesse d'exécution lors d'améliorations temporaires du temps de réponse des plateformes d'administration

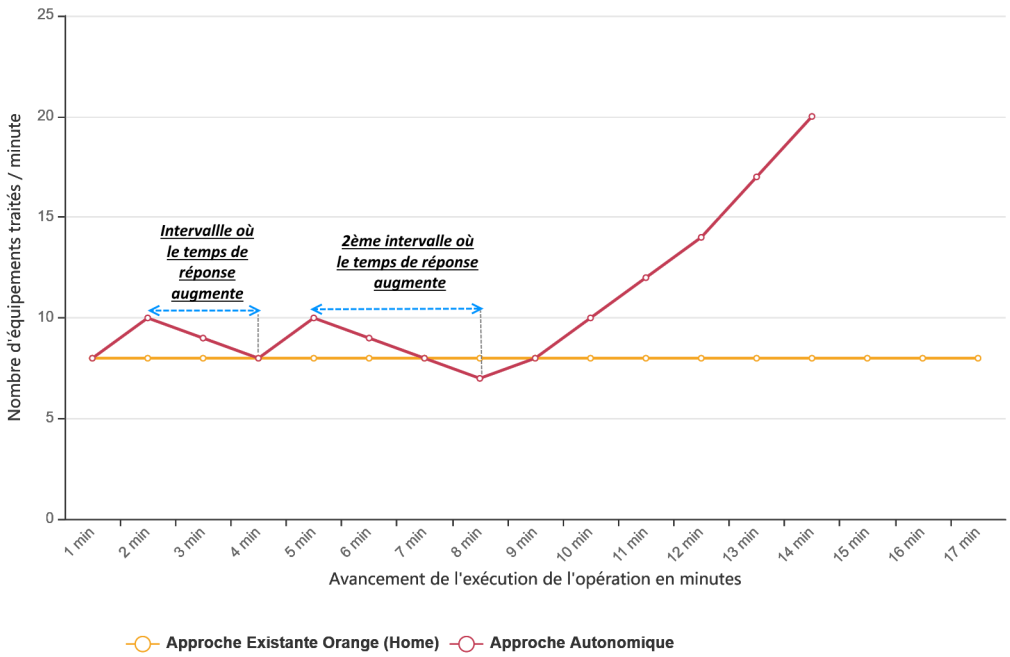

FIGURE 7.7 – Variation de la vitesse d'exécution lors de détériorations temporaires du temps de réponse des plateformes d'administration

augmente la vitesse d'exécution, ce qui se traduit par une pente plus forte dans la figure. À la minute 5, le temps de réponse des plateformes reprend sa valeur initiale, signe que les ressources ont regagné leur niveau initial. Le système autonomique rebaisse la vitesse d'accroissement à celle utilisée au début.

Dans la figure 7.7, la courbe de vitesse chute entre les minutes 2 et 4 ainsi qu'entre 5 et 8. Ces intervalles sont caractérisés par une augmentation des temps de réponses au-delà des seuils tolérés, synonyme de baisse des ressources actuellement disponibles. Afin d'éviter de surcharger les plateformes, le système autonomique a baissé la vitesse jusqu'à amélioration des temps de réponses mesurés. Par la suite, tant qu'aucune dégradation n'est détectée, la vitesse se remet à augmenter progressivement.

7.5.2 Synthèse

Le tableau 7.1 compare la solution d'administration de flottes domestiques d'Orange avec les plateformes de l'IdO et une approche utilisant une plateforme d'administration améliorée avec un système autonomique. Les critères sont le lancement d'opérations et le ciblage des équipements à mettre à jour qui se font manuellement pour l'existant et automatiquement pour l'approche autonomique. Le suivi des commandes représente la capacité d'une plateforme à récolter les traces d'exécution des commandes d'administration et de les exploiter. La solution domestique d'Orange et les plateformes IdO récoltent ce type de données mais ne les exploitent pas. L'ap-

proche autonomique se base sur ces traces pour s'adapter de manière continue et automatique aux ressources disponibles.

7.6 Limitations du protocole expérimental utilisé

Nos tests sont réalisés en local sur une station de travail ayant des capacités conséquentes par rapport à des machines de bureau classiques, mais celle-ci ne permet pas des tests visant des flottes plus importantes. Ce besoin de passage à l'échelle se manifeste vu le cadre des travaux qui visent l'administration de flottes de l'IdO qui sont caractérisées par une volumétrie de millions (voire Milliards) d'objets.
Des expérimentations sur des plateformes IdO commerciales seraient plus représentatives des bénéfices d'utiliser un système autonomique pour la gestion des objets et afin de mettre en avant la possibilité d'applications de ces travaux aux plateformes les plus utilisées dans le marché. L'obstacle principal à cela est que, pour ce faire, il faut avoir accès au code source des dites plateformes et réaliser un effort considérable de développement et intégration pour pouvoir mener ce genre d'expérimentations. De plus les spécificités en termes de protocoles et d'API rendent la tâche complexe. Cette dépendance vis-à-vis de la plateforme et du protocole est en cours d'adressage par des travaux dans l'équipe où se déroule cette thèse notamment au moyen de raisonnement sémantique et de modélisation au moyen d'une ontologie [35]. En outre, une API générique faisant abstraction des capacités d'administration de flottes des plateformes est en cours de réalisation. Ces travaux permettraient notamment d'éviter d'avoir à développer des versions différentes de notre système de gestion autonomique de plateformes en fonction de ladite plateforme automatisée.

