Fog Computing pour l'Internet des objets

Adila Mebrek

Thèse
de doctorat
de l'UTT

Adila MEBREK

Fog Computing pour l'Internet des objets

Champ disciplinaire :
Sciences pour l'Ingénieur

2020TROY0028 **Année 2020**

utt
UNIVERSITÉ DE TECHNOLOGIE
TROYES

THESE

pour l'obtention du grade de

DOCTEUR
de l'UNIVERSITE DE TECHNOLOGIE DE TROYES

EN SCIENCES POUR L'INGENIEUR

Spécialité : SYSTEMES SOCIOTECHNIQUES

présentée et soutenue par

Adila MEBREK

le 17 décembre 2020

Fog Computing pour l'Internet des objets

JURY

M. Hichem SNOUSSI	PROFESSEUR DES UNIVERSITES	Président
M. Samir AKNINE	PROFESSEUR DES UNIVERSITES	Rapporteur
M. Hassine MOUNGLA	MAITRE DE CONFERENCES - HDR	Rapporteur
Mme Valérie RENAULT	MAITRE DE CONFERENCES	Examinatrice
M. Moez ESSEGHIR	MAITRE DE CONFERENCES	Directeur de thèse
Mme Leïla MERGHEM-BOULAHIA	PROFESSEURE DES UNIVERSITES	Directrice de Thèse

Résumé

Le paradigme Fog computing est une approche prometteuse notamment dans le contexte de l'Internet des objets (IoT). Ce dernier est capable de proposer des ressources et fonctionnalités du cloud (tel que calcul, réseau, espace de stockage) à l'extrémité du réseau, plus près des utilisateurs finaux. Cette thèse analyse les performances du paradigme fog computing, dans le contexte des applications IoT sensibles à la latence. La première problématique traitée concerne la modélisation mathématique d'un système IoT-fog-cloud, ainsi que les métriques de performances du système en termes d'énergie consommée et de latence. Cette modélisation nous permettra par la suite de proposer diverses stratégies efficaces de distribution de contenus et d'allocation des ressources fog et cloud aux objets. La deuxième problématique abordée dans cette thèse est liée à la distribution de contenu et données des objets dans des systèmes fog/cloud. Afin d'optimiser simultanément les décisions de d'offloading et d'allocation des ressources, nous distinguons entre deux types d'applications IoT : (1) applications IoT à contenu statique ou avec des mises à jour peu fréquentes ; et (2) applications IoT à contenu dynamique. Pour le premier type d'applications, la génération des données et des requêtes par les objets n'est pas fréquente. Par conséquent, nous proposons une première solution centralisée implémentée dans le cloud, basée sur la théorie des graphes. Pour le passage à large échelle, une deuxième solution implémentée dans les objets et basée sur la théorie des jeux a été présentée. La théorie des jeux peut fournir un cadre mathématique pour la modélisation et l'analyse du problème d'allocation des ressources dans les systèmes fog/cloud. En ce qui concerne le deuxième type d'applications, au vu de la nature dynamique des de flux de données générés par les objets, nous avons développé des stratégies onlines. La première solution est centralisée (implémentée dans le cloud) basée sur les algorithmes génétiques. L'algorithme génétique permet de calculer rapidement et efficacement une solution optimale, notamment dans le cas des applications sensibles à la latence. À l'aide de la théorie des files d'attente, nous avons développé une stratégie de collaboration entre les nœuds fog performante en termes de consommation d'énergie et de latence. Enfin, nous avons proposé une solution basée sur le machine learning implémentée dans les objets. L'algorithme d'apprentissage est réactif, cela nous a permis de réaliser des solutions efficaces en termes d'énergie tout en respectant une latence minimale. Cette thèse montre que le fog peut compléter le cloud afin de générer des économies d'énergie pour les applications IoT sensibles à la latence.

***Index terms*—** Mots clés : Informatique dans les nuages, Informatique mobile, Internet des objets, Traitement réparti, Technologies de l'information et de la communication

Table des matières

Liste des figures

Liste des tableaux

Introduction

Actuellement, l'Internet des objets (IoT) a pour but d'interconnecter tout objet et toute personne via l'Internet. Selon l'estimation de Cisco [1], il y aura plus de 50 milliards de dispositifs IoT (ou "objets") connectés à divers réseaux d'ici 2020. Avec une telle quantité de dispositifs finaux capables de détecter/agir sur l'environnement physique, l'IoT façonne les interactions futures entre l'homme et le monde. Cependant, ces objets génèrent aussi un volume de données important à traiter, ce qui nous incite à exploiter l'ensemble des ressources du réseau pas seulement ceux des objets.

Le cloud computing [2], qui concentre des ressources de traitement et de stockage dans des centres de données, se positionne comme un élément clé des applications IoT (c'est-à-dire des applications liées à des capteurs/actionneurs pour le traitement de données sensorielles ou leur activation dans l'environnement). Basé sur des centres de données riches en ressources, le cloud fournit un pool de ressources *à grande capacité* et peut être utilisé pour combler les limites en ressources des objets. Cependant, étant situés dans le réseau principal, les centres de données sont éloignés des objets, ce qui rend l'utilisation du cloud désavantageuse pour les applications IoT sensibles au délai. En outre, la grande quantité de capteurs générera probablement un volume de données important. Le transfert de toutes les données brutes générées par les capteurs dans le cloud risque d'encombrer le réseau.

Le Fog computing [3, 4] étend les fonctionnalités du cloud computing à l'extrémité du réseau, plus rapproché des capteurs, des actionneurs et des périphériques IoT. Composé de périphériques variés dotés de différentes capacités de ressources, le fog permet de traiter les données localement (c'est-à-dire dans des périphériques plus proches des capteurs et des objets que le cloud), ce qui permet de réduire le temps de réponse des applications. De plus, les données volumineuses récoltées par des capteurs, peuvent être filtrées et agrégées via des analyses locales. De cette manière, seules les données pertinentes doivent être envoyées vers le cloud (pour le stockage et l'extraction ultérieure), ce qui réduit la consommation de bande passante dans le cœur du réseau.

0.1 Motivation

Alors que le trafic de données et la consommation d'énergie sont en plein essor dans le cadre des applications IoT sensibles au délai, le fog computing devient la solution la plus plausible pour faire face aux faiblesses des solutions classiques incluant seulement le cloud. Tirer parti des ressources locales fournies par le fog peut réduire considérablement ce trafic. Cependant, le développement d'une stratégie appropriée de gestion de ces ressources et distribution de contenus sur ces ressources reste un grand défi (c'est-à-dire, comment sélectionner la ressource appropriée pour traiter les requêtes d'un objet IoT). Ces décisions de positionner du contenu ont une incidence sur les performances des applications et des ressources.

Comme dans tout domaine émergent, les questions complexes dans le fog ne manquent pas. Certaines d'entre elles sont issues d'études antérieures sur le pair à pair (P2P), le Mobile ad hoc networks (MANET) et le cloud, alors que d'autres sont motivées par l'influence des récents développements en matière d'ingénierie de réseau et de qualité d'expérience utilisateur.

Une des questions les plus importantes pour le fog est de savoir où, quand et comment distribuer le calcul, la communication, le contrôle et le stockage le long du continuum cloud-to-things.

Plus précisément, il s'agit de répondre à la question *qui fait quoi, à quelle échéance et comment remonter les informations ?* Dans le cas du fog, la question devient :

— quelles tâches doivent s'exécuter dans le fog ? (Par exemple, celles qui requièrent un traitement en temps réel ou celles qui vérifient l'utilisation à moindre coût des ressources qui sont à l'état inactif) ;

— quelles sont les tâches qui vont dans le cloud ? (Par exemple, stockage massif, calcul intensif ou connectivité étendue) ;

— quelles sont les tâches qui vont dans les objets ?

— comment le fog, le cloud et les objets devraient interagir entre eux ?

Le fog devrait permettre le calcul et le stockage et permettre aux tâches d'être réaffectées de manière dynamique entre le fog, le cloud et les objets. Par conséquent, les interfaces permettant au fog d'interagir avec le cloud, les autres fogs, les objets et les utilisateurs doivent : (a) faciliter l'affectation flexible des fonctions de calcul, de stockage et de contrôle entre ces différentes entités ; (b) permettre un accès pratique des utilisateurs aux services du fog ; (c) permettre une gestion efficace du cycle de vie du système et des services.

— *Les interfaces fog-cloud :* Les interfaces du fog au cloud seront nécessaires pour prendre en charge les collaborations entre le cloud et le fog afin de fournir des services de bout en bout. Ils supporteront des fonctions pour, par exemple, permettre que :

1. le fog soit géré par le cloud ;
2. le fog et le cloud échangent des données ;
3. le cloud distribue des services sur le fog ;
4. les services du cloud sont fournis aux objets et aux utilisateurs finaux à travers le fog ;
5. le fog et le cloud collaborent afin de fournir des services de bout en bout.

Il est essentiel de déterminer quelles informations doivent être transmises via l'interface fog-cloud, la fréquence et la granularité de ces informations, et comment le fog et le cloud devraient réagir à ces informations ;

— *Les interfaces fog-fog :* différents nœuds ou systèmes fog peuvent collaborer pour prendre en charge conjointement une application. Par exemple, plusieurs nœuds fog peuvent partager les tâches de stockage de données et de calcul pour un ou plusieurs utilisateurs ou applications. Différents nœuds ou systèmes fog peuvent également collaborer pour offrir un espace de stockage les uns pour les autres. Une question importante consiste donc à savoir comment concevoir une interface et un protocole pour permettre aux nœuds fog de collaborer dans le même système fog ou dans différents systèmes ;

— *Les interfaces fog-objets/utilisateurs :* le fog fournira des services à un large éventail d'utilisateurs finaux et d'objets dotés de capacités très différentes. L'interface fog à objets/utilisateurs sera essentielle pour permettre aux utilisateurs finaux et aux objets d'accéder aux services fog de manière sécurisée et économe en ressources.

0.2 Contribution

Nous résumons nos cinq contributions dans ce manuscrit. Chaque contribution est composée d'une technique mobile spécifique, d'un paradigme de traitement et d'un outil mathématique. Dans la résolution des défis listés auparavant, nous procédons en deux phases :

1. Une modélisation mathématique de notre système IoT-fog-cloud, ainsi qu'une modélisation de nos métriques de performances du système ;
2. Sur la base de la modélisation, nous avons proposé différentes stratégies efficaces d'offloading [1] de requêtes et d'allocation des ressources du fog et du cloud aux objets.

1. pour plus de simplicité nous utiliserons le terme offloading pour décrire le déchargement

Pour surmonter les défis discutés dans la section précédente 0.1 et optimiser simultanément les décisions de déchargement et d'allocation des ressources dans un système fog/cloud, ce travail apporte les contributions suivantes :

1. Nous modélisons les composants d'un système IoT-Fog-Cloud; nous avons aussi proposé deux modèles pour mesurer la consommation d'énergie et la latence dans ce système.
2. Nous définissons deux classes d'applications IoT selon la nature du flux de données généré : (a) applications à contenu statique ou périodique; (b) applications à contenu dynamique. Pour résoudre notre problématique dans le cas des applications à contenu statique, nous avons :
   (a) Formulé une fonction objective, qui cherche à minimiser la consommation d'énergie du système, sous différentes contraintes;
   (b) Développé dans un premier temps une solution centralisée dans le cloud, basée sur la théorie des graphes, qui distribue l'ensemble requêtes de services de chaque utilisateur sur le fog et le cloud.
   (c) Proposé une solution distribuée développé dans les objets et basée sur la théorie des jeux; la solution permet d'affiner la décision de distribution, car chaque utilisateur vise à trouver la meilleure distribution de ces requêtes en tenant compte des décisions des autres utilisateurs qui rivalisent sur les ressources du système. Cette dernière proposition fait face au problème de passage à l'échelle dans la première solution,
3. En ce qui concerne le deuxième type d'applications, au vu de la nature dynamique des flux de données générés par les objets, nous avons proposée des stratégies centralisées (implémentées dans le cloud) à l'aide d'algorithmes génétiques. L'outil nous permet de calculer rapidement et efficacement une solution optimale, notamment dans le cas des applications sensibles à la latence grâce à une adaptation continue dans un environnement dynamique. À l'aide de la théorie des files d'attente, nous avons développé une stratégie de collaboration entre les nœuds fog performante en termes de consommation d'énergie et de latence. Enfin, pour implémenter une stratégie implémentée dans les objets qui partagent le même centre d'intérêt, nous avons proposé une solution basée sur des algorithmes de Machine Learning adaptée pour la résolution des problèmes de bandits à K bras. Ces algorithmes d'apprentissage réactifs nous ont permis de proposer des solutions efficaces en termes d'énergie tout en respectant une latence minimale. Notre travail montre que le fog peut compléter le cloud afin de générer des économies d'énergie pour les applications IoT sensibles à la latence.
4. Une évaluation par simulation de toutes les contributions proposées. L'évaluation compare nos approches à un ensemble d'algorithmes de déchargements et d'allocation des ressources, et analyse : (a) l'évolutivité en termes de passage à l'échelle; (b) qualité des résultats en termes de temps de réponse moyen, énergie consommée moyenne, coût de traitement des requêtes de services, etc.

Ces contributions ont été validées par trois publications [5, 6, 7]

0.3 Organisation du manuscrit

La suite de ce manuscrit est organisée comme suit. Le chapitre 1 détaille le contexte de ce travail. Le chapitre 2 décrit en détail l'état des différentes solutions d'allocation de ressources et de distribution de services proposés pour le fog computing. Le chapitre 3 fournit une modélisation du système fog/cloud, du délai de service et de la consommation énergétique du système. Dans le chapitre 4, nous considérons les applications IoT avec contenu statique (ou avec des mises à jour peu fréquentes); nous présentons deux solutions : (1) une première solution centralisée basée sur la théorie des graphes; (2) une deuxième solution distribuée basée sur la théorie des jeux. Ces deux solutions cherchent à traiter le problème de la distribution des requêtes (statiques/hors-ligne) générées par les objets sur les nœuds fog et le cloud, dans le contexte des applications IoT.

Le chapitre 5 introduit trois approches pour traiter le problème de l'allocation de ressources et la distribution des requêtes sur le fog et le cloud, dans le contexte des applications IoT à contenu dynamique (qui nécessitent un traitement en temps réel). Notre première approche est implémentée dans le cloud, tandis que les deux autres sont implémentées respectivement dans les nœuds fog et les objets (utilisateurs finaux).

Chapitre 1

Introduction à l'Internet des Objets et au Fog Computing

1.1 Introduction

Les infrastructures de l'informatique en brouillard (*Fog computing*) constituent un élément majeur de l'Internet. Grâce leur capacité de stockage et de calcul, ainsi que leur emplacement près des utilisateurs finaux, elles permettent de traiter les données à proximité de l'endroit où elles sont générées. Dans ce chapitre, nous allons introduire des définitions et des explications de certaines notions utilisées dans la présente thèse. Bien que le domaine de l'Internet des Objets, ou Internet of Things (IoT), soit largement adopté et déjà bien connu, nous commencerons ce chapitre par expliciter l'IoT, avant d'engager une courte discussion des quelques applications de cette technologie. Par la suite, nous détaillerons les nombreuses évolutions qui ont été nécessaires afin d'aboutir à l'infrastructure d'informatique en brouillard, ou Fog computing, telle que nous la connaissons aujourd'hui.

1.2 Internet des Objets

1.2.1 Vue d'ensemble

L'IoT a créé un paradigme qui intègre l'environnement physique nous entourant est intégré, grâce à des capteurs ou actionneurs connectés à Internet [1]. Les capteurs convertissent les paramètres physiques en données numériques, à partir desquelles nous pouvons tirer des informations ou des connaissances relatives à l'environnement physique. Les actionneurs transforment donc les décisions logiques en actions physiques, qui peuvent affecter le monde réel.

L'écosystème IoT contient à la fois des dispositifs matériels et des applications logicielles. Les applications IoT analysent les données sensorielles recueillies à partir de capteurs et réalisent des actions au moyen d'actionneurs. Une application IoT peut avoir plusieurs capteurs et actionneurs qui coopèrent afin d'obtenir des informations sur l'environnement et réagir en fonction des informations collectées [2]. L'IoT représente un réseau mondial de dispositifs interconnectés, plus connus sous le nom d'objets intelligents. La coopération des objets entre eux et entre services permet la création de nombreuses applications IoT innovantes [3].

1.2.2 Champs d'applications

Les applications IoT sont déployées dans divers secteurs, tels que la santé, la gestion des stocks et de la production, les chaînes d'approvisionnements alimentaires, le transport, la sécurité et la surveillance [4, 8, 9, 10]. À titre d'exemple, les objets intelligents peuvent être utilisés par des patients qui ont besoin de collecter des données concernant leur état de santé, comme leurs battements de cœur, leur pression artérielle et leur taux de glucose. Dans ce cas, il possible de suivre

l'état de santé des patients grâce à ces objets intelligents. Par ailleurs, l'IoT est aussi employé pour améliorer les maisons intelligentes : lorsque des capteurs détectent des variations de température, les systèmes de climatisation peuvent être contrôlés pour rétablir la température idéale. Les caméras de sécurité domestique sont capables de capturer n'importe quel intrus et de transmettre des alertes aux propriétaires de la maison via des applications mobiles. De plus, l'IoT peut surveiller les systèmes de transport pour créer des villes intelligentes. Une fois collectées et analysées, les données permettent de comprendre les mises à jour des réseaux de trafic et des systèmes de transport. Enfin, une chaîne logistique fonctionnant avec l'IoT donne la possibilité d'enregistrer et suivre en temps réel toutes les livraisons. Dès que les marchandises ont été expédiées vers leur destination, les enregistrements de livraison peuvent être mis à jour. Il est possible de cartographier ces cas d'utilisation vers un modèle générique, lequel doit permettre une intégration plus aisée des différents services, allant des capteurs à l'infrastructure de réseau, des interfaces de programmation d'application ou Application Programming Interface (API) au traitement de données volumineuses, et enfin de l'analyse à la modélisation prédicative.

1.3 Fog computing

Aujourd'hui, des milliards d'objets intelligents sont connectés à Internet. Cisco a estimé qu'en 2020, ils représenteraient plus de 50 milliards [11]. Bien que nous sachions aujourd'hui que cette estimation est surévaluée, le nombre d'objets connectés n'a pas cessé de croître. Ces objets ont tous des caractéristiques très diverses. Par exemple, les applications de contrôle de vols de drones et les applications de jeux et de réalité virtuelle peuvent nécessiter un traitement rapide de leurs données. Étant donné que presque toutes les données sensorielles sont générées par des dispositifs finaux, ces dernières doivent être transférées (et analysées) vers des appareils possédant d'avantage de ressources. Pour pouvoir répondre à une telle exigence, deux paradigmes d'infrastructure – le cloud computing et fog computing – ont été positionnés comme des catalyseurs clés des applications IoT, et sont présentés ci-après.

1.3.1 Du nuage au brouillard

Le cloud computing fournit un portail central d'informations. De nos jours, il est largement utilisé comme solution destinée à de nombreuses technologies comme l'IoT, car il offre plusieurs fonctionnalités. Nous pouvons notamment citer l'évolutivité, l'allocation de ressources à la demande, la réduction des efforts de gestion, les modèles de tarification flexibles (pay as-you go) et l'approvisionnement d'applications et de services simples. Les services les plus courants fournis par le cloud sont les suivants : logiciel en tant que service (SaaS), plateforme en tant que service (PaaS) et infrastructure en tant que service (IaaS), qui se dirigent tous vers XaaS (Anything as a Service). Cependant, il existe des limitations inhérentes au cloud computing. L'un des problèmes clés tient à la distance de propagation entre les utilisateurs finaux et le cloud, ce qui se traduit par une latence importante. Une petite latence de service peut être essentielle dans certains cas d'applications IoT sensibles au délai. En outre, parce que les données générées par des milliards d'objets IoT et de capteurs sont toutes transférées et traitées dans le cloud, les délais de traitement peuvent être considérablement réduits. C'est pourquoi le cloud computing entraîne également des coûts opérationnels et une consommation d'énergie élevée.

Au cours des dernières années, la mise en place du cloud décentralisé a permis de combler les lacunes de ce dernier en matière de prise en charge de l'IoT. Cela nécessite une nouvelle architecture, qui répartit les fonctions de calcul, de contrôle, de stockage et de mise en réseau à l'extrémité du réseau, plus proche des utilisateurs finaux. Le fog computing [12] est un paradigme informatique introduit pour relever les défis du cloud. En complétant le cloud, le fog comble le fossé entre le cloud et les objets pour permettre une meilleure utilisation des services IoT. D'une part, le fog communique directement avec les objets par le biais de connexions sur un ou plusieurs sauts, au moyen d'interfaces sans fil standard (Wi-Fi, Bluetooth, etc.). Grâce aux fonctions de calcul embar-

quées et aux contenus mis en cache, le fog peut fournir des applications prédéfinies aux utilisateurs finaux, sans l'aide du cloud ou d'Internet. D'autre part, le fog peut être connecté au cloud afin d'exploiter les riches fonctions et outils dans celui-ci. En raison de la distance, la latence de communication entre les objets et le cloud est toujours élevée par rapport à celle des objets et du fog. En comparaison, la consommation d'énergie et les coûts d'exploitation dans le fog computing sont également plus faibles.

Traditionnellement, le cloud est déployé avec une infrastructure importante, centralisée et coûteuse, telle que des passerelles de services, des passerelles de réseau, etc. La cinquième génération des réseaux cellulaires (5G) constitue une technologie importante pour les paradigmes du cloud distribués comme le fog. Dans les systèmes sans fil 5G, des dispositifs périphériques ultra denses, y compris des stations de base à petites cellules ou Base Station (BS), des points d'accès sans fil ou Access Point (AP), des ordinateurs portables, des tablettes et des smartphones seront déployés, chacun possédant une capacité de calcul équivalente à celle d'un ordinateur [13]. Le réseau cellulaire 5G à large spectre, ainsi que les immenses ressources de calcul et de stockage disponibles aux extrémités du réseau, joueront un rôle important dans la prise en charge efficace du fog computing.

1.3.2 Définition du fog computing

Introduit par Cisco en 2012 [14], le fog computing a ensuite été développé et défini par Open fog Consortium (OFC). Le fog computing relève d'une définition plus large du Edge computing, qui consiste à pousser des applications et des services, totalement ou partiellement, vers l'extrémité du réseau. En conséquence, certaines fonctions (traitement, stockage temporaire, agrégation de données) sont transférées à proximité des utilisateurs finaux et des objets connectés, ce qui améliore le temps de réponse des applications et des services [15]. Comme mentionné précédemment, bien que les paradigmes fog computing et cloud computing détiennent un fondement commun, à savoir la virtualisation, leurs caractéristiques sont différentes. Récemment, des propositions similaires au fog peuvent être trouvées sous différents noms, comme *Edge computing* [16], *Extreme Edge computing* [17] ou *Mobile Edge computing* [18].

Cependant, le terme fog computing ne désigne pas une seule architecture, mais une approche plus générique consistant à rapprocher l'intelligence plus près des utilisateurs finaux. Les infrastructures fog peuvent être hiérarchiques, et comporter une architecture de cloud en haut de cette hiérarchie [19]. L'idée est que plus l'emplacement des équipements du fog se trouve haut dans l'architecture (plus près du cloud), plus les ressources que le fog met à disposition des utilisateurs sont importantes. En contrepartie, la latence de propagation est plus élevée. Le cloud fournit une capacité de stockage et de calcul quasi illimitée, au prix d'une latence élevée. À l'inverse, le fog, plus proche de l'utilisateur, fournit une très faible latence, une faible capacité de calcul et de stockage. De la même manière, Perrera et al. [20] ont essayé de clarifier et d'élargir le concept du fog computing.

« Le fog computing est un scénario dans lequel un nombre considérable de dispositifs omniprésents et décentralisés hétérogènes (sans fil et parfois autonomes) communiquent et coopèrent potentiellement entre eux et avec le réseau pour effectuer des tâches de stockage et de traitement sans l'intervention de tiers. Ces tâches peuvent être destinées à prendre en charge des fonctions réseau de base ou de nouveaux services et applications s'exécutant dans un environnement en bac à sable. Les utilisateurs qui louent une partie de leurs appareils pour héberger ces services sont incités à le faire.»

La figure 1.1 illustre l'intégration du fog computing entre le cloud et les périphériques finaux. Au lieu d'envoyer toutes les données collectées au cloud, le fog traite (au moins partiellement) les données localement –au plus près du lieu dans lequel elles ont été générées. Le prétraitement des données en marge du réseau permet de pallier à certaines faiblesses du cloud, de réduire les délais et les besoins en bande passante, de rendre le calcul plus sensible au contexte.

