Université Fédérale

Toulouse Midi-Pyrénées

THÈSE

En vue de l'obtention du

DOCTORAT DE L'UNIVERSITÉ DE TOULOUSE

Délivré par : *l'Université Toulouse 3 Paul Sabatier (UT3 Paul Sabatier)*

Présentée et soutenue le *03/02/2021* **par :**

Tanissia DJEMAI

Placement optimisé de services dans les architectures Fog Computing et Internet of Things sous contraintes d'énergie, de QoS et de mobilité

JURY

Frédéric Le Mouël	INSA Lyon	Membre du Jury
Philippe Roose	IUT Bayonne	Membre du Jury
Didier Donsez	Université Grenoble 1	Membre du Jury
Hélène Coullon	IMT Atlantique	Membre du Jury
Marie-Pierre Gleizes	Université Toulouse 3	Membre du Jury
Patricia Stolf	Université Toulouse 2	Membre du Jury
Thierry Monteil	INSA Toulouse	Membre du Jury
Jean-Marc pierson	Université Toulouse 3	Membre du Jury

École doctorale et spécialité :
MITT : Domaine STIC : Réseaux, Télécoms, Systèmes et Architecture

Unité de Recherche :
Institut de Recherche en Informatique de Toulouse IRIT (UMR 5505)
Laboratoire d'Analyse et d'Architécture des Systémes LAAS-CNRS (UPR 8001)

Directeur(s) de Thèse :
Patricia STOLF, *Thierry MONTEIL* et *Jean-Marc PIERSON*

Rapporteurs :
Frédéric Le Mouël et *Philippe Roose*

Remerciements

Je tiens à adresser mes premiers remerciements à mes trois directeurs de thèse, Patricia Stolf, Thierry Monteil et Jean-Marc Pierson qui ont toujours su me guider et me conseiller pour aller jusqu'au bout de ce doctorat. Leur soutien et leur bienveillance ont été pour moi indispensables. Je les remercie d'avoir toujours été disponibles pour moi, de m'avoir aidée à améliorer mes compétences scientifiques et de m'avoir donnée la chance de pouvoir faire partie du monde de la recherche pendant ces trois années. Merci infiniment. Merci aussi pour votre bonne humeur à toute épreuve et pour les blagues que vous faisiez entre vous pendant les réunions qui me faisaient bien rire.

Je remercie également les différents membres du jury qui ont donné de leur temps et de leur énergie pour examiner mon travail.

Je tiens par ailleurs à remercier Marie-Pierre Gleizes et tous les membres du projet neOCampus de l'université de Toulouse III. Grâce à neOCampus j'ai eu l'occasion d'organiser et de participer à différents événements scientifiques et industriels qui m'ont permis de côtoyer des gens de différentes disciplines (et aussi de manger les bons petits gâteaux de Marie-Pierre).

Un grand merci à tous mes collègues et amis déjà docteurs ou en voie de le devenir, des équipes SEPAI à l'IRIT et SARA au LAAS. Merci à Benoit, Walid, Josué et karima d'avoir toujours été là pour moi en toute circonstance. Merci à Fadel, Nour, Imane, Nga, Ioana, Célement, Raoua, Ezekiel, Laurent et David pour votre écoute et vos conseils. Je remercie également Léo, Gustavo, Chao Peng et Morgan pour leur extrême gentillesse. Ces trois années passées avec vous les amis ont été extrêmement enrichissantes tant sur le plan professionnel que personnel. Je vous remercie pour tous ces bons moments où on a pu grandir et évoluer ensemble.

Je tiens à remercier chaleureusement mon ancien maître de stage (mon ancien maître Yoda) au LAAS, Pascal Berthou qui m'a donné l'envie de faire une thèse et qui a toujours été de bons conseils, d'une extrême gentillesse et d'une infinie sagesse. Tu es pour moi un vrai modèle à suivre.

Je remercie également tous les chercheurs des équipes SEPIA et SARA, avec qui j'ai pu discuter de mes travaux lors des différentes réunions d'équipes. Je m'estime extrêmement chanceuse d'avoir pu côtoyer tant d'esprits passionnés, brillants et humbles au sein des deux laboratoires IRIT et LAAS. Je remercie également tout le personnel des deux laboratoires qui font un travail incroyable.

Enfin, mon infinie reconnaissance et remerciement vont à maman qui est à la fois ma meilleure amie, ma psy (sûrement aussi devenue experte en Fog Computing maintenant) et qui a toujours su trouver les mots justes pour me motiver. Merci à mon papa et à mes soeurs d'avoir supporté mes crises existentielles ces trois dernières années et de m'avoir aidé à relativiser en toute circonstance.

"Médite fréquemment la rapidité avec laquelle passent et se dissipent les êtres et les événements. La substance est, en effet, comme un fleuve, en perpétuel écoulement ; les forces sont soumises à de continuelles transformations, et les causes formelles à des milliers de modifications. Presque rien n'est stable, et voici, tout près, le gouffre infini du passé et de l'avenir, où tout s'évanouit. Comment ne serait-il pas fou, celui qui s'enfle d'orgueil parmi ce tourbillon, se tourmente ou se plaint, comme si quelque chose, pendant quelque temps et même longtemps, pouvait le troubler ?"

-Marc Aurèl-

Abstract

The advent of the Internet of Things (IoT) raises various issues, both in terms of the development and deployment of IoT applications in computing infrastructures. Cloud Computing is the most widespread computing infrastructure today. It is based on data centers that communicate with each other and with users via monolithic, inflexible network equipments. The importance of revising this schema has been highlighted in order to meet the challenges of an IoT environment that is heterogeneous, mobile and generates a large amount of data that requires rapid processing. The classic IoT model, in which IoT objects send information via their gateways to the Cloud, which then provides services to the applications, finds extensions in the Fog or Edge approach, which enables services to be brought closer to users by relying on intermediate computing and communication equipments between users and data centers. The Fog Computing architecture allows exploiting the computing and storage capacities of the network infrastructure, in addition to that of the Cloud, for the deployment of IoT services and thus extending and bringing services closer to users. However, network equipments are heterogeneous and with low computing capacity, they cover a large geographical area and must cope with the mobility of IoT users. All this adds complexity to the problem of service placement and scheduling in order to optimize various parameters such as energy consumption, different costs related to placement and improving the applications quality of service requirements. The objective of our thesis is to propose IoT service placement strategies in a Fog infrastructure while taking into account the dynamic nature of the environment brought by user mobility, the energy cost of computing infrastructures and the QoS requirements of deployed applications.

Keywords : Fog Computing, Internet of Things, Energy, QoS, Mobility.

Résumé

L'avènement de l'Internet of Things (IoT) soulève diverses problématiques, tant au niveau du développement que du déploiement des applications IoT dans les infrastructures de calculs. Par ailleurs, l'infrastructure de calculs la plus répandue de nos jours est celle du Cloud Computing reposant sur des centres de données centralisés communicants entre eux et avec les utilisateurs par le biais d'équipements réseau monolithiques et peu flexibles. L'importance de revoir ce schéma a été mise en avant pour faire face aux défis d'un environnement de l'IoT, hétérogène, mobile et générant une grande quantité de données qui exigent des temps de traitements quasi-instantanés. Le modèle classique de l'IoT amenant les objets IoT à envoyer des informations via leurs passerelles au Cloud, qui ensuite fournit les services aux applications trouve des extensions dans l'approche Fog ou Edge permettant de rapprocher les services des usagers en s'appuyant notamment sur des équipements de calcul et de communication intermédiaires entre les utilisateurs et les centres de données. L'architecture Fog Computing est considérée comme étant l'un des schémas permettant d'exploiter la capacité de calcul et de stockage de l'infrastructure réseau en plus de celle du Cloud pour le déploiement des services IoT au plus près de leurs objets. Cependant, les équipements réseau sont hétérogènes et à faible capacité de calcul, ils couvrent une large zone géographique et doivent faire face à la mobilité des utilisateurs IoT. Tout ceci complexifie le problème du placement et de l'ordonnancement des services dans le but d'optimiser divers paramètres tels que l'énergie consommée, les différents coûts liés au placement et l'amélioration de la qualité de service des applications. L'objectif de notre thèse est de proposer des stratégies de placement de services IoT dans une architecture Fog tout en prenant en compte la nature dynamique de l'environnement apportée par la mobilité des objets IoT, le coût énergétique des infrastructures de calcul et les exigences en qualité de service des applications déployées.

Mots clés : Fog Computing, Internet des Objets, Énergie, QoS, Mobilité.

Table des matières

Introduction **1**

1 Concepts préliminaires, bibliographie et synthèse **9**
1.1 Introduction 10
1.2 Paradigme de l'Internet des objets 11
1.2.1 Définition 11
1.2.2 Les objets IoT 12
1.2.3 Les Applications IoT 20
1.3 Paradigme du Fog Computing 24
1.3.1 Définition 24
1.3.2 Architecture et éléments d'une infrastructure Fog Computing 25
1.3.3 Caractéristiques du Fog Computing 27
1.3.4 Concepts connexes au Fog : Edge et Mist computing . 28
1.4 Les Avantages et les défis du Fog Computing pour supporter les applications IoT 30
1.4.1 Un environnement Fog-IoT 31
1.5 Gestion des applications IoT dans les environnements Fog-IoT 31
1.5.1 Les flux de services (workflow) 32
1.5.2 Objectifs et finalités d'un ordonnancement de tâches . 33
1.5.3 Le placement de services vu comme un problème d'optimisation combinatoire 34
1.6 La consommation énergétique dans les environnements Fog-IoT 40
1.6.1 La consommation énergétique du secteur des technologies de l'information et des communications (ICT) . . . 40
1.6.2 Calcul et estimation de la consommation énergétique dans les environnement ICT 43
1.6.3 La minimisation de la consommation énergétique dans des environnements Fog-IoT 45
1.7 La qualité de services dans les environnements Fog-IoT 49
1.8 La mobilité dans les environnements Fog-IoT 51
1.9 Synthèse 53

2 Outils de simulation et bancs d'essai pour les environnements Fog-IoT **55**
2.1 Introduction 55
2.2 Source de données pour les applications IoT 56
2.2.1 Les critères à définir pour les applications IoT 56

2.2.2 Applications IoT de la littérature 57
2.2.3 Applications d'imagerie interactives et/ou collaboratives 57
2.2.4 Applications Santé 58
2.2.5 Applications d'environnement ambiant (smart home, smart building) 59
2.2.6 Applications de l'industrie 4.0 59
2.3 Outils et données pour les infrastructures Fog-IoT 60
2.3.1 Principales caractéristiques des simulations des environnements Fog-IoT 60
2.3.2 Topologies des infrastructures Réseau pour le Fog . . . 63
2.3.3 Taxonomie des simulateurs Fog et IoT existants 66
2.3.4 Plateforme et bancs d'essai pour le Fog Computing et l'IoT 69
2.4 MyMobileIFogSim 70
2.4.1 Architecture d'iFogSim 70
2.4.2 Contrainte de placement et limite réseau d'iFogSim . . 71
2.4.3 La migration et la mobilité 73
2.5 Synthèse 74

3 Placement de services en environnement statique 75
3.1 Introduction 75
3.2 Modèle des applications IoT 76
3.2.1 Représentation de l'application 76
3.2.2 Délai de réponse de l'application 78
3.3 Modèle de l'infrastructure Fog 80
3.3.1 Les noeuds de l'infrastructure 82
3.3.2 Les liens réseau 83
3.4 Formulation du problème de placement de services 85
3.4.1 La consommation énergétique 85
3.4.2 Violations des délais 86
3.4.3 Fonction objectif 87
3.5 Approches de résolution 87
3.5.1 Optimisation Discrète par essaims particulaires (DPSO) 87
3.5.2 Algorithme Génétique (GA) 98
3.6 Résultats 100
3.6.1 Test du DPSO et des heuristiques pour le placement de services 101
3.6.2 Test du DPSO et des méta-heuristiques pour le placement statique 107
3.7 Bilan des expérimentations et amélioration des approches de placement 115
3.8 Synthèse 118

4 Placement de services en environnement dynamique 121
4.1 Introduction 122
4.2 Modèles de mobilité : Définition, Taxonomie et Analyse de l'existant 123
4.3 Extension du modèle de l'environnement IoT-Fog statique . . 128
4.3.1 Zone de couverture des noeuds Fog 129
4.3.2 Noeuds IoT mobiles 129
4.4 Modèle de mobilité History-Based Transition Matrix (HTM) . 130
4.4.1 Principe et généralités du modèle 130
4.4.2 Méthodologie 130
4.4.3 Créneaux horaires et zones de mobilité 131
4.4.4 Création des matrices de transitions entre les zones . . 132
4.4.5 Réajustement ou réduction des matrices de transitions 133
4.4.6 Avantages et inconvénients de HTM 134
4.4.7 Améliorations du modèle HTM (Méta-modèle de mobilité et contrôle dynamique de l'environnement mobile) 134
4.5 Migration et redéploiement de services 136
4.5.1 Estimation du temps de migration d'un service 136
4.5.2 Estimation de la consommation énergétique de la migration d'un service 137
4.5.3 Fonction objectif 138
4.6 Approches de résolutions 141
4.6.1 Placement sans réévaluation 141
4.6.2 Placement avec réévaluation 143
4.7 Résultats 146
4.7.1 Tests de Validation du modèle de mobilité 146
4.7.2 Tests de Validation des équations de migration 149
4.7.3 Tests de Validation des méthodes de placements 151
4.8 Synthèse 162

Conclusion et Perspectives 165

Bibliographie 171

Table des figures

1 Nombre de publications avec les termes "Fog" et "IoT" entre 2015 et 2019. 3
2 Illustration de la problématique de placement dans le Fog. . . 5

1.1 Taxonomie des objets connectées [Dorsemaine 2015] 12
1.2 Taxonomie pour définir l'architecture d'une application [Mahmud 2020] 19
1.3 Intervalles des besoins en latence par groupes d'applications [Ricart 2020]. 24
1.4 Illustration de l'évolution de l'architecture Cloud vers le Fog. 25
1.5 Vue globale de l'architecture hiérarchique 3 tiers du Fog Computing. 26
1.6 Aspects de management d'application [Mahmud 2020] 31
1.7 Ordonnancement de tâches informatique [Dorsemaine 2015] . . 37
1.8 Catégories d'équipements ICT 44
1.9 Intervenir sur l'efficacité énergétique des noeuds de calcul [Orgerie 2014] 47

2.1 Topologies des applications temps réel et vidéo de surveillance collaborative et jeu de réalité augmentée de la littérature. . . . 58
2.2 Topologies des applications healthcare de la littérature. 59
2.3 Topologie d'application décomposée en 3 services pour un environnement ambiant [Brogi 2018]. 59
2.4 Topologie d'application de l'industrie 4.0 [Mann 2019a] 60
2.5 Architecture haut niveau d'iFogSim [Gupta 2016] 71
2.6 Principaux modules de MyMobileIFogSim 72
2.7 Diagramme UML des principales classes ajoutées à MyiFogSim (en jaune). 72

3.1 Représentation de la topologie d'une application mettant en évidence les chemins critiques et les différentes catégories de services définies dans cette section. 78
3.2 Représentation de l'infrastructure Fog 3-tiers. 82
3.3 Vecteur position d'une particule dans le DPSO où un élément du vecteur indexé par le service donne le numéro de la machine physique sur laquelle il sera placé. 91
3.4 Matrice vélocité d'une particule dans le DPSO où chaque ligne donne les chances de placement du service concerné par rapport à chaque machine physique. 92
3.5 Topologie du voisinage dans l'essaim 94

3.6 Estimation du bruit autour d'une particule selon [Deb 2002]. . 96
3.7 Illustration des processus de croisement et de mutation 99
3.8 Illustration de la méthodologie des expérimentations. 100
3.9 Topologie des applications à flux séquentiel unidirectionnel (SUD) : le premier service reçoit les données des capteurs et le dernier communique avec les actionneurs. 101
3.10 Topologie des applications Master-Slave (MS) : le service maître est le seul service qui communique avec la source et envoie ensuite des instructions aux services esclaves. Après avoir reçu la réponse de tous les services esclaves, le service maître envoie les instructions aux actionneurs. 101
3.11 Topologie Fog des expérimentations, composée d'une couche Cloud, couche de passerelles Fog et une couche IoT où pour chaque passerelle Fog 4 objets IoT y sont connectés. 103
3.12 Consommation énergétique et nombre de violations de délai par nombre d'applications à placer et pour chaque méthode de placement 105
3.13 Charge de services par couche et pour chaque méthode. . . . 106
3.14 Impact des topologies d'application SUD et MS sur les performances des méthodes DPSO, MDPSO et GA. 107
3.16 Facteur de dispersion des services d'une même application pour les méthodes DPSO, MDPSO et GA 109
3.15 Impact des topologies d'application SUD et MS sur l'utilisation de l'infrastructure par les méthodes DPSO, MDPSO et GA. . 109
3.17 Les valeurs moyennes des métriques de la consommation énergétique, violations de délai et temps d'exécution de chaque méthode par nombre d'applications placées. 110
3.18 Usage des noeuds de chaque couche pour le placement des applications par nombre d'applications placées 111
3.19 Utilisation de couches Fog pour chaque méthode en fonction du type de réseau d'accès. 112
3.20 Illustrations par histogrammes du nombre de comparaisons gagnantes pour chaque métrique entre les méthodes DPSO, GA et MDPSO, par classe d'applications : MC, RT, ST et BE. . . 113
3.21 Radar Graphe de la moyenne de toutes les expérimentations . 113
3.22 Diagramme algorithme de placement. 116
3.23 Dendrogramme des choix des méthodes initiales et des métaheuristiques 118

4.1 Taxonomie modèles de mobilité 124
4.2 Schématisation des matrices de transitions pour un objet . . . 133
4.3 Système de placement en ligne de applications IoT en environnement dynamique 143

4.4 Améliorations du système de placement en ligne des applications IoT en environnement dynamique 147
4.5 Ville de San Francisco subdivisée en quatre zones de mobilité . 148
4.6 Taux de succès moyen de l'estimation des positions des véhicules par véhicule et pour les 6 derniers jours de l'ensemble de données de San Francisco. 149
4.7 Taux de succès de la prédiction des positions par objet pour l'ensemble de données de San Francisco (avec l'ajout des données synthétiques pour piétons) avec réduction de matrices. . . 149
4.8 Temps et Consommation énergétique estimée et effective de la migration d'un service en fonction de sa taille mémoire et pour différentes valeurs de la bande passante du lien 20 MBps 60 MBps et 100 MBps. 150
4.9 Schéma de la méthodologie des expérimentations sur le système de placement avec réévaluation 151
4.10 Valeurs moyennes des métriques consommation énergétique (J), nombre de violations de délai et temps d'exécution (ms) en fonction du nombre d'objets IoT et pour chaque méthode de placement. 154
4.11 Nombre de migrations par méthode et par scénario de mobilité 157
4.12 Valeur de la Fitness par méthode et par scénario de mobilité (10^3) 157
4.13 Nombre de migrations par méthode et par modèle de mobilité 159
4.14 Valeur de la fitness par méthode et par modèle de mobilité . . 159
4.15 Nombre de migrations par méthode et par classes d'applications. 160
4.16 Valeur de la fitness par méthode et par classes d'applications. 160
4.17 Valeurs moyennes des métriques de consommation énergétique, de violations de délai et du temps d'exécution par nombre de noeuds mobiles et pour chaque stratégie. 161

Liste des tableaux

1.1 Technologies de communication de l'Internet of Things [Yaqoob 2017],[Beh 2018]. 17
1.2 Classe d'applications définies par [Ricart 2020]. 23
1.3 Différences ente le Cloud, le Fog , le Edge et Mist computing [Yousefpour 2019], [Mahmud 2020]. 29
1.4 Résumé des travaux de placement d'applications dans le Fog. 54

2.1 Synthèse des simulateurs supportant le paradigme du Fog. Le Nombre de citations est pris des résultats de recherche sur IEEE Xplore et researchegate, consultés en septembre 2020. 67

3.1 Classes d'applications IoT et leur priorité 79
3.2 Résumé des variables des applications IoT. 81
3.3 Résumé des variables de l'infrastructure Fog. 84
3.4 Résumé des variables du problème de placement statique. . . . 88
3.5 Caractéristiques détaillées des noeuds de l'infrastructure Fog ($k \in [0, 3]$). 102
3.6 Classification des méthodes DPSO, GA et MDPSO selon l'objectif visé. De la méthode la plus recommandée (1) à la moins recommandée (3). 115

4.1 Résumé des variables du problème de placement dynamique. . 140
4.2 Tableau résumant le choix des méthodes selon l'objectif, le degré de mobilité de l'environnement et les classes d'applications. 163

Introduction

Les technologies et les concepts numériques évoluent très rapidement et sont peu à peu en train d'articuler tous les secteurs de la vie quotidienne, en proposant des services allant du simple loisir aux services les plus critiques touchant à la santé ou à la sécurité des individus. Derrière cette forte évolutivité des concepts et des technologies du numérique, on peut dégager deux grands axes (tendances) qui orientent progressivement le monde du numérique et qui changent peu à peu la manière dont les infrastructures de calcul et de communication sont opérées et offrent leur services.

1. L'utilisation de sources d'énergie propres :
   L'efficacité énergétique est devenue depuis quelques années un objectif prioritaire et commun à tous les secteurs technologiques. Les infrastructures de calcul et de communication qui sont en constante expansion et sollicitation sont des sources importantes de consommation énergétique et d'émission de gaz à effet de serre. De plus, le secteur de l'information et des télécommunications (ICT) est au cœur des autres secteurs technologiques et a un impact direct sur l'efficacité de ces derniers. L'ICT est utilisé pour rendre d'autres secteurs verts (smart building, smart motors). Afin d'assurer une efficacité énergétique à plus large échelle, les infrastructures de calculs et de communication doivent être alimentées par des sources d'énergie propres, ce qui permettrait d'assurer la pérennité de ces infrastructures [Thi 2019], [Pierson 2019].
2. La gestion autonomique des ressources numériques :
   Les infrastructures ICT de plus en plus complexes et dynamiques requièrent une gestion intelligente et autonomique de leur ressources. Il devient indispensable de s'appuyer sur des modèles qui permettent de réagir rapidement au contexte d'exécution afin d'assurer la performance et la fiabilité des ressources et services proposés. Les infrastructures ICT doivent pouvoir considérer les informations de mobilité des utilisateurs, détecter les anomalies, les corriger et pouvoir changer leur comportement en fonction des situations. Ce qui s'inscrit dans le cadre du self-management, self-healing et self-configuration.

L'Internet of Things (IoT) fait partie des concepts numériques qui ont grandement motivé les besoins en énergie propre et en gestion autonomique des infrastructures de calcul et de communication. L'Internet of Things consiste à identifier et à faire coopérer des objets du quotidien via le réseau internet pour proposer divers services numériques [Ray 2018], [Vermesan 2009]. L'IoT est au coeur d'une génération massive de données et d'une hyper-utilisation des infrastructures de calcul et de communication. IBM estime

que depuis 2014, 2.5 quintillion d'octets de données sont produites chaque jour [Nguyen 2014]. Cisco [Horwitz 2019] prédit que d'ici 2025 il y aura 75 milliards d'objets connectés déployés à travers le monde et qu'en moyenne un individu posséderait entre 5 et 8 objets connectés.

L'utilisation exclusive du Cloud Computing en tant que paradigme de calcul pour supporter les services des environnements IoT n'est plus possible et fait face à de nombreuses limites dont la sur-utilisation et la surcharge des réseaux pour communiquer avec les centres de données, le nombre astronomique de services IoT et la nature de trafic qu'ils génèrent (vidéo, voix etc.), les problèmes de sécurité et les exigences en temps de réponse quasi instantané des applications IoT [Karagiorgou 2012], [Bonomi 2012b].

Le Fog Computing est apparu comme une solution permettant aux infrastructures de calcul et de communication de répondre aux besoins en Qualité de Service (QoS) et de supporter intelligemment et d'une manière autonomique les changements et les dynamicités induites par les environnements IoT.

Le Fog Computing est un paradigme de calcul distribué qui vise à étendre les capacités de calcul et de stockage du Cloud et de rapprocher les services des utilisateurs finaux (objets IoT) en exploitant des noeuds de calcul se trouvant entre ces utilisateurs finaux et les centres de données distants [Michaela 2018a].

Les infrastructures Fog combinent à la fois les structures réseau, les centres de données et les noeuds intermédiaires de calcul entre les utilisateurs et le Cloud devant être considérés comme un seul grand système à gérer. Le Fog hérite des caractéristiques énergivores de ces différents environnements. Selon une étude de [NATF 2015], en 2012 la consommation électrique ainsi que l'empreinte carbone mondiale des infrastructures ICT étaient respectivement de 4.7% et de 1.7%. Les auteurs de [Belkhir 2018] ont estimé que, si elle n'est pas contrôlée, la contribution relative des émissions de gaz à effet de serre (GES) des infrastructures ICT pourrait passer d'environ 1 à 1,6 % en 2007 à plus de 14 % en 2040, ce qui représente plus de la moitié de la contribution relative de l'ensemble du secteur des transports en 2018.

Par ailleurs, le nombre important d'objets et de services IoT induit une importante et une constante sollicitation de l'infrastructure ICT qui ne fait qu'augmenter sa consommation en énergie.

L'intérêt pour la minimisation de l'impact énergétique des infrastructures de calcul est depuis quelques années mis en avant par la communauté du Green Computing [Khattar 2019]. Beaucoup de stratégies et d'algorithmes permettant d'utiliser efficacement les ressources de calcul ont été proposés [Courchelle 2018], ainsi que des travaux qui intègrent l'utilisation d'énergies propres pour alimenter les centre des données [Grange 2018]. Concernant les infrastructures réseau, les considérations pour la minimisation énergétique

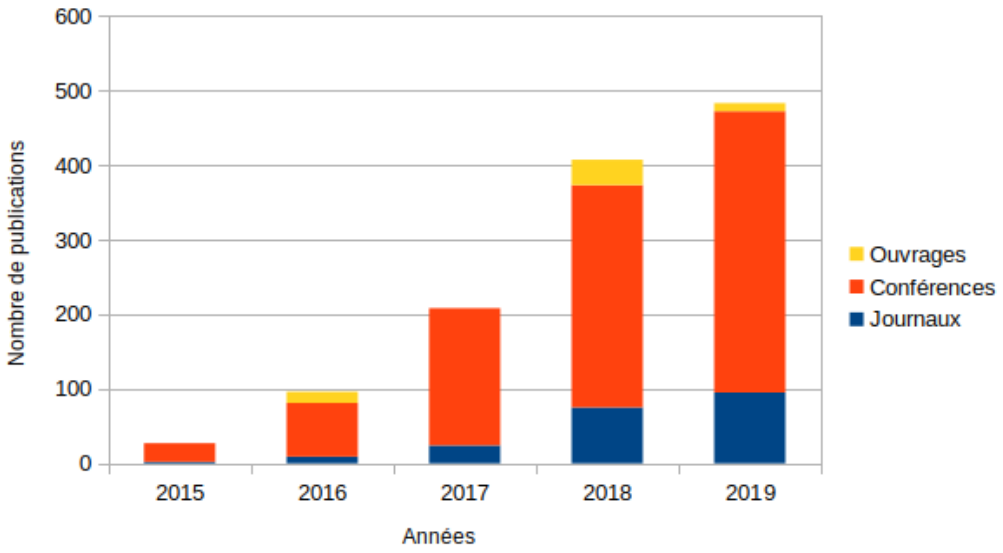


FIGURE 1 – Nombre de publications avec les termes "Fog" et "IoT" entre 2015 et 2019.

sont souvent intégrées dans les protocoles de routage ou de communication [Wang 2018a], [Wang 2018a] même si la priorité est souvent attribuée au temps d'acheminement de l'information et à son intégrité.

Les considérations énergétiques des environnements IoT se focalisent principalement sur les objets IoT et la durée de vie limitée de leurs batteries.

Le Fog Computing est un domaine de recherche assez récent et très peu de travaux ont été effectués sur la viabilité des infrastructures Fog pour supporter l'IoT ainsi que sur l'impact énergétique de ces environnements. La figure 1 présente le nombre de publications avec les mots clés "Fog" et "IoT" entre 2017 et 2019 (source IEEE xplorer data base).

Il est important d'établir des stratégies de gestion de ces environnements qui prennent en compte le facteur énergétique tout en considérant les différents besoins des services de l'IoT et leur dynamicité.

Contexte et Problématique de la thèse

Cadre de la thèse

Notre thèse s'inscrit dans le cadre de l'opération scientifique neOCampus de l'Université Toulouse III. L'opération, initiée en 2013, porte un ensemble de projets de recherche en relation avec les systèmes cyber-physiques ambiants et l'Internet des Objets (IoT). L'objectif étant de contribuer à la construction du campus du futur avec la conception et le déploiement de nombreux dispositifs logiciels et matériels inter-connectés, durable et intelligent alliant matériels

pédagogiques innovant, capteurs, systèmes de communication, de stockage et systèmes de localisations. Le projet regroupe actuellement 11 laboratoires de recherche qui joignent leurs compétences et leur expertises pour améliorer le confort au quotidien de la communauté universitaire tout en diminuant l'empreinte écologique des bâtiments et en réduisant les coûts de fonctionnement. neOCampus est simultanément un terrain d'expérimentations et d'innovations ouvert sous convention aux industriels en partenariat avec des chercheurs. Les services et/ou capteurs développés peuvent être aisément transférables dans les autres campus, écoles de l'Université Fédérale de Toulouse. Les principaux axes de recherche de neOCampus touchent à : l'énergie, la mobilité, l'environnement et la qualité de vie (intra et extra-bâtiment).

Problématique de la thèse

Une ville ou un campus intelligent est un système complexe qui possède de multiples dynamiques et des non linéarités induisant une impossibilité de prévoir les changements qui peuvent s'y produire. Dans cet environnement, instrumenté par des objets intelligents, de multiples services numériques peuvent être proposés. Ces services devant être déployés et gérés sur une infrastructure de calcul distribuée et large échelle.

Notre travail s'intéresse aux stratégies de placement des services composant les applications de l'Internet of Things sur une infrastructure de calcul Fog Computing. Ces emplacements devront assurer les besoins en qualité de services des applications tout en minimisant l'impact énergétique des infrastructures de calcul et de communication du Fog tout en considérant la dynamicité de l'IoT induite par la mobilité des noeuds.

La problématique de notre travail qui est le placement de services IoT dans les infrastructures Fog, illustrée en figure 2 a de nombreux verrous :

- La consommation énergétique conséquente des infrastructures de calculs, de communication et des objets IoT.
- Les exigences de plus en plus importantes de la qualité de service dues à la sensibilité des données et à la nature critique de certaine applications (sécurité).
- La dynamicité de l'environnement à travers la mobilité des noeuds IoT.
- les ressources de calcul et de stockage limités des noeuds Fog et des équipements IoT.
- L'hétérogénéité multi-niveaux des environnements IoT : structures des données, protocoles de communications, besoins en QoS, les fréquences d'envoi des capteurs, les capacités de calcul.
- La concurrence pour les ressources Fog entre différentes applications et utilisateurs.

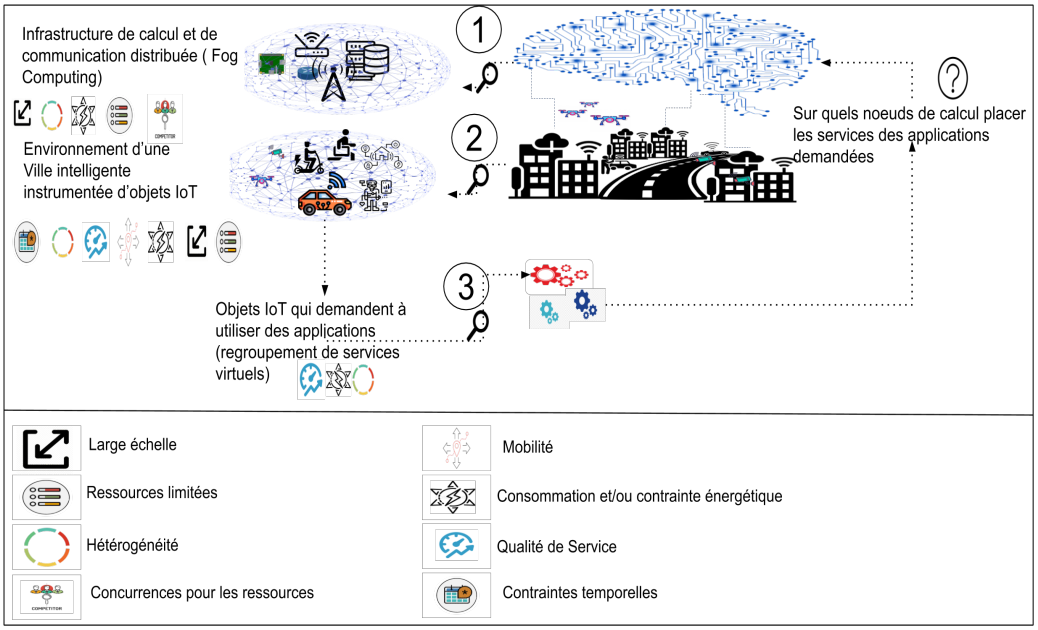


FIGURE 2 – Illustration de la problématique de placement dans le Fog.

Contributions

Cette section présente les contributions découlant de notre travail de thèse.

— **Modélisation des environnements Fog-IoT :**
Il s'agit dans un premier temps de définir et de modéliser les concepts relatifs aux environnements Fog et IoT. Nous exposons notre modèle de ressources Fog à savoir une représentation du système sous forme de graphe de ressources de différents types ainsi que notre représentation des caractéristiques des applications et services IoT. Nous définissons également la métrique de qualité de service considérée ainsi que le modèle d'estimation de la consommation énergétique de l'infrastructure Fog.

— **Modélisation du problème de placement de services IoT dans une infrastructure Fog statique :**
Cette première contribution présente le problème de placement de services IoT comme un problème d'optimisation multi-objectifs pour la réduction de la consommation énergétique de l'infrastructure Fog et l'amélioration de la Qualité de Service (QoS) dans l'ensemble des applications IoT. Nous proposons une approche de placement basée sur une méta-heuristique à intelligence par essaim avec le Discrete Particle Swarm Optimization (DPSO) et une approche méta-heuristique évolutionnaire avec l'algorithme génétique (GA) [Djemai 2019].

— **Placement de services IoT en environnement Fog dynamique**

et modèle de mobilité :
Cette seconde contribution intègre la notion de mobilité des objets IoT au problème de placement de services IoT dans le Fog. Nous avons dans un premier temps étendu le modèle Fog avec les variables de mobilité d'un noeud IoT. Ensuite, nous avons proposé un modèle de mobilité d'objet IoT à partir de leur historique de mouvement. Nous avons également proposé des équations estimant le temps et l'énergie consommée par la migration des services et des stratégies de placement et de migration de services IoT [Djemai 2020].

— **Extension du simulateur iFogSim et interconnexion avec une simulateur de moblité : SUMO :**
Nous avons également proposé une extension du simulateur iFogSim pour lever certaines contraintes réseau et supporter la mobilité des noeuds IoT via son intégration avec le simulateur SUMO.

Plan du manuscrit

Cette section présente le plan du manuscrit qui, suite à cette introduction s'articule en 4 chapitres et une conclusion générale.

— **Chapitre 1 :**
Ce chapitre permet de poser les définitions et concepts du Fog Computing, de l'Internet Of Things et d'identifier les avantages et les limites derrière l'utilisation des infrastructures Fog pour supporter les besoins des applications IoT. Nous présentons d'une manière globale le problème de placement de services vu comme un problème d'optimisation ainsi que les principales approches utilisées pour résoudre cette catégorie de problèmes. Ce chapitre essaie également de montrer l'impact énergétique considérable qui se trouve derrière l'utilisation des infrastructures de calcul et de communication étant à la base du Fog Computing et de l'Internet of Things. Enfin, nous présentons un état de l'art des travaux qui se sont intéressés au placement de services et à la gestion de la mobilité dans les environnements Fog-IoT.

— **Chapitre 2 :**
Ce chapitre est dédié à la présentation des sources de données pour les applications Internet of Things (IoT), qui sont assez rares et difficile d'accès, ainsi que les outils les plus pertinents à disposition de la communauté permettant de créer un environnement d'expérimentation pour le Fog Computing. Enfin, nous présentons les améliorations apportées au simulateur IFogSim que nous avons utilisé dans cette thèse.

— **Chapitre 3 :**
Ce chapitre présente notre modélisation mathématique des infrastructures Fog, des applications IoT et du problème de placement de services

IoT dans le Fog. Le problème considéré est un problème d'optimisation discrète multi-objectifs dont les deux objectifs considérés sont la réduction de la consommation énergétique des infrastructures Fog et la réduction des violations de QoS des applications IoT. Nous présentons une approche méta-heuristique pour la résolution du problème défini.

— **Chapitre 4 :**
Ce chapitre présente l'extension du modèle présenté dans le chapitre précédent pour intégrer la mobilité des noeuds IoT. Il présente également un modèle de mobilité des noeuds basé sur l'historique de leurs mouvements ainsi qu'une stratégie de placement et de migration de services en considérant les informations de mobilité.

— **Conclusion et Perspective :**
Ce chapitre donne un résumé des travaux réalisés lors de cette thèse, ainsi que des perspectives d'amélioration et d'évolution pouvant être envisagées.

Chapitre 1

Concepts préliminaires, bibliographie et synthèse

Sommaire

1.1 Introduction 10
1.2 Paradigme de l'Internet des objets 11
1.2.1 Définition 11
1.2.2 Les objets IoT 12
1.2.3 Les Applications IoT 20
1.3 Paradigme du Fog Computing 24
1.3.1 Définition 24
1.3.2 Architecture et éléments d'une infrastructure Fog Computing 25
1.3.3 Caractéristiques du Fog Computing 27
1.3.4 Concepts connexes au Fog : Edge et Mist computing 28
1.4 Les Avantages et les défis du Fog Computing pour supporter les applications IoT 30
1.4.1 Un environnement Fog-IoT 31
1.5 Gestion des applications IoT dans les environnements Fog-IoT 31
1.5.1 Les flux de services (workflow) 32
1.5.2 Objectifs et finalités d'un ordonnancement de tâches 33
1.5.3 Le placement de services vu comme un problème d'optimisation combinatoire 34
1.6 La consommation énergétique dans les environnements Fog-IoT 40
1.6.1 La consommation énergétique du secteur des technologies de l'information et des communications (ICT) 40
1.6.2 Calcul et estimation de la consommation énergétique dans les environnement ICT 43
1.6.3 La minimisation de la consommation énergétique dans des environnements Fog-IoT 45
1.7 La qualité de services dans les environnements Fog-IoT . . 49
1.8 La mobilité dans les environnements Fog-IoT 51
1.9 Synthèse 53

1.1 Introduction

L'Internet des objets ou «Internet of Things (IoT)» est un concept qui étend les infrastructures numériques actuelles avec les objets du quotidien. En rendant ces objets intelligents, addressables et dotés d'une capacité de calcul et de communication, le paradigme de l'IoT est considéré comme l'élément de base pour élaborer les services des villes intelligentes, de l'industrie 4.0 ou bien encore des futures systèmes de santé.

La mise en place d'un environnement IoT à large échelle est freinée par un certain nombre de défis tels que l'hétérogénéité des objets, leur durée de vie limitée, leur nature mobile et l'absence de plateformes pour gérer d'une manière autonome un très grand nombre d'objets et les services qu'ils proposent. La gestion des environnements IoT via les plateformes Cloud existantes est contrainte par le très grand nombre d'objets qui seront déployés (de l'ordre du milliard pour l'ensemble des plateformes), ainsi que par les besoins de traitement en temps réel et la privatisation de certaines applications [Bonomi 2012a].

Le Fog Computing est une infrastructure de calculs distribuée qui a pour principal objectif de rapprocher le calcul des noeuds utilisateurs finaux et de réduire ainsi les communications à destination des centres de données Cloud. Le Fog Computing permet de sécuriser davantage les applications en traitant les données utilisateurs localement, ce qui permet également de répondre aux besoins des applications dites sensibles au délai.

Le Fog Computing semble être l'infrastructure de calcul idéale pour supporter les besoins des applications IoT. Cependant, un certain nombre de défis restent à considérer avant de pouvoir combiner et utiliser efficacement le Fog Computing pour les environnements IoT.

Indépendemment l'un de l'autre, les environnements Fog et IoT sont considérés comme sources importantes de consommation énergétique. Cet aspect doit être pris en considération par les plateformes de gestion de ces infrastructures afin de réduire le coût énergétique et d'assurer la pérennité de ces systèmes.

La gestion de la Qualité de service (QoS) des applications IoT dans le Fog reste aussi une inconnue qui pourrait menacer l'utilisation de ces paradigmes à large échelle. Les applications IoT sont hétérogènes, certaines traitent des données sensibles touchant à la sécurité individuelle ou collective, d'autres exigent un traitement en temps réel et ont une importante consommation des ressources réseau. Il est important de pouvoir établir des stratégies aussi génériques que possible pour pouvoir prendre en compte ces besoins. La mobilité des noeuds est un autre aspect qui vient complexifier le processus de gestion des plateformes Fog-IoT.

Dans ce travail, nous présentons quelques approches pour améliorer l'effi-

cacité énergétique des infrastructures Fog-IoT et la QoS des applications qui y sont déployées tout en considérant la nature mobile de ces environnements.

Ce chapitre présente les concepts et notions nécessaires pour appréhender le reste du manuscrit ainsi qu'une bibliographie et une synthèse des travaux qui ont essayé de répondre à des problématiques similaires. Les deux premières sections présentent respectivement les paradigmes de l'IoT et du Fog Computing. La section 1.4 met en avant les avantages et les inconvénients de ces deux environnements combinés. Ensuite, dans la section 1.5, nous nous intéressons aux différents aspects liés à la gestion d'applications IoT dans un environnement Fog Computing. Les sections 1.6, 1.7, 1.8 présentent respectivement les trois aspects que nous avons pris en compte dans le cadre de cette thèse pour la gestion d'applications IoT déployées dans une infrastructure Fog Computing : l'énergie, la qualité de service (QoS) et la mobilité des noeuds.

1.2 Paradigme de l'Internet des objets

Le terme "Internet of Things" (IoT) a été utilisé pour la première fois en 1999 par Kevin Ashton pour parler des réseaux à Identification par Radio Fréquence (RFID). La portée du terme IoT a par la suite évolué grâce aux avancées dans le domaine des technologies de l'information et des télécommunication (TIC). Ces avancées ont permis l'élaboration d'une multitude d'équipements de calcul, légers et peu onéreux. Ces objets référencés par les termes "objet IoT", "objet connecté" ou "objets intelligents" peuvent intercepter les différents changements de leur environnement extérieur et agir en conséquence. L'origine sémantique de l'expression est composée de deux mots et concepts : Internet et objets, où Internet peut être défini comme le réseau mondial de réseaux informatiques inter-connectés, basé sur un protocole de communication standard, la suite Internet (UDP-TCP/IP), tandis que les objets sont des entités physiques qui ne sont pas précisément identifiables [ETS].

1.2.1 Définition

Le domaine de l'Internet des Objets est en constante évolution et à ce jour, on ne trouve pas de définition standardisée. Les travaux [Perera 2014] et [CERP-IoT 2010] définissent l'Internet des Objets comme suit : "L'Internet des objets permet aux personnes et aux objets d'être connectés à tout moment, en tout lieu, avec n'importe quel autre objet et n'importe quel utilisateur, en utilisant idéalement n'importe quel chemin/réseau et n'importe quel service."[Xu 2014b] prend la définition suivante : "Les objets ont des identités et des représentations virtuelles qui fonctionnent dans des espaces intelligents utilisant des interfaces intelligentes pour se connecter et communiquer dans des contextes sociaux, environnementaux et d'utilisateurs".

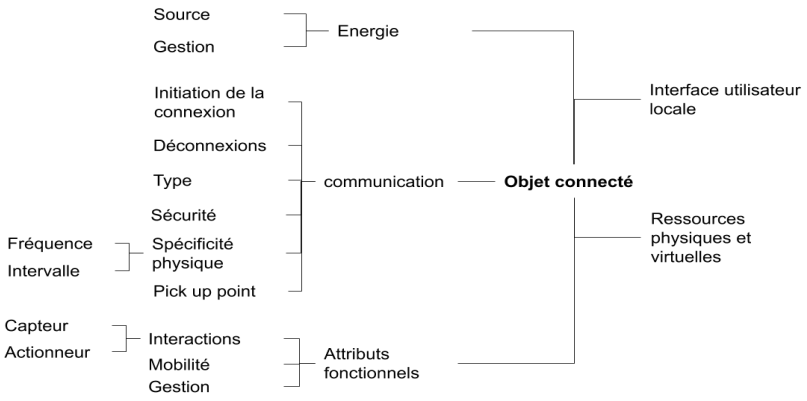


FIGURE 1.1 – Taxonomie des objets connectées [Dorsemaine 2015]

Dans le cadre de cette thèse, nous utiliserons la définition suivante qui est basée sur les travaux [Amir M. Rahmani 2017] : "L'Internet des Objets ou l'Internet of Things (IoT) peut être défini comme un réseau auto-adaptatif et intelligent connectant les objets intelligents du monde réel via le réseau Internet afin de proposer un nouveau type de services."

1.2.2 Les objets IoT

Un objet connecté est un objet physique, addressable, équipé de capteurs et ou d'actionneurs qui lui permettent d'interagir avec son environnement extérieur. Il peut également communiquer avec d'autres équipements via différentes technologies de communication (Wi-Fi, Bluetooth, réseaux de téléphonie mobile, réseau radio à longue portée de type Sigfox ou LoRa etc.), le reliant à Internet ou à un réseau local ou un autre réseau propriétaire. Les objets connectés ont plusieurs particularités à prendre en compte afin de pouvoir gérer efficacement un environnement IoT. [Dorsemaine 2015] établit une taxonomie des objets connectés, représentée en figure 1.1.

Les ressources physiques et virtuelles La capacité de stockage, de mémoire vive et de calcul d'un équipement IoT a un impact direct sur l'intelligence qu'il peut intégré et par conséquent sur les stratégies de placement de services possibles.

Les attributs fonctionnels

— Interactions : les données créées et les instructions reçues par un objet IoT se font via les capteurs et les actionneurs qu'il possède. Les capteurs permettent de récupérer les informations de l'environnement extérieur

de façon périodique ou événementielle. Il existe des capteurs à mémoire qui peuvent stocker des informations pendant un certain temps avant de les envoyer. Cela permet de réduire la consommation énergétique de l'équipement. Les capteurs sans mémoire doivent envoyer les données en continu, autrement les données récupérées à un instant donné seront perdues. Les actionneurs permettent d'interagir avec leur environnement à la suite d'une instruction reçue.

— Mobilité : Les objets connectés se divisent en deux catégories : les objets fixes conçus pour des environnements statiques comme les habitations, et les objets conçus pour les environnements dynamiques à l'instar des voitures autonomes et des drones.

— Gestion : Certains objets comme les caméras permettent une modification de leurs attributs d'interactions tels que la fréquence d'envoi des données, la taille des données et les paramètres de mise en veille.

L'énergie L'énergie est un aspect critique des environnements IoT, qui impacte dans certains cas la durée de vie des équipements ainsi que les stratégies de gestion applicables au système. Les objets IoT sont conçus avec pour objectif d'être de plus en plus petits et de moins en moins coûteux à la production ce qui impacte négativement leur efficacité énergétique [Dorsemaine 2015],[IEA 2017]. Les caractéristiques énergétiques d'un objet IoT se divisent en deux parties.

— La source énergétique de l'équipement : La manière dont un objet utilise sa puissance est extrêmement importante. Cette dernière détermine de nombreuses autres caractéristiques comme sa capacité à travailler en continu et la puissance de calcul du processeur. Dans [rfc 2014], l'auteur fournit quatre types de sources d'énergie pour un objet connecté : la récolte (l'énergie est recueillie dans l'environnement, par exemple des panneaux solaires), la recharge ou le remplacement périodique, la source primaire non remplaçable (la source d'énergie détermine la durée de vie de l'objet) et l'alimentation principale (la source d'énergie est pratiquement illimitée).

— La gestion de la consommation énergétique de l'équipement : La gestion de la source d'énergie peut se résumer à la façon dont la source d'énergie est utilisée et à la durée pendant laquelle l'objet peut fonctionner avec une quantité d'énergie donnée. [rfc 2014] décrit les trois approches pour la gestion de la consommation d'énergie des interfaces de communication des objets. L'objet est souvent éteint et se réveille périodiquement ou sur un signal donné.

La Communication IoT Un objet peut posséder plusieurs interfaces de communication qui utilisent différents protocoles. Dans ce qui suit nous allons

identifier les principales technologies de communication dans l'IoT.

Type de communications sans fil Les communications IoT sont majoritairement sans fils, elles peuvent être à longue, moyenne ou à courte portée et peuvent être aussi bien propriétaire que libre d'utilisation. La technologie réseau utilisée par les applications est un aspect non négligeable à considérer pour pouvoir gérer efficacement les performances ainsi que la consommation énergétique de ces dernières. Un système de gestion de ressources Fog doit prendre en considération les principales caractéristiques du réseau en terme de portée, latence, consommation énergétique, bande passante, sécurité et coût financier qui varient en fonction de la technologie réseau. Dans ce qui suit nous intéressons à la porté, bande passante et consommation énergétique des réseaux non filaires les plus utilisés dans l'IoT.

- Technologies courte portée (Indoor) : La concentration des appareils dans une zone géographique limitée permet d'utiliser des technologies de télécommunication à courte portée. Si nécessaire, plusieurs appareils communiquant localement peuvent créer un maillage couvrant une large zone.
  - Bluetooth Low Energy (BLE) : la technologie Bluetooth Low Energy est également appelée Bluetooth LE, BLE ou encore Bluetooth Smart. Elle est apparue en 2010 avec la sortie de la version 4.0 du Bluetooth Core Specification. Le Bluetooth Low Energy est une alternative au Bluetooth classique, avec une réduction du coût et de la consommation en puissance, tout en conservant une portée de communication équivalente. Toutes les versions du Bluetooth parues avant la Core Specification 4.0 sont dites «Bluetooth classique». Seul des topologies de communication en étoile sont autorisés, avec un maître central et quelques esclaves périphériques. Le Bluetooth Low Energie cherche à adresser des appareils à faible puissance de calcul, à faible coût de production et dont l'espérance de vie est à maximiser. L'idée est donc de réduire les temps d'activité par rapport à des appareils basés sur le Bluetooth classique. Les performances brutes telles que le débit de transmission en seront donc tout naturellement affectées. Le BLE a une portée de 50 m avec une latence de connexion de 6 ms [ble].
  - Zigbee : est un protocole radio développé par l'Alliance Zigbee. Contrairement aux dispositifs BLE, les réseaux Zigbee peuvent être organisés en mailles. Les cas d'utilisation de Zigbee sont assez similaires à ceux de Z-wave qui concernent la domotique en général. Cependant, puisque Zigbee est une norme ouverte, plus de fabricants peuvent produire des appareils Zigbee. Ce qui créer un écosystème plus diversifié, mais génère des problèmes d'interopérabilité entre

des appareils qui sont censés être basés sur la même technologie [Zigbee 1998].

- 6LowPan : est l'acronyme de "IPv6 over Low-Power Wireless Personal Area Networks", proposé dans l'IETF 5 RFC 49446. Le déploiement d'un réseau IP sur des réseaux à faible consommation de puissance, permet la création d'un réseau maillé au niveau des paquets (basé sur le modèle OSI) [N. Kushalnagar],[G. Montenegro]. 6LoWPAN combine à la fois la dernière version du protocole Internet (IPv6) et des réseaux personnel à faible énergie (Low-power Wireless Personal Area Networks (LoWPAN). 6LoWPAN permet donc à des équipements de faible puissance de transmettre de l'information avec un protocole compatible avec internet.

— Technologies à moyenne portée

- Wi-Fi : le Wi-Fi, aussi orthographié wifi suggère la contraction de «Wireless Fidelity», par analogie au terme «Hi-Fi» pour «High Fidelity». Le Wi-Fi est un ensemble de protocoles de communication sans fil régis par les normes du groupe IEEE 802.11 (ISO/CEI 8802-11). Un réseau Wi-Fi permet de relier par ondes radio plusieurs appareils informatiques (ordinateur, routeur, smartphone, modem Internet, etc.) au sein d'un réseau informatique afin de permettre la transmission de données entre eux [Wi-Fi]. Grâce aux normes Wi-Fi, il est possible de créer des réseaux locaux sans fil à haut débit. Le débit peut aller de 11 Mbit/s théoriques ou 6 Mbit/s réels en 802.11b, à 54 Mbit/s théoriques ou environ 25 Mbit/s réels en 802.11a ou 802.11g, 600 Mbit/s théoriques pour le 802.11n3,4 et 1,3 Gbit/s théoriques pour le 802.11ac normalisé depuis décembre 2013. La portée peut atteindre plusieurs dizaines de mètres en intérieur (généralement entre une vingtaine et une cinquantaine de mètres) s'il n'y a aucun obstacle gênant entre l'émetteur et l'utilisateur. Ainsi, des fournisseurs d'accès à Internet peuvent établir un réseau Wi-Fi connecté à Internet dans une zone à forte concentration d'utilisateurs (gare, aéroport, hôtel, train, etc.). Ces zones ou points d'accès sont appelés bornes ou points d'accès Wi-Fi ou « hot spots ».
- Z-Wave : est une technologie de communication sans fil à courte portée en réseau maillé basée sur une technologie radio propriétaire [Z-Wave 1999].

— Technologies longue portée

- Réseaux cellulaires

  Le réseau cellulaire est un réseau de communications destiné aux équipements mobiles. Le réseau est découpé en cellules qui possèdent chacune une station de base, ou BTS (Base Transceiver Station), qui s'occupe des transmissions radio sur la cellule et dessers les clients

d'une zone géographique donnée. Associés à la station de base, des canaux de signalisation vont permettre aux mobiles de communiquer avec la BTS et vice versa. Chaque station de base est reliée à un contrôleur de station de base ou BCS (Base Station Controller). Les technologies les plus répandues actuellement basé sur la norme GSM (Global System for Mobile communication) sont la 4G LTE et la 4G LTE+ et certains fournisseurs commencent à mettre en place les réseaux 5G

— LoRaWAN : est une technologie de communication soutenue par l'alliance LoRa , et contrairement à Sigfox, elle n'est pas liée à un opérateur. La topologie du réseau rendue possible par LoRa est cependant assez similaire à Sigfox : les appareils communiquent avec des passerelles qui sont connectées aux réseaux "traditionnels", et rendent les messages disponibles à l'utilisateur sur des serveurs dédiés. Lorsqu'un dispositif LoRa se réveille pour envoyer des données il est brièvement possible de lui envoyer un message, ce qui permet une communication bidirectionnelle [LoRa 2009].

— Sigfox : est à la fois un opérateur de réseau et une technologie de communication déployée par le dit opérateur. Les appareils Sigfox communiquent avec les passerelles Sigfox, qui sont connectés à Internet. Les messages produits par les appareils Sigfox sont donc stockés sur des serveurs pour être accessibles via une interface Web du côté client [Sigfox 2010].

— Narrowband Internet of Things (NB-IoT) : le Narrowband IoT désigne un standard de communication dédié à l'Internet des objets et normalisé en 2016. NB-IoT est standard de communication LPWAN dont la principale mission consiste à faciliter la communication d'importants volumes de données sur une très grande distance. Il permet d'avoir des débits de 20 à 250Kbit/s en download ou upload avec une latence inférieure à 10 secondes environ [Chen 2017].

Point de collecte, initialisation de la connexion et processus de déconnexion Le point de collecte local est le noeud qui récupère les données d'un certain nombre d'objets autour de lui. L'utilisation de ce point peut impacter la durée de vie des objets autour ainsi que leur consommation énergétique. Les objets pourront éviter l'utilisation de protocoles lourds pour échanger avec d'autres éléments du système.

Le processus de déconnexion est autorisé par certains objets comme les montres connectées et le composant "podomètre" qui peut collecter des données même si la montre n'est pas connectée au réseau.

L'initiation de la communication peut se faire par les objets ou par les points de collecte. Si l'objet initie la communication, il peut éteindre ses in-

Technologie	Puissance	Portée	Fréquence
WiFi 5 et 6	Modérée	Modérée [10m,100m]	Haute
LoRWan	Très faible	Longue [2km-10km]	Faible
Sigfox	Très faible	Longue [2km-10km]	Faible
Bluetooth Low Energy	Faible	Trés courte[1m-100m]	Faible
ZigBee	Très faible	Courte [10m-100m]	Faible
Z-wave	Très faible	Courte 100m	Faible
6LoWPAN	Très faible	Courte	Faible
LTE	Haute	30km	Haute
NB-IoT	Très faible	Longue	Faible

TABLE 1.1 – Technologies de communication de l'Internet of Things [Yaqoob 2017],[Beh 2018].

terfaces lorsqu'il ne les utilise pas et économise ainsi son énergie. Si c'est le point de collecte qui initie la communication l'objet devra continuellement attendre le signal de l'équipement.

Spécificité physique Le débit et la portée de communication sont intimement liés et ont un rôle important dans le choix des spécifications énergétiques : plus le débit et la portée sont faibles, plus la consommation électrique sera faible pour une certaine quantité de données à transmettre, c'est-à-dire la quantité maximale théorique de données que l'interface de communication peut envoyer dans un laps de temps donné. Elle est généralement mesurée en Kb/s ou Mb/s. Une bonne façon de classer les objets en fonction de ce critère pourrait être les technologies utilisées ou les plages de débits, c'est-à-dire la distance maximale théorique que les données peuvent atteindre tout en restant compréhensibles avec une technologie donnée. Pour classer les objets en fonction de cette caractéristique, on peut utiliser les technologies ou les plages de distances en mètres ou les types de réseaux (PAN, LAN, échelle du bâtiment, échelle de la ville, etc.).

Sécurité Les informations collectées et transmises par les objets IoT relèvent majoritairement de la vie privé des utilisateurs et de données sensibles et stra-

tégiques liées à des infrastructures collectives telles que les villes intelligentes.

Interface locale utilisateur Certains objets comme les smartphones possèdent des composants dédiés à une interaction directe avec l'utilisateur afin qu'il puisse le configurer et utiliser ses fonctions de base. Il existe quatre types d'interfaces :

- Actives : des parties de l'objet sont dédiées à l'interaction avec l'utilisateur (boutons).
- Passives : L'utilisateur ne peut pas interagir directement avec l'objet, mais il peut communiquer avec lui par l'intermédiaire de divers éléments (écran, lumières, sons, vibrations)
- Hybride : l'objet et l'utilisateur peuvent interagir entre eux en mode passif ou actif.
- L'objet et l'utilisateur ne peuvent jamais directement interagir l'un avec l'autre [Dorsemaine 2015].

Contraintes des objets IoT

- Hétérogénéité : l'hétérogénéité des objets IoT se manifeste à plusieurs niveaux : capacité de calcul, caractéristiques des capteurs et actionneurs ainsi que leurs protocoles de communications. Ces aspects comme précisé dans des travaux tels que [Zanella 2012], [Barnaghi 2012], [Foteinos 2013] complexifient le processus de contrôle et de gestion où on a les mêmes actions qui ne donnent pas nécessairement les mêmes résultats.
- Large échelle : le nombre important d'objets proposant des services et l'augmentation conséquente des données générées et échangées à travers les différents réseaux complexifie surcharge les infrastructures de calculs et complexifie la gestion efficace des besoins de leurs applications.
- Mobilité : Le nombre important d'objets IoT et l'aspect imprédictible de leurs mouvements perturbent l'efficacité des systèmes de gestion pour anticiper d'éventuelles dégradations de performances de leurs services et complexifient la gestion de la Qualité de Service (QoS) au cours du temps.
- Contraintes énergétiques et durée de vie : La stratégie de conception à faibles coûts de composants légers, amovibles, s'intégrant facilement à l'environnement extérieur impacte sur la qualité et la duré de vie des batteries des équipements IoT. Ce qui ajoute un nouvelle contrainte à prendre en compte pour la gestion des ressources IoT.
- Application et service sensibles au contexte et au délai : Certaines applications IoT telles que les applications de type réalité augmentée exigent des temps de réponse très courts inférieurs à 20ms alors qu'en moyenne le Cloud ne peut offrir un temps de réponse inférieur 100ms. A cela

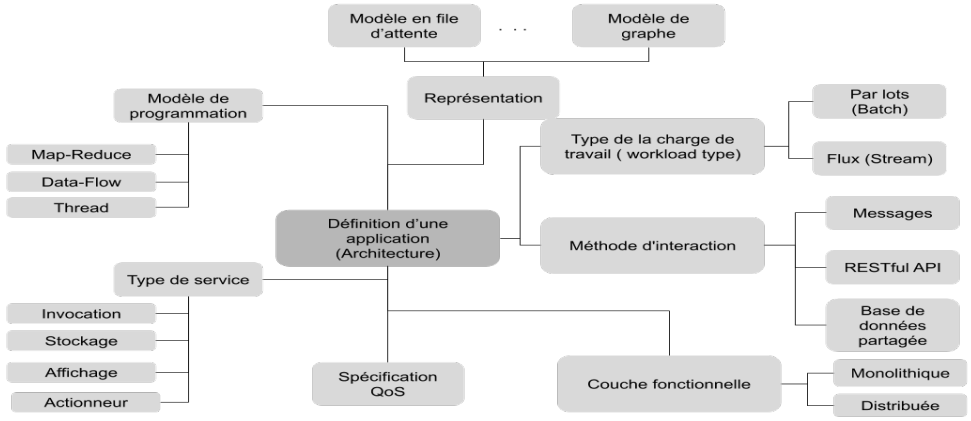


FIGURE 1.2 – Taxonomie pour définir l'architecture d'une application [Mahmud 2020]

s'ajoute, le nombre important d'utilisateurs qui doivent se partager la bande passante et ainsi impacter les délais d'accès au services du Cloud .