7.7 Conclusion

Nous avons montré avec l'expérimentation détaillée dans ce chapitre la capacité d'un système autonomique d'administration d'objets de l'IdO à réagir à des variations en termes de ressources disponibles (La disponibilité des ressources est mesurée via la variation du temps de réponses des plateformes en charge des objets lors de l'exécution d'opérations de mise à jour). Le système réagit à ces perturbations en ajustant la vitesse d'exécution des opérations en conséquent.

	Plateforme Orange 'Home'	Plateformes IdO	Plateforme + Syst. Autonomique
Lancement des opérations	Manuel	Manuel	Automatique
Ciblage des équipements	Manuel	Manuel	Automatique
Suivi des commandes	✗	✗	✓
Adaptation continue	✗	✗	✓

TABLE 7.1 – Comparaison entre les plateformes existantes et une plateforme avec système autonomique

Conclusion Générale et Perspectives

Conclusion

Cette thèse CIFRE, réalisée au sein d'Orange, s'est focalisée sur l'administration à distance de flottes d'équipements de l'IdO. Ce besoin émane d'une nécessité pour les entreprises d'assurer un bon fonctionnement de leurs flottes d'équipements afin d'assurer une bonne qualité de service d'une part, et de réduire les frais de maintenance d'autre part. Dans le cadre de l'IdO, ces flottes sont hétérogènes, ont des environnements dynamiques, ont besoin d'interopérabilité, et requièrent une capacité de passage à l'échelle vu leur volumétrie. Dans ce contexte, cette thèse propose une architecture autonomique permettant l'administration d'équipements de l'IdO de manière automatique et distribuée s'appuyant sur quatre gestionnaires autonomiques coordonnés. Cette approche permet une adaptation constante au contexte et à l'environnement de ces plateformes et de leurs équipements administrés.

Rappel du contexte et de la problématique

L'administration système est le déploiement de mises à jour logicielles et configurations sur un parc d'équipements. Ces mises à jour servent de moyen d'activation ou désactivation de services, de mise en marche d'équipements, et à déployer de nouvelles fonctionnalités ou colmater des failles de sécurité. Cette administration est aujourd'hui complexe à gérer au sein d'Orange dans le cadre de flottes d'équipements domestiques de télécommunication (Passerelles Internet et décodeurs TV). Cette complexité est due à des besoins en croissance de pilotage et configuration des solutions d'administration. Ce pilotage est aujourd'hui manuel, et avec la complexité induite par les défis de l'IdO (Hétérogénéité, Dynamicité, Interopérabilité), cette approche n'est plus viable. De plus, vu la volumétrie des flottes de l'IdO, un besoin de passage à l'échelle est identifié. Il est donc nécessaire d'automatiser de manière continue le pilotage des plateformes d'administration et d'adapter le déploiement de celles-ci dans l'infrastructure pour assurer une capacité de passage à l'échelle car la gestion manuelle utilisée aujourd'hui n'est pas envisageable.

Vu les besoins ciblés, la piste de l'informatique autonomique est explorée dans cette thèse. De nombreux travaux sont proposés dans l'état de l'art et visent l'ad-

ministration d'infrastructures nuagiques, ou de dimensionnement d'applications déployés dans un centre d'hébergement, tandis que d'autres visent le pilotage de bâtiments intelligents. Aucun des travaux analysés dans l'état de l'art ne cible l'automatisation de l'administration de flottes d'équipements de l'IdO. C'est sur cette problématique que se positionne cette thèse.

Contribution de la thèse

Cette thèse a proposé une architecture autonomique permettant l'administration de flottes d'équipements de l'IdO. Le premier objectif est d'automatiser le pilotage des plateformes d'administration afin d'assurer le maintien à jour de la flotte en termes de mise à jour de configurations et de logiciels tout en effectuant le suivi de l'exécution de ces opérations d'administration. Ces deux objectifs sont réalisés par deux gestionnaires autonomiques : (GenerOp) et (ExecSuivi). Le deuxième objectif est d'ajuster de manière continue la vitesse d'exécution des opérations (en termes de nombre d'équipements ciblés) lancées par les deux gestionnaires d'automatisation, en fonction des erreurs d'exécution rencontrées, et en même temps de réguler l'utilisation des ressources de l'infrastructure. Ces objectifs sont respectivement réalisés au moyen de deux gestionnaires autonomiques : (RegulVitesse) et (Déploiement).