Bien qu'il n'existe pas de définition stricte, les nœuds fog sont généralement considérés comme des dispositifs dotés de capacités (limitées) pour traiter ou stocker des données et pour les router dans le réseau. Une classification des nœuds fog adoptée par Mahmud et al. [21], inclut

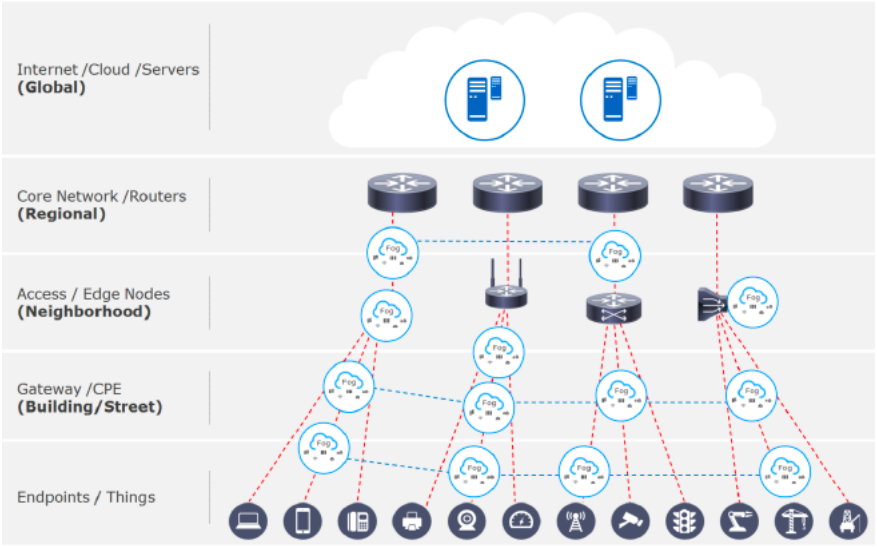


FIGURE 1.1 – Architecture conceptuelle du fog computing [33].

des nœuds capteurs bas de gamme, des périphériques tels que les cartes Raspberry et Arduino et des systèmes plus sophistiqués et plus puissants (quoique toujours limités) d'appareils comme les smartphones et les tablettes. Tous ces appareils (réfrigérateur intelligent, montre intelligente, smartphone, etc.) peuvent devenir un nœud et effectuer des traitements.

1.3.3 Différence entre le Cloud computing et le Fog computing

Les architectures du fog computing sont basées sur des nœuds ou clusters fog, dans lesquels plusieurs dispositifs du fog interviennent pour coopérer et effectuer un traitement. Les centres de données ou Data center (DC) sont les principaux composants physiques des clouds. C'est la raison pour laquelle le cloud computing engendre des coûts opérationnels et une consommation d'énergie élevée. Le fog est situé plus près de l'utilisateur, de sorte que la distance entre les utilisateurs et le fog soit d'un ou de plusieurs sauts [22]. En raison de la distance, la latence de communication pour le cloud est toujours élevée par rapport à celle du fog. En outre, le cloud constitue une approche plus centralisée, tandis que celle du fog est davantage distribuée et basée sur la distribution géographique[23].

Notons que l'interaction en temps réel n'est pas possible dans le cloud, en raison de sa latence élevée. Ce problème peut toutefois être aisément résolu par le fog computing.

Cependant, le taux de défaillance dans le fog est élevé en raison de la connectivité sans fil et de la gestion décentralisée [24, 22, 23, 25, 26]. En effet, la plupart des appareils dans les environnements fog auront une connexion sans-fil, car les principaux participants au système fog sont les objets intelligents [27]. Ces périphériques peuvent s'arrêter si le logiciel n'est pas géré correctement. De plus, il arrive parfois que le traitement dans le fog échoue dans d'autres cas. À titre d'exemple, si le dispositif du fog est pleinement utilisé par l'application du dispositif lui-même, il ne pourra effectuer aucun autre traitement. Par conséquent, la planification des applications et des ressources dans le fog est plus complexe. Dans le tableau 1.1, nous présentons les caractéristiques les plus importantes du fog et du cloud.

Il serait donc incorrect de dire que le fog puisse remplacer le cloud. Nous ne pouvons même pas conclure que le premier est meilleur que le second. En réalité, les deux contribuent différemment au développement de l'IoT en répondant à des perspectives et à des exigences distinctes.

1.3.4 Paradigmes et technologies connexes

De nombreuses architectures sont proposées pour pallier le manque de ressources sur les objets connectés, tout en permettant un passage à l'échelle et des calculs à faible latence. Dans ce

	Cloud	Fog
Taille	Des centres de données étendus avec un grand espace de stockage et des ressources de calculs.	Espace de stockage et des ressources de calculs limités. Un certain nombre de petits nœuds fog peuvent être utilisés par un système fog.
Déploiement centralisé	Nécessite un déploiement soigneux et une maintenance maximale.	Déployé en centralisé ou distribué par des entreprises locales avec une maintenance minimale.
Emplacement	Informations globales recueillies dans le monde entier. Prend généralement en charge toutes les applications qui ne nécessitent pas un traitement en temps réel ou à faible latence (tolérantes au délai).	Informations liées à des emplacements de déploiement spécifiques. Peut prendre en charge des applications plus sensibles au délai.
Forte mobilité	Afin de prendre en charge la mobilité, le chemin d'un service doit changer à chaque fois, car le service est fourni à partir d'un cloud principal.	Un service peut être implémenté sur plusieurs nœuds fog en suivant les dispositifs mobiles cibles.

TABLEAU 1.1 – Comparaison entre le Fog computing et le Cloud computing

qui suit, nous allons définir quelques paradigmes proches du fog computing.

Edge computing : Similaire au fog computing, il est l'une des solutions apportées aux problèmes décrits précédemment. L'idée générale est de réaliser les calculs sur des serveurs situés en périphérie du réseau, au plus près des utilisateurs et des objets connectés. Dans ce but, des serveurs sont déployés à proximité des utilisateurs, en périphérie du réseau [28, 29]. Dans le cas d'un réseau cellulaire, les serveurs sont déployés près de chaque BS. Les utilisateurs alors ont recours aux ressources fournies par les serveurs situés dans la cellule à laquelle ils sont connectés. Ces serveurs effectuent les calculs qui nécessitent un faible temps de réponse, et transmettent au cloud ceux qui exigent des ressources plus conséquentes. Une telle architecture est également utile lors du stockage de données : l'utilisateur interroge un serveur situé à la périphérie du réseau. Si ce dernier ne possède pas la ressource demandée, la requête est transmise à l'infrastructure cloud. Lorsque les données sont retournées par le cloud, elles sont mises en cache afin de pouvoir répondre directement aux futures requêtes. Si les serveurs situés à la périphérie du réseau ont des ressources moins importantes que celles du cloud, elles restent suffisantes pour effectuer certains calculs ayant besoin d'une faible latence.

Extreme Edge computing : Son objectif est le même que celui du Edge computing, c'est-à-dire traiter les requêtes au plus près des utilisateurs afin de limiter les sollicitations de l'infrastructure cloud. Toutefois, la mise en œuvre est différente : au lieu de placer des nouveaux serveurs près des utilisateurs, ce sont les utilisateurs qui mettent en commun et partagent leurs ressources à l'aide de protocoles P2P ou ad-hoc [30]. Les calculs et le stockage sont donc effectués sur les périphériques (objets connectés) eux-mêmes. Les différences entre ces modèles d'infrastructure sont résumées par [31].

Mobile Cloud computing : Ces infrastructures permettent à des clients mobiles et ayant de faibles ressources de pouvoir bénéficier des ressources des infrastructures cloud. Deux architec-

tures principales offrent cette possibilité [32]. La première consiste à déployer des serveurs (appelés *Cloudlets*) près des utilisateurs situés à l'extrémité du réseau. Cette approche est équivalente à celle du Edge computing ou encore au Mobile Edge computing [33]. Quant à la seconde architecture, elle vise à mettre en commun les ressources des différents périphériques situés à proximité, afin d'avoir une puissance de calcul suffisante pour les applications. Cette dernière approche est également connue sous le nom d'*Extreme Edge computing*. Des modèles hybrides combinant ces deux approches ont également été proposés [34].

Mobile Edge computing : Mobile Cloud computing (MCC) introduit un délai de traitement important lors du déchargement des calculs, consistant en un téléchargement de données relatives aux calculs (codes de programmation et données d'entrée, par exemple) vers le centre de cloud, afin de récupérer le résultat du calcul et l'exécution du code dans le centre de cloud. Pour faire face au problème de délai introduit par MCC, un autre paradigme, Mobile Edge Computing (MEC), est proposé. L'idée principale de MEC est d'amener les ressources de calcul et de stockage à l'extrémité du réseau mobile tout en respectant des constantes de délai strictes avec une courte distance de transmission de données. Rapprocher le calcul des utilisateurs mobiles est le concept clé de MEC. Ce dernier peut, par exemple, améliorer les petites cellules, les microcellules, les picocellules ou les femtocellules, avec une capacité de calcul et de stockage plus grandes [33]. La BS de contrôle permet de à mettre en œuvre une gestion dynamique et élastique des petites cellules et à connaître les informations relatives aux utilisateurs mobiles (par exemple, localisation), les informations relatives aux edges cloud (par exemple, ressources de calcul et de stockage), les informations relatives aux utilisateurs et les BS petites cellules (par exemple, état du réseau et interférences de signal).

1.3.5 Exemples d'applications du fog computing

De nombreux cas d'utilisations nécessitant le recours à une architecture fog computing ont été proposés.

Des feux de signalisation intelligents dans la ville intelligente : Plusieurs travaux [35] ont envisagé d'utiliser une infrastructure de fog computing pour réguler le trafic routier. L'idée émise est qu'un feu de signalisation transmette son état aux voitures proches afin que ces dernières ralentissent. L'état des feux et la position des véhicules sont régulièrement envoyés vers une plateforme de fog qui établit une politique globale afin que les usagers attendent le moins longtemps possible et connaissent la durée de trajet la plus courte. De façon plus générale, le fog a été proposé pour aider à la prise de décision dans un réseau de véhicules (VANET) [36].

L'utilisation du fog a été imaginée dans un contexte de *villes intelligentes* [37]. L'idée générale est d'utiliser des nœuds de fog comme une plateforme de traitement pour les données collectées par des capteurs au sein de la ville.

Traitement vidéo : Pour Satyanarayanan et al. [38], le fog pourrait être utilisé pour stocker des vidéos enregistrées par des caméras situées en marge du réseau. En effet, les flux vidéos sont conséquents et le cloud ne pourrait pas supporter un grand nombre de caméras. Toutefois, leur proposition ne se limite pas à cette idée ; la hiérarchie du fog permettrait de réaliser des traitements sur les flux vidéos, et notamment de créer une base de données centralisée dans le cloud qui permette une mise en cache des vidéos stockées dans le fog. Un autre cas d'utilisation des infrastructures de fog computing est la reconnaissance faciale [39]. Les passants sont filmés par une caméra et les images sont transmises à un nœud fog afin de reconnaître en temps réel les personnes.

1.4 Conclusion

Le fog computing a été introduit pour répondre à un besoin de traitement avec une faible latence que les infrastructures de cloud computing ne sont pas capables de fournir, mais également pour une question de passage à l'échelle. De fait, le cloud ne pouvait faire face à l'explosion du nombre d'utilisateurs. Il est également intéressant de noter que les approches de fog computing ou d'edge computing sont très semblables et ne se distinguent que sur des aspects mineurs. Autrement dit, le fog considère des ressources organisées de façon continue entre le cloud et les utilisateurs alors que l'edge computing ne considère qu'un niveau intermédiaire. En pratique, la plupart des solutions logicielles déployées pour de l'edge computing fonctionnent dans une infrastructure fog et inversement. Aujourd'hui, les infrastructures de fog sont introduites dans le réseau 5G afin de fournir une puissance de calculs aux téléphones portables [23, 27].

Dans le chapitre 2, nous allons détailler les différentes solutions d'allocation de ressources et de distribution de services proposées dans le fog computing.

Chapitre 2

Allocation de ressources et offloading de tâches dans le Fog Computing

2.1 Introduction

Après avoir présenté notre contexte de recherche, nous nous intéressons au fonctionnement de différentes solutions récentes et pertinentes d'allocation de ressources et de distribution de services qui ont été proposées pour le fog computing. Dans un premier temps, nous détaillerons les algorithmes pour le traitement dans le fog. Dans un second temps, nous présenterons les algorithmes pour le stockage et la distribution du contenu. D'autre part, nous étudierons les travaux qui ont couvert l'aspect réseau de communication dans les systèmes fog. Enfin, nous analyserons et évaluerons la consommation d'énergie dans l'ensemble du système. La section 2.2 décrira plusieurs critères auxquels il faut prêter attention pour évaluer les solutions proposées, tandis que la section 2.3 se concentrera sur des travaux connexes.

2.2 Description des critères d'évaluation

Pour pouvoir étudier les approches abordant les problèmes listés précédemment dans le contexte du fog computing et de l'IoT, il convient de prendre en compte plusieurs critères que nous allons détailler.

— **Qualité de service (QoS) :** le premier critère *(C1)* est la nécessité de respecter la QoS requise pour chaque application déployée dans le système fog. Bien que la QoS admette plusieurs définitions, dans notre cas d'étude la QoS repose sur le délai réseau, le débit et la consommation d'énergie. Un système fog est perçu comme un catalyseur prometteur pour les applications IoT sensibles à la latence, en raison de la proximité des nœuds fog avec les objets IoT. Cette proximité réduit la latence. Cependant, cette dernière varie considérablement, par exemple en fonction de la stratégie de gestion des ressources, du modèle de consommation d'énergie et de la politique de planification [40]. Ainsi, le critère de la QoS est donc nécessaire.

— **Évolutivité :** la dépendance vis-à-vis du cloud computing pour le traitement des applications IoT a été relevé dans de nombreux travaux de recherche impliquant des milliards d'appareils IoT [41]. Toutefois, la mise en œuvre d'une application dans le cloud, au sein d'un environnement où les objets IoT génèrent une quantité considérable de données n'est ni réalisable ni efficace. En conséquence, les systèmes fog doivent être opérationnels à larges échelles. Ils devraient pouvoir diminuer et augmenter d'une manière flexible. Des solutions sont nécessaires pour assurer cette évolutivité.

— **Fédération :** le troisième critère *(C3)* est le besoin de fédération. La couche du fog est répartie géographiquement sur une large échelle, chaque domaine du fog peut appartenir à un fournisseur différent. En outre, la couche cloud peut être exploitée par un autre fournisseur.

Par conséquent, le déploiement des applications nécessite la fédération de ces multiples fournisseurs, lesquels peuvent héberger les différents composants d'une application. Assurer la bonne coordination des interactions nécessaires entre les composants de l'application implique que ces fournisseurs coopèrent.

— **Hétérogénéité :** le quatrième critère *(C4)* est le besoin d'hétérogénéité en matière de ressources. L'hétérogénéité des périphériques ne se réfère pas seulement à la diversité des services et des protocoles, mais également à la variété des niveaux horizontaux et verticaux de l'architecture Fog. Les nœuds dans la couche fog et les nœuds dans la couche cloud sont très hétérogènes concernant leurs capacités de calcul et de stockage. Les nœuds fog ont probablement des capacités limitées par rapport au cloud. Il peut également y avoir des différences significatives entre les nœuds de différents domaines du fog et même entre les nœuds du même domaine. Pour utiliser les capacités de différents types de périphériques dans un environnement IoT, il est évident que l'application doit être conçue de manière à pouvoir exécuter son travail sur plusieurs périphériques, quels que soient sa capacité et son emplacement. Plus précisément, l'application doit pouvoir utiliser le maximum de ressources de calcul disponibles.

— **Mobilité :** le cinquième critère *(C5)* est la nécessité de supporter la mobilité. Les objets IoT/utilisateurs finaux et les nœuds fog peuvent être mobiles. En conséquence, le système devrait pouvoir gérer cette mobilité. La mobilité ne concerne pas seulement les objets IoT, mais également les périphériques de calcul et de stockage dans la couche fog. En conséquence, la gestion des objets mobiles sur deux plans différents (objets et nœuds fog), ainsi que la synchronisation de l'un avec l'autre, constituent un défi. Pour garantir la disponibilité des ressources et la réussite des tâches, la distribution, la duplication et la migration des tâches sont nécessaires. Ce mécanisme est déjà envisagé dans le cloud, mais il est important de le réexaminer en tenant compte de la nature distribuée du fog ainsi que de ces ressources limitées.

Après avoir détaillé notre problématique de recherche principale et nos critères d'études, nous allons présenter dans la section suivante les travaux récents et pertinents portant sur l'intégration du fog computing dans le contexte des applications IoT.

2.3 Problèmes d'allocation de services et de gestion des ressources dans le Fog Computing

Dans le paradigme de fog computing, la question de savoir quelle tâche ou requête doit être traitée dans la couche fog, ou doit être transmise au cloud, a gagné en importance. L'allocation de services est cruciale pour prendre en charge les applications IoT sensibles au délai. Simultanément, pour gérer les requêtes et tâches d'une grande quantité d'objets répartis géographiquement, les ressources de calcul et de stockage de la couche fog doivent être utilisées efficacement. Dans ce qui suit, nous allons aborder plusieurs problèmes d'allocation de services et de gestion des ressources.

2.3.1 Algorithmes pour le traitement de tâches

Dans ce qui suit, nous introduisons d'abord les algorithmes de partage de ressources de calcul. Nous présentons ensuite les algorithmes de planification de tâches et détaillons, après cela, des algorithmes de redistribution de charges et d'offloading. Selon les critères présentés à la section 2.2, les travaux qui présentés dans cette section seront comparés et résumés dans le tableau 2.1.

Partage de ressources : Dans les systèmes fog, l'un des premiers aspects qui a été étudié pour remédier au problème de traitement des tâches générées par les objets, est le partage des ressources de calcul et la coopération entre les nœuds fog. Ces aspects ont été introduits dans la couche fog,

avec pour objectif d'exécuter des requêtes et des tâches de calcul dans la couche fog au lieu du cloud. Certains des travaux [42, 43, 44] couvrent ces aspects.

Al-khafajiy et al. [42] ont introduit un algorithme permettant le partage des ressources de calcul entre les nœuds fog, en se focalisant particulièrement sur les petites cellules activées par le fog (fog-enabled small cells) dans les réseaux cellulaires. Les auteurs visent à former des clusters de petites cellules : chacun représente un groupe de petites cellules partageant des ressources pour que les appareils mobiles puissent libérer leur charge de travail. Leur objectif est de le réaliser avec une consommation minimale en énergie. À cette fin, ils ont modélisé leur fonction objective en tant que problème d'optimisation. Les auteurs visent à former simultanément des clusters et à y allouer des ressources de calcul et de communication tout en respectant les contraintes de latence de chaque utilisateur. En comparant leur solution avec d'autres stratégies de clustering, les auteurs montrent que leur méthode peut satisfaire un pourcentage plus élevé de demandes des utilisateurs. Dans ce travail, les auteurs prennent en compte les limites des ressources de calcul et de communication. Ils considèrent également la contrainte de latence perçue par les utilisateurs ainsi que l'énergie consommée. Ils répondent donc aux critères de QoS (C1) et d'hétérogénéité (C4). Cependant, les critères d'évolutivité élastique (C2) et de prise en charge de la mobilité (C5) ne sont pas satisfaits ; les auteurs ont évalué leur stratégie sur un scénario à petite échelle et n'ont pas tenu compte de la mobilité de l'utilisateur. De même, le critère de fédération (C3) n'est pas rempli, puisque les auteurs n'ont pas envisagé la possibilité d'impliquer plusieurs fournisseurs.

Yousefpour et al. [43] ont également abordé également le problème du partage des ressources de calcul entre les nœuds fog pour exécuter les demandes de calcul au sein du même domaine dans la couche fog, en vue de permettre l'exécution des demandes de calcul des utilisateurs. Ils ont défini une métrique d'utilité pour quelques nœuds, en prenant en compte le coût de communication, ainsi que le gain en tarification dans le cas où ils partagent leurs ressources. À l'aide de ces métriques, l'algorithme détermine d'abord une liste ordonnée de nœuds fog de préférence pour chaque demande de calcul. Ensuite, chaque nœud du domaine envoie des demandes de placement de pari à ses nœuds préférés. À la réception d'une offre et en fonction des demandes précédemment reçues, un nœud cible décide d'accepter ou de rejeter la demande. Cette opération conduit à un appariement un à un. Par conséquent, en fonction des capacités d'un nœud, des appariements supplémentaires sont envisagés dans une étape ultérieure, en suivant la liste ordonnée de demandes rejetées et non placées. L'évaluation de cette stratégie montre qu'elle surpasse l'approche gloutonne en matière d'utilité totale. Dans ce travail, les limitations de ressources du système fog sont modélisées grâce à cet algorithme : de cette façon, des ressources hétérogènes sont prises en charge. En outre, le critère de qualité de service (C1) est satisfait, bien que seule l'énergie consommée soit prise en compte dans les décisions de partage de ressources. De plus, l'évaluation est menée à petite échelle et ne tient pas compte de la mobilité des appareils ou des nœuds fog. En conséquence, les critères d'évolutivité élastique (C2) et de prise en charge de la mobilité (C5) ne sont pas satisfaits. Le critère de fédération (C3) n'est pas pertinent pour ce travail, car l'algorithme proposé ne fonctionne que dans un seul domaine fog. Enfin, le critère d'hétérogénéité (C4) est rempli.

Xiao et al. [44] abordent le même problème, mais considèrent le cas d'un système de fog mobile, qui comprend une couche fog formée par des appareils mobiles et un cloud accessible via le réseau cellulaire. Ils visent ainsi à optimiser l'utilisation du processeur (CPU), la bande passante et le partage de stockage pour répondre aux demandes de calcul. Ils étudient séparément l'optimisation avec les deux objectifs suivants :maximiser à la fois la somme et maximiser le produit des fonctions d'utilités. Ils résolvent les deux problèmes (somme et produit) en utilisant une optimisation convexe. L'évaluation de leur stratégie sur un ensemble de trois nœuds avec des mesures tirées du monde réel montre que cela permet de réduire le temps de latence des services et conduit à une efficacité énergétique élevée. Dans ce travail, les auteurs répondent aux critères de QoS (C1) et d'hétérogénéité (C4) en les représentant dans leur modèle. Néanmoins, les simulations ne sont menées que sur une petite échelle et la mobilité n'est pas couverte ; dans ces conditions, les critères d'évolutivité élastique (C2) et de soutien à la mobilité (C5) ne sont pas satisfaits. Enfin, le critère

de fédération (C3) n'est pas rempli, puisque les auteurs ne tiennent pas compte de l'existence de divers fournisseurs.

Planification des tâches : Puisque les systèmes fog offrent des capacités de calcul en marge du réseau, des questions importantes se posent notamment : comment gérer l'exécution des tâches? Plus précisément, comment décider quelle tâche exécuter dans la couche IoT, la couche fog et la couche cloud? Plus simplement, à quels nœuds une tâche particulière devrait être assignée? Quelles mesures prendre en compte dans les décisions dérivées? Plusieurs travaux de recherche ont abordé ces questions, en considérant uniquement la couche fog, comme c'est le cas dans [45, 46, 47]. D'autres travaux ont considéré simultanément la couche IoT et la couche fog : nous pouvons citer [48]. Enfin, certains travaux ont considéré les couches IoT, fog et cloud simultanément, par exemple dans [49, 50].

En commençant par des travaux axés sur la couche fog, Ni et al. [45] étudient le problème de planification des tâches dans la couche fog basée sur un réseau cellulaire, dans laquelle de petites cellules sont activées avec des capacités de calcul et forment les nœuds fog. Ils proposent une stratégie d'ordonnancement des tâches qui fonctionne selon deux étapes principales. La première consiste à allouer des ressources de calcul au niveau de chaque petite cellule sur une liste ordonnée d'utilisateurs qui lui sont associés et sur la base d'un objectif spécifique. À la fin de cette étape, il se peut que certaines demandes n'aient pas été traitées en raison du manque de ressources disponibles. En conséquence, dans la deuxième étape, des groupes de calcul sont construits pour leur traitement. Toujours dans cette étape, les demandes sont servies en fonction d'un certain ordre et en fonction d'un objectif particulier. Les auteurs testent trois variantes de leur algorithme avec diverses métriques de classement et objectifs de regroupement. Leurs résultats montrent que toutes les stratégies surpassent d'autres classifications telles que celles en matière de taux de satisfaction de l'utilisateur. De plus, les variantes basées sur la latence permettent d'obtenir une latence inférieure à celle des autres, ainsi que la variante centrée sur une consommation énergétique induit une consommation par utilisateur réduite. Globalement, dans ce travail, les auteurs représentent l'hétérogénéité des ressources dans leur algorithme et étudient la QoS du côté de l'utilisateur. Ils répondent donc aux critères de qualité de service (C1) et d'hétérogénéité (C4). Cependant, leur évaluation est réalisée dans un scénario à petite échelle et la mobilité de l'utilisateur n'est pas étudiée. En conséquence, le critère d'évolutivité élastique (C2) et le critère de prise en charge de la mobilité (C5) ne sont pas satisfaits. C'est encore le cas du critère de fédération (C3), car les auteurs ne tiennent pas compte de la possibilité d'avoir plusieurs fournisseurs.