— Faille de sécurité : Chaque année, les chercheurs du Global Risks Report travaillent avec des experts et des décideurs du monde entier pour identifier et analyser les risques les plus urgents auxquels le monde est confronté. Leur rapport de 2018 identifie les problèmes liés aux cyberattaques comme l'un de problèmes majeurs des systèmes IoT.
Il n'y a actuellement aucun consensus sur la façon de mettre en œuvre la sécurité dans l'IoT. [Balte 2015] résume les principaux axes impactant la sécurité dans l'IoT : le contrôle d'accès des noeuds, les mécanismes d'authentification, la privatisation et la confidentialité des données générées par les objets, l'introduction d'un langage de négociation de confiance bien défini, d'un mécanisme de négociation de confiance pour le contrôle d'accès au flux de données.
Les environnements IoT interagissent avec d'autres systèmes via des intergiciels. Ces derniers peuvent présenter d'importantes failles de sécurité et être sujets aux attaques dont les répercutions se feront sur un nombre considérables de noeuds (geler les activités d'une smart city, des industries etc). De plus, l'aspect mobilité des noeuds qui passent d'un réseau à un autre a également un impact sur la sécurité et peut fragiliser les dispositifs mis en place.

1.2.3 Les Applications IoT

Les applications de l'Internet of Things sont diverses et variées. Dans le contexte de cette thèse, une application IoT est un regroupement de services virtuels interagissant entre eux et avec un ou plusieurs objets IoT pour réaliser un service global. La particularité d'une application IoT comparée aux applications déjà existantes réside dans leurs sources de données qui viennent d'équipements IoT ainsi que leurs utilisateurs qui peuvent être aussi bien d'autres objets IoT qui interagissent avec l'environnement extérieur que des équipements standards. Les applications IoT doivent traiter un très grand nombre de données en très peu de temps.

[Mahmud 2020] propose une classification des principaux attributs et aspects d'une application IoT, représentés en Figure 1.2.

Couche fonctionnelle Une application peut effectuer plusieurs opérations sur les données. La couche fonctionnelle définit les unités qui effectuent ces opérations de traitement. [Mahmud 2020] propose de classer la couche fonctionnelle en deux catégories : (1) application monolithique où toutes les opérations sont encapsulées dans un seul programme. Dans ces applications, des techniques de parallélisme sont appliquées pour que le programme puisse s'exécuter sur plusieurs coeurs CPU d'un noeud. (2) Application distribuée où les opérations de calculs sont encapsulées dans différents programmes ce qui permet d'ajouter d'autres programmes et de modifier ou d'étendre les fonctionnalités de l'application.

Charge de travail La catégorie de données d'entrée d'une application est un paramètre important pour déterminer la stratégie de placement de ses services sur l'infrastructure. Il existe deux classes de charge de travail : (1) par lots où l'application est non-interactive et traite les données par ensembles émanant de plusieurs sources. (2) l'application à flux de données (stream) qui reçoit des données générées par différentes sources périodiquement. Les applications IoT temps réel sont souvent des applications "stream".

Qualité de service (QoS) des applications IoT Le déploiement d'un cadre efficace pour supporter les applications IoT présente de nombreux défis, l'un d'entre eux étant de répondre aux exigences en qualité de service de ces applications que ce soit en termes d'efficacité énergétique, de qualité des données collectées, de consommation des ressources réseau ou bien de la latence.

Concevoir et développer une méthode qui permet de garantir la QoS des applications IoT est un défi avec les plateformes actuelles dans le contexte "IoT programming models" (e.g. Amazon IoT, Google Cloud Dataflow, IBM Quark) et des méthodes de gestion de ressources [Alqahtani A 2016].

Il a été vu dans la section 1.2.2 que la nature hétérogène des équipements physiques et virtuels, des données et des différentes couches logicielles complexifient le traçage et la garantie de la QoS.

Dans un contexte purement commercial, les exigences en qualité de service sont formellement spécifiées dans un document d'accord de niveau de service (SLA) [Galati 2014] qui sert de base à l'accord juridique et à la compréhension des termes, conditions et engagements de service entre les consommateurs et les fournisseurs. Par exemple, le document SLA d'Amazon Web Services indiquant les termes, conditions et engagements pour ses services S3 et EC2 peut être trouvé à [Ama a] et [Ama b] respectivement.

Comme les applications IoT ont une architecture en couche et des flux de données complexes les traversant, il est nécessaire de modéliser d'abord les SLA pour les couches individuelles, puis de les agréger de manière holistique. Ce document d'agrégation de SLA (modèle) servira de base pour spécifier un SLA de bout en bout qui peut être utilisé pour spécifier les termes, conditions et engagements de service pour une application IoT.

Le groupe OpenIoT propose un gestionnaire de Qualité de service des applications IoT [Michaela 2018b]. Ce gestionnaire est spécifique aux applications IoT de type publisher/subscriber qui sont déployées dans les plateformes Cloud mais les principes et métriques mis en avant dans le document restent valables pour des cas plus génériques incluant les infrastructures Fog. Le composant "QoS manager" regroupe deux principaux modules : (1) la surveillance et la gestion des abonnements à la qualité de service, par laquelle le gestionnaire de la qualité de service regroupe les abonnements à de multiples données de capteurs (c'est-à-dire les requêtes continues) et détermine les besoins globaux des applications en ce qui concerne l'acquisition de données de capteurs ; et (2) la surveillance et la gestion des publications de la qualité de service, par laquelle le gestionnaire de la qualité de service regroupe les publications de données de capteurs surveillées et gère le processus d'acquisition de données de capteurs afin d'optimiser la consommation d'énergie et de bande passante tout en répondant aux besoins des applications.

Les auteurs identifient un ensemble de métriques Quality of Service (QoS) relatives au composant physique d'un capteur ainsi que des métriques pour estimer l'efficacité d'un réseau de composants physiques qui interagissent dans le cadre d'une seule application.

— Précision : qualité des capteurs qui déterminent l'exactitude et la sensibilité des mesures fournies par les capteurs.
— Consommation énergétique : calculée sur la base d'un modèle énergétique spécifique (par exemple, en considérant le nombre total/volume de paquets/données). En particulier, dans le cas des réseaux de capteurs sans fil (WSN), la consommation d'énergie est directement associée à la durée de vie du réseau de capteurs.

— Bande passante : la largeur de bande d'un capteur physique est une mesure du débit binaire, représentant les ressources de communication de données disponibles ou consommées, exprimées en bits par seconde ou en multiples de celles-ci (bit/s, kbit/s, Mbit/s, Gbit/s, etc.)
— Volume de données : le volume de données (quantité de données) produit par un capteur physique.
— Fiabilité : la fiabilité peut être mesurée en utilisant un capteur pour fournir des mesures en temps réel avec le champ d'application de ses paramètres techniques, de sorte que la fiabilité peut être corrélée avec la qualité du capteur.
— Latence : le délai d'attente subi par un système.
— Qualité réseau : la qualité d'un réseau de capteurs est déterminée par la qualité des données fournies en réponse à une requête.
— Optimisation des ressources et efficacité des coûts : mesure de la capacité du système à maximiser le bien-être social, défini comme une allocation efficace de ressources limitées dans la société pour optimiser l'utilisation des ressources.

Le document met aussi en évidence le cas d'un environnement dynamique où les applications IoT interagissent avec des objets mobiles. Les auteurs préconisent des récupérations de données opportunistes en fonction de la localisation des utilisateurs et de leurs besoins. La gestion de la batterie et de la durée de vie des noeuds est une question plus délicate considérant l'aspect mobilité qui est difficilement prédictible.

Dans son manuscrit de thèse [Banouar 2018], l'auteur présente un ensemble de critères QoS basés sur le document technique IEEE [IEE 1990], en considérant l'application IoT dans sa globalité et non pas en séparant composant par composant (les objets, les services etc.).

— La priorité : peut être considérée comme l'importance attribuée à une requête ou sa réponse par rapport à une autre. Le trafic de l'entité IoT ayant une priorité plus élevée doit être traité en priorité par le système, suivant une politique de gestion appropriée.
— Le délai : on distingue deux types de délai. Le premier est le délai d'aller-retour (RTT - Round Trip Time) qui correspond au temps nécessaire pour répondre à une demande. Il est considéré comme la somme du temps d'envoi de la demande et de la réponse, ainsi que du temps de traitement de différents composants parcourus dans le système IoT. Le deuxième type est le délai de bout en bout correspondant, par exemple, au transfert de données entre l'équipement et l'application IoT. Par exemple, [sko10] a fixé un délai maximal de 300 ms pour le transfert d'une électrocardiographie en temps réel.
— La périodicité : est l'intervalle temporel d'exécution des actions. Au-delà de cet intervalle, la donnée devient obsolète. La périodicité est une

Classe	Catégories d'applications	Exemples
Volumineuses (GB)	Temps Réel HD, 4K, Ultra-haute résolution d'images	fusion vidéo en temps réel, modélisation 3D des villes, lecture d'images médicales à distance DICOM
Rapides	Synchronisées avec l'environnement, systèmes cyber physique, Streaming	Réalité Augmentée sur smartphone, contrôleur de production
Spéciales	Besoin de privatisation, Application distribuées entre le Edge et le Clouds	Images médicales privées, Vidéo de surveillance, contrôle routier

TABLE 1.2 – Classe d'applications définies par [Ricart 2020].

caractéristique spécifique des applications IoT de surveillance. Les données collectées peuvent avoir différentes périodes selon l'activité métier.

— L'intégrité des données : garantit la non altération des données, que ce soit par accident ou par des parties non autorisées. Les données collectées provenant des équipements ne doivent pas être modifiées pour assurer la réalisation de l'action correcte par l'application IoT.
— La confidentialité : liée à l'assurance que les données ou le système IoT ne peuvent être accessibles que par ceux qui en ont l'autorisation. Cette propriété est essentielle dans ces systèmes qui doivent garantir que les informations des équipements ou machines ne sont accessibles qu'aux parties autorisées.
— La disponibilité : est définie comme le taux de disponibilité opérationnelle d'un système ou d'un composant lorsqu'il est requis pour utilisation. Pour les applications IoT critiques (par exemple, la surveillance des signes vitaux d'un patient), la disponibilité est une exigence importante et qui doit être très proche de 100%.
— La fiabilité : fait référence à la capacité d'un système ou d'un composant à exécuter ses fonctions dans des conditions données et pour une période de temps spécifiée.

[Ricart 2020] présente trois catégories d'applications présentées dans le tableau 1.2 : volumineuses, rapides et à traitement particulier. Ces dernières requièrent un autre paradigme de calcul que le Cloud si elles sont déployées en grand nombre.

L'auteur présente également un diagramme des besoins en latence par famille d'applications, présenté en figure 1.3. On voit que les besoins en latence sont très hétérogènes et peuvent varier de 100 ns à 10 s.

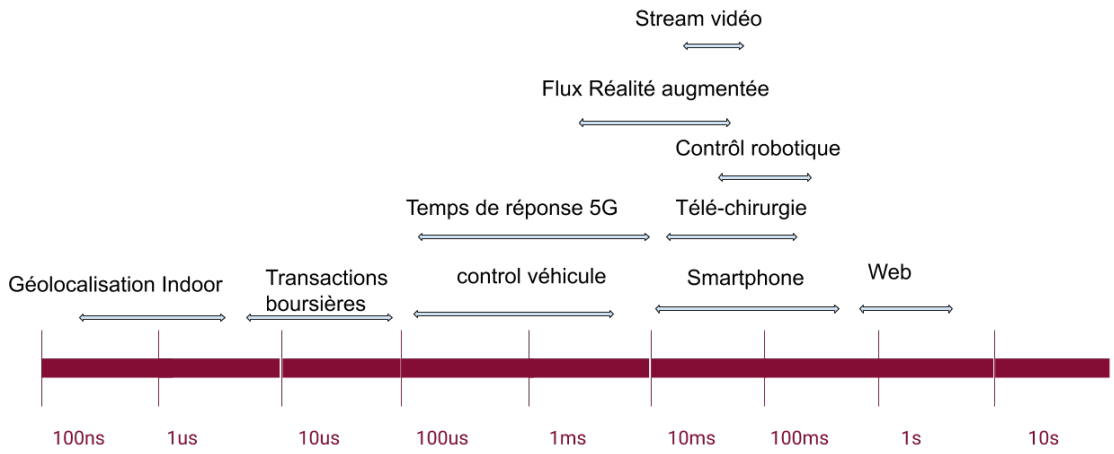


FIGURE 1.3 – Intervalles des besoins en latence par groupes d'applications [Ricart 2020].

1.3 Paradigme du Fog Computing

1.3.1 Définition

Le terme "Fog Computing" a été utilisé pour la première fois en 2012 par Jonathan Bar-Magen Numhauser dans sa proposition de thèse "Fog Computing introduction to a New Cloud Evolution" [Jonathan 2012]. L'auteur y présente les limites du Cloud Computing face aux exigences en QoS des nouvelles applications telles que les applications de l'Internet of Things (IoT) et à l'augmentation conséquente du nombre d'utilisateurs ainsi que le trafic réseau qu'ils génèrent.

Il propose un nouveau paradigme de calcul, qu'il appelle "Fog Computing". Ce dernier a pour pour but d'étendre les capacités et les services offerts par les infrastructures du Cloud Computing. Par la suite, le terme est repris par Flavio Bonomi en 2012 dans son article "Fog Computing and Its Role in the Internet of Things" [Bonomi 2012b]. Il met en évidence les avantages et les limites que peut avoir ce nouveau paradigme de calcul dans des environnements de type Internet of Things (IoT).

En 2015, l'université de Princeton en collaboration avec les industriels Cisco Systems, Intel, Microsoft et ARM fondent l'OpenFog consortium afin de promouvoir le Fog computing dans le monde industriel.

Ils publient en 2016 "OpenFog Reference Architecture" qui expose les principales caractéristiques des Architectures Fog [Working 2016] que nous allons voir dans la section 1.3.3.

En 2017, le National Institute of Standards and Technology (NIST) publie un rapport technique [Michaela 2018a] qui présente un modèle conceptuel

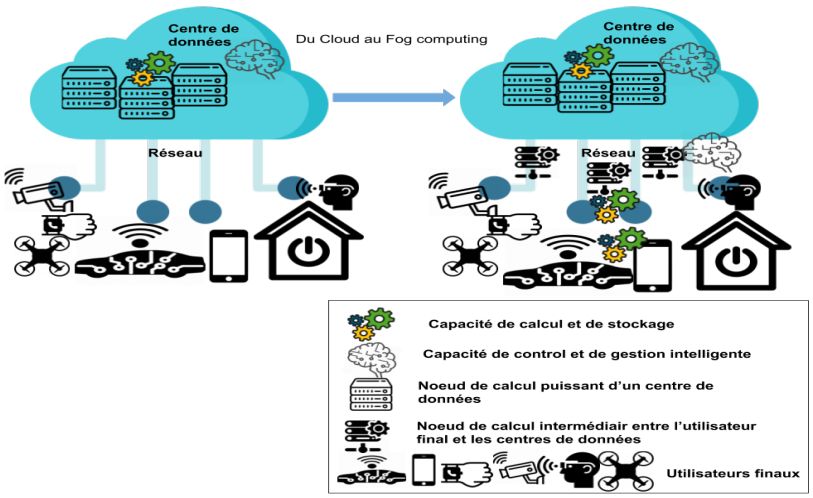


FIGURE 1.4 – Illustration de l'évolution de l'architecture Cloud vers le Fog.

standardisé du Fog, les caractéristiques du Fog, les stratégies de déploiement des services dans ce type d'infrastructures.

Dans le cadre de cette thèse, on considère la définition suivante qui regroupe à la fois celle du NIST [Michaela 2018a] et celle de l'OpenFog consortium [Working 2016] : *"Le Fog Computing ou Fog Networking est un paradigme de calculs distribué à large échelle et hiérarchique, qui permet d'augmenter les capacités de calcul, de stockage et de communication des serveurs Cloud par celles des noeuds intermédiaires situés entre ces serveurs et les utilisateurs finaux."*

1.3.2 Architecture et éléments d'une infrastructure Fog Computing

1.3.2.1 Les noeuds de calcul d'une infrastructure Fog Computing

Définition

Un noeud Fog est un équipement possédant une capacité de calcul, de stockage et de communication. Il peut être physique ou virtuel (un groupement de noeuds physiques qui forment un seul bloc de calcul et de stockage grâce à différentes techniques de virtualisation).

Type d'équipements Fog

Les équipements de calcul considérés dans le Fog, qu'ils soient physiques ou virtuels, sont très hétérogènes aussi bien au niveau de leur profil de consommation énergétique que des capacités de calcul offertes et des technologies de communication qu'ils utilisent. Les noeuds peuvent à la fois posséder des spécificités de noeuds réseau i.e. faire communiquer d'autres équipements tout en

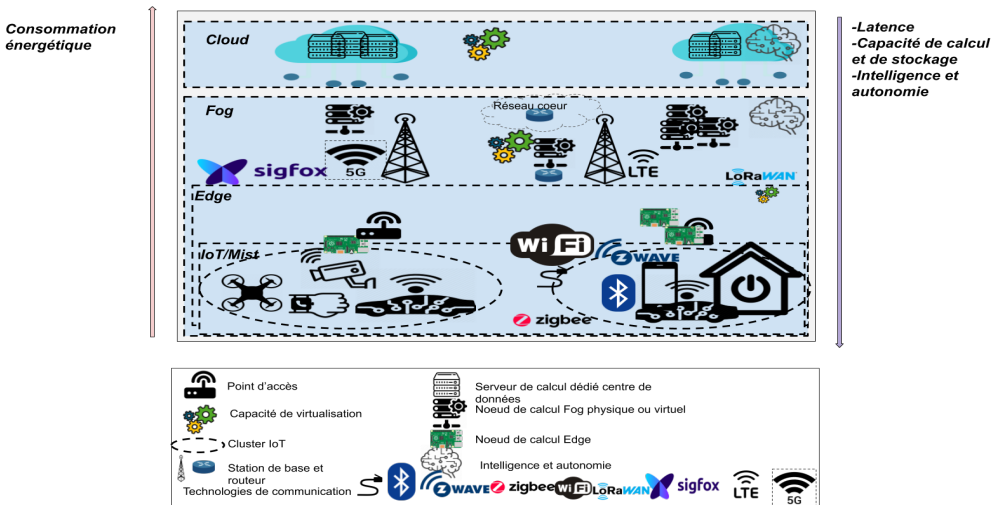


FIGURE 1.5 – Vue globale de l'architecture hiérarchique 3 tiers du Fog Computing.

ayant la capacité d'héberger des services et de faire du calcul. On trouve également des points d'accès qui sont uniquement chargés de l'acheminement de l'information. Voici dans ce qui suit une liste non exhaustive des équipements qu'on trouve actuellement dans des infrastructures Fog :

- Noeud physique : Passerelle, Raspberry, Micro-contrôleur, Noeud point d'accès, ordinateur personnel, autres objets ayant une capacité de calcul.
- Noeud virtuel : Machine virtuel, Container.
- Micro ou Nano centres de données regroupant des noeuds physiques et virtuels ayant des capacités de calculs réduites.

Déploiement d'un noeud Fog Selon le NIST [Michaela 2018a], il existe quatre types de déploiements possibles pour un noeud Fog (similaires à ceux du Cloud).

- Noeud Fog privé : noeud fourni pour un usage exclusif par une seule organisation comprenant plusieurs consommateurs (par exemple, des unités commerciales) Il peut être détenu, géré et exploité par l'organisation, un tiers ou une combinaison de ceux-ci, et il peut exister dans les locaux ou en dehors.
- Noeud Fog de communauté : est réservé à l'usage exclusif d'une communauté spécifique de consommateurs appartenant à des organisations qui partagent les mêmes préoccupations (par exemple, la mission, les exigences de sécurité, la politique et les considérations de conformité).
- Noeud Fog public : est mis à disposition du grand public pour une utilisation ouverte. Il peut être détenu, géré et exploité par une entreprise, une université, une organisation gouvernementale ou une combinaison

des deux.
— Noeud Fog hybride : est une agrégation de deux ou plusieurs noeuds Fog distincts (privés, communautaires ou publics) qui restent des entités uniques, mais qui sont liés entre eux par une technologie standardisée ou propriétaire qui permet la portabilité des données et des applications.

1.3.2.2 La communication dans une infrastructure Fog Computing

Le Fog est une infrastructure dynamique qui exploite des éléments déjà déployés pour proposer des services de calculs et de communications. Il peut ainsi utiliser toutes les technologies de communication déjà existantes. Cependant, des infrastructures comme le Fog sont majoritairement composées de technologies de communication sans fils aussi bien avec infrastructure (mobile) que sans infrastructure (Peer-to-Peer etc.). On inclut également les technologies des communications spécifiques aux réseaux de capteurs (WSN) ou d'une manière plus générique à l'Internet des Objets (IoT) que nous avons vu dans la section précédente 1.2.

1.3.3 Caractéristiques du Fog Computing

Le National Institute of Standards and Technology (NIST) et l'OpenFog consortium ont identifié un certain nombre de caractéristiques définissant les infrastructures Fog et que nous résumons dans la liste non exhaustive suivante :

— Distribution géographique
L'infrastructure Fog peut s'étendre sur une très large zone géographique à l'échelle d'une ville ou d'une région. Cette infrastructure va héberger des applications dont les composants peuvent être distribués dans n'importe quelle région.
— Hétérogénéité et hiérarchisation des noeuds
L'architecture Fog la plus répandue dans la littérature et qui est aussi la plus générique (également utilisée par le NIST) définit l'architecture Fog comme une architecture en couches 3 tiers. Ce modèle représente une hiérarchie entre les noeuds de calcul, allant des équipements utilisateurs jusqu'au centre de données en passant par des équipements Edge et Fog Computing. Le principe est que les capacités de calcul, de stockage et la consommation énergétique des noeuds augmentent en allant de la première couche de noeuds utilisateurs à la dernière représentant les centres de données. Cette architecture est représentée sur la figure 1.5
— Sensibilité à la mobilité et à la géolocalisation des utilisateurs
Le Fog est un environnement hétérogène et dynamique. Ceci est principalement dû à la mobilité des utilisateurs et des équipements à la périphérie du réseau. Il est donc important d'avoir les positions géogra-

phiques des utilisateurs et de connaître la portée des applications qu'ils utilisent.

— Sécurité
Le Fog Computing peut pallier aux failles de sécurités des réseaux de capteurs en offrant des noeuds intermédiaires avec des protocoles de communication plus sécurisés que ceux des objets connectés, que nous verrons dans la suite du chapitre. Le Fog permet également de gérer les services et applications au plus proche des utilisateurs. Les opérations de stockage de données, de filtrage et traitement ayant lieu au plus proche des utilisateurs évitent de faire transiter des informations sensibles vers les serveurs Cloud.

— Interopérabilité et fédération
Une infrastructure Fog étendue regroupe plusieurs domaines réseau et de calcul gérés par différents organismes. Ces domaines doivent coopérer pour assurer un déploiement fluide et une gestion transparente des services pour les utilisateurs.

— Autonomie et programmabilité
L'une des principales caractéristiques des infrastructures Fog est l'autonomie de décision pour la gestion du cycle de vie des applications qui y sont déployées. La facilité de programmabilité et de reconfiguration des équipements grâce à la virtualisation simplifie la gestion et assure l'évolutivité de l'infrastructure face à la dynamicité de l'environnement.

1.3.4 Concepts connexes au Fog : Edge et Mist computing

On trouve dans la littérature différents concepts liés et parfois à tort confondus avec le Fog Computing. Dans cette partie, nous allons définir ces termes afin de voir les différences et similarités avec le Fog Computing et éclaircir ce que nous considérons comme Fog tout au long de ce travail. La principale différence entre les concepts réside dans les équipements qui peuvent héberger du calcul et leur positionnement par rapport aux sources de données.

— Cloud Computing : Le Cloud est un paradigme de calcul se reposant exclusivement sur l'utilisation de centres de données pour héberger les calculs et les données utilisateurs. Le Fog computing est souvent vu comme l'évolution du Cloud vers un modèle plus distribué qui se rapproche des utilisateurs finaux et exploite des noeuds réseau pour le calcul utilisateur.

— Edge Computing : Même si dans certains travaux de la littérature, on trouve que le Edge et le Fog définissent le même concept, il y a trois principales différences qui sont identifiées par le NIST afin d'éviter la confusion des deux termes :
  — Le Fog computing considère aussi bien une infrastructure physique que virtuelle et multi-tiers, ce qui permet une flexibilité de reconfi-

Aspet	Cloud computing	Edge computing	Mist Computing	Fog computing
Opérateur	Fournisseur Cloud	Utilisateur	Utilisateur	Fournisseurs d'accès réseau et Cloud
Équipements élémentaires	Serveurs puissants	Passerelles IoT, Raspberry, smartphone	IoT microcontrôleur, smartphone	Points d'accès, équipements réseau, noeuds de calcul dédiés
Déploiement d'applications	Géré par le fournisseur Cloud	Utilisateurs	Programmé dans l'équipement	Demandé par l'utilisateur et gérer par différent fournisseurs.
Connectivité élémentaire	Internet	Bluetooth, ZigBee	Bluetooth, Wi-Fi	Wi-Fi, LTE
Consommation énergétique	Relativement élevée	Basse	Basse	Modérée
Disponibilité ressources	Élevée	Modérée	Basse	Modérée
Distance utilisateur	Grande	Petite	Très petite	Relativement petite
Organismes de standardisation	CSA, DMFT, NIST, OCC, GICTF	OpenEdge	-	OpenFog Consortium, IEEE,NIST

TABLE 1.3 – Différences ente le Cloud, le Fog , le Edge et Mist computing [Yousefpour 2019], [Mahmud 2020].

guration et une gestion intelligente et autonome de l'infrastructure qui peut être aussi bien distribuée, centralisée ou hybride. Le Edge est plus rigide et les applications peuvent rarement être déplacées d'un équipement à un autre et la communication entre utilisateur final et application est souvent directe.

- Le Edge est en général formé d'un nombre limité de noeuds peu performants en calcul.
- Le Fog est plus générique et peut inclure une partie de l'infrastructure qui a les spécificités du Edge computing au plus proche de l'utilisateur.

— Mist Computing : est un terme défini par le NIST [Michaela 2018a] et qu'on peut trouver dans différents travaux [Yousefpour 2019],[Amir M. Rahmani 2017],[Mahmud 2020]. Ce terme est utilisé pour parler de la couche de noeuds de calcul et les technologies de connexion qui sont au plus proche de l'utilisateur final et sont moins puissants que les noeuds Fog dédiés. On peut citer les Raspberry, Arduino ou différents types d'objets intelligents qui peuvent former ce qu'on appelle des nano centres de données.

On peut résumer les spécificités du Fog et son positionnement par rapport aux autres terminologies dans le tableau 1.3.

1.4 Les Avantages et les défis du Fog Computing pour supporter les applications IoT

Le Fog Computing a été pensé comme nouveau paradigme de calcul pour pallier les limites du Cloud Computing et répondre aux besoins des applications sensibles au délai et celles qui utilisent intensivement les ressources réseau.

Les infrastructures Fog Computing offrent les avantages suivants :

1. Réduction du trafic réseau et de la consommation énergétique des centres de données.
2. Amélioration de la sécurité en rapprochant le traitement et le stockage des données utilisateurs et en utilisant des méthodes de sécurité plus complexes qui ne peuvent pas être directement intégrées dans les équipements IoT.
3. Supporter le très grand nombre d'équipements IoT et leur distribution sur de larges zones géographiques par une structure hiérarchique, ce qui peut minimiser la sous-utilisation des ressources.
4. Améliorer la durée de vie des objets connectés en transférant leur charge vers des noeuds voisins (ou d'autres noeuds Fog ou Cloud).
5. Supporter des applications exigeantes en temps de réponse.
6. Une application IoT est liée à ses capteurs et ses actionneurs et ces derniers peuvent être éparpillés sur une large zone. La conscience de la localité du Fog est une caractéristique importante qui peut aider à améliorer la disponibilité et à la gestion efficace des échanges.

Le Fog semble être l'infrastructure idéale pour supporter les futures applications IoT. Cependant, il reste encore un certain nombre de verrous à lever pour qu'il puisse être efficacement utilisé. Voici une liste non exhaustive des défis apportés par l'utilisation du Fog en tant que paradigme de calcul.

1. Gérer efficacement les interactions entre les noeuds à la périphérie et les centres de données (offloading).
2. La découverte et l'estimation des ressources [Amir M. Rahmani 2017].
3. La problématique du placement et du déploiement des entités de gestion et de contrôle d'un environnement Fog.
4. La gestion du compromis entre efficacité d'utilisation des ressources de calculs et celle des ressources de communication.
5. Le problème du stockage distribué des données, leur persistance et leur intégrité.
6. L'absence d'un modèle économique pour les services offerts par les infrastructures Fog Computing.

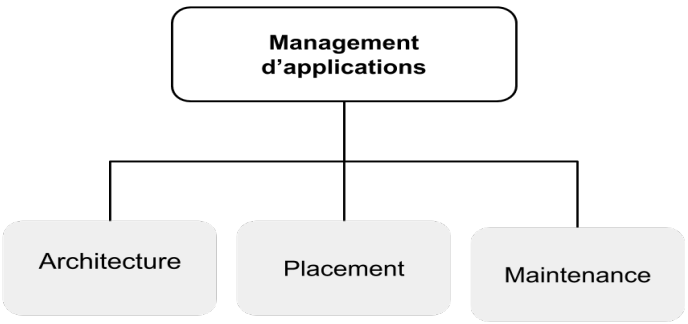


FIGURE 1.6 – Aspects de management d'application [Mahmud 2020]

7. Les contraintes spatiales pour le partage des ressources définies par la position des utilisateurs et la portée réseau des noeuds Fog.

1.4.1 Un environnement Fog-IoT

Un environnement Fog-IoT regroupe des infrastructures de calculs distribuées suivant le paradigme de calcul du Fog Computing et des noeuds terminaux utilisateurs de type objets IoT. Un environnement Fog-IoT permet de déployer des applications de type IoT. Il regroupe à la fois les noeuds des centres des données, les noeuds Fog et les noeuds IoT.

1.5 Gestion des applications IoT dans les environnements Fog-IoT

Le processus de gestion d'applications peut être décomposé en trois grandes étapes, représentées en figure 1.6.

La première étape, l'architecture logicielle, permet de définir les spécificités de l'application, les technologies qu'elle utilise et les modules qui la composent.

La deuxième étape, le placement, permet de trouver un emplacement physique aux services composants l'application sur l'infrastructure de calcul tout en respectant d'éventuelles contraintes et des objectifs donnés.

La dernière étape, la maintenance, gère le cycle de vie de l'application sur l'infrastructure et permet d'effectuer des ajustements en fonction de l'état du système et des besoins utilisateurs.

Dans ce travail, nous nous intéressons à l'étape du placement d'applications en environnement statique avec le chapitre 3 et en environnement dynamique avec le chapitre 4.

Le placement de services est un cas particulier des problèmes d'ordonnancement de tâches, une problématique très étudiée par la communauté des

systèmes distribués et qui a suscité d'autant plus d'intérêt avec l'apparition du Cloud Computing et son utilisation en tant que principal paradigme de calcul. Il a été démontré à plusieurs reprises qu'une mauvaise politique de gestion des ressources physiques et virtuelles du Cloud a un impacte négatif sur le plan économique, énergétique et qualitatif de ces systèmes [Arya 2014], [Fakhfakh 2014].

Cette section donne un bref aperçu du problème du placement de services et ses différentes variantes dans le Cloud et par la suite l'extension de ce problème aux environnements distribués Fog-IoT qui apporte de nouveaux défis à considérer et qui ont été cités dans la section 1.4.

On trouve plusieurs termes qui font référence au même problème : placement de services, placement de composants, d'opérateurs ou de machines virtuelles, etc.

Dans ce travail nous utilisons le terme générique "service" qui peut faire référence à n'importe quelle entité virtuelle : machines virtuelles (VMs), containers ou "bundle OSGi" .

Un service est une unité indépendante virtuelle avec une capacité de calcul, de communication et de stockage qui interagit avec d'autres services et/ou avec les équipements utilisateurs pour effectuer une tâche donnée.

1.5.1 Les flux de services (workflow)

L'entrée d'algorithmes de planification de tâches (ici services) est généralement un modèle de workflow qui définit les services et leur inter-dépendance sans spécifier l'emplacement physique des ressources sur lesquelles les services sont exécutées. Il existe deux types de workflow : déterministe et non déterministe. Dans un modèle déterministe, les dépendances des tâches et des données d'E/S sont connues à l'avance, tandis que dans un modèle non déterministe, elles ne sont connues qu'au moment de l'exécution [Adhikari 2019]. Le flux de tâches est la représentation abstraite d'une application qui peut être classée selon le domaine ou le type de données en workflow scientifique, business ou synthétique [Adhikari 2019]. Considérant la dépendance entre les tâches d'un workflow, on peut avoir des workflow séquentiels, parallèles ou une combinaison des deux.

Le modèle d'application peut être classé en trois catégories : flux de travail unique, flux multiple ou flux ensembliste.

— Flux de travail unique : Il s'agit d'un modèle d'application traditionnel. Les tâches du workflow sont exécutées de manière séquentielle et indépendante sur les serveurs.

— Flux multiples : l'application de cette catégorie est composée de plusieurs types de workflows scientifiques. Cependant, les instances de workflow peuvent ne pas avoir la même propriété et elles peuvent varier en fonction de leur structure, de leur taille et de leurs besoins en res-

sources. Pour ce type d'application, le nombre et le type des workflows varient dynamiquement. Ainsi, l'ordonnanceur doit utiliser l'algorithme dynamique pour déployer les tâches vers les ressources appropriées sur le serveur. Chacun des workflows d'une application peut avoir des exigences de QoS différentes et s'exécute indépendamment sur les serveurs. Ces types de workflows sont composés d'un certain nombre de tâches dépendantes et chaque tâche doit être assignée à une instance de machine physique ou virtuelle appropriée.

— Ensembles de flux : les applications de cette catégorie sont également composées de plusieurs workflows scientifiques dont l'effet combiné produit la sortie souhaitée d'une application. Cependant, les workflows des applications sont interdépendants et s'exécutent indépendamment sur les serveurs. Les workflows dans les ensembles d'une application ont une structure similaire avec différentes tailles de tâches. Chacun des workflows de l'application planifie les ressources appropriées sur les serveurs. Par exemple, les applications vidéo doivent regrouper une seule grande image en mosaïque de l'ensemble du ciel en combinant différents types de mosaïques de différentes régions du ciel.

1.5.1.1 Type d'ordonnancements de services

L'ordonnancement de services peut se faire avec ou sans contraintes et peut être statique ou dynamique [Fakhfakh 2014].

— Ordonnancement statique (hors ligne) : ce type d'algorithme est basé sur la création d'un plan de planification avant d'exécuter des tâches de workflow. Ainsi, il considère l'état initial de la plateforme Cloud. Des algorithmes statiques peuvent être prévus pour planifier un ou plusieurs workflows.

— Ordonnancement dynamique (en ligne) : les algorithmes dynamiques consistent à provisionner des ressources et à planifier des tâches à l'exécution pour prendre en compte l'aspect dynamique du Cloud. Ils recalculent périodiquement les solutions afin d'optimiser le coût de calcul en fonction des conditions et ressources actuelles du réseau et mettent à jour certains paramètres tels que les coûts de communication.

1.5.2 Objectifs et finalités d'un ordonnancement de tâches

Un ordonnancement est guidé par un ou plusieurs objectifs à atteindre. Les éventuelles solutions trouvées sont comparées entre elles selon un ou plusieurs indicateurs numériques représentant les objectifs à atteindre. Ci-dessous se trouve une liste non exhaustive des catégories d'objectifs qu'on trouve dans la littérature.

— La Qualité de service (QoS) : qui peut être représentée par plusieurs métriques selon les applications et le contexte ainsi que les besoins utilisateurs ou besoins système. La métrique du temps de réponse et l'un des indicateurs les plus utilisés pour parler de qualité de service d'une application. D'autres critères comme la disponibilité, la robustesse, la scalabilité sont des critères fonctionnels de la qualité de service.
— L'utilisation des ressources (UR)(efficacité) : permet d'identifier si certaines ressources sont sous-utilisées ou sur-utilisées. Ceci permet d'établir des politiques d'équilibrage de charge de travail entre les équipements et ainsi rentabiliser toute l'infrastructure. Les métriques pour cette catégorie sont souvent la charge CPU d'un équipement comparé à la charge moyenne de l'infrastructure ou le nombre de tâches traitées par seconde.
— L'énergie : Comme nous avons pu le voir précédemment l'énergie est un élément important dans l'ICT. Les métriques liées à cet objectif sont en général l'efficacité énergétique des équipements (énergie utile/ énergie consommée), l'énergie totale, et la puissance moyenne consommée.
— Coût financier (CF) : le coût financier d'un placement dans le Cloud varie selon le fournisseur.
— Coût opérationnel (CO) : les coûts de la mise en place et de l'utilisation de la politique d'ordonnancement peuvent être définis par une métrique temporelle, par l'utilisation de ressource ou encore l'énergie.

1.5.3 Le placement de services vu comme un problème d'optimisation combinatoire

Le placement de services dans le Cloud ou le Fog est principalement modélisé et traité comme un problème d'optimisation. Il existe plusieurs approches pour résoudre ce type de problèmes présentées par la Figure 1.7. On distingue deux grandes familles de problèmes d'optimisation : (1) les problèmes à variables continues et (2) les problèmes à variables discrètes (problème combinatoire). Le placement de services fait partie des problèmes d'optimisation combinatoire.

Un problème d'optimisation combinatoire consiste à trouver dans un ensemble discret de solutions, une ou plusieurs solutions parmi les meilleurs solutions réalisables du problème. La notion de meilleure solution étant définie par la valeur d'une ou plusieurs fonctions objectifs.

La résolution d'un problème d'optimisation combinatoire consiste à trouver une solution optimale dans un ensemble discret et fini de possibilités. En théorie la résolution du problème revient à tester toutes les solutions réalisables, et de comparer leurs qualités pour identifier la meilleure. Cependant, en pratique, l'énumération de toutes les solutions peut prendre un temps considérablement long. La théorie de la complexité donne des outils pour mesurer

ce temps de recherche. De plus, comme l'ensemble des solutions réalisables est défini de manière implicite, il est aussi parfois très difficile de trouver ne serait-ce qu'une solution réalisable.

Quelques problèmes d'optimisation combinatoire peuvent être résolus de manière exacte en un temps polynomial, par exemple par un algorithme glouton, un algorithme de programmation dynamique ou en montrant que le problème peut être formulé comme un problème d'optimisation linéaire en variables réelles.

Dans la plupart des cas, le problème est NP-difficile et, pour le résoudre, il faut faire appel à des algorithmes de branch and bound, à l'optimisation linéaire en nombres entiers ou encore à la programmation par contraintes. En pratique, la complexité acceptable n'est souvent que polynomiale. On se contente alors d'avoir une solution approchée au mieux, obtenue par une heuristique ou une métaheuristique. Pour certains problèmes, on peut prouver une garantie de performance, c'est-à-dire que pour une méthode donnée l'écart entre la solution obtenue et la solution optimale est borné. Dans ces conditions, à domaine égal, un algorithme est préférable à un autre si, à garantie égale ou supérieure, il est moins complexe.

Classe et complexité d'un problème Tout problème d'optimisation peut se ramener à un problème de décidabilité. Un problème est dit décidable s'il existe un algorithme qui le résout en un temps fini. Si aucun algorithme ne peut le résoudre en un temps fini, ce dernier est dit indécidable. La théorie de la complexité [Wegener 2005] permet d'estimer le temps et les ressources de calcul nécessaires à la résolution d'un problème. Les problèmes dits décidables se décomposent en trois classes de complexité temporel :

- **Classe P** Un problème est dit polynomial, s'il existe un algorithme de complexité polynomiale capable de le résoudre. Ainsi l'ensemble des problèmes polynomiaux forme le class P.
- **Classe NP** Un problème P est dans NP si pour toute instance de ce problème. Il est résolvable en temps polynomial. Cette classe rassemble la plus part des problème d'optimisation combinatoire.
- **Classe NP-Complet** Un problème NP-complet est un problème NP qui domine tout les autres. Aucun algorithme peut le résoudre en temps polynomial. Un problème P domine un problème P' (P' est réductible polynomialement en P) si :
  - si on peut transformer toute instance de P en une instance de P' grâce à un algorithme polynomial.
  - I est solution de P si est seulement si I' est solution de P'.

Dans ce qui suit nous allons brièvement décrire les méthodes heuristiques et métaheuristiques qui sont les catégories d'approches les plus utilisées dans la littératures. Les résolutions exactes sont quant à elles souvent non applicables

dans des systèmes réels.

(1) Approche heuristique pour le placement de services

Il existe quatre classes d'heuristiques de planification pour les applications de workflow, à savoir la planification de tâches individuelles, la planification de listes, la planification basée sur les clusters et la duplication [Fakhfakh 2014].

Les types d'algorithmes que nous allons définir dans cette partie font partie des approches de résolution gloutonnes (Greedy Algorithms).

Les algorithmes Glouton font partie des schémas de résolutions les plus simples et les plus rapides. Les solutions sont construites de manière itérative sans jamais être remises en question. À une itération donnée, on détermine l'élément à inclure dans la solution partielle en évaluant le coût de la solution construite. L'élément engendrant la plus petite augmentation de coût est choisi. Cette approche ne permet pas de connaître le coût ou l'impacte à long terme du choix d'un élément à une itération donnée. De plus, les contraintes du problème peuvent faire échouer l'algorithme si à un itération donnée aucun élément possible ne donne une solution valide. Les algorithmes gloutons ou itératifs connus aussi sous le nom de méthodes de descentes ou méthodes classiques peuvent facilement se retrouver piégés dans un optimum local, qui constitue la meilleure solution accessible compte tenu de l'hypothèse initiale. Pour améliorer l'efficacité de la méthode, on peut appliquer l'algorithme à plusieurs reprises avec des conditions initiales différentes et choisir la meilleure solution obtenue. Cette procédure augmente considérablement le temps de calcul de l'algorithme sans garantir l'obtention de la configuration optimale. L'application d'une méthode itérative est particulièrement inefficace lorsque le nombre d'optimum locaux croit exponentiellement avec la taille du problème.

— Algorithme pour un service
  La planification de tâches individuelles est la méthode de planification la plus simple pour planifier des applications de workflow et elle prend une décision de planification basée uniquement sur une seule tâche
— Algorithme de listes
  Une heuristique de planification de liste hiérarchise les tâches de workflow et organise les tâches en fonction de leurs priorités. Il y a deux phases principales dans une heuristique de planification de liste, la phase de priorisation de la tâche et la phase de sélection des ressources.
— Algorithmes de clusterisation et de duplication
  La planification basée sur les clusters et la planification basée sur la duplication sont conçues pour éviter le temps de transmission des résultats entre les tâches interdépendantes vis à vis des données, de manière à réduire le temps d'exécution global. La planification basée sur le cluster regroupe les tâches et affecte les tâches du même cluster à la même ressource, tandis que la planification basée sur la duplication utilise

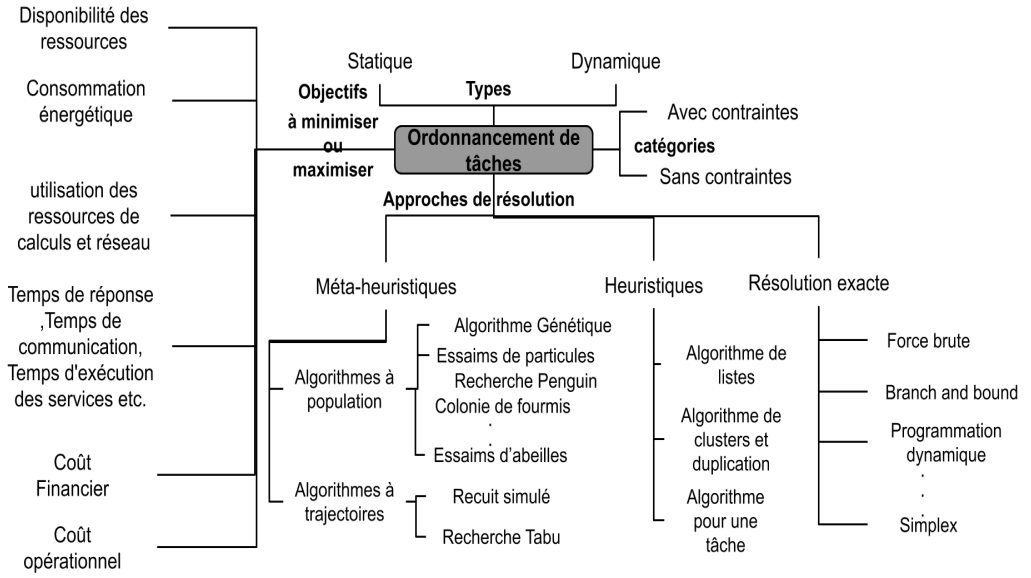


FIGURE 1.7 – Ordonnancement de tâches informatique [Dorsemaine 2015]

le temps d'inactivité d'une ressource pour dupliquer certaines tâches parentes, qui sont également planifiées sur d'autres ressources.

(2) Approche Meta-heuristique pour le placement de services

Le placement de services dans le Cloud ou le Fog fait parti des problèmes dits NP-difficiles [Adhikari 2019]. Pour sa résolution, on a souvent recourt à des heuristiques conçues et adaptées pour un problème particulier ou des métaheuristiques, des méthodes génériques qui peuvent s'appliquer à tous les problèmes d'optimisation et s'adapter à leur dynamicité. Ces dernières années, diverses méthodes efficaces, évolutionnaires et basées sur des stratégies de coopération, ont été proposées pour résoudre les problèmes NP-difficiles dans lesquels, aucune solution en temps polynomial ne peut être trouvée. On résume les différents aspects d'un ordonnancement de tâches avec le schéma présenté en figure 1.7.

Une métaheuristique désigne un algorithme qui résout un problème d'optimisation sans garantie d'optimalité, dans des temps de calcul acceptables. Les approches métaheuristiques utilisent des stratégies génériques indépendantes du problème étudié et qui guident la recherche dans l'espace de ses solutions. Elles sont de ce fait applicables à tout problème d'optimisation.

Les métaheuristiques sont inspirées par des analogies avec la physique (recuit simulé), avec la biologies (algorithmes évolutionnaires, recherche tabou) ou encore avec l'éthologie (colonie de fourmi, essaim particulairs, essaim d'abeilles) [DREO 2003]. Les approches métaheuristiques ont en commun un certains nombre de caractéristiques :

Efficacité des métaheuristiques

— Capacité des métaheuristique à s'extraire du minimum local :
Pour surmonter la problématique du minimum local, les métaheuristiques autorisent de temps à autre une dégradation temporaire de la solution. Ce processus est appelé mouvement de remontée qui est la base des méthodes dites de voisinage (recherche tabou et recuit simulé). En fonction de chaque méthode, un mécanisme de gestion des dégradations de la solution est appliqué pour éviter la divergence de la recherche. Les stratégies à population ou dite distribuées comme les algorithmes évolutionnaires ont également des mécanismes de contrôle de la dégradation.

— Applicabilité à tous les problèmes d'optimisation et adaptabilité aux changement :
Les métaheuristiques s'appliquent aussi bien aux problèmes d'optimisation continus que discrets, mono-objectif et multi-objectifs où il s'agit d'optimiser simultanément plusieurs objectifs. Les métaheuristiques peuvent également être utilisées pour de l'optimisation multimodales, où on essaye de repérer tout un jeu d'optimum globaux et locaux et pour la résolution de problèmes dynamiques avec une ou plusieurs fonctions objectives variant cours du temps.

— Nature stochastique :
Les métaheuristiques sont en partie stochastiques, ce qui leur permet de faire face à l'explosion combinatoire des possibilités.

— Coopération et hybridation :
Les méthodes ne s'excluent pas entre elles et peuvent être combinées pour résoudre un problème donné.

Inconvénients des métaheuristiques

— Calibration des paramètres :
Toutes les métaheuristiques ont au moins un paramètre d'entrée et au moins un critère de fin qu'il faut déterminer. Souvent, le choix des valeurs des paramètres se fait expérimentalement. Il existe cependant des résultats théoriques pour certaines méthodes.

— Initialisation (Solutions de départ) :
L'initialisation des solutions de départ est une étape commune à toutes les métaheuristiques et cette dernière impacte les futures régions explorées. Plus les solutions initiales sont bonnes, plus la métaheuristique aura de chance de visiter des régions de l'espace des solutions intéressantes et donc de converger rapidement vers une bonne solution. Souvent, l'initialisation se fait uniformément sur l'espace de recherche ou en utilisant une heuristique gloutonne.

— Compromis entre diversification et intensification :
Il est souvent difficile de trouver le bon compromis entre l'exploration de

l'espace (diversification) qui vise à récolter l'information sur le problème optimisé et l'exploitation des solutions (intensification) visant à utiliser l'information récoltée pour définir les zones à parcourir. Le compromis entre ces deux aspects se fait en partie grâce aux choix des paramètres de chaque métaheuristique.

— Temps de résolution et consommation de ressources :
Contrairement aux algorithmes gloutons, les métaheuristiques à population consomment plus de ressources de calculs, de mémoire et ont des temps d'exécution plus grands.

Métaheuristiques évolutionnaires

Les algorithmes évolutionnaires (AEs) sont des méthodes de recherche apparues à la fin des années 1950 s'inspirant de l'évolution des espèces en biologie. Elles connaissent un grand succès depuis le développement considérable des puissances de calcul et des capacités de stockage des machines. Un algorithme évolutionnaire manipule un ensemble de solutions appelé population d'individus. Un individu représente une solution du problème. Le principe d'une stratégie évolutionnaire est de faire évoluer cette population d'individus grâce à des opérateurs de reproduction et de sélection. Un opérateur de reproduction permet la recombinaison et la variation des caractères héréditaires des parents pour créer des nouveaux individus. La sélection permet la survie des individus les plus performants. Des opérations de perturbations sont introduites (mutation) pour assurer l'aspect exploratoire de la stratégie et introduire une dimension stochastique au processus de recherche. L'algorithme Génétique (GA) est la méthode la plus connue de cette famille de métaheuristiques.

Métaheuristiques de trajectoire

Les métaheuristique de trajectoire sont des méthodes à solution unique. Les méthodes à solution unique commencent par une solution initiale et s'en éloigne progressivement en suivant une certaine trajectoire dans l'espace de recherche. Les méthodes de trajectoire englobent essentiellement la méthode de descente, la méthode du recuit simulé, la recherche tabou, la méthode GRASP, la recherche à voisinage variable, la recherche locale itérée et leurs variantes.

Métaheuristiques à intelligence par essaim

L'intelligence en essaim (Swarm Intelligence) est inspirée des phénomènes biologiques rencontrés en éthologie [Bonabeau 1999]. Cette catégorie de métaheuristiques recouvre un ensemble d'algorithmes, à base de population d'agents simples (entités capables d'exécuter certaines opérations), qui interagissent localement les uns avec les autres et avec leur environnement. Ces entités, ont généralement une capacité individuelle très limitée mais peuvent coopérer pour effectuer des tâches plus complexes et nécessaires à leur survie.

Bien qu'il n'y ait pas de structure de contrôle centralisée qui dicte la façon dont les agents individuels devraient se comporter, les interactions locales entre les agents conduisent souvent à l'émergence d'un comportement collectif global et auto-organisée. Deux exemples phares d'algorithmes de l'intelligence en essaim sont les algorithmes de colonies de fourmis et les algorithmes d'optimisation par essaim particulaire. D'autres algorithmes d'optimisation qui proviennent d'analogies avec des phénomènes biologiques naturels ont été proposés. Parmi les plus significatifs d'entre eux figurent : l'algorithme Bacterial foraging optimization [Liu 2002] qui s'inspire du comportement de recherche de nourriture et de reproduction des bactéries, l'optimisation par colonie d'abeilles (Bee Colony Optimization) [Karaboga 2007] les systèmes immunitaires artificiels [Farmer 1986], l'algorithme Penguin [Gheraibia 2013], et l'algorithme d'optimisation basée sur la biogéographie insulaire [Simon 2009].

1.6 La consommation énergétique dans les environnements Fog-IoT

Dans cette partie nous allons voir dans un premier temps et à travers différentes études l'impacte du Cloud, des réseaux et de l'IoT sur la consommation énergétique. Dans un deuxième temps, nous allons aborder plus en détails les efforts et les approches pour estimer la consommation énergétique des environnement Fog-IoT.

1.6.1 La consommation énergétique du secteur des technologies de l'information et des communications (ICT)

En considérant uniquement des infrastructures Cloud, le secteur des Technologies de l'information et de la Communication (ICT) consommerait un peu plus d'un cinquième de l'électricité mondiale selon [tsu 2017]. Le problème énergétique de l'ICT est d'autant plus important avec l'avènement de l'IoT et l'utilisation du Fog comme paradigme de calcul. Selon l'étude [eno] faite en 2014, les 14 milliards d'objets connectés déjà déployés à travers le monde consomment 400 TWh (terawatt-heure) par an. Les auteurs estiment que ce chiffre pourrait atteindre les 1400 TWh après le déploiement des 50 milliards d'objets prévu pour fin 2020.

La part des ICT dans les émissions mondiales de gaz à effet de serre a augmenté de moitié depuis 2013, passant de 2.5% à 3.7% des émissions mondiales. Dans les pays de l'OCDE, les émissions de CO2 des TIC ont augmenté d'environ 450 millions de tonnes depuis 2013. Sur la même période, les émissions globales de CO2 de l'OCDE ont diminué de 250 millions de tonnes [Ferrboeuf 2019]. L'expansion rapide des TIC entraîne une augmentation rapide de leur empreinte énergétique directe. Cette empreinte comprend l'éner-

gie utilisée pour la production et l'utilisation des équipements TIC (serveurs, réseaux, terminaux). Cette dernière a augmenté de 9% par an. Par rapport à 2010, la consommation directe d'énergie générée par 1 euro investi dans les technologies numériques a augmenté de 37%. L'intensité énergétique du secteur des TIC augmente de 4% par an, ce qui contraste fortement avec la tendance de l'évolution de l'intensité énergétique du Produit Intérieur Brute (PIB) mondial, qui diminue de 1.8% par an. L'explosion des utilisations de la vidéo (Skype, Streaming, etc.) et la consommation accrue d'équipements numériques fréquemment renouvelés sont les principaux moteurs de cette inflation.

Selon l'étude de l'IEA [IEA 2019], les environnements IoT sont à l'origine d'une génération de données importante (phénomène du Big Data). Le trafic de données internet a triplé entre 2015 et 2019 et suit une augmentation exponentielle. 90% des données aujourd'hui ont été générées entre 2018 et 2020. Des études estiment que le trafic internet atteindra les 4.2 zettabytes en 2022 (cisco 2915, 2018, 2019,iea).

Le nombre d'utilisateurs internet mobiles a atteint les 3.6 milliards en 2018 et devrait atteindre les 5 milliards d'utilisateurs aux alentours de 2025. Le nombre d'équipements Internet of Things a atteint les 7.5 milliards en 2018 et on estime qu'il atteindra les 25 milliards en 2025 (GSM 2019) pour chaque bit de données transitant sur le réseau depuis le centre de données jusqu'aux utilisateurs finaux, 5 autres bits de données sont transmis dans et entre les centres de données (Cisco, 2016). La nature de la transmission de données évolue rapidement et le trafic provenant des appareils sans fils et mobiles va représenter plus de 70 % du trafic IP total d'ici 2022, contre environ la moitié en 2018 [Jalali 2017].

Concernant le cas de la France l'étude faite par [dat] montre que le volume de données transitant par les réseaux mobiles continue d'augmenter et arrive à une augmentation d'environ 45% en une année, soit 1,5 exaoctets de données pour le 4e semestre de 2019.

1.6.1.1 Les centres de données (Le Cloud)

Les auteurs de [Gre 2020] estiment que le secteur informatique est responsable de 7% de la consommation électrique mondiale. Ils montrent que les centres de données produisent 25% de gaz à effet de serre. En raison de l'impact énergétique non négligeable des centres de données, de nombreux travaux ont abordé le problème de la minimisation de la consommation d'énergie dans le Cloud [Baliga 2011], [Kumar 2010], [Beloglazov 2010]. Dans un document de 2012, "How Clean is Your Cloud", Greenpeace propose cette comparaison : "Si le Cloud était un pays, il aurait la cinquième plus grande demande d'énergie au monde". La demande mondiale d'électricité des centres de données en 2018 était estimée à 198 TWh, soit près de 1% de la demande finale

mondiale d'électricité [Shehabi 2018]. Sur la base des tendances actuelles en matière d'efficacité du matériel et de l'infrastructure des centres de données, la demande mondiale en énergie des centres de données devrait diminuer légèrement pour atteindre 191 TWh en 2021 (Cisco, 2018 ; Masanet et al., 2018 ; Shehabi et al., 2016). Ceci malgré une augmentation prévue de 80% du trafic des centres de données et de 50% de la charge de travail des centres de données au cours des trois prochaines années (Cisco, 2018). Le document de 2014, intitulé "Data Center Efficiency Assessment", du National Resources Defense Council (NRDC) [NRD 2014] indique que les 2 millions de serveurs informatiques dans près de 3 millions de centres de données utilisés aux États-Unis pour les activités en ligne consomment suffisamment d'électricité pour alimenter tous les foyers de New York pendant 2 ans".

Bien que des plans soient en cours pour construire les futurs centres de données dans des endroits où des sources d'énergie plus propres sont disponibles, ces derniers consomment déjà de très grandes quantités d'énergie, et cette consommation ne fera qu'augmenter avec l'IoT.

1.6.1.2 Les réseaux de transmission

Les réseaux de données ont consommé environ 260 TWh dans le monde en 2018, soit environ 1.1 % de la demande mondiale totale d'électricité, les réseaux mobiles représentant les deux tiers. À court terme, l'éventail des résultats énergétiques possibles est large, dépendant en grande partie de la croissance de la demande de données et du rythme des nouvelles améliorations de l'efficacité énergétique de ressources de calcul. Dans un scénario d'amélioration modérée de l'efficacité des serveurs de calcul de 10 % par an (estimation prudente basée sur les améliorations historiques), la demande d'électricité pourrait augmenter de près de 10% pour atteindre environ 280 TWh en 2021. En revanche, dans un scénario d'amélioration du rendement élevé de 20 % par an (basé sur les taux obtenus dans des réseaux bien gérés et à forte utilisation de capacité), la demande pourrait baisser de 25 % pour atteindre environ 190 TWh.

Cet éventail de résultats potentiels souligne l'importance de la politique de placement de services pour réaliser de nouveaux gains d'efficacité.

Selon [IEA 2017], une connexion câblée est le moyen le plus économe en énergie pour communiquer numériquement. Si la connexion se fait par un réseau cellulaire, la consommation d'énergie augmente considérablement. Une étude publiée en 2015 par le Centre for Energy Efficient Télécommunications [Decker 2015] montre qu'en 2015 l'utilisation du Cloud à travers les réseaux non filaires a consommé 43 TWh contre seulement 9.2 TWh en 2012, soit une augmentation de 460%. Cela représente une augmentation de l'empreinte carbone de 6 mégatonnes de CO2 en 2012 à 30 mégatonnes de CO2 en 2015, soit l'équivalent de l'ajout de 4,9 millions de voitures sur les routes. Jusqu'à

90% de cette consommation est imputable aux technologies des réseaux d'accès sans fil, les centres de données ne représentant que 9 %.

1.6.1.3 Les terminaux : IoT, Mobile

Un rapport de l'Agence internationale de l'énergie (AIE) [IAE 2016] estime pour 2025 une consommation énergétique annuelle de 46 TWh pour 5 types d'applications IoT : domotique, éclairage intelligent, éclairage public intelligent, routes intelligentes et appareils intelligents. L'impact environnemental de ces infrastructures est également important. Selon Peter Corcoran, membre IEEE, l'énergie est l'un des trois principaux déterminants de la "durabilité à long terme de l'Internet des objets" (les deux autres étant la vie privée et la cybersécurité).

Les auteurs de [Hazas 2016] notent que bien que de nombreuses communications machine to machine (m2m) consomment très peu d'énergie, certaines d'entre elles comme les communications entre voitures autonomes, requièrent des quantités d'énergie démesurées, tant à la source que dans le Cloud et s'ajoute à cela le coût de la fabrication. Les auteurs expliquent en outre que si chaque objet individuellement ne consomme pas beaucoup d'énergie, le simple volume des transmissions pourrait placer les communications m2m en tête des concurrents en matière de consommation d'énergie. Certains prédisent que les communications m2m vont représenter 45% du trafic Internet d'ici 2022.