Les gestionnaires (GenerOp), (ExecSuivi) et (RegulVitesse) sont coordonnés et hiérarchisés afin de réaliser leurs objectifs respectifs. Le gestionnaire (Déploiement) est indépendant et hiérarchiquement [74] positionné au-dessus des autres gestionnaires. Il contrôle le déploiement des plateformes d'administration sur l'infrastructure et le nombre d'instances du gestionnaire (ExecSuivi) en fonction des ressources disponibles sur l'infrastructure à un moment donné.

Perspectives

Nous distinguons dans ce qui suit deux catégories de perspectives envisageables pour cette thèse. Celles qui concernent l'évolution de la proposition d'architecture autonomique pour l'administration de flottes d'équipements de l'IdO, puis celles visant la validation expérimentale.

Perspectives architecturales

Détection de comportements anormaux d'équipements

La première perspective architecturale envisagée pour cette thèse concerne les aspects sécurité. En effet, on ne traite pas la sécurisation du processus de mise à jour des équipements de l'IdO alors que ces derniers sont souvent cibles et biais de cyberattaques [40], [75], [39] [56]. Une piste à explorer est de surveiller le comportement des équipements à partir de données de fonctionnement non-personnelles récoltées par les plateformes d'administration (p. ex. Nombre de paquets réseaux envoyés, fréquence de connexion, taille des paquets). Ces données peuvent servir de moyen de détection d'anomalies de comportement une fois analysées et comparées à des

données de fonctionnement normal avec des techniques d'apprentissage profond. Des travaux sur ce sujet sont en cours dans l'équipe de recherche à Orange [57].

Gestion des ressources

La deuxième perspective concerne la gestion de ressources. En effet, une approche de régulation à base de contrôle continu semble être plus adéquate pour la régulation de la vitesse de décomposition des opérations dans RegulVitesse, mais qui n'a pas pu être explorée durant ce travail de thèse. En effet, le contrôle continu permet d'éviter les re-configurations fréquentes qui arrivent lorsque les données d'entrée varient beaucoup dans un laps de temps court.

De plus, du côté de la gestion de ressources d'infrastructure, il serait possible de dimensionner de manière continue les machines virtuelles qui hébergent les plateformes d'administration système en plus de les migrer dans les bons nœuds de calcul de l'infrastructure disponible comme dans notre approche. Cela permettrait de mettre à disposition des ressources supplémentaires pour utilisation par d'autres applications.

D'autres critères concernant le placement des plateformes d'administration et leurs gestionnaires, peuvent être explorées comme ceux concernant la confidentialité des données des équipements. Ces critères peuvent provenir de contraintes légales liant les propriétaires de ces équipements et leurs utilisateurs.

Perspectives expérimentales

Volumétrie des équipements

La première perspective expérimentale envisagées concernent la volumétrie des équipements utilisée et les plateformes d'administration mises en œuvre. En effet, les expérimentations ont été réalisées avec des équipements virtuels exécutés sur une station de travail. La volumétrie des objets composant l'IdO est de l'ordre des milliards, ce qui laisse envisager des flottes de plusieurs dizaines de millions d'objets. Il est nécessaire d'effectuer des expérimentations montrant la viabilité de l'approche proposée pour des volumétries de cet ordre.

Interopérabilité de plusieurs plateformes

Un des principaux défis induit par l'IdO est l'interopérabilité. Dans le cadre de ces travaux de thèse cela se matérialise par de multiples plateformes d'administration gérant plusieurs types de flottes et de services. La perspective expérimentale envisagée est de mettre en place des démonstrateurs ayant recours à plusieurs plateformes de capacités d'administration différentes. L'obstacle principal à cela est que, pour ce faire, il faut avoir accès au code source des dites plateformes et réaliser un effort considérable de développement et intégration pour adapter la sortie du système de gestion autonomique à l'entrée des plateformes d'administration d'équipements. Les spécificités en termes de protocoles et d'API rendent la tâche d'autant

plus complexe. Cette dépendance vis-à-vis de la plateforme et du protocole est en cours d'adressage par des travaux dans l'équipe où se déroule cette thèse notamment au moyen de raisonnement sémantique et de modélisation au moyen d'une ontologie [35]. En outre, une API générique faisant abstraction des capacités d'administration de flottes des plateformes est en cours de réalisation. Ces travaux permettraient notamment d'éviter d'avoir à développer des versions différentes de notre système de gestion autonomique de plateformes en fonction de ladite plateforme automatisée.

Bibliographie

[1] Microsoft Announces Availability of Systems Management Server 1.2. https://news.microsoft.com/1996/08/19/microsoft-announces-availability-of-systems-management-server-1-2/, 1996. [Online ; accessed 10-July-2020]. (Cité en pages 10 et 21.)

[2] An architectural blueprint for autonomic computing. urlQUIBUG, 2001. [Online ; accessed 30-July-2020]. (Cité en pages v, 28, 30, 31 et 33.)