Datta et al. [46] traitent également du problème de planification des tâches dans la couche fog. Cependant, contrairement à Ni et al. [45], les auteurs considèrent que chaque nœud fog représente un serveur de calcul. Dans ce contexte, ils visent à trouver une correspondance entre les tâches et les serveurs, de sorte à minimiser la probabilité de blocage des tâches, tout en respectant la contrainte de latence de l'utilisateur. Pour résoudre ce problème, les auteurs ont proposé trois politiques différentes. La première adopte une approche aléatoire : un nœud fog est sélectionné de manière aléatoire pour exécuter une tâche générée. La seconde est une stratégie de latence minimale qui sélectionne le nœud fog, impliquant une latence minimale pour exécuter une nouvelle tâche, en fonction de l'état du système. Enfin, la troisième stratégie cible la capacité, les attributs d'une tâche entrante et le nœud fog avec le maximum de ressources restantes dans une liste de candidats. Ces politiques sont comparées dans un environnement de simulation. Les résultats montrent que la probabilité de blocage est la plus faible dans le cas de la stratégie de latence la plus basse. Dans ce travail, les auteurs ne tiennent pas compte des différences dans les limitations de ressources entre chaque nœud fog. Ils ne répondent donc pas au critère d'hétérogénéité (C4). Cependant, ils respectent celui de QoS (C1), car ils imposent des seuils sur la latence et l'énergie consommée. Le critère d'évolutivité élastique (C2) et le critère de prise en charge de la mobilité (C5) ne sont pas satisfaits à leur tour. Les auteurs évaluent leurs politiques uniquement dans un scénario à petite échelle et n'étudient pas l'impact de la mobilité sur les performances de l'algorithme. En outre, le critère fédération (C3) n'est pas satisfait et la possibilité de sous-traiter des

tâches à d'autres systèmes fog n'est pas envisagée.

Les travaux présentés dans [47, 51] étudient à leur tour le problème de planification des tâches dans la couche fog, en examinant des solutions prêtant attention au comportement de l'utilisateur. Kiani et al. [47] proposent un algorithme d'allocation de ressources proactif pour réserver tous les types de ressources aux utilisateurs finaux. Leur algorithme intègre l'historique des données des utilisateurs et attribue davantage de ressources aux utilisateurs fidèles d'un service demandé et au fournisseur du service en général, leur offrant ainsi une QoS plus élevée. Karagiannis et al. [51] ont approfondi l'idée d'allocation des ressources en tenant compte des différences entre les types de périphériques. En outre, ils complètent leur stratégie d'allocation des ressources par une stratégie de tarification considérant également compte de la fidélité des utilisateurs. L'évaluation des deux stratégies a été effectuée sur un petit ensemble d'utilisateurs, permettant une allocation adaptative des ressources. Ces stratégies sont utiles pour à éviter le gaspillage des ressources. Dans ces travaux, les auteurs modélisent l'hétérogénéité des ressources disponibles et considèrent la qualité de service du côté de l'utilisateur. En conséquence, ils satisfont aux critères de QoS (C1) et d'hétérogénéité (C4). Cependant, leurs évaluations ne couvrent pas un scénario à grande échelle et les algorithmes ne considèrent pas le cas des dispositifs mobiles. Par conséquent, les critères d'évolutivité élastique (C2) et de prise en charge de la mobilité (C5) ne sont pas satisfaits. Enfin, les auteurs ne tiennent pas compte de la présence de différents fournisseurs, ne respectent pas, de ce fait, le critère de fédération (C3).

En examinant les couches fog et IoT, Moghaddam et al. [49] étudient simultanément la planification des tâches et le placement des images de tâches. Plus précisément, ils considèrent que les tâches peuvent être exécutées soit sur le serveur de calcul dans la couche fog, soit sur les périphériques intégrés, et que les images de tâches peuvent être enregistrées sur les serveurs de stockage. Leur stratégie vise à optimiser conjointement la planification des tâches et le placement des images afin de minimiser le temps d'exécution maximum. Cela implique d'équilibrer les charges entre les utilisateurs finaux et les serveurs de calcul, de placer efficacement des images de tâches sur les serveurs de stockage et d'équilibrer les demandes d'E/S entre les serveurs de stockage. Ils modélisent leur problème en tant que problème non linéaire à nombres entiers mixte (mixed-integer nonlinear problem).). Pour le résoudre, ils proposent un algorithme qui fonctionne en trois étapes et permet de minimiser les trois éléments du temps de traitement, à savoir de calcul, d'E/S et de transmission. Dans les deux premières étapes, l'algorithme minimise indépendamment le temps d'E/S et le temps de calcul, sur la base d'un modèle de programme linéaire à nombres entiers mélangés. Dans la troisième étape, les résultats obtenus sont combinés pour minimiser le temps d'exécution de la tâche. Les résultats montrent que l'algorithme proposé surpasse les solutions gloutonnes, en variant les différents taux d'arrivée des tâches, les taux de traitement des clients et les taux de lecture du disque. Dans leur modèle et leur algorithme, les auteurs prennent en compte des limites du stockage et des ressources de calcul des nœuds fog. De plus, ils visent à minimiser le temps total de traitement de la tâche. Ils répondent donc aux critères de qualité de service (C1) et d'hétérogénéité (C4). Cependant, ils ne couvrent pas la mobilité des utilisateurs, ce qui a un impact significatif sur le temps total de traitement. De plus, leurs évaluations sont effectuées sur un scénario à petite échelle, sans donner de détails sur les paramètres de simulations, laissant ainsi le critère d'élasticité (C2) et le critère de prise en charge de la mobilité (C5) non satisfaits. Enfin, ils ne tiennent pas compte de la présence de différents opérateurs et ne remplissent donc pas le critère de fédération (C3).

Alors que les études précédentes n'envisageaient pas la possibilité d'exécuter des tâches dans la couche cloud, Prazeres et al. [48] expliquent l'utilité du cloud et proposent un algorithme permettant de répartir efficacement la charge de travail sur les couches fog et cloud. Ils considèrent que dans un domaine de la couche fog, un gestionnaire de serveur fog reçoit les requêtes d'un utilisateur final et il est responsable de la mise en correspondance des requêtes de l'utilisateur avec les ressources fog. À la réception d'une requête, le gestionnaire de serveur fog vérifie si suffisamment de ressources de calcul sont disponibles dans le domaine. Selon les ressources disponibles, le fog peut respectivement exécuter toutes les tâches; seulement une partie et en retarder l'exécution

du reste ; ou transférer et exécuter les tâches dans le serveur cloud. Les résultats de la simulation montrent que l'algorithme proposé est plus efficace que d'autres stratégies existantes visant à optimiser le délai de réponse ou à permettre l'équilibrage de charge. Cela se traduit par un temps de réponse maximum plus bas et des valeurs de temps de traitement maximum plus faibles, à des coûts de traitement moindres. Dans ce travail, les auteurs modélisent l'hétérogénéité des limitations des ressources de calcul dans les couches fog et cloud et considèrent qu'il existe suffisamment de ressources réseau pour la communication entre les deux. En outre, ils visent à satisfaire la contrainte de latence du côté de l'utilisateur. En conséquence, ils répondent au critère de qualité de service (C1) et d'hétérogénéité (C4). Néanmoins, ils effectuent leur évaluation dans un environnement de simulation à petite échelle. Ils ne couvrent pas non plus la mobilité de l'utilisateur, ce qui constitue une violation du critère d'évolutivité élastique (C2) et du critère de prise en charge de la mobilité (C5). Enfin, ils ne tiennent pas compte de la présence de différents fournisseurs et ne répondent donc pas au critère de fédération (C3).

Deng et al. [50] ont également abordé le problème de planification des tâches entre la couche cloud et la couche fog. Contrairement à Prazeres et al. [48], les auteurs visent à réduire la consommation d'énergie du système tout en tenant compte des contraintes supplémentaires. En particulier, ils tiennent compte des limites des nœuds fog et du cloud en termes de capacités de calcul, des limites de la bande passante de communication entre le nœud fog et le cloud, des contraintes de délai du côté de l'utilisateur et de l'équilibrage de charge entre la couche fog et la couche cloud. En raison de sa complexité, le problème à résoudre est divisé en trois parties. Dans la première partie, ils se concentrent sur l'optimisation de la puissance dans la couche fog pour une charge de travail donnée, à l'aide de méthodes d'optimisation convexes. Dans la deuxième partie, ils optimisent la puissance dans la couche cloud pour une charge de travail en entrée, sur la base d'une heuristique d'optimisation linéaire. Dans la troisième étape, ils optimisent la communication réseau entre le fog et le cloud, à l'aide d'un algorithme d'optimisation combinatoire. Les auteurs évaluent leur stratégie en considérant un scénario avec un ensemble de nœuds fog et nœuds cloud. Leurs résultats montrent que plus la charge attribuée aux nœuds fog est importante plus la consommation d'énergie du système augmente tandis que le délai diminue. En effet, les nœuds de la couche cloud sont plus puissants, mais la consommation d'énergie de ce dernier est élevée comparée à celle du fog. Au cours des simulations, des délais de communication supplémentaires sont imposés par les auteurs. Dans ce travail, les limitations en ressources ainsi que la contrainte de délai par utilisateur sont couvertes. En conséquence, les critères de qualité de service (C1) et d'hétérogénéité (C4) sont satisfaits. Néanmoins, les critères d'évolutivité élastique (C2) et de prise en charge de la mobilité (C5) ne sont pas remplis, du fait la mise en œuvre d'un scénario d'évaluation à petite échelle et du non-respect de la mobilité des dispositifs. Le critère de fédération (C3) (C3) n'est pas satisfait : un seul fournisseur fog et un seul fournisseur de cloud interviennent.

Offloading et redistribution de la charge : Comme indiqué dans la section précédente, plusieurs algorithmes de planification des tâches ont été proposés jusqu'à présent dans le contexte des systèmes fog. Bien qu'ils permettent de répartir les tâches de calcul sur les trois couches du système, ils n'ont pas inclus en compte le déséquilibre possible entre les nœuds en termes de charge. En effet, ces algorithmes se concentrent sur la minimisation de la probabilité de blocage des tâches, de la latence dans le système ou de la consommation d'énergie, ce qui peut conduire à un déséquilibrage de charges entre les nœuds. Ce constat souligne la nécessité des algorithmes qui effectuent l'offloading et la redistribution de la charge dans le système. Ce qui met en avant la nécessité des algorithmes qui effectuent l'offloading et la redistribution de la charge dans le système. Dans ce contexte, les travaux décrits dans [52, 53] se sont concentrés sur l'offloading des objets ou des end users vers des nœuds fog. En revanche, les travaux [54, 55] se sont uniquement penchés sur le fog, et ont abordé l'offloading et la redistribution de la charge. À leur tour, le travail [56] examine également sur le fog et propose des algorithmes de redistribution de tâches et/ou l'injection de tâches redondantes dans le système. Enfin, dans [57], les auteurs étudient les migrations dans l'ensemble du système, un aspect orthogonal au offloading et à la redistribution des charges.

Yu et al. [52] s'attaquent au problème du point de vue des appareils mobiles et cherchent à offloader leur charge. Pour étudier le problème, ils modélisent les interactions entre des fonctions sous forme d'un graphe. Dans ce graphe, les fonctions sont modélisées par des nœuds et leurs interactions sont cartographiées sur des arêtes. L'objectif des auteurs est de diviser le graphe en deux parties : l'une qui s'exécute sur l'appareil mobile et l'autre qui est déchargée sur les nœuds de la couche fog, avec un temps d'exécution le plus petit possible. Les résultats expérimentaux montrent que l'offloading dans la couche fog est plus performant en matière de temps de réponse et de consommation d'énergie que dans la couche cloud, ou que si tout se trouvait exécuté sur le dispositif mobile. Dans ce travail, les auteurs prennent en compte les différentes limitations des ressources disponibles et visent à satisfaire les contraintes de QoS, remplissant ainsi les critères de QoS (C1) et d'hétérogénéité (C4). Cependant, l'évaluation s'effectue dans un environnement à petite échelle; le critère d'évolutivité élastique (C2) n'est donc pas rempli. La mobilité n'est pas non plus couverte par l'étude. En outre, la fédération n'est pas prise en compte, ce qui laisse les critères de la mobilité (C5) et de la fédération (C3) insatisfaits.

Shah-Mansouri et al. [53] étendent la portée de [52] et envisagent d'offloader simultanément des serveurs MCC et des appareils mobiles vers un domaine dans la couche fog. Plus précisément, les auteurs considèrent qu'un système MCC est déployé sur des unités situées à l'extrémité d'une route dans une zone urbaine et proposent d'ajouter des bus (nœuds fog) dotés des capacités de calcul. Dans ce contexte, les auteurs proposent une stratégie de offloading basée sur un algorithme génétique. L'objectif est d'offlpader des appareils à faible coût de transmission et d'énergie, tout en garantissant une grande utilité et en respectant le délai d'application de l'utilisateur. L'approche s'appuie ensuite sur ces solutions pour créer de nouvelles possibilités dans les étapes de croisement et de mutation des algorithmes génétiques. Les solutions les plus efficaces sont de nouveau sélectionnées. La procédure continue alors à s'exécuter jusqu'à ce que le nombre d'itérations dépasse une limite définie. L'évaluation de la stratégie montre qu'elle permet d'offloader des appareils avec un faible coût, même en présence d'un nombre élevé d'utilisateurs. En outre, il est démontré que le coût diminue considérablement en présence d'un nombre élevé de bus. Dans ce travail, les auteurs considèrent que les dispositifs sont homogènes dans un domaine fog. Ils ne répondent donc pas au critère d'hétérogénéité (C4). Cependant, ils fixent un seuil pour le délai de réponse, satisfaisant ainsi le critère de qualité de service (C1). Les évaluations sont menées dans une petite zone de 2x2 km^2. En conséquence, le critère d'évolutivité élastique (C2) n'est pas satisfait. Quant au critère de prise en charge de la mobilité (C5), il est satisfait puisque les auteurs mènent leur étude en tenant compte des mouvements des bus. Enfin, les auteurs se concentrent sur le transfert vers un seul domaine fog; le critère de fédération (C3) n'est donc pas pertinent pour leur étude.

Jošilo et al. [54] ne ciblent que l'offloading dans la couche fog et se concentrent sur un scénario dans lequel des DC sont déployés dans la couche fog. Dans ce contexte, ils étudient le problème d'offloading des petits DC vers les grands DC voisins. Plus précisément, en cas de requêtes bloquées dans un DC de petite taille, ils les transmettent à un DC voisin, de plus grande taille, de secours selon une probabilité d'offloading. Les auteurs caractérisent analytiquement les performances du système et les complètent par des résultats numériques. Sur la base de leur analyse, les auteurs montrent que la stratégie proposée réduit considérablement le taux de blocage des requêtes dans un petit serveur, avec un impact mineur sur le taux de blocage des grands DC. Dans ce travail, les auteurs considèrent l'hétérogénéité des DC, répondant ainsi au critère d'hétérogénéité (C4). Comme ils visent également à réduire le taux de blocage des requêtes, ils satisfont au critère de QoS (C1). De même, le critère d'évolutivité élastique (C2) est validé, puisque les auteurs considèrent le cas d'une charge élevée. En revanche, le critère de prise en charge de la mobilité (C5) n'est pas pertinent pour ce travail, car les méthodes fonctionnent au niveau des centres de données. Enfin, le critère de fédération (C3) n'est pas rempli, car les auteurs ne traitent pas la possibilité d'avoir des DC appartenant à des fournisseurs différents.

Dans une perspective similaire, Lyu et al. [55] proposent un algorithme d'équilibrage de charge dynamique dans la couche fog. Cette stratégie permet de gérer l'arrivée et la sortie dynamique de

nœuds dans un domaine. L'algorithme commence par une étape d'atomisation qui cartographie les ressources physiques en ressources virtuelles. Un graphe représentant le système est ensuite construit. Dans ce graphe, chaque ressource virtuelle est représentée par un nœud qui a une certaine capacité. Un lien relie chaque couple de nœuds virtuels et se trouve pondéré par la bande passante du lien de communication qui les unit. Dans un processus itératif, le graphe est partitionné en supprimant les liens les uns après les autres, en fonction d'un seuil de poids minimal garantissant un degré compatible de répartition des tâches sur les machines virtuelles. À l'arrivée d'un nouveau nœud fog, l'algorithme redistribue les charges dans son voisinage afin de les équilibrer, tout en tenant compte du degré de distribution de la tâche et des liens entre les nœuds. Une stratégie inverse est adoptée en cas de suppression de nœud. L'algorithme est évalué par rapport à une approche statique dans la littérature. Les résultats de l'évaluation montrent que la stratégie proposée entraîne une réduction du nombre de déplacements dans le graphe, ce qui implique un coût de migration inférieur. En outre, il faut moins de temps pour obtenir des résultats par rapport à la stratégie statique. Dans ce travail, les auteurs considèrent les nœuds fog comme homogènes, laissant ainsi le critère d'hétérogénéité (C4) non satisfait. Cependant, ils visent à pourvoir aux requêtes des utilisateurs en matière de délai, et répondent par conséquent au critère de QoS (C1). Toutefois, le critère d'évolutivité élastique (C2) n'est pas validé, car les évaluations sont effectuées sur un scénario comportant seulement dix nœuds fog et dix périphériques mobiles. Le travail répond au critère de mobilité (C5), car les auteurs rendent compte de l'arrivée et de la sortie des nœuds fog en raison de leur mobilité. Enfin, la nécessité du critère de fédération (C3) n'est pas pertinente pour ce travail, car les auteurs opèrent au niveau d'un seul domaine fog.

Liang et al. [56] introduisent un algorithme permettant de redistribuer les tâches et/ou d'injecter des tâches redondantes dans la couche fog. Le cadre fonctionne en considérant le compromis entre la charge de communication et la latence de traitement des tâches. En fonction des caractéristiques du système et des exigences imposées, l'un des deux L'algorithme de codage est utilisé. Le premier algorithme de codage vise à minimiser l'utilisation de la bande passante. L'algorithme exécute davantage de calculs sur chaque nœud, ce qui nécessiterait un échange d'informations plus faible entre les nœuds. Ceci conduit donc à une charge de communication réduite. Le second algorithme de codage vise la minimisation de la latence. Il fonctionne en injectant des calculs redondants sur les nœuds, permettant ainsi de réduire le temps de calcul, dans le cas où certains nœuds seraient plus lents que d'autres ou bloqués. Les algorithmes de codage utilisés sont conçus de manière à gérer l'hétérogénéité des nœuds concernant les capacités de calcul. Ainsi, le travail répond au critère d'hétérogénéité (C4). Le cadre permet de sélectionner l'algorithme de codage en fonction de la latence de calcul. Le critère de gestion de la qualité de service (C1) est donc rempli. Cependant, l'évaluation ne couvre pas un tel scénario. Le critère d'évolutivité (C2) n'est donc pas validé, de même que le critère de mobilité (C5), puisque l'impact de la mobilité n'est pas couvert. Enfin, le critère de fédération (C3) n'est pas rempli dans le cadre de ce travail visant un seul fournisseur.

Du et al. [57] abordent un aspect complémentaire de la redistribution de la charge de travail, car ils traitent de la migration dans le système fog, en accordant une attention particulière à la mobilité des appareils. Leur objectif est de permettre les migrations tout en minimisant les coûts de placement et de migration, en respectant la contrainte de latence du consommateur et en tenant compte de la mobilité ce dernier. Pour ce faire, ils proposent de construire un plan de migration pour chaque opérateur, en déclenchant les migrations à des intervalles de temps discrets. Le plan de migration est obtenu à partir d'une structure chronologique montrant les migrations possibles d'un opérateur dans le temps en fonction de la mobilité de l'utilisateur. Dans cette structure temporelle, le chemin le plus court reflétant la plus faible utilisation du réseau est choisi comme plan de migration. Au cours de l'étape suivante, les plans de migration sont coordonnés entre les opérateurs, afin de garantir une quantité suffisante de ressources pour la migration. Un algorithme similaire est introduit pour les modèles de mobilité incertains, avec des plans de migration incluant des cibles supplémentaires liées à des probabilités de transfert pondérées. L'évaluation de cette stratégie montre qu'elle économise respectivement 49 % et 27 % des ressources d'utilisa-

tion du réseau en cas de mobilité complète ou incertaine. Dans cette étude, les limitations en ressources hétérogènes sont prises en compte, ainsi que la qualité de service de l'utilisateur et des évaluations sont effectuées sur un échantillon de 1 000 véhicules présentant des algorithmes de mobilité réalistes, ce qui indique que la méthode peut être utilisée à grande échelle dans un environnement réel. Sur cette base, les critères d'hétérogénéité (C4), de qualité de service (C1) et d'évolutivité élastique (C2) sont remplis. La mobilité reste au cœur de la méthode proposée. Par conséquent, le critère de prise en charge de la mobilité (C5) est rempli. En outre, différents coûts de ressources sont considérés, ce qui permet de prendre des décisions au sein d'une fédération. En conséquence, le critère de fédération (C3) est satisfait.

Le travail de Du et al. [58] étend la coopération entre le fog et le cloud dans les environnements de cloud computing mobile pour améliorer les services d'offloading vers l'équipement utilisateur mobile intelligent (UE) avec des tâches intensives en calcul. Les auteurs ont abordé le problème de l'optimisation conjointe des décisions d'offloading et de l'allocation des ressources de calcul, de la puissance d'émission et de la bande passante radio tout en garantissant l'équité des utilisateurs et le délai maximal tolérable. Ce problème d'optimisation est formulé pour minimiser le coût pondéré maximal du délai et de la consommation d'énergie parmi tous les UE, qui est un problème de programmation non linéaire à nombres entiers mixtes. Les auteurs ont proposé un algorithme sous-optimal de faible complexité pour résoudre le problème ci-dessus, dans lequel les décisions de déchargement sont obtenues par relaxation semi-définie et randomisation, et l'allocation des ressources est obtenue à l'aide de la théorie de la programmation fractionnaire et de la double décomposition lagrangienne. Les auteurs considèrent les nœuds fog comme homogènes, laissant ainsi le critère d'hétérogénéité (C4) non satisfait. Cependant, ils visent à pourvoir aux requêtes des utilisateurs en matière de délai, et répondent par conséquent au critère de QoS (C1). Toutefois, le critère d'évolutivité élastique (C2) n'est pas validé, car les évaluations sont effectuées sur un scénario comportant seulement dix nœuds fog et dix périphériques mobiles. Le travail répond au critère de mobilité (C5), car les auteurs rendent compte de l'arrivée et de la sortie des nœuds fog en raison de leur mobilité. Enfin, la nécessité du critère de fédération (C3) n'est pas pertinente pour ce travail, car les auteurs opèrent au niveau d'un seul domaine fog. Tao et al. [59] étudient un problème d'efficacité énergétique avec des performances garanties dans mobile-edge computing. Pour réduire la consommation d'énergie avec une meilleure performance des tâches, les auteurs présentent un schéma d'offloading des requêtes. Le schéma présenté est déterminé par la consommation d'énergie et la capacité de bande passante à chaque intervalle de temps et applique des conditions KKT pour le résoudre. Les critères de qualité de service (C1) et d'hétérogénéité (C4) sont satisfaits. Néanmoins, les critères d'évolutivité élastique (C2) et de prise en charge de la mobilité (C5) ne sont pas remplis, du fait la mise en œuvre d'un scénario d'évaluation à petite échelle et du non-respect de la mobilité des dispositifs. Le critère de fédération (C3) (C3) n'est pas satisfait : un seul fournisseur fog et un seul fournisseur de cloud interviennent.

2.3.2 Algorithmes pour le stockage et la distribution de contenu

Outre le calcul, le stockage et la distribution de contenu constituent des aspects importants des systèmes fog. Les travaux antérieurs ont principalement considéré ces aspects dans le contexte spécifique des réseaux d'accès par Fog Radio Access Networks (F-RAN), comme c'est le cas de Tandon et Simeone [60], Park et al. [61], Hung et al. [62], et Xiang et al. [63]. D'autres travaux ont couvert cet aspect pour les systèmes fog, opérant sur le réseau de communication sous-jacent, comme c'est le cas de Do et al. [64], Jingtao et al. [65] et Malensek et al. [66].