Pour démontrer le risque d'une augmentation substantielle de la consommation d'énergie avec la prolifération prévue de l'IoT, l'Agence internationale de l'énergie (AIE) a publié un rapport orienté sur les terminaux utilisateurs à haut rendement énergétique (4E) [IEA 2019]. Le rapport a estimé que la consommation annuelle mondiale d'énergie en mode veille de certains équipements IoT pourrait atteindre les 46 TWh d'ici 2025. Le rapport a en outre fourni des lignes directrices pour la sélection des technologies ou protocoles de communication (Bluetooth Smart, ZigBee, Wi-Fi etc.) si l'on veut réduire au minimum l'énergie en veille.

1.6.2 Calcul et estimation de la consommation énergétique dans les environnement ICT

On trouve dans la littérature de nombreux travaux qui traitent de la consommation d'énergie des services dans le Cloud [Kliazovich 2015], [Khosravi 2013], [Khosravi 2017], [Thiam 2019]. Cependant, comme la combinaison de l'IoT et du Fog est relativement nouvelle, il y a peu de travaux sur la consommation d'énergie de ces infrastructures.

Dans cette section, nous abordons d'abord la consommation d'énergie des terminaux IoT et leurs réseaux de support. Ensuite, nous discuterons de l'impact de plusieurs paramètres du système Fog-IoT sur l'énergie [Jalali 2017].

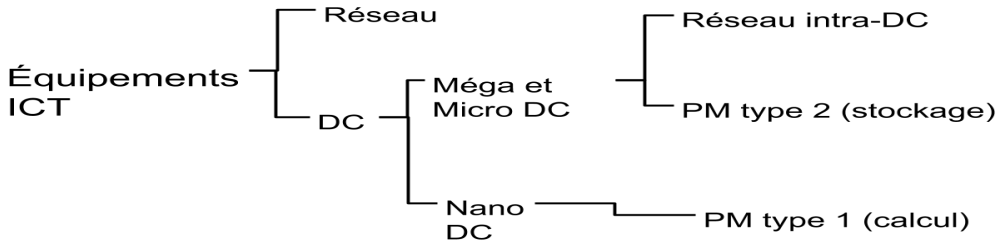


FIGURE 1.8 – Catégories d'équipements ICT

[Ahvar 2019] propose dans un premier temps une taxonomie des différentes déclinaisons d'architecture ICT, de la plus centralisée à la plus décentralisée d'entre elles, en passant par le Fog et le Edge computing. Dans un deuxième temps, les auteurs proposent un modèle générique pour le calcul de la consommation énergétique de chacune des architectures identifiées. Les auteurs ont montré qu'une architecture purement distribuée qui n'utilise pas les centre de données et leurs systèmes de refroidissement, consomme entre 14% et 25% d'énergie en moins par rapport à un système entièrement centralisé et partiellement distribué.

Les architectures proposées sont : "Complètement centralisée (Cloud)", " Partiellement centralisée (Fog/Edge)", "Distribuée avec contrôle centralisé (Fog/Edge)" et "Complètement distribuée (P2P)". Les équipements ICT, présentés en figure 1.8, qui composent les différentes architectures sont divisés en noeuds centre de données (DC) et équipements réseau. Ils proposent trois type de centre de données en fonction du nombre de machine physiques (PMs) : méga-DC si le nombre de machines physique dépasse les 10000, micro-DC, si le nombre de machines physiques est inférieur à 500 et nano-DC avec une seule machine physique. Les machines physiques ont été divisées en deux catégories : machines dédié au calculs ou (PM de type 1 ou CN) et machines de stockage (PM de type 2). Chaque type présente un profil énergétique différent. Les équipements réseau se divisent en noeuds routeurs, switch et liens réseau.

Les auteurs proposent également un modèle générique pour estimer la consommation d'énergie de l'infrastructure qui comprend les centres de données (méga, micro et nano) et le réseau de communication entre les centres de données et les utilisateurs sans inclure les terminaux utilisateurs.

Les entrées du modèle considèrent un certain nombre d'utilisateurs finaux qui utilisent des machines virtuelles actives pendant une certaine période de temps T définie en seconde. La topologie réseau considérée et en "fat tree".

Le modèle proposé divise pour chaque type d'équipement ICT la consommation d'énergie en parties statiques et dynamiques. La consommation d'énergie statique est la consommation d'énergie sans tenir compte de la charge de

travail (c'est-à-dire que les ressources sont inutilisées). Le coût dynamique est calculé sur la base de l'utilisation actuelle des ressources du Cloud par les machines virtuelles actives.

L'énergie consommée en statique est estimée comme la somme de la partie statique réseau et de celle de chaque centre de données (DC) incluant l'énergie consommée pour le refroidissement (exprimée en Power Usage Effectiveness (PUE)) le tout étant multiplié par le période T.

Dans de nombreux travaux, les équations d'estimation de la consommation énergétique de machines physiques des DCs sont linéaires en fonction de leur utilisation CPU.

Pour la partie dynamique des réseaux intra-DC, la consommation énergétique dynamique de la partie réseau dépend du nombre moyen de paquets traités par l'équipement (routeur ou switch), l'énergie pour traiter un paquet et l'énergie pour sauvegarder et transférer un octet. Les deux derniers paramètres varient en fonction du type de l'équipement.

1.6.3 La minimisation de la consommation énergétique dans des environnements Fog-IoT

Étant données les différentes études prévoyant l'importante augmentation de la consommation énergétique du secteur ICT, principalement due aux environnements Fog et IoT, de nombreux travaux proposent des solutions pour minimiser cette consommation et réduire l'impact énergétique dans Le Fog et L'IoT. Dans ce qui suit nous allons résumer ces travaux. La réduction de la consommation énergétique des infrastructures de calcul peut être et a été adressée de différentes manières dans la littérature. Certaines techniques comme l'ordonnancement de tâches restent applicables au Fog. d'autres, comme la consolidation de charge avec arrêt des machines ne peuvent être utilisées que dans les centres de données.

1.6.3.1 Aspects à considérer et approches utilisées pour minimiser la consommation énergétique dans les environnements Fog

[Jalali 2017] identifie un ensemble paramètres à considérer pour pouvoir agir et minimiser la consommation des infrastructures Fog.

— Technologies du réseau d'accès :
Nous avons pu voir dans les sections précédentes l'hétérogénéité des technologies de communications et leurs propriétés. Certaines technologies sont plus adaptées pour certains équipements et certaines applications. A titre d'exemple les applications streaming ont de meilleurs performances via les réseaux cellulaires, alors que les applications qui ne consomment pas beaucoup de bande passante et qui interagissent rarement avec les équipements sources et destinations sont plus adap-

tées pour des réseaux LoRa. Le tableau 1.1 identifie les caractéristiques des technologies de communications.

— Plateforme de gestion de l'infrastructure Fog :
Les paradigmes et outils utilisé pour gérer l'infrastructure IT pourraient être une source d'inefficacité énergétique. Il existe des technologies prometteuses pour déployer cette plateforme comme la virtualisation de fonction réseau (NFV) et les réseaux définis par logiciels (SDN). Ces technologies pourraient aider à améliorer l'efficacité des infrastructures de type Fog grâce à la virtualisation.

— Type d'applications :
Les auteurs dans [Jalali 2016] indiquent que les paramètres tels que le nombre de téléchargements, le nombre de mises à jour et la quantité de données montantes et descendantes jouent un rôle important sur la consommation d'énergie des applications.
Il ressort de l'article [Jalali 2016] que les applications nécessitant une puissance de calcul importante sont plus efficaces placées dans le Cloud. Les auteurs ont montré que les économies d'énergie réalisées grâce au Fog Computing viennent des applications qui génèrent et distribuent des données en continu avec les utilisateurs finaux et ayant un faible taux d'accès aux données provenant de l'extérieur du Fog. L'exemple le plus connu de ce types d'application sont les applications de surveillance sans reconnaissance faciale.
Les auteurs ont également montré que les applications avec des calculs lourds consomment plus d'énergie dans le Fog. Les serveurs Fog sont souvent limités en capacité de calcul et donc si on y exécute une application à calcul lourd, cette application risque de prendre toutes les capacités du serveur et prendra plus de temps à s'exécuter dans le Fog que dans le Cloud. Il serait donc plus judicieux de placer cette catégorie d'applications dans le Cloud. Il convient de noter que ces résultats sont basés sur l'hypothèse que le système IoT est alimenté par la grille électrique, et non les réseaux locaux intelligents (appelés micro-réseaux) à énergie renouvelable.

— Puissance statique (idle) des équipements Fog :
Les serveurs physiques des centres de données sont partagés de manière dynamique entre de nombreux services et utilisateurs. Si un service IoT placé dans le Cloud est inactif pendant un certain temps, le serveur peut être affecté à d'autres tâches. Ce partage permet de réduire le nombre de serveurs sous utilisés. En raison des contraintes de capacité des noeuds dans le Fog les stratégies de ré-allocation entre noeuds Fog ne peuvent pas être appliquées. Par conséquent, les noeuds Fog resteront inactifs pendant de longues périodes entre les tâches d'un service et ce qui rendra le Fog inefficace énergiquement.

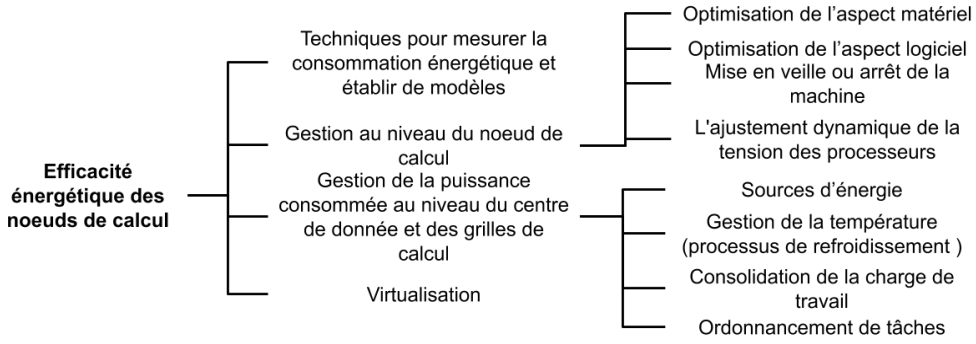


FIGURE 1.9 – Intervenir sur l'efficacité énergétique des noeuds de calcul [Orgerie 2014]

— L'utilisation des systèmes microgrid dans le Fog :
Il est nécessaire de concevoir des systèmes dynamiques et temps réel pour gérer la consommation énergétique des infrastructures de calcul et faire correspondre efficacement les disponibilités d'énergie renouvelable et les demandes IoT.

[Orgerie 2014] met en évidence les aspects pouvant agir sur l'efficacité énergétique des infrastructures de calculs, représentés en figure 1.9 et que nous expliquons brièvement dans ce qui suit.

— La précision des modèles utilisés pour estimer l'énergie a un impact direct sur l'efficacité et l'applicabilité des politiques choisies.
— Le choix des technologies de virtualisation peut améliorer l'efficacité énergétique d'un noeud. La virtualisation augmente l'efficacité énergétique de l'infrastructure de calcul en permettant la mutualisation des coûts énergétiques et des coûts d'exploitation de plusieurs serveurs sur un seul. Cependant, selon l'application et ses contraintes, des techniques sont plus adaptée que d'autres (containers, machines virtuelles etc.)
— Il existe plusieurs moyens d'intervenir au niveau du noeud de calcul pour améliorer son efficacité énergétique. Le premier étant un effort du concepteur pour améliorer les aspects matériel et logiciel du noeud, le second intervient dans une politique de gestion dynamique consistant à ajuster la fréquence des processeurs pour réduire la consommation énergétique due au calcul et enfin des politiques qui permettent d'éteindre la machine.

— Il existe également des approches qui interviennent à plus haut niveau, agissant sur le centre de données dans sa globalité. En appliquant des politiques de gestion tel que l'ordonnancement de tâches et les techniques de consolidations consistant à migrer la charge de travail d'un noeud à un autre ou bien en agissant sur les sources d'énergie qui alimentent les centres de données.

1.6.3.2 Le placement des applications IoT dans le Fog

Le problème de placement de services dans le Fog est communément défini comme un problème d'optimisation visant à garantir une ou plusieurs exigences en qualité de service du système, tels que : le temps de réponse, l'efficacité d'utilisation des ressources, la minimisation de la consommation du réseau. À cette fin, différentes approches ont été utilisées pour une résolution exacte ou approchée sous différentes contraintes système : [Tran 2017], [Skarlat 2017b], [Canali 2019], [Brogi 2019], [Nardelli 2019], [Guerrero 2018]. Ces travaux considèrent un environnement Fog-IoT sans nœuds mobiles et se concentrent uniquement sur les exigences de qualité de service.

L'infrastructure du Fog combine un nombre important de noeuds de calcul et réseau qui entraînent une importante consommation d'énergie. Les auteurs de [Liu 2018] tentent de minimiser l'énergie, le temps et les coûts d'exécution des nœuds mobiles en transférant leurs tâches vers des dispositifs de calcul complémentaires. Cependant, cette approche peut induire une utilisation excessive des liens réseau et augmenter la consommation d'énergie des noeuds restants. Les travaux de [Mebrek 2017] abordent le problème de l'affectation des noeuds IoT aux passerelles Fog et propose une résolution par algorithme génétique (GA) et algorithme de puissance incrémentale de diffusion (BIP), pour réduire la consommation d'énergie des nœuds mobiles sous contrainte de délai. Cette approche ne tient pas compte de la consommation d'énergie des nœuds Fog et des besoins en qualité de service des applications après l'affectation.

Contrairement aux travaux cités nous considéreront dans notre approche les objectifs de minimisation de l'énergie de tout l'infrastructure Fog ainsi que la minimisation du nombre de violations de délai des applications. Nous avons considéré plusieurs heuristiques et méta-heuristiques et comparé leurs performances avec différentes classes d'applications et différentes technologies d'accès réseau.

1.7 La qualité de services dans les environnements Fog-IoT

La qualité de service ou Quality of Service (QoS) désigne la capacité d'un service à répondre par ses caractéristiques aux différents besoins de ses utilisateurs ou consommateurs [Bathelot 2015]. En informatique, la Quality de service (QoS) est une description ou une mesure de performance d'un service donné : service réseau, service Cloud Computing ou un service applicatif. Pour donner une évaluation quantitative de la QoS, plusieurs métriques peuvent être considérés selon l'environnement ciblé (réseau, calcul, applicatif).

(1) Les réseaux :

La qualité de service (QoS) dans le domaine des télécommunications est représentée par la capacité des réseaux à véhiculer dans de bonnes conditions un type de trafic donné, en termes de disponibilité, débit, délais de transmission, gigue, taux de perte de paquets.

La gestion de la Qualité de service (QoS) a pour but d'optimiser les ressources d'un réseau ou d'un processus et de garantir de bonnes performances aux applications critiques. La qualité de service permet d'offrir aux utilisateurs des débits et des temps de réponse différenciés par applications (ou activités) suivant les protocoles mis en œuvre au niveau de la structure. Elle permet ainsi aux fournisseurs de services (départements réseaux des entreprises, opérateurs) de s'engager formellement auprès de leurs clients sur les caractéristiques de transport des données applicatives sur leurs infrastructures.

(2) Le Cloud :

La qualité de services (QoS) dans les environnements Cloud est représentée par des métriques telles que la disponibilité et le persistance d'un service. La gestion de la Qualité de service dans le Cloud vise à allouer les ressources de calcul des centres de données pour assurer les métriques citées précédemment. Le niveau de la QoS est déterminé dans un contrat dit "Sevice level agreement" (SLA) qui est établi entre l'utilisateur et le fournisseur de services Cloud. L'hétérogénéité des ressources de calcul et les mécanismes d'isolation et des plateformes de contrôle des centre de donnée complexifient grandement l'analyse, la prédiction et la gestion de la QoS dans le Cloud [Ardagna 2014]

(3) Les environnements IoT :

Nous avons vu dans la section 1.2.3 que la qualité de servie d'une application IoT peut être considéré au niveau d'un composant (service, capteur, actionneur) ou d'une manière plus globale en considérant toute l'application d'un point de vu logiciel (virtuel).

La qualité de service (QoS) dans les réseaux et dans le Cloud a été largement étudiée. Cependant, il existe très peu de travaux pour l'instant qui s'intéressent à la Qualité de Service des systèmes qui combinent les infrastructure du Cloud Computing et l'Internet des objets (Cloud-IoT) ainsi que les

environnements Fog-IoT.

Les auteurs [Aazam 2014] discute les problématiques qui découlent de l'utilisation des infrastructure Cloud Computing pour l'Internet des Objets. Selon les auteurs le stockage des données IoT devrait se faire dans le Cloud pour assurer la longévité des objets. Le modèle économique des services IoT pourrait également être simplifié en utilisant le Cloud (reprendre les modèles Cloud déjà existants).

Même si contrairement au Cloud, le Fog peut réduire les latences des applications, Ce dernier doit assurer un certain nombre de critères QoS qui restent à déterminer. Les approches pour arriver à gérer la QoS et les différents besoins des applications dans le Fog restent aussi un sujet à étudier. Un environnement complexe tel que le Fog peut entraîner des problèmes tels que la perte de messages, le partitionnement du réseau et des pannes de courant. Dans les applications déployées à grande échelle, les problèmes et les incohérences apparaissent fréquemment en fonction de la manière dont les services sont distribués .

Les auteurs de [Babu 2018] identifient quartes critères qui permettrons au Fog d'assurer la qualité de service.

— Assurer la connectivité : Des aspects tels que le clustering réseau ou la virtalisation dans un réseau diversifié ou hétérogène, offrent de nouvelles possibilités pour réduire les coûts et étendre la connectivité.
— La Fiabilité : La fiabilité dans le Fog est similaire aux exigences de fiabilité des infrastructures distribuées. Elle peut être améliorée en vérifiant périodiquement le temps nécessaire pour reprendre une tâche après un échec, en reprogrammant la tâche qui a échoué ou en répliquant les nœuds pour exécuter la tâche en parallèle. Cependant, la reprogrammation et la vérification périodique de l'échec et de la reprise de la tâche ne sont pas adaptées au Fog. Alors que la réplication peut fonctionner dans certains cas.
— Le délais : Les applications sensibles aux délais doivent avoir un traitement en temps réel. Le système doit pouvoir anticiper les régions des requêtes à venir pour les clients qui utilisent fréquemment les mêmes applications et doit pouvoir effectuer un traitement pour rendre les données disponibles lorsque les clients les recherchent.
— Capacité : La bande passante du réseau et la capacité de stockage sont les deux principales caractéristiques à prendre en compte en matière de capacité. Pour atteindre ces deux caractéristiques, il est très important de savoir où se trouvent les données dans le Fog, ce qui signifie que la localisation des données rapidement est primordiale.

1.8 La mobilité dans les environnements Fog-IoT

Bien que la mobilité soit un élément très important qui participe à créer l'aspect dynamique et incertain des environnements IoT et Fog, son impact sur les politiques de placement et de gestion des infrastructures reste très peu étudié. La mobilité des environnements Fog-IoT est principalement due aux noeuds de périphérie qui peuvent être aussi bien considérés comme noeud de calcul ou comme noeud utilisateurs. L'impact de la mobilité sur l'énergie et la qualité de service (QoS) reste difficile à anticiper et est le résultat des effets combinés entre le changement du comportement utilisateur, les politiques de gestion utilisées et les technologies sollicitées [IEA 2019].

[Bittencourt 2017] étudie le problème de l'ordonnancement dans les infrastructure Fog Computing, en considérant que la mobilité des utilisateurs a une influence sur le temps d'exécution des applications. Les auteurs comparent trois politiques différentes d'ordonnancement d'applications : premier arrivé premier servi, priorité par délais de réponse et dernier arrivé dernier servi. Les informations sur la mobilité de l'utilisateur ne sont pas utilisées dans les algorithmes de placement. De plus, la simulation de la mobilité est uniquement basée sur l'augmentation et la diminution du nombre de nœuds IoT connectés sur les différentes passerelles Fog.

Dans le domaine de la recherche sur la virtualisation des fonctions réseau (NFV), [Mouradian 2019] propose un modèle de programmation linéaire en nombre entier pour minimiser une fonction pondérée qui regroupe le "makespan", le temps de l'exécution de toute les fonctions et le coût d'exécution des fonctions. Les auteurs utilisent un modèle de mobilité de points aléatoires où la vitesse et la direction des nœuds mobiles sont choisies de manière aléatoire et indépendante. Ce modèle néglige l'aspect temporel, spatial et géographique qui définit la plupart des mouvements d'objets IoT.

[Cziva 2018] propose un moyen de reprogrammer dynamiquement le placement optimal de la fonction de réseau virtuel (VNF) basé sur la fluctuation temporelle de la latence du réseau en utilisant la théorie de l'arrêt optimal qui considère le moment de la prise de décision de reprogrammation afin de minimiser la latence de bout en bout de tous les utilisateurs. Ils modélisent le placement comme un programme linéaire entier (ILP) afin de minimiser le nombre de migrations tout en garantissant qu'il ne dépasse pas un nombre maximum de violations pour certaines applications. Cependant, par rapport à notre travail, cette approche est uniquement basée sur la métrique de violation de latence.

Aucun des travaux cités précédemment n'introduit l'information de mobilité à travers la modélisation de leur approche de placement et dans la logique de placement.

L'aspect mobilité a été largement abordé dans le domaine du "Mobile Cloud

Computation" (MCC) via des stratégies de migration, où la plupart des travaux tentent de trouver un compromis entre la réduction du coût induit par le processus de migration et la garantie des exigences de qualité de service des applications/utilisateurs souvent représentée par la latence. La plupart des travaux identifiés s'appuient sur des modèles de processus de décision de Markov 1-D ou 2-D (MDP). L'objectif est de produire un ensemble de décisions de migration, en tenant compte de la position des nœuds mobiles, généralement définie par les chaînes de Markov : [Ksentini 2014], [Wang 2019], [Wang 2015], [Aissioui 2018]. Les travaux de [Ksentini 2014] tentent d'assurer un bon compromis entre le coût de la migration et l'amélioration de la qualité d'expérience des utilisateurs. Ils abordent le problème de décision de migration en appliquant des algorithmes d'itérations sur la valeur et d'itérations sur la politique pour résoudre les équations d'optimalité de Bellman. Des travaux similaires ont été réalisés dans [Wang 2019] avec un système plus complexe qui prend en compte un processus de Markov 2-D, avec des utilisateurs multiples et des tailles de créneaux horaires hétérogènes. Ils considèrent également le compromis entre les coûts à court terme et à long terme des décisions prises. [Wang 2015] essaye de pallier à la complexité de la recherche de la solution optimale en proposant un nouvel algorithme pour que la solution soit plus rapide que les algorithmes d'itération de valeur et de politique. Les auteurs de [Aissioui 2018] introduisent le "Follow Me edge-Cloud" (FMeC), un cadre visant à garantir l'exigence en latence du système automobile dans les futurs réseaux 5G. Ils placent les services automobiles au nœud de calcul de périphérie le plus proche des véhicules. Le scénario proposé ne prend en compte qu'une seule voiture et ne tient pas compte de l'état de chaque nœud lors de la migration. D'autres travaux tels que [Ouyang 2018] utilisent une approche théorique pour aborder le problème de l'optimisation à long terme à chaque instant avec l'approche d'optimisation de Lyapunov. Ces approches sont des méthodes coûteuses en temps et en argent et ne se prêtent pas bien à un déploiement sur des systèmes réels.

L'approche MDP présente certaines limites telles que les besoins importants en données nécessaires pour estimer une fonction de probabilité de transition et une fonction de récompense pour chaque action possible. En outre, à mesure que la taille du problème augmente, il devient difficile de résoudre les MDP de manière optimale sur le plan des calculs.

L'autre catégorie d'approches qui pourraient être utilisée pour traiter la mobilité repose sur des méthodes de prévision et d'apprentissage. Les auteurs de [Ojima 2018] proposent de prédire la mobilité des utilisateur en utilisant la méthode du filtre de Kalman pour gérer l'affectation des ressources aux utilisateur. [Zhang 2016] proposent le framework SEGUE basé sur un MDP à une dimension pour les décisions de migration en se basant sur l'état du réseau et des serveurs et utilisent la méthode ARIMA pour prédire la QoS

du système. Les méthodes de prédiction ont pour principal inconvénient l'importante consommation de ressources de calcul et le temps de prédiction qui est difficilement applicable en ligne. De plus, la précision des prédiction est directement impactée par la qualité des données utilisées.

Dans notre travail, on propose dans un premier temps et afin d'éviter les inconvénients des méthodes de migration et de prédiction, un stratégie de placement de services qui exploite les informations de mobilité des objets IoT pour minimiser la consommation énergétique et la violation du délai des applications. Dans un deuxième temps, nous proposons de gérer le placement de services grâce à un système de placement en ligne en intégrant la possibilité de replacer les services au cours du temps. Nous proposons également un modèle de mobilité simple d'utilisation qui pourra être utilisé comme source d'information par les stratégies de placement.

1.9 Synthèse

Dans ce chapitre nous avons dans un premier temps établi les définitions l'IoT et du Fog Computing ainsi que les défis apportés par ces derniers qui peuvent empêcher une gestion efficace de l'infrastructure et dégrader la Qualité de service des applications qui y sont déployées.

Les environnements Fog-IoT sont une cause importante et non négligeable de l'augmentation de la consommation énergétique des infrastructures IT. Il a également été montré dans plusieurs travaux que la politique de gestion des ressources physiques et virtuelles a un impact direct sur l'efficacité énergétique de ces infrastructures.

D'autre part, nous avons abordé la mobilité et son importance dans ces environnements, tout système de gestion de services doit prendre cet aspect en considération dans les diverses politiques de placement ou de maintenance qu'il propose.

Nous avons par la suite établi une bibliographie et un résumé de travaux qui ont essayé de répondre à la problématique de placement de services et d'application IoT dans le Fog et positionner nos travaux par rapport à l'existant.

Le tableau 1.4 donne un résumé global des travaux qui ont chercher à améliorer l'efficacité des infrastructure Fog-IoT et des différents aspects système considérés par chaque travail ainsi que les objectifs et la méthode de résolution utilisée. Ici cinq catégories d'objectifs sont identifiées : (1) la qualité de service (QoS), (2) l'utilisation des ressources (UR), (3) le coût financier (CF), (4) le coût opérationnel et (5) l'énergie. La majorité des travaux modélisent les infrastructures Fog en structures hiérarchiques de neouds de calcul et les applications en graphes de services.

Citations	Objectif	Algorithme type	Algorithme	Environnement de tests	Modèle d'applications	Modèle de l'infrastructure	Catégorie de objectif
[Tran 2017],[Skarlat 2017b]	Max. utilisation noeuds Fog	Métaheuristique	Algorithme Génétique (GA)	iFogSim	DAG	Hiérarchique	UR
[Canali 2019]	Min. latence et temps de traitement des capteurs vers les noeuds Fog	Métaheuristique	Algorithme Génétique (GA)	KINTRO	DAG	Hiérarchique (cité de modena, Italie)	QoS
[Brogi 2019]	Max. QoS, Min. consommation ressources Fog	Métaheuristique	Algorithme Génétique (GA) et Monte Carlo	FogTorch II (java)	Multi-services (4 service)	Barabasi-alber topology	QoS + UR
[Nardelli 2019]	Min. Temps de réponse des application	Heuristique et Métaheuristique	Recherche tabu et local , greedy first fit	Brite, slover, google plateform	Data stream distribued workflow (sequential, diamond, replicated)	Waxman de 36 à 100	QoS
[Maia 2019a]	Min. nombre de violations de délais	Exatce, Métaheuristique	linéraisation, GA	Python, CPLEX MILP	un service par application, Scalable (monolithique)	Edge hiérarchique	QoS
[Mann 2019b]	Min. coût financier	Heuristique	Heuristique Fogpart	Testbesd, Gurobi	Facotry application "fiab"	un fog et un Cloud (hiérarchique)	CF
[Mahmud 2019]	Min. temps de réponse des applications	Heuristique	Heuristique	Testbed FogBus+iFogSim	Mutli services (industrie 4.0)	Hiérarchique	QoS
[Amira Rayane 2018]	Min. latence applications	Heuristique	Résolution exacte,Heuristique proposée	iFogsim, CPLEX	multi-services (6 services séquentiels)	Hiérarchique	QoS
[Charântola 2019]	Min. délais des applications	Heuristique	Heuristique proposée	iFogSim	multi-services (VSTO,EEG)	Hiérarchique	QoS
[Benamer 2018]	Min. temps de réponse applications	Exacte, heuristic	Heuristique proposée	iFogSim	muti-services	Hiérarchique	QoS
[Benamer 2019]	Min. temps de réponse application, Min. énergie	Métaheuristique	Recherche Pingouin	iFogSim	multi-services (VSTO,EEG,HealthCare)	Hierarchique	QoS, Energie
[Maia 2019b]	Min. nombre de violations de délais d'application	Métaheuristique	Exacte résolution ,GA	iFogSim	Multi-services	Hiérarchique	QoS
[Rezazadeh 2018]	Min. énergie, délais d'application, coût opérationnel	Métaheuristique	Recuit simulé	iFogSim	Multi-services (HelathCare)	Hiérarchique	QoS, Énergie, CO

Table 1.4 – Résumé des travaux de placement d'applications dans le Fog.

Chapitre 2

Outils de simulation et bancs d'essai pour les environnements Fog-IoT

Sommaire

2.1 Introduction 55
2.2 Source de données pour les applications IoT 56
2.2.1 Les critères à définir pour les applications IoT 56
2.2.2 Applications IoT de la littérature 57
2.2.3 Applications d'imagerie interactives et/ou collaboratives . . . 57
2.2.4 Applications Santé 58
2.2.5 Applications d'environnement ambiant (smart home, smart building) 59
2.2.6 Applications de l'industrie 4.0 59
2.3 Outils et données pour les infrastructures Fog-IoT 60
2.3.1 Principales caractéristiques des simulations des environnements Fog-IoT 60
2.3.2 Topologies des infrastructures Réseau pour le Fog 63
2.3.3 Taxonomie des simulateurs Fog et IoT existants 66
2.3.4 Plateforme et bancs d'essai pour le Fog Computing et l'IoT . 69
2.4 MyMobileIFogSim 70
2.4.1 Architecture d'iFogSim 70
2.4.2 Contrainte de placement et limite réseau d'iFogSim 71
2.4.3 La migration et la mobilité 73
2.5 Synthèse 74

2.1 Introduction

Le Fog Computing et l'Internet des objets (IoT) sont des domaines de recherche relativement jeunes. A ce jour, il existe très peu de simulateurs et de plateformes de test englobant toutes les caractéristiques attribuées aux environnements Fog-IoT et qui permettraient d'effectuer des expérimentations se rapprochant au mieux d'un environnement réel. De plus, les données sur les

caractéristiques des applications IoT et leur comportement une fois déployées sont rares, difficilement accessibles (souvent ce sont des données industrielles) et très peu documentées.

Ce chapitre donne un aperçu de quelques applications IoT qui sont très utilisées dans la littérature et les possibles topologies réseau d'une infrastructure Fog-IoT. Nous résumons également les outils que l'on peut trouver dans la littérature pour créer un environnement Fog-IoT ainsi que leurs avantages et limites. Enfin, nous présentons les améliorations apportées au simulateur que nous avons sélectionné pour effectuer nos expérimentations.

2.2 Source de données pour les applications IoT

Depuis quelques années, le domaine de l'IoT est considéré comme un axe de recherche stratégique par de nombreux industriels comme Cisco, IBM, Amazon ou encore Google. Cependant, il est encore difficile de trouver des sources d'informations complètes sur ces systèmes et plus particulièrement sur la partie logicielle des applications IoT. Les applications IoT et leurs objets sont très hétérogènes et souvent, les paradigmes de programmation ainsi que les plateformes de gestions IoT ne sont pas standardisés. Ce qui complexifie l'établissement et l'évaluation de politiques de contrôle de gestion des ressources et de la qualité de service (QoS). Des efforts pour développer des plateformes IoT standardisées sont en cours de développement mais se basent uniquement sur la communication avec les équipements IoT [om2m 2015]. Les organismes de standardisation donnent des indications haut niveau pour le déploiement mais le paradigme de programmation d'applications est très peu documenté. Ces informations sont essentielles pour permettre une conception et une étude efficace des stratégies de déploiement et de gestion du cycle de vie des applications dans les infrastructures Fog.

2.2.1 Les critères à définir pour les applications IoT

Afin de créer des stratégies de placement d'applications IoT pertinentes et applicables à l'échelle réelle, il est primordial d'identifier avec précision les spécificités des applications IoT et de déterminer les principaux éléments qui les distinguent des applications existantes.

Voici ci-dessous une liste non exhaustive des principaux éléments à déterminer :

— Déterminer les classes d'applications IoT et leurs besoins en temps de réponse, leurs profils de consommation des ressources de calculs, réseau et énergie.
— Définir les fréquences et les moments d'utilisation des applications ainsi que la fréquence d'interaction avec ses équipements.

— Identifier les applications et services pouvant être mutualisés entre plusieurs utilisateurs ainsi que la localisation de ces servies.

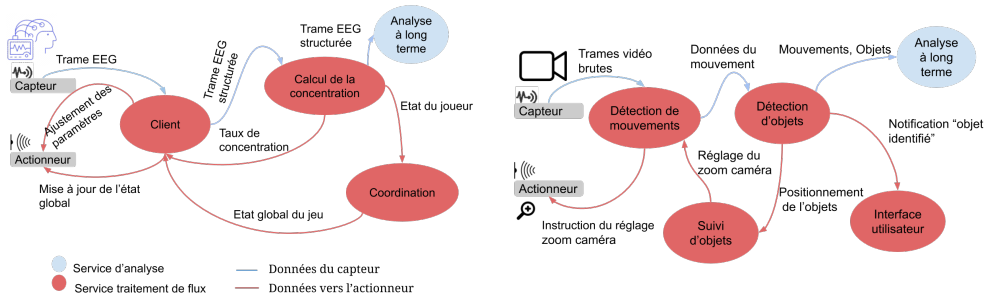
2.2.2 Applications IoT de la littérature

Un très grand nombre de travaux proposent de représenter les applications IoT en regroupement de services interdépendants aussi appelés modules ou opérateurs qui accomplissent une tâche globale et interagissent avec un ou plusieurs objets IoT. Cette représentation permet d'affiner la granularité du déploiement et de gestion de la dite application dans un environnement distribué et mobile [Giang 2015].

Dans cette section, nous présentons les topologies des applications les plus utilisées dans la littérature dans différents secteurs. On remarque que les applications IoT se composent en moyenne de 4 services, même si cela dépend du choix de développeur. Il est très rare de trouver des applications avec un nombre de services supérieur à 5. Concernant les besoins des applications, le critère le plus répandu est souvent le nombre minimum de coeurs CPU à allouer à un service ou encore le nombre d'instructions que va consommer ce dernier ainsi que la taille moyenne des données échangées entre les services. De part le type de l'application et son contexte d'utilisation (besoin métier) l'application aura un délai de réponse à ne pas dépasser. La fréquence d'envoi, la taille des données envoyées et du traitement de flux dépend aussi des capacités des objets IoT qui est très variable. Les topologies d'applications les plus utilisées dans la littérature sont les topologies : "Séquentielle", "Diamant", "Maitre-Esclave" et en couche "Répliquée".

2.2.3 Applications d'imagerie interactives et/ou collaboratives

Que ce soit pour de la vidéo surveillance via plusieurs équipements fixes et mobiles ou du suivi d'objets ou d'individus (reconnaissance faciale) ou encore pour créer un environnement de réalité augmentée interactif, les applications incluant une ou plusieurs caméras sont souvent les exemples les plus étudiés dans la littérature dans les environnements Fog-IoT [Gupta 2016], [Hong 2013].



(a) Topologie d'application de jeu de réalité augmentée avec Electro Tractor Beam Game (EEG)[Gupta 2016].

(b) Topologie d'application de surveillance vidéo [Gupta 2016], [Hong 2013].

FIGURE 2.1 – Topologies des applications temps réel et vidéo de surveillance collaborative et jeu de réalité augmentée de la littérature.

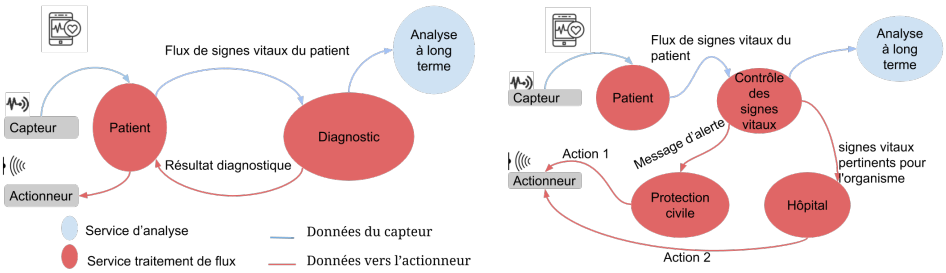
La figure 2.1a présente une application de jeu en ligne entre plusieurs joueurs munis d'un casque de détection de signaux qui permet aux joueurs d'interagir avec un écran. Cette application est composée de 3 services dont un service propre à chaque joueur (client) et deux services chargés de la synchronisation des informations des joueurs et enfin un service de sauvegarde et d'analyse des données récoltées à long terme.

La figure 2.1b présente une application de surveillance collaborative et de traitement d'images composée de 4 services de traitement et d'un service d'analyse. Ce schéma est le schéma de services standard de la majorité des applications de vidéo surveillance.

2.2.4 Applications Santé

La santé est parmi les secteurs qui risquent de bénéficier grandement des facilités apportées par les objets IoT et leurs applications. L'application la plus prometteuse reste celle des opérations chirurgicales à distance via des robots. Cette application pourrait être grandement utilisée en médecine de guerre. Pour l'instant, les applications de santé existantes utilisent des objets IoT comme les smartphone ou d'autres objets portables (wearables) et se focalisent principalement sur la surveillance de patients à risque. L'objectif est d'agir en conséquence suite aux analyses des différentes métriques collectées.

Ces données sont souvent soumises à la protection de la vie privée ce qui complexifie le processus de collecte à des fins de recherche. Les figures 2.2a et 2.2b sont les topologies d'applications de santé les plus étudiées dans la littérature [Perez Abreu 2020a], [Benamer 2019].



(a) Topologie d'application healthcare 1 [Perez Abreu 2020a].

(b) Topologie d'application healthecare 2 [Benamer 2019].

FIGURE 2.2 – Topologies des applications healthcare de la littérature.

2.2.5 Applications d'environnement ambiant (smart home, smart building)

Les applications d'environnement ambiant pour les bâtiments ou les maisons intelligentes sont des applications simples non sujettes à la mobilité pour la plupart des cas. Le travail [Brogi 2018] propose une topologie d'application assez simple décomposée en trois services que nous présentons dans la figure 2.3

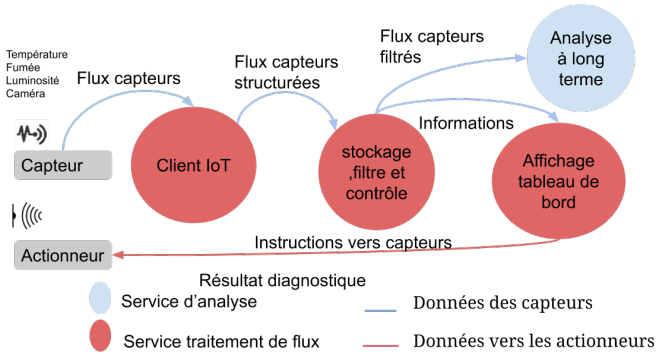


FIGURE 2.3 – Topologie d'application décomposée en 3 services pour un environnement ambiant [Brogi 2018].

2.2.6 Applications de l'industrie 4.0

Les auteurs [Mann 2019a] définissent l'application "Factory in a Box" (FiaB). L'application présente un environnement de production complet qui permet de contrôler des robots à distance dans un container à l'aide de lunettes intelligentes. La figure 2.4 illustre la topologie de l'application FiaB. Les auteurs définissent également le taux de transfert moyen des données par jour entre les services ainsi que le nombre minimum de CPU devant être alloués à chaque service et le coût financier pour l'exécution d'un service, qui est calculé selon les tarifs de la plateforme EC2 d'Amazon. Les auteurs [Mahmud 2019]

proposent également une application industrie 4.0 avec de la surveillance et de l'analyse d'images pour la gestion des risques d'accidents du travail avec des robots.

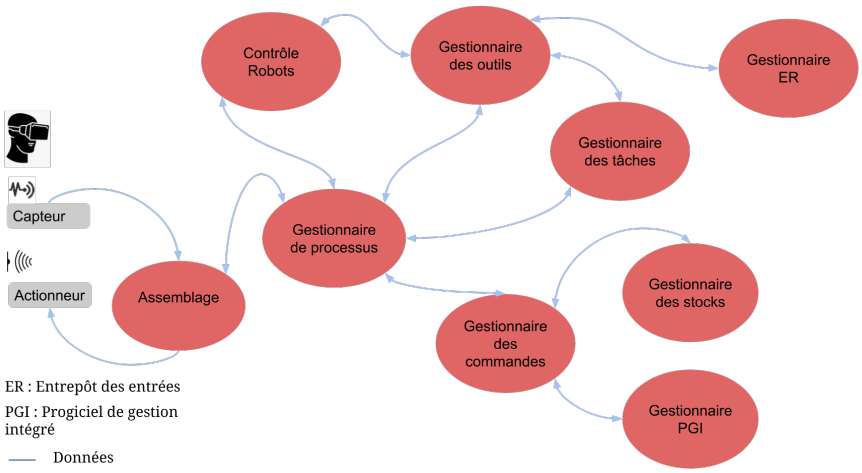


FIGURE 2.4 – Topologie d'application de l'industrie 4.0 [Mann 2019a]

2.3 Outils et données pour les infrastructures Fog-IoT

Un contexte Fog combine à la fois le réseau, les systèmes de calcul et les utilisateurs mobiles, ce qui le rend beaucoup plus instable et dynamique au niveau des performances comparé au Cloud. Le Fog est sujet à différentes fluctuations et l'impact du réseau est beaucoup plus important que dans les infrastructures Cloud.

Dans cette section nous allons présenter les aspects que devrait supporter un outil de simulation Fog-IoT, nous allons ensuite présenter les outils existants qui pourraient être utiliser comme bloc de base pour supporter chaque aspect d'un simulateur (calcul, réseau et mobilité). Enfin, nous présentons une taxonomie des simulateurs Fog (Fog-IoT) existants et permettant d'avoir un ou plusieurs aspects cités précédemment.

2.3.1 Principales caractéristiques des simulations des environnements Fog-IoT

Dans cette section, nous présentons l'ensemble des besoins en simulation des environnements Fog-IoT et plus particulièrement pour supporter le placement des applications IoT dans des infrastructures Fog Computing.

(1) Ressources et modèles de calcul

Une infrastructure Fog regroupe des noeuds de calcul avec des capacités hétérogènes (mémoire, stockage et processeur). Chaque noeud doit pouvoir gérer

différemment ses ressources. Par exemple, les noeuds doivent avoir plusieurs types d'ordonnancement pour les ressources processeur : temps et mémoire partagés.

(2) Ressources réseau et modèles de communication

Contrairement aux architectures de calculs telles que le Cloud où les éléments les plus importants sont les ressources de calcul, dans le Fog, le réseau est aussi important de part son impact que le calcul. Le réseau doit faire partie des éléments à considérer pour l'établissement de stratégies d'optimisation de ressources. Les simulateurs Fog doivent offrir des comportements réseaux détaillés avec la possibilité de choisir entre différents algorithmes de routage, de détailler les spécificités des liens et la possibilité de faire de la gestion de la Qualité de Service (QoS) sur les liens (allocation d'une partie de la bande passante pour des flux ou des utilisateurs spécifiques). Aujourd'hui, il existe une multitude de réseaux non filaires pour l'IoT et qu'il faudrait implémenter dans les simulateurs Fog-IoT. Les paradigmes du réseau défini par logiciel (SDN) et de la virtualisation de fonctions réseau (NFV) remplacent peu à peu les réseaux classiques. Les simulateurs doivent offrir la possibilité de placer des contrôleurs SDN dans l'infrastructure et offrir des services réseau sous forme de fonctions NFV. Il serait également intéressant d'intégrer des intergiciels IoT dans l'infrastructure de calcul ce qui permettrait d'avoir des estimations de coûts réseau plus réalistes.

(3) Outils de virtualisation

La virtualisation est à la base de la majorité des paradigmes de calculs actuels (Cloud et Fog). Cette dernière permet d'exploiter efficacement les infrastructures physiques. Il existe actuellement plusieurs approches et technologies de virtualisation comme KVM, Docker, Kubernetess, LXC ou LXD. Les approches de virtualisation ont chacune un impact diffèrent sur la consommation énergétique des infrastructures ainsi que sur le temps de migration des entités virtuelles. Elles peuvent également restreindre ou limiter le stratégies de gestion des ressources de calcul.

(4) Modèles de coûts

Comme toute infrastructure de calcul/communication, hétérogène et largement distribuée, le Fog engendre divers coûts qui doivent apparaître lors des simulations.

Parmi ces coûts nous pouvons citer :

— Les coûts financiers dus à l'utilisation d'un noeud de calcul ou d'un lien de communication et les coûts de déploiement et d'opération de l'infrastructure par zone. L'aspect Financier du Fog est très peu étudié.

Or, il est essentiel de pouvoir établir de nouveaux modèles économiques en fonction des spécificités du Fog.
— Les coûts de mise en place de la qualité de services (QoS) et des politiques de sécurité pour le fournisseur, pour l'opérateur et pour les utilisateurs finaux.
— Le coût d'exécution d'un service sur une machine (en temps, en capacité etc.)
— Le coût de communication entre les services (en temps, en bande passante etc.)

(5) Sources et modèles énergétiques

Les simulateurs Fog-IoT doivent offrir la possibilité de choisir les sources d'alimentation des noeuds, de faire varier leurs profils de consommation énergétique en fonction du type de noeuds et permettre d'exploiter plusieurs modèles de consommation énergétique pour le calcul et pour les communications. Il est également important d'intégrer des modèles de charge et de décharge de batterie pour les objets IoT et des modèles d'énergie "harvesting", qui est définie comme le processus permettant de capturer, stocker et transformer de l'énergie de source externes (solaire, thermique, éolienne etc.) par de petits composants sans fils autonomes.

(6) Modèles des réseaux de capteurs et des objets IoT

Les objets IoT sont hétérogènes tant dans le nombre et le type de capteurs/actionneurs qu'ils possèdent et des protocoles de communication qu'ils utilisent que dans leurs profils de consommations énergétique (ainsi que leur stratégie d'économie d'énergie).

En plus de leurs capacités de calcul relativement faibles, la fréquence d'interaction des objets IoT avec les autres noeuds de calcul varie d'un objet à un autre (pouvant être représentée par différentes lois de probabilité). Les simulateurs Fog-IoT doivent présenter tous les aspects cités précédemment en plus de détails comme la taille des paquets supportée, la durée de vie des composants, le protocole de communication et de sécurité ainsi que des modèle de pannes.

Dans certains scénarios plusieurs objets IoT fonctionnent comme une seule entité, les interactions dans ces clusters et leurs caractéristiques doivent également être intégrées dans les simulateurs.

(7) Modélisation haut niveau des applications

Les applications IoT tendent à être déployées sous forme de services (unités de calcul indépendantes pouvant être déployées sur différents noeuds physiques). Les besoins de ces services ainsi que leurs interactions sont très peu documentés. De plus, le déploiement des services sur une plateforme de calcul

distribuée rend la mesure des performances de l'application en question très complexe en environnement réel. Les données traversent plusieurs couches de calcul et différents domaines réseaux opérés par des organismes privés ou publics. L'échange d'informations entre deux opérateurs qui est souvent régie par différents contrats est un processus chronophage et complexe. De plus les mesures peuvent être impactées par des perturbations externes qui ne sont pas facilement identifiables. Pour cela, il est très important d'avoir accès à ces aspects en simulation. Il serait également intéressant d'avoir des services (ou micro-services) sur étagère qui permettraient de construire des applications plus complexes.

(8) Mobilité et géolocalisation
Dans des environnements dynamiques et large échelle comme le Fog et l'IoT, il est important de pouvoir localiser les noeuds dans les espaces 3D et d'avoir des modèles de mobilité par catégorie d'objets (drones, véhicules autonomes, smartphone etc.) ou de pouvoir importer des données réelles. Il faudrait également implémenter les mécanismes de gestion de la mobilité au niveau réseau avec le handover/handoff des terminaux utilisateur et le replacement des services et fonctions réseau.

(9) Mise à l'échelle (Scalabilité)
Un simulateur Fog doit permettre le déploiement d'un très grand nombre de noeuds de calcul, de noeuds réseau et d'objets IoT sur une large zone géographique. Ceci permettrait à plusieurs fournisseurs et opérateurs d'infrastructures (calcul et réseau) d'établir des stratégies communes pour anticiper et minimiser leurs coûts d'investissements et d'opérations.

(10) Orchestration et gestion du cycle de vie des services IoT Il est important de pouvoir simuler les principales caractéristiques des systèmes de gestion d'infrastructures virtualisées comme Openstack, Cloudstack ou Cloudera et d'avoir les principales fonctionnalités de ces outils. En offrant la possibilité de démarrer, d'arrêter, de migrer ou de répliquer les services à la demande. Le système doit également pouvoir gérer les interactions entre le noeuds Cloud et les noeuds Fog pour établir des politiques de décharge (offloading).

2.3.2 Topologies des infrastructures Réseau pour le Fog

Les infrastructures distribuées sont représentées principalement par des structures de graphes. Il est important d'avoir des topologies variées pour étudier leur impact sur les stratégies de placement d'applications.

Dans ce qui suit, nous allons présenter les générateurs de topologies réseau et les simulateurs réseau et les simulateurs de mobilité les plus complets et les

plus utilisés dans la littérature.

2.3.2.1 Générateurs de topologies réseau

(1) BRITE

BRITE [Medina 2001] permet de générer différentes topologies réseau. Il est également possible d'ajouter des topologies personnalisées ou d'importer des modèles de générateurs comme GT-ITM, Inet, et NLANR AS. Les topologies crées peuvent être exportées dans des simulateurs réseau comme ns2, SSFNET, JavaSim et OmNet++. BRITE est libre d'accès et est développé en Java et C++.

(2) CAIDA

Le Center for Applied Internet Data Analysis (CAIDA) est un organisme qui s'intéresse à l'analyse du comportement du réseau internet. Il propose un générateur de topologies réseau [CAIDAGroup 2019] dont le comportement est créé à partir d'études statistiques sur les données internet réelles. Le projet est soutenu par une très grande communauté et est mis à jour régulièrement. CAIDA peut s'intégrer avec des outils comme BRITE.

(3) NetworkX

NetworkX [Hagberg 2008] est un outil développé en Python, qui permet de générer des structures de graphes complexes et offre une multitude d'algorithmes de traitement de graphes.

2.3.2.2 Simulateurs Réseaux et mobilité

(1) ns-3

ns-3 [Riley 2010] est un simulateur réseau à évènements discrets en libre accès développé en c++ et python. Il est utilisé et supporté par une grande communauté scientifique. Ce dernier est fréquemment amélioré et étendu par de nouveaux modèles et fonctionnalités réseau. ns-3 permet de simuler des phénomènes réseau comme la congestion et permet de suivre l'évolution des paquets dans l'infrastructure. Il propose plusieurs protocoles de communication et de routage. Il a également un module pour la mobilité des noeuds.

(2) OMNeT++

OMNeT++ [Varga 2010] est un simulateur à évènements-discrets développé en c++. Il permet de modéliser et de simuler des réseaux de communication. Il supporte le déploiement de réseaux de capteurs, du Peer-to-Peer et d'autres réseaux sans fils. Il est utilisé pour supporter la partie réseau dans beaucoup de simulateurs comme Veins (simulation de réseaux de véhicules) ou FogNetSim++ que nous allons voir par la suite. Son architecture modulaire permet

de développer facilement des extensions. Tout comme ns-3, il a un module dédié à la mobilité des noeuds. OMNeT++ permet de développer le code des micro-services et ne les présente pas uniquement comme des boites noires avec des données entrantes et sortantes.

(3) Simulation of Urban MObility (SUMO)

Très peu de simulateurs prennent en charge la mobilité des noeuds. Il existe des simulateurs de mobilité très complets mais qui s'intéressent à la mobilité d'un point de vu réseau comme ns-2/ns-3 ou GNS3 ou qui s'intéressent uniquement aux réseaux véhiculaires. Dans ce qui suit nous nous intéressons à SUMO [Krajzewicz 2002], le simulateur que nous avons utilisé dans nos travaux. SUMO est un simulateur de mobilité en libre accès. Il est principalement utilisé pour la simulation de trafic véhiculaire. Il permet d'intégrer de la mobilité piétonne et de définir d'autres types d'objets mobiles. SUMO permet de créer différents scénarios de mobilité et de récolter plusieurs métriques de mobilité. Il est facilement interfaçable à d'autres simulateurs grâce à son API simple d'utilisation, sa large communauté et à sa large documentation technique. L'outil est développé et principalement maintenu par l'Institut des systèmes de transport Allemand sous licence EPL 2.0.

(4) Veins

Veins [Sommer 2011] est un outil en libre accès pour la simulation des réseaux véhiculaires. Il est développé à partir de deux outils que nous avons décrits précédemment, OMNet++ (pour le réseau) et SUMO (pour la mobilité et la simulation du traffic de véhicules). Le simulateur s'intéresse uniquement au comportement de mobilité des véhicules, qui est très détaillé, et ne permet pas d'avoir d'autres comportements de mobilité.

2.3.2.3 Simulateurs infrastructures de calcul

(1) SimGrid

SimGrid [Brennand 2016] est un simulateur d'environnement de calcul distribué et de systèmes à large échelle comme les grilles de calcul, le Cloud ou les systèmes peer-to-peer (P2P). Il permet de déployer des applications distribuées et d'élaborer des stratégies de gestion de ressources. Les simulations SimGrid sont rapides et peuvent être exécutées dans une seule machine. Il est possible d'ajouter des modules en langage C++, C, Python ou Java. La communauté de SimGrid est très active et le simulateur est régulièrement amélioré.

(2) DCworms

Data Center Workload and Resource Management Simulator (DCworms)

[Kurowski 2013], [Rostirolla 2019] est un simulateur à évènements discrets développé par Poznan Supercomputing et le Networking Center (PSNC) à partir du framework GSSIM. DCworms est développé en JAVA et permet de modéliser des infrastructures de centre de données distribuées. Ce simulateur propose des modèles d'estimation de la consommation et de l'efficacité énergétique des infrastructures de calcul et permet de déployer des stratégies de gestion des ressources de calcul.

(3) CloudSim

CloudSim [Goyal 2012] permet de modéliser des infrastructures Cloud distribuées et d'établir des politiques pour la gestion des ressources de calcul tout en modélisant des applications distribuées. Il permet également d'estimer la consommation énergétique des noeuds de calcul. Il est également possible d'insérer des éléments dynamiquement durant la simulation. Le simulateur est développé en Java et son architecture modulaire facilite l'intégration de nouvelles fonctionnalités. Cependant, il consomme plus de ressources de calcul comparé à SimGrid et les simulations prennent plus de temps. En raison de sa nature extensible beaucoup de travaux ont été réalisés pour l'enrichir. Le travail de [Beloglazov 2012] propose le premier modèle de consommation d'énergie. Par la suite, [Moreno 2019] propose DartCSIM, une extension qui permet la simulation de l'énergie et du réseau simultanément. D'autres extensions existent comme CloudAnalyst [Wickremasinghe 2010] et CloudReports [Sá 2014].

Les trois simulateurs de plateforme de calcul décrits précédemment sont les plus utilisés par la communauté HPC mais en ce qui concerne l'intégration des fonctions Fog et IoT, CloudSim est actuellement l'outil le plus utilisé.

2.3.3 Taxonomie des simulateurs Fog et IoT existants

Des études récentes [Perez Abreu 2020b], [Markus 2020], [Ahvar 2019] montrent que plus d'une dizaine de simulateurs Fog ont été proposés au cours de ces cinq dernières années, CloudSim [Goyal 2012] étant la base d'un grand nombre d'entre eux.

Dans ce qui suit nous allons décrire brièvement quelques simulateurs Fog. Les simulateurs sont comparés entre eux selon deux catégories de critères : (1) les caractéristiques génériques d'un simulateur avec le langage de programmation, la scalabilité et la popularité (en nombre de citations), (2) les caractéristiques du simulateur pour supporter les aspects Fog-IoT qui sont pertinents pour notre travail : la topologie des infrastructures Fog, le modèle des applications IoT, le modèle énergétique, la fonctionnalité de migration des services et la mobilité des objets IoT. Le tableau 2.1 présente la comparaison des simulateurs Fog décrits dans cette section.

TABLE 2.1 – Synthèse des simulateurs supportant le paradigme du Fog. Le Nombre de citations est pris des résultats de recherche sur IEEE Xplore et researchegate, consultés en septembre 2020.

Nom	Citation	Langage	Migration	Énergie	Mobilité	Scalabilité	Citations	Topologie	Applications
iFogSim	[Buyya 2019]	JAVA	Non	Calcul linéaire	Non	Oui	580	Arbre	Modules
EdgeCloudSim	[Sonmez 2017]	JAVA	Non	Partielle	Partielle	Oui	51	Arbitraire	Modules
MyiFogSim	[Lo 2018]	JAVA	Oui	Partielle	Partielle	Oui	43	Arbre	Modules
FogTorch	[Brogi 2017]	JAVA	Non	Oui	Non	Non	179	Arbitraire	Modules
FogNetSim++	[Qayyum 2018a]	JAVA	Non	Oui	Oui	Oui	42	Arbitraire	Modules
YAFS	[Lera 2019a]	Python	Oui	Oui	Oui	Oui	5	Arbitraire	Modules
EmuFog	[Mayer 2017]	JAVA	Non	Non	Non	Partielle	11	Arbitraire	Docker

2.3.3.1 Yet Another Fog Simulator (YAFS)

YAFS [Lera 2019b] est un simulateur pour les réseaux Cloud/Fog. Le principal objectif de YAFS est l'évaluation des performances des stratégies de placement, de programmation et de routage. Parmi les mesures rapportées par YAFS figurent l'utilisation réseau, le temps de réponse et le temps d'attente. Les données brutes résultantes sont rapportées dans un journal et permettent aux utilisateur de calculer d'autres mesures de qualité de service. YAFS est un logiciel libre d'accès sous licence MIT et est développé en Python.

2.3.3.2 FogNetSim++

FogNetSim++ [Qayyum 2018b] est un simulateur en libre accès, qui se concentre sur les aspects réseau. Il a été développé à partir du simulateur à évènements discrets OMNeT++ (en C++). Le simulateur offre la possibilité de faire du hand-over et propose différents protocoles de communication IoT comme MQTT ou CoAP. Il propose également quelques modelés de mobilité.

2.3.3.3 EdgeCloudSim

EdgeCloudSim [Sonmez 2018] est une extension de CloudSim en libre accès sous licence GNU General et développé en java. Ce simulateur se focalise sur divers concepts du Edge computing et permet de modéliser des aspects réseau (propriété des liens et capacités réseaux), des aspects de calculs (exécution de taches, ordonnancement de machines virtuelles) et des aspects Fog (mobilité, offloading). Il est possible de déployer des architectures multi-couches avec plusieurs serveurs Edge/Fog gérés par un noeud Cloud. Les métriques de sortie sont les délais dans les réseaux locaux (Local Area Network -LAN) et dans les réseaux étendus (Wide Area Network -WAN), le taux d'échecs d'exécution des services et le taux moyen d'échecs d'exécution des services dus à la mobilité, le taux d'utilisation des VMs et le temps d'exécution des services.

Il est possible de définir des modèles de mobilité avec le module "mobile client layer". Le simulateur ne propose pas de modèles de consommation énergétique mais l'utilisateur peut définir ses propres modèles de coûts. La mi-

gration de services n'est pas supportée et les méthodes pour créer des clusters de calcul n'est possible qu'avec les noeuds appartenant à la même couche.

2.3.3.4 FogTorch

FogTorch [Brogi 2017] est un outil libre d'accès sous licence MIT et développé en Java. L'outil permet de tester des stratégies de déploiement d'applications en établissant des critères de QoS. Le simulateur utilise des simulations de Monte Carlo pour implémenter les variations de la bande passante des liens réseau. Le simulateur a deux métriques de sortie : (1) La garantie de QoS assurance et (2) La consommation des ressources Fog (pourcentage de mémoire consommée). Le simulateur permet de modéliser différentes topologies réseau mais ne supporte pas la mobilité des noeuds.

2.3.3.5 EmuFog

EmuFog [Mayer 2017] est un émulateur d'environnements Fog en libre accès sous licence MIT. Cet outil développé en JAVA, offre la possibilité de déployer des applications sous forme de containers Docker. Cet émulateur est construit à partit de MaxiNet qui est une version étendue de MiniNet pouvant être utilisée sur plusieurs machines physiques. MiniNet permet de créer des réseaux définis par logiciels [Keti 2015]. Il est possible d'utiliser des topologies réseaux générées à partir d'outils comme BRITE ou CAIDA. EmuFog permet de garder les historiques des consommations CPU et mémoire de chaque noeud. Il est possible de spécifier les emplacements des services IoT sur les machines physiques. L'outil n'a pas d'interface permettant de récupérer des métriques globales comme le temps de réponse d'une application ou l'énergie consommée par cette dernière.