[3] IoT Device Management Revenues to Climb to USD 20.5 Billion by 2023. https://www.abiresearch.com/press/iot-device-management-revenues-climb-us205-billion-2023/, 2018. [Online ; accessed 20-July-2020]. (Cité en page 4.)

[4] TR-069 CPE WAN Management Protocol - Issue : 1 Amendment 6, Version : 1.4. https://www.broadband-forum.org/technical/download/TR-069.pdf, 2018. [Online ; accessed 10-July-2020]. (Cité en pages 10, 13 et 47.)

[5] Aptitude. https://doc.ubuntu-fr.org/aptitude, 2019. [Online ; accessed 17-July-2020]. (Cité en page 12.)

[6] Global IoT market will grow to 24.1 billion devices in 2030, generating USD 1.5 trillion annual revenue. https://transformainsights.com/news/iot-market-24-billion-usd15-trillion-revenue-2030, 2019. [Online ; accessed 20-July-2020]. (Cité en pages 4 et 18.)

[7] Microsoft Store for Business and Education. https://docs.microsoft.com/en-us/microsoft-store/, 2019. [Online ; accessed 18-July-2020]. (Cité en page 12.)

[8] TR-069 - Device Management Architecture. https://www.qacafe.com/tr-069-training/session-overview/, 2019. [Online ; accessed 18-July-2020]. (Cité en page 14.)

[9] TR-069 - FAQ. https://www.incognito.com/tutorials/faq-tr-069/, 2019. [Online ; accessed 08-July-2020]. (Cité en pages 13 et 47.)

[10] About BroadBand Forum. https://www.broadband-forum.org/about-bbf, 2020. [Online ; accessed 19-July-2020]. (Cité en pages 2, 10 et 47.)

[11] Amazon Web Services IoT. https://aws.amazon.com/iot/, 2020. [En ligne ; visité le 25-Aout-2020]. (Cité en page 100.)

[12] Android Documentation. https://developer.android.com/tv, 2020. [Online ; accessed 17-July-2020]. (Cité en page 13.)

[13] DELL EMC Support. https://www.dell.com/support/home/fr/fr/frbsdt1/drivers/driversdetails?driverid=8h6hj&oscode=w12r2, 2020. [En ligne ; visité le 21-Aout-2020]. (Cité en page 1.)

[14] Derniers résultats consolidés. https://www.orange.com/fr/Investisseurs/Resultats-et-presentations/Folder/Derniers-resultats-consolides, 2020. [Online ; accessed 20-July-2020]. (Cité en pages 4 et 18.)

[15] Eclipse Leshan Project. https://github.com/eclipse/leshan/, 2020. [Online ; accessed 19-July-2020]. (Cité en pages 6, 116 et 117.)

[16] Getting Started with LWM2M. https://www.tesla.com/support/software-updates, 2020. [En ligne ; visité le 21-Aout-2020]. (Cité en page 2.)

[17] Graph and Beyond. https://www.arangodb.com/, 2020. [En ligne ; visité le 25-Aout-2020]. (Cité en page 119.)

[18] HPE Support. https://support.hpe.com/hpesc/public/docDisplay?docLocale=enUS&docId=a00097382enus, 2020. [En ligne ; visité le 21-Aout-2020]. (Cité en page 1.)

[19] IBM Watson. https://www.ibm.com/cloud/internet-of-things, 2020. [En ligne ; visité le 25-Aout-2020]. (Cité en page 100.)

[20] IPSO Smart Objects. http://openmobilealliance.org/wp/OMNA/LwM2M/LwM2MRegistry.html, 2020. [En ligne ; visité le 30-Aout-2020]. (Cité en page 47.)

[21] Messaging that just works - RabbitMQ. https://www.rabbitmq.com/, 2020. [En ligne ; visité le 25-Aout-2020]. (Cité en page 119.)

[22] Microsoft Azure IoT. https://azure.microsoft.com/en-us/overview/iot/, 2020. [En ligne ; visité le 25-Aout-2020]. (Cité en pages 100 et 117.)

[23] Microsoft IoT Hub Developper Guide, Device Twin. https://docs.microsoft.com/en-us/azure/iot-hub/iot-hub-devguide-device-twins, 2020. [En ligne ; visité le 10-Septembre-2020]. (Cité en pages v, 48 et 49.)

[24] MQTT : The Standard for IoT Messaging. https://mqtt.org/, 2020. [En ligne ; visité le 25-Aout-2020]. (Cité en page 119.)

[25] Nos Engagements RSE Orange Business Services. https://www.orange-business.com/fr/nos-engagements-rse, 2020. [En ligne ; visité le 21-Aout-2020]. (Cité en page 11.)

[26] OMA - Device Management Protocol. http://openmobilealliance.org/release/DM/V2_0-20160209-A/OMA-TS-DM_Protocol-V2_0-20160209-A.pdf, 2020. [Online ; accessed 19-July-2020]. (Cité en page 2.)

[27] Open Mobile Aliance : About. https://omaspecworks.org/about/, 2020. [Online ; accessed 19-July-2020]. (Cité en page 2.)