F-RAN est une extension du concept Cloud-RAN (C-RAN). Plus précisément, le C-RAN permet la cloudification des fonctionnalités radio dans les réseaux cellulaires, offrant un degré de flexibilité important dans la gestion du réseau de radio cellulaire. Cependant, cet avantage est contrebalancé par une latence élevée dans le système. F-RAN le complète dans ce sens, en gardant une partie des fonctionnalités de la radio proche des utilisateurs, en permettant la mise en cache du contenu et en servant le contenu aux utilisateurs avec une latence faible. Tandon et Simeone [60] analysent les performances du système F-RAN en mettant l'accent sur le compromis qui peut être

TABLEAU 2.1 – Comparaison entre les différents algorithmes pour le traitement de tâches dans le Fog Computing.

Algorithme	Méthode proposée pour						Critères d'évaluation				
	Allocation de ressources	Provisionnement des ressources	Partage de ressources	Répartition de charges	L'efficacité énergétique	L'équilibrage de charges	C1	C2	C3	C4	C5
Al-khafji et al. [42]	✓	✗	✓	✗	✓	✗	✓	✗	✗	✓	✗
Yousefpour et al. [43]	✓	✗	✓	✗	✓	✓	✗	✗	✗	✓	✗
Xiao et al. [44]	✗	✗	✓	✗	✓	✗	✓	✗	✗	✓	✗
Ni et al. [45]	✓	✗	✗	✗	✗	✗	✓	✗	✗	✓	✗
Delta et al [46]	✗	✗	✗	✗	✗	✓	✓	✗	✗	✗	✗
Kiani et al. [47] , Karagiannis et al.[51]	✓	✗	✗	✗	✗	✗	✓	✗	✗	✓	✗
Prazeres et al. [48]	✗	✗	✗	✓	✗	✓	✓	✗	✗	✓	✗
Moghaddma .[49]	✗	✗	✗	✗	✗	✓	✓	✗	✗	✓	✗
Deng et al. [50]	✗	✗	✗	✓	✓	✓	✓	✗	✗	✓	✗
Yu et al. [52]	✗	✗	✗	✓	✓	✓	✓	✗	✗	✓	✗
Shan-Mansouri et al. [53]	✗	✗	✗	✓	✗	✓	✓	✗	✗	✗	✓
Jošilo et al. [54]	✓	✗	✗	✗	✗	✗	✓	✓	✗	✓	✗
Lyu et al. [55]	✓	✗	✗	✗	✗	N/A	✓	✗	✗	✓	✗
Liang et al. [56]	✗	✗	✗	✓	✗	✓	✓	✓	✗	✓	✗
Du et al. [57]	✓	✓	✗	✓	✗	✗	✓	✓	✓	✓	✓
Tao et al. [59]	✓	✓	✗	✓	✗	✗	✓	✓	✓	✓	✓

obtenu entre les mesures de capacité de latence, de mise en cache et de fronthaul. Ils décrivent des résultats analytiques dans le cas de deux utilisateurs desservis par deux nœuds périphériques. Leur analyse montre que selon la capacité de fronthaul, deux régimes différents peuvent être identifiés. Le premier est un régime de capacité à faible trafic frontal qui implique une latence optimale dans le cas où les caches sont de grande capacité. Le second est un régime de haute capacité de charge qui nécessite à la fois la mise en cache et l'utilisation du cloud pour atteindre une latence optimale. Dans ce travail, les auteurs considèrent que toutes les caches ont la même taille, laissant ainsi le support du critère d'hétérogénéité (C4) non satisfait. Cependant, ils visent à optimiser la latence dans le système, répondant ainsi au critère de QoS (C1). Dans le même temps, leurs évaluations ne sont effectuées que pour un nombre restreint d'utilisateurs et de nœuds fog : de fait, le critère d'évolutivité élastique (C2) n'est pas validé. De même, ils ne traitent pas de la mobilité des utilisateurs, et ne remplissent donc pas le critère de prise en charge de la mobilité (C5). Enfin, comme ils considèrent les opérations dans le contexte d'un fournisseur de réseau cellulaire, le critère de fédération (C4) n'est pas pertinent pour leur travail.

Park et al. [61] analysent les performances des modes de transfert matériel et logiciel dans un système F-RAN, afin de fournir du contenu aux utilisateurs. En mode de transfert dur (*hard transfer mode*), ils estiment que le traitement de la bande de base n'est effectué que du côté de l'unité de bande de base ou (Baseband Unit (BBU). En revanche, dans le mode de transfert souple (*soft transfer mode*), ils considèrent que le précodage centralisé est effectué sur la BBU et est complété par un précodage local du côté du remote radio heads (RRH). Dans ce contexte, ils étudient le problème de minimisation de la latence de diffusion et évaluent en conséquence les performances des deux modes. Leurs résultats indiquent que le mode de transfert soft-transfer réduit le temps de latence si la capacité en liaison directe est faible ou si le rapport signal sur bruit ou Signal to Noise Ratio (SNR) est élevé. Au contraire, le mode hard-transfer est plus efficace dans les autres régimes. Dans ce travail, les auteurs considèrent différentes limitations de capacité des caches par rapport aux ressources RRH. Par conséquent, le travail répond au critère d'hétérogénéité (C4). Le critère de QoS (C1) est également rempli puisque les auteurs visent à le minimiser. Toutefois, les critères d'évolutivité élastique (C2) et de prise en charge de la mobilité (C5) ne sont pas validés : en effet, les auteurs ne couvrent pas un scénario à grande échelle ni n'évaluent l'impact de la mobilité de l'utilisateur. Enfin, le critère de fédération (C3) n'est pas pertinent pour leur travail, car les auteurs considèrent un seul fournisseur de réseau cellulaire.

Hung et al. [?] s'intéressent également à l'analyse d'un système F-RAN. En fonction des caractéristiques du contenu, ils visent à identifier celui qu'il faut mettre en cache dans le cloud, ainsi que celui qu'il faut mettre en cache dans la couche fog. Leur objectif est de le réaliser

Hung et al. [62] s'intéressent également à l'analyse d'un système F-RAN. En fonction des caractéristiques du contenu, ils visent à identifier celui qu'il faut mettre en cache dans le cloud, ainsi que celui qu'il faut mettre en cache dans la couche fog. Leur objectif est de le réaliser à un coût de communication moindre, tout en respectant les capacités de liaison de communication et la taille des caches. Les résultats indiquent qu'il est préférable de sauvegarder des contenus Internet de rang élevé et de grande taille sur le cloud. À l'inverse, les fichiers de petite taille sont mieux sauvegardés à proximité des utilisateurs. Dans ce travail, les auteurs ne prennent pas en compte les différences dans les capacités des nœuds. Ils ne répondent donc pas au critère d'hétérogénéité (C4). À l'inverse, ils considèrent le délai comme faisant partie de leur analyse, répondant ainsi au critère de QoS (C1). Le critère d'évolutivité élastique (C2) et le critère de prise en charge de mobilité (C5) ne sont pas satisfaits, car les auteurs procèdent à des évaluations dans le cas d'un seul nœud fog et uniquement avec 100 utilisateurs. En outre, ils n'analysent pas ce qui se passe si les dispositifs se déplacent. Enfin, la nécessité du critère de fédération (C3) n'est pas pertinente pour leur étude, car ils ciblent un fournisseur de réseau cellulaire en particulier.

Xiang et al. [63] vont au-delà de la mise en cache dans le cloud et la couche fog d'un système F-RAN et considèrent que le contenu peut être mis en cache sur l'équipement de l'utilisateur. Dans ce scénario, les auteurs visent à optimiser l'efficacité énergétique du système en améliorant la sélection du mode de communication et l'allocation des ressources lors de la fourniture de contenu

aux utilisateurs. En particulier, ils estiment qu'il existe trois modes de communication différents : modèle d'appareil à appareil, antenne unique et coordination entre antennes. Ils proposent un algorithme qui repose sur l'optimisation des essaims de particules (particle swarm optimization) et montrent que l'approche arrive à un compromis entre l'efficacité énergétique moyenne et le délai moyen. Dans leurs travaux, les auteurs ne tiennent pas compte de l'hétérogénéité des appareils : ils ne remplissent pas le critère d'hétérogénéité (C4). En revanche, ils prêtent attention à la latence côté utilisateur et tentent de la satisfaire, répondant ainsi au critère de la QoS (C1). Cependant, les auteurs ne couvrent pas la scalabilité, ni le support de la mobilité (C2 et C5), ce qui les laisse non validés. Enfin, la nécessité du critère de fédération (C3) n'est pas pertinente pour leur travail, car l'étude cible un fournisseur de réseau cellulaire.

Do et al. [64] considèrent la gestion de contenu dans un réseau de distribution de contenu à base de fog. Ils visent à optimiser la distribution du contenu des DC aux nœuds fog, de manière à maximiser l'utilité correspondante, tout en minimisant l'empreinte carbone. Compte tenu du grand nombre de nœuds fog couverts par les réseaux de distribution de contenu, les auteurs proposent, via un algorithme distribué, de résoudre le problème. Ils divisent également ce dernier en de nombreux sous-problèmes, qui peuvent être résolus en un petit nombre d'itérations. Leurs résultats numériques confirment la performance attendue. Sur la base de simulations effectuées pour un ensemble de 100 utilisateurs, l'algorithme converge vers la solution optimale en quelques itérations. Dans ce travail, les auteurs prennent en compte la limitation de la capacité de charge de travail et supposent qu'une bande passante suffisante est disponible entre les DC et les nœuds du fog pour permettre la distribution de contenu. Ils remplissent donc le critère d'hétérogénéité (C4). Malgré leur pertinence, la qualité de service côté utilisateur et l'énergie consommée pour la distribution de contenu ne sont pas considérées comme faisant partie de la phase de distribution de contenu, ce qui entraîne l'invalidation du critère de qualité de service (C1).. Bien que l'algorithme soit conçu pour faire face à des systèmes fog à grande échelle, son évaluation est conduite dans un scénario à petite échelle avec des paramètres arbitraires. Par conséquent, le critère d'évolutivité élastique (C2) n'est pas rempli. En outre, aucune discussion concernant la mobilité de l'utilisateur n'est abordée, ce qui pourrait également conduire à la non-satisfaction du critère de mobilité (C5). Enfin, la nécessité du critère de fédération (C3) n'est pas pertinente pour l'étude, car les auteurs agissent au niveau d'un seul fournisseur de réseau de diffusion de contenu.

Jingtao et al. [65] s'attaquent au problème de connexion des nœuds cache au moindre coût dans les systèmes fog. En particulier, ils supposent la présence de clusters fog statiques, chaque cluster étant constitué de serveurs de film, de Web, de fichiers et de jeux et chaque serveur Web incluant son propre cache. Ils modélisent leur problème en utilisant une structure de graphe, dans laquelle chaque nœud représente un serveur générique et chaque lien relie deux serveurs à une connexion. Chaque lien est pondéré en fonction du coût de connexion. Pour résoudre le problème, les auteurs considèrent un algorithme d'arbre de Steiner. Ce dernier vise à interconnecter un ensemble de nœuds d'intérêt par un réseau de la plus courte longueur tout en permettant l'ajout des sommets et des arêtes supplémentaires au réseau. L'algorithme proposé fonctionne selon les étapes suivantes : tout d'abord, il construit une nouvelle structure de graphe sur l'ensemble des nœuds d'intérêt, reliés par des arêtes dont les poids représentent les chemins les plus courts entre eux. Ensuite, l'algorithme génère le spanning-tree correspondant. À l'étape suivante, l'arborescence est développée pour inclure les nœuds cachés. Ensuite, l'étendue minimale du graphe développé est dérivée. L'évaluation de la stratégie sur un ensemble de quatre grappes montre qu'elle surpasse l'algorithme du plus court chemin. Dans cette étude, les auteurs considèrent les limites en termes de ressources réseau et répondent ainsi au critère d'hétérogénéité (C4). Cependant, ils ne prennent pas en compte la qualité de service du côté des périphériques, laissant le critère de qualité de service (C1) non satisfait. Les évaluations sont également menées sur un scénario à petite échelle, mobilisant des paramètres arbitraires. Le critère d'évolutivité élastique (C2) n'est donc pas rempli. De plus, la mobilité des appareils n'est pas couverte ; par conséquent, le critère de prise en charge de la mobilité (C5) n'est pas satisfait. Le critère de fédération (C3) n'est pas non plus validé, car les auteurs ne discutent pas la possibilité d'existence de plusieurs fournisseurs fog.

TABLEAU 2.3 – Comparaison entre les différents algorithmes pour le stockage et la distribution de contenu dans le Fog Computing.

Algorithme	Méthode proposée pour						Critères d'évaluation				
	Allocation de ressources	Provisionnement des ressources	Partage de ressources	Répartition de charges	L'efficacité énergétique	L'équilibrage de charges	C1	C2	C3	C4	C5
Tandon et al [60]	✗	✗	✗	✓	✗	✓	✓	✗	✗	✗	✗
Park et al. [61]	✗	✗	✗	✗	✗	✗	✓	✗	✗	✓	✗
Hung et al. [62]	✗	✗	✗	✓	✗	✗	✓	✗	✗	✗	✗
Xiang et al. [63]	✗	✗	✗	✓	✓	✓	✓	✗	✗	✗	✗
Do et al. [64]	✗	✗	✗	✓	✗	✓	✗	✗	✗	✓	✗
Jingtao et al. [65]	✗	✗	✓	✓	✗	✗	✗	✗	✗	✓	✗
Malensek et al. [66]	✓	✗	✗	✓	✗	✗	✗	✗	✓	✓	✗

Malensek et al. [66] considèrent des techniques d'échantillonnage sur des flux de données qui permettent une meilleure utilisation des ressources de stockage dans les systèmes fog. Ils proposent une structure de données hiérarchique, appelée Spillway, pour gérer efficacement l'échantillonnage. De plus, ils utilisent une extension de la technique de prélèvement de réservoir. Cette dernière est une technique d'échantillonnage aléatoire qui opère sur un tableau de taille fixe. Dans leur cas, les entrées sont insérées dans le tableau jusqu'à ce qu'il soit entièrement complété. Une fois remplies, les nouvelles entrées remplacent celles existantes avec une certaine probabilité. La technique étendue utilise le même concept sur la structure du déversoir, un groupe de réservoirs organisés en une structure hiérarchique. De cette façon, la structure du déversoir permet des analyses à court et à long terme. Les résultats de l'évaluation montrent que la technique employée conduit à une erreur demeurant inférieure à 0,25%. Le critère d'hétérogénéité (C4) est satisfait dans ce travail, car les nœuds hétérogènes en termes de technologie de stockage et de capacités sont gérés. Le critère (C2) n'est cependant pas rempli : les auteurs n'étudient pas l'impact de l'échantillonnage sur la qualité de service. Les évaluations sont effectuées dans un environnement réel impliquant seulement quelques appareils. Par conséquent, le critère d'évolutivité (C3) n'est pas validé. Il en est de même pour le critère de mobilité (C4), puisque cette dernière n'est pas abordée. Enfin, le critère de fédération (C5) est rempli, car les auteurs utilisent des mécanismes spécifiques pour permettre le stockage fédéré et la récupération d'informations dans plusieurs domaines.

Le tableau 2.3 résume les travaux qui présentés dans cette section.

2.3.3 Algorithmes de gestion de la consommation énergétique

Parallèlement à la hausse du prix des carburants, les préoccupations environnementales orientent les efforts de recherche vers la consommation d'énergie dans les systèmes informatiques, et tout particulièrement dans les systèmes fog. Des travaux comme [43, 44, 67, 68, 69, 70] ont analysé et évalué la consommation d'énergie dans l'ensemble du système, alors que les travaux [45, 49, 53, 63, 71] ont examiné la conception de stratégies visant à réduire la consommation

d'énergie dans le système. Le tableau 2.5 résume les travaux qui présentés dans cette section.

Les travaux présentés dans [67, 68, 69] se concentrent sur l'analyse de la consommation d'énergie dans les systèmes fog. Dans [67, 68], Sarkar et al. considèrent en particulier la performance des systèmes fog dans le contexte de l'IoT. Ils définissent et comparent plusieurs mesures, notamment la consommation d'énergie, la latence et les émissions de CO2 dans un système mixte Fog-Cloud et un système cloud. En simulant des services IoT en temps réel dans 100 villes desservies par 8 DC, les auteurs concluent que le fog computing est plus efficace que le cloud computing. Du point de vue de la latence, ils observent qu'avec seulement 25% des requêtes de services traitées en temps réel (dans la couche fog), la latence du service diminue de 30% par rapport à celle du cloud. À son tour, la consommation d'énergie diminue de 42,2% et les émissions de CO2 de plus de 50%, ce qui se traduit par une réduction significative des coûts. Avec plus de requêtes de service traitées dans le fog, les auteurs relèvent de meilleurs résultats. Dans ce travail, ils ne tiennent pas compte de l'hétérogénéité des appareils, laissant le critère d'hétérogénéité (C4) non satisfait. Cependant, ils évaluent le délai et l'énergie consommée dans le système fog-cloud et valident ainsi le critère de QoS (C1). De même, ils répondent au critère d'évolutivité élastique (C2), car ils effectuent leurs simulations avec des centaines de milliers d'utilisateurs dans plus de 100 villes. La prise en charge du critère de mobilité (C5) n'est pas satisfaite, car les auteurs ne considèrent pas les mouvements des nœuds dans le système. Enfin, la nécessité d'une fédération (C3) n'est pas pertinente pour cette étude, car elle n'affecte pas l'analyse de la mesure de l'énergie consommée.

Jalali et al. [69] abordent également l'analyse de la consommation d'énergie dans les systèmes fog et compare cette énergie consommée avec celle d'un système cloud. Ils considèrent une couche fog où des nano centres de données (nDC) forment les nœuds fog. Les résultats de leur analyse concordent avec ceux de [67, 68]. Ils montrent que les nDC sont plus écoénergétiques que les DC du cloud, en rapprochant le contenu des utilisateurs finaux. Toutefois, si le réseau de connexion est inefficace du point de vue énergétique ou si la durée d'activité du nDC est élevée, il est noté que la consommation d'énergie augmente rapidement. Dans cette étude, l'hétérogénéité entre les nœuds fog et les nœuds cloud est prise en compte. Le critère d'hétérogénéité (C4) est donc rempli. Les auteurs visent également à traiter les requêtes générées par les objets et, par conséquent, le critère de qualité de service (C1) est satisfait. Toutefois, le critère d'évolutivité élastique (C2) n'est pas satisfait, car l'évaluation est réalisée sur un scénario à petite échelle. Le critère de prise en charge de la mobilité (C5) n'est pas satisfait non plus, car la mobilité des utilisateurs n'est pas considérée dans le système. Enfin, la nécessité d'une fédération (C3) n'est pas pertinente pour ce travail, car elle n'affecte pas l'analyse.

D'autres travaux proposent des algorithmes à exploiter dans le système fog et ont évalué l'impact de leurs stratégies sur la consommation d'énergie. Yousefpour et al. [43] proposent une stratégie de clustering permettant aux petites cellules de partager leurs ressources afin de servir les appareils mobiles de leur charge de travail. Dans leur évaluation, ils montrent que leur stratégie permet de satisfaire un pourcentage plus élevé des requêtes des utilisateurs par rapport aux autres stratégies de clustering, au prix d'une consommation d'énergie intermédiaire par utilisateur. Xiao et al. [44] se concentrent également sur le partage des ressources dans les systèmes fog dans le contexte des réseaux cellulaires, et optimisent l'utilisation du CPU, de la bande passante et du contenu. Dans leur évaluation, ils montrent que l'algorithme proposé permet de réduire le temps de latence et conduit à un rendement énergétique élevé. Enfin, Cao et al. [70] conçoivent des stratégies de détection et d'analyse des chutes dans le cadre d'une application d'atténuation des accidents vasculaires cérébraux. Les composants de leur application sont répartis entre la couche fog et la couche cloud. L'évaluation de leur algorithme montre qu'il en résulte un faible taux d'échecs et un faible taux de fausses alarmes. De plus, le délai de réponse et la consommation d'énergie de la solution proposée sont proches des stratégies existantes.

Moghaddam et al. [49] répartissent les charges de travail entre les nœuds du fog et les nœuds cloud avec une consommation d'énergie du système minimale. Leurs résultats montrent que si une charge plus importante est attribuée aux nœuds fog, la consommation d'énergie du système augmente, tandis que le délai du système diminue. En effet, les nœuds cloud sont plus puissants et

TABLEAU 2.5 – Comparaison entre les différents algorithmes de gestion de la consommation énergétique dans le Fog Computing.

Algorithme	Méthode proposée pour						Critères d'évaluation				
	Allocation de ressources	Provisionnement des ressources	Partage de ressources	Répartition de charges	L'efficacité énergétique	L'équilibrage de charges	C1	C2	C3	C4	C5
Shakar et al. [67, 68]	✗	✗	✗	✓	✓	✗	✓	✓	✗	✗	✗
Jalili et al. [69]	✗	✗	✗	✓	✓	✗	✓	✓	✗	✓	✗
Coa et al. [70]	✗	✗	✗	✗	✓	✓	✗	✓	✗	✗	✗
Chen et al. [71]	✓	✗	✗	✗	✓	✗	✓	✓	✗	✓	✗
Ninging et al. [72]	✗	✗	✗	✓	✗	✓	✓	✗	✗	✓	✗
Lee et al. [73]	✓	✗	✗	✗	✓	✓	✓	✓	✗	✗	✗

plus efficaces en énergie que les nœuds fog, tout en imposant des délais de communication supplémentaires. Shah-Mansouri et al. [53] couvrent le problème des cloudlets et des périphériques qui offload leur charge de travail dans la couche fog. En ce qui concerne les critères d'évaluation, toutes les études précédentes répondent au critère de qualité de service (C1). Le critère d'hétérogénéité (C4) est satisfait, alors que le critère d'évolutivité élastique (C2) et le critère de prise en charge de la mobilité (C5) ne sont pas satisfaits. De plus, le critère de fédération (C3) reste non pertinent pour la plupart des contributions.

2.4 Conclusion

Sur la base des recherches sur l'affectation et la planification des ressources dans les systèmes fog, notre analyse montre que la plupart des chercheurs se sont concentrés sur l'affectation des ressources dans le fog computing. D'autres travaux de recherche sont donc nécessaires pour étudier le partage des ressources et la répartition de la charge de travail. En outre, des recherches supplémentaires sont nécessaires pour traiter des problèmes d'efficacité énergétique, d'équilibrage de charge, de niveaux de service et de qualité de service dans le fog. Les chapitres suivant 4 et 5 prennent en compte tous les critères susmentionnés (telle que l'évolutivité qui est un critère très important dans le contexte des applications IoT), recherche des résultats proches de l'optimum et utilise des approches spécifiquement pour accélérer la recherche. Nous détaillerons nous contributions dans le chapitre 3.

Chapitre 3

Vue d'ensemble des contributions

3.1 Introduction

L'architecture traditionnelle du cloud computing est purement centralisée. Par conséquent, il est difficile pour elle de fournir simultanément des services en temps réel ou à faible latence à des milliards d'objets IoT. Puisque le cloud est développé sur l'idée de la virtualisation des services [74], cela peut être évoqué comme étant une des causes de cette lacune majeure du cloud computing. En effet, dans le processus de virtualisation, les fournisseurs d'un service dans le cloud approvisionnent leurs utilisateurs de plusieurs emplacements distants, ce qui peut altérer la latence perçue ; d'où la nécessité d'un nouveau paradigme distribué qui peut répondre aux exigences des milliards d'objets IoT.

Le fog computing est confronté à plusieurs défis. D'une part, avec les applications IoT sensibles à la latence et aux immenses volumes de données qui sont en augmentation constante, il est important de garantir la qualité de service aux utilisateurs finaux. D'autre part, les plateformes fog ont tendance à être limitées en ressources et en énergie. Ainsi, il faut considérer la gestion des ressources dans les systèmes fog. Une solution efficace pour minimiser l'énergie consommée consisterait à répartir de la manière la plus favorable possible les requêtes/tâches générées par les objets IoT sur le fog et le cloud avec pour but de réduire les communications objets/cloud et fog/cloud. De plus, bien que le fog computing retienne de plus en plus l'attention, la conception d'une solution décentralisée à haute efficacité énergétique pour les applications IoT sensibles à la latence à grande échelle reste un défi. Dans ce but, ce travail cherche à résoudre les problèmes susmentionnés en proposant des approches de déchargement des requêtes/tâches et des stratégies d'allocation de ressources pour des systèmes fog/cloud.