2.3.3.6 iFogSim et MyiFogSim

iFogSim [Gupta 2016] est un simulateur en libre accès développé en JAVA à partir de CloudSim [Calheiros 2011]. Le simulateur permet de modéliser et de simuler un environnement Fog Computing incluant des objets IoT, il permet d'évaluer la gestion des ressources de calcul et de tester des politiques de placement et d'ordonnancement de services dans le Fog. Les métriques de sortie sont la consommation énergétique du calcul, l'utilisation réseau et les temps d'exécution des services.

iFogSim est jusqu'à présent le simulateur le plus utilisé dans la littérature pour les travaux liés au Fog. Cependant, il a deux principales limites :

— La mobilité des noeuds n'est pas prise en charge.
— Les classes liées au réseau et à la gestion des communications sont très limitées et contiennent uniquement des attributs statiques (latence,

bande passante). En plus d'une architecture arborescente simple qui est peut représentative des architectures réelles souvent denses et hyper-connectée le routage est imposé dans un seul sens : les informations du capteur doivent allé du noeud le plus bas vers le plus haut et la réponse vers les actionneurs doit descendre d'un noeud père à un noeud fils. Cet aspect limite l'étude et les possibilités pour les stratégies de placement de services et nous place dans un contexte peu réaliste.

MyiFogSim [Lo 2018] est une extension d'iFogSim qui permet de supporter la migration des services déclenchée par la mobilité des objets. Les auteurs proposent 3 techniques de migrations et introduisent les mécanismes de handover/handoff.

Cependant les stratégies proposées pour la migration présentent les limites suivantes :

— Migration d'une seule machine virtuelle ou container à la fois.
— La stratégie de migration proposée est basée sur le mouvement des utilisateurs d'une cellule à une autre (follow me strategy) et il ne permet pas d'établir des stratégies de migrations pro-actives.
— La mobilité introduite est très rudimentaire et consiste uniquement en un changement de vitesse ou de direction aléatoire, ce qui ne représente pas la mobilité des objets IoT. Le mobilité IoT est souvent régie par un comportement humain et ce dernier ne peut pas être considéré comme purement aléatoire.
— Le simulateur ne propose pas de modèle pour estimer le temps et l'énergie de la migration de services.

Grâce à sa communauté très active et aux nombreuses études sur le Fog menées avec ce simulateur, nous avons choisi d'effectuer nos expérimentations sur les environnements Fog-IoT avec iFogSim/MyiFogSim.

Nous avons levé certaines des contraintes citées précédemment et nous avons intégré ce simulateur avec le simulateur de mobilité SUMO. Nous présentons ces modifications par la suite en section 2.4.

2.3.4 Plateforme et bancs d'essai pour le Fog Computing et l'IoT

Dans cette section nous présentons deux plateformes qui permettraient de tester nos propositions en environnement réel Fog et IoT et qui pourraient être utilisées pour étendre nos expérimentations. Comme la mobilité est un aspect important dans notre travail, il ne nous a pas été possible d'utiliser c'est deux plateformes car l'une [Adjih 2015] propose des modèles très simples et l'autre [Mouël 2019] est récente et n'est arrivée que vers la fin de la thèse.

2.3.4.1 FIT IoT-Lab

FIT IoT-Lab [Adjih 2015] est une plateforme de calcul regroupant plus de 1500 noeuds de calcul distribués sur 5 sites en France (Grenoble, Lille, Lyon, Strasbourg et Saclay). La plateforme est libre d'accès et permet de déployer des applications IoT. Elle offre un large panel de machines de calcul et de composants IoT. La plateforme permet également l'utilisation de plusieurs technologies radio comme le LoRa, le LR-WPANs, le Sub-GHz et le Bluetooth Low Energy (BLE). La communauté commence également à proposer des noeuds IoT mobiles avec des circuits de mouvements prédéfinis.

2.3.4.2 Yet Another PI platform (YOUPI)

YOUPI [Mouël 2019] est une plateforme Fog/Edge Computing pouvant communiquer avec d'autres infrastructures de calcul comme Grid500 et FIT IoT Lab. YOUPI permet de capturer des propriétés dynamiques de l'environnement comme la mobilité, les interférences, les comportements utilisateurs. Les expérimentations peuvent être rejouées pour effectuer des analyses hors lignes.

2.4 MyMobileIFogSim

Dans cette partie, nous présentons les améliorations apportées à iFogSim pour l'aspect réseau et pour la mobilité avec l'intégration du simulateur SUMO pour supporter la mobilité des objets IoT.

2.4.1 Architecture d'iFogSim

L'architecture du simulateur iFogSim, décrite par la figure 2.5 se compose de plusieurs couches. La première couche modélise les capteurs et les actionneurs qui interagissent avec l'environnement extérieur et qui sont les sources de données et les puits des instructions transitant dans l'infrastructure. Les caractéristiques des objets IoT comme la fréquence d'envoi et la taille des données envoyées par les capteurs et les instructions reçues par les actionneurs peuvent être adaptées pour simuler des objets particuliers comme des caméras, des drones ou des smartphones. iFogSim définit les noeuds Fog comme des noeuds réseau pouvant faire du calcul. La topologie des noeuds de calcul est organisée en une hiérarchie arborescente de noeuds. Le noeud Cloud est la racine, les feuilles sont les objets IoT et les noeuds intermédiaires sont les noeuds Fog. La communication est seulement possible entre une paire de noeuds parent-fils.

L'infrastructure a trois principaux services : (1) contrôle des composants permettant de récupérer des informations sur : l'utilisation des ressources,

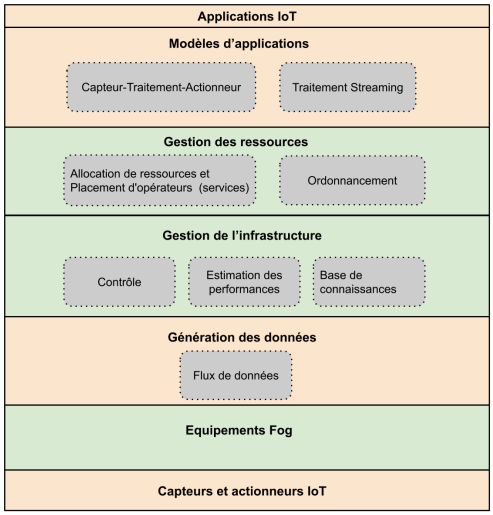


FIGURE 2.5 – Architecture haut niveau d'iFogSim [Gupta 2016]

la puissance consommée par les machines et la disponibilité des noeuds, (2) gestion des ressources permettant d'ordonnancer les taches et effectuer des stratégies de placement, et (3) contrôle de la consommation d'énergie. Les classes pour calculer les coûts d'exécution, le modèle énergétique et migrer des machines virtuelles sont directement héritées de CloudSim. Le simulateur permet également de modéliser des applications sous forme de graphes de services (appelés modules ou opérateurs).

2.4.2 Contrainte de placement et limite réseau d'iFogSim

iFogSim présente une contrainte de placement sur les services des applications. Si un service s_j est successeur du service s_i alors s_j doit être placé soit sur le même noeud ou sur un noeud de couche supérieure. Le simulateur impose également de placer les services liés à un objet donné, uniquement sur les noeuds de la même branche que cet objet. Cette contrainte n'est pas réaliste et dans notre cas elle pourrait éliminer des solutions de placement de services intéressantes.

Pour pouvoir permettre un placement de services sans contraintes nous avons ajouté une table de routage à chaque noeud pour pouvoir transférer les données vers le bon service. Cette table est mise à jour à chaque placement ou replacement de services. Nous avons également dû ajouter un modèle qui permet d'estimer la consommation énergétique des communications réseau en nous basant sur les formules de [Krief 2012].

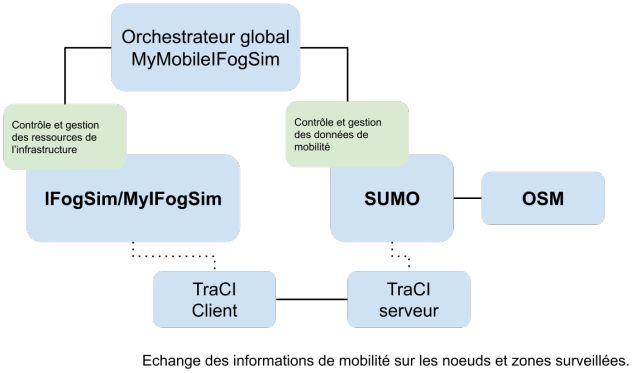


FIGURE 2.6 – Principaux modules de MyMobileIFogSim

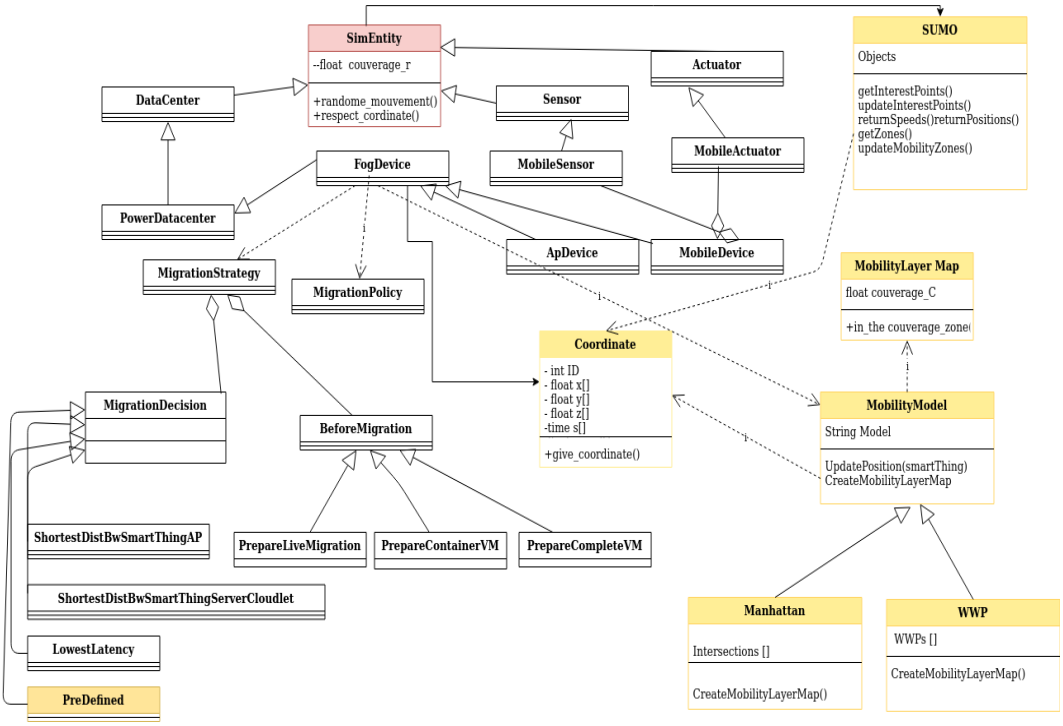


FIGURE 2.7 – Diagramme UML des principales classes ajoutées à MyiFogSim (en jaune).

2.4.3 La migration et la mobilité

Dans un premier temps, pour pallier les limitations liées à la mobilité, nous avons ajouté deux modèles de mobilité de la littérature : le modèle Weighted way point (WWP) pour les mouvements piétons et le modèle Manhattan pour les mouvements de véhicules. Comme le montre le diagramme UML de la figure 2.7, il est possible d'ajouter d'autres modèles en implémentant l'interface "MobilityModel". Pour chaque modèle, le simulateur ajoute une cartographie avec les spécificités du modèle en question gérée avec la classe "MobilityLayerMap".

Par la suite, pour tirer avantage des outils de mobilité existants et pouvoir établir des scénarios de mobilité plus détaillés et réalistes, nous avons interfacé MyIFogSim à SUMO, le simulateur de mobilité introduit précédemment. Le simulateur permet de visualiser les scénarios, tout en récoltant des informations de mobilité variées macroscopiques (métriques génériques liées à un endroit donné comme le flux d'entrées/sorties d'une autoroute) et microscopiques (métriques spécifiques à un noeud mobile comme sa vitesse ou sa direction etc.).

La figure 2.6 illustre les principaux blocs d'intégration des deux simulateurs. L'API On-line Interaction (TraCI) que nous avons utilisé, expose les primitives permettant d'échanger les informations de mobilité des noeuds entre les deux simulateurs à chaque pas de temps lors de la simulation. Les deux simulateurs sont gérés par un orchestrateur global (développé en Python) qui permet d'ajuster les zones de mobilité selon leurs densités en noeuds, les points d'intérêts de calcul (les noeuds Fog) et de mobilité (les bâtiments populaires). L'orchestrateur sélectionne les métriques de mobilité à récolter en fonction des besoins du gestionnaire de l'infrastructure Fog.

L'utilisateur doit fournir à l'orchestrateur global les informations concernant l'infrastructure Fog, les applications IoT à déployer et la zone de mobilité à étudier à partir d'OpenStreetMap (OSM) et doit choisir le nombre d'objets mobiles (véhicule, piéton ou autre).

iFogSim transmet via TraCI les informations sur la position des noeuds Fog ainsi que leur charge en calcul et le nombre d'utilisateurs qui y sont connectés. Ceci se traduira dans la carte visualisée sur SUMO par un code couleur.

Le scénario de mobilité peut être généré aléatoirement grâce à un module de SUMO mais peut également être spécifié via des matrices origines-destinations. Ensuite, le simulateur déduira le plus court chemin de chaque paire origine-destination. Il est également possible de spécifier chaque route empruntée par chaque noeud manuellement ou d'importer des données de mobilité existantes. Nous avons également intégrer une stratégie de migration pro-active (qui n'attend pas que l'objet se déplace pour déclencher la migration de ses services) dans la classe "preDefined" et proposé une formule d'estimation du temps et de l'énergie consommés par la migration, décrite

dans le chapitre 4.

2.5 Synthèse

Dans ce chapitre nous avons pu voir les applications IoT les plus utilisées dans la littérature. Nous avons comparé les simulateurs Fog-IoT existants selon différents critères. Actuellement, aucun des simulateurs proposé ne regroupe tous les aspects d'un environnement Fog-IoT que nous avons identifiés en section 2.3.1. Nous avons également proposé avec "MyMobileIFogSim" des améliorations du simulateur iFogSim que nous avons par la suite intégré avec le simulateur de mobilité SUMO.

Chapitre 3

Placement de services en environnement statique

Sommaire

3.1 Introduction 75
3.2 Modèle des applications IoT 76
3.2.1 Représentation de l'application 76
3.2.2 Délai de réponse de l'application 78
3.3 Modèle de l'infrastructure Fog 80
3.3.1 Les noeuds de l'infrastructure 82
3.3.2 Les liens réseau 83
3.4 Formulation du problème de placement de services 85
3.4.1 La consommation énergétique 85
3.4.2 Violations des délais 86
3.4.3 Fonction objectif 87
3.5 Approches de résolution 87
3.5.1 Optimisation Discrète par essaims particulaires (DPSO) . . . 87
3.5.2 Algorithme Génétique (GA) 98
3.6 Résultats 100
3.6.1 Test du DPSO et des heuristiques pour le placement de services 101
3.6.2 Test du DPSO et des méta-heuristiques pour le placement statique 107
3.7 Bilan des expérimentations et amélioration des approches de placement 115
3.8 Synthèse 118

3.1 Introduction

Considérer un problème de placement de services dans un environnement statique (sans mobilité de noeuds) est une approche qui a pour avantage d'anticiper ou d'estimer les besoins en calculs et en communications d'une zone Fog donnée. Cette approche se fait dans le cadre des études pour le dimensionnement des infrastructures. Les gestionnaires d'infrastructures réseau et de calculs doivent effectuer cette étude pour minimiser leurs coûts d'investissement et d'opération tout en estimant les éventuels besoins utilisateurs.

Dans ce chapitre nous posons le modèle mathématique de notre système Fog-IoT dans un environnement statique, la représentation de l'infrastructure Fog, des applications IoT ainsi que le modèle de consommation énergétique et la métrique représentant la qualité de service (QoS) d'une application. Nous formulons ensuite le problème de placement hors ligne des services IoT dans l'infrastructure Fog avec pour objectif de minimiser la consommation énergétique de l'infrastructure et les violations des délais de réponse des applications IoT.

3.2 Modèle des applications IoT

On considère des applications IoT multi-services faisant partie de la catégorie de workflow de services déterministes définis dans la section 1.5.1 du chapitre 2. La topologie d'une application est représentée par un graphe orienté de services qui s'échangent des données. Cette représentation est utilisée dans plusieurs travaux sur l'IoT dont [Buyya 2019], [Giang 2015] et [Hilman 2020]. La figure 3.1 donne un exemple de topologie.

3.2.1 Représentation de l'application

Une application a_i est composée d'un nombre n_i de services. Les interactions entre les services et leurs dépendances sont représentées par un graphe orienté $\mathbb{G}_i = (\mathbb{S}_i, \mathbb{E}_i)$. Les noeuds de l'ensemble $\mathbb{S}_i$ représentent les services de l'application et les arcs du graphe de l'ensemble $\mathbb{E}_i$ représentent la dépendance de données entre les différents services. Le service initial ou la source du graphe est une abstraction du capteur de l'application s_0^i et le dernier service $s_{n_i+1}^i$ représente l'actionneur de l'application.

3.2.1.1 Les services

Une application IoT reçoit des données de l'environnement extérieur par l'intermédiaire de capteurs. Ces données transitent dans le réseau et sont traitées par un ensemble de services virtuels. Par la suite, des instructions sont transmises vers les actionneurs. À partir de la définition précédente, nous pouvons décomposer les services d'une application IoT en plusieurs catégories.

Un service $s_j^i \in \mathbb{S}_i$ peut faire partie de l'une des catégories suivantes :

(1) Service source de génération de données (capteur)

Nous considérons un service fictif source s_0^i qui représente le capteur de l'équipement utilisateur de l'application. Ce dernier sera chargé d'envoyer les données avec une certaine fréquence $freq_0^i$ suivant un modèle ou une distribution de probabilité adéquate.

(2) Service destination/puis (actionneur)

Le service destination $s^i_{n_i+1}$ est un service fictif représentant le capteur du noeud utilisateur, qui va recevoir des instructions.

(3) Service de traitement flux

Un service de traitement de flux est un service faisant partie de l'ensemble des services qui traitent les données transitant du capteur vers l'actionneur.

(4) Service d'analyse

Un service d'analyse est un service qui ne fait pas partie de l'ensemble des services à traitement de flux. C'est-à-dire qu'il ne se trouve pas dans un chemin de données (chemin dit critique) allant d'un capteur vers son actionneur. Les services d'analyse n'interagissent pas (ou interagissent très rarement) avec l'utilisateur (noeud IoT) et sont utilisés à des fin d'analyses à long terme des performances de l'application et du comportement des noeuds utilisateurs.

Un service s^i_j de traitement de flux ou d'analyse est l'abstraction d'une entité de calcul virtuelle telle qu'une machine virtuelle ou des containers, ils sont définis par les caractéristiques suivantes :

- tec^i_j : représente la technologie de déploiement du service : machine virtuelle (VM), container (CT) ou Bundles OSGi etc.
- $inst^i_j$ représente la quantité d'instructions CPU, en Millions d'instructions (MI) que devra traiter le service à la réception de chaque flux de données.
- ram^i_j : représente les besoins en mémoire vive en MB.
- d^i_j : représente le délai d'exécution maximal du service s^i_j en secondes.
- $pred^i_j$: représente l'ensemble des services prédécesseurs de s^i_j dans le graphe.
- $succ^i_j$: représente l'ensemble des services successeurs du services s^i_j dans le graphe.

3.2.1.2 Liens et dépendances entre les services IoT

Chaque arc orienté $e^i_{jj'} \in \mathbb{E}_i$, allant du service s^i_j au service $s^i_{j'}$ représente une dépendance de données entre eux. L'arc $e^i_{jj'}$ est défini par les informations suivantes :

- $data^i_{jj'}$: représente le volume de données en kB, échangés entre s^i_j et $s^i_{j'}$ et qui traverse le réseau à chaque communication.
- $d^i_{jj'}$: représente le délai de communication maximal entre s^i_j et $s^i_{j'}$ en secondes.

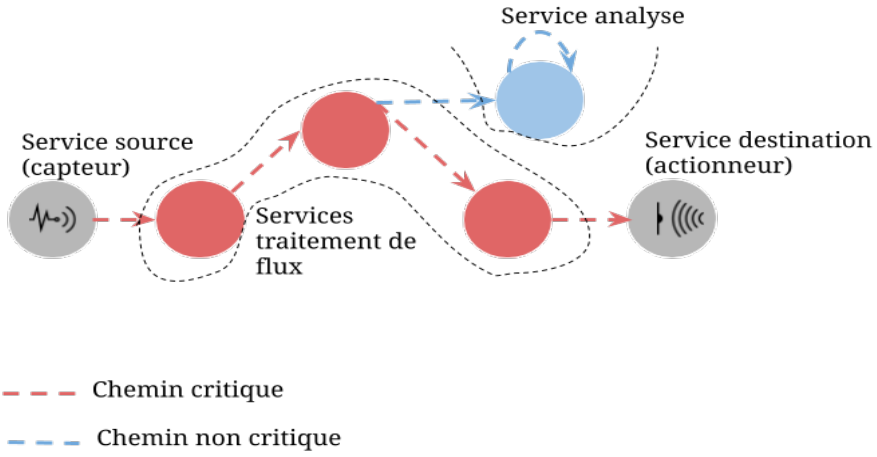


FIGURE 3.1 – Représentation de la topologie d'une application mettant en évidence les chemins critiques et les différentes catégories de services définies dans cette section.

3.2.1.3 Ensemble de chemins critiques $\mathbb{CP}_i$

Une application IoT peut être composée de plusieurs types de services. Les chemins critiques sont définis par les développeurs de l'application et déterminent un ensemble de services de traitement de flux et d'arcs allant du capteur à l'actionneur.

Une application possède au moins un chemin critique. Un chemin critique $cp_q^i \in \mathbb{CP}_i$ est une séquence simple (les arcs ne se répètent pas) et finie d'arcs orientés dans la même direction appartenant à l'ensemble $\mathbb{E}_i$ qui lient un ensemble de services d'un traitement de flux de $\mathbb{S}_i$ et qui démarre d'un service capteur et se termine par un service actionneur [Williamson].

3.2.2 Délai de réponse de l'application

Classes d'applications et temps de réponse maximal vers l'utilisateur

Chaque application IoT a_i appartient à une classe Υ_i parmi celles définies dans [Guevara 2017]. Une priorité lui est associée. Le Tableau 3.1 présente les classes : Mission Critique (MC), Temps Réel (RT), Streaming (ST) et Best Effort (BE) ayant respectivement un niveau de priorité de 0, 1, 2, et 3 (0 étant le niveau le plus prioritaire).

Estimation du temps d'exécution attendu pour un service

Le temps d'exécution [Wilhelm 2008] dépend de la complexité de la requête à exécuter en nombre d'instructions et de la capacité de calcul de la machine dans laquelle cette dernière est exécutée. Dans notre cas, un service est défini

TABLE 3.1 – Classes d'applications IoT et leur priorité

QoS / Class	Best-Effort (BF)	Streaming (ST)	Real-Time (RT)	Mission Critical (MC)
Délai (ms)	–	150	50	20
Demande en bande passante	Basse	Élevée	Élevée	Élevée
Fréquence de communication	Basse	Medium	Elevée	Élevée
Demande CPU	Basse	Basse-Medium	Basse-Medium	Medium-Élevée
Localisation des données	Distante	Locale-Proximité-Distante	Locale-Proximité-Distante	Locale-Proximité-Distante
Mobilité	Élevée-Intermédiaire-Basse	Élevée-Intermédiaire-Basse	Élevée-Intermédiaire-Basse	Élevée-Basse
Exemples	Partage de fichiers	Streaming vidéos	Jeux de Réalité Augmentée	Application de santé
Priorité	3	2	1	0

par un nombre d'instructions processeur. On considère α^u_{ijk} le temps d'exécution estimé du service s^i_j qui est utilisé par l'objet IoT m_u et qui est placé sur la machine m_k. La formule de α^u_{ijk} est présentée un peu plus tard en section 3.4 avec l'équation (3.10).

Estimation du temps de communication moyen attendu entre deux services

En considérant que les services s^i_j et $s^i_{j'}$ sont placés respectivement sur les machines m_k et $m_{k'}$, le temps de communication entre ces deux services dépend de la latence et de la bande passante des liens entre les machines m_k et $m_{k'}$. $\beta^u_{ijj',kk'}$ est le temps de communication entre les services s^i_j et $s^i_{j'}$ de l'application a_i associée au noeud IoT m_u et placés respectivement sur les noeuds m_k and $m_{k'}$. Cette valeur dépend des capacités des liens de communication entre les machines qui hébergent les services et de la taille des données échangées entre ces derniers. La formule de $\beta^u_{ijj',kk'}$ est présentée par la suite avec l'équation (3.12) de la section 3.4.

Estimation du temps de réponse de l'application

Le temps de réponse d'une application peut dépendre de plusieurs facteurs, comme le temps de réponse moyen des services qui la composent ou le temps de réponse maximum garanti par le fournisseur de service. $d^u_i(t)$ est le temps de réponse effectif de l'instance de l'application a_i dédiée à l'objet m_u. Ce dernier correspond au temps entre l'envoi d'un paquet de données du capteur de l'objet m_u et l'arrivée de l'instruction résultante de cet envoi sur l'actionneur. Le délai

dépend de la position de l'objet IoT et de l'emplacement des services. $d_i^u(t)$ est estimé pour chaque instance de l'application $a_i \in \mathbb{A}$ utilisée par l'objet IoT $m_u \in \mathbb{U}$.

Violation du délai

En plus du temps d'exécution des services et du délai de communication entre ces derniers, chaque application a_i a un délai de réponse global d_i^{max} qui représente le temps maximal autorisé pour le traitement d'un flux de données allant du capteur à son actionneur. Ce temps est une métrique qui représente la qualité de service (QoS) de l'application. Cette valeur est prise en fonction du délai maximal autorisé par la classe à laquelle l'application appartient ou peut être renseigné par l'utilisateur ou une formule mathématique donnée.

Une violation de délai ou l'échec du temps de réponse d'une application a_i pour un objet IoT donnée m_u se produit lorsque le temps de réponse effectif $d_i^u(t)$ dépasse le temps de réponse maximum qui lui a été attribué. Le nombre de violations de délai est défini comme étant le nombre de fois où le temps de réponse effectif des requêtes du capteur vers l'actionneur est supérieur à son délai maximum de réponse d_i^{max}.

Priorité de l'application

Le niveau de priorité d'une application ψ_i, peut être défini directement par sa classe (Ex : 0,...,3) ou bien à partir du délai maximum de l'application s'il a été renseigné par l'utilisateur comme défini par l'équation (3.1). Ceci permet d'avoir une plus grande précision et établir des priorités entre les applications appartenant à la même classe.

$$\psi_i = \frac{1}{d_i^{max}} \tag{3.1}$$

La table 3.2 récapitule les notations des variables pour les applications IoT.

3.3 Modèle de l'infrastructure Fog

Le National Institute of Standards and Technology (NIST) ainsi que la majorité des travaux de la littérature définissent le Fog comme une hiérarchie entre différentes couches de noeuds de calcul et de communication. A partir de la définition précédente, nous considérons une infrastructure Fog hiérarchique 3-tiers, représentée par un graphe non orienté $\mathbb{G} = (\mathbb{M}, \mathbb{L})$ et illustré par la figure 3.2, où $\mathbb{M}$ est l'ensemble des nœuds de calcul et $\mathbb{L}$ l'ensemble des liens de communication.

La première couche de noeuds regroupe l'ensemble des terminaux ou objets IoT $\mathbb{U}$, la deuxième couche regroupe les noeuds Fog (cloudlet) dans l'ensemble

	Variable	Type	Description
Indices	i	$[0, A-1]$	Indice des applications.
	j	$[1, n_i]$	Indice des services de l'application a_i qui ne sont ni des capteurs ni des actionneurs.
Applications	a_i	Symbole	Application i.
	n_i	Symbole	Nombre de services de l'application a_i.
	Υ_i	$\{MC, RT, ST, BE\}$	Classe de l'application a_i.
	ψ_i	$\{0, 1, 2, 3\}$ ou calculée	Niveau de priorité de l'application a_i.
	d_i^{max}	$\{20, 50, 150, \infty\}$	Délai de réponse maximal de l'application a_i.
	$\mathbb{G}_i = (\mathbb{S}_i, \mathbb{E}_i)$	Symbole	Graphe de l'application a_i avec S_i l'ensemble des services et E_i l'ensemble des arcs.
	$\mathbb{CP}_i$	Ensemble	Ensemble de chemins critiques de l'application a_i.
	$\mathbb{A}^t$	Ensemble	Ensemble des applications IoT à l'instant t.
	$\mathbb{A}_u^t$	Ensemble	Ensemble des applications utilisée par l'objet IoT m_u à l'instant t.
	$\mathbb{N}_u^t$	Ensemble	Ensemble des services de l'objet IoT m_u à l'instant t.
	N	$\mathbb{N}$	Nombre total de services.
Services	s_j^i	Symbole	$j$ème service de l'application a_i.
	s_0^i	Symbole	Service source (capteur).
	$s_{n_i+1}^i$	Symbole	Service destination (actionneur).
	$inst_j^i$	Symbole	Quantité d'instructions CPU, en Millions d'instructions (MI) que devra traiter le service s_j^i.
	ram_j^i	Symbole	Besoin en mémoire du service s_j^i en MB.
	$pred_j^i$	Ensemble	L'ensemble de services prédécesseurs du service s_j^i.
	$succ_j^i$	Ensemble	L'ensemble de services successeurs du service s_j^i.
	tec_j^i	{VM, CT, OSGi}	Technologie de déploiement du service.
Liens	$e_{jj'}^i$	Symbole	Arc orienté allant du service s_j^i au service $s_{j'}^i$.
	$data_{jj'}^i$	Symbole	Volume de données en kB, échangés entre s_j^i et $s_{j'}^i$.
	$d_{jj'}^i$	Symbole	Temps de communication entre s_j^i et $s_{j'}^i$.

TABLE 3.2 – Résumé des variables des applications IoT.

$\mathbb{F}$, et la dernière couche regroupe les noeuds Cloud $\mathbb{C}$.

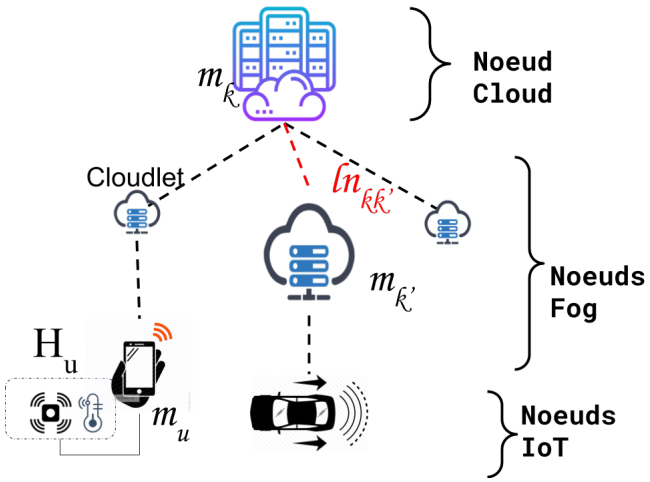


FIGURE 3.2 – Représentation de l'infrastructure Fog 3-tiers.

3.3.1 Les noeuds de l'infrastructure

Chaque noeud physique $m_k \in \mathbb{M}$ est défini par :

- Sa catégorie, $\eta_k \in \{IoT, Fog, Cloud\}$
- Ses capacités, définies par les vecteurs $\Omega_k^{max} =< cpu_k^{max}, mem_k^{max}, disk_k^{max} >$, dont les attributs représentent respectivement les capacités maximales de calcul en millions d'instructions par seconde (MIPS), de mémoire et de stockage en MB du noeud. $\Omega_k^t =< cpu_k^t, mem_k^t, disk_k^t >$ donne les capacités disponibles à l'instant t.
- Un ensemble $\mathbb{I}_k$ d'interfaces de communications hétérogènes.
- Sa puissance électrique de calcul au repos pc_k^{min} et sa puissance dynamique maximale pc_k^{max}, qui est la puissance consommée par la machine au maximum de sa charge de travail CPU.
- Sa puissance électrique de communication de ses interfaces au repos pn_k^{min} et la puissance dynamique maximale pn_k^{max}. Dans le travail que nous présentons, la puissance de communication maximale et minimale d'un noeud est déduite de la manière suivante : un noeud a une capacité de transmission maximale c_k^{max} qui selon les travaux proposés par [Krief 2012] peut être estimée avec la formule de l'équation (3.2).

$$c_k^{max} = \frac{1}{2} \sum_{ln_{kk'} \in \mathbb{L}} bw_{kk'} \tag{3.2}$$

Les puissances maximale et minimale sont ensuite déduites avec les équations (3.3.1) et (3.4) selon [Krief 2012].

$$pn_k^{max} = c_k^{max\frac{3}{2}} \tag{3.3}$$

$$pn_k^{min} = 0.85c_k^{max\frac{3}{2}} \tag{3.4}$$

Les nœuds ont en outre des caractéristiques spécifiques en fonction de leur type η_k :

— Un noeud IoT possède des capacités de calcul et de stockage et un ensemble de capteurs $\mathbb{H}_k^0$ et d'actionneurs $\mathbb{H}_k^1$.
— Un noeud Fog peut avoir la capacité de connecter des nœuds IoT à un réseau externe par le biais d'un équipement de point d'accès (AP) et dispose d'une capacité de calcul et de stockage par le biais d'un serveur dit cloudlet Server (CS).
— Un noeud Cloud est défini comme un serveur de centre de données avec une capacité de calculs supposée illimitée.

3.3.2 Les liens réseau

Chaque lien réseau $ln_{kk'} \in \mathbb{L}$ entre deux nœuds m_k et $m_{k'}$ a les caractéristiques suivantes :

— $nt_{kk'}$: représente la technologie du lien.
— $bw_{kk'}$: représente la bande passante qui est définie par le nombre d'octets par seconde qu'un lien physique est capable d'envoyer de sa source à sa destination. Elle est mesurée en MB/s (MBps).
— $lc_{kk'}$: représente la latence mesurée en milliseconde (ms) et qui est le temps écoulé entre l'envoi et l'arrivée d'un paquet de données de la machine m_k à la machine $m_{k'}$. La latence dépend de l'état du réseau au moment où le paquet est envoyé.

$ln_{kk'}$ peut être un lien direct physique de m_k à $m_{k'}$ ou être composé de plusieurs liens. Si $ln_{kk'}$ est un lien composé, nous supposons que l'algorithme de routage donne un chemin optimal $\Delta_{ln_{kk'}}$ composé des liens physiques.

Si l'on considère l'ensemble $ln_{qp} \in \Delta_{ln_{kk'}}$ de liens physiques de $ln_{kk'}$, la bande passante et la latence de $ln_{kk'}$ sont respectivement définies par les équations (3.5) et (3.6).

$$bw_{kk'} = \min_{ln_{qp}} \in \Delta_{ln_{kk'}}(bw_{qp}) \tag{3.5}$$

$$lc_{kk'} = \sum_{ln_{qp} \in \Delta_{ln_{kk'}}} lc_{qp} \tag{3.6}$$

Les variables et leurs descriptions sont récapitulées dans le tableau 3.3.

	Variable	Type	Description
Indices	k	$[0, M-1]$	Indice des machines physiques.
	u	$[0, M-1]$	Indice des objets IoT.
Infrastructure	$\mathbb{M}$	Ensemble	Ensemble des machines physiques de l'infrastructure Fog (IoT, Fog ou Cloud).
	$\mathbb{U}$	Ensemble	Ensemble des objets IoT.
	$\mathbb{F}$	Ensemble	Ensemble des noeuds de type Fog.
	$\mathbb{C}$	Ensemble	Ensemble des noeuds Cloud.
	$\mathbb{L}$	Ensemble	Ensemble des liens réseau de l'infrastructure Fog.
Machine physique	m	Symbole	Une machine physique (IoT, Fog ou Cloud).
	Ω_k^{max} et Ω_k^t	Vecteur	Vecteurs des capacités maximale et disponible à l'instant t de la machine m_k.
	cpu_k^{max} et cpu_k^t	$\mathbb{R}$	Les capacités CPU maximale et disponible à l'instant t de la machine m_k en MI.
	mem_k^{max} et mem_k^t	$\mathbb{R}$	Capacité mémoire vive maximale et disponible à l'instant t de la machine m_k.
	$disk_k^{max}$ et $disk_k^t$	$\mathbb{R}$	Capacité de stockage maximale et disponible à l'instant t de la machine m_k.
	pc_k^{min}	$\mathbb{R}$	Puissance électrique de calcul minimale consommée par la machine m_k (machine au repos).
	pc_k^{max}	$\mathbb{R}$	Puissance électrique de calcul maximale consommée par la machine m_k.
	pn_k^{min}	$\mathbb{R}$	Puissance électrique de communication minimale consommée en moyenne par toutes les interfaces de la machine m_k.
	pn_k^{max}	$\mathbb{R}$	Puissance électrique de communication maximale consommée par toutes les interfaces de la machine m_k.
	c_k^{max}	$\mathbb{R}$	Capacité maximale de transmission de la machine m_k.
	$\mathbb{I}_k$	Ensemble	Ensemble des interfaces de communication de la machine m_k.
Lien réseau	$ln_{kk'}$	Symbole	Un lien réseau physique ou virtuel.
	$nt_{kk'}$	Symbole	Technologie réseau du lien entre les machines m_k et $m_{k'}$.
	$bw_{kk'}$	Symbole	Bande passante maximale du lien entre les machines m_k et $m_{k'}$..
	$lc_{kk'}$	$\mathbb{R}$	Latence du lien entre les machines m_k et $m_{k'}$.

TABLE 3.3 – Résumé des variables de l'infrastructure Fog.

3.4 Formulation du problème de placement de services

Nous considérons l'intervalle $[t_i, t_f]$ subdivisé en T pas de temps ainsi qu'un ensemble $\mathbb{M} = \{m_0, .., m_k, .., m_{M-1}\}$ de noeuds Fog et un ensemble $\mathbb{A}^0 = \{a_0, .., a_i, .., a_{A-1}\}$ d'applications IoT arrivées à l'instant $t_i = 0$. $N = \sum_{a_i \in \mathbb{A}} n_i$ est le nombre de services de toutes les applications de $\mathbb{A}^0$. On considère $m_u \in \mathbb{U} \subset \mathbb{M}$ l'ensemble des noeuds IoT utilisateurs des applications dans $\mathbb{A}$. Un noeud IoT $m_u \in \mathbb{U}$ a un ensemble d'applications $\mathbb{A}_u^t$. $\mathbb{N}_u^t$ est l'ensemble de tous les services de toutes les applications utilisées par m_u et demandées à l'instant t.

On cherche à placer les services des applications IoT dans l'infrastructure Fog tout en minimisant une fonction objectif $F : \mathbb{N}^N \longmapsto \mathbb{R}$ sur l'intervalle de temps $[t_i, t_f]$. La fonction F est une combinaison de la consommation énergétique de l'infrastructure $f_1(t_i, t_f)$ et du nombre de violations de délai des applications $f_2(t_i, t_f)$ (voir 3.4.3 et équation (3.16)).

Soit x_{ijk}^u, une variable de décision définie par l'équation (3.7) comme suit :

$$x_{ijk}^u = \begin{cases} 1, & \text{si le service } s_j^i \text{ de l'application } a_i \text{ demandé par le noeud} \\ & m_u \text{ est placé sur la machine } m_k \\ 0, & \text{sinon} \end{cases} \tag{3.7}$$

3.4.1 La consommation énergétique

Le premier objectif considéré et défini par l'équation (3.8), est la somme de la consommation énergétique des parties réseau et calcul de l'infrastructure Fog pendant T pas de temps.

Pareillement à plusieurs autres travaux [Beloglazov 2012], [Gschwandtner 2014] et étant donnée sa simplicité d'utilisation et d'interprétations nous proposons un modèle de consommation énergétique linéaire en fonction des taux d'utilisation du processeur et des interfaces réseau d'une machine. D'autres modèles tels que celui proposé par [Xu 2014a] peuvent être utilisés avec les approches de résolutions proposées.

$$f_1(t_i, t_f) = f_C(t_i, t_f) + f_N(t_i, t_f) \tag{3.8}$$

La consommation énergétique du calcul sur l'intervalle $[0, T-1]$ correspondant à T pas de temps est estimée selon l'équation (3.9).

$$f_C(0, T-1) = \sum_{t=0}^{T-1} \sum_{m_u \in \mathbb{U}} \sum_{a_i \in \mathbb{A}_u^t} \sum_{s_j^i \in \mathbb{S}_i} \sum_{m_k \in \mathbb{M}} x_{ijk}^u \alpha_{ijk}^u \gamma_k^c \tag{3.9}$$

α_{ijk}^u, défini par l'équation (3.10), représente le temps de calcul du service s_j^i

de l'application a_i associé au noeud IoT m_u et placé sur la machine physique m_k. Ce temps dépend de la capacité de calcul de la machine physique cpu_k^{max} et du service s_j^i en MIPS.

$$\alpha_{ijk}^u = \frac{inst_j^i + \sum_{u\in\mathbb{U}} \sum_{a_i\in\mathbb{A}^0} \sum_{s_l^i\in\mathbb{S}_i - s_j^i} inst_l^i x_{ilk}^u}{cpu_k^{max}} \tag{3.10}$$

γ_k^c définit la puissance électrique de calcul de la machine m_k. Elle représente la différence entre les puissances maximale et minimale de la machine. $\gamma_k^c = pc_k^{max} - pc_k^{min}$.

En supposant que les communications sont uniformément distribuée sur un pas temps, la consommation énergétique de la communication est estimée par l'équation (3.11) comme suit :

$$f_N(t_i, t_f) = \sum_{t=t_i}^{t_f} \sum_{m_u\in\mathbb{U}} \sum_{a_i\in\mathbb{A}_u^t} \sum_{s_j^i\in\mathbb{S}_i} \sum_{m_k\in\mathbb{M}} x_{ijk}^u x_{ij'k'}^u \beta_{ijj',kk'}^u \gamma_{kk'}^n \tag{3.11}$$

$\beta_{ijj',kk'}^u$ est le temps de communication entre les services s_j et $s_{j'}$ de l'application a_i associés au noeud IoT m_u et placés respectivement sur les noeuds m_k and $m_{k'}$. Ce temps dépend de la latence et de la bande passant du lien réseau entre les machines m_k et $m_{k'}$ ainsi que de la taille des données échangées.

$$\beta_{ijj',kk'}^u = \frac{1}{T}\frac{data_{jj'}^i}{bw_{kk'}} + lc_{kk'} \tag{3.12}$$

$\gamma_{kk'}^n$ est la puissance électrique de communication entre les noeuds m_k et $m_{k'}$ et est définie comme suit : $\gamma_{kk'}^n = pn_k^{max} + pn_{k'}^{max} - pn_k^{min} - pn_{k'}^{min}$.

3.4.2 Violations des délais

La seconde métrique est une métrique de qualité de service (QoS) dont le calcul est présenté par l'équation (3.13). L'équation présente le calcul du nombre moyen de violations de délais dans l'ensemble d'application à placer.

$$f_2(0, T-1) = \frac{1}{T}\sum_{t=0}^{T-1} \sum_{u\in\mathbb{U}} \sum_{a_i\in\mathbb{A}} w_i^u(t) \tag{3.13}$$

$w_i^u(t)$ comptabilise une violations de délai pour chaque pair d'applications/objets et est définie par l'équation (3.14).

$$w_i^u(t) = \begin{cases} 1, & \text{si } d_i^u(t) > d_i^{max} \\ 0, & \text{sinon} \end{cases} \tag{3.14}$$

Pour chaque application a_i et objet m_u à un instant t, le temps de réponse de l'application pour l'objet m_u est défini par l'équation (3.15).

$$\begin{aligned} d_i^u(t) = & \sum_{j \in \mathbb{S}_i} \sum_{k \in \mathbb{M}} x_{ijk}^u \alpha_{ijk}^u \\ & + \sum_{j,j' \in \mathbb{S}_i} \sum_{k,k' \in \mathbb{M}} x_{ijk}^u x_{ij'k'}^u \beta_{ijj',kk'}^u \end{aligned} \tag{3.15}$$

3.4.3 Fonction objectif

Les deux objectifs sont regroupés dans une seule formule, définie par l'équation (3.16). Le but est de pouvoir utiliser une approche d'optimisation mono-objectif, ce qui est recommandé par certain travaux [Pires 2015a], [Ihara 2015], [Chamas 2017]. Le calcul du front de Pareto peut être pénalisant et augmente considérablement les temps de résolution et la consommation des ressources de stockage. Nous proposons une formule un peu différente des sommes pondérées que nous trouvons habituellement dans la littérature. Notre objectif avec cette formule et de donner plus de poids à la minimisation de la consommation d'énergie. L'équation (3.16) présente la fonction objectif globale à minimiser sur un intervalle à T pas de temps

$$F(0, T-1) = f_1(0, T-1)(1 + f_2(0, T-1)) \tag{3.16}$$

3.5 Approches de résolution

Le problème du placement de services fait partie de la catégorie de problèmes de décision NP-difficile [Pires 2015b]. De ce fait, nous proposons une approche méta-heuristique pour résoudre en des temps raisonnables le problème détaillé en section 3.4. Les approches choisies ont prouvé leur efficacité dans les environnements Cloud et Fog [Liu 2016a], [Pandey 2010], [Skarlat 2017a], [Skarlat 2017b] avec différents critères d'optimisation mais n'ont pas été étudiées dans un environnement IoT-Fog. Cette section explique les trois approches utilisées pour résoudre le problème de placement de service hors ligne.

3.5.1 Optimisation Discrète par essaims particulaires (DPSO)

3.5.1.1 Principe de l'optimisation par essaims particulaires (PSO)

L'optimisation par essaims particulaires [Kennedy 1995] est une méthode d'optimisation semi-stochastique à population appelée essaim. Cette méthode s'inspire du comportement individuel et collectif au sein de groupes d'animaux

	Variable	Type	Description
Problème	x^u_{ijk}	Binaire	Variable de décision.
	$F(t_i, t_f)$	$\mathbb{R}$	Fonction objectif globale sur l'intervalle $[t_i, t_f]$.
	T	$\mathbb{N}$	Nombre total de pas de temps.
	t_i	$\mathbb{N}$	Début de la fenêtre de temps.
	t_f	$\mathbb{N}$	Fin de la fenêtre de temps.
Énergie	$f_1(t_i, t_f)$	$\mathbb{R}$	Fonction objectif de l'énergie consommée par le calcul et le réseau dans $[t_i, t_f]$.
	$f_C(t_i, t_f)$	$\mathbb{R}$	Consommation énergétique due aux calculs dans $[t_i, t_f]$.
	$f_N(t_i, t_f)$	$\mathbb{R}$	Consommation énergétique due aux communications dans $[t_i, t_f]$.
	γ^c_k	$\mathbb{R}$	Constante de la puissance électrique consommée pour les calculs de la machine m_k.
	$\gamma^n_{kk'}$	$\mathbb{R}$	Constante de la puissance électrique consommée pour les communications de la machine m_k.
QoS	$f_2(t_i, t_f)$	$\mathbb{R}$	Fonction objectif du nombre moyen de violations de délai.
	$w^u_i(t)$	Binaire	Violations de délai de l'application a_i utilisée par l'objet m_u à l'instant t.
	$d^u_i(t)$	$\mathbb{R}$	Temps de réponse effectif de l'instance de l'application a_i pour l'objet m_u à l'instant t.
	α^u_{ijk}	$\mathbb{R}$	Temps d'exécution du service s^i_j de l'application a_i placé dans la machine m_k.
	$\beta^u_{ijj',kk'}$	$\mathbb{R}$	Temps de communication entre les services s^i_j et $s^i_{j'}$ placés respectivement dans les machines m_k et $m_{k'}$

TABLE 3.4 – Résumé des variables du problème de placement statique.

sociaux devant parcourir de grandes distances et optimiser leurs déplacements en termes d'énergie dépensée. Nous citons à titre d'exemple la formation en V dans le processus migratoire des oiseaux. L'algorithme a été initialement pensé pour résoudre les problèmes d'optimisation à variables continues (variables de décision) et a été rapidement adapté pour les problèmes de décision binaires et discrets à l'instar des problèmes de placement (solution prenant uniquement certaines valeurs). Kennedy et Eberhart ont introduit la version Binary Particle Swarm Optimization (BPSO) [Kennedy 1997]. Le PSO manipule une population d'individus appelés particules. Chaque particule représente une solution (réalisable ou non) du problème. Ces particules parcourent l'espace de recherche et interagissent entre elles pour trouver une solution acceptable en un temps raisonnable. Le comportement de recherche de la particule, qui résulte de ses variations de direction et de vitesse dans l'espace de recherche, est influencé par une composante dite cognitive φ_1 et une composante dite sociale φ_2, que nous allons détailler par la suite.

Quelque soit la version du PSO modélisée (binaire, discrète ou continue), une particule est toujours définie avec les paramètres suivants :

1. Sa position dans l'ensemble de recherche.
2. Sa vitesse, lui permettant de se déplacer d'une solution à une autre. Au cours des itérations, les particules changent de positions et évoluent en fonction des particules voisines (influence sociale) et de leurs expériences passées (influence cognitive).
3. Son voisinage, qui regroupe un ensemble de particules interagissant directement avec la particule (s'échangeant les informations de positionnement et la valeur de leurs fitness).

A tout instant, chaque particule peut connaître :

1. La meilleure position déjà visitée, "Personal best" (Pb).
2. La position du meilleur voisin de l'essaim, "Neighbor best" (Nb).
3. La valeur de la fonction objectif associée à sa position actuelle. A chaque itération, cette valeur est comparée à la meilleure valeur obtenue auparavant.

3.5.1.2 Modélisation Binaire pour le problème de placement de service (BPSO)

Dans la modélisation binaire du PSO, la position de la particule est définie par une matrice binaire qui représente une solution du problème. La vitesse de la particule appelée matrice vélocité, est une matrice de probabilité où chaque élément $v^{it}_{ij,k}$ représente la probabilité qu'un service s^i_j soit placé dans une machine m_k dans la solution de l'itération it. La fonction sigmoid $sig(.)$,

définie par l'équation (3.19), assure que les éléments de la matrice vélocité restent dans l'intervalle $[0, 1]$.

— Toute particule Pr de l'essaim et sa vélocité V, définies respectivement comme des matrices, sont mises à jour à chaque itération it selon les équations (3.17) et (3.18) comme suit :

$$V^{it+1} = \omega V^{it} + \varphi_1 \omega_1^{it}(Pb - Pr^{it}) + \varphi_2 \omega_2^{t}(Nb - Pr^{it}) \tag{3.17}$$

$$Pr^{it+1}(i,j) = \begin{cases} 1, & \text{si } rand() \geq sig(v^{it+1}(ij,k)) \\ 0, & rand() < sig(v^{it+1}(ij,k)) \end{cases} \tag{3.18}$$

$$sig(v^{it+1}(i,j)) = \frac{1}{1 + \exp^{-v^{it+1}(i,j)}} \tag{3.19}$$

— Les paramètres φ_1 and φ_2 sont définis respectivement comme les constantes cognitive et sociale qui modulent la magnitude du prochain mouvement de la particule vers la meilleure position qu'elle a déjà découvert ou vers la meilleure position donnée par les particules avec laquelle elle est autorisée à communiquer. Selon les travaux de la littérature [Marini 2015], le valeurs de φ_1 et φ_2 doivent variées dans l'intervalle $[0, 4]$. Il a par ailleurs été montré dans [Kennedy 1995] que $\varphi_1 = \varphi_2 = 2$ sont les meilleures valeurs dans de nombreux problèmes.

— Les paramètres Pb et Nb représentent respectivement la meilleure position visitée par la particule Pr et la meilleure position visitée par ses voisines.

— Il existe deux approches qui permettent d'éviter le phénomène oscillatoire des particules (particule piégée dans une partie de l'espace qui oscille entre des solutions déjà explorées) : (1) la limitation des valeurs de la vitesse entre une borne minimale et une borne maximale, $v(ij,k) \in [-V_{max}, V_{max}]$ ou (2) l'utilisation d'un coefficient de restriction ω pour pondérer les éléments de la matrice vélocité. Cette limitation évite aux particules de se déplacer trop rapidement d'une région à une autre. Selon la stratégie choisie, la valeur ω peut rester constante ou évoluer au cours des itérations [Eberhart 2000], [Marini 2015].

— Les paramètres ω_1 et ω_2 sont deux matrices dont les éléments $\omega_1(ij,k), \omega_2(ij,k)$ sont choisis aléatoirement entre $[0, 1]$, afin d'introduire un comportement aléatoire dans le processus de recherche de la particule.

3.5.1.3 Modélisation Discrète pour le problème de placement de service (DPSO)

Les positions des particules dans la version discrète du PSO prennent leurs valeurs dans $\mathbb{N}$. Pour le problème de placement de services défini en section

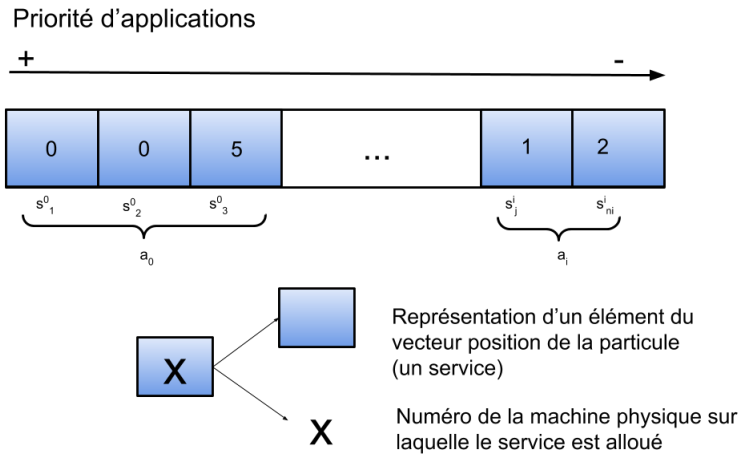


FIGURE 3.3 – Vecteur position d'une particule dans le DPSO où un élément du vecteur indexé par le service donne le numéro de la machine physique sur laquelle il sera placé.

3.4, le DPSO est plus intéressant pour une utilisation moins intensive de la mémoire lors de son implémentation. Ce dernier permet d'éviter le problème de saturation des valeurs (liés à la fonction sigmoid) et de ce fait, réduit la perte d'information dans la matrice vélocité. Il est également important de préciser que le passage du DPSO au BPSO est simple et rapide [Izakian 2009].

Considérons un espace de recherche discret N-dimensionnel, N étant le nombre de services des applications à placer et $\mathbb{P}$ l'essaim particulaire de taille P_{size}. Chaque particule $Pr \in \mathbb{P}$ évolue sur un nombre it_{max} d'itérations en cherchant à minimiser la fonction objectif définie par l'équation (3.16).

(1) Représentation de la position et la vitesse d'une particule

$Pr^{it} \in \mathbb{N}^N$ donne la position de la particule Pr à l'itération it, qui dans le cas discret est définie par un vecteur de taille N dont les valeurs pr_{ij}^{it} varient dans $[0, M-1]$, $\forall j \in [0, N-1]$, $\forall i \in [0, A-1]$. Par exemple, dans la figure 3.3 qui donne une représentation de Pr^{it}, le service s_3^0 est placé dans la machine physique 5.

$pr_{ij}^{it} = k$ implique qu'à l'itération it le service s_j^i est placé sur la $k^{ème}$ machine m_k avec $k \in [0, M-1]$.

Cela peut également s'écrire de la manière suivante en considérant la représentation matricielle de la particule avec la matrice Mx à l'itération it :

$pr_{ij}^{it} = k \Longleftrightarrow Mx^{it}(ij, k) = 1 \wedge \forall l \in \{0, .., N-1\} - \{k\}, Mx^{it}(ij, l) = 0.$

La vélocité de la particule dans un cas discret est définie par une matrice V qui est de dimensions NXM avec N le nombre total de services des applications et M le nombre de machines (Fog, Cloud et IoT). La figure 3.4 donne

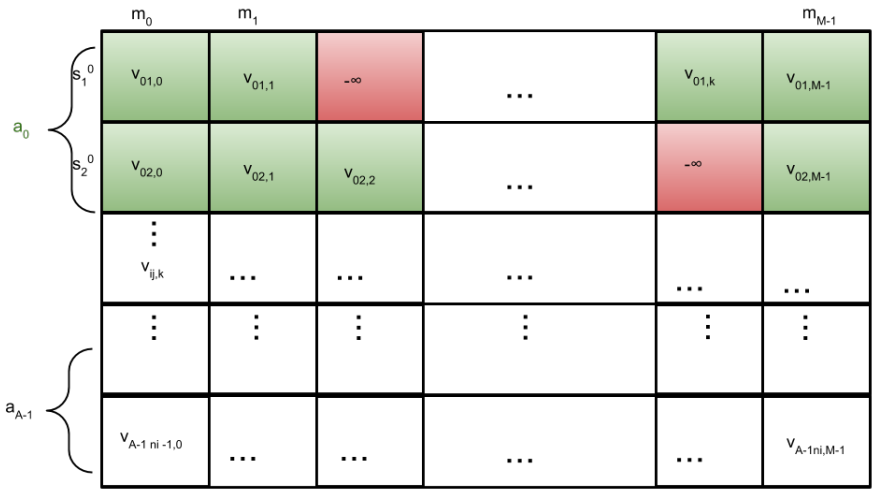


FIGURE 3.4 – Matrice vélocité d'une particule dans le DPSO où chaque ligne donne les chances de placement du service concerné par rapport à chaque machine physique.

une représentation de la matrice V.

Chaque élément $v(ij,k) \in \mathbb{R}$ exprime les chances pour que le service s_j^i soit placé sur la machine m_k.

(2) Équation du mouvement d'une particule

Après le calcul de la nouvelle vélocité, le nouveau vecteur position de la particule est déduit avec l'équation (3.20)

$$pr_{ij}^{it} = k \iff v^{it}(ij,k) = \max_{\forall l \in [0,M-1]} \left\{ v^{it}(ij,l) \right\} \tag{3.20}$$

Le différence entre le DPSO et sa version binaire se trouve dans la représentation de la position de la particule qui est un vecteur dans le DPSO et une matrice dans le BPSO. La représentation vectorielle permet de réduire la consommation de la mémoire. De plus, le DPSO accepte des valeurs non binaires pour la matrice vélocité et ne se repose plus sur la fonction sigmoid pour normaliser ses valeurs. Le problème de la saturation de la sigmoid conduit à une perte d'informations considérable. Le DPSO introduit également la valeur de la fitness dans l'équation de calcul de la nouvelle vélocité et n'utilise pas les indices des particules. Le paramètre ω gère le degré d'influence de l'ancienne matrice vélocité pour calculer la nouvelle matrice. Selon les travaux de la littérature [Marini 2015], [Kennedy 1995], [Eberhart 2000], la valeur du paramètre doit varier dans l'intervalle $[\omega_{min}, \omega_{max}]$ avec $\omega_{min} = 0.4$ and $\omega_{max} = 0.9$. Les stratégies permettant une décroissance linéaire du paramètre au court des

itérations ont montré les meilleurs résultats pour la convergence de l'algorithme.

En considérant les informations précédentes, nous proposons de mettre à jour ω selon la formule proposée par [Xin 2009] et qui est définie dans l'équation (3.21) comme suit :

$$\omega^{it} = \omega_{max} - \frac{(\omega_{max} - \omega_{min})}{it_{max}} * it \tag{3.21}$$

(3) Validité de la position de la particule (de la solution)

La contrainte due à la topologie physique ou à des besoins définis par l'utilisateur peut réduire l'espace des solutions réalisables pour un service donné s_j^i.

Chaque service $s_j^i \in \mathbb{S}_i$ a un ensemble de machines valides où ce dernier peut s'exécuter. Pour assurer cette contrainte avec le DPSO et donc garantir que les machines non valides pour un service donné ne seront jamais choisies, on affecte la valeur $-\infty$ dans les cases concernées de la matrice vélocité.
Si le service s_j^i ne peut être placé dans la machine m_k alors nous avons l'équation (3.22). Par exemple, dans la matrice vélocité de la figure 3.4 le service s_1^0 ne peut pas être placé sur la machine m_2.

$$\forall Pr \in \mathbb{P}, \forall i \in [0, A-1], \forall it \in [0, it_{max} - 1] \Rightarrow v^{it}(ij, k) = -\infty \tag{3.22}$$

(4) Population initiale

Initialement, les particules de l'essaim $\mathbb{P}$ sont uniformément distribuées dans l'espace de recherche.