[28] Open-Mobile Alliance (OMA) - Lightweight Machine to Machine (LWM2M) DM Protocol. https://omaspecworks.org/what-is-oma-specworks/iot/lightweight-m2m-lwm2m/, 2020. [Online ; accessed 19-July-2020]. (Cité en pages 6, 21 et 116.)

[29] Orange LiveObjects. https://liveobjects.orange-business.com/, 2020. [Online ; accessed 19-July-2020]. (Cité en pages 5, 100, 101 et 112.)

[30] Pay TV and OTT Providers Android TV set-top-box. https://sites.google.com/view/droid-tv/pay-tv-provider, 2020. [Online ; accessed 17-July-2020]. (Cité en page 13.)

[31] "Rbc analyst says 52 million google home devices sold. https://voicebot.ai/2018/12/24/rbc-analyst-says-52-million-google-home-devices-sold-to-date/, 2020. [En ligne ; visité le 20-Aout-2020]. (Cité en page 1.)

[32] Thing in the future. https://www.thinginthefuture.com/, 2020. [En ligne ; visité le 25-Aout-2020]. (Cité en pages 20, 74 et 79.)

[33] TR-181 Issue 2 Amendment 13. https://www.broadband-forum.org/technical/download/TR-181_Issue-2_Amendment-13.pdf, 2020. [En ligne ; visité le 29-Aout-2020]. (Cité en pages v et 47.)

[34] Windows 10 1903 update history. https://support.microsoft.com/en-us/help/4498140/windows-10-update-history, 2020. [Online ; accessed 30-July-2020]. (Cité en page 20.)

[35] F. Aissaoui, S. Berlemont, M. Douet, and E. Mezghani. A semantic model toward smart iot device management. In L. Barolli, F. Amato, F. Moscato, T. Enokido, and M. Takizawa, editors, *Web, Artificial Intelligence and Network Applications*, pages 640–650, Cham, 2020. Springer International Publishing. (Cité en pages 113, 124 et 130.)

[36] A. Al-Shishtawy, J. Höglund, K. Popov, N. Parlavantzas, V. Vlassov, and P. Brand. Distributed control loop patterns for managing distributed applications. In *Second IEEE International Conference on Self-Adaptive and Self-Organizing Systems, SASO 2008, Workshops Proceedings, October 20-24, 2008, Venice, Italy*, pages 260–265. IEEE Computer Society, 2008. (Cité en pages 34, 36 et 37.)

[37] F. A. Alaba, M. Othman, I. A. T. Hashem, and F. Alotaibi. Internet of things security : A survey. *Journal of Network and Computer Applications*, 88 :10–28, Jun 2017. (Cité en page 18.)

[38] F. Alvares de Oliveira, R. Sharrock, and T. Ledoux. Synchronization of multiple autonomic control loops : Application to cloud computing. In M. Sirjani, editor, *Coordination Models and Languages*, pages 29–43, Berlin, Heidelberg, 2012. Springer Berlin Heidelberg. (Cité en pages 37, 38 et 40.)

[39] M. Antonakakis, T. April, M. Bailey, M. Bernhard, E. Bursztein, J. Cochran, Z. Durumeric, J. A. Halderman, L. Invernizzi, M. Kallitsis, et al. Understanding

the mirai botnet. In *26th {USENIX} security symposium ({USENIX} Security 17)*, pages 1093–1110, 2017. (Cité en page 128.)

[40] L. Atzori, A. Iera, and G. Morabito. The internet of things : A survey. *Computer Networks*, 54(15) :2787 – 2805, 2010. (Cité en pages 11 et 128.)

[41] A. Bashar. Autonomic scaling of cloud computing resources using bn-based prediction models. In *2013 IEEE 2nd International Conference on Cloud Networking (CloudNet)*, pages 200–204, 2013. (Cité en page 34.)

[42] R. A. Brooks. Intelligence without representation. *Artificial Intelligence*, 47(1) :139 – 159, 1991. (Cité en page 29.)

[43] B. F. Cooper and K. Schwan. Distributed stream management using utility-driven self-adaptive middleware. In *Proceedings of the Second International Conference on Automatic Computing*, ICAC '05, page 3–14. IEEE Computer Society, Jun 2005. (Cité en page 35.)

[44] R. Das, J. O. Kephart, J. Lenchner, and H. Hamann. Utility-function-driven energy-efficient cooling in data centers. In *Proceedings of the 7th international conference on Autonomic computing*, ICAC '10, page 61–70. Association for Computing Machinery, Jun 2010. (Cité en page 35.)

[45] J. Granjal, E. Monteiro, and J. Sá Silva. Security for the internet of things : A survey of existing protocols and open research issues. *IEEE Communications Surveys Tutorials*, 17(3) :1294–1312, 2015. (Cité en page 11.)

[46] S. M. K. Gueye. *Coordination Modulaire de Gestionnaires Autonomes par Controle Discret. (Modular coordination of autonomic managers by discrete control).* PhD thesis, University of Grenoble, France, 2014. (Cité en pages 37, 38, 39 et 40.)