Le reste du chapitre est organisé comme suit. Dans la section 3.2, nous allons détailler nos hypothèses de notre travail et présenter le modèle mathématique de notre système fog/cloud. Par la suite, dans la section 3.3 nous allons présenter deux modèles mathématiques utilisés dans nos solutions pour mesurer le délai de service et l'énergie totale consommée. Le modèle du système ainsi que les modèles de mesure présentés dans ce chapitre seront repris dans les chapitres 4 et 5 pour permettre d'appliquer des algorithmes d'optimisation sur le système et d'évaluer les performances de ces algorithmes.

3.2 Modèle du système

Dans cette section, nous commençons par présenter les principales hypothèses utilisées dans nos contributions. Dans un second temps, nous introduisons notre modélisation mathématique d'un système fog/cloud. Cette modélisation sera utilisée pour évaluer les performances de nos solutions.

3.2.1 Hypothèses

Un système générique de fog/cloud peut être considéré comme une structure réseau à trois niveaux, où :

— Le premier niveau est constitué d'un ensemble d'objets IoT connectés à Internet. Un objet exécute une ou plusieurs applications IoT et peut utiliser cette connexion pour éventuellement distribuer ses requêtes sur l'ensemble des nœuds fog disponibles.

— Le deuxième niveau est constitué d'un ensemble de nœuds fog. Différentes applications sont exécutées sur les nœuds fog et le cloud. Ainsi, la proximité de la couche fog des objets IoT peut être exploitée pour réduire la surcharge d'accès au cloud. Les nœuds fog sont supposés être capables de s'auto-adapter en fonction de la charge.

— Le fog peut traiter la plupart des requêtes générées par les objets IoT et transmettre le reste au cloud pour y répondre ultérieurement. Pour optimiser l'utilisation des ressources, un mécanisme doit matcher correctement la correspondance entre les ressources des nœuds fog et les requêtes des objets.

— Le troisième niveau est le cloud. Le cloud computing est responsable du stockage permanent et de l'analyse approfondie de l'ensemble des données globales. Contrairement à l'architecture traditionnelle du cloud, sollicitée pour chaque petite requête et pour le stockage de chaque bloc de données pertinent ou non, dans un système fog-cloud, le cloud est désormais utilisé de manière périodique et contrôlée. Cela permet de gagner en efficacité. Nous avons un serveur cloud qui héberge un certain nombre de machines de calcul homogènes. Les requêtes et les données sont envoyées d'abord à un nœud fog, et transférées vers le serveur cloud, si ce dernier doit traiter les requêtes.

3.2.2 Modélisation mathématique du système

Dans notre travail, nous supposons que notre infrastructure fog/cloud comprend des objets IoT, une ou plusieurs couches de fog computing et au moins un DC cloud.

Comme illustré dans la figure 3.1, un système fog/cloud comprend un ensemble d'objets IoT $\mathscr{O}$, un ensemble de nœuds fog $\mathscr{FN}$, et un cloud DC $\mathscr{C}$. Les liens entre les objets et les nœuds fog ont une capacité limitée notée par $c_{o,f}$. Les données et les requêtes générées par les objets sont transmises aux nœuds fog via la passerelle edge ou l'AP le plus proche. Les nœuds fog sont connectés au DC du cloud avec des liens à bande passante élevée. Si un nœud fog ne peut pas traiter une requête, il la transmet au DC du cloud.

— Chaque objet $o \in \mathscr{O}$ est défini par un tuple $t_o =< i_o, l_o, \delta_o, \gamma_o >$ où i_o est l'identifiant de l'objet o, l_o est la localisation spatiale de l'objet, $\delta_o$est son type (smartphones, etc.) et γ_o désigne un vecteur de ses capacités matérielles ;

— Chaque passerelle $gn \in \mathscr{GN}$ est définie par un tuple $t_{gn} =< i_{gn}, l_{gn}, \gamma_{gn}, \Lambda_{gn} >$ où i_{gn} est l'identifiant de la passerelle, l_{gn} et γ_{gn} ont les mêmes caractéristiques que dans les objets et Λ_{gn} contient la table d'index de tous les périphériques accessibles depuis gn ;

— Chaque nœud fog $f \in \{$ est défini par un tuple $t_f =< i_f, l_f, \gamma_f, \Pi_f, \Gamma_f >$ où i_f, l_f et γ_f ont les mêmes caractéristiques que dans les passerelles, $\Pi_f \subset \mathscr{GN}$ contient le vecteur de toutes les passerelles accessibles depuis fn et Γ_f est l'ensemble de toutes les applications exécutées dans le nœud fog ;

— $\mathscr{C}$ est le DC du cloud défini par le tuple $t_c =< i_c, l_c, \gamma_c, \bar{\Gamma}_c >$, où i_c, γ_c ont les mêmes caractéristiques que les nœuds fog et $\widehat{\Gamma}_c$ est l'ensemble des applications IoT intégrées dans $\mathscr{C}$.

Une application qui s'exécute sur une infrastructure fog peut être définie comme un ensemble de composants indépendants travaillant ensemble et devant respecter un profil de qualité de service. Une application Iot A est définie par le tuple $A =< i, \Delta, \mathscr{R} >$, où :

— i est l'identifiant de l'application ;

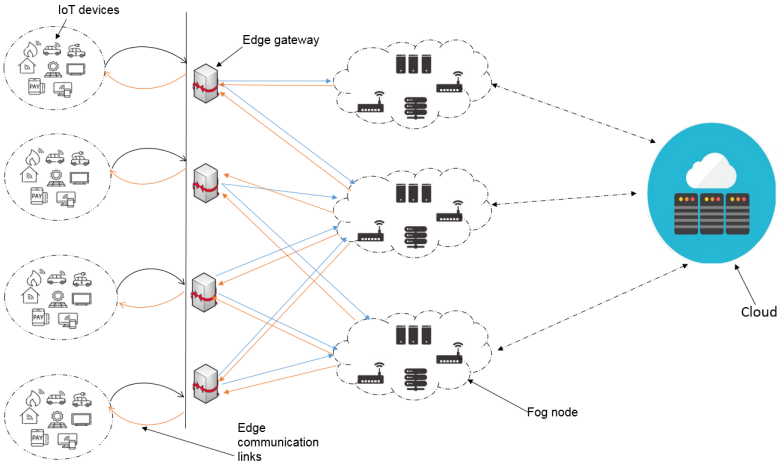


FIGURE 3.1 – Une architecture d'un système fog/cloud.

— Δ est un vecteur qui englobe les exigences matérielles et logicielles de l'application ;

— $\mathscr{R}$ est l'ensemble des requêtes d'objets ; chaque requête est identifiée par $< \mathrm{S}, \mathrm{D}, d >$, où $\mathrm{S} \in \mathscr{O}$ est la source de la requête, $\mathrm{D} \in \mathscr{FN}$ est la destination ; la requête doit atteindre la destination dans un délai d.

3.3 Modélisation mathématique du délai de service et de l'énergie consommée

3.3.1 Modélisation du délai

Dans cette partie, des modèles de latence simples, génériques et basés sur des mesures sont décrits et appliqués au système, aux réseaux et aux services. Ces modèles sont utilisés pour établir des estimations des différents délais du système pour un large éventail de scénarios. Les modèles du délai et de l'énergie consommée du cloud computing et du fog computing présentés dans cette section seront repris dans les chapitres 4 et 5 pour modéliser les performances de nos solutions.

Pour estimer le délai total d'un système fog/cloud, nous adoptons des modèles qui s'appliquent à l'ensemble des équipements du réseau. Dans le but de modéliser le délai, nous proposons deux types de modèles : un *modèle basé sur le flux de données* et un *modèle basé sur le temps*. L'approche pour construire un modèle basé sur le flux de données consiste à mesurer la charge qui traverse les équipements de la couche fog à la couche cloud, ce qui est pratique et utilise le débit de la machine en bits par seconde. En revanche, pour construire un modèle basé sur le *délai par utilisateur*, un autre modèle est proposé, basé sur le temps.

Modèle basé sur le flux de données : La latence d'une requête N_o de taille K_o, générée par $o \in \mathscr{O}$ et transmise à un nœud fog $f \in \mathscr{FN}$, représente le temps de réponse. Elle est modélisée comme la somme de : (a) la latence de transmission ($\mathrm{D}^{\mathrm{Tr}}_{o,f}$), (b) la latence de traitement ($\mathrm{D}^{\mathrm{P}}_{o,f}$), et dans le cas où le nœud fog transmet la requête N_o pour qu'elle soit traitée dans le cloud, nous incluons (c) le délai consommé du fait de la transmission de la requête au cloud ($\mathrm{D}^{\mathrm{Fwd}}_{f,c}$). Ainsi, on peut réécrire le délai total comme suit :

— *Latence de transmission :* soit $\mathrm{D}^{\mathrm{Tr}}_{o,f}$, le délai de transmission de la requête N_o au nœud fog f ; ce délai peut être calculé comme suit :

$$\mathrm{D}^{\mathrm{Tr}}_{o,f} = \mathrm{K}_o * \mathrm{DX}_{o,f} + \mathrm{R}_o * \mathrm{DX}_{o,f} \tag{3.1}$$

où R_o est la taille du résultat après traitement de la requête N_o (en bits) et $\mathrm{DX}_{o,f}$ est le délai de transmission d'un bit de l'objet o au nœud fog f.

— *Latence de traitement :* le délai de traitement de la requête N_o dans un nœud fog ou dans le cloud f est défini comme le temps nécessaire pour répondre à la requête après l'analyse des données accumulées dans les dispositifs du fog. Nous modélisons ce délai de traitement comme suit :

$$D^{Pr}_{o,f} = K_o \times \frac{T^f_{\theta_f}}{\theta_f} \tag{3.2}$$

où K_o est la taille de la requête N_o (en bits), $T^f_{\theta_f}$ le délai de traitement d'une instruction d'un CPU fog (en secondes) et θ_f la taille d'une instruction dans le CPU (en bits). Le rapport $\frac{T^f_{\theta_f}}{\theta_f}$ représente le délai de traitement d'un bit dans f. Le concept de ce modèle a été largement utilisé pour la conception de politiques d'allocation de ressources de calcul [75]. Nous supposons que dans ce cas le fog effectue un équilibrage de charge pour l'ensemble des requêtes à traiter générées par les objets. En d'autres termes, les cycles CPU au sein des nœuds fog sont alloués proportionnellement à chaque requête de sorte que chaque objet subit la même latence d'exécution.

— *Latence du transfert vers le cloud :* pour les requêtes traitées dans le cloud, des latences supplémentaires de transmission et de traitement doivent être ajoutées. Par conséquent, cette latence peut être exprimée comme suit :

$$D^{Fwd}_{o,f} = K_o * DX_{f,c} + K_o * \frac{T^c_{\theta_c}}{\theta_c} + R_o * DX_{f,c} \tag{3.3}$$

où $DX_{f,c}$ est le délai de transmission d'un bit du nœud fog f au cloud c et le rapport $\frac{T^c_{\theta_c}}{\theta_c}$ est la même que celle pour le nœud fog.

Modèle basé sur le temps : Dans ce modèle, nous supposons que les nœuds fog sont regroupés dans des instances fog comprenant plusieurs ressources interconnectées par des réseaux locaux. Nous considérons que chaque nœud fog est un système de file d'attente M/M/n, c'est-à-dire qu'un système de délai d'Erlang peut être envisagé. Une instance fog (FI) est composée de k nœuds fog dans lesquels des serveurs n^k ($0 \leq n^k \leq n * k$) sont activés. Enfin, comme indiqué précédemment, les nœuds fog doivent coopérer avec le DC du cloud pour fournir des services IoT aux objets. Nous supposons que toutes les requêtes des objets parviennent à la couche fog. Ensuite, le processus décide de traiter ou de transmettre une requête est basé selon le temps de traitement local (dans le nœud fog), qui dépend de plusieurs facteurs tels que : la quantité de calcul à effectuer, l'état de la file d'attente (en termes de charge actuelle) et la capacité de calcul du nœud fog.

Nous considérons chaque nœud fog comme un système de délai Erlang, où un nœud fog $f \in \mathcal{FN}$ est représenté par un modèle de file d'attente M/M/n_f, avec n_f serveurs identiques. Les requêtes arrivent selon un processus de Poisson avec un taux $\lambda_f > 0$. Les temps de service sont indépendants et répartis de manière exponentielle avec un taux de $\mu_i > 0$. Lorsqu'une requête trouve tous les serveurs occupés, elle est placée dans une file d'attente à capacité infinie jusqu'à ce qu'elle soit servie par le premier serveur disponible. Une requête générée par un objet, mesurée par sa taille, est une variable aléatoire exponentielle indépendante et identiquement distribuée x avec une moyenne $\overline{x}$. Par conséquent, le temps de traitement des requêtes dans le nœud fog est également une variable exponentielle $\xi = \frac{x}{s}$ avec une moyenne $\overline{\xi} = \frac{\overline{x}}{s}$, où s est la vitesse de traitement d'un serveur du nœud f. Le taux de service moyen d'un nœud fog est défini comme suit : $\mu_f = \frac{1}{\overline{\xi}} = \frac{s}{\overline{x}}$, et le taux d'utilisation du nœud est égal à $\rho = \frac{\lambda_f}{n_f \mu_f} = \frac{\lambda_f}{n_f} \times \frac{\overline{x}}{s}$. Soit $a_f = \frac{\lambda_f}{\mu_f}$ la charge allouée dans le nœud f.

Nous considérons que le cloud héberge un nombre infini de serveurs avec des ressources à capacités infinies. Nous modélisons le cloud en tant que M/G/∞. Le DC disposant de ressources de calcul suffisantes pour traiter les requêtes générées par les objets, le temps d'attente en file des requêtes peut être négligé dans ce cas.

— *Modèle de traitement dans le fog :* étant donné qu'un nœud fog est considéré comme un modèle de file d'attente M/G/n_f avec un taux de service égal à μ_f. Chaque objet envoie toutes ses requêtes générées au nœud fog le plus proche, où chaque requête peut être traitée localement ou transmise au cloud. Nous supposons que chaque requête est transmise au fog en sans-fil. Si un objet $o \in \mathcal{O}$ envoie une requête de taille K_o à un nœud fog $f \in \mathcal{FN}$, le délai de transmission de la requête $\frac{1}{\mu_{wl}}$ peut être défini par :

$$\frac{1}{\mu_{wl}} = \frac{1}{K} B \log_2 (1 + \frac{g_{of} P_{tx,of}}{B N_0}) \tag{3.4}$$

où $g_{of} = \beta_1 d_{of}^{-\beta_2}$ est le gain du canal entre l'objet o et le nœud fog f, avec d_{of} la distance entre eux. β_1 et β_2 sont respectivement l'exposant d'affaiblissement sur le chemin et la constante d'affaiblissement . La puissance consommée pour la transmission de l'objet IoT o au serveur du nœud f est donnée par $P_{tx,of}$. Enfin, B est la bande passante du canal et N_0 est la densité spectrale de puissance de bruit.

— *Modèle de traitement dans le cloud :* étant donné que le cloud est considéré comme un modèle de file d'attente M/G/∞ avec un taux de service μ_{cld}, ce délai est plus rapide que le temps de service dans le fog. Le temps de réponse lorsqu'une requête est en cours de traitement dans le DC du cloud peut être représenté comme suit :

$$R_{cld} = \frac{1}{\mu_{cld}} \tag{3.5}$$

Notons μ_{wn} le délai moyen de transmission d'une requête depuis le fog $f \in \mathcal{FN}$ vers le cloud. Nous utilisons la même équation que (3.4) pour définir μ_{wn}, où l'objet est remplacé par un nœud fog et le nœud fog par le cloud.

3.3.2 Modélisation de la consommation d'énergie

Tout comme pour le délai, pour modéliser l'énergie consommée dans un système fog/cloud, il convient de s'appuyer sur deux modèles, le premier est basé sur le flux de données et le deuxième est basé sur le temps.

Modèle basé sur le flux de données : Lorsqu'un objet o transmets une requête N_o à un nœud fog f, l'énergie consommée pour traiter N_o peut être divisée en trois valeurs : (a) l'énergie de transmission $E_{o,f}^{Tr}$, qui inclut l'énergie consommée par les périphériques de routage du réseau et dans le nœud fog; (b) $E_{o,f}^{Pr}$ l'énergie consommée pour traiter N_o ; et enfin (c) l'énergie de transmission d'un nœud fog vers le cloud, notée $E_{f,c}^{Fwd}$ si N_o doit être traitée dans le cloud.

— *L'énergie consommée pour la transmission :* l'énergie consommée pour la transmission d'une requête vers un nœud fog est la somme de trois composantes : (a) l'énergie consommée pour le traitement initial requis, c'est-à-dire pour l'acheminement au sein du réseau entre l'objet et le nœud ; (b) l'énergie consommée pour la transmission de la requête dans le nœud fog ; (c) l'énergie consommée pour la réception de la requête. L'énergie totale consommée dans le système pour traiter N_o peut être modélisée par une fonction linéaire par morceaux [76], et définie comme suit :

$$E_{o,f}^{Tr} = K_o * E_{o,f}^{Acc} + K_o \times h_f \times \left(\frac{P_f^{idle}}{C_f^{idle}} + \frac{P_f^{max}}{U * C_f^{max}}\right) \tag{3.6}$$

où $E_{o,f}^{Acc}$ est l'énergie consommée pour transmettre un bit de l'objet o au nœud f, h_f est le nombre de sauts dans un nœud fog (équipements de routage), P_f^{idle} est la puissance consommée à l'état inactif ou idle d'un dispositif fog (kw) et P_f^{max} la puissance consommée

de l'élément à sa capacité maximale (kw). Notre modèle utilise le fait que la consommation d'énergie d'un dispositif fog augmente de manière linéaire avec la croissance du trafic, de la valeur de la consommation d'énergie en mode veille à l'énergie consommée lorsque le dispositif est pleinement utilisé. Même si un dispositif est complètement inactif, il consommera toujours plus de 70 % de sa puissance maximale. Notons $\frac{\mathrm{P}_f^{idle}}{\mathrm{C}_f^{idle}}$ l'énergie consommée d'un dispositif dans le fog à l'état inactif et $\frac{\mathrm{P}_f^{max}}{\mathrm{C}_f^{max}}$ l'énergie maximale consommée par le dispositif. Sachant que $0 < \mathrm{U} < 1$ est le taux d'utilisation du périphérique fog, C_f^{idle} le volume de données qu'un périphérique peut gérer à l'état inactif (bit/s) et C_f^{max} le volume de données maximal qu'un périphérique peut gérer à l'état actif (bit/s). Par conséquent $\frac{\mathrm{P}_f^{max}}{\mathrm{U}*\mathrm{C}_f^{max}}$ correspond à la consommation d'énergie estimée du dispositif pour recevoir et/ou transmettre un bit.

— *L'énergie consommée pour le traitement :* la consommation d'énergie pour le traitement de la requête N_o dans le nœud fog f est donnée par :

$$\mathrm{E}_{o,f}^{\mathrm{P}r} = \mathrm{K}_o * \left(\mathrm{P}_f^{idle} * \frac{\mathrm{T}_{\mathrm{N}_o}}{\mathrm{T}_{tot}} + \mathrm{P}_f^{max} * \frac{\mathrm{T}_{act,\mathrm{N}_o}}{\mathrm{T}_{tot}}\right) \tag{3.7}$$

où T_{tot} est le délai nécessaire au traitement de la requête N_o. Ce délai inclut une période appelée $\mathrm{T}_{act,\mathrm{N}_o}$, qui correspond au temps au cours duquel le nœud échange des données avec l'objet qui a généré la requête. Pendant le temps restant de T_{tot}, la requête peut toujours rester dans le nœud fog f mais dans un état inactif. Ce temps inactif s'appelle $\mathrm{T}_{\mathrm{N}_o}$, sachant que la consommation d'énergie pour le traitement est directement liée à la puissance consommée à l'état inactif $\mathrm{P}_f^{idle} \times \frac{\mathrm{T}_{\mathrm{N}_o}}{\mathrm{T}_{tot}}$, et la puissance estimée à l'état actif $\mathrm{P}_f^{max} \times \frac{\mathrm{T}_{act,\mathrm{N}_o}}{\mathrm{T}_{tot}}$.

— *L'énergie consommée du transfert vers le cloud :* dans certains cas, la requête N_o peut nécessiter plus de ressources que le nœud fog ne peut en offrir. Ainsi, l'intervention du cloud est nécessaire. Si N_o doit être traitée dans le cloud, alors le nœud f transmet cette dernière au cloud en utilisant une connexion filaire. De manière similaire au traitement dans le fog, l'énergie consommée par le cloud pour traiter N_o est égale à :

$$\mathrm{E}_{f,c}^{\mathrm{F}wd} = \mathrm{E}_{f,c}^{\mathrm{T}r} + \mathrm{E}_{f,c}^{\mathrm{P}r} \tag{3.8}$$

Les équations (3.6) et (3.7) peuvent être utilisées pour définir $\mathrm{E}_{f,c}^{\mathrm{T}r}$ et $\mathrm{E}_{f,c}^{\mathrm{P}r}$ où le nœud fog remplace l'objet et le serveur cloud remplace le nœud fog dans ces équations.

Modèle basé sur le temps : Tout comme pour le délai, on peut mesurer l'énergie consommée du système en utilisant les caractéristiques des files d'attente.

— *Modèle de traitement du fog :* notons que la fréquence du cycle de calcul d'un nœud fog $f \in \mathscr{FN}$ est fr_f, la consommation d'énergie du calcul dans le nœud fog f est calculée comme suit : κfr_f^{β}, où les paramètres κ et β dépendent de l'architecture matérielle. Étant donné que la vitesse d'exécution du nœud s est linéairement proportionnelle à la fréquence d'horloge fr_f, à savoir $s \propto fr_f$, et que le serveur d'un nœud fog consomme toujours une quantité d'énergie statique, notée par P^*, quand il est inactif. Ainsi, la puissance moyenne d'énergie peut être exprimée comme suit :

$$\mathrm{P}_f = \kappa s^{\phi} + \mathrm{P}^* \tag{3.9}$$

— *Modèle de traitement du cloud :* L'énergie consommée par le cloud pour traiter une requête N_o notée par P_c peut également être donnée par l'équation (3.9).

3.4 Conclusion

Dans ce chapitre, un rappel des objectifs de la thèse a été présenté. Nous avons aussi modélisé mathématiquement notre système et nous avons examiné deux modèles de mesure du délai et la consommation d'énergie d'un système fog-cloud (c'est-à-dire allant des objets, des dispositifs de routage du fog et du cloud aux unités de calcul du fog et du cloud). Dans le premier modèle basé sur les flux de données, la consommation totale d'énergie (et le délai) est déterminée d'une manière incrémentale selon le flux de données qui traverse les dispositifs du système. En revanche le modèle basé sur le temps met en évidence une modélisation de l'énergie et délai qui porte sur les objets (c'est-à-dire que nous modélisons ce qu'a consommé un objet seulement).

Ces modèles seront utilisés dans les chapitres 4 et 5 pour permettre d'appliquer des algorithmes d'optimisation sur le système et d'évaluer les performances de ces algorithmes.

Chapitre 4

Solutions Fog pour les applications IoT avec contenu statique

4.1 Introduction

Le modèle Internet des objets basé sur le cloud a mis en évidence un grand potentiel pour offrir des services de stockage et de calcul aux utilisateurs de l'IoT. Le fog computing a été proposé pour compléter le cloud computing et étendre le rôle de l'IoT à l'extrémité du réseau. Dans ce chapitre, nous nous intéressons aux applications IoT avec contenu statique (ou avec des mises à jour peu fréquentes). Par exemple, une application où les utilisateurs transmettent des requêtes d'accès à un contenu présent dans le système, et où ce contenu peut être un ensemble de fichiers de données (tels que des fichiers vidéo). Ce genre d'applications génère un trafic considérable et pourrait facilement changer le paysage du trafic Internet. Cette augmentation du trafic s'accompagne d'un accroissement de la consommation d'énergie pour le transport, le traitement et le stockage des données.