$\forall s_j^i \in \mathbb{S}_i,\ pr_{ij}^0$ doit être choisi uniformément dans l'ensemble des machines valides pour le service s_j^i. Les éléments des matrices vélocités V^0 sont initialisées par lignes (services) à 1 pour les machines valides et à $-\infty$ pour les machines non valides pour le service en question.

(5) Topologie du voisinage

La topologie du voisinage d'une particule définit les communications autorisées dans l'essaim et qui vont influencer les caractéristiques exploratoires de l'algorithme ainsi que sa vitesse de convergence. Dans sa version initiale [Kennedy 1995], l'algorithme du PSO permet à toutes les particules de communiquer entre elles pour échanger leurs meilleures positions et déterminer la meilleure solution de tout l'essaim. Cette approche de communication globale peut bloquer la recherche dans un optimum local. Pour pallier ce problème, la population peut être divisée en sous-ensembles où la communication est autorisée uniquement entre les particules d'un même ensemble

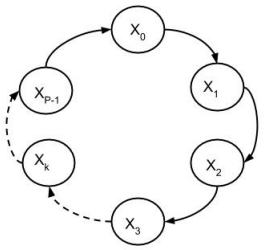


FIGURE 3.5 – Topologie du voisinage dans l'essaim

[Muñoz Zavala 2013]. Les travaux de la littérature identifient deux catégories de voisinage :

1. Le voisinage géométrique où les voisins de chaque particule sont calculés selon une certaine métrique comme par exemple la valeur de la fonction fitness.
2. Le voisinage social qui peut être défini à travers les indices des particules. Ce dernier à plusieurs reprises a donné de meilleurs solutions comparé à un algorithme de voisinage géométrique [Marini 2015], [Kennedy 1995], [Muñoz Zavala 2013].

En nous basant sur la littérature, nous avons défini un voisinage social circulaire de deux particules représenté en figure 3.5. Les étapes du DPSO sont présentées avec le pseudo algorithme 1.

3.5.1.4 Optimisation Discrète multi-objective par essaim particulaires (MDPSO)

La méthode du DPSO manipule les deux objectifs précédents, consommation d'énergie définie par l'équation (3.8) et nombre de violations de délai défini par l'équation (3.13), dans une seule formule qui est combinaison des deux objectifs précédents. Utiliser l'approche mono-objectif pour un problème d'optimisation à plus d'un objectif peut conduire à ignorer des solutions intéressantes pour un des objectifs ou plusieurs d'entre eux. Dans cette section, nous présentons la version muti-objectifs du DPSO (MDPSO) dont les étapes sont décrites par le pseudo algorithme 2.

Le MDPSO utilise les approches "Fast non dominant sorting" et "Crowding assignment" décrits dans les travaux [Li 2003], [Deb 2002] et [Coello 2004]. Avant de définir ces approches, nous devons définir quelques concepts propres aux méthodes d'optimisation multi-objectifs : (1) la dominance et (2) le Front de Pareto.

Algorithm 1 DPSO pour le placement de services IoT

1: Initialiser le nombre d'itérations $it = 0$.
2: Générer uniformément la population initiale $\mathbb{P}^0$ et l'ensemble des vélocités associées $\mathbb{V}^0$.
3: **while** $it < it_{max}$ **do**
4: **for all** $Pr^{it} \in \mathbb{P}^{it}$ **do**
5: Mettre à jour la matrice vélocité V^{it} de la particule, selon la formule de l'équation(3.17).
6: Déduire le nouveau vecteur position Pr^{it}, selon la formule de l'équation (3.20).
7: Mettre à jour l'information sur la meilleure solution trouvée par la particule Pb si $(F(Pr^{it}) < F(Pb))$.
8: **for all** Voisins de la particule Pr^{it} **do**
9: Mettre à jour Nb si $(F(Pr^{it}) < F(Nb))$
10: **end for**
11: **end for**
12: Incrémenter it.
13: **end while**
14: Retourner la meilleure particule, celle qui minimise la fonction donnée en (3.16).

(1) Dominance

On dit qu'une particule A domine une particule B si pour toutes les fonctions objectifs à minimiser (ou à maximiser) les valeurs des fitness de A sont meilleures que celles de B ou alors si A est meilleure pour un seul objectif et pour le reste des objectifs elles sont équivalentes. Les particules A et B ne se dominent pas, si chaque solution est meilleure par rapport à l'autre pour un seul objectif uniquement ou si elles sont égales pour touts les objectifs.

(2) Front de Pareto

Une optimalité ou une efficacité de Pareto est un état où on ne peut pas améliorer une solution sans dégrader une autre. Le Front ou l'ensemble de Pareto en optimisation est l'ensemble de solutions qui ne se dominent pas entre elles et qui dominent toutes les autres solutions. En utilisant des méthodes de résolutions approchées nous obtenons un Front de Pareto dit approximé.

(3) Le tri rapide par non dominance Le "non dominanted sorting algorithm" est un algorithme qui trie les solutions (la population de particules, de chromosomes ou autre) selon leur niveau de dominance. L'algorithme se fait en plusieurs itérations où à chaque itération les particules sont comparées entre elles afin de déduire les solutions dominantes (Front de Pareto approximé). Ces solutions sont retirées de l'ensemble et sont archivée. A la prochaine itération l'algorithme effectue les mêmes comparaisons et déduit le prochain ensemble de solutions dominantes. L'algorithme continue ce processus jusqu'à ce qu'il

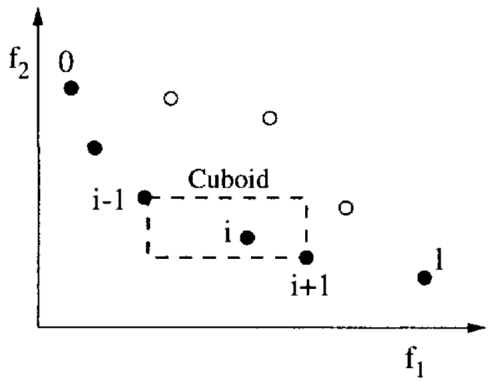


FIGURE 3.6 – Estimation du bruit autour d'une particule selon [Deb 2002].

ne reste plus de solutions à comparer. Le première ensemble est appelé "ensemble de solutions dominantes de niveau 1", le deuxième ensemble est appelé " ensemble de solutions dominantes de niveau 2" et ainsi de suite.

Le "Fast non dominated sorting algorithm" est une optimisation du nombre de comparaisons entre les solutions pour réduire la complexité de l'algorithme décrit précédemment. Cette optimisation est présentée dans [Deb 2002].

(4) Le bruit autour d'une solution La bruit ou la densité autour d'une solution (particule) permet d'avoir une estimation du nombre de solutions qui sont très proches d'elle (et qui sont du même Front). Pour cela, on doit considérer le cuboid le plus grand contenant la particule seule et qui n'inclut aucune autre solution. Le taille moyenne d'un côté du cuboid est appelée "crowding distance". Le cuboid est construit à partir des deux solutions les plus proches de la particule de chaque côté du Front. Le figure 3.6 illustre ce processus.

(5) Représentation de la position et de la vitesse d'une particule avec le MDPSO Les définitions et les représentations des particules et des vélocités restent similaires à l'approches DPSO.

La formule de mise à jour des vélocités est différente du DPSO car on a deux fonctions objectif séparées et présentées dans l'équation (3.23). Nous avons donc un paramètre de contrôle cognitif et social pour chaque objectif. φ_{11} et φ_{12} représentent respectivement le paramètre cognitif et social lié au premier objectif f_1 (l'énergie) et φ_{21} et φ_{22} représentent respectivement le paramètre cognitif et social lié au second objectif f_2 (les violations de délai). Pareillement, nous avons une variable aléatoire associée à chaque objectif et pour chaque aspect, cognitif et social, avec les variables ω_{11}, ω_{12}, ω_{21} et ω_{22}.

Chaque particule a une valeur du bruit autour d'elle "crowding value" et un compteur de "dominance".

Les particules sont ordonnées en plusieurs niveaux de dominance comme défini dans le travail [Deb 2002] et le Front de Pareto estimé (niveau de dominance 1 du trie par dominance présenté plus haut) est sauvegardé dans une archive.

Initialement, les solutions non dominées sont ajoutées à l'archive de Pareto. Après chaque itération, où les étapes sont identiques au DPSO, les particules non dominées de l'essaim sont comparées aux solutions sauvegardées dans l'archive. Les solutions non dominées remplacent les solutions dominées de l'archive. L'archive de Pareto a une capacité maximale de solutions définie par le taille de l'essaim. Si le nombre de particule de l'archive dépasse le nombre maximal, les solutions appartenant aux régions les plus dense selon le critère du crowding distance sont éliminées à chaque itération. La diversité de la population est assurée par l'utilisation du "crowding distance" [Deb 2002].

$$\begin{aligned} v^{it+1}(ij,k) &= \omega^{(it+1)} v^{it}(ij,k) + \varphi_{11}\omega_{11}^{it+1}(ij,k)[f1(Pb^{it}) - f1(Pr^{it})] \\ &+\varphi_{12}\omega_{12}^{it+1}(ij,k)[f1(Nb^{it}) - f1(Pr^{it})] + \varphi_{21}\omega_{21}^{it+1}(i,j)[f2(Pb^{it}) \\ &- f2(Pr^{it})] + \varphi_{22}\omega_{22}^{it+1}(i,j)[f2(Nb^{it}) - f1(Pr^{it})] \end{aligned} \tag{3.23}$$

Algorithm 2 MDPSO pour le placement de services IoT

1: Initialiser le nombre d'itération $it = 0$.
2: Générer uniformément la population initiale $\mathbb{P}^0$ et l'ensemble des vélocités associées V^0.
3: **while** $it < it_{max}$ **do**
4: **for all** $Pr^{it} \in \mathbb{P}^{it}$ **do**
5: Mettre à jour la matrice vélocité V^{it} de la particule Pr^{it}, selon l'équation (3.23).
6: Déduire le nouveau vecteur position Pr^{it} selon l'équation (3.20).
7: Calculer le front de Pareto et mettre à jour la meilleure solution trouvée Pb.
8: **for all** Particule voisine de Pr^{it} **do**
9: Calculer le front de Pareto et mettre à jour Nb.
10: **end for**
11: **end for**
12: Calculer le bruit (Crowding Distance) autour de Pr^{it}.
13: Ordonner la population Pr^t par ordre décroissant du bruit.
14: Incrémenter it.
15: **end while**
16: Retourner la première particule de l'ensemble ordonné.

3.5.2 Algorithme Génétique (GA)

L'algorithme Génétique (GA) est une méta-heuristique semi-stochastique à population appartenant à la catégorie des méthodes évolutionnaires (se basant sur des concepts bioinspirés). Le GA a montré une grande efficacité pour la résolution des problèmes NP-difficiles à variables discrètes [Liu 2016b]. Dans ce qui suit nous présentons l'application des opérateurs génétiques pour notre problème ainsi que les étapes du GA avec le pseudo algorithme 3.

Algorithm 3 Algorithme Génétique pour le placement de services

1: Générer uniformément la population initiale de taille P_{size} et calculer la valeur de la fonction objectif de chaque chromosome définie par l'équation (3.16).
2: **while** $it < \sigma_3$ **do**
3: **while** Nombre de chromosomes génères est inférieur à $\frac{\sigma_2}{2} * P_{size}$ **do**
4: Sur σ_2 pourcentage de la population (choisie aléatoirement) est appliqué le processus de sélection par tournoi one-way et deux chromosomes parents c_1, c_2 sont choisis.
5: Croiser les chromosomes précédemment sélectionnés c_1, c_2 et déduire le chromosome résultant c_3.
6: Appliquer l'opération mutation sur le chromosome résultat c_3 avec une probabilité σ_1.
7: Calculer la fitness de c_3 selon(3.16).
8: Sauvegarder c_3 et incrémenter le nombre de chromosome générés.
9: **end while**
10: Ordonner l'ancienne population et les chromosomes générés selon leur fitness et de garder les P_{size} meilleurs individus pour la prochaine génération.
11: Incrémenter it.
12: **end while**
13: Retourner le chromosome qui minimise (3.16).

(1) Représentation d'une solution (un chromosome)

Pareillement à la modélisation des particules du DPSO et du MDPSO, un chromosome est représenté par un vecteur de taille N indexé par les services des applications à placées. Les applications sont ordonnées par ordre croissant de priorité et les services par ordre croissant des demandes CPU. Un élément du vecteur donne le numéro de la machine physique sur laquelle le service va être placé.

(2) Initialisation (population initiale)

La population initiale de chromosomes est générée uniformément dans l'espace de recherche.

(3) Les opérateurs

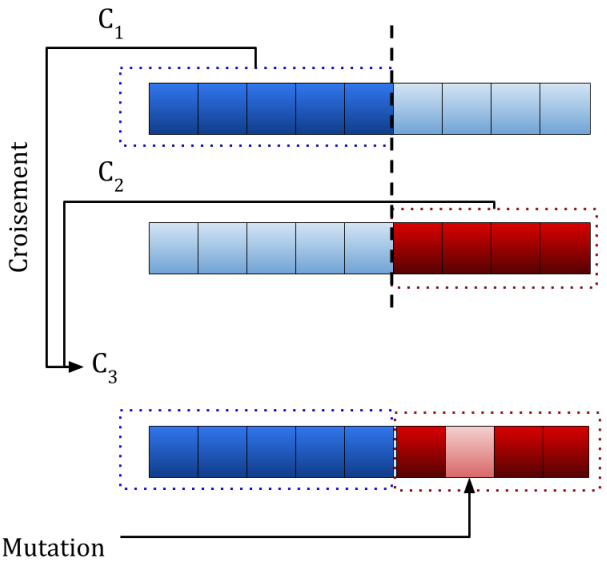


FIGURE 3.7 – Illustration des processus de croisement et de mutation

(3.1) Sélection

Sur σ_2% des meilleurs individus deux chromosomes sont sélectionnés aléatoirement. Cette opération est répétée jusqu'à avoir $\frac{\sigma_2}{2} * P_{size}$ de chromosomes valides générés par le croisement.

(3.2) Croisement

Après la sélection d'un couple de chromosomes c_1 et c_2. Le premier parent est celui ayant la meilleure valeur de la fitness. En sachant que les applications sont classées par priorité et les services par taille de demande CPU. Le chromosome fils c_3 est créé à partir des premiers gènes de c_1 et des derniers gènes de c_2 (50% des gènes de chaque parent). La validité de chaque chromosome est vérifiée après le croisement.

(3.3) Mutation

Chaque chromosome issu d'une opération de croisement a une probabilité σ_1 de subir une mutation sur l'un de ses gènes sauf si la fitness du chromosome généré est la meilleure de toute la population alors la probabilité de mutation est à 0. La mutation s'effectue en choisissant aléatoirement un service et un noeud dans l'ensemble des noeuds valides (condition de capacité et d'emplacement).

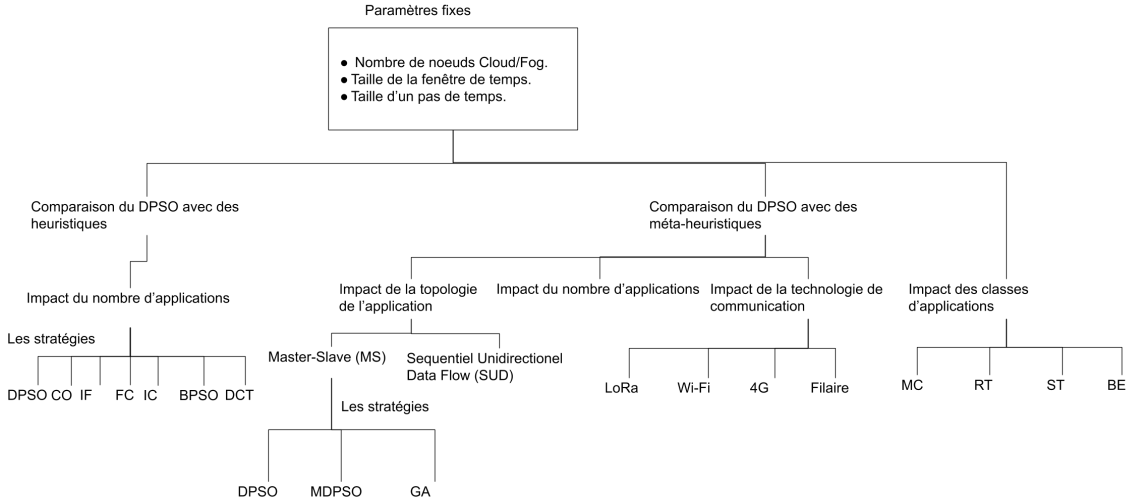


FIGURE 3.8 – Illustration de la méthodologie des expérimentations.

(3) Population de la prochaine génération

Sur la population composée des chromosomes fils et des parents, les meilleurs individus selon la valeur de la fitness sont sélectionnés pour la prochaine génération.

(4) Configuration

Les paramètres du GA ont été choisis expérimentalement selon ses meilleures performances. Le taux de mutation $\sigma_1 = 0.02$, le taux de crossover $\sigma_2 = 0.8$, la taille de la population $P_{size} = 30$ et le nombre de génération $\sigma_3 = 40$.

3.6 Résultats

Les expérimentations ont été faites en deux parties. La première partie compare les performances de la méthode DPSO avec des heuristiques First Fit simples qui utilisent certaines couches de l'infrastructure de calcul et une heuristique de la littérature que nous allons définir par la suite. La deuxième partie des expérimentations compare la méthode DPSO avec des stratégies de placement plus élaborées MDPSO et GA. La méthodologie des expérimentations est présentée sur la figure 3.8. Les expériences ont été conduites sur une machine Intel core i7-7700 CPU@3.60GHz x8 et les méthodes sont implémentées en JAVA dans le simulateur iFogSim [Gupta 2016].

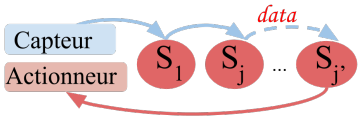


FIGURE 3.9 – Topologie des applications à flux séquentiel unidirectionnel (SUD) : le premier service reçoit les données des capteurs et le dernier communique avec les actionneurs.

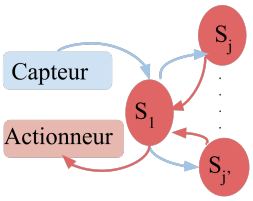


FIGURE 3.10 – Topologie des applications Master-Slave (MS) : le service maître est le seul service qui communique avec la source et envoie ensuite des instructions aux services esclaves. Après avoir reçu la réponse de tous les services esclaves, le service maître envoie les instructions aux actionneurs.

3.6.1 Test du DPSO et des heuristiques pour le placement de services

Dans cette partie, l'approche Discrete Particle Swarm Optimization (DPSO) et sa version binaire (BPSO) sont comparées aux stratégies suivantes : CloudOnly, FogOnly, IoTFogOnly (IF), IoTCloud (IC), FogCloud (FC) et Dicothomous Modules Mapping (DCT), une méthode de la littérature. L'objectif étant de montrer l'efficacité des méthodes proposées et d'observer l'impact de l'utilisation des noeuds de certaines couches de l'infrastructure de calcul plutôt que d'autres.

Pour une infrastructure de calcul Fog fixe, nous faisons varier le nombre d'applications et nous comparons les différentes approches de placement en considérant la consommation énergétique de l'infrastructure et le nombre de violations de délai dans l'ensemble des applications.

Nous considérons dans nos expérimentations deux types de topologies d'applications données dans [Buyya 2019] et représentées avec les figures 3.9 et 3.10.

Nous considérons l'ensemble d'applications $\mathbb{A}^0$ à l'instant $t_i = 0$ de taille $A \in \{4, 8, 12, 16, 20, 24, 28, 32\}$. Chaque application a_i a un nombre $n_i = 3$ de services de traitement de flux. Nous avons pris la charge de travail moyenne des services en millions d'instructions et la taille des données traversant le réseau en (kB) proportionnellement aux informations trouvées dans les exemples d'applications donnés dans [Buyya 2019]. Chaque application a_i est caractérisée par la topologie de communication entre ses services, ses exigences en

TABLE 3.5 – Caractéristiques détaillées des noeuds de l'infrastructure Fog ($k \in [0, 3]$).

Niveau	Nom	Nombre	Noeud	CPU (MIPS)	RAM (MB)	Bande passante (MBps)	Pc^{max}-Pc^{min} (W)
0	Cloud	4	Cloud	120000	64000	10000	318-145
1	Fog	1	Proxy	60000	8000	10000	169-70
2		2	d0(LoRa) d1(Wired)	6750 6750	1000 1000	10000 10000	10-45 10-45
2		2	d2(Wi-Fi) d3(4G-LTE)	13500 13500	2000 2000	10000 10000	20-70 20-70
3	IoT	16	m0-k m1-k m2-k m3-k	2800 4500 2800 4500	1000 3000 1000 3000	0.25/1/1000/1000 0.25/1/1000/1000 0.25/1/1000/1000 0.25/1/1000/1000	5.1-1.9 6.1-1.1 5.1-1.9 6.1-1.1

qualité de service (QoS) et sa priorité. Nous avons utilisé le tableau 3.1 pour générer un ensemble d'applications IoT de différentes classes qui ont été identifiées dans [Guevara 2017] avec leur priorité associée (0 est la plus haute priorité) et leur délai de réponse maximal. Chaque application interagit avec une paire de capteurs/actionneurs placés de manière aléatoire sur les objets IoT de l'infrastructure. Les capteurs de l'infrastructure envoient simultanément des données toutes les 5 ms. Nous arrêtons la simulation lorsque tous les capteurs ont envoyé 500 paquets de données.

Pareillement aux travaux de [Wang 2018b] et [Csr 2017], nous considérons une infrastructure Fog fixe avec 3 couches de noeuds : (1) Cloud, (2) Proxy et passerelle Fog et (3) couche objets IoT.

Pour le centre de données et le noeud proxy, nous avons utilisé les informations d'un centre de données local composé de 4 noeuds physiques Intel dual Xeon E5 2699 v3 18-cores. Les spécifications processeur ont été récupérées du site Intel et l'évaluation des MIPS résultent du 7-zip benchmark [Intel 2015], [7-cpu].

Pour les noeuds Fog, on simule les caractéristiques des routeurs Cisco IR809 et IR829 [Cisco].

Pour la couche IoT, on considère deux catégories d'équipements : des smartphone Samsung galaxy S5 et des Raspberry pi 3+ B [Halfacree 2018], [Wimmer 2014].

Le simulateur impose une structure arborescente pour représenter l'infrastructure Fog multi-couches décrite par la figure 3.11. Les caractéristiques des noeuds de l'infrastructure sont décrites dans le tableau 3.5. Les technologies d'accès utilisées sont la 4G-LTE et LoRa considérées comme des réseaux d'accès attractifs pour l'IoT et des technologies plus classiques et largement déployées : Wi-Fi et filaire.

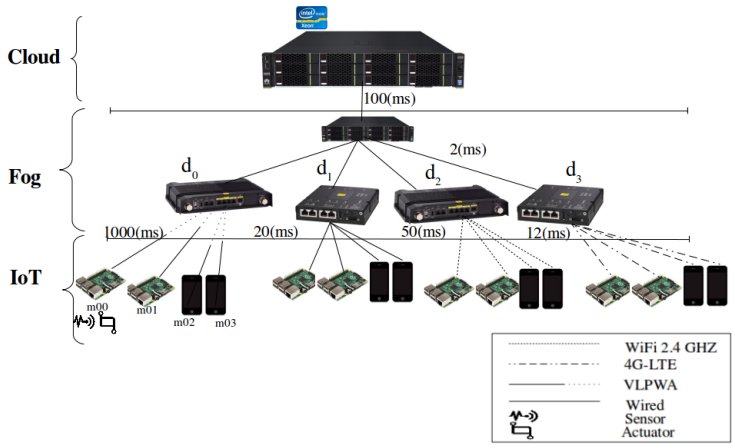


FIGURE 3.11 – Topologie Fog des expérimentations, composée d'une couche Cloud, couche de passerelles Fog et une couche IoT où pour chaque passerelle Fog 4 objets IoT y sont connectés.

3.6.1.1 Les méthodes

Le Discrete Particle Swarm Optimization (DPSO) est comparé à un ensemble de stratégies de placement simples qui ont pour avantage d'avoir un temps de résolution négligeable comparés aux méta-heuristiques. Toutes les heuristiques sont des algorithmes de listes qui considèrent uniquement un sous-ensemble de noeuds de calcul et qui utilisent l'approche First Fit qui consiste à choisir le premier élément trouvé respectant les contraintes du problème pour le placement d'un service.

Pour toutes les heuristiques, les noeuds de calcul et les services sont respectivement ordonnés par ordre croissant des capacités (disponibles) et des demandes CPU.

1. **CloudOnly (CO)** - Cette méthode place tous les services dans les centres de données.
2. **IoTFogOnly (IF)** - Cette méthode considère uniquement les noeuds IoT et Fog, pour chaque application elle essaye de placer tous les services sur le premier noeud le plus proche du noeud utilisateur (en nombre de sauts) ayant les capacités suffisantes (First Fit).
3. **FogCloud (FC)** - Cette méthode place les services d'une même application dans les noeuds Fog et si les capacité des noeuds sont dépassées les services sont placés dans le Cloud.
4. **IoTCloud (IC)** - Cette méthode place les services dans les noeuds IoT et si les capacité des noeuds sont dépassées les services sont placés dans le Cloud.
5. **Dicothomous Module Mapping (DCT)** - Cette méthode de la lit-

térature [Taneja 2017] effectue une recherche dichotomique sur la liste ordonnée des noeuds de calcul pour chaque service.

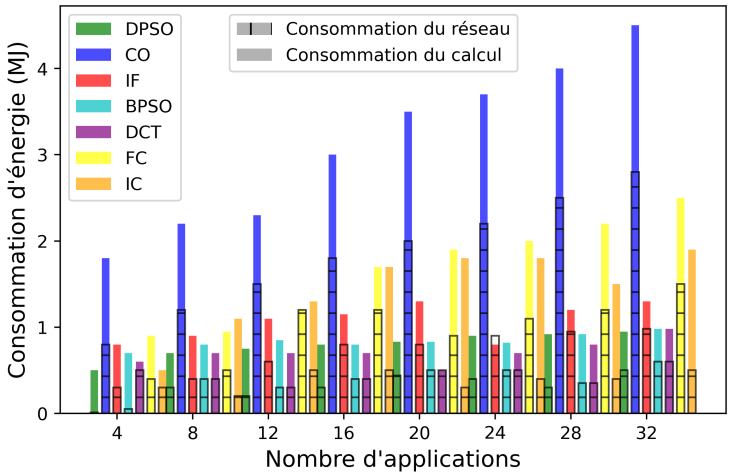
6. **Binary Particle Swarm Optimization (BPSO)** - Utilise la méthode du BPSO décrite dans la section 3.5.1.2.
7. **Discret Particle Swarm Optimization (DPSO)** - Utilise l'algorithme du DPSO présenté dans la section 3.5.1.3. Nous avons conduit un ensemble d'expérimentation avec les paramètres dans les intervalles suivants :
   — Les constantes sociale φ_1 et cognitive φ_2 dans $[1, 4]$.
   — La taille de la population P_{size} dans $[10, 80]$.
   Les valeurs trouvées expérimentalement où le DPSO est le plus performant sont en adéquation avec les valeurs de la littérature [Marini 2015], [Izakian 2009] avec $\varphi_1 = 2$, $\varphi_2 = 2$ et $P_{size} = 40$.

Considérant 50 exécutions indépendantes, les figures 3.12b et 3.12a présentent respectivement le nombre moyen de violations de délai et la consommation énergétique de l'infrastructure (communications et calculs) obtenus par chaque méthode de placement pour différentes tailles du problème (en nombre d'applications) et un ensemble hétérogènes d'applications Mission critiques (MC), Temps Réel (RT), Streaming (ST) et Best Effort (BE).

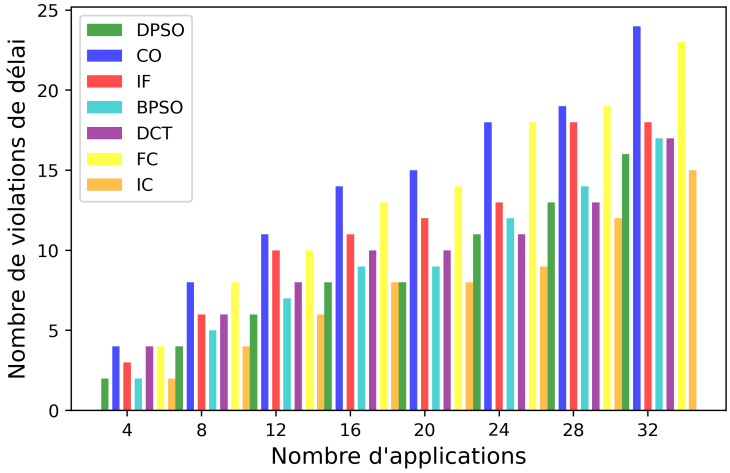
On remarque que l'approche DPSO offre le meilleur compromis entre la minimisation du nombre de violations de délai et de la consommation énergétique, suivie par BPSO qui est un peu moins efficace, ce qui est dû au phénomène de saturation de la fonction Sigmoid qui conduit le BPSO à avoir un comportement aléatoire. Le DPSO minimise en moyenne la consommation d'énergie et le nombre de violations de délai respectivement de 8% et de 15% comparé au BPSO.

La méthode DCT donne d'assez bons résultats étant donné que cette dernière considère tous les noeuds pour le placement comparé aux autres heuristiques. DCT minimise la consommation énergétique respectivement de 7% et 13% comparé au DPSO et au BPSO. Cependant, le nombre de violations de délai augmente de 16% et 9% comparé respectivement au DPSO et BPSO.

L'approche CloudOnly (CO) est la méthode qui consomme le plus d'énergie. L'utilisation du Cloud pour placer de petits services (en demande de calcul) et qui communiquent souvent avec les capteurs/actionneurs des noeuds IoT peut augmenter drastiquement la consommation énergétique et les temps de réponses des applications IoT. A contrario, l'utilisation des couches IoT et Fog sans le Cloud avec l'approches IoTFogOnly (IF) réduit l'utilisation des ressources réseau et la consommation énergétique mais l'exécution des services prend plus de temps et conduit à des dégradations de performances (violations de délai) comme on peut le voir avec la figure 3.12b. De plus, les couches Fog et IoT ne sont pas suffisantes pour héberger tous les services, considérant les capacités de calcul et de stockage limitées des noeuds Fog et IoT. Prendre



(a) Consommation énergétique des communications et du calcul



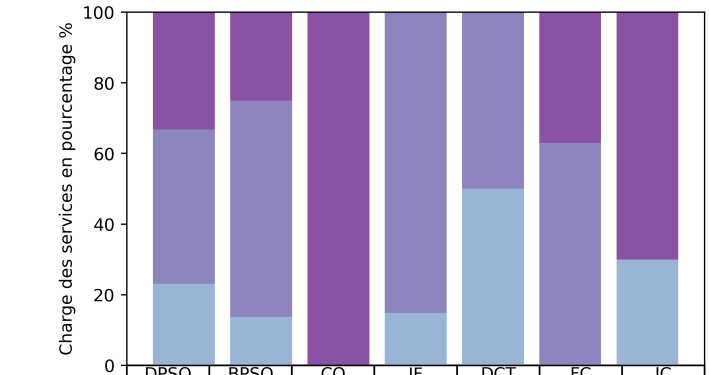
(b) Nombre moyen de violations de délai

FIGURE 3.12 – Consommation énergétique et nombre de violations de délai par nombre d'applications à placer et pour chaque méthode de placement

plus de temps pour exécuter les services des applications peut également induire à une consommation énergétique importante. Avec les approches IC et FC, qui utilisent respectivement les couches IoT-Cloud et Fog-Cloud, on peut observer que IC consomme moins d'énergie comparée à FC quand le nombre d'applications augmente. Ce qui se justifie par l'utilisation des noeuds IoT à faible consommation énergétique, lorsque le nombre d'applications augmente de plus en plus de services sont placés dans le Cloud ce qui réduit leur temps d'exécution et améliore le temps de réponse comme nous pouvons le voir avec la figure 3.12b.

L'approche IC est intéressante pour la réduction des délais en utilisant les noeuds IoT (proximité) et en exploitant la puissance de calcul du Cloud. L'approche FC consomme un peu plus d'énergie que l'approche IC mais ne réduit par pour autant le nombre de violations de délai, les temps de réponses augmentent à cause du temps de communication entre les utilisateurs et leurs services.

Nous pouvons conclure des analyses précédentes, que l'utilisation de seulement deux catégories de noeuds sur trois (peu importe lesquelles) n'est pas suffisant pour établir une stratégie efficace qui minimise efficacement la consommation énergétique et assure le respect des délais de réponses des applications IoT. L'approche DCT considère les 3 couches pour le placement ce qui donne de bons résultats pour l'énergie et le délai. Cependant, elle reste moins efficace que le DPSO et le BPSO.



	DPSO	BPSO	CO	IF	DCT	FC	IC
Cloud Layer	33.2	25.0	100.0	0.0	0.0	37.0	70.0
Fog Layer	43.7	61.3	0.0	85.2	50.0	63.0	0.0
IoT Layer	23.1	13.7	0.0	14.8	50.0	0.0	30.0

FIGURE 3.13 – Charge de services par couche et pour chaque méthode.

La figure 3.13 présente la charge de services par couche pour chaque méthode de placement. Ces derniers résultats confirment les observations précédentes et montrent que le DPSO utilise les 3 catégories de noeuds d'une manière plus au moins homogène pour pouvoir à la fois minimiser l'énergie consommée et réduire le nombre de violations de délai.

Pour le placement d'une application composée de 3 services et avec la moitié du temps dédié à la simulation des exécutions des services après leurs placements, la méthode DPSO prend 109s alors que les méthodes CloudOnly, IoTFogOnly, IC, FC and DCT prennent respectivement 84ms, 211ms, 114ms, 241ms et 167ms. En contrepartie, le DPSO minimise la consommation énergétique et le nombre de violations de délai 80% et 31% par rapport à CloudOnly, 38% et 31% par rapport à FogOnly, 61% et 31% par rapport FC, 38% et -15% (augmente) par rapport à IC, 9% et 7% par rapport au BPSO et -30% et 15% par rapport à DCT.

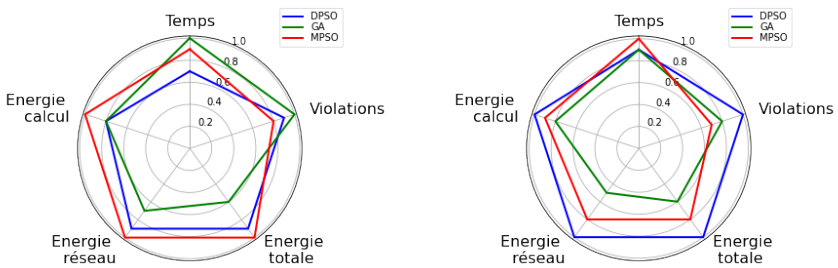
Même si les heuristiques sont plus rapides que l'approche DPSO, on peut voir que les gains apportés par la méthode en énergie et QoS sont plus intéressants.

Des analyses précédentes et en conclusion générale, nous pouvons dire que le DPSO exploite efficacement les ressources des couches Fog et Cloud, en réduisant le nombre de violations de délai tout en réduisant la consommation énergétique de l'infrastructure Fog mais le temps de résolution de cette méthode est non négligeable et peut compromettre son utilisation en temps réel.

3.6.2 Test du DPSO et des méta-heuristiques pour le placement statique

Dans cette partie, nous comparons les performances du DPSO à des stratégies plus complexes. L'objectif étant de voir le comportement des méthodes selon différentes variantes du problème. Nous voulons voir l'impact des topologies d'applications, des classes d'applications et de leur nombre, du nombre d'objets IoT et des technologies d'accès réseau sur les performances des méthodes.

3.6.2.1 Impact de la topologie de l'application



(a) SUD (b) MS

FIGURE 3.14 – Impact des topologies d'application SUD et MS sur les performances des méthodes DPSO, MDPSO et GA.

Nous reprenons les deux catégories de graphes d'applications décrites précédemment par les figures 3.10 et 3.9. Chaque topologie est instanciée selon une des classes d'application présentées par le tableau 3.1. On considère des ensembles homogènes de chaque topologie composés de 25% d'interactive Real-Time (RT) applications, 25% de Streaming applications (ST), 25% et Mission Critical applications (MC) et 25% de Best Effort applications (BE). Chaque application a une paire capteur/actionneur. Après le placement des services, on lance les simulations, ces dernières s'arrêtent dès que chaque capteur a envoyé 500 paquets de données.

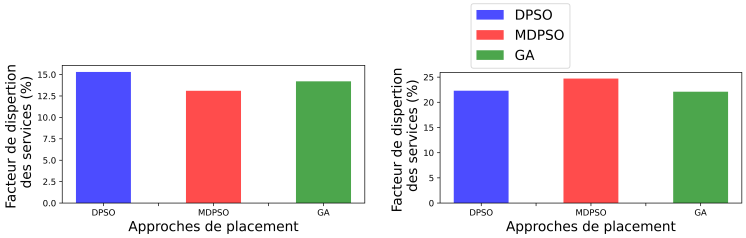
On considère une seul branche de l'infrastructure Fog présentée par la figure 3.11 : un noeud IoT connecté à une passerelle Fog. La passerelle peut être accessible par un lien filaire de 10 GB, un lien 4G LTE, un lien LoRa ou un lien Wi-Fi. La passerelle Fog est aussi connectée par un lien filaire à un autre noeud Fog qui fait office de proxy pour le noeud Cloud.

Pour chaque technologie d'accès réseau (filaire, 4G-LTE, Wi-Fi and LoRa), 50 exécutions ont été effectuées avec un ensemble d'applications de taille : 4, 12, 36 et pour chaque méta-heuristique. Les demandes des applications et les caractéristiques des noeuds de calculs sont prises uniformément dans les intervalles dont la valeur moyenne est précisée dans le tableau.

Les figures 3.14a et 3.14b présentent respectivement pour chaque topologie d'applications des radars graphes où les points des pentagones donnent respectivement les valeurs moyennes normalisées de la consommation énergétique du réseau et du calcul, du nombre de violations de délai et du temps d'exécution des méthodes pour toutes les expérimentations considérant tout type de réseau d'accès.

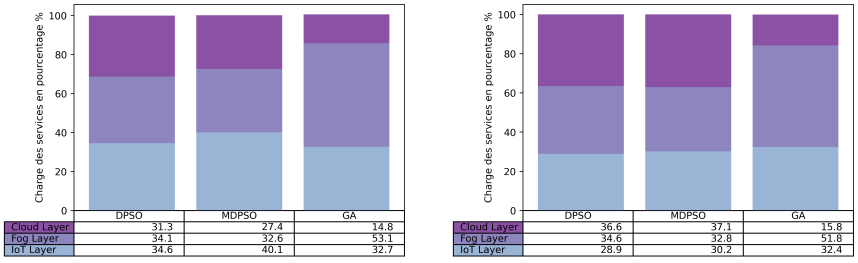
On peut voir que le GA est plus performant avec les topologies Master-Slave pour la minimisation du nombre de violations de délai. La structure des graphes MS comporte moins de dépendances de données entre les services que les graphes SUD. Les services esclaves peuvent s'exécuter en parallèle est ainsi réduire le temps d'exécution de l'application. Avec les topologies SUD, le dernier service doit attendre l'exécution de tous ses prédécesseurs pour envoyer l'instruction finale à l'actionneur. En comparant les méthodes entre elles, on voit que le MDPSO reste le plus performant pour la minimisation de violations de délai.

Le GA reste le plus performant pour la minimisation de l'énergie suivi du MDPSO et du DPSO. Cependant le DPSO a des temps de résolutions plus petits pour les deux topologies.



(a) SUD (b) MS

FIGURE 3.16 – Facteur de dispersion des services d'une même application pour les méthodes DPSO, MDPSO et GA



(a) SUD (b) MS

FIGURE 3.15 – Impact des topologies d'application SUD et MS sur l'utilisation de l'infrastructure par les méthodes DPSO, MDPSO et GA.

Des figures 3.15a et 3.15b qui présentent respectivement l'utilisation des couches Fog par les méthodes DPSO, MDPSO et GA pour chaque topologie d'application, on remarque que pour la topologie MS, le noeud Cloud est plus utilisé par toutes les méthodes comparée à la topologie SUD.

Si on considère les graphes SUD, on remarque que les méthodes DPSO et MDPSO placent leurs services plus intensivement dans les noeuds IoT alors que le GA les place dans le Fog.

On remarque que le MDPSO utilise plus les noeuds IoT pour les applications SUD que MS. Les services sont placés au plus proche des utilisateurs finaux IoT alors que le GA sollicite plus les noeuds Fog. Ce qui explique que le MDPSO soit plus performant avec les applications SUD. Le DPSO utilise plus intensivement les noeuds IoT et Cloud comparé au GA et moins que le MDPSO, ce qui explique les résultats moins bons pour la minimisation des deux objectifs.

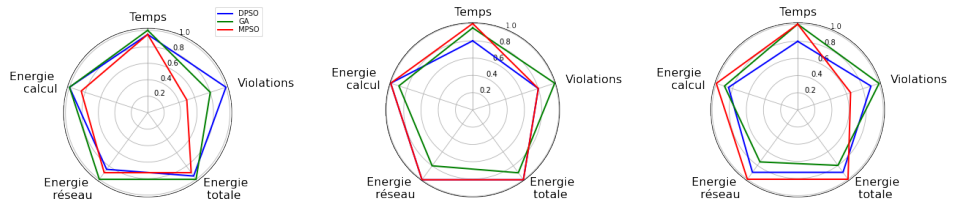
La figure 3.16 présente la moyenne des dispersions entre les services d'une même application en pourcentage, considérant différentes tailles de l'ensemble d'application pour chaque topologie de graphe. La dispersion entre les services s_j^i et $s_{j'}^i$, placés respectivement sur les machines m_k et $m_{k'}$, est le nombre de noeuds qui séparent les deux machines et cette valeur est égale à 0 si $m_k = m_{k'}$.

La dispersion de l'application est la moyenne des dispersions de chaque paire de services s'échangeant des données. Nous pouvons voir que pour chaque méthode le pourcentage de dispersion des services est assez faible (inférieur à 25%). Les résultats des méthodes sont assez proches pour les topologies MS et on observe une différence pour les topologies SUD où le facteur de dispersion du MDPSO est inférieur à ceux des méthodes GA et DPSO.

Des précédentes analyses, nous pouvons dire qu'il y a peu d'impact entre la dispersion des services et les performances des méthodes pour la minimisation du nombre de violations et de la consommation énergétique.

3.6.2.2 Impact du nombre d'applications IoT

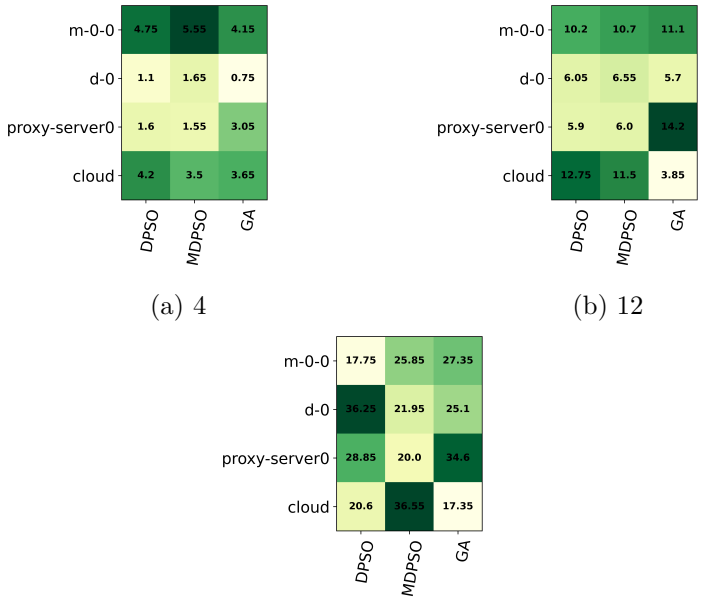
Dans cette partie, nous considérons un seul objet IoT et nous faisons varier le nombre d'applications demandées par cet objet.



(a) 4 (b) 12 (c) 36

FIGURE 3.17 – Les valeurs moyennes des métriques de la consommation énergétique, violations de délai et temps d'exécution de chaque méthode par nombre d'applications placées.

Les figures 3.17a, 3.17b et 3.17c présentent les valeurs moyennes des métriques de la consommation énergétique, violations de délai et temps d'exécution de chaque méthode par nombre d'applications placées. Nous pouvons observer que pour un petit nombre d'applications le MDPSO est plus performant pour les minimisations des deux objectifs comparé au GA et au DPSO. Lorsque le nombre d'applications à placer augmente (12, 36) le GA donne de meilleurs résultats pour la minimisation de l'énergie. Le DPSO est plus performant si le nombre d'applications augmente alors que le temps d'exécution du MDPSO est toujours plus important comparé à celui du DPSO et du GA. Ceci peut être justifié par le temps supplémentaire pris par le MDPSO pour estimer et sauvegarder les solutions du front de Pareto approximé qui augmente avec la taille du problème.



(a) 4

(b) 12

(c) 36

FIGURE 3.18 – Usage des noeuds de chaque couche pour le placement des applications par nombre d'applications placées

La figure 3.18 présente les valeurs moyennes de 120 exécutions donnant le pourcentage de la charge des noeuds en services pour chaque méthode de placement en augmentant le nombre d'applications placées.

Les expérimentations ont montré que contrairement à la topologie des applications, le nombre d'applications impacte très peu sur la façon dont les noeuds sont utilisés. On remarque que le GA utilise plus les noeuds de la couche Fog comparé aux méthodes DPSO et MDPSO, qui sollicitent plus intensément le Cloud et utilisent les deux autres types de noeuds d'une manière équitable. On remarque que le MDPSO et le DPSO ont plus ou moins la même distribution de services sur les différentes couches de noeuds (Cloud, Fog, IoT) quelque soit l'application.

On peut voir que le GA utilise plus intensivement les noeuds IoT et Fog comparé aux méthodes DPSO et MDPSO. Pour un petit nombre d'applications, le MDPSO utilise plus intensivement la couche IoT ce qui a tendance à réduire les violations de délai. Lorsque la taille du problème augmente (nombre d'application) la méthode sollicite plus intensément le noeud Cloud et très peu les noeuds Fog.

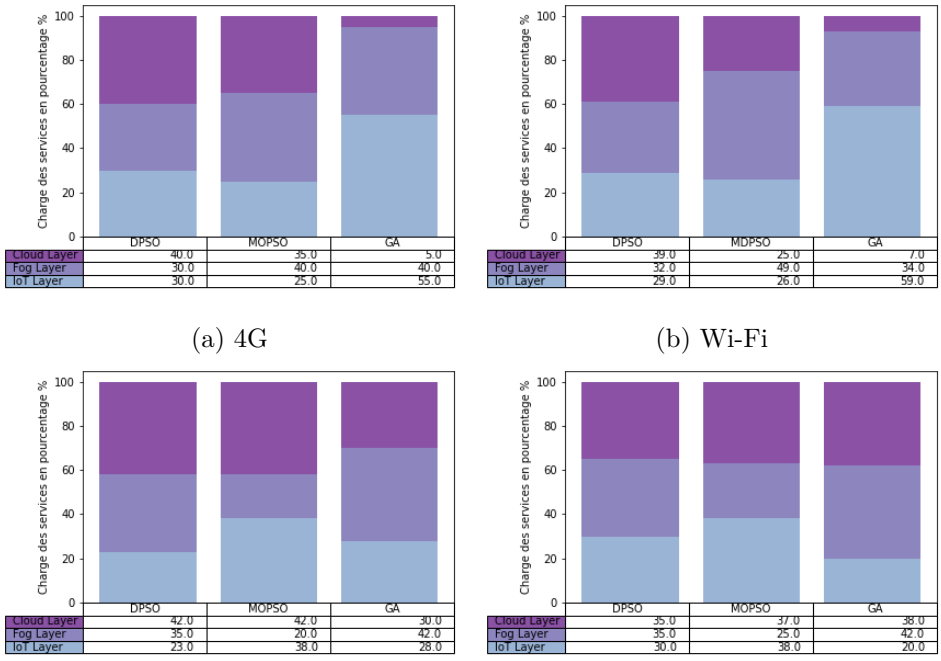
3.6.2.3 Impact des technologies d'accès

Nous considérons un ensemble de 4 applications de chaque classe et une topologie Fog avec une seule technologie d'accès. La figure 3.19 présente l'uti-

lisation des couches pour chaque méthode en fonction des technologies réseau de l'infrastructure Fog. Nous pouvons observer que pour un réseau d'accès avec des caractéristiques 4G et Wi-Fi le GA utilise plus les noeuds IoT et Fog ce qui lui permet de minimiser la consommation énergétique, les méthodes DPSO et MDPSO utilisent plus les noeuds Fog et Cloud.

Avec le réseau LoRa et les réseaux filaires, on observe que le GA, le DPSO et MDPSO utilisent plus le Cloud comparé aux noeuds IoT et Fog et comparé à l'utilisation dans les réseaux 4G (même si le taux est plus élevé pour le GA)

Le comportement des méthodes est assez cohérent si on considère que les réseaux 4G et Wi-Fi sont assez énergivorent comparés aux réseaux LoRa et aux réseaux filaires [IEA 2017].



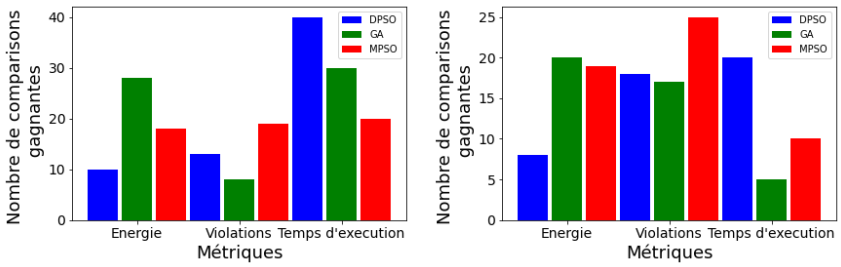
(a) 4G (b) Wi-Fi

(c) LoRa (d) Filaire

FIGURE 3.19 – Utilisation de couches Fog pour chaque méthode en fonction du type de réseau d'accès.

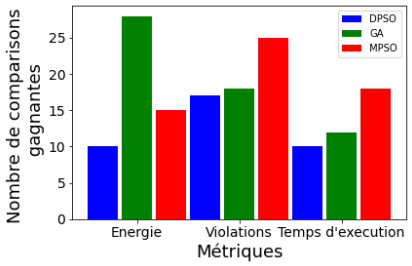
3.6.2.4 Impact des classes d'applications

Dans cette partie nous nous intéressons au comportement des méthodes pour le placement de services selon les classes de leurs applications selon leurs classes : Mission Critical (MC), Real-Time (RT), Streaming (ST) et Best Effort (BE).

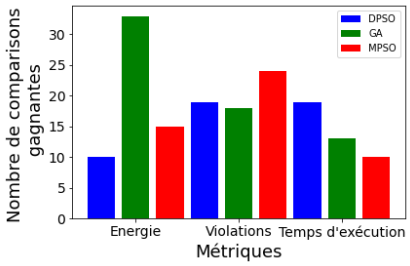


(a) MC

(b) BE



(c) ST



(d) RT

FIGURE 3.20 – Illustrations par histogrammes du nombre de comparaisons gagnantes pour chaque métrique entre les méthodes DPSO, GA et MDPSO, par classe d'applications : MC, RT, ST et BE.

La figure 3.20 présente le nombre de fois où une méthode est meilleure pour au moins un objectif par objectif et par classe d'application.

Nous pouvons voir que la tendance est la même pour toute les classes et que la GA est performant pour la minimisation de la consommation énergétique le DPSO et MDPSO sont plus performants pour la minimisation du nombre de violations de délai.

3.6.2.5 Comportement des méthodes en moyenne considérant les expérimentations dans leur globalité

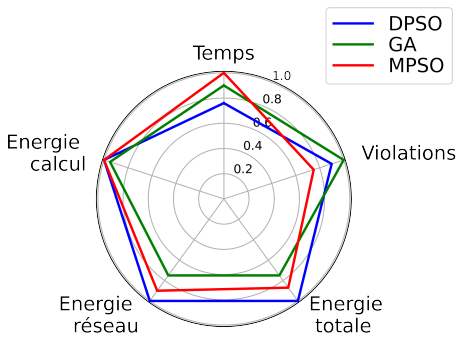


FIGURE 3.21 – Radar Graphe de la moyenne de toutes les expérimentations

Si nous analysons les résultats dans leur globalité avec la figure 3.21, qui présente les valeurs des différentes métriques considérant toute les tailles du problème (4-12-36) et topologies des graphes d'applications (SUD-MS) et un ensemble hétérogène de classes d'applications et de technologies d'accès, nous pouvons faire deux principales observations.

(1) L'approche MDPSO est plus performante que les approches DPSO et GA pour la réduction du nombre de violations de délai alors que le GA est plus performant comparé au DPSO et au MDPSO pour la réduction de la consommation énergétique.

En effet, l'approche MDPSO réduit le nombre de violations de délai de 37% comparée au GA et à 31% comparée au DPSO. Le GA réduit la consommation énergétique totale de 25% comparé au DPSO et de 28% comparé au MDPSO.

Ces résultats peuvent être justifiés par la nature multi-objectifs du MDPSO. Ce qui lui permet un meilleur exploration pour minimiser le nombre de violations de délai.

Le GA et le DPSO manipulent une somme pondérée des deux objectifs (énergie et délai) définie par l'équation Eq (3.16). Cette formule donne un avantage à la minimisation de l'énergie (les valeurs sont plus grandes que le nombre de violations de délai) et ce qui conduit les deux approches à trouver de meilleurs solutions pour la minimisation de l'énergie.

(2) DPSO est plus performant que le GA pour réduire le nombre de violation de délai de 15% et est plus performant que le MDPSO pour la réduction de la consommation d'énergie de 8%. Même si le DPSO utilise la même somme pondérée que le GA, il est capable de trouver de meilleures solutions pour réduire le nombre de violations de délai. Ceci est dû à la diversification de l'exploration grâce aux équations de mise à jour de la vélocité qui intègrent des paramètres stochastiques et qui s'avèrent plus efficaces que la mutation dans le GA. Les particules ont une certaine liberté d'exploration et indépendance alors que les chromosomes du GA résultent des individus des générations précédentes. Nous pouvons conclure des analyses précédentes que le MDPSO serait plus adapté pour les petits problèmes (figure 3.17a). Le GA devrait être utilisé quand la taille du problème augmente et lorsque la minimisation de l'énergie est plus importante que la QoS.

Si le paramètre QoS est plus important, alors le MDPSO est plus intéressant mais si le temps de résolution rentre également en compte alors le DPSO donne un bon compromis entre les temps de d'exécution et la minimisation du nombre de violations de délai.

Le tableau 3.6 résume les conclusions précédentes en spécifiant quelle méthode choisir selon la priorité de l'objectif. Les méthodes sont ordonnées de 1 (la meilleure approche) à 3 (l'approche la moins performante).

Méthode/Objectifs	Énergie	Violations	Temps	Énergie-Violation	Énergie-Temps	Violation-Temps
DPSO	3	2	1	3	2	1
GA	1	3	2	2	1	3
MDPSO	2	1	3	1	3	2

TABLE 3.6 – Classification des méthodes DPSO, GA et MDPSO selon l'objectif visé. De la méthode la plus recommandée (1) à la moins recommandée (3).

3.7 Bilan des expérimentations et amélioration des approches de placement

Nous avons vu à travers les différentes expérimentations que les technologies d'accès, les classes d'applications ainsi que la taille du problème avaient un impact sur les performances des méthodes et que le comportement de chaque approche était singulier face à ces facteurs. Bien qu'étant rapides, les approches heuristiques vues précédemment sont très peu performantes pour les objectifs définis et restent assez simples en ne considérant pas tous les aspects du problème. Quant aux approches méta-heuristiques utilisées, elles ont deux inconvénients majeurs :

— Un temps de calcul important (pouvant atteindre plusieurs minutes).
— Le fait qu'elles n'intègrent pas directement les notions de consommation de ressources par classes d'applications, des technologies d'accès et les informations des objets dans leur logique de placement.

Par rapport aux objectifs d'optimisation considérés et à leur modélisation, on remarque que le critère QoS défini est assez souple. Le placement autorise des solutions qui violent les délais de réponse quelle que soit l'application. Or, il est impératif que le placement de certaines applications respecte leur délai de réponse. En particulier pour les applications sensibles de classe Mission Critique (MC), touchant à la sécurité des individus et certaines applications Temps Réel (RT). Pour obtenir de meilleures solutions de placements, il est important de ne pas considérer seulement l'aspect priorité entre les applications mais également le profil de consommation des différentes ressources réseau et de calcul de ces dernières. Quant à l'énergie, l'utilisation d'une équation linéaire en fonction du taux d'utilisation CPU et des interfaces des noeuds est suffisante mais il est possible d'utiliser d'autres modèles avec les méthodes de placement utilisées.

Nous proposons dans ce contexte une approche de placement hybride qui combine à la fois les performances exploratoires des méta-heuristiques et qui réduit leur coût d'exécution avec l'utilisation des heuristiques. L'algorithme prend également en considération les facteurs vus précédemment pour le placement. Le diagramme des étapes de l'algorithme est présenté dans la figure 3.22.

Nous proposons un ensemble de règles d'associations présentées par le dendrogramme de la figure 3.23 et déduites des expérimentations précédentes. Ces

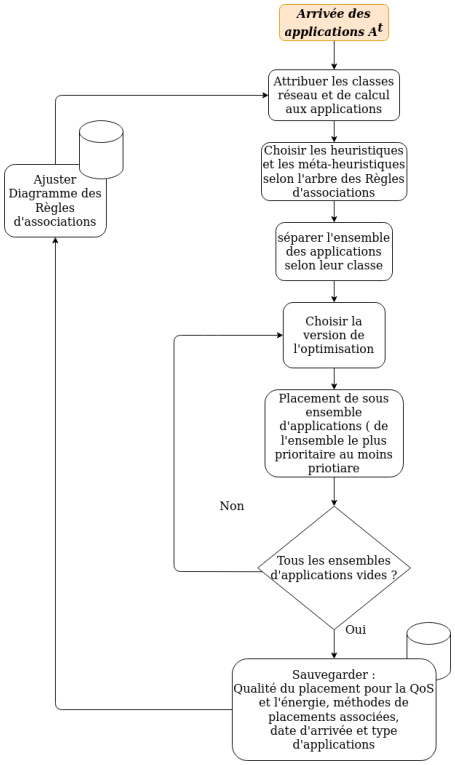


FIGURE 3.22 – Diagramme algorithme de placement.

règles aident à choisir les heuristiques d'initialisations et les méta-heuristiques pour le placement. On décompose les besoins en ressources des applications en deux catégories : besoin réseau (R) et besoin de calcul (C). Chaque catégorie a trois niveaux de demande : haut (H), moyen (M) et bas (L). Nous aurons donc pour les réseaux les niveaux (R-H), (R-M) et (R-L) et pour le calcul (C-H), (C-M), (C-L).

Les labels "haut", "moyen" et "bas" sont relatifs dans l'ensemble d'applications à placer et sont attribués comme suit.

— Pour le réseau, nous calculons la taille moyenne du paquet de données échangées allant du capteur vers l'actionneur de chaque application. En suite, les applications sont affectées selon la valeurs trouvé à une des classes : R-H, R-M ou R-L.

— Pour le calcul, nous calculons le nombre moyen d'instructions que doit traiter toute l'application pour un flux allant du capteur vers l'actionneur. Ensuite, les applications sont affectées selon la valeurs trouvé à une des classes : R-H, R-M ou R-L.

Si on a un ensemble homogène d'applications selon les priorités, nous ajoutons des contraintes sur la QoS selon leur classe.

Pour les classes Mission Critique (MC) la minimisation de F, définie par l'équation (3.16) se fait sous contrainte de QoS stricte. Pour avoir une solution

valide, aucune violation de délai n'est autorisée. Pour les classes temps réel (RT) et Streaming (ST) la minimisation de F se fait avec une tolérance sur la contrainte de QoS. C'est-à-dire qu'un certain taux de violations est autorisé. Pour les applications Best Effort (BE) la priorité est l'économie d'énergie donc nous optons pour la minimisation de l'énergie sans contrainte sur la QoS.

Comme cette amélioration propose de traiter des ensembles homogènes d'applications (par classe).

L'ordre des applications dans les chromosomes et les particules se fera comme suit :

— Si la demande réseau est plus importante que la demande de calcul, les applications sont classées par ordre décroissant de leur demande réseau et les services sont classés par ordre de précédence (du capteur vers l'actionneur).
— Si la demande en calcul est plus importante que la demande réseau, les applications sont classées par ordre décroissant de leur demande en calcul et les services sont classés par ordre de précédence (du capteur vers l'actionneur).
— Si la demande de consommation en calcul et en communication des applications est similaire i.e. RH-CH, RM-CM et RL-CL, les applications sont ordonnées selon la valeur de leur priorité, calculée par l'équation (3.1) (de la plus petite à la plus grande valeur).