[47] M. Z. Hasan, E. Magana, A. Clemm, L. Tucker, and S. L. D. Gudreddi. Integrated and autonomic cloud resource scaling. In *2012 IEEE Network Operations and Management Symposium*, pages 1327–1334, 2012. (Cité en page 34.)

[48] T. Kastrinogiannis, N. Tcholtchev, A. Prakash, R. Chaparadza, V. Kaldanis, H. Coskun, and S. Papavassiliou. Addressing stability in future autonomic networking. In K. Pentikousis, R. Agüero, M. García-Arranz, and S. Papavassiliou, editors, *Mobile Networks and Management*, pages 50–61, Berlin, Heidelberg, 2011. Springer Berlin Heidelberg. (Cité en pages 35 et 36.)

[49] J. Kephart and D. Chess. The vision of autonomic computing. *Computer*, 36(1) :41–50, Jan 2003. (Cité en pages 28, 29 et 30.)

[50] B. Khargharia, S. Hariri, and M. S. Yousif. Autonomic power and performance management for computing systems. In *Proceedings of the 2006 IEEE International Conference on Autonomic Computing*, ICAC '06, page 145–154. IEEE Computer Society, Jun 2006. (Cité en page 35.)

[51] D. Kusic, J. O. Kephart, J. E. Hanson, N. Kandasamy, and G. Jiang. Power and performance management of virtualized computing environments via lookahead control. In *Proceedings of the 2008 International Conference on Autonomic*

Computing, ICAC '08, page 3–12. IEEE Computer Society, Jun 2008. (Cité en page 35.)

[52] Q. Li and G. Clark. Mobile security : A look ahead. *IEEE Security Privacy*, 11(1) :78–81, Jan 2013. (Cité en page 22.)

[53] M. Litoiu, M. Shaw, G. Tamura, N. M. Villegas, H. Müller, H. Giese, R. Rouvoy, and E. Rutten. What Can Control Theory Teach Us About Assurances in Self-Adaptive Software Systems ? In R. de Lemos, D. Garlan, C. Ghezzi, and H. Giese, editors, *Software Engineering for Self-Adaptive Systems 3 : Assurances*, volume 9640 of *LNCS*. Springer, May 2017. (Cité en pages 31 et 87.)

[54] H. Liu, M. Parashar, and S. Hariri. A component-based programming model for autonomic applications. In *Autonomic Computing, International Conference on*, pages 10–17, Los Alamitos, CA, USA, may 2004. IEEE Computer Society. (Cité en page 35.)

[55] D. M. Chess, G. Pacifici, and A. Tantawi. Experience with collaborating managers : Node group manager and provisioning manager. In *Proceedings of the Second International Conference on Automatic Computing*, ICAC '05, page 39–50. IEEE Computer Society, Jun 2005. (Cité en page 35.)

[56] M. Miettinen, S. Marchal, I. Hafeez, N. Asokan, A.-R. Sadeghi, and S. Tarkoma. Iot sentinel : Automated device-type identification for security enforcement in iot. In *2017 IEEE 37th International Conference on Distributed Computing Systems (ICDCS)*, pages 2177–2184. IEEE, 2017. (Cité en page 128.)

[57] N. Najari, S. Berlemont, G. Lefebvre, S. Duffner, and C. Garcia. Network traffic modeling for iot-device re-identification. In *2020 International Conference on Omni-layer Intelligent Systems (COINS)*, pages 1–6, 2020. (Cité en page 129.)

[58] R. Nathuji, C. Isci, and E. Gorbatov. Exploiting platform heterogeneity for power efficient data centers. In *Proceedings of the Fourth International Conference on Autonomic Computing*, ICAC '07, page 5. IEEE Computer Society, Jun 2007. (Cité en page 35.)

[59] R. Nathuji and K. Schwan. Virtualpower : Coordinated power management in virtualized enterprise systems. In *Proceedings of Twenty-First ACM SIGOPS Symposium on Operating Systems Principles*, SOSP '07, page 265–278, New York, NY, USA, 2007. Association for Computing Machinery. (Cité en page 35.)

[60] M. Parashar and S. Hariri. Autonomic computing : An overview. In J.-P. Banâtre, P. Fradet, J.-L. Giavitto, and O. Michel, editors, *Unconventional Programming Paradigms*, pages 257–269, Berlin, Heidelberg, 2005. Springer Berlin Heidelberg. (Cité en page 29.)

[61] G. Privat. Wot graph as multiscale digital-twin for cyber-physical systems-of-systems. 2019. (Cité en page 20.)

[62] O. Rashid, R. Thompson, P. Coulton, and R. Edwards. A comparative study of mobile application development in symbian and j2me using example of a live football results service operating over gprs. In *IEEE International Symposium on Consumer Electronics, 2004*, pages 203–207, 2004. (Cité en page 11.)