Le problème traité dans ce chapitre est de savoir comment utiliser pleinement les ressources du fog afin que davantage de requêtes de services puissent être exécutées aux extrémités, tout en réduisant la pression sur le réseau et en garantissant un délai de service et une consommation d'énergie optimaux. Dans le chapitre 3, nous avons modélisé mathématiquement un système fog/cloud ainsi que le délai et l'énergie consommée pour approvisionner une application IoT. Cette modélisation va nous permettre de mieux modéliser notre problématique dans le contexte des applications à contenu statique et de la résoudre plus efficacement. Dans ce chapitre, nous cherchons à développer des stratégies de distribution de requêtes tout en allouant des ressources d'un système fog/cloud d'une manière optimale dans le contexte des applications IoT à contenu statique. Deux approches d'optimisation ont été proposées :

— Une première solution centralisée implémentée dans le cloud qui apporte une vision globale de l'état du système ;

— Une deuxième solution décentralisée implémentée dans les objets, qui a pour but de souligner la concurrence des objets sur les ressources du système.

Dans la section suivante, nous présentons notre première solution centralisée pour résoudre le problème d'allocation de ressources et de distribution des requêtes dans le système.

4.2 Solution implémentée au niveau du cloud

Dans cette section, nous modélisons les défis identifiés grâce une fonction de coût, qui est ensuite utilisée pour modéliser un problème d'optimisation combinatoire. Un algorithme est ensuite proposé pour résoudre notre problématique de manière itérative en utilisant une série d'instances de coupes minimales. Pour cela, nous commençons par identifier deux types de coûts pour

la transmission et le traitement dans le fog et le cloud. Ces derniers seront utilisés pour modéliser le problème d'optimisation combinatoire; nous démontrons par la suite que le problème est NP-difficile ou NP-hard et nous présenterons notre algorithme itératif.

4.2.1 Modèle de coût

Dans cette approche, nous étudions le problème de la distribution d'un ensemble de requêtes pour chaque objet sur un ensemble de nœuds fog et le DC de manière à minimiser le coût total. Nous examinons conjointement deux mesures de performance du système qui sont l'énergie consommée et la latence, et nous modélisons deux types de coûts énergétiques en utilisant le modèle basé sur le flux de données décrit précédemment dans 3.3.1 et 3.3.2.

En considérant le système décrit précédemment dans le chapitre 3 qui est composé de O objets, de GN passerelles, de FN nœuds fog et d'un serveur cloud, nous supposons qu'un ensemble d'applications Γ est déployé sur tous les FN nœuds fog. Chaque nœud fog peut répondre aux exigences de toutes les applications qui sont déployées [60], à savoir :

$$\forall m \in \mathscr{FN} : \gamma_m > \sum_{\mathscr{A} \in \Gamma} \Delta_{\mathscr{A}} \tag{4.1}$$

Un objet o génère et veut distribuer un ensemble de N_o requêtes $\mathscr{R}_o = \{r_{o,1}, r_{o,2}, \ldots, r_{o,N_o}\}$ sur un ensemble de nœuds fog appropriés et sur le cloud. En plus d'interagir avec les nœuds fog, un objet interagit avec le cloud via le nœud fog le plus proche.

Nous commençons par définir des contraintes de latence et de consommation énergétique pour la transmission d'une requête vers un nœud fog; celles-ci sont décrites comme suit :

$$D^{Tr}_{o_i,f}(r_{i,l}) \leq d \tag{4.2}$$

Et,

$$E^{Tr}_{o_i,f}(r_{i,l}) \leq E \tag{4.3}$$

Nous considérons également une contrainte sur la bande passante totale d'un objet pour la transmission d'un ensemble de requêtes :

$$\mathscr{B}(o_i, f) \leq c_e \tag{4.4}$$

Coût du traitement dans le fog : Le coût de traitement de $r_{o_i,l}$ dans le nœud fog f est dénoté par $E^{Pr}_{o_i,f}(r_{o_i,l})$, et peut être calculé en utilisant l'équation (3.7). Ce coût dépend de plusieurs variantes telles que la taille de la requête, les caractéristiques matérielles du nœud fog f et le coût énergétique nécessaire pour maintenir le nœud fog au repos.

Coût du traitement dans le cloud : Dans notre système fog/cloud, certaines des requêtes transmises au fog soient traitées dans le cloud. Si la requête $r_{o_i,l}$ générée par o_i est traitée par le cloud (plus précisément, elle est transférée d'un nœud fog f au cloud), cette dernière est égale à $E^{Tr}_{o_i,c}(r_{i,l}) = E^{Tr}_{o_i,f}(r_{i,l}) + E^{Tr}_{f,c}(r_{i,l})$. Nous désignons l'énergie consommée par le cloud pour le traitement de la requête par $E^{Pr}_{o_i,c}(r_{o_i,l})$.

4.2.2 Modélisation du problème

Le problème d'optimisation conjointe de traitement et de communication d'un ensemble de requêtes est décrit grâce à une matrice de décision x; pour un nœud fog f et pour une requête l générée par un objet o_i, x est définie :

$$x^f_{i,l} = \begin{cases} 1 & \text{traiter la requête } r_{o_i,l} \text{ dans } f \in \mathscr{FN} \\ 0 & \text{traiter la requête } r_{o_i,l} \text{ dans le cloud} \end{cases} \tag{4.5}$$

On peut modéliser notre problématique comme suit :

$$\min_{x_{i,l}^f, r_{i,l}} \sum_{o_i \in \mathcal{O}} \sum_{l=1}^{N_i} \Big[\sum_{f \in \mathcal{FN}} \alpha_f x_{i,l}^f \Big(E_{o_i,f}^{Tr}(r_{i,l}) + E_{o_i,f}^{Pr}(r_{o_i,l}) \Big) + \beta (1 - \prod_{f \in \mathcal{FN}} x_{i,l}^f) \big(E_{o_i,c}^{Tr}(r_{i,l}) + E_{o_i,c}^{Pr}(r_{o_i,l}) \big) \Big] \quad (4.6)$$

$$\text{tel que (4.2)- (4.4)} \quad (4.7)$$

où α_f et β sont des poids relatifs. En fonction des exigences de notre système, nous ajustons les valeurs de α_f et de β mettent en évidence soit le coût dans les nœuds fog ou celui dans le cloud.

Lemme 1. *Le problème d'optimisation dans* (4.6) *est* NP $-$ *hard*.

Démonstration. La preuve est effectuée en approximant le problème en (4.6) à un problème de localisation d'installations non condensées Uncapacitated Facility Location Problem (UFL), connu pour être NP-hard [77]. Ce dernier peut être réalisé en créant une réduction dans laquelle nous considérons la distance dans UFL en tant que coût de traitement dans le cloud; on peut remplacer les fonctions de coût dans UFL sont représentées par les coûts de traitement dans les nœuds fog. □

Pour chaque objet o_i, la fonction objective a pour but de déterminer le vecteur de distribution de ses requêtes sur les nœuds fog et le cloud noté par $x_i = [x_{i,1}^m, \ldots, x_{i,N_i}^m]$, $\forall m \in \mathcal{FN} \cup \{c\}$, de telle sorte que le coût du système total soit minimal et que les conditions (4.2)- (4.4) soient vérifiées.

Dans la section suivante, nous proposons un algorithme de résolution à faible complexité basé sur l'approche *coupe minimum* [78]. Tout d'abord, nous remodelions l'équation (4.6) pour rendre notre problème linéaire. Ensuite, nous détaillerons la procédure de résolution de notre coupe minimum pour obtenir les distributions binaires optimales pour x_i $\forall o_i \in \mathcal{O}$.

4.2.3 Algorithme de distribution de Requêtes-Nœud fog

Notre solution tente d'affecter l'ensemble de requêtes de chaque objet au nœud fog approprié ou au cloud afin de le desservir, et tel que le coût soit total est minimal. Dans un premier temps, l'algorithme affecte chaque requête générée au nœud fog le plus proche. Étant donné que la communication entre les nœuds fog est maintenue avec des liaisons rapides, nous améliorons la solution initiale de manière itérative par des ajustements locaux. Ces ajustements peuvent être obtenus simultanément en résolvant un problème de coupe minimale. Nous proposons ensuite un algorithme de déplacement par expansion également appelé *expansion move* inspiré par [79], qui permet de minimiser le coût total en résolvant le problème de coupe minimale. Plus précisément, nous construisons un graphe et nous encodons tous les coûts en pondération sur les extrémités de celui-ci. Ensuite, nous montrons que la résolution de la coupe minimale du graphe correspond à la décision de distribution optimale. Ainsi, le problème de découpe de graphe peut être résolu efficacement avec une complexité de recherche considérablement réduite.

Transformer la fonction objective : Le second terme de l'équation (4.6) apporte des termes d'ordre élevé en x, ce qui accroît la complexité du problème. Nous pouvons transformer ce terme en une somme de termes quadratiques et linéaires epar l'introduction d'une variable auxiliaire y_f pour chaque $f \in \mathcal{FN}$. Pour un f particulier, on peut récrire le second terme de l'équation (4.6) comme suit :

$$-\prod_{f \in \mathcal{FN}} x_{i,l}^f = \min_{y \in \{0,1\}} \big((N_i - 1) y_f - \sum_{f \in \mathcal{FN}} x_{i,l}^f y_f \big) \quad (4.8)$$

Nous notons qu'il est difficile de calculer avec précision l'énergie totale consommée pour le traitement dune requête dans un nœud fog. En effet, lorsque plusieurs requêtes sont transmises par des objets, la consommation d'énergie de la transmission et celle du traitement peuvent se chevaucher de manière imprévisible; cela dépend de la décision de distribution, de la communication et de l'ordre de planification des requêtes générées par les autres objets présents dans le système. Afin d'affiner la décision de distribution, nous introduisons les contraintes suivantes dans notre solution :

— $x^m_{o_i,l} = 1$ où $f \in \mathscr{FN}$ si et seulement si on a :

$$\mathrm{E}^{\mathrm{Tr}}_{o_i,f}(r_{i,l}) + \mathrm{E}^{\mathrm{Pr}}_{o_i,f}(r_{i,l}) < \mathrm{E}^{\mathrm{Tr}}_{o_i,c}(r_{i,l}) + \mathrm{E}^{\mathrm{Pr}}_{o_i,c}(r_{i,l}) \tag{4.9}$$

— Sinon on a $x^m_{o_i,l} = 0$

Nous introduisons aussi w^i une matrice auxiliaire de taille $\mathrm{L} \times \mathrm{FN} + 1$

$$w^i_{l,m} \triangleq [x^m_{i,1}, \ldots, x^m_{i,l}],\ \forall o_i \in \mathscr{O},\ \forall m \in \mathscr{FN} \cup \{c\} \tag{4.10}$$

Enfin, notre problème peut être modélisé comme suit :

$$\sum_{i=1}^{\mathrm{O}} (b^i)^{\mathrm{T}} w^i \tag{4.11}$$

où $b^i_{\star,m} \triangleq \left[\alpha_1(\mathrm{E}^{\mathrm{Tr}}_{o_i,1}(r_{i,l}) + \mathrm{E}^{\mathrm{Pr}}_{o_i,1}(r_{i,l}), \ldots, \alpha_{\mathrm{FN}}(\mathrm{E}^{\mathrm{Tr}}_{o_i,\mathrm{FN}}(r_{i,l}) + \mathrm{E}^{\mathrm{Tr}}_{o_i,\mathrm{FN}}(r_{i,l})), \beta(\mathrm{E}^{\mathrm{Tr}}_{o_i,c}(r_{i,l}) + \mathrm{E}^{\mathrm{Tr}}_{o_i,c}(r_{i,l}))\right]$

Optimisation via l'expansion move : Dans cette partie, nous montrons qu'il est possible d'obtenir une *expansion move* optimal en résolvant simplement un problème de coupe minimum. Plus précisément, nous construisons un graphe à partir de notre système, dans lequel objets, nœuds fog et DC sont des sommets et nous encodons tous les coûts en pondérations sur les extrémités du graphe. Ensuite, nous cherchons la coupe minimum qui correspond à l'attribution optimale de nœuds fog-requêtes. Nous procédons comme suit :

— *Construction du graphe :* nous modélisons notre système de fog/cloud comme un graphe orienté $\mathscr{G} = (\mathscr{V}, \mathscr{E})$, où $\mathscr{V}$ correspond à l'ensemble des sommets du graphe et $\mathscr{E}$ correspond à l'ensemble des liens. L'ensemble des sommets se compose du serveur cloud $\{c\}$, l'ensemble des nœuds fog $\mathscr{FN}$ et l'ensemble des objets $\mathscr{O}$. Nous plaçons d'abord les sommets objets, ensuite l'ensemble des nœuds fog et le cloud. Enfin, nous ajoutons un nœud source s et un nœud évier t, où s correspond à la décision 0 et t correspond à la décision 1 (c'est-à-dire $\mathscr{V} = \mathscr{FN} \cup \mathscr{O} \cup \{c\} \cup \{s, t\}$.

Nous encodons tous les coûts dans notre graphe comme suit : pour chaque objet o, nous ajoutons un lien de o_i à f avec le poids $\alpha_f(\mathrm{E}^{\mathrm{Tr}}_{o_i,f} + \mathrm{E}^{\mathrm{Pr}}_{o_i,f})$, si cela vérifie aussi le critère (4.9). Nous divisons l'ensemble $\mathscr{FN}$ en deux sous-ensembles $\mathscr{FN}^+$ et $\mathscr{FN}^-$, où pour chaque lien (o_i, f) $f \in \mathscr{FN}^+$, nous ajoutons un lien depuis la source s au noeud o_i avec le poids κ. De même, pour chaque (o_j, f) $f \in \mathscr{FN}^-$, nous ajoutons un lien depuis f à l'évier t avec le poids κ.

— *Expansion move optimal :* pour chaque objet o_i, nous définissons $\mathscr{FN}_i \subseteq \mathscr{FN}$ et $\mathscr{GN}_i \subseteq \mathscr{GN}$, les sous-ensembles de passerelles et de nœuds fog vérifiant les contraintes (4.2)- (4.4) et (4.9). Nous considérons un nœud fog pendant l'opération d'expansion; un objet o_i est autorisé à conserver son ancien nœud fog attribué, ou à basculer vers le nouveau nœud fog si ce dernier peut traiter la requête de l'objet à un coût inférieur. Cette opération est appelée expansion move. Nous cherchons à déterminer l'expansion move optimale qui entraîne une réduction du coût total de l'énergie. Cette étape est répétée jusqu'à ce qu'il n'y ait plus de choix d'affectation nœud fog-requête, c'est-à-dire jusqu'à ce qu'aucun mouvement d'expansion visant à réduire l'énergie consommée ne puisse plus être trouvé.

L'affectation d'une requête peut alors être obtenue en calculant la coupe minimale sur $\mathscr{G}$ en utilisant l'algorithme de flot maximal Edmonds-Karp [80]. Plus précisément, nous affectons la requête après l'expansion move conformément aux contraintes (4.2)- (4.4) et (4.9).

Dans la section suivante, nous allons présenter la solution décentralisée qui sera implémentée dans les objets.Par la suite, nous allons effectuer une série de simulations pour valider la solution centralisée et la solution distribuée. Nous allons aussi comparer entre les performances deux solutions.

Algorithm 1 Algorithme d'assignation (AA)

for $l = 1$ à N_o, $\forall o \in \mathcal{O}$ **do**
  for $m \in \mathcal{F}\mathcal{N} \cup \{c\}$ **do**
    $w \leftarrow Expension(m)$;
  end for
  Calculer le coût du système donné par (5.25);
end for

Algorithm 2 Algorithme de l'Expansion

Construire le graphe auxiliaire $\mathcal{G}$ / (4.2)- (4.4) et (4.9);
Obtenir le graphe minimum min-cut $\mathcal{G}$ en utilisant l'algorithme Karp max-flow;
Extraire la décision d'expansion x depuis le graphe cut;
for $o_i \in \mathcal{O}$ **do**
  $w^i[l,m] \leftarrow f$ if $x^m_{i,l} = 1$;
end for
Return $\{w^i, \forall o_i \in \mathcal{O}\}$

4.3 Solution implémentée au niveau des utilisateurs finaux

Nous cherchons à étudier conjointement la distribution requêtes de services et l'allocation des ressources dans un système fog/cloud, tout en maximisant les performances de celui-ci. Puisqu'il est crucial de considérer la concurrence entre les utilisateurs pour accéder aux ressources du système, nous adoptons une approche basée sur la théorie des jeux afin de modéliser cette concurrence. La théorie des jeux peut fournir un cadre mathématique optimal pour la modélisation et l'analyse du problème d'allocation des ressources dans les systèmes fog computing. Une approche efficace en énergie est proposée pour trouver un résultat stable pour notre problématique.

Nous commençons par modéliser les défis identifiés dans le chapitre 3 sous forme de coûts par utilisateur (ou objet). Ce dernier est utilisé pour modéliser notre problématique sous forme d'un problème d'équilibre de Nash généralisé (Generalized Nash equilibrium problem ou GNEP). Nous étudions par la suite la solution triviale centralisée de notre problématique, puis nous proposons une solution décentralisée. Pour ce faire, nous transformons notre GNEP en un problème d'équilibre de nash pénalisé (Nash Equilibrium Problem ou NEP) et modélisons des conditions de Karush-Kuhn-Tucker pour lisser le NEP pénalisé en un système de méthode de Newton non lisse avec recherche de ligne Armijo. Enfin, nous appliquons la méthode de Newton semi-lisse pour résoudre le problème, et montrons que l'algorithme correspondant peut converger vers un équilibre efficace après plusieurs itérations.

4.3.1 Modélisation du coût par utilisateur

Considérons l'architecture décrite dans la section 3.2.2, où $\mathcal{N}$ est un ensemble de N utilisateurs distribués uniformément dans le système, $\mathcal{M}$ est un ensemble de M nœuds fog et c un serveur cloud. Les nœuds fog sont connectés au cloud via des liaisons filaires. La bande passante consommée par les objets pour accéder à la couche fog est limitée. Nous nous concentrons sur des scénarios dans lesquels les objets décident de manière autonome de tirer parti des ressources du système fog et du cloud.

Un utilisateur $u \in \mathcal{N}$ exécute une ou plusieurs instances d'applications IoT, et génère un ensemble de requêtes $\{r_{u,1}, r_{u,2}, \cdots\}$. Soit $S_{u,i}$ la taille (en octets) de la requête $r_{u,i}$, pour chaque requête, l'utilisateur u peut décider de la transmettre à un nœud fog $j \in \mathcal{M}$ ou au cloud (identifié par 0). Nous définissons le vecteur de probabilité $x_u = \{x_{u,0}, x_{u,1}, \cdots x_{u,M}\}$, tel que $x_{u,j}$ représente la probabilité qu'un utilisateur u transmette ses requêtes à $j \in \mathcal{M} \cup \{0\}$. Enfin, on note par $\mathcal{P}$ l'ensemble des distributions de probabilité de tous les utilisateurs sur $\mathcal{M} \cup \{0\}$.

Nous divisons notre coût en deux parties : la partie communication et la partie calcul. Dans ce qui suit, nous allons présenter ces deux parties en termes de délais de service et d'énergie consommée.

Modèle de communication : Nous supposons que les utilisateurs génèrent des requêtes en suivant un processus de Poisson avec un taux λ_u, $\forall u \in \mathcal{N}$. La taille des requêtes dans ce cas est une variable aléatoire qui suit une loi exponentielle avec une moyenne $\overline{S}_u$. Nous modélisons la file d'attente de transfert d'un utilisateur comme un système M/M/1. Le temps d'attente moyen d'une requête dans la file d'attente de l'utilisateur avant d'être transmise à un nœud de fog j peut être exprimé par :

$$\overline{D}_u^W = \frac{\rho_u^T}{\mu_u - \lambda_u} \tag{4.12}$$

où $1/\mu_u$ est le temps moyen nécessaire pour transmettre une requête et $\rho_u^T = \lambda_u/\mu_u$ est le taux d'utilisation de la file d'attente. Nous supposons que $\lambda_u < \mu_u$, ce qui revient à supposer que $\rho_u^T < 1$.

De manière similaire, nous définissons l'énergie moyenne consommée pour qu'un utilisateur conserve une requête dans sa file d'attente en :

$$\overline{E}_u^W = P_u \times D_u^W \tag{4.13}$$

où P_u est la puissance consommée pour stocker une requête dans une file d'attente.

Si u décide de transmettre une requête à un nœud fog j, le délai moyen nécessaire pour transmettre cette requête est donné par $\overline{D}_{u,j}^{Tr}$ et il est proportionnel à la taille de la requête. Le délai moyen est calculé comme la somme des deux équations (3.1) et (4.12) . Dans le cas où u décide de transmettre sa requête au cloud, $D_{u,0}^{Tr}$ est égal à $\overline{D}_{u,j}^{Tr} + \overline{D}_{j,0}^{Fwd}$.

Nous notons par $\overline{E}_{u,j}^{Tr}$ l'énergie moyenne consommée pour transmettre une requête de u au nœud fog j ; celle-ci est calculée en additionnant les deux équations (3.6) et (4.13). L'énergie moyenne consommée $\overline{E}_{u,0}^{Tr}$ est alors exprimée par la somme de $\overline{E}_{u,j}^{Tr}$ et $\overline{E}_{j,0}^{Fwd}$.

Modèle de calcul : Pour modéliser le délai nécessaire pour le traitement d'une requête dans un nœud de fog, nous considérons que chaque nœud $j \in \mathcal{M}$ maintient une file d'attente au sein de laquelle les requêtes sont placées en FIFO avant d'être traitées. Nous supposons que les files d'attente de traitement sont suffisamment grandes pour sauvegarder toutes les requêtes reçues. Nous notons par $\overline{D}_{u,j}^{Pr}$ avec $j \in \mathcal{M} \cup \{0\}$ le délai moyen nécessaire pour traiter une requête et il est donné par l'équation (3.2).

En considérant l'équation (3.2) et les hypothèses décrites précédemment dans 3.2.1, nous modélisons le processus d'exécution de chaque nœud fog en tant que système M/G/1, et le cloud en tant que système M/G/∞. Nous notons par $\overline{D}$ et ${}^2\overline{D}$ la moyenne et le deuxième moment d'un délai D. Ainsi, le délai de traitement moyen dans un nœud fog $j \in \mathcal{M}$ est exprimé par :

$$\overline{D}_{u,j}^{P} = \frac{\sum_{v \in \mathcal{N}} \lambda_v x_{v,j} \, {}^2\overline{D}_{v,j}^{Pr}}{2(1 - \sum_{v \in \mathcal{N}} \lambda_v x_{v,j} \, \overline{D}_{v,j}^{Pr})} + \overline{D}_{u,j}^{Pr} \tag{4.14}$$

où la file d'attente est stable tant que la charge de travail dans le nœud fog j (le taux utilisation du nœud) $\rho_{u,j}^P = \sum_{v \in \mathcal{N}} \lambda_v x_{v,j} \, \overline{D}_{v,j}^{Pr^k} < 1$.

Lors du traitement d'une requête, le délai de traitement $\overline{D}_{u,j}^{P}$ peut être scindé en deux parties. La première partie est le temps inactif, qui correspond au temps passé dans la file d'attente du nœud fog noté par $\overline{D}_{u,j}^{W}$ (ou $\overline{D}_{u,0}^{W} = 0$ dans le cas du cloud). La deuxième partie correspond à un délai de service qui dépend de la taille de la requête et les paramètres du nœud fog, il également appelé temps actif $\overline{D}_{u,j}^{Pr}$,

L'énergie moyenne consommée pour traiter une requête $\overline{\mathrm{E}}^{\mathrm{P}}_{u,j}$ est donnée par l'équation (3.7) ; celle-ci peut être réécrite comme suit :

$$\overline{\mathrm{E}}^{\mathrm{P}}_{u,j} = \overline{\mathrm{S}}_u \times \left[\mathrm{P}^{idle}_j \frac{\overline{\mathrm{D}}^{\mathrm{W}}_{u,j}}{\overline{\mathrm{D}}^{\mathrm{P}}_{u,j}} + \mathrm{P}^{max}_j \frac{\overline{\mathrm{D}}^{\mathrm{P}r}_{u,j}}{\overline{\mathrm{D}}^{\mathrm{P}}_{u,j}} \right] \tag{4.15}$$

où $\overline{\mathrm{D}}^{\mathrm{W}}_{u,j} = \frac{\sum_{v \in \mathcal{N}} \lambda_v x_{v,j}\, {}^2\overline{\mathrm{D}}^{\mathrm{P}r}_{v,j}}{2(1 - \sum_{v \in \mathcal{N}} \lambda_v x_{v,j}\, \overline{\mathrm{D}}^{\mathrm{P}r}_{v,j})}$.