Le tri des noeuds de calcul se fait comme suit :

— Si la demande réseau est plus importante que la demande de calcul, les noeuds sont classés selon leur proximité à l'objet IoT (en nombre de sauts).
— Si la demande en calcul est plus importante que la demande réseau, les noeuds sont classés par ordre décroissant des capacités de calcul.
— Si la demande de consommation en calcul et en communication des applications est similaire i.e. RH-CH, RM-CM et RL-CL, les noeuds sont classés par ordre décroissant des capacités CPU disponible pour les noeuds Fog, les noeuds IoT et ensuite le Cloud. Si les noeuds ont des capacités similaires alors ils ont classé selon leur proximité à l'objet IoT (en nombre de sauts).

Extension des stratégies avec d'autres règles et machine learning

Il serait possible d'affiner les détails de la consommation des ressources réseau en différenciant entre les applications qui sollicitent le réseau pour faire monter des informations (Up) ou pour récupérer des informations (Down) et inversement ou avec un taux de Up équivalent au taux de Down. Les applications pourraient également être catégorisées selon la fréquence de communication avec leurs objets IoT ou les fréquences de communications de leurs services critiques.

L'arbre que nous proposons est fait avec des règles fixes déduites des expérimentations précédentes. Il serait possible de garder un historique des placements, de récolter des métriques sur les résultats et changer les règles si nécessaire selon la qualité des placements antérieurs. On affecte des poids en fonction de la qualité des solutions précédentes (et qui change donc dynamiquement) pour choisir les méthodes lors du prochain placement.

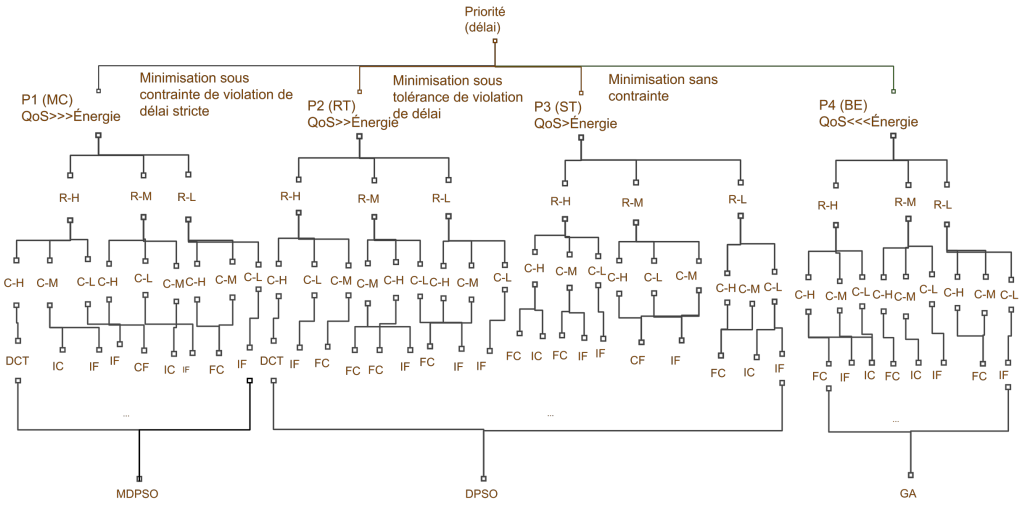


FIGURE 3.23 – Dendrogramme des choix des méthodes initiales et des méta-heuristiques

Nous pouvons également considérer d'autres métriques pour calculer la proximité entre l'objet IoT et un noeud Fog qui peut être soit la technologie de communication ou soit la bande passante disponible.

3.8 Synthèse

Dans ce chapitre nous avons présenté notre modélisation des infrastructures Fog Computing et des applications IoT. Nous avons ensuite posé le problème de placement de services dans le Fog sous forme de problème d'optimisation dont la résolution a pour objectif de minimiser la consommation énergétique du système et le nombre de violations de délai des applications. A cet effet, nous avons proposé des approches méta-heuristiques à essaim avec le Discrete Particle Swarm Optimization (DPSO), sa version multi-objectifs (MDPSO) et une approche évolutionnaire avec l'algorithme génétique (GA) ainsi que quelques heuristiques gloutonnes. A travers les tests et évaluations des méthodes, nous avons essayé d'identifier l'impact des différents facteurs variables de l'environnement sur l'utilisation de l'infrastructure par les méthodes et leurs performances pour les objectifs définis. Ainsi nous avons évalué

les méthodes selon :

— La topologie des applications IoT.
— Les technologies d'accès réseau.
— La scalabilité vue à travers l'augmentation du nombre d'applications demandées par le même objet et par l'augmentation du nombre d'objets IoT.

Nous avons également proposé une stratégie placement hybride afin de tirer avantage de méthodes étudiées dans ce chapitre.

Dans le prochain chapitre nous allons considérer un environnement Fog-IoT dynamique via la mobilité des objets IoT.

Chapitre 4

Placement de services en environnement dynamique

Sommaire

4.1 Introduction 122

4.2 Modèles de mobilité : Définition, Taxonomie et Analyse de l'existant 123

4.3 Extension du modèle de l'environnement IoT-Fog statique 128

4.3.1 Zone de couverture des noeuds Fog 129

4.3.2 Noeuds IoT mobiles 129

4.4 Modèle de mobilité History-Based Transition Matrix (HTM) 130

4.4.1 Principe et généralités du modèle 130

4.4.2 Méthodologie 130

4.4.3 Créneaux horaires et zones de mobilité 131

4.4.4 Création des matrices de transitions entre les zones 132

4.4.5 Réajustement ou réduction des matrices de transitions 133

4.4.6 Avantages et inconvénients de HTM 134

4.4.7 Améliorations du modèle HTM (Méta-modèle de mobilité et contrôle dynamique de l'environnement mobile) 134

4.5 Migration et redéploiement de services 136

4.5.1 Estimation du temps de migration d'un service 136

4.5.2 Estimation de la consommation énergétique de la migration d'un service 137

4.5.3 Fonction objectif 138

4.6 Approches de résolutions 141

4.6.1 Placement sans réévaluation 141

4.6.2 Placement avec réévaluation 143

4.7 Résultats 146

4.7.1 Tests de Validation du modèle de mobilité 146

4.7.2 Tests de Validation des équations de migration 149

4.7.3 Tests de Validation des méthodes de placements 151

4.8 Synthèse 162

4.1 Introduction

La mobilité des objets est une caractéristique importante des environnements IoT et Fog qui amènent notamment de la dynamicité. Cet aspect a été jusque là très peu considéré dans les travaux de placement de services dans ces environnements [Cziva 2018], [Bittencourt 2017].

Placer, déployer ou tout simplement gérer le cycle de vie des applications dans un environnement dynamique est un problème complexe généré par le très grand nombre de noeuds mobiles et l'impossibilité d'estimer avec exactitude et en des temps raisonnables leurs déplacements. Il est pourtant primordial d'exploiter les informations de mobilité des objets dans les stratégies de décision liées au placement d'applications et à l'allocation des ressources de calcul. Ces stratégies vont répondre aux besoins des applications et assurer une gestion efficace des infrastructures de calcul et de communication au cours du temps. De plus, le déploiement distribué des services d'une application sur un très grand nombre de noeuds de calcul peut induire une consommation énergétique non négligeable des infrastructures Fog. Ce phénomène peut être amplifié par les déplacements des objets au cours du temps.

Dans ce chapitre, nous allons aborder la problématique du placement d'applications dans un environnement à objets mobiles. Le but est de répondre aux besoins et objectifs d'optimisation définis dans la chapitre précédent : minimiser la consommation énergétique de l'infrastructure Fog et les violations du délai de réponse des applications.

Les contributions présentées dans ce chapitre peuvent être divisées en trois parties :

1. Proposition et évaluation d'algorithmes de placement de services sans réévaluation de la solution au cours du temps (sans migration).
   (a) Un algorithme génétique probabiliste, Mobility-aware Genetic Algorithm (MGA) et une heuristique gloutonne, Mobility Greedy Heuristic (MGH) pour le placement de services en considérant l'information de mobilité de noeuds.
   (b) Comparaison des performances des approches de placement avec des stratégies de migration de la littérature, Shortest Access Point (SAP) et K-Shortest Access Point (KSAP), basées sur le principe du "Follow me services" .
2. Établissement d'un modèle de mobilité basé sur l'historique de positionnement des noeuds et intégration de la migration au MGA.
   (a) Proposition de trois heuristiques de placement rapides prenant en compte un seul aspect (dimension) du problème à la fois : état de l'infrastructure, informations des objet IoT et caractéristiques des applications.

   (b) Utilisation de l'algorithme génétique pour réévaluer le placement précédent.

   (c) Comparaison avec une méta-heuristique de la littérature adaptée à des problèmes dynamiques, Penguins Search Optimization Algorithm (PeSOA) et les stratégies de migrations SAP et KSAP.

3. Proposition d'une architecture et d'un simulateur pour un système adaptatif de contrôle de la mobilité et de l'infrastructure ICT afin de minimiser les coûts de contrôle et de gestion de l'infrastructure.

   (a) Extension du simulateur MyiFogSim pour intégrer des modèles de mobilité et de migration à la demande.

   (b) Extension de MyiFogSim avec le simulateur de mobilité SUMO.

   (c) Description du Framework de gestion de la mobilité et de l'infrastructure ICT.

Ce chapitre est structuré comme suit : dans la section 4.2, nous présentons un état de l'art, un résumé et une classification des catégories de modèles de mobilité de la littérature. La section 4.3 présente la suite de la modélisation du système et du problème de placement présentés dans le chapitre 3 avec les aspects de mobilité. La section 4.4 présente le principe du modèle de mobilité généré à partir des traces de localisation. Dans la section 4.5, nous présentons les équations d'estimation du temps et de l'énergie consommée pour la migration de services. La section 4.6 expose les approches de résolution du problème de placement de services IoT dans le Fog avec mobilité ainsi qu'une architecture d'amélioration du système de placement proposé. Enfin, la section 4.7 présente les résultats de validation du modèle de mobilité History-based Transition Matrix (HTM), des équations de migration et des approches de placements proposées. Enfin, nous terminons ce chapitre par une synthèse des travaux proposés et quelques perspectives.

4.2 Modèles de mobilité : Définition, Taxonomie et Analyse de l'existant

Un modèle de mobilité permet de décrire les caractéristiques du mouvement d'un objet ou d'un groupe d'objets mobiles. Généralement, le modèle propose des formules descriptives fonction de paramètres comme les positions antérieures de l'objet, les instants de ses positions, la durée qu'il passe à chaque endroit (temps de pause), sa vitesse, sa direction ainsi que l'impact d'autres objets sur son mouvement. Les modèles de mobilité peuvent également être construits à partir d'inférences statistiques et utilisent des lois de probabilité pour décrire et estimer le mouvement des objets. Un modèle de mobilité permet d'anticiper dans certains cas les chemins que va emprunter l'objet et joue

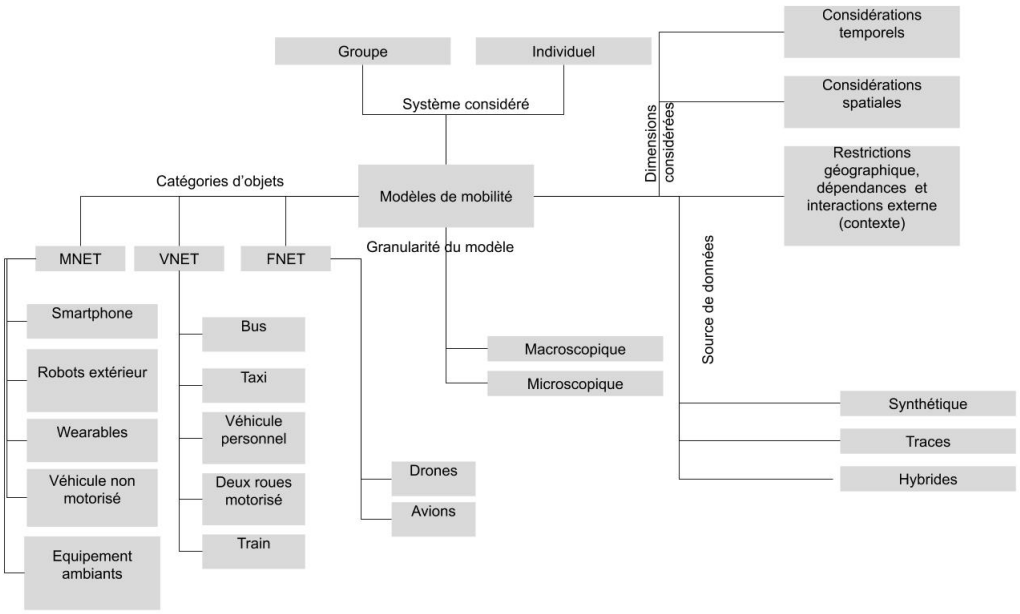


FIGURE 4.1 – Taxonomie modèles de mobilité

un rôle déterminant pour l'élaboration et l'évaluation de modèles de décisions dans des environnements dynamiques tel que l'IoT et le Fog.

La recherche sur les modèles de mobilité a connu un fort engouement avec l'avènement des réseaux Ad-hoc. Cependant, depuis quelques années on constate très peu de nouveaux modèles ou d'amélioration proposées pour les modèles existants. Les principaux nouveaux modèles sont proposés exclusivement pour les systèmes de véhicules connectés. Les travaux de [Baumann 2007] ont montré l'impact que peut avoir le modèle de mobilité sur l'évaluation des performances réseau et la gestion des ressources et en particulier, pour l'acheminement des données. Un modèle de mobilité peu réaliste peut surévaluer les performances des stratégies proposées.

Comme nous pouvons le voir avec le schéma de la figure 4.1, un modèle de mobilité peut être catégorisé selon plusieurs aspects :

1. *La nature des données du mouvement :* On trouve dans la littérature un certain nombre de modèles qu'on peut décomposer selon la source de données du mouvement en modèle synthétique, modèle de trace ou modèle hybride.

   Les travaux [Baumann 2007], [Grzybek 2012] et [Silva 2015] ont montré que les modèles à base de traces sont plus réalistes, pertinents et moins coûteux que les modèles analytiques purement synthétiques. [Musolesi 2009] a proposé un modèle basé sur le comportement social des noeuds mobiles, un aspect particulièrement important des environnements IoT et en particulier dans des environnements de villes intel-

ligentes où les services sont principalement conçus pour répondre aux besoins des habitants.

Les modèles à base de traces ont pour inconvénient d'être appliqués et étudiés dans des scénarios limités et des environnements bien définis. Ce qui peut facilement les rendre inutilisables dans d'autres scénarios ou d'autres zones étudiées. De plus, les études statistiques sur les traces effectuées à un moment donné peuvent devenir obsolètes au cours du temps.

Pour pallier aux inconvénients cités précédemment, les travaux de la littérature essayent d'identifier des comportements périodiques qu'on peut généraliser à partir des traces en faisant varier les environnements et en augmentant la durée de collecte des données.

Le projet CRAWDAD [Kotz 2009], principalement destiné à la communauté scientifique, propose une large variété de modèles de mobilité hybrides : des modèles synthétiques améliorés par des données collectées en environnements réels.

Le modèle de mobilité que nous proposons dans ce chapitre se base sur la caractérisation de traces pour extraire des informations génériques. D'autres travaux ont également effectué la même démarche en utilisant des traces. Il existe plusieurs approches pour caractériser les traces de mobilité. [Xia 2012], [Ahmed 2010] se sont intéressés respectivement à la mobilité des taxis de la ville Beijing et des bus de la ville de Seattle. Ils ont collecté des informations sur la durée d'inter-contact (ICT), qui représente l'intervalle de temps s'écoulant entre un premier contact entre une paire de véhicule et leur prochain contact.

La majorité des travaux à base de traces se font sur des données de véhicules comme les taxis ou les bus. Il est plus complexe d'identifier des motifs dans les mouvements des véhicules de particuliers ainsi que ceux de leurs objets IoT (pour les piétons). Au lieu d'essayer de dégager des modèles pour décrire le mouvement de chaque véhicule, les auteurs de [Füßler 2006] s'intéressent au comportement de mobilité sur les autoroutes d'une manière générique, en récoltant des informations comme les flux d'arrivées et de départs et la densité en véhicules. Cette étude ne peut malheureusement pas être appliquée à la mobilité en milieu urbain. [Uppoor 2012], [Uppoor 2014], [Naboulsi 2013] ont travaillé sur des traces en zone urbaine de la ville de Cologne en Allemagne. Les deux premiers travaux se sont intéressés aux flux de véhicules ainsi qu'à la densité des rues de Cologne, le dernier travail a étudié les traces à partir d'une perspective réseau. [Silva 2015] propose un modèle véhiculaire dans un environnement urbain en considérant une approche macroscopique origine-destination. Cette méthode construit le modèle uniquement avec les points de départs et d'arrivées des noeuds mobiles.

[Law 2000] propose une méthodologie pour pouvoir inférer des caractéristiques à partir d'une étude statistique sur un échantillon de données.

2. *La granularité de ses paramètres ou les métriques prises en compte :* Généralement, on compte deux niveaux de granularité pour les métriques de mobilité : (1) microscopique et (2) macroscopique. Les modèles microscopiques donnent une vue détaillée du mouvement d'un objet à chaque instant. Ces modèles, à fine granularité, prennent en considération la vitesse, la direction de chaque objet. Ils permettent, entre autre, de représenter des systèmes mobiles réalistes et détaillés. Ils prennent en compte les variations de vitesse de chaque objet et des scénarios de dépassement entre les véhicules. Ce type de modèles est très présent dans les réseaux véhiculaires Ad-hoc (VANET). Les modèles macroscopiques s'intéressent à des données plus génériques et non pas aux caractéristiques détaillées de chaque noeud, par exemple les taux de départs et d'arrivées d'objets dans un intervalle de temps donné, la densité en noeuds d'une zone, la distribution de probabilité des flux allant d'un point A à un point B etc. L'approche macroscopique s'intéresse à identifier des points d'intérêts qui influencent le comportement de mobilité des noeuds. Les aspects microscopiques et macroscopiques sont tous deux nécessaires pour établir un modèle de mobilité réaliste et complet. On trouve dans la littérature un très grand nombre de modèles microscopiques contrairement aux modèles macroscopiques qui ont été très peu étudiés [Silva 2015].

   [Silva 2015] propose un modèle macroscopique se focalisant sur l'origine et la destination des véhicules. Le modèle est générique et est déduit par inférences statistiques à partir de traces de mobilité existantes. Les auteurs insistent sur le fait que les modèles macroscopiques sont très peu étudiés. Le modèle macroscopique origine-destination s'intéresse à l'origine du départ et au lieu de la destination, aux moments des départs et des arrivées, aux temps des trajets ainsi qu'aux distances parcourues par les objets.

3. *Les dimensions espace et temps prises en compte :* On peut classifier le modèle de mobilité selon les dimensions de la mobilité considérées tels que la dépendance temporelle, la dépendance spatiale et les restrictions géographiques. Le travail de [Bai] propose une classification des modèles de mobilité individuels et de groupes pour les réseaux Ad-hoc. Ce dernier les regroupe en modèles aléatoires, modèles à corrélations temporels, modèles à corrélation spatial et modèles à restrictions géographiques. Cependant, aujourd'hui et particulièrement dans les environnements de l'Internet des Objet, plusieurs modèles regroupent toutes les caractéristiques précédentes. Dans la mobilité des noeuds IoT, il est important de considérer à la fois les dépendances spatiales, tem-

porelles et les restrictions géographiques ainsi que certaines spécificités concernant le type de l'objet. De plus, si cet objet est transporté par un piéton, le comportement social doit également compter. Les auteurs de [Sofia 2013] proposent une taxonomie des modèles de mobilité créés ou réutilisés pour les réseaux sans fils.

— *Modèles à dépendance temporelle* - Les modèles à dépendance temporelles s'intéressent souvent à la vitesse des noeuds et sont souvent de type microscopique, la vitesse d'un noeud à l'instant t dépend des vitesses précédentes et si l'on considère qu'elle dépend uniquement de la vitesse à $t-1$ alors c'est un modèle Markovien.

— *Modèles à dépendance spatiale* - Les Modèles à dépendance spatiale, considèrent que la position actuelle d'un noeud et sa direction impacte sa vélocité ainsi que sa prochaine destination.

— *Modèles à restrictions géographiques* - Ces modèles considèrent que les noeuds ne peuvent pas se déplacer librement dans tout l'espace et imposent des restrictions de déplacement qui peuvent dépendre des autres noeuds, des bâtiments, des piétons etc.

4. *Les classes d'objets mobiles :* Pour l'instant on distingue trois grandes familles : les réseaux véhiculaires (VANET) les réseaux ou les systèmes d'objets IoT volants ou Flying Ad-hoc Netwtork (FANET) et les réseaux mobiles Ad-hoc (MANET). On peut également regrouper dans chaque famille les objets selon les caractéristiques qui nous intéressent. [Kumari 2015] a proposé une sélection de modèles de mobilité existants pour représenter au mieux les systèmes d'objets volants dans un réseau. Il est nécessaire d'exploiter les spécificités des objets IoT et des modèles existants afin d'établir des modèles plus précis mais qui peuvent être génériques, adaptatifs au cours du temps et en fonction des catégories d'objets étudiés.
5. *Le système en mouvement considéré :* Un modèle de mobilité peut décrire un mouvement individuel ou un mouvement de groupe.

Nous donnons quelques exemples de modèles individuels et de groupe les plus utilisés dans la littérature selon la nature des données représentant le mouvement.

Modèles de mobilité individuels

— Modèle synthétique : marches aléatoire (modèle Brownian) [Doyle 1984], marche aléatoire probabiliste, Random Waypoint [Broch 2001], Gauss-Markov [Liang 2003].

— Modèles à base de traces : modèle Manhattan [Bai 2003], modèle simple Human Mobility [Munjal 2011].

— Modèles hybrides : modèle Weighted Waypoint [Hsu 2005], modèle City Section [Hossain 2009].

Modèles de mobilité de groupes Les modèles de groupes sont des modèles pertinents pour les environnements IoT. Souvent, les objets IoT interagissent entre eux pour proposer un ou plusieurs services. On peut citer comme exemples les systèmes de surveillance collaboratifs regroupant les caméras de drones, des caméras au sol fixes et les caméras de smartphones et le Crowdsourcing à partir de smartphone.

— Modèles synthétiques : modèle Reference Point Group Mobility [Hong 1999], modèle Column Mobility [Alam 2014], modèle Nomadic Mobility [Bujari 2017].

— Modèles hybrides : modèle Community based [Musolesi 2006], modèle unified relationship matrix [Gunasekaran 2009], modèle home cell community [Vastardis 2012].

Les modèles de mobilité proposés dans la littérature sont principalement utilisés à des fins de validations d'environnement de simulation. Dans notre travail, nous exploitons ces modèles pour établir des stratégies de placement dans des environnements dynamiques.

4.3 Extension du modèle de l'environnement IoT-Fog statique

Le problème considéré ici est le déploiement des services qui composent les applications IoT tout en prenant en compte la mobilité des objets IoT présents dans une zone donnée. Le but étant de pouvoir minimiser la consommation énergétique de l'infrastructure et de réduire le nombre de violations de délai des applications.

Pour des raisons de simplification de l'étude, on émet les hypothèses suivantes sur notre système :

1. On considère un système divisé en T pas de temps $\mathbb{T} = \{t_0, t_1, .., t_{T-1}\}$. La durée d'un pas de temps est définie comme la moitié du temps pris par le noeud le plus rapide du système pour parcourir la plus courte distance entre deux points d'intérêt.
2. On suppose que le mouvement d'un noeud est négligeable pendant un pas de temps i.e que le noeud peut changer de position uniquement au début ou à la fin d'un pas de temps.
3. On suppose que la durée d'un service est supérieure à la durée de la fenêtre de temps de l'étude.
4. On suppose que si deux objets IoT m_u et $m_{u'}$ demandent une application a_i, chaque objet aura ses propres instances des services de a_i.
5. Suivant les spécifications établies pour les réseaux de 5ème génération, un utilisateur sera toujours couvert par le réseau et aura donc toujours

accès à une station de base. A partir de là, on suppose que chaque objet IoT m_u est connecté en étant attaché à une seule station de base à la fois et pour chaque pas de temps $t \in \mathbb{T}$.

6. On suppose que les communications entre les objets IoT et leurs services sont uniformément distribuées sur la fenêtre de de l'étude $W_{\mathbb{T}}$ qui est divisée en T pas de temps.

En plus du modèle et des variables définies dans le chapitre 3 de l'environnement Fog statique, la modélisation est étendue avec les aspects et notions suivantes.

4.3.1 Zone de couverture des noeuds Fog

Chaque noeud m_k a une position géographique à chaque pas de temps t, donnée par les coordonnées (x_k^t, y_k^t) et une zone de couverture de rayon r_k^t. Le rayon de la zone de couverture des noeuds autre que les noeuds Fog est égale à 0.

Nous tenons à souligner le fait que les connexions réseau dans le Fog sont souvent non filaires et par conséquent le rayon de couverture d'un noeud Fog peut varier au cours du temps à cause de perturbations extérieures qui peuvent par exemple être liées aux conditions climatiques.

Les noeuds Fog regroupent deux types d'équipements : le point d'accès (AP) qui donne l'accès réseau aux objets IoT, et l'équipement serveur, dit *cloudlet* (SC), pour héberger les services.

4.3.2 Noeuds IoT mobiles

On rappelle que $\mathbb{U} \in \mathbb{M}$ est le sous-ensemble de noeuds IoT de l'infrastructure Fog. Les noeuds IoT se déplacent selon un modèle de mobilité ϑ_u (e.g. le modèle Manhattan [Bai 2003]) et demandent un ensemble d'applications $\mathbb{A}_u^t$ à l'instant $t \in \mathbb{T}$.

On considère P_{uk}^t la probabilité pour qu'un objet IoT m_u soit dans la zone de couverture d'un noeud Fog m_k à l'instant t.

Le calcul de cette probabilité varie selon le modèle de mobilité de l'objet ϑ_u. Nous donnons dans ce qui suit un exemple où l'objet a pour modèle de mobilité le Weighted Waypoint (WWP) [Hsu 2005].

On considère une zone géographique en 2 dimensions. Nous avons selon le modèle WWP un ensemble de lieux dits "waypoints" représentant des points d'intérêts pour les noeuds mobiles i.e la mobilité des noeuds est uniquement considérée entre ces lieux définis au préalable.

Nous avons les noeuds Fog positionnés aléatoirement sur la même zone géographique observée.

On calcule la distance euclidienne entre les différents waypoints et les points d'accès (noeuds Fog) de l'infrastructure de calcul. Sous l'hypothèse

qu'un waypoint peut être attaché à seulement un seul point d'accès Fog à la fois, on associe donc chaque waypoint à son point d'accès le plus proche.

P_{ur}^{t} est la probabilité que l'objet m_u soit au waypoint r à l'instant t et est donc égale à P_{uk} précédemment définie.

$$P_{uk}^{t} = P_{ur}^{t} \tag{4.1}$$

Cette approche est applicable sur les modèles de mobilité macroscopiques, qui considèrent la mobilité d'une zone, d'un lieu ou d'un chemin donné.

Pour les modèles microscopiques qui considèrent les vitesses et les directions de chaque noeud, il faudra calculer la distance de la prochaine position du noeud prédite par son modèle de mobilité par rapport aux points d'accès Fog.

4.4 Modèle de mobilité History-Based Transition Matrix (HTM)

Dans cette partie, nous proposons un modèle de mobilité pouvant s'appliquer à tout type d'objets mobiles géolocalisables sur une surface 2D. Ce modèle utilise les données de positions des objets. Il est idéal dans les zones urbaines et les environnements de villes intelligentes et est adapté pour un réajustement au cours du temps.

4.4.1 Principe et généralités du modèle

Nous proposons un modèle de mobilité macroscopique. En partant du principe que les environnements dynamiques qui ont pour moteur l'humain (comme les "smart city") sont moins sujets à l'aléatoire et ont des comportements de mobilité récurrents. En considérant les puissances de connectivité et de couvertures assurées par les réseaux tel que la 4G-LTE et la 5G, il est majoritairement superflu de considérer des modèles de mobilité à granularité fine qui donnent des positions géographiques précises à chaque instant pour pouvoir fournir les services demandés par les objets mobiles.

Le modèle proposé détermine les déplacements de classes d'objets IoT en fonction des créneaux horaires et de leur déplacement entre des régions bien définies dans une zone de mobilité observable.

4.4.2 Méthodologie

On suppose qu'on a une zone observable de superficie connue. On considère le plus petit carré qui peut contenir toute la zone. Les principales étapes du modèle se résument comme suit :

1. Définir les intervalles de temps à l'échelle humaine d'une journée découpés en plusieurs tranches horaires.
2. Subdiviser la zone en plusieurs petites zones selon les spécificités du réseau de connexion. Il faut que chaque zone soit couverte par au moins un point d'accès.
3. Créer les matrices de transition pour chaque objet m_u mobile et pour chaque intervalle I_x de chaque jour observable d_x du jeu de données. Un élément pz_{ij} de la matrice de transition $MZ_{d_x}^{uI_x}$ donne donc la probabilité qu'un objet se déplace vers une zone z_j en sachant qu'il se trouve dans la zone z_i.
4. Récupérer les informations macroscopiques des zones : densité moyenne par classe d'objets et flux d'objets.
5. Calculer et mettre à jour les informations des objets et leur profil utilisateur : la granularité et la mise à jour des modèles de mobilité des objets se fait en fonction de leur profil utilisateur. c'est à dire le type d'applications qu'ils demandent et la fréquence des demandes par intervalle de temps au cours de la journée ainsi que le type ou la classe des objets IoT.
6. Détecter les objets qui ont le même profil utilisateur et/ou le même comportement de mobilité au cours d'une journée pour réduire le nombre de matrices à utiliser pour la prédiction.

4.4.3 Créneaux horaires et zones de mobilité

On considère une fenêtre de temps de 24 heures qu'on va subdiviser selon une journée type d'activités en 4 créneaux horaires où généralement le comportement de mobilité des individus change très peu dans chaque créneau. $I_1 = [6am, 12pm], I_2 =]12pm, 6pm], I_3 =]6pm, 12am], I_4 =]12am, 6am]$. On considère que les moments où il y a le plus d'activité et de mobilité se font pendant les 3 premières heures de chaque intervalle et les comportements sont généralement relativement différents d'un intervalle à un autre.

Considérant l'environnement d'une ville intelligente, le dernier intervalle I_4 va généralement présenter des heures creuses (sans demandes d'applications) qui pourront être exploitées pour mettre à jour les matrices et établir un nouveau découpage de zones si nécessaire.

On considère les classes d'objets IoT mobiles suivantes : les objets pour piéton (M1) (smartphone, wearbales), les véhicules (M2) et le reste des objets (M3).

On considère à l'instant t_0 le plus petit carré recouvrant la zone géographique étudiée z_0 .

A l'état initial, la zone de mobilité considérée z_0 est subdivisée en plus petites zones $z_0 = \{z_{01}, .., z_{0n(z_0)}\}$. Le nombre de zones initialement créées

$n(z_0)$ est calculé en fonction de la plus petite portée radio (le plus petit rayon de couverture) $r^0_{min} = min_{k\in\mathbb{M}}\{r^0_k\}$ des noeuds Fog présents dans z_0 et de la superficie de la zone z_0. Le calcul du nombre de zones $n(z_0)$ est présenté par l'équation Eq 4.2.

$$n(z_0) = \frac{sup(z_0)}{(r^0_{min})^2} \tag{4.2}$$

4.4.3.1 Caractéristiques des zones de mobilité

Pour chaque zone et pas de temps t on peut avoir les informations suivantes (collectées au cours du temps) :

- La vitesse moyenne par classe d'objet du précédent pas de temps $t-1$.
- Les noeuds Fog positionnés dans la zone.
- Le flux d'arrivée et de départ par classe d'objets du précédent pas de temps $t-1$.
- Le profil utilisateur dominant du précédent pas de temps $t-1$.
- La fréquence de demande d'applications par tranche horaire et par classe du précédent pas de temps $t-1$.

4.4.4 Création des matrices de transitions entre les zones

Au lieu d'avoir un système central qui reçoit l'historique des mouvements des noeuds et les informations sur les zones pour calculer les matrices de transition de chaque objet, on propose que les informations de la topologie des zones soient connues par chaque zone et peut se faire via des méthodes de découverte de topologies de la littérature comme [Cisco 1994], [IEEE 2002].

A la suite de cela, chaque contrôleur de zone peut communiquer les informations sur la topologie aux noeuds mobiles pour que ces derniers puissent calculer leur matrice de transition indépendemment.

Pareillement à l'infrastructure de calcul, on considère un système de contrôle hiérarchique. Si le noeud mobile n'a pas la capacité de calcul suffisante, son historique de positions est sauvegardé dans sa passerelle Fog la plus proche et sa matrice de transition est calculée au niveau cette dernière. Si la passerelle n'a pas la capacité de calcul suffisante le processus remonte à un noeud de niveau supérieur (dans notre cas le Cloud).

On suppose que le contrôleur est choisi parmi les noeuds Fog selon sa puissance de calcul ou selon sa popularité (le plus grand nombre d'utilisateurs connectés au cours de la journée).

On rappelle que la consistance, le séquencement des messages et la synchronisation dans les systèmes distribués ne fait pas partie de notre travail et on se place donc dans un cas idéal où les informations arrivent aux bons moments.

En partant du principe que le comportement des noeuds mobiles est relativement différent entre les intervalles, il faut avoir au moins une matrice de transition par intervalle de temps.

Pour créer les matrices, l'historique des positions est parcouru. Ensuite, considérant chaque zone comme origine, on recense le nombre de fois où le noeud était dans une zone z_{0q} et est passé à une zone z_{0p} ainsi que le créneau horaire correspondant. Ensuite, on fait la moyenne des valeurs et on déduit les probabilités de transition. La figure 4.2 illustre les matrices d'un objet IoT m_u pour une journée d_x.

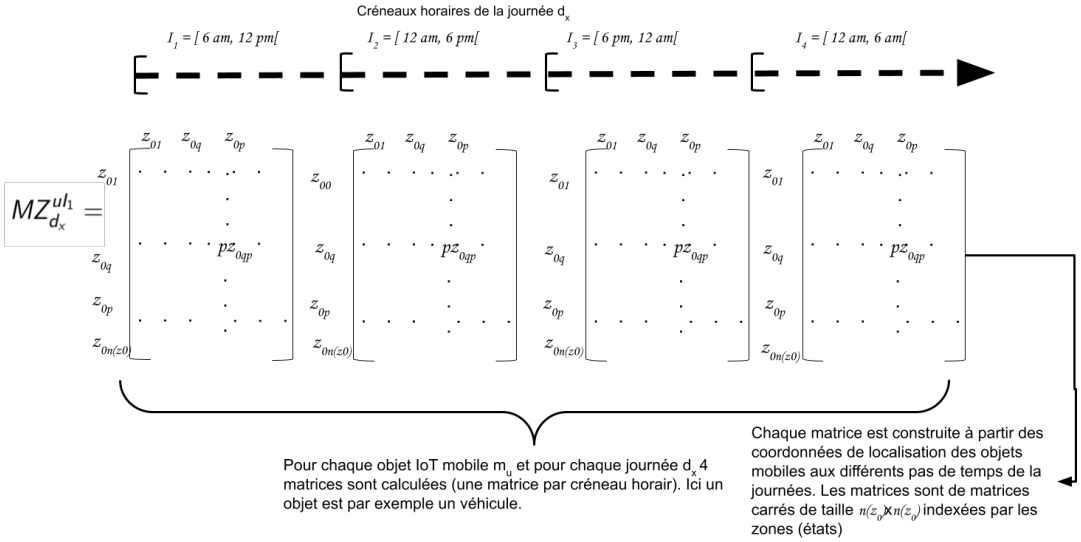


FIGURE 4.2 – Schématisation des matrices de transitions pour un objet

4.4.5 Réajustement ou réduction des matrices de transitions

Partant du principe que l'environnement est dynamique et large échelle, pour pouvoir utiliser cette méthode dans un système réel en des temps raisonnables, il est important de réduire le nombre d'états (nombre de zones) des matrices de transition et/ou le nombre de matrices de transitions, au détriment de la qualité de l'information et de la précision concernant le déplacement. Cette perte peut ne pas être importante considérant qu'avec les réseaux 5G l'utilisateur aura toujours accès à ses services et les applications demandées ne sont pas critiques et n'ont pas de contraintes de QoS importantes (classe Best Effort(BE)).

Nous proposons deux approches pour réduire le nombre de matrices.

— (1) Réduction par classe d'objets (M1, M2, M3) :
La réduction du nombre de matrice se fait en effectuant une moyenne sur les matrices par classe d'objets.

— (2) Réduction par profils utilisateurs (C1, C2, C3, C4) :
Cette approche permet de dire s'il est important de perdre en précision pour les informations de mobilité en fonction de la catégorie d'application utilisée par les objets mobiles. Si l'application est critique (MC), les matrices ne sont pas réduites (fusionnées). Si ce sont des applications Best Effort, l'information de mobilité est non pertinente alors le nombre de matrices est réduit. Cette méthode est intéressante dans le cas où les applications utilisées dépendent du chemin que va emprunter le noeud mobile.

4.4.6 Avantages et inconvénients de HTM

Le modèle de mobilité proposé présente un certains nombre d'avantages que nous précisons ici :

— Le modèle est simple d'utilisation et applicable à n'importe quel noeud mobile géolocalisable.
— Le modèle peut être facilement amélioré ou personnalisé en lui intégrant d'autres politiques de réduction de matrices.
— Si on a des modèles donnés bien précis, on pourra toujours générer des valeurs à partir de ce modèle et repasser à cette approche (avec perte d'information bien sur).
— Comme cette approche utilise la notion de zones, elle pourra être directement utilisée par les méthodes de placement de services que nous présentons dans la section 4.6.

Le modèle de mobilité HTM a également certaines limites et inconvénients :

— La méthode propose uniquement de réduire le nombre de matrices et pas le nombre d'états (zones).
— Le réduction des matrices induit une perte d'informations, ce qui peut dégrader la qualité de l'estimation du modèle.
— Le modèle considère uniquement des informations de mobilité macroscopiques.
— La méthode génère des coûts de communication supplémentaires dûs à l'échange d'informations sur la mobilité tel que l'historique des positions.
— Les zones sont créées initialement et statiquement selon les caractéristiques réseau des noeuds Fog.

4.4.7 Améliorations du modèle HTM (Méta-modèle de mobilité et contrôle dynamique de l'environnement mobile)

Dans cette partie, nous proposons un méta-modèle de mobilité dynamique intégrant un système de contrôle de la granularité en partant d'améliorations du modèle précédemment proposé History-based Transition Matrix (HTM).

Comme nous avons pu le voir dans la partie précédente, le modèle HTM est simple d'utilisation mais a un certain nombre de limites, qui dans certains scénarios pourraient rendre les estimations qu'il fournit peu pertinentes pour les stratégies de placement. Par exemple, si nous nous plaçons dans une configuration où les zones de départs regroupent un très grand nombre de noeuds Fog chacune et si l'objet mobile considéré se déplace dans une seule zone.

Le méta-modèle m-HTM regroupe les dimensions temps, espace et les restrictions géographiques et contrairement au modèle HTM, il considère aussi bien des métriques macroscopiques que microscopiques ainsi que des aspects de mobilité individuels ou de groupes. La prise en compte d'un ou de plusieurs aspects cités précédemment dépend de : (1) Les besoins du système de gestion des ressources de l'infrastructure et (2) des coûts résultants du monitoring de la mobilité.

Les modèles se basant sur l'historique de localisation ont un coût dû à l'échange de ces informations entre les noeuds et à leur traitement. Les approches purement microscopiques sont trop lourdes, difficilement applicables en ligne et pour un très grand nombre d'objets mobiles. Pour cela, nous proposons un système qui équilibre entre les deux modes de précisions en choisissant des métriques microscopiques ou macroscopique en fonction du contexte, de l'état de l'infrastructure Fog et des besoins des objets IoT.

Les coûts dûs au système de contrôle de la mobilité sont les suivants :

— Les taux d'utilisation des ressources réseau et de calcul.

— Le temps pour créer et mettre à jour les matrices.

Les besoins du système de gestion de l'infrastructure peuvent être liés à plusieurs aspects. Dans notre cas, on parle de la précision du modèle dont les stratégies de placement ont besoin.

Le méta-modèle essayera d'identifier des motifs de mobilité récurrents et des motifs aléatoires pour chaque objet ou groupe d'objets. Cette approche permet d'éviter l'utilisation des matrices de transition et de directement déduire la prochaine position d'un noeud.

Les zones du modèle HTM étaient définies statiquement selon les spécificités réseau, m-HTM permet de changer les dimensions des zones en les fusionnant ou en les subdivisant dynamiquement au cours du temps selon le besoin et les caractéristiques des zones à chaque instant t. Le nombre de zones et les identifications des zones dans le modèle sont à présent indexés par le temps $n(z_q^t)$.

L'objectif de ce découpage dynamique est d'éviter des précisions non nécessaires considérant des profils utilisateurs non sensibles. De plus, la réduction du nombre de zones permet de réduire le nombre d'états des matrices. En plus de permettre un contrôle du nombre d'états, cette approche réduit considérablement les coûts de l'estimation et de son utilisation pour le placement de services.

Il est important de récolter des méta-data tel que les classes d'applications utilisées selon les zones, par objet. Ceci permet de déterminer les stratégies de divisions/fusion des zones. La zone ayant le plus d'utilisateurs peut bénéficier d'une attention particulière.

En plus de garder le profil utilisateur par type d'application comme le fait le modèle HTM, il est important de garder les moments où ses applications sont demandées et la fréquence de demande par zone. Les horaires "creuses" où il n'y pas de services demandés sont implicitement déduites. Ces horaires peuvent être exploitées pour ajuster le modèle sans impacter les performances des applications utilisateurs.

Le modèle HTM propose de regrouper les objets selon leurs caractéristiques physiques ou selon leur classe d'application. Nous proposons d'ajouter une identification de groupe d'objets dynamique selon la similarité de leurs trajets.

En plus du modèle à base de matrices de transitions (HTM), le méta-modèle considère un ensemble d'autres modèles de la littérature. Le but étant d'affecter le modèle le moins coûteux et le plus approprié aux objets ou aux groupes d'objets.

La méthode proposée est des plus pertinentes pour les architectures Fog qui considèrent souvent des zones ou des cellules de gestion autonomes pouvant se réduire par exemple aux quartiers d'une ville.

De plus, l'approche proposée n'observe pas uniquement les caractéristiques de mobilité des objets à l'instar des méthodes de la littérature précédemment citées mais intègre la nature des objets et leur profil utilisateurs, ainsi que les informations de la zone étudiée.

4.5 Migration et redéploiement de services

Dans cette section, nous considérons que l'infrastructure Fog permet de faire des migrations de services entre les noeuds. On propose donc des formules d'estimation du temps et de l'énergie consommée par le processus de migration. Les formules que nous définirons par la suite sont génériques et indépendantes de toute technologie, algorithme ou stratégie de migration.

On considère que toutes les migrations démarrent au même moment et s'exécutent en parallèle dans un réseau dédié.

4.5.1 Estimation du temps de migration d'un service

On peut intuitivement déduire que le temps dû à la migration d'un service dépend en partie et principalement de la taille mémoire de ce dernier et la bande passante du lien reliant le noeud source au noeud destination.

Plusieurs travaux de la littérature [Strunk 2012], [Kuno 2011], [Salfner 2011], [Akoush 2010] ont proposé un modèle linéaire d'estima-

tion du temps de migration, en considérant la taille mémoire du service et la bande passante du canal de transmission. Établir un modèle pour la consommation énergétique de la migration ne fait pas partie de l'objectif de notre travail. Nous utilisons un modèle générique haut niveau indépendant du type de migration utilisé et de la nature des services à migrer.

Considérant qu'un service s_j^i, placé sur le noeud physique m_k, va être migré vers le noeud $m_{k'}$ à l'instant t_r dit instant de réévaluation, le temps de migration $t_{mig}^{ij}(t_r)$ de ce service est estimé selon l'équation (4.3). On approxime le temps de migration avec le temps de transfert, qui selon plusieurs travaux de la littérature représente la plus grande partie du temps de la migration.

$$\forall s_j^i, t_{mig}^{ij}(t_r) = \frac{ram_j^i}{bw_{kk'}^{t_r}} + lc_{kk'} \tag{4.3}$$

Le temps de migration d'une application a_i défini par t_{mig}^i est estimé par le temps de transfert maximum de ses services. Le temps de migration maximal des services d'une application donnée est le temps où toute l'application reste inaccessible et est défini par l'équation (4.4). De ce temps on peut déduire la dégradation de la qualité de service.Nous rappelons que $lc_{kk'}$ et $bw_{kk'}^{t_r}$ définis dans le chapitre 3, section 3.4 représentent respectivement la latence et la bande passante du lien entre les noeuds m_k et $m_{k'}$. Le besoin en mémoire vive du service s_j^i est défini par ram_j^i.

Le temps de migration total t_{mig} défini par l'équation (4.5) est la valeur maximale des temps de migration des applications.

$$\forall a_i, t_{mig}^i(t_r) = \max_{s_j^i \in \mathbb{S}_i} \{t_{mig}^{ij}(t_r)\} \tag{4.4}$$

$$t_{mig}(t_r) = \max_{a_i \in \mathbb{A}} \{t_{mig}^i(t_r)\} \tag{4.5}$$

4.5.2 Estimation de la consommation énergétique de la migration d'un service

Soit $\mathbb{M}_{mig}$ l'ensemble des couples de machines sources et destination des migrations de services déduites entre la solution de placement à l'instant $t_{r-\tau}$ et la nouvelle solution donnée à t_r .

τ dépend de la stratégie de réévaluation choisie (il peut être périodique ou calculé par d'autres stratégies). On suppose que la puissance active des machines restent constante durant le temps de migration.

Pareillement à des travaux de la littérature [Yu 2016], nous estimons la puissance consommée pendant la migration comme étant la puissance consommée par le transfert des données entre les machines sources et destinations.

Nous supposons que les transferts entre les machines sources et destinations des différentes migrations se font parallèlement. Nous définissons le temps

de migration total t_{mig}^{max}, comme étant le temps le plus long pris entre deux machines.

Soit $y_{kk'}^{uij}$ une variable binaire égale à 1 si le service s_j^i de l'application a_i utilisé par l'objet m_u migre de la machine m_k à la machine $m_{k'}$ à t_r et est égale à 0 sinon. L'équation (4.6) présente l'estimation de la consommation énergétique de la migration de services déclenchée à l'instant t_r. Nous rappelons que $\gamma_{kk'}^n$ défini dans le chapitre 3 section 3.4 est la puissance consommée pour l'échange d'un octet entre les machines m_k et $m_{k'}$.

$$E_{mig}(t_r) = \sum_{(k,k')\in\mathbb{M}_{mig}} \gamma_{kk'}^n \frac{1}{bw_{kk'}^{t_r}} \sum_{m_u\in\mathbb{U}} \sum_{i\in\mathbb{A}_u} \sum_{j\in\mathbb{S}_i} y_{kk'}^{uij} ram_j^i + lc_{kk'} \tag{4.6}$$

L'équation (4.9) définit la fonction coût de migration à l'instant t_r en fonction de l'énergie consommée et du nombre de violations.

On normalise les fonctions $E_{mig}(t_r)$ et $t_{mig}(t_r)$ selon les équations (4.7) et (4.8).

$$f_3(t_r) = \frac{E_{mig}(t_r) - E_{min}}{E_{max} - E_{min}} \tag{4.7}$$

$$f_4(t_r) = \frac{t_{mig}(t_r) - t_{min}}{t_{max} - t_{min}} \tag{4.8}$$

E_{max} étant la somme de l'énergie de toutes les machines à leur utilisation maximale pendant T pas de temps. E_{min} est la somme de l'énergie des machines au repose sur un seul pas de temps. $t_{min} = 0$ est le temps si aucune migration n'est déclenchée. t_{max} est fixé à la durée de la fenêtre de temps.

$$G(t_r) = f_3(t_r) + f_4(t_r) \tag{4.9}$$

4.5.3 Fonction objectif

On présente dans ce qui suit les équations d'estimation déjà définies dans le chapitre 3 et qui sont impactées par la mobilité des noeuds.

On rappelle que les variables α_{ij}^u et $\beta_{ijj'}^u$ présentées dans le chapitre 3 et définies dans le tableau 3.4 représentent respectivement le temps d'exécution du service s_j^i dans la machine m_k et le temps de communication entre les services s_j^i et $s_{j'}^i$ placés respectivement sur les machines m_k et $m_{k'}$. Avec la mobilité des objets IoT les latences de communication varient et donc la variable de communication β présentée par l'équation (4.12) dépend de t.

Comme nous considérons la possibilité de replacer les services, la variable de décision, équation (4.10), ainsi que le temps d'exécution d'un service, équation (4.11), vont également dépendre de t.

$$x_{ijk}^{ut} = \begin{cases} 1, & \text{si le service j de l'application } i \text{ demandé par le noeud } u \\ & \text{est placé dans la machine } k \text{ à l'instant } t. \\ 0, & \text{sinon.} \end{cases} \tag{4.10}$$

$$\alpha_{ijk}^{ut} = \frac{inst_j^i + \sum_{u\in\mathbb{U}} \sum_{a_i\in\mathbb{A}^0} \sum_{s_l^i\in\mathbb{S}_i - s_j^i} inst_l^i x_{ilk}^{ut}}{cpu_k^{max}} \tag{4.11}$$

Pour un objet IoT m_u et une application a_i, nous avons :

$$\beta_{ijj',kk'}^{ut} = \frac{1}{T}\frac{data_{jj'}^i}{bw_{kk'}} + lc_{kk'}^t \tag{4.12}$$

(1) l'énergie de la communication

L'équation (4.13) présente la fonction d'estimation de la consommation énergétique du réseau en fonction des probabilités de positionnement des noeuds à chaque pas de temps. La version statique de cette équation est présentée dans l'équation (3.11) du chapitre 3.

$$f_N(0,T-1) = \sum_{t=0}^{T-1} \sum_{m_u\in\mathbb{U}} \sum_{m_{k''}\in\mathbb{M}-\{m_u\}} P_{uk''}^t \sum_{a_i\in\mathbb{A}_u^t} \sum_{s_j^i\in\mathbb{S}_i} \sum_{m_k\in\mathbb{M}} x_{ijk}^u x_{ij'k'}^u \beta_{ijj',kk'}^{ut} \gamma_{kk'}^n \tag{4.13}$$

On rappelle que $f_1(0,T-1)$ est la somme de l'énergie consommée par le calcul et par la communication sur $[0,T-1]$ avec $f_1(0,T-1) = f_C(0,T-1) + f_N(0,T-1)$.

(2) L'estimation du temps de réponse de l'application

L'équation (4.14) donne l'estimation du temps de réponse de l'application à chaque pas de temps en fonction des positions des objets IoT.

$$\begin{aligned} d_i^u(t) = & \sum_{j\in\mathbb{S}_i} \sum_{k\in\mathbb{M}} x_{ijk}^{ut} \alpha_{ijk}^{ut} \\ & + \sum_{m_{k''}\in\mathbb{M}-\{m_u\}} P_{uk''}^t \sum_{j,j'\in\mathbb{S}_i} \sum_{k,k'\in\mathbb{M}} x_{ijk}^{ut} x_{ij'k'}^{ut} \beta_{ijj',kk'}^{ut} \end{aligned} \tag{4.14}$$

Le nombre moyen de violations de délai est présenté par l'équation 4.15

$$f_2(0,T-1) = \frac{1}{T} \sum_{t=0}^{T-1} \sum_{u\in\mathbb{U}} \sum_{a_i\in\mathbb{A}} w_i^u(t) \tag{4.15}$$

avec $w_i^u(t)$ égale à 1 si le temps de réponse de l'application $d_i^u(t)$ est supé-

	Variable	Type	Description
Problème	x_{ijk}^{ut}	Binaire	Variable de décision à l'instant t
	α_{ijk}^{ut}	$\mathbb{R}$	Temps d'exécution du service s_j^i de l'application a_i placé dans la machine m_k à l'instant t
	$\beta_{ijj',kk'}^{ut}$	$\mathbb{R}$	Temps de communication entre les services s_j^i et $s_{j'}^i$ placés respectivement dans les machines m_k et $m_{k'}$ à l'instant t
	$y_{kk'}^{iju}$	Binaire	Variable de migration du service j de l'objet u et de l'application i de la machine k vers machine k'.
	G	$\mathbb{R}$	Fonction coût de la migration
	H	$\mathbb{N}$	Fonction objectif globale de la réévaluation
	t_r	$\mathbb{N}$	Temps de réévaluation.
Migration	f_3	$\mathbb{R}$	Fonction normalisée du temps de migration.
	f_4	$\mathbb{R}$	Fonction normalisée de l'énergie consommée par la migration.
	t_{mig}	$\mathbb{R}$	Temps de migration global.
	E_{mig}	$\mathbb{R}$	Energie consommée par la migration.
Mobilité	r_k^t	$\mathbb{R}$	Rayon de couverture du noeud k à l'instant t.
	ϑ_u	Symbole	Modèle de mobilité du noeud u.
	P_{uk}^t	$[0,1]$	La probabilité qu'un objet IoT u soit dans la zone de couverture d'un noeud Fog k.
	z_0	Symbole	Zone de mobilité observée.
	$n(z_0)$	$\mathbb{N}$	Zone de mobilité observée.
	$sup(z_0)$	$\mathbb{R}$	Superficie de la zone observée.

TABLE 4.1 – Résumé des variables du problème de placement dynamique.

rieure au délai maximum d_i^{max} et égale à 0 sinon.

(3) La fonction objectif sans réévaluation

Pour les approches sans réévaluation de placement la fonction objectif est la suivante sur l'intervalle $[0, T-1]$.

$$F(0, T-1) = f_1(0, T-1)[1 + f_2(0, T-1)] \tag{4.16}$$

(4) La fonction objectif avec réévaluation

Pour les approches qui considèrent la réévaluation du placement la fonction objectif intègre le temps et l'énergie de la migration à l'instant t_r et la fonction définie par l'équation (4.16) sur l'intervalle $[T - t_r - 1, T - 1]$.

La fonction objectif H est la somme de la fonction F sur l'intervalle $[T - t_r, T]$ et le coût de migration définit par G à l'instant t_r.

$$H(T - t_r - 1, T - 1) = F(T - t_r - 1, T - 1) + G(t_r) \tag{4.17}$$

Le tableau 4.1 résume l'ensemble des variables du problème.

4.6 Approches de résolutions

4.6.1 Placement sans réévaluation

Mobility-aware Genetic Algorithm (MGA) Les approches se basant sur le principe de l'algorithme Génétique (GA) ont montré une grande efficacité pour la résolution de problèmes NP-difficiles et en particulier des problèmes de placement de services dans les environnements Cloud [Liu 2016b]. On a également pu voir son efficacité pour la résolution de problème dans les environnements Fog statiques [Canali 2019], [Brogi 2019]. Un chromosome c est défini par un vecteur de taille N donnant la solution du placement de tous les services des applications. Chaque élément du vecteur a pour index l'indice du service à placer s_j^i et donne l'indice de son noeud Fog m_k .

Les objets IoT sont ordonnés par ordre croissant du nombre d'applications demandées. Les applications de chaque objet sont à leur tour ordonnées par ordre croissant de leur priorité et les services sont ordonnés par ordre croissant de leur demande de calcul processeur.

La fitness des chromosomes est calculée selon l'équation (4.16) où les probabilités de positions des noeuds sont déduites du modèle de mobilité.

Pour chaque objet IoT m_u qui demande une application à un instant t_0 et pour chaque pas de temps $t \in \mathbb{T}$, on récupère l'information de mobilité de m_u qui est la probabilité P_{uk}^t que l'objet m_u soit positionné sous la couverture réseau du noeud Fog m_k. Le MGA est présenté avec le Pseudo Algorithme (4).

Les valeurs des paramètres du MGA ont été choisis expérimentalement en fonction de ses meilleures résultats.

— Taux de mutation $\sigma_1 = 0.01$.
— Taux de crossover $\sigma_2 = 0.8$.
— Nombre de génération $\sigma_3 = 20$.
— Taille de la population $P_{size} = 20$.

Mobility Greedy Heuristic (MGH) La principale idée de l'heuristique MGH est d'estimer les chemins le plus probables des noeuds sur la fenêtre de temps. Ensuite, en fonction de ces chemins, la méthode choisit les placements qui minimisent la consommation énergétique et les temps de communication entre les services.

Les étapes de MGH sont comme suit :

1. Calcul glouton des chemins les plus probables, en sélectionnant la prochaine position du noeud ayant la plus grande probabilité dans la matrice de transition.
2. Les objets IoT sont ordonnés selon le nombre croissant des demandes d'applications.

Algorithm 4 Mobility-aware Genetic Algorithm (MGA) pour le placement de services IoT

Récupérer les probabilités de positionnement pour chaque noeud mobile à l'instant t_0.
Générer uniformément la population initiale de taille P_{size} et calculer les fitness de chaque chromosome définie par l'équation (4.16).
while Le nombre de générations σ_3 n'est pas atteint **do**
  while La taille de la prochaine génération n'atteint pas $\sigma_2 P_{size}$ individus **do**
    Sur $(100 * \sigma_2)\%$ de la population appliquer la sélection par tournoi 1-way pour choisir les parents c_1, c_2.
    Appliquer le Crossover sur c_1, c_2 et obtenir le chromosome c_3 en prenant les premiers gènes de c_1 et le reste de c_2.
    Appliquer la mutation sur c_3 avec la probabilité σ_1.
    Calculer la fitness de c_3 définie par l'équation (4.16) où la partie probabiliste due à la mobilité est détaillée par l'équation (3.11).
    Ajouter c_3 à la population de prochaine génération.
  end while
  Garder $(100 * (1 - \sigma_2))\%$ des meilleurs individus de la population et les injecter dans la nouvelle génération.
end while

3. Les services de chaque application sont ordonnés dans un premier temps par dépendance de données (chemin capteur vers actionneur) ensuite, par ordre croissant de leurs demandes de calcul.
4. Pour chaque pas de temps $t \in T$, objet $m_u \in \mathbb{U}$, application $a_i \in \mathbb{A}_u$, l'algorithme essaie de placer le service $s_j^i \in a_i$ considérant les contraintes de capacités des noeuds et de placement des services. MGH estime l'énergie consommée par le calcul f_C, le temps de communication $d_i^u(t)$ et l'énergie consommée pour la communication f_N du sous graphe composé des services précédemment placés et le nouveau service à placer s_j^i.
   Les valeurs calculées, f_C ,f_N and $d_i^u(t)$ sont normalisées et ensuite le placement qui donne la plus petite valeur de cette formule $Q = \frac{1}{T}(f_C + f_N + d_a^t)$ est choisi. Le placement de s_j^i n'est jamais remis en question lors du placement suivant.

Nous tenons à faire remarquer que la méthode MGH a une formule fitness différente de celle de l'approche MGA mais les objectifs sont similaires, minimiser le nombre de violations de délai et la consommation énergétique. L'approche MGA place les services de toutes les applications au même moment alors il est possible d'estimer le nombre de violations de délai dans l'ensemble d'applications. A contrario, l'approche MGH place les services séquentiellement et pour chaque nouveau service à placer, l'algorithme minimise le temps de communication entre le service à placer et les services précédemment placés.

4.6.2 Placement avec réévaluation

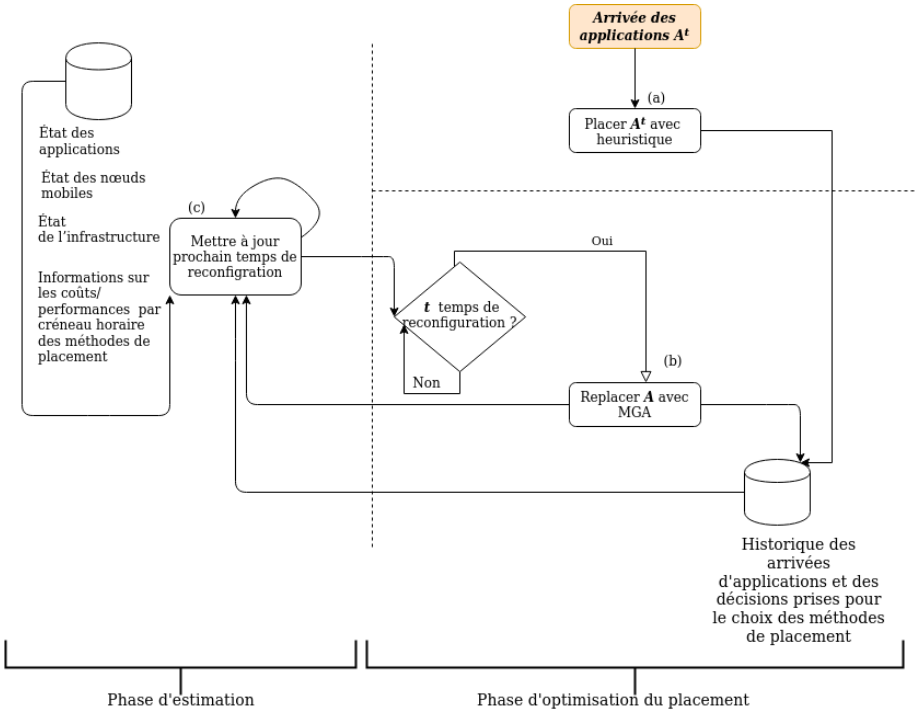


FIGURE 4.3 – Système de placement en ligne de applications IoT en environnement dynamique

Dans cette partie, on considère que le placement peut être réévalué au cours du temps et les services peuvent être déplacés d'une machine à une autre via la migration. Les solutions deviennent datées par l'indice du pas de temps pendant lequel cette solution a été choisie. De plus, les coûts seront calculés sur des sous intervalles qui ne se chevauchent pas dans $[0, T-1]$.