[63] E. Rutten, N. Marchand, and D. Simon. Feedback Control as MAPE-K loop in Autonomic Computing. In *Software Engineering for Self-Adaptive Systems III. Assurances.*, volume 9640 of *Lecture Notes in Computer Science*, pages 349–373. Springer, Jan. 2018. (Cité en pages 31 et 87.)

[64] R. Schaller. Moore's law : past, present and future. *IEEE Spectrum*, 34(6) :52–59, Jun 1997. (Cité en page 21.)

[65] S. Siboni, A. Shabtai, N. O. Tippenhauer, J. Lee, and Y. Elovici. Advanced security testbed framework for wearable iot devices. *ACM Trans. Internet Technol.*, 16(4), Dec. 2016. (Cité en page 18.)

[66] A. N. Sylla. *Support intergiciel pour la conception et le déploiement adaptatifs fiables, application aux batiments intelligents.* PhD thesis, Université Grenoble Alpes. (Cité en pages 39 et 40.)

[67] A. N. Sylla, M. Louvel, E. Rutten, and G. Delaval. Design framework for reliable multiple autonomic loops in smart environments. In *2017 International Conference on Cloud and Autonomic Computing (ICCAC)*, pages 131–142, 2017. (Cité en pages 39 et 40.)

[68] N. Tcholtchev, R. Chaparadza, and A. Prakash. Addressing stability of control-loops in the context of the gana architecture : Synchronization of actions and policies. In T. Spyropoulos and K. A. Hummel, editors, *Self-Organizing Systems*, pages 262–268, Berlin, Heidelberg, 2009. Springer Berlin Heidelberg. (Cité en page 35.)

[69] G. Tesauro, W. E. Walsh, and J. O. Kephart. Utility-function-driven resource allocation in autonomic systems. In *Proceedings of the Second International Conference on Automatic Computing*, ICAC '05, page 342–343. IEEE Computer Society, Jun 2005. (Cité en page 35.)

[70] S. K. Tesfatsion, E. Wadbro, and J. Tordsson. Autonomic resource management for optimized power and performance in multi-tenant clouds. In *2016 IEEE International Conference on Autonomic Computing (ICAC)*, pages 85–94. IEEE, 2016. (Cité en page 34.)

[71] D. Vecchiato, M. Vieira, and E. Martins. The perils of android security configuration. *Computer*, 49(6) :15–21, Jun 2016. (Cité en page 22.)

[72] D. Vecchiato, M. Vieira, and E. Martins. Risk assessment of user-defined security configurations for android devices. In *2016 IEEE 27th International Symposium on Software Reliability Engineering (ISSRE)*, page 467–477, Oct 2016. (Cité en page 22.)

[73] B. Vignau, R. Khoury, and S. Hallé. 10 years of iot malware : A feature-based taxonomy. In *2019 IEEE 19th International Conference on Software Quality, Reliability and Security Companion (QRS-C)*, page 458–465, Jul 2019. (Cité en page 18.)

[74] D. Weyns, B. Schmerl, V. Grassi, S. Malek, R. Mirandola, C. Prehofer, J. Wuttke, J. Andersson, H. Giese, and K. M. Göschka. *On Patterns for De-*

centralized Control in Self-Adaptive Systems, page 76–107. Lecture Notes in Computer Science. Springer, 2013. (Cité en pages 34, 37, 61, 66 et 128.)

[75] Z. Yan, P. Zhang, and A. V. Vasilakos. A survey on trust management for internet of things. *Journal of Network and Computer Applications*, 42 :120 – 134, 2014. (Cité en pages 11 et 128.)

[76] K. Zandberg, K. Schleiser, F. Acosta, H. Tschofenig, and E. Baccelli. Secure firmware updates for constrained iot devices using open standards : A reality check. *IEEE Access*, 7 :71907–71920, 2019. (Cité en page 17.)

[77] A. Zanella, N. Bui, A. Castellani, L. Vangelista, and M. Zorzi. Internet of things for smart cities. *IEEE Internet of Things Journal*, 1(1) :22–32, 2014. (Cité en page 11.)

Publications liées à la thèse

— N. Ayeb et al. "Towards an Autonomic and Distributed Device Management for the Internet of Things," 2019 IEEE 4th International Workshops on Foundations and Applications of Self* Systems (FAS*W), Umea, Sweden, 2019, pp. 246-248, doi : 10.1109/FAS-W.2019.00065.

— N. Ayeb et al. "Contrôle d'une campagne de déploiement d'une mise à jour logicielle sur un parc de périphériques", FR 1910794, demande de brevet déposée le 30 septembre 2019 auprès de l'INPI.

— N. Ayeb et al. "Coordinated autonomic loops for target identification, load and error-aware Device Management for the IoT", FedCSIS 2020 - 15th Federated Conference on Computer Science and Information Systems, Sep. 2020, Sofia, Bulgaria. pp.1-10.

— N. Ayeb et al. "Autonomic device management for keeping an IoT fleet up to date" , 24th Conference on Innovation in Clouds, Internet and Networks, March 2021, Paris, France, (soumis).