4.3.2 Modélisation du problème

Dans ce qui suit, nous modélisons le coût total pour chaque utilisateur u à l'état stationnaire en fonction de son vecteur de décision (x_u) et des vecteurs de décision de tous les autres utilisateurs présents dans le système x_{-u}. Soit $\mathrm{D}^{\mathrm{SLA}}_u$ et E^{max}_u les limites supérieures respectivement du délai et de l'énergie consommée pour l'utilisateur u. On peut donc définir la fonction du coût de u comme suit :

$$\mathrm{C}_u(x_u, x_{-u}) = \left[\frac{\overline{\mathrm{D}}_u}{\mathrm{D}^{\mathrm{SLA}}_u} \Big/ \frac{\overline{\mathrm{E}}_u}{\mathrm{E}_u}^{max} \right] \tag{4.16}$$

Où,

$$\overline{\mathrm{D}}_u = \sum_{j \in \mathcal{M}_u \cup \{0\}} x_{u,j} (\overline{\mathrm{D}}^{\mathrm{T}r}_{u,j} + \overline{\mathrm{D}}^{\mathrm{P}}_{u,j}) \tag{4.17}$$

Et,

$$\overline{\mathrm{E}}_u = \sum_{j \in \mathcal{M}_u \,\cup\, \{0\}} x_{u,j} (\overline{\mathrm{E}}^{\mathrm{T}r}_{u,j} + \overline{\mathrm{E}}^{\mathrm{P}}_{u,j}) \tag{4.18}$$

Déterminer la distribution optimale des requêtes pour chaque utilisateur u revient à résoudre le problème d'optimisation suivant :

$$\min \mathrm{C} = [\lambda_u \mathrm{C}_u(x_u, x_{-u})]_{u \in \mathcal{N}} \tag{4.19}$$

$$\text{tel que :} \tag{4.20}$$

$$\mathrm{C1} : \sum_{j \in \mathcal{M}_u \cup \{0\}} x_{u,j} = 1 \tag{4.21}$$

$$\mathrm{C2} : x_{u,j} \geq 0, \quad \forall j \in \mathcal{M}_u \cup \{0\} \tag{4.22}$$

$$\mathrm{C3} : \rho^{\mathrm{T}}_u < 1 \tag{4.23}$$

$$\mathrm{C4} : \rho^{\mathrm{P}}_j < \mathrm{U}_{max}, \ \forall j \in \mathcal{M}_u \tag{4.24}$$

$$\mathrm{C5} : \mathrm{P}_u + \mathrm{P}^{\mathrm{T}}_u < \mathrm{B}_u \tag{4.25}$$

Les contraintes (C1) et (C2) garantissent que les probabilités d'attribution une requête à un nœud fog sont positives et que leur somme est égale à 1. La contrainte (C4) garantit que l'utilisation du nœud fog ne dépasse pas sa capacité maximale U_{max}. Sachant que B_u est le niveau de la batterie, la contrainte (C5) garantit que l'utilisateur u dispose de suffisamment d'énergie pour enregistrer et transmettre ses requêtes.

La fonction objective et le vecteur de décision de chaque utilisateur dépendent des stratégies des autres utilisateurs, ce qui signifie que le problème peut être considéré comme un problème généralisé d'équilibre de Nash. Pour résoudre le GNEP (4.19), nous résoudrons le problème d'inégalité variationnelle VI(K, F)[2] associé, où l'ensemble K est donné par :

$$\mathrm{K} := \Big(\prod_{u \in \mathcal{N}} \widetilde{\mathcal{K}_u} \Big) \cup \mathcal{X}$$

$\widetilde{\mathcal{K}_u}$ représente l'ensemble des solutions possibles de l'utilisateur u sans prendre en considération la contrainte (C4) :

$$\widetilde{\mathcal{K}_u} := \{ x_u \in \mathcal{P} \mid \sum_{j \in \mathcal{M}_u \cup \{0\}} x_{u,j} = 1, \rho^{\mathrm{T}}_u < 1, \mathrm{P}_u + \mathrm{P}^{\mathrm{T}}_u < \mathrm{B}_u \}$$

Pour considérer cette contrainte, nous définissons l'ensemble $\mathscr{X}$ comme :

$$\mathscr{X} := \{x_u \in \mathscr{P} | \rho_j^{\mathrm{P}} \leq \mathrm{U}_{max} \ \forall j \in \mathscr{M}_u\}$$

F est défini comme suit :

$$\mathrm{F} = \begin{pmatrix} \nabla_{x_1} \lambda_1 \mathrm{C}_1(x_1, x_{-1}) \\ \vdots \\ \nabla_{x_{\mathrm{N}}} \lambda_{\mathrm{N}} \mathrm{C}_{\mathrm{N}}(x_{\mathrm{N}}, x_{-\mathrm{N}}) \end{pmatrix} \tag{4.26}$$

Avec,

$$\nabla_{x_u} \lambda_u \mathrm{C}_u(x_u, x_{-u}) = \begin{pmatrix} h_{u,0} \\ h_{u,1} \\ \vdots \\ h_{u,\mathrm{M}} \end{pmatrix} \tag{4.27}$$

où $h_{u,j} = \frac{\partial \lambda_u \mathrm{C}_u(x_u, x_{-u})}{\partial x_{u,j}}$, avec $j \in \mathscr{M} \cup \{0\}$ sont les dérivés du premier ordre.

Théorème 2. *Le GNEP dérivé a au moins une solution.*

(La preuve du théorème est détaillée dans la section A.1 de l'annexe.)

Notons qu'en général, le GNEP (4.19) peut avoir un ensemble infini de solutions. Dans ce qui suit, nous montrons qu'un équilibre peut être déterminé dans certaines conditions. À cette fin, nous montrons que F est monotone. La monotonicité de F est une condition suffisante pour que différents algorithmes puissent résoudre notre problème de VI.

Théorème 3. *Si la taille de la requête est distribuée de manière exponentielle, le taux d'arrivée est* λ_u *et*

$$max_{u \in \mathscr{N}} \lambda_u \overline{\mathrm{D}}_{u,j}^{\mathrm{P}r} \leq \frac{1 - \mathrm{U}_{max}}{\mathrm{N}}, \ \forall j \in \mathscr{M}$$

alors, F *est monotone.*

(La preuve du théorème est détaillée dans la section A.2 de l'annexe.)

Compte tenu de la monotonie de F, nous pouvons utiliser la méthode Extra-gradient [81] pour résoudre notre VI(F, K), ce qui équivaut à résoudre le problème (4.19).

Proposition 4. *En déterminant la taille du pas et la constante de Lipschitz avec une stratégie linéaire, on peut éviter la sensibilité des valeurs d'initialisation. Par conséquent, notre solution converge.*

Démonstration. Sachant que F est monotone, la convergence de la méthode Extra-gradient peut être montrée sous l'hypothèse que la constante de Lipschitz et le pas de la méthode sont déterminés avec une stratégie linéaire. Par conséquent, d'après [82] la méthode Extra-gradient converge à grande vitesse. □

4.3.3 Distribution non coopérative des requêtes

Sachant que l'architecture fog computing est de nature décentralisée et que les objets IoT sont supposés être intelligents et autonomes, nous avons opté pour une distribution décentralisée des requêtes basée sur la théorie des jeux. Cette dernière permet aux utilisateurs de prendre leurs décisions de distribution localement et de trouver une solution mutuellement satisfaisante. Cela allège le fardeau de la gestion centralisée lors du passage à un grand nombre d'objets et réduit les coûts de communication entre le fog et le cloud.

À partir des discussions susmentionnées, les utilisateurs sont considérés comme des joueurs dans notre jeu non coopératif et créent des profils de stratégie pour maximiser leur intérêt (c'est-à-dire minimiser leurs coûts). Dans cette partie, nous modélisons notre jeu non coopératif et proposons un algorithme permettant d'atteindre rapidement une solution stable ; nous prouvons par la suite qu'il s'agit de l'équilibre de Nash.

Analyse de l'existence de l'équilibre :

L'existence de l'équilibre dans un jeu non coopératif est une condition essentielle pour la stabilité du système. Dans ce qui suit, nous allons monter que notre solution tend vers un équilibre de Nash.

Définition 1. (stratégie optimale) Nous définissons le GNEP $\Gamma =< \mathcal{N}, \mathrm{K}, (\mathrm{C}_v)_{v\in\mathcal{N}} >$ où $(x_v)_{v\in\mathcal{N}}) \in \mathscr{P}$. Γ est un jeu stratégique dans lequel chaque utilisateur joue une stratégie mixte $(x_v)_{v\in\mathcal{N}} \in \mathcal{K}_u$. La stratégie optimale pour u est $x_u^\star \in \mathcal{K}_u$ où

$$\mathrm{C}_u(x_u^\star, x_{-u})) < \mathrm{C}_u(x_u, x_{-u}^\star))$$

Nous exprimons la stratégie optimale d'un utilisateur u comme suit :

$$x_u^\star \in \arg\min_{x_u\in\mathcal{K}_u} \mathrm{C}_u(x_u, x_{-u})),\ \forall (x_u, x_{-u}) \in \mathrm{K}$$

Enfin, une stratégie pure de Nash Equilibrium $(x_u^\star)_{u\in\mathcal{N}}$ de Γ, qui correspond à l'équilibre de notre jeu non coopératif est exprimée comme suit :

$$\mathrm{C}_u(x_u^\star, x_{-u}^\star)) < \mathrm{C}_u(x_u, x_{-u}^\star)),\ \forall x_u \in \mathcal{K}_u$$

Lemme 5. *Le jeu non coopératif a au moins un équilibre.*

Démonstration. Le théorème 2 garantit que le GNEP a au moins une solution. Dans ce cas, il suffit de montrer que toute solution du problème VI(K, F) est un équilibre du jeu non coopératif.

Étant donné que la fonction F est continue sur K, il en résulte que $\mathrm{C}_u(x_u, x_{-u}))$ est différentiable sur K. D'après le théorème 3, nous savons aussi que $\mathrm{C}_u(x_u, x_{-u}))$ est une fonction convexe. Par conséquent, toute solution du problème VI(K, F) correspond à un équilibre de Nash pure de Γ [83] et constitue donc un équilibre dans les stratégies mixtes du jeu non coopératif. □

Algorithme de distribution optimal pour un utilisateur :

Nous nous focalisons sur la mise en œuvre de l'approche décentralisée pour le GNEP modélisé auparavant. Comme nous l'avons déjà démontré, le GNEP (4.19) est conjointement convexe. Ainsi, nous pouvons utiliser la méthode de la fonction de pénalité exponentielle pour le résoudre. Tout d'abord, nous remodelions notre problème en une séquence de lissage de problèmes d'équilibre de Nash pénalisé en utilisant une pénalisation partielle des contraintes de couplage. Ensuite, nous utilisons les conditions de Karush-Kuhn-Tucker (KKT) pour lisser les problèmes d'équilibre de Nash pénalisés en un système d'équations non lisses. Enfin, nous appliquons la méthode de Newton semi-lisse pour résoudre le problème.

Sachant que dans (4.19) (C1), (C2), (C3) et (C5) ne dépendent que de la décision de l'utilisateur u, tandis que (C4) inclut les décisions des autres utilisateurs, cela qui signifie qu'il s'agit d'une contrainte couplée. En sacrifiant partiellement la contrainte de couplage (C4), le problème devient un problème standard d'équilibre de Nash. En remplaçant la contrainte de couplage par une fonction de pénalité exponentielle, nous pouvons redéfinir le problème (4.19) pour chaque utilisateur u comme suit :

$$\begin{aligned}
&\min\ \lambda_u \mathrm{C}_u(x_u, x_{-u}) + \frac{1}{\rho_u^{\mathrm{T}}} \sum_{k=0}^{k-1} \exp\left[\rho_u^{\mathrm{T}} g_{u,k}(x_u, x_{-u})\right] \\
&\text{s.t. :} \\
&\mathrm{C6} : \sum_{j\in\mathcal{M}_u\cup\{0\}} x_{u,j} = 1 \\
&\mathrm{C7} : x_{u,j} \geq 0,\ \ \forall j \in \mathcal{M}_u \cup \{0\} \\
&\mathrm{C8} : \rho_u^{\mathrm{T}} < 1 \\
&\mathrm{C9} : \mathrm{P}_u + \mathrm{P}_u^{\mathrm{T}} < \mathrm{B}_u
\end{aligned} \tag{4.28}$$

où $g_{u,k}(x_u, x_{-u})$ est la description courte de $\rho_j^{\mathrm{P}} - \mathrm{U}_{max} < 0,\ \forall j \in \mathcal{M}$.

Théorème 6. *Soit $\{x^k\}$ une séquence positive qui tend vers l'infini. Pour chaque k, x_u^k est la solution du problème d'équilibre de Nash lorsque $\rho_u^{\mathrm{T}} = \rho_u^{\mathrm{T}^k}$. Si $x_u^\star \subset \{x_u^k\}$ (cela respecte ainsi toutes les contraintes), alors $x_u^\star$ est une solution du GNEP dans* (4.19).

Démonstration. Voir le théorème 1 dans Référence [84]. □

Le GNEP d'origine est redéfini comme un ensemble de NEP classiques. À partir du théorème 6, nous pouvons déduire que résoudre l'ensemble des NEP équivaut à résoudre le GNEP. Nous définissons $\grave{\Gamma} =< \mathcal{N}, \mathrm{K}, (\grave{\mathrm{C}}_v)_{v\in\mathcal{N}} >$ le nouveau NEP non coopératif défini dans (4.28).

Théorème 7. *Le nouveau jeu non coopératif $\grave{\Gamma}$ a au moins un équilibre de Nash.*

Démonstration. Nous notons par $\mathcal{K}_u$ l'ensemble des stratégies de l'utilisateur u. Celui-ci représente un sous-ensemble fermé convexe et supérieur-limité convexe sur l'espace euclidien, et la nouvelle fonction de minimisation dans (4.28) est également continuellement différentiable et convexe pour toute stratégie dans $x_u \in \mathcal{K}_u$. Par conséquent, comme dans le cas de la preuve du lemme 5, nous pouvons également affirmer que $\grave{\Gamma}$ a au moins un équilibre de Nash. □

Pour simplifier, nous remplaçons les contraintes (C6), (C7), (C8) et (C9) par $h_{u,k}^{(i)} \leq 0$, $i = 1, \cdots, 4$. La condition KKT pour les NEP (4.28) est définie comme suit :

$$\begin{aligned} &\min \nabla_{x_u} \lambda_u \mathrm{C}_u(x_u, x_{-u}) + \frac{1}{\rho_u^{\mathrm{T}}} \sum_{k=0}^{k-1} exp\left[\rho_u^{\mathrm{T}} g_{u,k}(x_u, x_{-u})\right] \\ &\nabla_{x_u} g_{u,k}(x_u, x_{-u}) + \sum_{k=0}^{k-1} \sum_{i=0}^{3} \gamma_{u,k}^i \nabla_{x_u} h_{u,k}^i(x_u) = 0 \end{aligned} \tag{4.29}$$

ici $x = (x_1, \cdots, x_{\mathrm{N}})$ est le vecteur de stratégie de tous les utilisateurs, et $\gamma = (\gamma_{u,k}^i)$ est le coefficient KKT condition pour tous les utilisateurs.

Assembler tous les systèmes pour $u = 1, \cdots, \mathrm{N}$ conduit au système équivalent suivant :

$$\begin{aligned} &\mathrm{L}(x, \gamma) = \\ &\left\{ \begin{array}{l} \nabla_{x_u} \lambda_u \mathrm{C}_u(x_u, x_{-u}) + \frac{1}{\rho_u^{\mathrm{T}}} \sum_{k=0}^{k-1} exp\left[\rho_u^{\mathrm{T}} g_{u,k}(x_u, x_{-u})\right] \\ \nabla_{x_u} g_{u,k}(x_u, x_{-u}) + \sum_{k=0}^{k-1} \sum_{i=0}^{3} \gamma_{u,k}^i \nabla_{x_u} h_{u,k}^i(x_u) \end{array} \right\}_{u=1}^{\mathrm{N}} \\ &= 0 \end{aligned} \tag{4.30}$$

Il est évident que $\mathrm{L}(x, \gamma)$ est un vecteur nul de grande dimension. Pour résoudre x et γ, nous introduisons la fonction Fischer-Burmeister (FB) $f(a, b) := \sqrt{a^2 + b^2} - (a + b)$. Ainsi, le nouveau système est exprimé par :

$$\mathrm{I}(x, \gamma) = \left\{ \begin{array}{l} \mathrm{L}(x, \gamma) \\ \mathrm{F}(\mathrm{V}(x), \gamma) \end{array} \right\} = 0 \tag{4.31}$$

Nous utilisons la méthode de Newton semi-lisse pour résoudre le système $\mathrm{I}(x, \gamma) = 0$, ce qui équivaut à la résolution du problème (4.28). Nous définissons d'abord la fonction valeur $\Phi(x, \gamma) := \frac{1}{2}\mathrm{I}^{\mathrm{T}}(x, \gamma)\mathrm{I}(x, \gamma)$. Ensuite, nous utilisons la méthode Newton semi-lisse inspirée par [85].

La solution non coopérative présentée dans l'algorithme 3 est décrite comme suit : les joueurs de notre jeu agissent de manière asynchrone. Le coût prévu et la probabilité de distribution pour tous les joueurs sont initialisés à 0. Chaque joueur utilise l'algorithme 3 pour mettre à jour sa propre stratégie optimale de manière alternée. Par conséquent, afin de recalculer la stratégie de distribution, nous réduisons le nombre de requêtes attribuées aux nœuds fog dans la partie précédente. De plus, nous supposons que la déviation notée par ε est une condition de convergence. Elle est utilisée pour déterminer si le système a atteint l'équilibre de Nash. L'algorithme passe lorsque la différence d'écart cumulé entre deux itérations adjacentes est inférieure à ε. Lors de la mise à jour de la stratégie, chaque utilisateur calcule l'écart cumulatif puis le diffuse aux autres joueurs.

Ainsi, ces derniers peuvent décider de continuer à mettre à jour leurs stratégies de distribution des requêtes en fonction des informations publiées. Nous démarrons l'algorithme périodiquement ou lorsque les paramètres du système changent.

Algorithm 3 Decentralized Requests Distribution Algorithm (DRD)

Initialiser ε, $Cost$ et x_u, $\forall u \mathcal{N}$, $k = 0$ avec $w^k = (x^k, \gamma^k)$.
while $||\Phi(w^k)|| \leq \varepsilon$ **do**
  Choisir $V^k \in \partial\Phi(w^k)$, résoudre la valeur S^k du système $V^k \times S^k = -I(w^k)$
  Mettre : $w^k = w^k + \alpha^k$ et $k = k + 1$
  for $\forall u \in \mathcal{N}$ **do**
    Mettre à jour la stratégie de distribution des demandes en x_u^k.
    Calculer le coût en utilisant (4.30).
  end for
  Annoncer l'écart cumulatif$||\Phi(w^k)||$ pour tous les joueurs.
end while

4.4 Évaluation des performances

Dans cette section, des simulations approfondies seront effectuées pour évaluer l'efficacité des algorithmes proposés dans ce chapitre. Nous commencerons par analyser les performances de la solution centralisée présentée dans l'algorithme 1. Par la suite, nous étudierons les performances de la solution décentralisée décrite dans l'algorithme 3. Enfin, nous comparerons l'approche implémentée dans le cloud à celle implémentée dans les objets.

4.4.1 Configuration de la simulation

Les utilisateurs sont répartis sur une superficie de $100m \times 100m$. Les paramètres des simulations sont détaillés dans le tableau 4.1. Pour simplifier, supposons que nous avons dix nœuds fog et que la capacité de calcul F_{fog} de chaque nœud est de 4 (MIPS). Le temps de transmission d'un nœud fog au cloud est de $D^{core} = 0.5$ (secondes), et la capacité de calcul F_0 d'un serveur cloud est de 10 (MIPS). La complexité d'une instruction dans le fog κ_j est une distribution uniforme sur [64,65550] bites. La puissance consommée par chaque router correspond respectivement à 20 W en veille et à 40 W en activité. Les dispositifs du fog computing, au sein d'un nœud, consomment 3,7 MW au total. La consommation d'énergie du cloud serveur est comprise entre [9,7 , 77,4] MW. Nous avons développé dans MATLAB l'algorithme centralisé (Centralized Requests Distribution - CRD) décrit dans 1 et l'algorithme décentralisé (Decentralized Requests Distribution - DRD) décrit dans 3.

En raison de la nature non déterministe de notre système, nous avons considéré 20 exécutions, chacune avec un état de file d'attente aléatoire initialisé différemment, et avons choisi la moyenne des solutions trouvées.

Nous évaluons l'efficacité de nos approches en comparant nos deux algorithmes avec d'autres algorithmes de base : (1) Greedy, dans lequel les requêtes sont envoyées aux nœuds fog les plus proches, puis sur le cloud en fonction de la capacité de service, tout en respectant les contraintes déjà décrites; (2) Best-Effort, dans lequel tous les nœuds fog sont activés et fonctionnent en permanence tant que la file d'attente correspondante n'est pas vide; (3) Cloud-Only, un algorithme où toutes les requêtes sont transférées et traitées dans le cloud; (4) Fog-Only, dans lequel toutes les requêtes sont traitées par des nœuds fog et pour chaque stratégie (CRD ou DRD), il convient d'appliquer l'algorithme sans considérer les ressources du cloud.

Pour l'approche centralisée, nous étudions deux types de scénarios : le premier scénario à faible charge, dans lequel seules des requêtes faibles (en termes de taille) sont générées, et un second scénario à forte charge, comprenant uniquement des requêtes lourdes. Les critères de performance utilisés dans l'évaluation sont les suivantes : le coût énergétique, l'impact du nombre de

TABLEAU 4.1 – Paramètres de simulation

Paramètres	Description	Valeur
N	Nombre d'utilisateurs	{100, 1400}
P_u^T	Puissance consommée pour la transmission de données	0.4 W
$\overline{S}_u$	Taille moyenne des requêtes	[0.1- 3.4]Mb
λ_u et N_o	Flux de données (requêtes)	[0.3,0.8] (requêtes/s)
U_{max}	Seuil de stabilité	0.7
B_u	Niveau de la batterie d'un utilisateur	∞
D^{Tr} et D^{Pr}	Limites de délai d'une application	[15- 25] ms
α	Poids sur les nœuds fog	[1-3]
β	Poids sur le cloud	[1-5]
c_e	Capacité d'un lien de liaison	[10, 100] Mbps
d	Délai	[1- 10] ms

requêtes et des nœuds fog sur le coût total, l'impact de α et β sur le coût total et la durée d'exécution de l'algorithme.

Pour l'approche décentralisée, nous étudions l'impact du nombre d'utilisateurs et de nœuds fog sur les performances du système en termes de délai et d'énergie consommée. Nous comparons par la suite la solution égoïste (non coopérative) proposée avec une solution optimale sociale inspirée de [86]. Notre objectif consiste à comprendre comment fonctionne un comportement égoïste des objets en comparaison d'un comportement social. Enfin, nous comparons l'algorithme DRD avec deux autres algorithmes existants, qui se basent sur la théorie des jeux afin de trouver des solutions stables ; à savoir la méthode de la propulsion par convexe (SCAM) proposée dans [58] et l'algorithme LODCO (Dynamic Computation Offloading) basé sur l'optimisation de Lyapunov et présenté dans [59].

4.4.2 Analyse des performances de CRD

Nous commençons par évaluer le coût total en fonction de N_o le nombre de requêtes générées dans un intervalle. La figure 4.1 montre que les performances de CRD et de Fog-Only sont proches dans le scénario de charge faible. Nous constatons une légère différence de coût entre Fog-Only et Greedy au début de la simulation ; celle-ci s'accroît avec un nombre de requêtes élevé. Dans un scénario à faible charge, les performances des trois solutions sont proches, car ces dernières disposent de suffisamment de ressources pour répartir les requêtes sur les nœuds fog. Nous pouvons également observer que pour toute valeur de N_o, l'algorithme CRD fonctionne mieux que Fog-Only. Cette distinction est due au fait que davantage de requêtes sont envoyées vers le cloud dans l'algorithme CRD, ce qui permet de ne pas surcharger les nœuds fog. Par conséquent, Fog-Only et Greedy ne peuvent pas être entièrement efficaces au sein de systèmes très chargés. Les performances de Cloud-Only sont proches de celles de Best-Effort, avec des valeurs inférieures à N_o. Enfin, Best-Effort fournit les moins bonnes performances, quand N_o augmente.