Nous proposons trois heuristiques simples et rapides pour le placement en ligne des services à l'arrivée des applications. Chaque heuristique exploite une dimension du problème pour placer les services le plus rapidement possible.

La première heuristique, Application Classe Greedy Heuristic (AGH) considère le niveau de priorité de la classe d'application pour le placement des services comme décrit dans le Pseudo algorithme 5.

La deuxième heuristique, Infrastructure Greedy Heuristic (IGH), décrite dans le Pseudo algorithme 6, place les services en fonction de la charge de travail des nœuds physiques.

La troisième heuristique, User based Greedy Heuristic (UGH) place les services en fonction des informations sur la mobilité des nœuds des utilisateurs (objets IoT). L'approche est présentée par le Pseudo algorithme 7. Par la suite, le placement est réévalué par l'algorithme génétique à des moments bien précis. Le système de placement est décrit dans la figure 4.3.

Algorithm 5 Applications Class based Greedy Heuristic

1: Ordonner les applications A^t par ordre croissant de leur priorité.
2: Ordonner les services de chaque application par leur dépendance de données, puis par ordre croissant de leur demande de calculs CPU.
3: **while** Toutes les applications $a_i \in \mathbb{A}$ ne sont pas placées **do**
4: **while** Tous les services s_j^i de l'application a_i ne sont pas placés **do**
5: Ordonner les noeuds de calcul par ordre croissant de leur capacité de calcul disponible.
6: Si a_i est de priorité 0 ou de priorité 1 essayer de placer tous les services dans les noeuds IoT ou Fog ensuite le Cloud (choix par first fit).
7: Si a_i est de priorité 3 essayer de placer les services de petite taille dans le Fog et les services de plus grande taille dans le Cloud.
8: Si a_i est de priorité 4 placer tous ses services dans le Cloud.
9: **end while**
10: **end while**

Algorithm 6 Infrastructure State based Heuristic

1: Récupérer l'état de l'infrastructure, sauvegardé lors du placement précédent.
2: Choisir une application aléatoirement dans l'ensemble d'arrivée et choisir les services aléatoirement.
3: **while** Toutes les applications $a_i \in \mathbb{A}$ ne sont pas placées **do**
4: **while** Tous les services s_j^i de l'application a_i ne sont pas placés **do**
5: Ordonner la liste des noeuds Fog par ordre croissant de leur capacité CPU disponible (CPU utilisé total / Maximum cpu de la machine)
6: Placer le service s_j^i dans le premier noeud ayant les capacité suffisantes.
7: **end while**
8: **end while**

Applications based placement heuristic l'heuristique UGH utilise le principe du Round Robin avec priorité sur l'ensemble des objets IoT ayant demandés une application.

- Les objets sont ordonnés par ordre décroissant du nombre d'applications prioritaires.
- L'algorithme construit quatre listes (une liste par classe d'application) et les utilisateurs sont ordonnés par ordre décroissant du nombre d'applications.
- Le Round Robin (RR) est appliqué en commençant par la liste la plus prioritaire.
- Entre deux applications de la même priorité le choix est aléatoire.
- Les noeuds de calcul sont ordonnés par ordre croissant de distance par rapport à l'objet IoT (noeud le plus proche en nombre de sauts).

Algorithm 7 Users based Greedy Heuristic

```
Créer listes d'applications par classe d'application.
Ordonner les objets IoT (utilisateurs) par ordre décroissant du nombre d'applications dans chaque liste.
while Toutes les listes d'applications non vides do
    while Toutes les applications a_i ∈ 𝔸 de la liste la plus prioritaire ne sont pas placées do
        Ordonner les noeud Fog par ordre croissant de distance par rapport à l'objet IoT.
        while Toutes les applications s_j^i de l'application a_i ne sont pas placées do
            Placer s_j^i sur le premier noeud ayant les capacités de calcul suffisantes.
        end while
        Appliquer la sélection RR sur la liste en cour pour choisir la prochaine application à placer.
    end while
    Passer à la prochaine liste.
end while
```

4.6.2.1 Periodic et QoS-aware Mobility Migration Genetic Algorithm

Après le placement rapide des applications par les heuristiques, le placement est réévalué soit périodiquement à chaque pas de temps avec le Priodic Mobility-aware Genetic Algorithm de périodicité égale à 1 (PMGA-1) ou à des instants où le taux de violations de délai des applications dépasse un certain seuil, variable selon leurs classes, avec le QoS-Mobility Genetic Algorithm (QMGA). Dans notre cas on a fixé le taux de violation des applications MC, RT, ST et BE respectivement à 10%, 30%, 40%, 100%.

L'algorithme va re-calculer une nouvelle solution puis en comparant l'ancien placement avec la nouvelle solution, les informations de migrations sont déduites et le coût de ces migrations est estimés par l'équation (4.9).

4.6.2.2 Amélioration du système de placement proposé

On peut décomposer nos propositions d'amélioration du système de placement en deux catégories. La première concerne les moments et/ou les déclencheurs de la méthode de replacement (MGA). La deuxième concerne le choix des heuristiques rapides pour le placement en ligne des applications à leur arrivée.

— Utiliser le MGA périodiquement peut être coûteux en terme de calculs, du nombre de migrations et en temps de résolution. De plus, relancer l'algorithme peut être parfois inutile car l'état du système n'a pas changé et donc l'algorithme ne vas pas trouvé de meilleures solutions ou alors trouver des solutions qui améliorent que très peu les objectifs

visés et par conséquent ne sont pas rentable par rapport aux coûts générés pour le replacement. Le QoS-Mobility Genetic Algorithm (QMGA) ne se lance pas périodiquement mais uniquement si le taux de QoS, représenté ici par une seule métrique, descend au dessus d'un certain seuil. Cette approche avec une définition des critères QoS statique reste inadaptée pour des environnements dynamiques où les besoins en QoS évoluent au cours du temps. De plus, l'algorithme engendre des coûts de communications supplémentaires pour récupérer les informations des métriques QoS. Afin de pouvoir utiliser la réévaluation efficacement (à des moments pertinents pour nos objectifs), il faudrait pouvoir estimer les prochains moments de réévaluation par des méthodes d'estimation et d'apprentissage en fonction des métriques récoltées et de la qualité des placements sur les jours/mois précédents (en partant du principe que le comportement humain dans une ville connaît des motifs répétitifs d'une journée à une autre). De plus, les critères QoS doivent également être déduits dynamiquement et les dépassements de délai doivent être estimés à l'avance.

— Au lieu d'utiliser les heuristiques rapides aléatoirement, il serait intéressant d'avoir une méta-heuristique qui choisit la méthode de placement en fonction de l'ensemble d'applications à placer et de l'état de l'infrastructure Fog.

La figure 4.4 donne une vue globale du système de placement amélioré avec le système de contrôle de mobilité présenté dans le méta modèle de mobilité en section 4.4.7.

On considère un système en deux phases.

— La phase 1e pour l'estimation et la récolte d'informations sur l'environnement de mobilité et l'état de l'infrastructure de calcul.
— La phase 2 pour le placement regroupant le choix de l'heuristique et le replacement avec le MGA.

A l'arrivée des applications A^t, le module (a) choisit l'heuristique gloutonne du placement en fonction de la nature de l'ensemble d'applications arrivées, de la mobilité des objets et de l'état de l'infrastructure. Les informations sur la qualité du placement et les coûts sont sauvegardés. En parallèle, les données sur la mobilité et l'état de l'infrastructure sont mises à jour et seront utilisées pour estimer le prochain temps de réévaluation.

4.7 Résultats

4.7.1 Tests de Validation du modèle de mobilité

On valide le modèle proposé précédemment en section 4.4 avec un ensemble de données de mobilité des taxis de la ville de San Francisco fournis par le

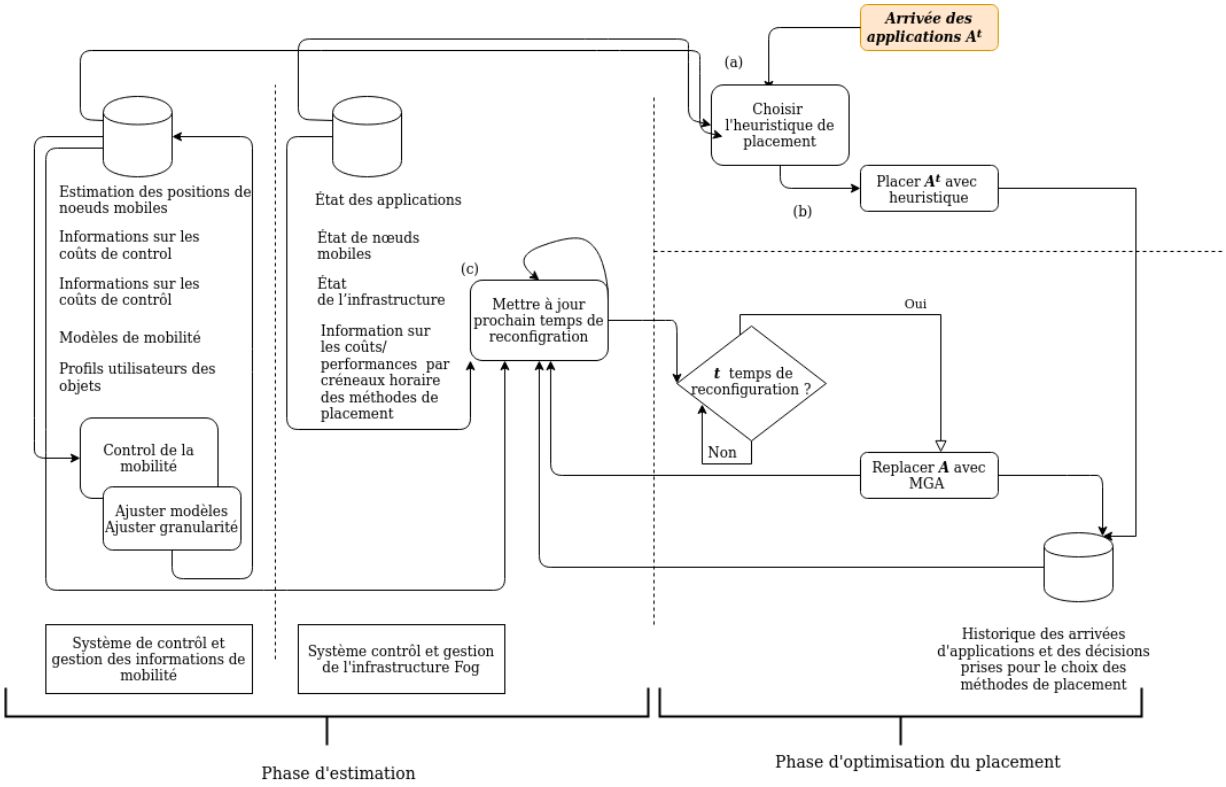


FIGURE 4.4 – Améliorations du système de placement en ligne des applications IoT en environnement dynamique

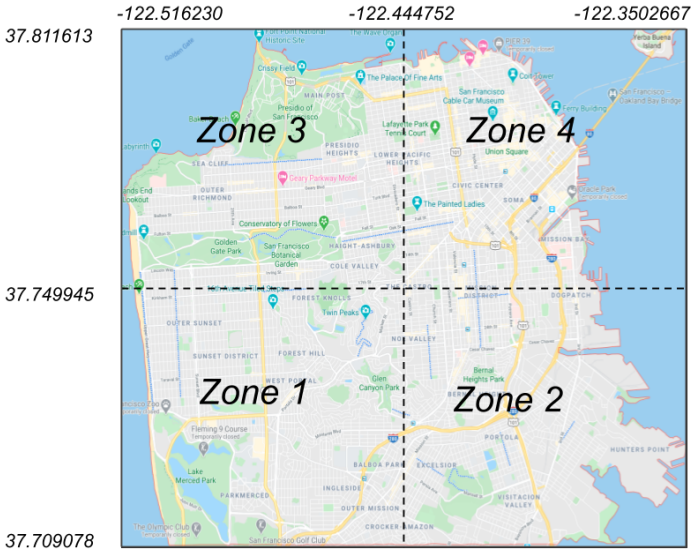


FIGURE 4.5 – Ville de San Francisco subdivisée en quatre zones de mobilité

projet CRAWDAD [Kotz 2009]. Dans un premier temps, nous utilisons ces données seules. Par la suite, on ajoute des données synthétiques de mobilité piétonne, pour tester nos stratégies de réduction de matrices décrites en 4.4.5. L'ensemble des données récoltées dans la ville de San Francisco pendant 30 jours consécutifs [Kotz 2009] regroupe les positions (longitude et latitude) de 500 taxis ainsi que leurs vitesses instantanées. Les données sont envoyées à des fréquences différentes de la journée. A ces données, on ajoute les informations des créneaux horaires, des zones et des profils utilisateurs. La zone z_0 de San Fransisco représentée par la figure 4.5 a une superficie $sup(z_0)$ de $120km^2$ et est subdivisée en 4 zones.

Pour ces expérimentations, nous avons considéré les 24 premiers jours de données pour créer les matrices de transition entre les zones et nous avons estimé les positions des noeuds mobiles sur les 6 derniers jours (convention des 80% des données pour le modèle et 20% pour les tests [Peter 2006], [Nisonger 2008]). On cherche à vérifier si les matrices produites représentent bien les mouvements des utilisateurs les six jours restants. Sous l'hypothèse que les déplacements sont indépendants d'un jour j à un jour $j+1$, les six jours restants sont considérés comme étant 6 instances différentes du 25ème jour.

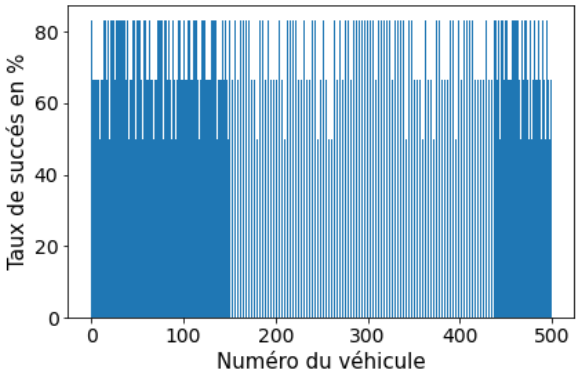


FIGURE 4.6 – Taux de succès moyen de l'estimation des positions des véhicules par véhicule et pour les 6 derniers jours de l'ensemble de données de San Francisco.

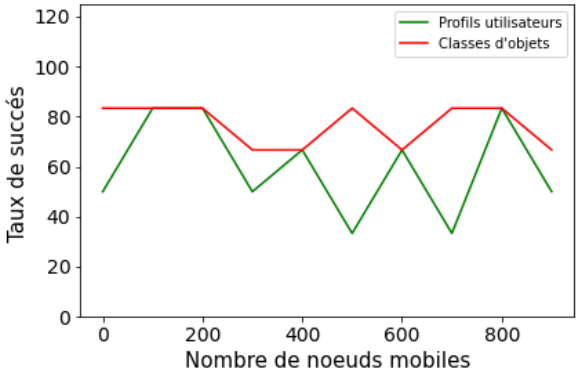


FIGURE 4.7 – Taux de succès de la prédiction des positions par objet pour l'ensemble de données de San Francisco (avec l'ajout des données synthétiques pour piétons) avec réduction de matrices.

La figure 4.6 présente le taux de succès moyen de l'estimation des positions de chaque véhicule au courant des six derniers jours. Le taux de succès moyen de l'estimation sur l'ensemble des 500 véhicules est de 74%.

La figure 4.7 présente le taux de succès moyen de l'estimation sur les 6 derniers jours par objet mobile et pour les deux stratégies de réduction de matrices, classes d'objets et profils utilisateurs. On remarque que la stratégie de réduction par classes d'objets donne de meilleures estimations.

4.7.2 Tests de Validation des équations de migration

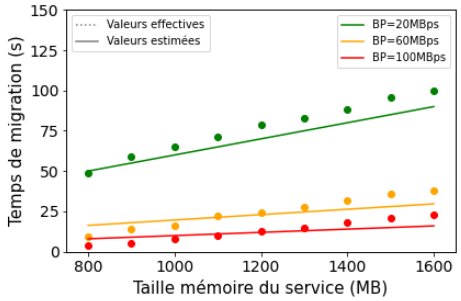
Cette partie présente des expérimentations pour valider les équations d'estimation du temps et de l'énergie de migration décrites respectivement dans les sections 4.5.1 et 4.5.2.

Nous avons effectué la migration d'une machine virtuelle en faisant varier la taille de la mémoire allouée et en variant la bande passant du lien entre la

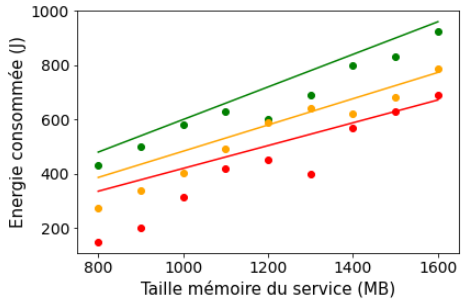
machine source et destination.

Les expérimentations sont faites sur une machine source Intel I5-680 Dual Core 3.6 MHz processor, 4 GB DDR3-1333 SDRAM memory avec une interface réseau Ethernet de 1 Gbit/s et une machine destination Intel® Core i7-1065G7 8 Core 1.30GHz processor, 16 GB avec une interface réseau Ethernet de 1 Gbit/s.

La puissance consommée des interfaces réseau a été mesurée avec l'outil PowerTop et pour réguler la bande passante nous avons utilisé l'outil WonderShaper. Pour la migration du service nous avons utilisé DMTCP avec un programme en C qui affiche une séquence de nombres.



(a) Temps de migration estimé et effectif



(b) Consommation énergétique estimée et effective de la migration

FIGURE 4.8 – Temps et Consommation énergétique estimée et effective de la migration d'un service en fonction de sa taille mémoire et pour différentes valeurs de la bande passante du lien 20 MBps 60 MBps et 100 MBps.

La figure 4.8 présente en 4.8a les temps de migrations estimés et effectifs en secondes et en 4.8b les valeurs de la consommation énergétique estimées et effectives en Joule de la migration d'un service en fonction de sa taille mémoire en MB.

De 4.8a, on remarque que pour toutes les valeurs de la bande passante du lien, les courbes d'estimation du temps suivent la même tendance que les valeurs réelles.

De 4.8b, on constate que l'équation du modèle surestime en moyenne la

consommation énergétique de la migration. Cependant, l'ordre et le taux de croissance sont respectés par les courbes d'estimation pour toutes les valeurs de bande passante.

4.7.3 Tests de Validation des méthodes de placements

Dans cette partie, on présente les performances des méthodes de placement décrites précédemment en environnement dynamique. Le schéma de la figure 4.9 résume les catégories d'expérimentations effectuées. On décompose nos tests en deux parties. L'évaluation des méthodes sans réévaluation de placement et les performances des méthodes avec réévaluation du placement au cours du temps.

Les expérimentations ont été faites sur une version étendue du simulateur MyiFogSim [Lo 2018]. On a fixé la superficie de la zone étudiée ainsi que l'infrastructure de calcul regroupant les noeuds Fog et Cloud. La fenêtre temporelle est fixée à une journée de 24 heures avec un pas de temps de 15 min.

Comme le montre la figure 4.9, les expérimentations se font en deux phases : (1) évaluation des approches de placement sans réévaluation et (2) évaluation des approches de placement avec réévaluation.

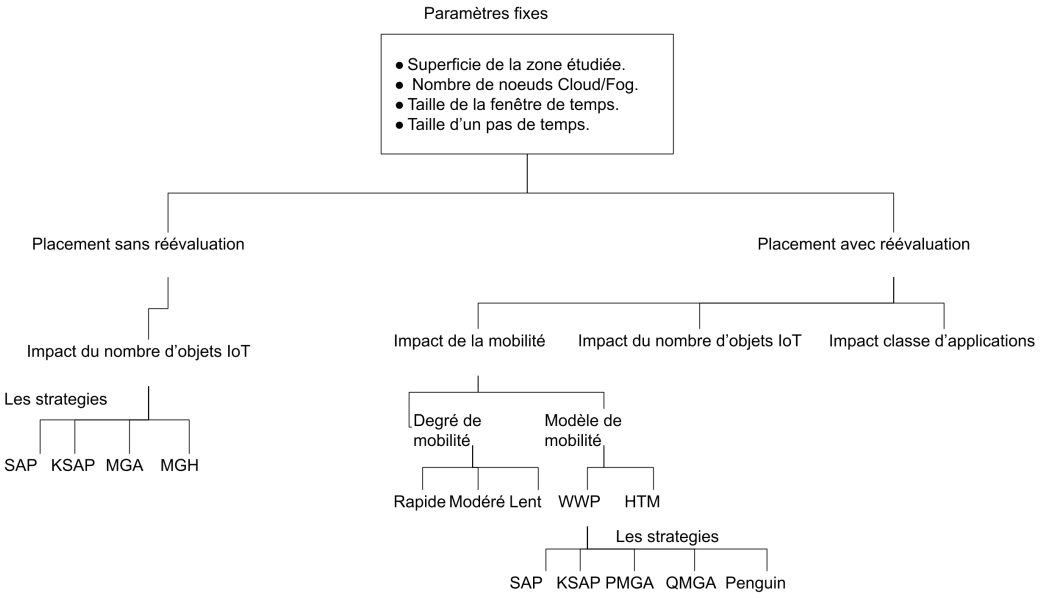


FIGURE 4.9 – Schéma de la méthodologie des expérimentations sur le système de placement avec réévaluation

4.7.3.1 Scénario des expérimentations et méthodes de la littérature

On considère le scénario de la ville de San Fransico où un nombre fixe de piétons et de véhicules se déplacent à l'intérieur de la zone z_0. On considère les smartphones/"wearables" des piétons et les véhicules comme étant les deux classes d'objets IoT mobiles du système.

Les piétons se déplacent entre des endroits bien définis appelés "points d'intérêt" selon le modèle Weighted Waypoints mobility model (WWP) de la littérature [Hsu 2005] et le modèle utilisé pour estimer les déplacements des véhicules est le HTM présenté précédemment. La zone est subdivisée en 6 petites zones contenant un noeud Fog chacune.

Chaque objet IoT utilise des applications interagissant avec ses capteurs et ses actionneurs qui peuvent être de classe Mission Critical (MC), Real-Time (RT), Streaming (ST) ou Best Effort (BE) dont les besoins QoS sont spécifiés dans le tableau 3.1 de la section 3.2.2 du chapitre 3.

4.7.3.2 Les méthodes de comparaison

— Le Static Genetic Algorithm (SGA) est simplement l'algorithme génétique qui ne considère pas l'information de mobilité. Cette méthode a pour but de montrer que la prise en compte de la mobilité dans le MGA et le MGH est effectivement pertinente. Le SGA a exactement les mêmes configurations de paramètre que le MGA.

— Le Shortest Access Point migration Server Cloudlet (SAP) est une méthode de migration dont la stratégie de placement initiale consiste à placer les services des objets sur les noeuds Fog les plus proches si leur capacité le permet. Si les capacités sont insuffisantes, leur placement se fait dans le Cloud. Par la suite, à chaque pas de temps et si il y a un mouvement des objets, leurs services sont migrés vers le noeud Fog le plus proche si ses capacités sont suffisantes et sinon les services restent à leur emplacement initial.

— K-Users Shortest Access Point migration Server (KSAP) consiste à appliquer la même approche de placement et de migration que SAP sur uniquement K objets mobiles choisis aléatoirement. La valeur minimale du paramètre K est de 0 et nous donne une politique de placement sans migrations. La valeur maximale de K est le nombre d'objets mobiles et donne la stratégie SAP. Après expérimentations avec différentes valeurs de K nous présentons la configuration qui donne les résultats les plus avantageux pour la méthode avec K égal à 50% des objets IoT.

— La méta-heuristique Penguins Search Optimization Algorithm (PeSOA) [Gheraibia 2013] est une méthode inspirée du comportement des penguins partant à la recherche de nourriture dans banquise. La méthode Penguin Search Metaheuristic Aware Application Provisioninge

(PsAAP) proposée dans la littérature pour le placement de services dans le Fog [Benamer 2020] a pour objectif de maximiser l'utilisation des ressources. Nous utilisons cette méthode avec la même fitness que le MGA pour les comparaisons du système de placement avec réévaluation. La méthode (PeSOA) est connue pour être efficace dans les environnements dynamiques [Benamer 2020], [Gheraibia 2013]. Notre objectif est de voir s'il serait plus intéressant d'utiliser une autre méta-heuristique pour la réévaluation sachant que jusque là cette comparaison de performance n'a pas été effectuée dans la littérature.

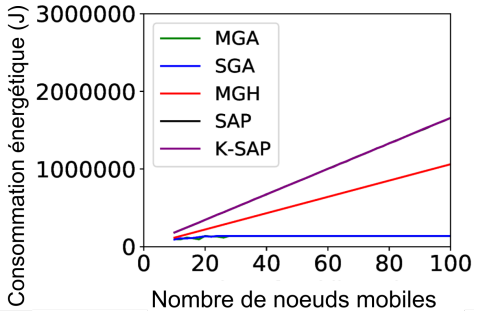
4.7.3.3 Tests sur les méthodes sans réévaluation de placement

Dans cette première partie des expérimentations, nous nous intéressons aux méthodes de placement sans réévaluation MGH et MGA. On compares ces approches à des méthodes de migration de la littérature Shortest Access Point migration Server Cloudlet (SAP) et K-Users Shortest Access Point migration Server Cloudlet (KSAP) et à la méthode Static Genetic Algorithm (SGA). On considère un environnement de noeuds mobiles piétons. L'objectif de ces expérimentations est de montrer les performances des méthodes de placement comparées aux méthodes de migrations de la littérature pour la minimisation de la consommation énergétique et du nombre de violations de délai. Les piétons se déplacent selon le modèle Weighted WayPoint (WWP). Ce modèle a été créé à partir d'inférences statistiques sur des données obtenues par un sondage effectué à l'Université de Californie du sud [Hsu 2005]. Le WWP définit un ensemble de points d'intérêts ou de places, appelées waypoints. La probabilité qu'un noeud mobile se déplace d'un point d'intérêt A à un point d'intérêt B dépend de la localisation courante et du temps. Différentes probabilités de transition sont données considérant des tranches horaires au cours de la journée. Chaque piéton est muni d'un smartphone qui représente ici l'objet IoT.

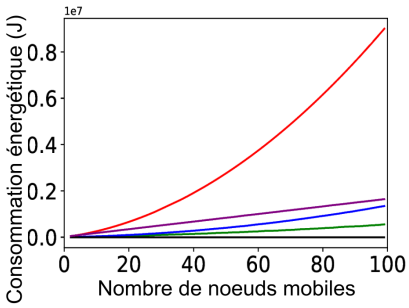
(1) Impact du nombre d'objets IoT sur la consommation énergétique due aux calculs

La figure 4.10a présente la consommation énergétique moyenne du calcul f_C en joules, en fonction du nombre de noeuds mobiles (nombre d'applications) pour chaque méthode de placement et de migration. On remarque que les performances du MGA et du SGA pour minimiser l'énergie du calcul sont assez rapprochées. Ce qui peut être expliqué par le fait que la consommation énergétique due aux calculs des services n'est pas impactée par la mobilité des objets IoT. En effet, les services restent placés dans les mêmes machines au cours du temps.

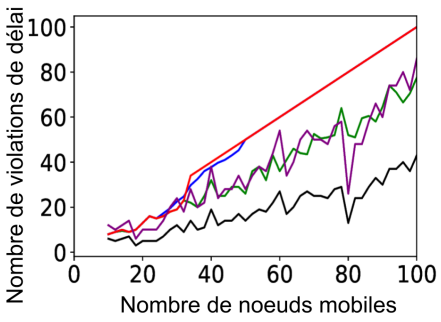
Les deux méta-heuristiques donnent de meilleurs résultats comparé à la méthode MGH. Ce résultat peut être une conséquence de l'approche gloutonne



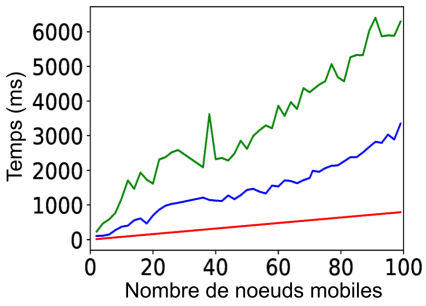
(a) Consommation moyenne d'énergie du calcul en fonction du nombre d'objets IoT.



(b) Consommation moyenne d'énergie de la communication en fonction du nombre d'objets IoT.



(c) Nombre moyen de violations de délais en fonction du nombre d'objets IoT.



(d) Valeur moyenne du temps d'exécution en (ms) des approches en fonction du nombre d'objets IoT.

FIGURE 4.10 – Valeurs moyennes des métriques consommation énergétique (J), nombre de violations de délai et temps d'exécution (ms) en fonction du nombre d'objets IoT et pour chaque méthode de placement.

de la méthode qui ne réévalue pas le placement des services dans leur globalité, ainsi qu'aux erreurs d'estimations des trajets des objets IoT. Les stratégies de migration SAP et KSAP sont les moins performantes. Ces méthodes induisent un coût énergétique supplémentaire due au processus de migration.

(2) Impact du nombre d'objets IoT sur la consommation énergétique due aux communications

La figure 4.10b présente la consommation énergétique due à la communication f_N en Joules en fonction du nombre de noeuds mobiles (nombre d'applications) pour chaque méthode de placement et de migration. On remarque que l'approche SAP, qui migre les services selon les positions des objets IoT, donne les meilleures performances suivie par le MGA, qui tout en évitant les surcoûts énergétiques de la migration donne d'aussi bon résultats pour la minimisation de l'énergie de communication.

Considérant toutes les tailles du problème (en nombre d'objets IoT), le MGA réduit la consommation d'énergie de la communication de 73%, 55% et 93% comparé respectivement aux approches KSAP, SGA and MGH, alors que l'approche SAP surpasse le MGA d'un peu moins de 20%. En migrant uniquement les services de K objets, la méthode KSAP est moins efficace. Le SGA est moins efficace pour minimiser cette métrique car ce dernier considère uniquement les positions initiales des objets pour faire les choix de placement.

(3) Impact du nombre d'objets mobiles sur le nombre de violations de délai

La figure 4.10c présente le nombre moyen de violations de délais f_2 en fonction du nombre d'objets IoT, pour chaque méthode de placement. On remarque que l'approche SAP donne les plus petites valeurs de violations, suivies par MGA, MGH and SGA.

A chaque pas de temps t, SAP migre les services vers les noeuds Fog les plus proches des objets, ce qui réduit les délais de communication entre les objets et leurs services. L'approche réduit en moyenne, pour chaque taille du problème, le nombre de violations de délais de 48% comparé au MGA. L'approche KSAP, en déplaçant uniquement la moitié des services, donne de moins bons résultats. Le MGA a des performances similaires à KSAP avec une réduction moyenne du nombre de violations de 0.68% comparé à KSAP, ce qui est du à son approche probabiliste considérant les informations de mobilité des objets IoT. Le MGA réduit le nombre de violations de 28% et 29% comparé aux approches SGA et MGH.

(4) Impact du nombre d'objets mobiles sur le temps d'exécution des méthodes de placements

La figure 4.10d présente le temps d'exécution moyen des méthodes MGA,

SGA et MGH en fonction du nombre d'objets IoT. Étant une heuristique gloutonne, l'approche MGH est plus rapide comparée aux méthodes SGA et MGA qui de part leur nature de méta-heuristique ont des temps d'exécution plus importants. Le MGA a un temps additionnel pour l'utilisation des informations de mobilité. Cependant, on remarque que le MGA ne prend que 6 secondes pour le placement de 270 services (3x90) dans une infrastructure de 90 objets IoT.

Nous tenons à préciser que les résultats présentés par la figure 4.10d donnent uniquement les temps pris par les méthodes pour le choix du placement et que les méthodes SAP et KSAP sont des stratégies de migration, par conséquent leur temps n'est pas comparable avec celui des méthodes de placements.

4.7.3.4 Tests sur les méthodes avec réévaluation de placement

Dans cette deuxième partie des expérimentations nous considérons l'impact de (1) la mobilité, (2) du nombre d'objets IoT et des classes d'applications sur les performances des méthodes de placement.

(1) Impact de l'environnement mobile sur les méthodes de placement

L'objectif de cette partie des expérimentations est de voir l'impact de la mobilité sur les méthodes de placement via les métriques définies plus haut. On peut diviser l'environnement de mobilité ici de deux façons : (1) le "degré" de mobilité de l'environnement qui détermine d'une manière globale entre un environnement avec des noeuds qui sont souvent en mouvement contre un environnement où les objets sont beaucoup moins actifs au cours du temps. (2) le modèle de mobilité utilisé pour les piétons et les véhicules.

(1.1) Impact du degré de mobilité sur les méthodes de placement

Dans cette sous partie nous allons voir le comportement des méthodes selon le degré de mobilité des noeuds.

Pour cela nous considérons trois scénarios : le scénario de base "mobilité normale" où les objets IoT se déplacent selon un modèle à chaque pas de temps, et deux autres scénarios, "mobilité rapide" et "mobilité lente " où les objets se déplacent respectivement deux fois plus rapidement et deux fois plus lentement par rapport au scénario de base.

Les paramètres suivants, qui pourraient impacter les résultats et les interprétations concernant l'influence de la mobilité, ont été fixés comme suit :

- Le modèle de mobilité homogène : modèle pour piétons (WWP).
- Le nombre d'utilisateurs ou d'objet IoT : 500 smartphones.
- Le nombre d'applications : une application par objet IoT. Une application pouvant avoir entre 3 et 5 services.

— Le taux d'arrivées des applications λ : les applications arrivent suivant une loi de poisson de paramètre $\lambda = \frac{card(\mathbb{U})}{T}$. On rappelle que T est le nombre de pas de temps et que $card(\mathbb{U})$ est le nombre d'objets IoT. Ce ratio implique que la durée moyenne séparant les arrivées est de $\frac{T}{card(\mathbb{U})}$.

— La fenêtre temporelle sur une journée est divisée en 96 pas de temps.

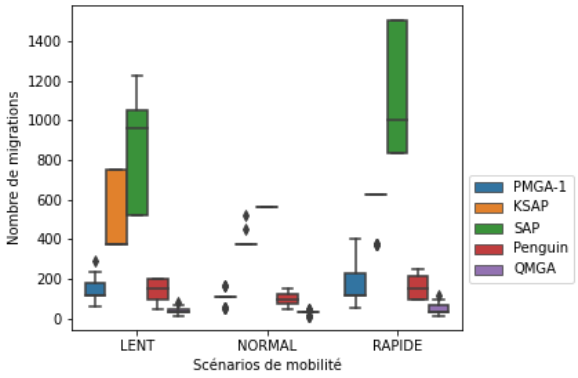


FIGURE 4.11 – Nombre de migrations par méthode et par scénario de mobilité

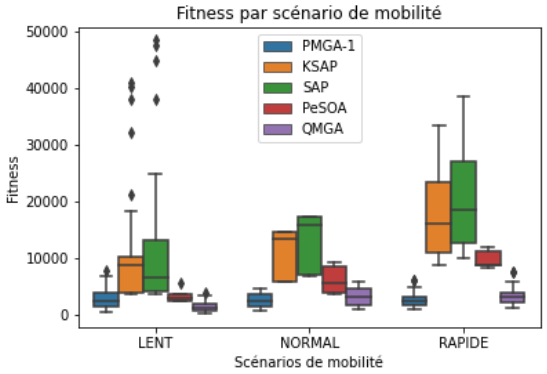


FIGURE 4.12 – Valeur de la Fitness par méthode et par scénario de mobilité (10^3)

La figure 4.11 présente le nombre de migrations pour chaque degré de mobilité et pour chaque méthode. En observant le nombre de migrations des méthodes d'un scénario à un autre, on remarque que le nombre de migrations de toutes les méthodes est proportionnel au degré de mobilité. Autrement dit, plus les noeuds se déplacent fréquemment, plus il y a de migrations. On remarque que le nombre de migrations des méthodes SAP et KSAP est assez élevé quelque soit le scénario et que la différence d'un scénario à un autre dépend de l'augmentation du degré de mobilité des noeuds.

Les méta-heuristiques Periodic Mobility-aware Genetic Algorithm of periodicity 1 (PMGA-1), QoS-Mobility aware Genetic Algorithm (QMGA) et Penguins Search Optimization Algorithm (PeSOA) restent assez stables. Même si

le nombre de migrations augmente du scénario le plus lent au plus rapide, cette augmentation reste négligeable en comparaison à celles des méthodes KSAP et SAP. En considérant les informations de mobilité, les méthodes placent les services de telle sorte à réduire le coût moyen de la fitness (figure 4.17).

On remarque que l'approche QMGA donne le plus faible taux de migration dans le scénario "lent" comparé aux approches PeSOA et PMGA-1. Cependant, la tendance s'inverse en augmentant le degré de mobilité. Le QMGA ne se lance que si un certain seuil de violation de délais est dépassé. Un environnement plus rapide engendre plus de déplacements et donc plus de violations d'applications sensibles.

On remarque que même si le nombre de migrations des méthodes PMGA-1 et PeSOA augmente avec le degré de mobilité, cette augmentation reste moins importante en comparaison à celle de QMGA. En moyenne, le nombre de migrations de la méthode PMGA-1 est plus important respectivement d'un taux de 3% et 17% comparé aux approches PeSOA et QMGA.

Enfin, en remarque générale, nous pouvons constater que le nombre moyen de services migrés par les méta-heuristiques reste assez faible comparé au nombre total placés dans l'infrastructure ($< 18\%$ du nombre total de service).

Le figure 4.12 présente les valeurs de la fitness pour chaque degré de mobilité et pour chaque méthode. On observe que les valeurs de la fitness augmentent proportionnellement avec le degré de mobilité. On remarque également que les méthodes KSAP et SAP ont les valeurs les plus importantes. Ces méthodes ont pour seul objectif de rapprocher les services des objets IoT et le nombre important de migrations qu'elles induisent engendre un coût énergétique supplémentaire .

Pour les méta-heuristiques, on remarque que les performances du QMGA et PMGA-1 sont assez proches. Le QMGA est plus performant dans le scénario "rapide". Pour les autres scénarios le PMGA-1 est plus performant. Ce qui peut être justifié par le fait que dans un environnement plus rapide le QMGA est relancé plus fréquemment (plus de violations) et à des moments plus pertinents que le PMGA-1 (seulement s'il y a dépassement de seuil de violations de délai).

Le PeSOA donne de moins bons résultats que les méthodes QMGA et PMGA-1. Le PeSOA se lance périodiquement comme le PMGA-1, ces résultats sont donc la conséquence du comportement exploratoire de la méthode, moins performant que l'algorithme génétique. On remarque également que PeSOA fonctionne mieux en environnement lent comparé aux autres méthodes qui ne semblent pas être impactées négativement.

En moyenne sur tous les scénarios le PMGA-1 réduit la fitness de 61%, 48%, 15% et 3% comparé respectivement aux méthodes SAP, KSAP, PeSOA et QMGA. Les résultats rapprochés des méthodes QMGA et PMGA-1 peuvent s'expliquer par le fait que le PMGA-1 s'enclenchant périodiquement engendre un coût énergétique supplémentaire dû à un plus grand nombre de migra-

tions par contre il compense en trouvant des placements qui réduisent le coût moyen de la consommation énergétique des services. Le QMGA se déclenche plus rarement et donc réévalue moins fréquemment les solutions de placement précédentes.

Nous pouvons conclure des résultats précédents que les moments de lancement de la politique de réévaluation sont cruciaux et doivent tenir compte du degré de mobilité de l'environnement.

(1.2)Impact du modèle de mobilité sur les méthodes de placement

Dans cette sous partie nous allons voir le comportement des méthodes selon le modèle de mobilité des noeuds.

Nous nous plaçons dans le scénario "mobilité normale" et on considère les deux modèles WWP (pour piéton) et modèle HTM (pour véhicules) avec 250 objets IoT et une application par objet.

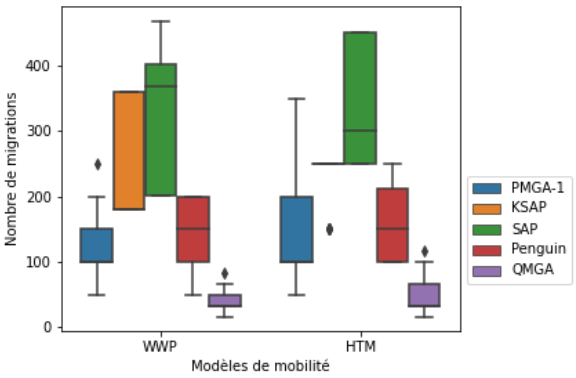


FIGURE 4.13 – Nombre de migrations par méthode et par modèle de mobilité

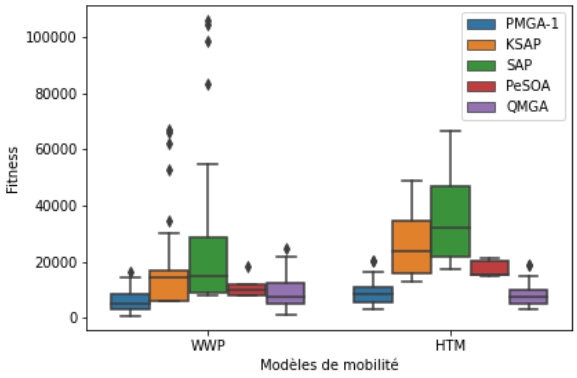


FIGURE 4.14 – Valeur de la fitness par méthode et par modèle de mobilité

La figure 4.13 présente le nombre de migrations pour chaque modèle de mobilité et pour chaque méthode. On constate que le modèle n'influe pas sur le nombre de migrations quelque soit l'approche de placement. On remarque

que pour les deux modèles utilisés (WWP et HTM), le nombre moyen de migrations des méta-heuristiques est assez similaire et que les tendances des méthodes restent les mêmes d'un modèle à un autre. On peut déduire que le modèle n'a que peu d'impact sur les performances des méthodes.

La figure 4.14 présente la valeur de la fitness pour chaque modèle de mobilité et pour chaque méthode. On remarque que les méta-heuristique QMGA et PMGA-1 sont plus performantes avec le modèle HTM. Ce qui peut être justifié par le découpage avantageux des zones où on a un noeud Fog par zone et où dans ce scénario les estimations HTM ont été plus précises que le WWP. Le PeSOA quant à lui donne de meilleurs résultats avec le modèle WWP. Les comparaisons de performance entre les méthodes restent les mêmes quelque soit le modèle. En moyenne le PMGA-1 minimise la fitness de 2%, 18% , 42% et 67% en comparaison avec les méthodes QMGA, PeSOA, KSAP et SAP.

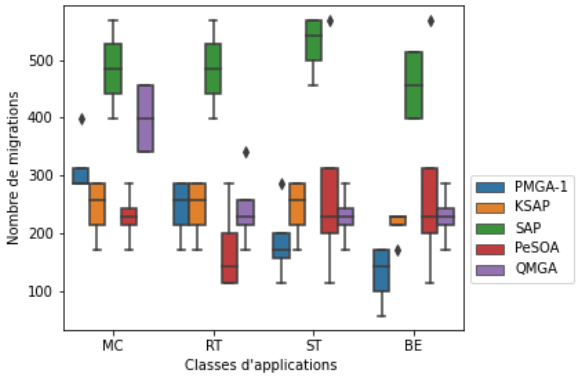


FIGURE 4.15 – Nombre de migrations par méthode et par classes d'applications.

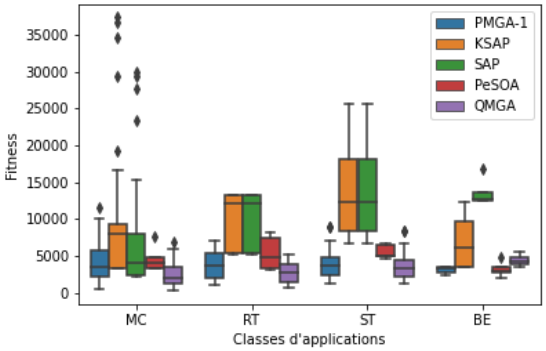
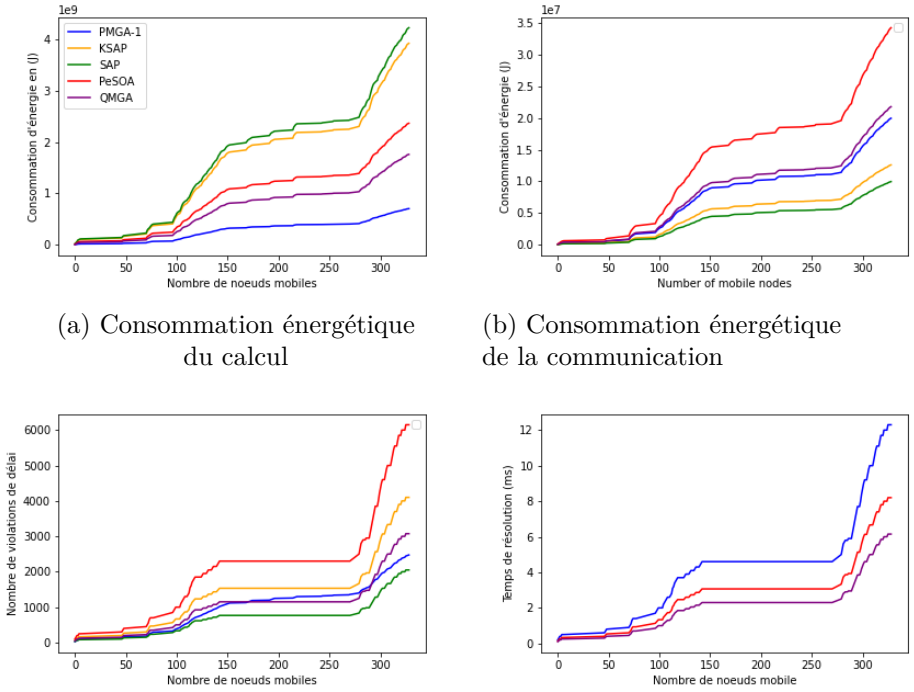


FIGURE 4.16 – Valeur de la fitness par méthode et par classes d'applications.

(2) Impact des classes d'applications sur les stratégies

La figure 4.15 présente le nombre de migrations par classe d'applications pour chaque méthode. On remarque que les approches SAP et KSAP ont les



(a) Consommation énergétique du calcul

(b) Consommation énergétique de la communication

(c) Violations de délais

(d) Temps moyen d'exécution

FIGURE 4.17 – Valeurs moyennes des métriques de consommation énergétique, de violations de délai et du temps d'exécution par nombre de noeuds mobiles et pour chaque stratégie.

mêmes comportements de migration quelque soit la classe d'application. Pour le reste des méthodes on voit que le nombre de migrations des services de classe MC et RT est plus important comparé aux applications ST et BE. On constate également qu'il est plus important pour les méthodes PMGA-1 et QMGA comparé à la méthode PeSOA.

La figure 4.16 présente les valeurs de la fitness en fonction des classes d'applications. On remarque que les méthodes méta-heuristiques sont plus performantes pour la minimisation des classes Best Effort (BE) qui, n'ayant pas de forte contraintes QoS sont plus flexibles pour le placement. On remarque également que les valeurs de la fitness sont plus importantes pour les applications RT et ST qui consomment plus de ressources de calcul et en même temps ont besoin d'être proches des objets à cause de leur contrainte du temps de réponse. Cela engendre beaucoup de migrations et de placements de services répartis entre le Cloud (pour les calculs intensifs) et entre les noeuds Fog/IoT (pour être au plus proche des noeuds IoT).

(3) Impact du nombre d'objets IoT sur les méthodes (scalabilité)

Les figures 4.17a, 4.17b et 4.17c présentent respectivement les valeurs de

l'énergie consommée par le calcul et par la communication et du nombre de violations de délai en fonction du nombre d'objets mobiles pour chaque méthode. On remarque que les méta-heuristiques sont plus performantes pour la minimisation de la consommation d'énergie due aux calculs comparées aux stratégies de migration KSAP et SAP. Le PMGA-1 minimise l'énergie de 83%, 71%, 37%, 14% comparé aux approches SAP, KSAP, PeSOA et QMGA.

Pour l'énergie de communication, on remarque que de part son plus grand nombre de migrations le PMGA est une stratégie de la méthode SAP suivi par le QMGA. Le PMGA minimise en moyenne l'énergie de communication de -13%, -6%, 3% et 29% comparé respectivement aux méthodes SAP, KSAP, QMGA et PeSOA. On remarque que pour la minimisation du nombre de violations de délais les méthodes PMGA et QMGA ont des performance similaires et se rapprochent des résultats des performances de la méthode SAP et sont en moyenne meilleures que la méthode KSAP pour cet objectif. Le PMGA minimise les violation de délai de -5%, 11%, 0.5%, 41% comparée respectivement aux approches SAP, KSAP, QMGA et PeSOA.

La figure 4.17d présente le temps moyen d'exécution des méthodes en fonction du nombre d'objets mobiles. On observe que l'approche la plus rapide est le PeSOA. Le PeSOA est en moyenne 3 fois et 1.5 fois plus rapide que les méthodes PMGA et QMGA.

4.8 Synthèse

Dans ce chapitre, nous avons présenté une extension du modèle et du problème de placement d'applications IoT dans le Fog en intégrant la mobilité des objets IoT. Nous avons par la suite proposé un modèle de mobilité, des méthodes de placement de services et un système de placement intégrant la migration et les informations de mobilité. Dans un deuxième temps, nous avons identifié les limites et inconvénients du système de placement proposé et nous avons présenté une vue générique d'un système de placement amélioré. Enfin, nous avons présenté des résultats expérimentaux qui valident le modèle de mobilité présenté, les équations de migration ainsi que les performances des méthodes de placement. Le tableau 4.2 résume les analyses des expérimentations sur l'évaluation des stratégies de placement en donnant un ordre sur les performances des méthodes pour les objectifs : Énergie, QoS et temps d'exécution pour chaque méthode de placement avec les différents degrés de mobilité présentés et les différentes classes d'applications. La notation $x > y$ implique qu'en moyenne la méthode x est plus performante que la méthode y et la notation $x \approx y$ implique que les méthodes sont en moyenne de performances équivalentes. On remarque que pour les environnements rapides et les classes d'applications critiques la méthode QMGA est meilleure pour l'énergie et la QoS. Pour les environnements plus lents, PMGA-1 est meilleure pour la mini-

misation de l'énergie et pour les applications non critiques les méthodes sont plus ou moins équivalentes. Concernant les temps de placement, la méthode QMGA est la moins coûteuse, suivie par PeSOA et PMGA-1.

	Degré de mobilité			Classes d'applications			
Objectifs	Lent	Normal	Rapide	MC	RT	ST	BE
Énergie	PMGA-1 > *QMGA* > *PeSOA*	PMGA-1 > *QMGA* > *PeSOA*	PMGA-1 ≈ *QMGA* > *PeSOA*	PMGA-1 > *QMGA* > *PeSOA*	*QMGA* ≈ PMGA-1 > *PeSOA*	*QMGA* ≈ PMGA-1 > *PeSOA*	*PeSOA* > PMGA-1 > *QMGA*
QoS	*QMGA* > *PeSOA* > PMGA-1	*QMGA* > PMGA-1 > *PeSOA*	*QMGA* > PMGA-1 > *PeSOA*	*QMGA* > PMGA-1 > *PeSOA*	*QMGA* > PMGA-1 > *PeSOA*	*QMGA* > PMGA-1 > *PeSOA*	-
Temps	*QMGA* > *PeSOA* > PMGA-1	*QMGA* > *PeSOA* > PMGA-1	*QMGA* ≈ *PeSOA* > PMGA-1	*QMGA* > *PeSOA* > PMGA-1	*QMGA* > *PeSOA* > PMGA-1	*QMGA* > *PeSOA* > PMGA-1	*QMGA* > *PeSOA* > PMGA-1

TABLE 4.2 – Tableau résumant le choix des méthodes selon l'objectif, le degré de mobilité de l'environnement et les classes d'applications.

Conclusion et Perspectives

Conclusion

Dans ce travail, nous nous sommes intéressé à la gestion des applications Internet of Things dans les infrastructures Fog Computing. Notre objectif était de réduire l'impact énergétique des infrastructures de calcul et de communication qui, en attendant la transition vers l'utilisation de sources d'énergie propres, est essentiel pour réduire leur impact environnemental et garantir leur pérennité. L'impact énergétique du Fog et la gestion des ressources de calcul et de communication dans ces infrastructures est une problématique assez récente et qui est encore très peu étudiée. Ces premières études permettent d'avoir une idée sur la viabilité des solutions Fog d'un point de vue énergétique pour supporter les applications IoT.

Les applications de l'Internet of Things génèrent énormément de complexité provenant entre autres de leurs hétérogénéités (protocole de communication, demande de calculs, objets IoT etc.), leurs modélisations distribuées en services inter-dépendants et leurs besoins différents en Qualité de Service. Les applications IoT ont pour la plupart de fortes exigences en QoS, principalement traduites par des besoins en temps de réponse et requièrent des délais de traitement quasi-instantanés.

Dans la première partie de ce travail, nous avons proposé une modélisation des systèmes Fog-IoT. Nous avons ensuite abordé le problème de placement de services IoT dans les infrastructures en tant que problème d'optimisation visant à minimiser les violations des délais de réponse et la consommation énergétique de l'infrastructure de calcul et de communication. Nous avons proposé une approche de résolution à base de méta-heuristique à intelligence par essaim et un algorithme génétique que nous avons comparés à des stratégies de placement de la littérature ainsi qu'à des heuristiques proposées. A travers les diverses expérimentations proposées nous avons évalué l'impact de la topologie des applications et du nombre d'objets IoT sur les performances des stratégies proposées. Nous avons également cartographié la manière dont les différents types de noeuds de calcul étaient utilisés par ces méthodes. Suite à cette première partie, nous avons introduit la mobilité des noeuds IoT dans notre modélisation et dans nos expérimentations. La mobilité est en effet un élément clé des environnements Fog-IoT et est la principale cause de la dynamicité et de l'incertitude complexifiant le processus de placement de services IoT. Nous avons proposé un modèle de mobilité ainsi que des stratégies de placement de services en ligne et des stratégies de réévaluation des solutions de placement permettant de migrer des services si besoin. Nous avons également proposé un système de gestion global pour réduire les coûts de ces stratégies

de gestion et d'ajuster le contrôle de la mobilité selon le contexte (état de l'infrastructure) et les besoins des stratégies de placement.

Les problématiques de l'utilisation du Fog/Edge Computing pour déployer et gérer le cycle de vie des applications IoT sont très nombreuses et peuvent être abordées de différentes manières. Nous nous sommes intéressé au placement des services des applications IoT en considérant un problème d'optimisation multi-critères pour minimiser la consommation énergétique des infrastructures Fog et les violations des temps de réponses des applications. Nos travaux peuvent être améliorés et étendus de plusieurs façons. Nous donnons dans ce qui suit quelques indications d'améliorations et d'évolutions.

Perspectives

Les recherches menées au cours de cette thèse ont amené à faire un certain nombre de considérations concernant l'étendu de la problématique abordée, tant au niveau de la modélisation et des approches proposés que des expérimentations effectuées et des outils choisis. Nous présentons ici diverses ouvertures de recherche.

Perspective à court terme

Extension de la modélisation des environnements Fog-IoT

Il s'agit ici d'améliorer le modèle proposé dans le chapitre 3.

— L'amélioration du modèle 3-tiers de l'infrastructure Fog, qui regroupe uniquement trois catégories de noeuds de calcul : Cloud, Fog et IoT peut se faire en considérant plusieurs catégories de noeuds Fog. A chaque noeud peut être associé un profil de consommation énergétique adéquat. Il serait également important de pouvoir modéliser les clusters de noeuds Fog et de proposer des modèles dynamique pour les liens et le routage réseaux. Pour garantir l'accessibilité d'un service ou pour d'autre considérations de sécurité, il serait intéressant de pouvoir intégrer une formule qui évalue le niveau de confiance des noeuds Fog. Ce qui permettrait de savoir si un service critique pourra s'exécuter convenablement une fois placé.

— L'amélioration du modèle des applications peut se faire en différenciant entre les services avec ou sans état et en identifiant si les services peuvent être mutualisés par plusieurs objets IoT ou si chaque objet IoT a son instance du service.

Estimation de la consommation énergétique

Dans ce travail, nous avons utilisé un modèle de consommation énergétique linéaire et simplifié qui dépend uniquement de la consommation CPU et de

la taille des données échangées entre les services. Il serait possible d'utiliser d'autres modèles de consommation d'énergie, probabilistes et qui considèrent plusieurs éléments de calcul et de communication à la fois. Il serait également intéressant d'intégrer des modèles de sources vertes et de faire les placements en considérant cet aspect ainsi que l'état de l'infrastructure ICT.

Intégration de nouvelles métriques de qualité de service (QoS) et du coût économique

Ce travail s'est essentiellement intéressé au temps de réponse des applications. Cependant, les applications IoT peuvent avoir des besoins en QoS sur des ressources réseau (Un besoin spécifique en bande passante) ou sur des ressource de calcul (Un besoin spécifique en mémoire et/ou en puissance de calcul).

Les ressources Fog peuvent être gérées par différents organismes (privé, public, site client). La nature distribuée, hétérogène et dynamique des services IoT (les services sont constamment déplacés, répliqués ou migrés selon la mobilité des utilisateurs) ainsi que la dynamicité des infrastructures avec le traffic qui est constamment redirigé et les services réseau constamment déplacés (SDN/NFV) complexifient l'établissement d'un modèle économique.

Étant donnés les facteurs dynamiques décrits précédemment, il serait intéressant de considérer des critères de QoS comme la fiabilité et la disponibilité d'un service IoT pour établir les décisions de placement. On définit la fiabilité et la disponibilité selon [Banouar 2018] comme suit :

(1) la fiabilité : se réfère à la capacité d'un système ou d'un composant à exécuter ses fonctions dans des conditions données et pour une période de temps spécifiée.

(2) la disponibilité : définie comme le taux de disponibilité opérationnelle d'un système ou d'un composant lorsqu'il est requis pour utilisation. Pour les applications IoT critiques (par exemple, la surveillance des signes vitaux des soins de santé), la disponibilité est une exigence importante que l'on souhaite être très proche des 100%.

Amélioration des méta-heuristiques

Le temps d'exécution considérable des méta-heuristiques est le principal inconvénient qui empêche leur utilisation massive dans des systèmes de décisions réels. Nous estimons que l'établissement d'un critère d'arrêt dynamique qui serait étudié rigoureusement permettrait de considérer les méta-heuristiques comme des méthodes de résolution viables pour les systèmes réels. De plus, les méthodes de parallélisme et d'optimisation distribuée offrent la possibilité d'exploiter les noeuds de calcul Fog (qui sont peu ou pas utilisés) et permettraient de réduire les temps de résolutions des méta-heuristiques. Il serait intéressant d'utiliser des approches d'apprentissage pour réduire les coûts

d'utilisation des méthodes de placement et estimer les instants où la migration ou le replacement des services est nécessaire.

Amélioration du modèle de mobilité

Il est important d'étudier d'autres approches d'estimation et de prédiction comme les méthodes à moyenne mobile auto-régressive intégrée (ARIMA), adéquates pour les séries temporelles, et qui permettrait d'améliorer le modèle de mobilité présenté dans le chapitre 4. Il serait également intéressant d'introduire d'autres stratégies de réduction de matrices par exemple selon la similarité des trajets empruntés pas les noeuds mobiles au courant d'une certaine période de temps. Il faudrait également prévoir d'étudier des modèles de mobilité de groupe et des modèles basés sur les liens sociaux. Cette catégorie de modèles permettrait d'avoir une vue généralisée et d'étudier d'autres stratégies de gestion de ressources de calcul et de communication pouvant être utilisées pour les flottes de véhicules, de drones ou n'importe quel groupe d'objets IoT accomplissant une tache commune.

Amélioration du simulateur et extension des expérimentations

Pour obtenir un environnement Fog-IoT complet en simulation, il faudrait intégrer ns-3 ou un autre simulateur réseau avec MyMobileIFogSim pour supporter les aspects réseau et avoir plus de métriques liées à la communication des services. L'émulateur EmuFog serait également un bon outil à exploiter et qui permettrait de développer une vraie application IoT et de voir son déploiement à une petite échelle pour ensuite passer à son déploiement sur une des plateformes d'expérimentations Fog présentées dans le chapitre 2. Il faudrait également étudier l'impact d'autres topologies Fog et d'autres graphes d'applications IoT sur les méthodes de placement.