Résumé

Avec l'avènement de l'internet des objets (IdO) qui se base sur des objets hétérogènes, dynamiques et de haute volumétries, des besoins en administration à distance sont requis pour un bon fonctionnement de ces objets. Il s'agit par exemple de mise à jour de logiciels, de configurations, de résolution de problèmes à distance ainsi que de récolte de données de fonctionnement. Ces opérations d'administration permettent d'assurer une bonne qualité de service et d'expérience pour les utilisateurs. Elles permettent en outre, le déploiement de nouvelles fonctionnalités, de correctifs logiciels, et de mise à jour de sécurité.

Les plateformes industrielles existantes d'administration montrent leurs limites avec des parcs formés d'objets statiques, en termes de capacités et d'environnements, comme les passerelles internet domestiques et décodeurs de flux TV. Ces plateformes sont opérées manuellement par des équipes d'administrateurs systèmes et requièrent une expertise conséquente.

Concernant les flottes de l'IdO, l'hétérogénéité se traduit en un ensemble d'équipements ayant des capacités différentes de calcul et de connectivité réseau. La dynamicité concerne les environnements de ces équipements qui varient en termes de services en cours d'exécution, de qualité du lien réseau, de capacité restante de calcul. La volumétrie des objets de d'IdO impose un besoin de passage à l'échelle afin de gérer des milliards d'équipements contrairement aux flottes composée de millions d'équipements aujourd'hui.

Par suite, l'administration de flottes de l'IdO requiert une adaptation constante de ces opérations en termes de nature, de vitesse et de cible. Les approches manuelles existantes ne permettent pas de réaliser ces opérations en prenant en compte les spécificités de l'IdO.

Afin d'adresser cette problématique, ce travail de thèse industrielle chez Orange, vise à appliquer le paradigme de l'informatique autonomique au pilotage et la distribution des plateformes d'administration. L'objectif est d'assurer que les besoins en administration des flottes de l'IdO soient automatiquement réalisés, et ce, avec une consommation optimale de ressources de calcul et de réseau, ainsi qu'avec un nombre le moins élevé possible, d'erreurs d'exécution.

Notre proposition s'appuie sur quatre boucles autonomiques coordonnées. Deux d'entre elles sont responsables de l'automatisation du maintien à jour de la flotte d'équipements tandis que les deux autres sont chargées de la régulation de l'utilisation des ressources assurant ainsi un passage à l'échelle vertical et horizontal.

Notre proposition est validée au travers de deux prototypes. Le premier sert de démonstrateur de l'utilisabilité de notre approche pour le pilotage d'une plateforme industrielle d'administration de l'IdO (Live Objects d'Orange) qui est utilisée en production. Le deuxième démontre les capacités de passage à l'échelle vertical de notre proposition. Il s'appuie sur des technologies à code source ouverts. Les résultats sont encourageants par rapport aux approches existantes (p. ex. Vitesse d'exécution multipliée par deux sans augmentation du taux d'équipements en dysfonctionnement).

Abstract

With the expansion of Internet of Things (IoT) that relies on heterogeneous ; dynamic ; and massively deployed devices ; Device Management (DM), which consists of firmware update, configuration, troubleshooting and tracking, is required for proper quality of service and user experience, deployment of new functions, bug fixes and distribution of security patches.

Existing Home and IoT industrial DM platforms are already showing their limits with a few static home and IoT devices (e.g., routers, TV Decoders). Currently, these platforms are mainly manually operated by experts such as system administrators, and require extensive knowledge and skills. Heterogeneity implies that devices have diverse compute and network capabilities. Dynamicity translates to variation of devices environments (e.g., network quality, running services, nearby devices). The massive aspect is reflected in fleets composed of billions of devices as opposed to millions currently.

Therefore, IoT device administration requires launching administration operations that assure the well-functioning of device fleets. These operations are to be adapted in terms of nature, speed, target, accordingly to devices current service requirements, computing capabilities and network conditions. Existing manually operated approaches cannot be applied on these massive and diverse devices forming the IoT.

To tackle these issues, our work in an industrial research context, at Orange Labs, proposes applying autonomic computing to platform operation and distribution. It aims to ensure that administration requirements of a device fleet are automatically fulfilled using the optimal amount of resources and with the least amount of execution errors.

Specifically, our contribution relies on four coordinated autonomic loops. The first two loops are responsible for handling fleet variation and update operations dispatching, while the remaining two others focus on vertical and horizontal scalability. Our approach allows automatic administration platform operation, more accurate and faster error diagnosis, vertical and horizontal scaling along with simpler IoT DM platform administration.

For experimental validation, we developed two prototypes : one that demonstrates the usability of our approach with Orange's industrial IoT platform for its piloting, while the other one demonstrates vertical scalability using extended open-source remote administration software. Our prototypes show encouraging results, such as two times faster firmware upgrade operation execution speed, compared to existing legacy telecommunication operator approaches.