Perspectives à long terme

Estimation des instants de réévaluation des placements (reconfiguration)

Les stratégies proposées dans le chapitre 4 sont périodiques ou se basent sur une comparaison de seuil pour réévaluer les placements des services précédents. Il faudrait collecter des métriques sur la qualité des placements ainsi que sur la pertinence de la réévaluation, l'état de l'infrastructure et des applications et d'une manière autonomique [Huebscher 2008], [Rostirolla 2019] le système doit pouvoir estimer les prochains temps de réévaluation de telle sorte à minimiser les coûts de l'utilisation des méthodes de placement, des migrations et de maximiser les gains sur nos critères d'optimisation.

Impact des réseaux de 5ème génération (5G) Les réseaux de 5ème génération promettent des performances et des économies d'énergie sans précédents, ce qui pourrait faciliter le placement de services au plus proche des utilisateurs ou même un placement entièrement dans le Cloud sans impacter la qualité de service des applications. Il serait donc intéressant d'étudier un environnement Fog avec une infrastructure 5G.

Vers un système d'orchestration Fog-IoT distribué Les orchestrateurs actuels qui sont pour la plupart centralisés et exclusifs à la gestion de ressources Cloud, ont des stratégies d'allocation de ressources très simples qui privilégient la rapidité de placement à l'optimisation des ressources. Pour l'énergie, on trouve des stratégies de gestion qui ne peuvent pas être appliquées dans le cas du Fog comme la consolidation des machines virtuelles (voir chapitre 1). La conception des orchestrateurs actuels n'est donc pas adaptées à des infrastructures comme le Fog. Il faudrait des stratégies d'orchestrations multi-domaines qui intègrent la gestion des ressources réseau et des ressources de calcul et qui interagissent avec les gestionnaires des sources énergétiques et les contrôleurs de mobilité.

Le positionnement des orchestrateurs doit être dynamique. Le système doit disposé d'un ou plusieurs orchestrateurs fixes qui peuvent décider, selon le contexte, de placer des orchestrateurs virtuels dans des zones de mobilité données et qui vont gérer les ressources de chaque zone en semi-autonomie avec un modèle MAPE-k [Nguyen 2014] par exemple. L'ensemble des orchestrateurs participerons à des processus d'optimisation distribuée et à des moments précis, les informations sont remontées aux orchestrateurs fixes si nécessaire.

Décider des la superficies des zones semi-autonomes est aussi une problématique à étudier et qui va dépendre d'informations comme le nombre d'utilisateurs, le type d'applications utilisées, la versatilité de ses informations et les politiques de sécurité nécessaire dans chaque zone ainsi que les ressources qu'elle regroupe. Les orchestrateurs pourront ainsi gérer le cycle de vie des applications, en élaborant des stratégies de migration de réplications des services pro-actives à la mobilité des noeuds.

Considérant les capacités de calcul limités des noeuds Fog/IoT, les technologies les plus adéquates pour ces améliorations sont les orchestrateurs de containers tels que Kubernetes, Docker ou LXD qui pourront être combinés à des contrôleurs SDN comme RYU ou Opendaylight. L'algorithme d'orchestration devra se faire en plusieurs phases, où chaque phase vise à optimiser un type de ressource ou à réduire un coût. L'algorithme peut décider en autonomie de ne pas faire toutes les phases en fonction des applications, de l'état de l'infrastructure Fog et des contraintes de temps de réponse données.

Bibliographie

[7-cpu] 7-cpu. *7-Zip LZMA Benchmark*. `https://www.7-cpu.com/`. Accessed : 2020-05-3. 102

[Aazam 2014] Mohammad Aazam, Imran Khan, Aymen Alsaffar et Eui-nam Huh. *Cloud of Things : Integrating Internet of Things and cloud computing and the issues involved*. pages 414–419, 01 2014. 50

[Adhikari 2019] Mainak Adhikari, Tarachand Amgoth et Satish Narayana Srirama. *A Survey on Scheduling Strategies for Workflows in Cloud Environment and Emerging Trends*. ACM Comput. Surv., vol. 52, no. 4, août 2019. 32, 37

[Adjih 2015] Cedric Adjih, Emmanuel Baccelli, Eric Fleury, Gaetan Harter, Nathalie Mitton, Thomas Noël, Roger Pissard-Gibollet, Frederic Saint-Marcel, Guillaume Schreiner, Julien Vandaele et Thomas Watteyne. *FIT IoT-LAB : A large scale open experimental IoT testbed*. 12 2015. 69, 70

[Ahmed 2010] Shabbir Ahmed et Salil Kanhere. *Characterization of a large-scale Delay Tolerant Network*. pages 56 – 63, 11 2010. 125

[Ahvar 2019] E. Ahvar, A. Orgerie et A. Lébre. *Estimating Energy Consumption of Cloud, Fog and Edge Computing Infrastructures*. IEEE Transactions on Sustainable Computing, pages 1–1, 2019. 44, 66

[Aissioui 2018] A. Aissioui, A. Ksentini, A. M. Gueroui et T. Taleb. *On Enabling 5G Automotive Systems Using Follow Me Edge-Cloud Concept*. IEEE Transactions on Vehicular Technology, vol. 67, no. 6, pages 5302–5316, 2018. 52

[Akoush 2010] S. Akoush, R. Sohan, A. Rice, A. W. Moore et A. Hopper. *Predicting the Performance of Virtual Machine Migration*. Dans 2010 IEEE International Symposium on Modeling, Analysis and Simulation of Computer and Telecommunication Systems, pages 37–46, Aug 2010. 136

[Alam 2014] Tanweer Alam, Prabhakar Singh et Parveen Kumar. *SEARCHING MOBILE NODES USING MODIFIED COLUMN MOBILITY MODEL*. International Journal of Computer Science and Mobile Computing, 01 2014. 128

[Alqahtani A 2016] Buyya R Ranjan R Alqahtani A Solaiman E. *End-to-End QoS Specification and Monitoring in the Internet of Things*. IEEE Technical Committee on Cybernetics for Cyber-Physical Systems, 06 2016. 20

[Ama a] *CContrat de niveau de service Amazon S3.* https://d1.awsstatic.com/legal/amazons3service/Amazon_S3_Service_Level_Agreement_-_October_2019_-_16_FRA.pdf. Accessed : 2020-05-3. 21

[Ama b] *Contrat de niveau de service Amazon Compute.* https://d1.awsstatic.com/legal/AmazonComputeServiceLevelAgreement/Amazon_Compute_Service_Level_Agreement_-_October_2019_-_16_FRA%20(1).pdf. Accessed : 2020-05-3. 21

[Amir M. Rahmani 2017] Jürgo-Sören Preden Axel Jantsch Amir M. Rahmani Pasi Liljeberg. Fog computing in the internet of things : Intelligence at the edge. Springer Nature, décembre 2017. 12, 29, 30

[Amira Rayane 2018] Benamer Amira Rayane, Hana Teyeb et Nejib Hadj-Alouane. Latency-aware placement heuristic in fog computing environment, pages 241–257. 10 2018. 54

[Ardagna 2014] Casale G. Ciavotta M. et al. Ardagna D. *Quality-of-service in cloud computing : modeling techniques and their applications.* Journal of Internet Services and Applications, vol. 5, no. 11, 2014. 49

[Arya 2014] L. K. Arya et A. Verma. *Workflow scheduling algorithms in cloud environment - A survey.* Dans 2014 Recent Advances in Engineering and Computational Sciences (RAECS), pages 1–4, 2014. 32

[Babu 2018] R. Babu, Jayashree Kanniappan et R. Abirami. *Fog Computing Qos Review and Open Challenges.* International Journal of Fog Computing, vol. 1, pages 109–118, 07 2018. 50

[Bai] Fan Bai et Ahmed Helmy. *Chapter 1 : A SURVEY OF MOBILITY MODELS in Wireless Adhoc Networks.* 126

[Bai 2003] Fan Bai, Narayanan Sadagopan et Ahmed Helmy. *IMPORTANT : A framework to systematically analyze the Impact of Mobility on Performance of RouTing protocols for Adhoc NeTworks.* volume 2, pages 825 – 835 vol.2, 03 2003. 127, 129

[Baliga 2011] J. Baliga, R. W. A. Ayre, K. Hinton et R. S. Tucker. *Green Cloud Computing : Balancing Energy in Processing, Storage, and Transport.* Proceedings of the IEEE, vol. 99, no. 1, pages 149–167, Jan 2011. 41

[Balte 2015] Anup Satish Balte, Asmita Kashid et B. P. Patil. *Security Issues in Internet of Things (IoT) : A Survey.* 2015. 19

[Banouar 2018] Yassine Banouar. *Gestion autonomique de la QoS au niveau middlewaredans l'IoT.* PhD thesis, Universite Toulouse 3 Paul Sabatier (UT3 Paul Sabatier), 2018. 22, 167

[Barnaghi 2012] Payam Barnaghi, Wei Wang, Cory Henson et KERRY TAYLOR. *Semantics for the Internet of Things : Early Progress and Back*

to the Future. International Journal on Semantic Web Information Systems, vol. 8, 01 2012. 18

[Bathelot 2015] Bertrand Bathelot. *Qualité de service*. `https://www.definitions-marketing.com/definition/qualite-de-service/`, 2015. Accessed : 2020-05-3. 49

[Baumann 2007] R. Baumann, S. Heimlicher et M. May. *Towards Realistic Mobility Models for Vehicular Ad-hoc Networks*. Dans 2007 Mobile Networking for Vehicular Environments, pages 73–78, 2007. 124

[Beh 2018] *Best Uses of Wireless IoT Communication Technology*. `https://industrytoday.com/best-uses-of-wireless-iot-communication-technology/`, 2018. Accessed : 2020-05-3. xvii, 17

[Belkhir 2018] Lotfi Belkhir et Ahmed Elmeligi. *Assessing ICT global emissions footprint : Trends to 2040 recommendations*. Journal of Cleaner Production, vol. 177, pages 448 – 463, 2018. 2

[Beloglazov 2010] A. Beloglazov et R. Buyya. *Energy Efficient Resource Management in Virtualized Cloud Data Centers*. Dans 2010 10th IEEE/ACM International Conference on Cluster, Cloud and Grid Computing, pages 826–831, May 2010. 41

[Beloglazov 2012] Anton Beloglazov et Rajkumar Buyya. *Optimal Online Deterministic Algorithms and Adaptive Heuristics for Energy and Performance Efficient Dynamic Consolidation of Virtual Machines in Cloud Data Centers*. Concurrency and Computation : Practice and Experience, vol. 24, 09 2012. 66, 85

[Benamer 2018] Amira Rayane Benamer, Hana Teyeb et Nejib Ben Hadj-Alouane. *Latency-Aware Placement Heuristic in Fog Computing Environment*. Dans OTM Conferences, 2018. 54

[Benamer 2019] Amira Rayane Benamer, Hana Teyeb et Nejib Ben Hadj-Alouane. *Penguin Search Aware Proactive Application Placement*. Dans ICA3PP, 2019. 54, 58, 59

[Benamer 2020] Amira Rayane Benamer, Hana Teyeb et Nejib Ben Hadj-Alouane. *Penguin Search Aware Proactive Application Placement*. Dans Sheng Wen, Albert Zomaya et Laurence T. Yang, editeurs, Algorithms and Architectures for Parallel Processing, pages 229–244, Cham, 2020. Springer International Publishing. 153

[Bittencourt 2017] L. F. Bittencourt, J. Diaz-Montes, R. Buyya, O. F. Rana et M. Parashar. *Mobility-Aware Application Scheduling in Fog Computing*. IEEE Cloud Computing, vol. 4, no. 2, pages 26–35, 2017. 51, 122

[ble] *intro-to-bluetooth-low-energy.* `https://www.bluetooth.com/bluetooth-resources/intro-to-bluetooth-low-energy/`. Accessed : 2020-05-3. 14

[Bonabeau 1999] Eric Bonabeau, Marco Dorigo et Guy Theraulaz. From natural to artificial swarm intelligence. Oxford University Press, Inc., USA, 1999. 39

[Bonomi 2012a] Flavio Bonomi et Rodolfo Milito. *Fog Computing and its Role in the Internet of Things.* Proceedings of the MCC workshop on Mobile Cloud Computing, 08 2012. 10

[Bonomi 2012b] Flavio Bonomi, Rodolfo Milito, Jiang Zhu et Sateesh Addepalli. *Fog Computing and Its Role in the Internet of Things.* Dans Proceedings of the First Edition of the MCC Workshop on Mobile Cloud Computing, MCC '12, page 13–16, New York, NY, USA, 2012. Association for Computing Machinery. 2, 24

[Brennand 2016] C. A. R. L. Brennand, J. M. Duarte et A. P. Silva. *SimGrid : A simulator of network monitoring topologies for peer-to-peer based computational grids.* Dans 2016 8th IEEE Latin-American Conference on Communications (LATINCOM), pages 1–6, 2016. 65

[Broch 2001] Josh Broch, David Maltz, David Johnson, Yih-chun Hu et Jorjeta Jetcheva. *A Performance Comparison of Multi-Hop Wireless Ad Hoc Network Routing Protocols.* vol. 85-97, 03 2001. 127

[Brogi 2017] A. Brogi et S. Forti. *QoS-Aware Deployment of IoT Applications Through the Fog.* IEEE Internet of Things Journal, vol. 4, no. 5, pages 1185–1192, 2017. 67, 68

[Brogi 2018] Antonio Brogi, Stefano Forti et Ahmad Ibrahim. Predictive analysis to support fog application deployment. 01 2018. xiii, 59

[Brogi 2019] Antonio Brogi, Stefano Forti, Carlos Guerrero et Isaac Lera. *Meet Genetic Algorithms in Monte Carlo : Optimised Placement of Multi-Service Applications in the Fog.* 07 2019. 48, 54, 141

[Bujari 2017] A. Bujari, Carlos Calafate, Juan-Carlos Cano, Pietro Manzoni, Claudio Palazzi et Daniele Ronzani. *Flying ad-hoc network application scenarios and mobility models.* International Journal of Distributed Sensor Networks, vol. 13, page 155014771773819, 10 2017. 128

[Buyya 2019] R. Buyya et S. N. Srirama. Modeling and simulation of fog and edge computing environments using ifogsim toolkit, pages 433–465. 2019. 67, 76, 101

[CAIDAGroup 2019] CAIDAGroup. *Internet topology at router- and AS-levels, and the dual router+AS Internet topology generator.* `https://www.caida.org/research/topology/generator/`, 2019. Accessed : 2020-09-4. 64

[Calheiros 2011] Rodrigo N. Calheiros, Rajiv Ranjan, Anton Beloglazov, César A. F. De Rose et Rajkumar Buyya. *CloudSim : a toolkit for modeling and simulation of cloud computing environments and evaluation of resource provisioning algorithms*. Software : Practice and Experience, vol. 41, no. 1, pages 23–50, 2011. 68

[Canali 2019] Claudia Canali et Riccardo Lancellotti. *GASP : Genetic Algorithms for Service Placement in Fog Computing Systems*. Algorithms, vol. 12, page 201, 09 2019. 48, 54, 141

[CERP-IoT 2010] . CERP-IoT. *Cluster of European Research Porjects on the Internet of Things Vision and Challengesfor Realising theInternet of Things*. `http://www.internet-of-things-research.eu/pdf/IoT_Clusterbook_March_2010.pdf`, 2010. Accessed : 2020-05-3. 11

[Chamas 2017] N. Chamas, F. López-Pires et B. Baran. *Two-phase virtual machine placement algorithms for cloud computing : An experimental evaluation under uncertainty*. Dans 2017 XLIII Latin American Computer Conference (CLEI), pages 1–10, 2017. 87

[Charântola 2019] Danilo Charântola, Alexandre Mestre, Rafael Zane et Luiz Fernando Bittencourt. *Component-based Scheduling for Fog Computing*. pages 3–8, 12 2019. 54

[Chen 2017] Min Chen, Yiming Miao, Yixue Hao et Kai Hwang. *Narrow Band Internet of Things*. IEEE Access, vol. PP, pages 1–1, 09 2017. 16

[Cisco] Cisco. *Cisco iox router*. `https://community.cisco.com/t5/cisco-iox-documents/cisco-iox-nodes-isr819-cgr1120-1240-ir829-809-hardware-and/ta-p/3619076`. Accessed : 2020-05-3. 102

[Cisco 1994] Cisco. *Cisco Discovry protocol*. `https://www.cisco.com/en/US/technologies/tk652/tk701/technologies_white_paper0900aecd804cd46d.html`, 1994. Accessed : 2020-05-3. 132

[Coello 2004] C. A. C. Coello, G. T. Pulido et M. S. Lechuga. *Handling multiple objectives with particle swarm optimization*. IEEE Transactions on Evolutionary Computation, vol. 8, no. 3, pages 256–279, June 2004. 94

[Courchelle 2018] Ines Courchelle, Tom Guerout, Georges Da Costa, Thierry Monteil et Yann Labit. *Green Energy efficient scheduling management*. Simulation Modelling Practice and Theory, vol. 93, 09 2018. 2

[Csr 2017] Prabhu Csr. *Overview - Fog Computing and Internet-of-Things (IOT)*. EAI Endorsed Transactions on Cloud Systems, vol. 3, page 154378, 12 2017. 102

[Cziva 2018] Richard Cziva, Christos Anagnostopoulos et Dimitrios Pezaros. *Dynamic, Latency-Optimal vNF Placement at the Network Edge*. pages 693–701, 04 2018. 51, 122

[dat] *Indicateurs d'activité des opérateurs de communications électroniques*. https://www.data.gouv.fr/fr/datasets/indicateurs-dactivite-des-operateurs-de-communications-electroniques/. Accessed : 2020-05-3. 41

[Deb 2002] K. Deb, A. Pratap, S. Agarwal et T. Meyarivan. *A fast and elitist multiobjective genetic algorithm : NSGA-II*. IEEE Transactions on Evolutionary Computation, vol. 6, no. 2, pages 182–197, April 2002. xiv, 94, 96, 97

[Decker 2015] Kris. De Decker. *Why We Need a Speed Limit for the Internet*. https://www.resilience.org/stories/2015-10-21/why-we-need-a-speed-limit-for-the-internet/, 2015. Accessed : 2020-05-3. 42

[Djemai 2019] T. Djemai, P. Stolf, T. Monteil et J. Pierson. *A Discrete Particle Swarm Optimization Approach for Energy-Efficient IoT Services Placement Over Fog Infrastructures*. Dans 2019 18th International Symposium on Parallel and Distributed Computing (ISPDC), pages 32–40, 2019. 5

[Djemai 2020] T. Djemai, P. Stolf, T. Monteil et J. M. Pierson. *Mobility Support for Energy and QoS aware IoT Services Placement in the Fog*. Dans 2020 International Conference on Software, Telecommunications and Computer Networks (SoftCOM), pages 1–7, 2020. 6

[Dorsemaine 2015] Bruno Dorsemaine, Jean-Philippe Gaulier, jean-philippe Wary, Nizar Kheir et Pascal Urien. *Internet of Things : A Definition Taxonomy*. 09 2015. xiii, 12, 13, 18, 37

[Doyle 1984] Peter G. Doyle et J. Laurie Snell. Random walks and electric networks. Mathematical Association of America, 1984. 127

[DREO 2003] Johann DREO, Alain PETROWSKI, Patrick Siarry et Eric Taillard. Métaheuristiques pour l'optimisation difficile. Algorithmes. EYROLLES, 2003. 37

[Eberhart 2000] R. C. Eberhart et Y. Shi. *Comparing inertia weights and constriction factors in particle swarm optimization*. Dans Proceedings of the 2000 Congress on Evolutionary Computation. CEC00 (Cat. No.00TH8512), volume 1, pages 84–88 vol.1, 2000. 90, 92

[eno] *Energy Harvesting Wireless Power for the Internet*. https://www.enocean.com/. Accessed : 2020-05-3. 40

[ETS] *Internet ofThingsin2020 ROADMAP FOR THE FUTURE*. https://docbox.etsi.org/erm/Open/CERP%2020080609-10/Internet-of-Things_in_2020_EC-EPoSS_Workshop_Report_2008_v1-1.pdf. Accessed : 2020-05-3. 11

[Fakhfakh 2014] Fairouz Fakhfakh, Hatem Hadj Kacem et Ahmed Hadj Kacem. *Workflow Scheduling in Cloud Computing : A Survey.* pages 372–378, 09 2014. 32, 33, 36

[Farmer 1986] J.Doyne Farmer, Norman H Packard et Alan S Perelson. *The immune system, adaptation, and machine learning.* Physica D : Nonlinear Phenomena, vol. 22, no. 1, pages 187 – 204, 1986. Proceedings of the Fifth Annual International Conference. 40

[Ferrboeuf 2019] Hugue. Ferrboeuf. *Lean ICT-Toward Digital Sobriety.* `https://theshiftproject.org/wp-content/uploads/2019/03/Lean-ICT-Report_The-Shift-Project_2019.pdf`, 2019. Accessed : 2020-05-3. 40

[Foteinos 2013] Vassilis Foteinos, Dimitris Kelaidonis, George Poulios, Vera Stavroulaki, Panagiotis Vlacheas, Panagiotis Demestichas, Raffaele Giaffreda, Abdur Rahim Biswas, Stephane Menoret, Gerard Nguengang, Matti Etelapera, Nechifor Septimiu-Cosmin, Marc Roelands, Filippo Visintainer et Klaus Moessner. *A Cognitive Management Framework for Empowering the Internet of Things.* Dans Alex Galis et Anastasius Gavras, editeurs, The Future Internet, pages 187–199, Berlin, Heidelberg, 2013. Springer Berlin Heidelberg. 18

[Füßler 2006] Holger Füßler, Marc Torrent-Moreno, Matthias Transier, Roland Krüger, Hannes Hartenstein et Wolfgang Effelsberg. *Studying vehicle movements on highways and their impact on Ad Hoc connectivity.* ACM SIGMOBILE Mobile Computing and Communications Review, vol. 10, pages 26–27, 10 2006. 125

[G. Montenegro] J. Hui G. Montenegro N. Kushalnagar et D. Culler. *Transmission of IPv6 Packets over IEEE 802.15.4 Networks.* `https://tools.ietf.org/html/rfc4944`. Accessed : 2020-05-3. 15

[Galati 2014] Adriano Galati, Karim Djemame, Martyn Fletcher, Mark Jessop, Michael Weeks et John McAvoy. *A WS-Agreement Based SLA Implementation for the CMAC Platform.* volume 8914, pages 159–171, 09 2014. 21

[Gheraibia 2013] Youcef Gheraibia et Abdelouahab Moussaoui. *Penguins Search Optimization Algorithm (PeSOA).* Dans Moonis Ali, Tibor Bosse, Koen V. Hindriks, Mark Hoogendoorn, Catholijn M. Jonker et Jan Treur, editeurs, Recent Trends in Applied Artificial Intelligence, pages 222–231, Berlin, Heidelberg, 2013. Springer Berlin Heidelberg. 40, 152, 153

[Giang 2015] N. K. Giang, M. Blackstock, R. Lea et V. C. M. Leung. *Developing IoT applications in the Fog : A Distributed Dataflow approach.* Dans 2015 5th International Conference on the Internet of Things (IOT), pages 155–162, 2015. 57, 76

[Goyal 2012] Tarun Goyal, Ajit Singh et Aakanksha Agrawal. *Cloudsim : Simulator for cloud computing infrastructure and modeling*. Procedia Engineering, vol. 38, pages 3566–3572, 12 2012. 66

[Grange 2018] Léo Grange, Georges Da Costa et Patricia Stolf. *Green IT scheduling for data center powered with renewable energy*. Future Generation Computer Systems, vol. 86, 03 2018. 2

[Gre 2020] *CLICKING CLEAN : WHO IS WINNING THE RACE TO BUILD A GREEN INTERNET ?*, 2020. 41

[Grzybek 2012] Agata Grzybek, Marcin Seredynski, Grégoire Danoy et Pascal Bouvry. *Aspects and trends in realistic VANET simulations*. pages 1–6, 06 2012. 124

[Gschwandtner 2014] P. Gschwandtner, M. Knobloch, B. Mohr, D. Pleiter et T. Fahringer. *Modeling CPU Energy Consumption of HPC Applications on the IBM POWER7*. Dans 2014 22nd Euromicro International Conference on Parallel, Distributed, and Network-Based Processing, pages 536–543, 2014. 85

[Guerrero 2018] Carlos Guerrero, Isaac Lera et Carlos Juiz. *A lightweight decentralized service placement policy for performance optimization in fog computing*. Journal of Ambient Intelligence and Humanized Computing, Jun 2018. 48

[Guevara 2017] J. C. Guevara, L. F. Bittencourt et N. L. S. da Fonseca. *Class of service in fog computing*. Dans 2017 IEEE 9th Latin-American Conference on Communications (LATINCOM), pages 1–6, 2017. 78, 102

[Gunasekaran 2009] S. Gunasekaran et N. Nagarajan. *An Improved Realistic Group Mobility Model for MANET Based On Unified Relationship Matrix*. 03 2009. 128

[Gupta 2016] Harshit Gupta, Amir Vahid Dastjerdi, Soumya Ghosh et Rajkumar Buyya. *iFogSim : A Toolkit for Modeling and Simulation of Resource Management Techniques in Internet of Things, Edge and Fog Computing Environments*. Software : Practice and Experience, 06 2016. xiii, 57, 58, 68, 71, 100

[Hagberg 2008] Aric A. Hagberg, Daniel A. Schult et Pieter J. Swart. *Exploring Network Structure, Dynamics, and Function using NetworkX*. Dans Gaël Varoquaux, Travis Vaught et Jarrod Millman, editeurs, Proceedings of the 7th Python in Science Conference, pages 11 – 15, Pasadena, CA USA, 2008. 64

[Halfacree 2018] Gareth Halfacree. *Benchmarking the Raspberry Pi 3 B+*. `https://medium.com/@ghalfacree/benchmarking-the-raspberry-pi-3-b-plus-44122cf3d806`, 2018. Accessed : 2020-05-3. 102

[Hazas 2016] Mike Hazas, Janine Morley, Oliver Bates et Adrian Friday. *Are there limits to growth in data traffic ? : on time use, data generation and speed*. pages 1–5, 06 2016. 43

[Hilman 2020] Muhammad Hilman, Maria Rodriguez et Rajkumar Buyya. *Multiple Workflows Scheduling in Multi-tenant Distributed Systems : A Taxonomy and Future Directions*. ACM Computing Surveys (CSUR), vol. 53, pages 1–39, 02 2020. 76

[Hong 1999] Xiaoyan Hong, Mario Gerla, Guangyu Pei et Ching-Chuan Chiang. *A Group Mobility Model for Ad Hoc Wireless Networks*. Dans Proceedings of the 2nd ACM International Workshop on Modeling, Analysis and Simulation of Wireless and Mobile Systems, MSWiM '99, page 53–60, New York, NY, USA, 1999. Association for Computing Machinery. 128

[Hong 2013] Kirak Hong, David Lillethun, Umakishore Ramachandran, Beate Ottenwälder et Boris Koldehofe. *Mobile Fog : A Programming Model for Large-Scale Applications on the Internet of Things*. Dans Proceedings of the Second ACM SIGCOMM Workshop on Mobile Cloud Computing, MCC '13, page 15–20, New York, NY, USA, 2013. Association for Computing Machinery. 57, 58

[Horwitz 2019] Lauren Horwitz. *The future of IoT miniguide : The burgeoning IoT market continues*. `https://www.cisco.com/c/en/us/solutions/internet-of-things/future-of-iot.html`, 2019. Accessed : 2020-08-4. 2

[Hossain 2009] M. S. Hossain et M. Atiquzzaman. *Stochastic Properties and Application of City Section Mobility Model*. Dans GLOBECOM 2009 - 2009 IEEE Global Telecommunications Conference, pages 1–6, 2009. 127

[Hsu 2005] Wei-jen Hsu, Kashyap Merchant, Haw-wei Shu, Chih-hsin Hsu et Ahmed Helmy. *Weighted waypoint mobility model and its impact on ad hoc networks*. ACM SIGMOBILE Mobile Computing and Communications Review, vol. 9, pages 59–63, 01 2005. 127, 129, 152, 153

[Huebscher 2008] Markus Huebscher et Julie McCann. *A survey of Autonomic Computing - Degrees, models, and applications*. ACM Comput. Surv., vol. 40, 08 2008. 168

[IAE 2016] IAE. *"Energy Efficiency of the Internet of Things, 4E EDNA Technology and Energy Assessment Report"*, 2016. 43

[IEA 2017] . IEA. *Digitalisation and Energy, IEA, Paris 2017*. `https://www.iea.org/reports/digitalisation-and-energy`, 2017. Accessed : 2020-05-3. 13, 42, 112

[IEA 2019] . IEA. *IEA (2019), World Energy Outlook 2019*. https://www.iea.org/reports/world-energy-outlook-2019, 2019. Accessed : 2020-05-3. 41, 43, 51

[IEE 1990] *IEEE Standard Glossary of Software Engineering Terminology*. IEEE Std 610.12-1990, pages 1–84, 1990. 22

[IEEE 2002] IEEE. *Link Layer Discovery Protocol and MIB v0.0*. https://www.ieee802.org/1/files/public/docs2002/lldp-protocol-00.pdfl, 2002. Accessed : 2020-05-3. 132

[Ihara 2015] Diego Ihara, Fabio Lopez-Pires et Benjamin Baran. *Many-Objective Virtual Machine Placement for Dynamic Environments*. 12 2015. 87

[Intel 2015] Intel. *Intel® Xeon® Processor E7-8870 v3*. https://ark.intel.com/content/www/us/en/ark/products/84682/intel-xeon-processor-e7-8870-v3-45m-cache-2-10-ghz.html, 2015. Accessed : 2020-05-3. 102

[Izakian 2009] H. Izakian, B. T. Ladani, A. Abraham et V. Snáel. *A DISCRETE PARTICLE SWARM OPTIMIZATION APPROACH FOR GRID JOB SCHEDULING*. 2009. 91, 104

[Jalali 2016] F. Jalali, K. Hinton, R. Ayre, T. Alpcan et R. S. Tucker. *Fog Computing May Help to Save Energy in Cloud Computing*. IEEE Journal on Selected Areas in Communications, vol. 34, no. 5, pages 1728–1739, 2016. 46

[Jalali 2017] Fatemeh Jalali. *Greening IoT with Fog : A Survey*. 06 2017. 41, 43, 45

[Jonathan 2012] Bar-Magen Numhauser Jonathan. *Fog Computing introduction to a New Cloud Evolution*. PhD thesis, University of Alcala, Spain, 2012. 24

[Karaboga 2007] Dervis Karaboga et Bahriye Basturk. *Artificial Bee Colony (ABC) Optimization Algorithm for Solving Constrained Optimization Problems*. Dans Patricia Melin, Oscar Castillo, Luis T. Aguilar, Janusz Kacprzyk et Witold Pedrycz, editeurs, Foundations of Fuzzy Logic and Soft Computing, pages 789–798, Berlin, Heidelberg, 2007. Springer Berlin Heidelberg. 40

[Karagiorgou 2012] Sophia Karagiorgou, Georgios Stamoulis et Panagiotis Kikiras. *Enabling QoS in the Internet of Things*. CTRQ 2012, The Fifth ..., 01 2012. 2

[Kennedy 1995] J. Kennedy et R. Eberhart. *Particle swarm optimization*. Dans Proceedings of ICNN'95 - International Conference on Neural Networks, volume 4, pages 1942–1948 vol.4, 1995. 87, 90, 92, 93, 94

[Kennedy 1997] J. Kennedy et R. C. Eberhart. *A discrete binary version of the particle swarm algorithm.* Dans 1997 IEEE International Conference on Systems, Man, and Cybernetics. Computational Cybernetics and Simulation, volume 5, pages 4104–4108 vol.5, 1997. 89

[Keti 2015] F. Keti et S. Askar. *Emulation of Software Defined Networks Using Mininet in Different Simulation Environments.* Dans 2015 6th International Conference on Intelligent Systems, Modelling and Simulation, pages 205–210, 2015. 68

[Khattar 2019] Nagma Khattar, Jagpreet Sidhu et Jaiteg Singh. *Toward energy-efficient cloud computing : a survey of dynamic power management and heuristics-based optimization techniques.* The Journal of Supercomputing, vol. 75, 08 2019. 2

[Khosravi 2013] Atefeh Khosravi, Saurabh Garg et Rajkumar Buyya. *Energy and Carbon-Efficient Placement of Virtual Machines in Distributed Cloud Data Centers.* pages 317–328, 08 2013. 43

[Khosravi 2017] Atefeh Khosravi, achlan Andrew et Rajkumar Buyya. *Dynamic VM Placement Method for Minimizing Energy and Carbon Cost in Geographically Distributed Cloud Data Centers.* IEEE Transactions on Sustainable Computing, vol. PP, pages 1–1, 06 2017. 43

[Kliazovich 2015] Dzmitry Kliazovich, Pascal Bouvry, Fabrizio Granelli et Nelson Fonseca. Energy consumption optimization in cloud data centers, pages 191–215. 04 2015. 43

[Kotz 2009] David Kotz, Tristan Henderson, Ilya Abyzov et Jihwang Yeo. *CRAWDAD dataset dartmouth/campus (v. 2009-09-09).* Downloaded from https ://crawdad.org/dartmouth/campus/20090909, septembre 2009. 125, 148

[Krajzewicz 2002] Daniel Krajzewicz, Georg Hertkorn, Christian Feld et Peter Wagner. *SUMO (Simulation of Urban MObility) ; An open-source traffic simulation.* pages 183–187, 01 2002. 65

[Krief 2012] Francine Krief. Le green networking : Vers des réseaux efficaces en consommation énergétique. Hermès - Lavoisier, 2012. 71, 82

[Ksentini 2014] A. Ksentini, T. Taleb et M. Chen. *A Markov Decision Process-based service migration procedure for follow me cloud.* pages 1350–1354, 2014. 52

[Kumar 2010] K. Kumar et Y. Lu. *Cloud Computing for Mobile Users : Can Offloading Computation Save Energy ?* Computer, vol. 43, no. 4, pages 51–56, April 2010. 41

[Kumari 2015] Kanta Kumari et Sunil Maakar. *A Survey : Different Mobility Model for FANET.* 06 2015. 127

[Kuno 2011] Y. Kuno, K. Nii et S. Yamaguchi. *A Study on Performance of Processes in Migrating Virtual Machines*. Dans 2011 Tenth International Symposium on Autonomous Decentralized Systems, pages 567–572, March 2011. 136

[Kurowski 2013] K. Kurowski, A. Oleksiak, W. Piątek, T. Piontek, A. Przybyszewski et J. Węglarz. *DCworms – A tool for simulation of energy efficiency in distributed computing infrastructures*. Simulation Modelling Practice and Theory, vol. 39, pages 135 – 151, 2013. S.I.Energy efficiency in grids and clouds. 66

[Law 2000] Averill Law et David Kelton. *Simulation Modeling and Analysis*. 01 2000. 126

[Lera 2019a] I. Lera, C. Guerrero et C. Juiz. *YAFS : A Simulator for IoT Scenarios in Fog Computing*. IEEE Access, vol. 7, pages 91745–91758, 2019. 67

[Lera 2019b] Isaac Lera, Carlos Guerrero et Carlos Juiz. *YAFS : A simulator for IoT scenarios in fog computing*. IEEE Access, vol. PP, pages 1–1, 07 2019. 67

[Li 2003] Xiaodong Li. *A Non-dominated Sorting Particle Swarm Optimizer for Multiobjective Optimization*. Dans Erick Cantú-Paz, James A. Foster, Kalyanmoy Deb, Lawrence David Davis, Rajkumar Roy, Una-May O'Reilly, Hans-Georg Beyer, Russell Standish, Graham Kendall, Stewart Wilson, Mark Harman, Joachim Wegener, Dipankar Dasgupta, Mitch A. Potter, Alan C. Schultz, Kathryn A. Dowsland, Natasha Jonoska et Julian Miller, editeurs, Genetic and Evolutionary Computation — GECCO 2003, pages 37–48, Berlin, Heidelberg, 2003. Springer Berlin Heidelberg. 94

[Liang 2003] Ben Liang et Haas Zygmunt. *Predictive Distance-Based Mobility Management for Multidimensional PCS Networks*. IEEE/ACM Trans. Netw., vol. 11, pages 718–732, 10 2003. 127

[Liu 2002] Y. Liu et K. Passino. *Passino, K.M. : Biomimicry of Social Foraging Bacteria for Distributed Optimization : Models, Principles, and Emergent Behaviors. Journal of Optimization Theory And Applications 115(3), 603-628*. Journal of Optimization Theory and Applications, vol. 115, pages 603–628, 12 2002. 40

[Liu 2016a] D. Liu, X. Sui et L. Li. *An energy-efficient virtual machine placement algorithm in cloud data center*. Dans 2016 12th International Conference on Natural Computation, Fuzzy Systems and Knowledge Discovery (ICNC-FSKD), pages 719–723, Aug 2016. 87

[Liu 2016b] D. Liu, X. Sui et L. Li. *An energy-efficient virtual machine placement algorithm in cloud data center*. Dans 2016 12th International

Conference on Natural Computation, Fuzzy Systems and Knowledge Discovery (ICNC-FSKD), pages 719–723, 2016. 98, 141

[Liu 2018] L. Liu, Z. Chang, X. Guo, S. Mao et T. Ristaniemi. *Multiobjective Optimization for Computation Offloading in Fog Computing.* IEEE Internet of Things Journal, vol. 5, no. 1, pages 283–294, 2018. 48

[Lo 2018] Lo. *MyiFogSim : A Simulator for Virtual Machine Migration in Fog Computing.* pages 47–52, 1pes, Márcio and Higashino, Wilson and Capretz, Miriam and Bittencourt, Luiz Fernando 2018. 67, 69, 151

[LoRa 2009] . LoRa. *LoRa.* `https://lora-alliance.org/`, 2009. Accessed : 2020-05-3. 16

[Mahmud 2019] R. Mahmud, A. N. Toosi, K. Rao et R. Buyya. *Context-aware Placement of Industry 4.0 Applications in Fog Computing Environments.* IEEE Transactions on Industrial Informatics, pages 1–1, 2019. 54, 59

[Mahmud 2020] Md Mahmud, Kotagiri Ramamohanarao et Rajkumar Buyya. *Application Management in Fog Computing Environments : A Taxonomy, Review and Future Directions.* ACM Computing Surveys, 05 2020. xiii, xvii, 19, 20, 29, 31

[Maia 2019a] Adyson Maia, Yacine Ghamri-Doudane, Dario Vieira et Miguel Franklin de Castro. *Optimized Placement of Scalable IoT Services in Edge Computing.* 04 2019. 54

[Maia 2019b] Adyson Maia, Yacine Ghamri-Doudane, Dario Vieira et Miguel Franklin de Castro. *Optimized Placement of Scalable IoT Services in Edge Computing.* 04 2019. 54

[Mann 2019a] Zoltan Mann, Andreas Metzger, Johannes Prade et Robert Seidl. Optimized application deployment in the fog, pages 283–298. 10 2019. xiii, 59, 60

[Mann 2019b] Zoltán Ádám Mann, Andreas Metzger, Johannes Prade et Robert Seidl. *Optimized Application Deployment in the Fog.* Dans Sami Yangui, Ismael Bouassida Rodriguez, Khalil Drira et Zahir Tari, editeurs, Service-Oriented Computing, pages 283–298, Cham, 2019. Springer International Publishing. 54

[Marini 2015] Federico Marini et Beata Walczak. *Particle swarm optimization (PSO). A tutorial.* Chemometrics and Intelligent Laboratory Systems, vol. 149, pages 153 – 165, 2015. 90, 92, 94, 104

[Markus 2020] Andras Markus et Attila Kertesz. *A survey and taxonomy of simulation environments modelling fog computing.* Simulation Modelling Practice and Theory, vol. 101, page 102042, 2020. Modeling and Simulation of Fog Computing. 66

[Mayer 2017] R. Mayer, L. Graser, H. Gupta, E. Saurez et U. Ramachandran. *EmuFog : Extensible and scalable emulation of large-scale fog computing infrastructures.* Dans 2017 IEEE Fog World Congress (FWC), pages 1–6, 2017. 67, 68

[Mebrek 2017] Adila Mebrek, Leïla Merghem-Boulahia et Moez Esseghir. *Efficient green solution for a balanced energy consumption and delay in the IoT-Fog-Cloud computing.* pages 1–4, 10 2017. 48

[Medina 2001] Alberto Medina, Anukool Lakhina, Ibrahim Matta et John Byers. *BRITE : an approach to universal topology generation.* pages 346–353, 02 2001. 64

[Michaela 2018a] Iorga Michaela, Feldman Larry, Barton Robert, J.Martin Michael, Goren Nedim et Mahmoudi Charif. *Fog Computing Conceptual Model Recommendations of the National Institute of Standards and Technology.* Rapport technique, National Institute of Standards and Technology, 2018. 2, 24, 25, 26, 29

[Michaela 2018b] Iorga Michaela, Feldman Larry, Barton Robert, J.Martin Michael, Goren Nedim et Mahmoudi Charif. *Fog Computing Conceptual Model Recommendations of the National Institute of Standards and Technology.* Rapport technique, National Institute of Standards and Technology, 2018. 21

[Moreno 2019] Gabriel A. Moreno, Cody Kinneer, Ashutosh Pandey et David Garlan. *DARTSim : An Exemplar for Evaluation and Comparison of Self-Adaptation Approaches for Smart Cyber-Physical Systems.* Dans Proceedings of the 14th International Symposium on Software Engineering for Adaptive and Self-Managing Systems, SEAMS '19, page 181–187. IEEE Press, 2019. 66

[Mouradian 2019] C. Mouradian, S. Kianpisheh, M. Abu-Lebdeh, F. Ebrahimnezhad, N. T. Jahromi et R. H. Glitho. *Application Component Placement in NFV-Based Hybrid Cloud/Fog Systems With Mobile Fog Nodes.* IEEE Journal on Selected Areas in Communications, vol. 37, no. 5, pages 1130–1143, 2019. 51

[Mouël 2019] Frédéric Le Mouël. *YOUPI : feedbacks, ongoing and future works of a Fog/Edge Computing platform.* `https://www.irit.fr/journeescloud2019/wp-content/uploads/2019/09/11-JourneesCloud19-LeMouel.pdf`, 2019. Accessed : 2020-08-4. 69, 70

[Munjal 2011] Aarti Munjal, Tracy Camp et William Navidi. *SMOOTH : a simple way to model human mobility.* pages 351–360, 10 2011. 127

[Muñoz Zavala 2013] Angel Eduardo Muñoz Zavala. *A Comparison Study of PSO Neighborhoods.* Dans Oliver Schütze, Carlos A. Coello Coello,

Alexandru-Adrian Tantar, Emilia Tantar, Pascal Bouvry, Pierre Del Moral et Pierrick Legrand, editeurs, EVOLVE - A Bridge between Probability, Set Oriented Numerics, and Evolutionary Computation II, pages 251–265, Berlin, Heidelberg, 2013. Springer Berlin Heidelberg. 94

[Musolesi 2006] Mirco Musolesi et Cecilia Mascolo. *A Community Based Mobility Model for Ad Hoc Network Research.* REALMAN '06, page 31–38, New York, NY, USA, 2006. Association for Computing Machinery. 128

[Musolesi 2009] Mirco Musolesi et Cecilia Mascolo. Mobility Models for Systems Evaluation, page 43. 2009. 124

[N. Kushalnagar] G. Montenegro N. Kushalnagar et C. Schumacher. *IPv6 over Low-Power Wireless Personal Area Networks (6LoWPANs) : Overview, Assumptions, Problem Statement, and Goals.* `https://tools.ietf.org/html/rfc4919`. Accessed : 2020-05-3. 15

[Naboulsi 2013] Diala Naboulsi et Marco Fiore. *On the Instantaneous Topology of a Large-Scale Urban Vehicular Network : the Cologne Case.* Proceedings of the International Symposium on Mobile Ad Hoc Networking and Computing (MobiHoc), 07 2013. 125

[Nardelli 2019] M. Nardelli, V. Cardellini, V. Grassi et F. L. Presti. *Efficient Operator Placement for Distributed Data Stream Processing Applications.* IEEE Transactions on Parallel and Distributed Systems, vol. 30, no. 8, pages 1753–1767, 2019. 48, 54

[NATF 2015] . NATF. *Impact of ICT on world energy consumption.* `https://www.euro-case.org/impact-of-ict-on-world-energy-consumption/`, 2015. Accessed : 2020-08-4. 2

[Nguyen 2014] Thod Nguyen. *Media : Analytics, Algorithms, AI, Big Data.* `https://researcher.watson.ibm.com/researcher/view_group.php?id=4933`, 2014. Accessed : 2020-08-4. 2, 169

[Nisonger 2008] Thomas Nisonger. *The "80/20 Rule" and Core Journals.* Serials Librarian - SERIALS LIBR, vol. 55, pages 62–84, 07 2008. 148

[NRD 2014] *Data Center Efficiency Assessment : scaling up energy efficiency across the Data Center Industry : evaluating Key Drivers and Barriers*, 2014. 42

[Ojima 2018] T. Ojima et T. Fujii. *Resource management for mobile edge computing using user mobility prediction.* Dans 2018 International Conference on Information Networking (ICOIN), pages 718–720, 2018. 52

[om2m 2015] . om2m. *Eclipse OM2M Opense Source.* `https://www.eclipse.org/om2m/`, 2015. Accessed : 2020-05-3. 56

[Orgerie 2014] Anne-Cecile Orgerie, Marcos Dias de Assuncao et Laurent Lefevre. *A Survey on Techniques for Improving the Energy Efficiency of Large-Scale Distributed Systems.* ACM Comput. Surv., vol. 46, no. 4, mars 2014. xiii, 47

[Ouyang 2018] T. Ouyang, Z. Zhou et X. Chen. *Follow Me at the Edge : Mobility-Aware Dynamic Service Placement for Mobile Edge Computing.* IEEE Journal on Selected Areas in Communications, vol. 36, no. 10, pages 2333–2345, Oct 2018. 52

[Pandey 2010] S. Pandey, L. Wu, S. M. Guru et R. Buyya. *A Particle Swarm Optimization-Based Heuristic for Scheduling Workflow Applications in Cloud Computing Environments.* Dans 2010 24th IEEE International Conference on Advanced Information Networking and Applications, pages 400–407, April 2010. 87

[Perera 2014] C. Perera, A. Zaslavsky, P. Christen et D. Georgakopoulos. *Context Aware Computing for The Internet of Things : A Survey.* IEEE Communications Surveys Tutorials, vol. 16, no. 1, pages 414–454, 2014. 11

[Perez Abreu 2020a] David Perez Abreu, Karima Velasquez, Marilia Curado et Edmundo Monteiro. *A comparative analysis of simulators for the Cloud to Fog continuum.* Simulation Modelling Practice and Theory, vol. 101, page 102029, 2020. Modeling and Simulation of Fog Computing. 58, 59

[Perez Abreu 2020b] David Perez Abreu, Karima Velasquez, Marilia Curado et Edmundo Monteiro. *A comparative analysis of simulators for the Cloud to Fog continuum.* Simulation Modelling Practice and Theory, vol. 101, page 102029, 2020. Modeling and Simulation of Fog Computing. 66

[Peter 2006] Erridge Peter. *The Pareto principle.* 201, vol. 419, 10 2006. 148

[Pierson 2019] J. Pierson, G. Baudic, S. Caux, B. Celik, G. Da Costa, L. Grange, M. Haddad, J. Lecuivre, J. Nicod, L. Philippe, V. Rehn-Sonigo, R. Roche, G. Rostirolla, A. Sayah, P. Stolf, M. Thi et C. Varnier. *DATAZERO : Datacenter With Zero Emission and Robust Management Using Renewable Energy.* IEEE Access, vol. 7, pages 103209–103230, 2019. 1

[Pires 2015a] F. L. Pires et B. Barán. *A Virtual Machine Placement Taxonomy.* Dans 2015 15th IEEE/ACM International Symposium on Cluster, Cloud and Grid Computing, pages 159–168, 2015. 87

[Pires 2015b] Fabio Lopez Pires et Benjamín Barán. *Virtual Machine Placement Literature Review.* CoRR, vol. abs/1506.01509, 2015. 87

[Qayyum 2018a] Tariq Qayyum, Asad Malik, Muazzam Khan, Osman Khalid et Samee Khan. *FogNetSim++ : A Toolkit for Modeling and Simulation of Distributed Fog Environment*. IEEE Access, vol. PP, pages 1–1, 10 2018. 67

[Qayyum 2018b] Tariq Qayyum, Asad Malik, Muazzam Khan, Osman Khalid et Samee Khan. *FogNetSim++ : A Toolkit for Modeling and Simulation of Distributed Fog Environment*. IEEE Access, vol. PP, pages 1–1, 10 2018. 67

[Ray 2018] P.P. Ray. *A survey on Internet of Things architectures*. Journal of King Saud University - Computer and Information Sciences, vol. 30, no. 3, pages 291 – 319, 2018. 1

[Rezazadeh 2018] Z. Rezazadeh, D. Rahbari et M. Nickray. *Optimized Module Placement in IoT Applications Based on Fog Computing*. Dans Electrical Engineering (ICEE), Iranian Conference on, pages 1553–1558, 2018. 54

[rfc 2014] *Terminology for Constrained-Node Networks*. `https://tools.ietf.org/html/rfc7228`, 2014. Accessed : 2020-05-3. 13

[Ricart 2020] Glen Ricart. *Application Latency usignit*. `http://inlab.lab.asu.edu/nsf/files/NSF-workshop-2-Ricart.pdf`, 2020. Accessed : 2020-05-3. xiii, xvii, 23, 24

[Riley 2010] George F. Riley et Thomas R. Henderson. The ns-3 network simulator, pages 15–34. Springer Berlin Heidelberg, Berlin, Heidelberg, 2010. 64

[Rostirolla 2019] Gustavo Rostirolla. *Ordonnancement dans un centre de calculs alimenté par des sources d'énergie renouvelables sans connexion au réseau avec une charge de travail mixte basée sur des phases*. PhD thesis, 2019. Thèse de doctorat dirigée par Stolf, Patricia et Caux, Stéphane Informatique et Télécommunications Toulouse 3 2019. 66, 168

[Salfner 2011] Felix Salfner, Peter Tr et Andreas Polze. *Downtime Analysis of Virtual Machine Live Migration*. 2011. 136

[Shehabi 2018] Arman Shehabi, Sarah Smith, Eric Masanet et Jonathan Koomey. *Data center growth in the United States : Decoupling the demand for services from electricity use*. Environmental Research Letters, vol. 13, 10 2018. 42

[Sigfox 2010] . Sigfox. *sigfox*. `https://www.sigfox.com/en`, 2010. Accessed : 2020-05-3. 16

[Silva 2015] Fabrício A. Silva, Azzedine Boukerche, Thais R.M.B. Silva, Linnyer B. Ruiz et Antonio A.F. Loureiro. *A novel macroscopic mobility*

model for vehicular networks. Computer Networks, vol. 79, pages 188 – 202, 2015. 124, 125, 126

[Simon 2009] Dan Simon. *Biogeography-Based Optimization*. Evolutionary Computation, IEEE Transactions on, vol. 12, pages 702 – 713, 01 2009. 40

[Skarlat 2017a] O. Skarlat, M. Nardelli, S. Schulte et S. Dustdar. *Towards QoS-Aware Fog Service Placement*. Dans 2017 IEEE 1st International Conference on Fog and Edge Computing (ICFEC), pages 89–96, May 2017. 87

[Skarlat 2017b] Olena Skarlat, Matteo Nardelli, Stefan Schulte, Michael Borkowski et Philipp Leitner. *Optimized IoT Service Placement in the Fog*. Service Oriented Computing and Applications, vol. 11, pages 427–443, 10 2017. 48, 54, 87

[Sofia 2013] Rute Sofia et Andrea Ribeiro. *A survey on mobility models for wireless networks*. vol. SITI-TR-11-1, 07 2013. 127

[Sommer 2011] Christoph Sommer, Reinhard German et Falko Dressler. *Bidirectionally Coupled Network and Road Traffic Simulation for Improved IVC Analysis*. IEEE Transactions on Mobile Computing (TMC), vol. 10, no. 1, pages 3–15, January 2011. 65

[Sonmez 2017] C. Sonmez, A. Ozgovde et C. Ersoy. *EdgeCloudSim : An environment for performance evaluation of Edge Computing systems*. Dans 2017 Second International Conference on Fog and Mobile Edge Computing (FMEC), pages 39–44, 2017. 67

[Sonmez 2018] Cagatay Sonmez, Atay Ozgovde et Cem Ersoy. *EdgeCloudSim : An environment for performance evaluation of edge computing systems*. Transactions on Emerging Telecommunications Technologies, vol. 29, page e3493, 08 2018. 67

[Strunk 2012] A. Strunk. *Costs of Virtual Machine Live Migration : A Survey*. Dans 2012 IEEE Eighth World Congress on Services, pages 323–329, June 2012. 136

[Sá 2014] Thiago Sá, Rodrigo Calheiros et Danielo Gomes. Cloudreports : An extensible simulation tool for energy-aware cloud computing environments, pages 127–142. 10 2014. 66

[Taneja 2017] M. Taneja et A. Davy. *Resource aware placement of IoT application modules in Fog-Cloud Computing Paradigm*. Dans 2017 IFIP/IEEE Symposium on Integrated Network and Service Management (IM), pages 1222–1228, 2017. 104

[Thi 2019] Minh-Thuyen Thi, Jean-Marc Pierson, Georges Da Costa, Patricia Stolf, Jean-Marc Nicod, Gustavo Rostirolla et Marwa Haddad. *Negotiation Game for Joint IT and Energy Management in Green Datacenters*. Future Generation Computer Systems, vol. 110, 11 2019. 1

[Thiam 2019] C. Thiam et F. Thiam. *optimizing electrical energy consumption in cloud data center*. Dans 2019 Third International Conference on Intelligent Computing in Data Sciences (ICDS), pages 1–5, Oct 2019. 43

[Tran 2017] Minh Quang Tran, Duy Nguyen, Van An Le, Hai Nguyen et Anh Truong. *Toward service placement on Fog computing landscape*. pages 291–296, 11 2017. 48, 54

[tsu 2017] *Tsunami of data' could consume one fifth of global electricity by 2025*. `https://www.theguardian.com/environment/2017/dec/11/tsunami-of-data-could-consume-fifth-global-electricity-by-2025`, 2017. Accessed : 2020-05-3. 40

[Uppoor 2012] Sandesh Uppoor et Marco Fiore. *Insights on Metropolitan-Scale Vehicular Mobility from a Networking Perspective*. 06 2012. 125

[Uppoor 2014] S. Uppoor, O. Trullols-Cruces, M. Fiore et J. M. Barcelo-Ordinas. *Generation and Analysis of a Large-Scale Urban Vehicular Mobility Dataset*. IEEE Transactions on Mobile Computing, vol. 13, no. 5, pages 1061–1075, 2014. 125

[Varga 2010] Andras Varga. Omnet++, pages 35–59. Springer Berlin Heidelberg, Berlin, Heidelberg, 2010. 64

[Vastardis 2012] Nikolaos Vastardis et Kuanli Yang. *An enhanced community-based mobility model for distributed mobile social networks*. Journal of Ambient Intelligence and Humanized Computing, vol. 5, 02 2012. 128

[Vermesan 2009] Ovidiu Vermesan, Peter Friess, Patrick Guillemin, Sergio Gusmeroli, Harald Sundmaeker, Alessandro Bassi, Ignacio Jubert, Margaretha Mazura, Mark Harrison, Markus Eisenhauer et Pat Doody. *Internet of Things Strategic Research Roadmap*. 01 2009. 1

[Wang 2015] S. Wang, R. Urgaonkar, M. Zafer, T. He, K. Chan et K. K. Leung. *Dynamic service migration in mobile edge-clouds*. Dans 2015 IFIP Networking Conference (IFIP Networking), pages 1–9, May 2015. 52

[Wang 2018a] F. Wang et G. Wang. *Study on Energy Minimization Data Transmission Strategy in Mobile Cloud Computing*. Dans 2018 IEEE SmartWorld, Ubiquitous Intelligence Computing, Advanced Trusted Computing, Scalable Computing Communications, Cloud Big Data Computing, Internet of People and Smart City Innovation (SmartWorld/SCALCOM/UIC/ATC/CBDCom/IOP/SCI), pages 1211–1218, 2018. 3

[Wang 2018b] Pan Wang, Shidong Liu, Feng Ye et Xuejiao Chen. *A Fog-based Architecture and Programming Model for IoT Applications in the Smart Grid*. CoRR, vol. abs/1804.01239, 2018. 102

[Wang 2019] S. Wang, R. Urgaonkar, M. Zafer, T. He, K. Chan et K. K. Leung. *Dynamic Service Migration in Mobile Edge Computing Based on Markov Decision Process.* IEEE/ACM Transactions on Networking, vol. 27, no. 3, pages 1272–1288, 2019. 52

[Wegener 2005] Ingo Wegener et R. Pruim. Complexity theory : Exploring the limits of efficient algorithms. Springer-Verlag, Berlin, Heidelberg, 2005. 35

[Wi-Fi] Aliance. Wi-Fi. *wi-fi.* `https://www.wi-fi.org/`. Accessed : 2020-05-3. 15

[Wickremasinghe 2010] Bhathiya Wickremasinghe, Rodrigo Calheiros et Rajkumar Buyya. *CloudAnalyst : A CloudSim-Based Visual Modeller for Analysing Cloud Computing Environments and Applications.* pages 446–452, 01 2010. 66

[Wilhelm 2008] Reinhard Wilhelm, Jakob Engblom, Andreas Ermedahl, Niklas Holsti, Stephan Thesing, Guillem Bernat, Christian Ferdinand, Reinhold Heckmann, Tulika Mitra, Frank Mueller, Isabelle Puaut, Peter Puschner, Jan Staschulat et Per Stenstrom. *The worst-case execution-time problem - overview of methods and survey of tools.* ACM Trans. Embedded Comput. Syst., vol. 7, 01 2008. 78

[Williamson] E.A.B.S.G. Williamson. Lists, decisions and graphs. S. Gill Williamson. 78

[Wimmer 2014] Florian Wimmer. *Review Samsung Galaxy S5 Smartphone.* `https://www.notebookcheck.net/Review-Samsung-Galaxy-S5-Smartphone.116068.0.html`, 2014. Accessed : 2020-05-3. 102

[Working 2016] OpenFog Consortium Architecture Working. *OpenFog Architecture Overview.* Rapport technique, OpenFog Consortium, 2016. 24, 25

[Xia 2012] C. Xia, D. Liang, H. Wang, M. Luo et W. Lv. *Characterization and modeling in large-scale urban DTNs.* Dans 37th Annual IEEE Conference on Local Computer Networks, pages 352–359, 2012. 125

[Xin 2009] J. Xin, G. Chen et Y. Hai. *A Particle Swarm Optimizer with Multi-stage Linearly-Decreasing Inertia Weight.* Dans 2009 International Joint Conference on Computational Sciences and Optimization, volume 1, pages 505–508, 2009. 93

[Xu 2014a] Donghong Xu et Ke Wang. *Stochastic Modeling and Analysis with Energy Optimization for Wireless Sensor Networks.* International Journal of Distributed Sensor Networks, vol. 2014, pages 1–5, 05 2014. 85

[Xu 2014b] Li Xu, Wu He et Shancang Li. *Internet of Things in Industries : A Survey*. IEEE Transactions on Industrial Informatics, vol. 10, pages 2233–2243, 11 2014. 11

[Yaqoob 2017] Ibrar Yaqoob, Ibrahim Hashem, Yasir Mehmood, Abdullah Gani, Salimah Mokhtar et Sghaier Guizani. *Enabling Communication Technologies for Smart Cities*. IEEE Communications Magazine, vol. 55, 01 2017. xvii, 17

[Yousefpour 2019] Ashkan Yousefpour, Caleb Fung, Tam Nguyen, Krishna Kadiyala, Fatemeh Jalali, Amirreza Niakanlahiji, Jian Kong et Jason P. Jue. *All one needs to know about fog computing and related edge computing paradigms : A complete survey*. Journal of Systems Architecture, vol. 98, pages 289 – 330, 2019. xvii, 29

[Yu 2016] B. Yu, Y. Han, X. Wen et Z. Xu. *SMPA : An Energy-Aware Service Migration Strategy in Cloud Networks*. Dans 2016 IEEE 9th International Conference on Cloud Computing (CLOUD), pages 984–989, June 2016. 137

[Z-Wave 1999] . Z-Wave. *Z-Wave*. `https://www.z-wave.com/`, 1999. Accessed : 2020-05-3. 15

[Zanella 2012] Andrea Zanella, Nicola Bui, Angelo Castellani, Lorenzo Vangelista et Michele Zorzi. *Internet of Things for Smart Cities*. Internet of Things Journal, IEEE, vol. 1, 01 2012. 18

[Zhang 2016] W. Zhang, Y. Hu, Y. Zhang et D. Raychaudhuri. *SEGUE : Quality of Service Aware Edge Cloud Service Migration*. Dans 2016 IEEE International Conference on Cloud Computing Technology and Science (CloudCom), pages 344–351, 2016. 52

[Zigbee 1998] . Zigbee. *intro-to-bluetooth-low-energy*. `https://zigbeealliance.org/`, 1998. Accessed : 2020-05-3. 15