La figure 4.2 illustre le coût énergétique du système en fonction du nombre de nœuds fog dans un scénario à charge lourde. Nous pouvons constater que notre algorithme CDC atteint le coût le plus bas parmi les quatre algorithmes pour toutes les valeurs de M (nombre de nœuds fog), ce qui vérifie l'efficacité de notre algorithme. L'algorithme Greedy active tous les nœuds fog, entraînant ainsi la plus grande consommation d'énergie, donc un coût élevé. Nous remarquons également que même avec un nombre plus élevé de nœuds fog, CRD surpasse Fog-Only. En effet, la stratégie des CRD tire parti des ressources du fog d'une manière plus efficace. . En d'autres termes, il est possible de distribuer davantage de requêtes et d'obtenir une consommation d'énergie minimale donc un coût moindre. De plus, comme le montre la figure 4.2, la consommation d'énergie dans le système diminue avec le nombre de nœuds fog, car la quantité de ressources allouées à chaque objet augmente. Cependant, à partir d'un certain nombre de nœuds fog $M > 40$, l'énergie

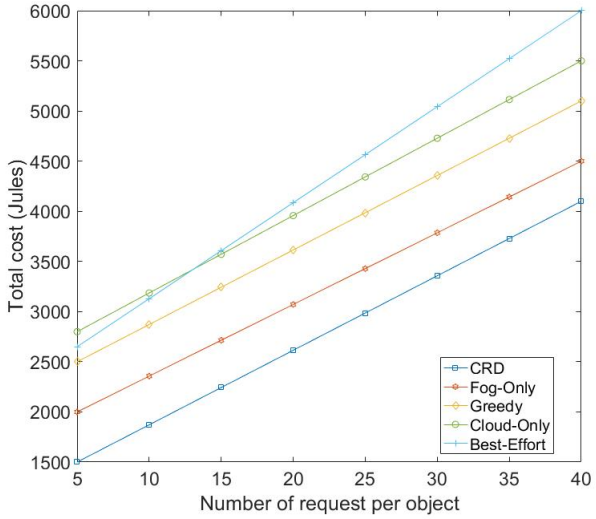


FIGURE 4.1 – Coût total du système en fonction du nombre des requêtes par objet dans un scénario à faible charge.

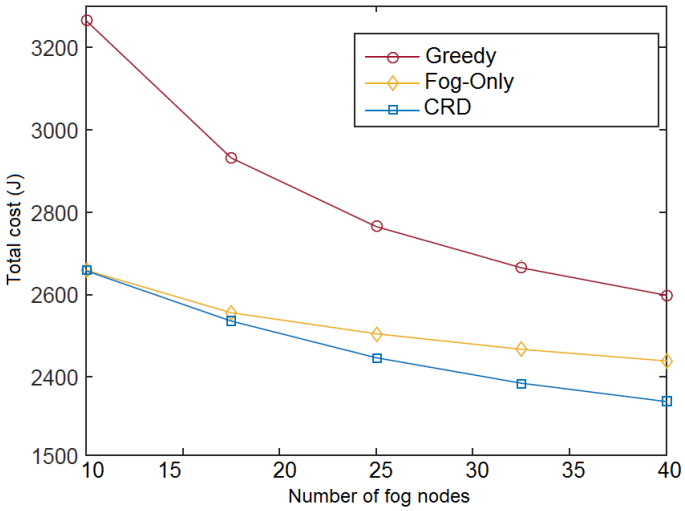
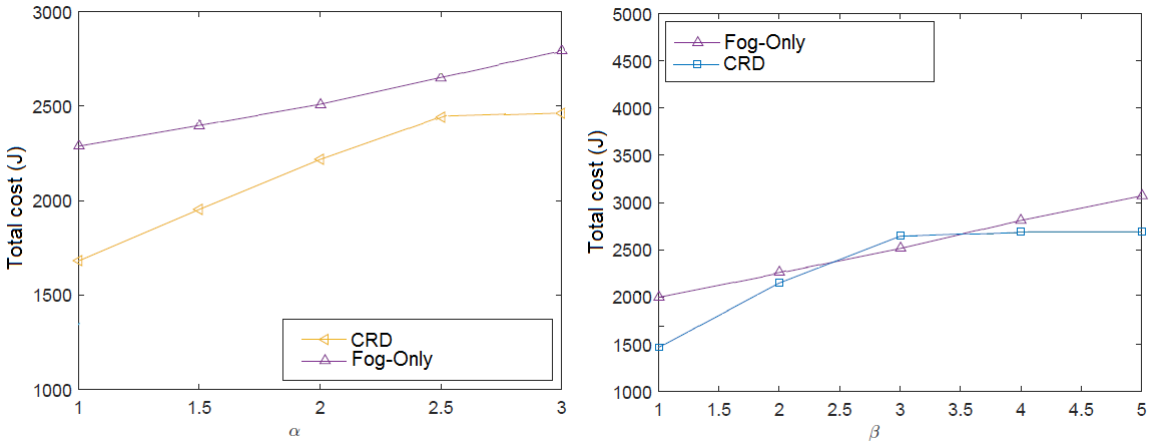


FIGURE 4.2 – Coût total du système en fonction du nombre des nœuds fog dans un scénario à charge lourde.

consommée du système augmente d'une manière exponentielle. Cela est dû au fait que l'énergie consommée par les nœuds fog (même à l'état inactif) reste élevée.

Dans la figure 4.3a, nous étudions le coût du système en fonction des poids α et β sur le coût d'utilisation des nœuds fog et le cloud. Lorsque α est grand, la majorité des requêtes est susceptible d'être traitée par les nœuds fog . Plus important encore, il a été démontré que CRD était plus flexible que Fog-Only. En outre, lorsque α devient plus grand, les résultats de Fog-Only sont proches de ceux de CRD. Cependant, en cas d'augmentation de β, les performances de CRD dépassent celles de Fog-Only.

Avec la croissance de α, davantage de requêtes doivent être traitées dans le fog de manière à ce que les traitements locaux et distants puissent être pleinement exploités en vue de s'adapter à une charge importante. Nous pouvons voir que le coût augmente dans le scénario à forte charge, en raison de l'augmentation du traitement dans le fog (accès aux nœuds fog). Lorsque α approche 2.4, le comportement de CRD, contrairement à Fog-Only, continue de croître, car le temps de traitement dans les nœuds fog est beaucoup plus long à cause de l'alourdissement de la charge des ressources fog. Nous pouvons également observer dans la figure 4.3b que le coût obtenu par



(a) (b)

FIGURE 4.3 – Le coût du système en fonction des poids α et β dans un scénario à faible charge.

le Fog-Only augmente avec l'augmentation de β, ce qui est conforme à notre hypothèse. En revanche, le comportement des CRD est très différent de celui-ci : avec moins de requêtes traitées localement (β augmente), le coût n'augmente pas. En effet, les systèmes CRD peuvent tirer parti des nœuds fog et du cloud pour rechercher des opportunités de distribution via les ressources distantes.

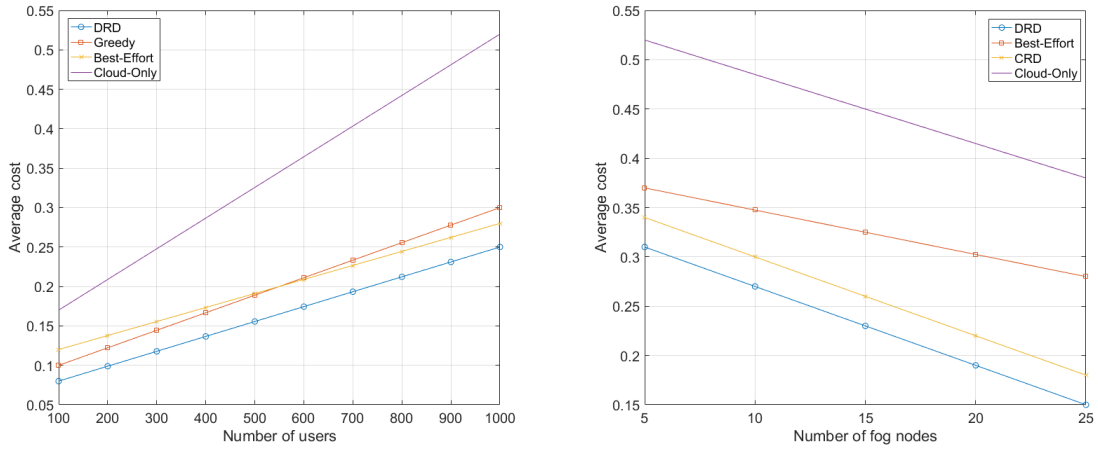
4.4.3 Analyse des performances de DRD

Dans cette section, nous allons étudier l'efficacité de l'algorithme décentralisé, en comparant à des algorithmes de base et à trois approches inspirées de l'état de l'art [86], [58] et [59]. Pour ce faire, nous allons étudier l'impact du nombre d'utilisateurs et de nœuds fog sur les performances du système en termes de délai, d'énergie consommée et de coût total.

Dans la figure 4.4a, nous étudierons l'impact du nombre d'utilisateurs et des nœuds fog sur les performances de DRD. En règle générale, un plus grand nombre d'utilisateurs peuvent entraîner un délai de traitement et une consommation d'énergie accrue, ce que nous pouvons facilement observer sur la courbe de la figure 4.4a. L'algorithme Best-Effort sollicite tous les serveurs, ce qui génère une consommation d'énergie accrue et un coût plus élevé. Quand N = 600, l'algorithme Greedy a des valeurs de coût proches de celles du Best-Effort. La raison de cette légère différence tient à ce que, lorsqu'il y a moins d'utilisateurs (faible charge de travail), les objets traitent leurs requêtes dans les nœuds fog présents dans leurs voisinages, impliquant ainsi un délai et une consommation de l'énergie faible. Dans cette figure, nous remarquons également que l'approche de Cloud-Only produit le coût le plus élevé ; cela s'explique par le délai de transmission considéré dans la fonction coût.

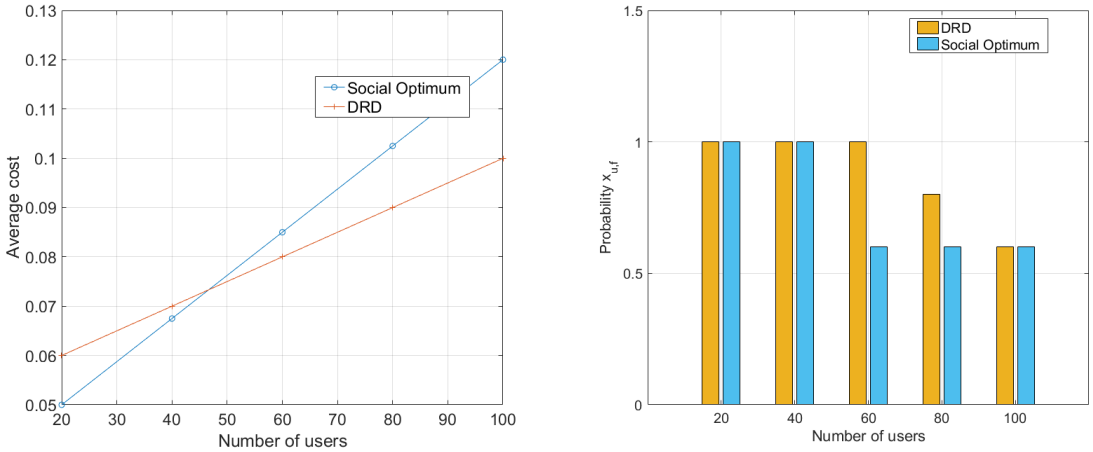
Dans la figure 4.4b, nous évaluons le coût en fonction du nombre de nœuds fog avec 1000 utilisateurs, pour $\lambda_u = 6(requêtes/s)$. La figure montre que notre algorithme DRD a le coût le plus bas parmi les quatre algorithmes, ce qui permet de vérifier l'efficacité de notre solution. Les algorithmes Best-Effort et Greedy détiennent un coût inférieur à celui de l'algorithme Cloud-Only, car ce dernier possède le délai de service le plus élevé, alors que les différences de délai entre les autres algorithmes sont minimes. Notons que pour l'ensemble de paramètres ci-dessus, le nombre d'utilisateurs ne satisfait pas la condition suffisante du théorème 2. Cependant, nos simulations montrent que la méthode DRD converge même pour des instances plus importantes du problème.

Nous comparons à présent l'algorithme DRD égoïste avec une solution optimale sociale (Social Optimum) proposée dans [86], en vue d'analyser les performances du comportement égoïste des utilisateurs. Nous analysons les performances des deux solutions, en fonction du nombre d'utilisateurs. La figure 4.5a montre que dans le cas d'un faible nombre d'utilisateurs, les deux solutions coïncident, car les ressources de calcul des nœuds fog peuvent prendre en charge toutes



(a) (b)

FIGURE 4.4 – Le coût du système par rapport au nombre d'utilisateurs et au nombre de nœuds fog.



(a) (b)

FIGURE 4.5 – Social Optimum vs. DRD.

les requêtes. Notons également que contrairement à la solution non coopérative, les utilisateurs basculent plus tôt leurs requêtes vers le cloud, car ils n'agissent pas socialement, ce qui pourrait représente la meilleure solution quand N > 50. Le comportement d'un utilisateur égoïste consiste à décharger le plus possible son calcul sur les nœuds fog, tout en essayant de prédire la stratégie des autres utilisateurs, ce qui est démontré dans la figure 4.5b. Cependant, cette stratégie implique que les performances des utilisateurs se dégradent à mesure que leur nombre augmente quand N > 100.

Dans la figure 4.6, nous comparons notre algorithme avec deux algorithmes inspirés de la méthode d'approximation convexe (SCAM) proposée dans [58] et l'algorithme d'offloading dynamique par calcul dynamique basé sur l'optimisation de Lyapunov (LODCO) proposé dans [59]. En faisant varier le nombre d'utilisateurs, nous pouvons constater que l'algorithme que nous présentons offre de meilleures performances du système. Cela est dû au fait qu'avec l'algorithme proposé, le point GNE peut être atteint plus rapidement, ce qui constitue une solution mutuellement satisfaisante que personne n'est incité à écarter.

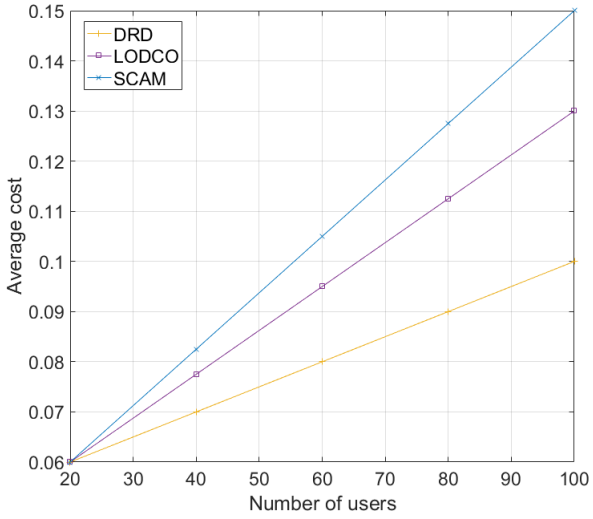


FIGURE 4.6 – Comparaison des performances des DRD, LODCO et SCAM en fonction du nombre d'utilisateurs.

4.4.4 Comparaison des performances de CRD et DRD

Rappelons que l'algorithme CRD est basé sur une stratégie centralisée de résolution d'un graphe coupe-min, alors que l'algorithme DRD repose sur une stratégie mixte statique basée sur la théorie des jeux. Dans cette section, nous allons comparer la solution centralisée et la solution décentralisée.

Afin d'évaluer l'efficacité de calcul de l'algorithme DRD, nous avons mesuré le temps nécessaire pour calculer une solution par la méthode CRD ainsi que pour calculer une solution optimale par la méthode DRD. Dans la figure 4.7, nous observons que, bien que l'algorithme CRD repose sur la résolution d'un graphe, le temps nécessaire au calcul de la solution optimale augmente de manière exponentielle en fonction du nombre d'utilisateurs ; pour 600 utilisateurs, celui-ci est déjà difficile à calculer. Contrairement à CRD, l'algorithme DRD peut atteindre un équilibre rapidement, même dans le cas d'un nombre d'utilisateurs élevé. Par conséquent, les utilisateurs bénéficient en moyenne de la solution centralisée, cette dernière offrant une meilleure stratégie pour réduire les coûts de chacun d'entre eux tout en respectant toutes les contraintes. Notre algorithme CRD est le plus économe en énergie. Toutefois, lorsque le nombre d'utilisateurs est davantage élevé ou que le nombre de nœuds fog plus est faible, DRD est plus efficace.

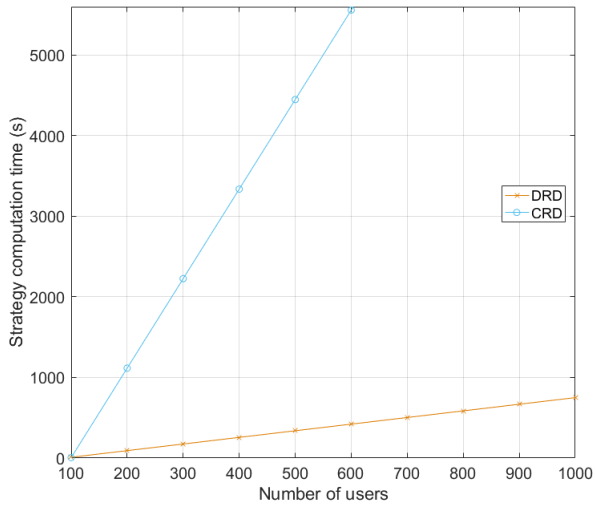


FIGURE 4.7 – Temps d'exécution de CRD et DRD en fonction du nombre d'utilisateurs.

4.5 Conclusion

Dans ce chapitre, nous avons mené une étude formelle du problème de mise en œuvre d'une application IoT à contenu statique dans un système fog/cloud. Nous avons modélisé les principaux défis d'allocation de ressources et de distribution de requêtes de services optimaux. Pour résoudre cette problématique, nous avons d'abord proposé une solution centralisée basée sur la résolution itérative d'une série de coupes minimales de graphes. Nous avons ensuite présenté une solution décentralisée basée sur la théorie des jeux, qui met l'accent sur le comportement égoïste des utilisateurs. Chaque utilisateur essaie de minimiser le coût de traitement de ses requêtes en considérant la stratégie des autres. Les performances des deux algorithmes proposés sont confirmées par de nombreuses simulations.

Alors que l'intersection des applications IoT et fog computing attire de plus en plus l'attention, les solutions fournies dans le présent chapitre peuvent servir de base à de futures explorations et encourager à poursuivre cette direction. Cependant, plusieurs questions de recherche demeurent encore en suspens, notamment le développement de la collaboration entre les nœuds fog, ou une gestion d'un flux de données dynamique. Dans le chapitre 5, nous allons nous intéresser à résoudre notre problématique dans le contexte des applications IoT à contenu dynamique. Cette démarche revient à proposer des algorithmes en ligne où les requêtes nécessitent un traitement en temps réel.

Chapitre 5

Solutions Fog pour les applications IoT avec contenu dynamique

5.1 Introduction

Il existe des applications IoT dont les sources de données sont principalement les objets et dont le contenu change rapidement, telles que les applications de surveillance vidéo à domicile - dans une application de vidéo surveillance, l'image ou la vidéo est mise à jour en permanence. Dans le cas d'un système fog/cloud, les données générées par la surveillance vidéo sont hébergées dans des nœuds fog, et les utilisateurs accèdent à ces données à distance (via le réseau). En raison de la limitation des ressources du fog et de la nature dynamique des réseaux, l'utilisation des ressources fog doit être soigneusement conçue et optimisée, conformément aux décisions de distribution des requêtes de services et à l'allocation des ressources aux utilisateurs finaux. Aussi, l'implémentation d'un système coopératif fog/cloud efficace est nécessaire pour répondre aux besoins des applications IoT à contenu dynamique.

Dans le but de minimiser le coût total du déploiement des applications IoT à contenu dynamique dans un système fog/cloud, nous avons proposé trois stratégies en ligne d'allocation des ressources d'un système fog/cloud et de distribution de requêtes de service.

— Dans notre première solution, nous avons proposé une approche centralisée basée sur les algorithmes génétiques, afin de trouver la meilleure affectation requête-nœud fog pour une consommation d'énergie efficace et un délai optimal. Nous proposons également deux algorithmes heuristiques qui s'inspirent de la première solution pour pallier le problème du temps d'exécution dans les algorithmes génétiques.

— Dans notre deuxième solution, nous avons présenté deux algorithmes de collaboration : une collaboration verticale (entre les nœuds fog et le cloud) et une collaboration horizontale (les nœuds fog entre eux), avec comme objectif de minimiser le coût total de l'exécution d'applications IoT. Ici, une nouvelle conception de système est proposée, dans laquelle différents caractéristiques et théorèmes de file d'attente sont appliqués.

— Dans la troisième solution, nous avons remodelisé notre problématique sous forme de Bandit manchot. Nous avons ensuite proposé un algorithme distribué basé sur le Machine Learning. Chaque objet décide où traiter ses requêtes, et grâce à l'algorithme d'apprentissage celui-ci peut améliorer cette décision.

Dans la section suivante, nous allons présenter notre première approche centralisée basée sur les algorithmes génétiques.

5.2 Solution implémentée au niveau du cloud

Dans cette partie, nous viserons à réduire la consommation d'énergie dans un système fog/cloud tout en considérant le délai dans le contexte des applications IoT sensibles à la latence.

Nous considérerons l'architecture présentée dans le chapitre 3, avec O représentant l'ensemble des objets IoT, F l'ensemble des nœuds fog et c le serveur du cloud. Nous définissons les mesures de performance de notre système en utilisant le modèle basé sur les flux de données déjà présenté dans 3.3.1 et 3.3.2.

Lorsqu'un objet o transmet une requête N au nœud fog f, l'énergie consommée pour traiter N peut être divisée en trois valeurs : (a) l'énergie de transmission $\mathrm{E}^{Tr}_{o,f}$ décrite en (3.6) ; (b) $\mathrm{E}^{Pr}_{o,f}$ l'énergie consommée pour calculer et stocker N décrite dans (3.7) ; enfin (c) l'énergie consommée pour la transmission d'un nœud de fog vers le cloud référencée dans (3.8), notée $\mathrm{E}^{Fwd}_{o,f}$ si N doit être traitée dans le cloud.

De même, la latence d'une requête N traitée dans un nœud fog f est modélisé comme la somme de (a) la latence de transmission donnée par l'équation (3.1) ($\mathrm{D}^{Tr}_{o,f}$), (b) la latence de traitement définie dans (3.2) ($\mathrm{D}^{Pr}_{o,f}$), et dans le cas où le nœud fog transmet la requête à traiter dans le cloud, (c) le délai consommé du fait de la transmission de la requête au cloud référencé dans ($\mathrm{D}^{Fwd}_{o,f}$).

5.2.1 Modélisation du problème

Nous modélisons le système fog/cloud comme un graphe non orienté G(V, E), où l'ensemble V contient l'ensemble des nœuds fog F, l'ensemble des objets O et le cloud c. Chaque nœud dans F est un candidat à assigner à un nœud dans O ; cette relation est représentée par un lien. Si un lien $e = (i, j) \in \mathrm{E}$ existe entre un objet $i \in \mathrm{O}$ et un nœud fog $j \in \mathrm{F}$, la pondération du lien représente la quantité d'énergie consommée. Nous notons que l'énergie consommée et la latence pour traiter une requête dépendent non seulement de la distance entre i et j, mais aussi de la charge de travail dans le nœud fog j.

Notre problématique peut être modélisée comme suit : étant donné notre graphe G(V, E), chaque nœud fog est sélectionné par un sous-ensemble d'objets en vue de traiter leurs requêtes. Si un objet i sélectionne un nœud fog j, cela a un effet sur l'énergie consommée et le délai de traitement de tous les nœuds fog. Soit $p_{i,j}$, notée p_e, la quantité d'énergie consommée par j pour servir i. Considérant $d_{i,j}$ le délai pris par j pour satisfaire la requête de i, nous cherchons à déterminer un ensemble d'objets à associer à un nœud fog pour fournir un service tel que la consommation totale d'énergie soit minimisée et le délai total est inférieur ou égal à un seuil donné. Ce dernier peut être décrit sous forme de problème de programmation binaire en nombres entiers avec des contraintes. Nous pouvons exprimer notre problème comme suit :

$$\min \sum_{e=0}^{|\mathrm{E}|} p_e * x_e \tag{5.1}$$

où $\mathrm{X} = \{x_e |\ e = 1, \cdots, |\mathrm{E}|\} \in \{0,1\}^{|\mathrm{E}|}$, x_e est une variable de décision binaire pour indiquer que la requête de l'objet i sera traitée dans le nœud fog j.

Ce problème est soumis aux contraintes suivantes :

— S'assurer que chaque objet est affecté à un seul nœud fog :

$$\forall i \in \mathrm{O} : \sum_{j=0}^{m} x_{i,j} = 1 \tag{5.2}$$

— Le fog sélectionné doit respecter la contrainte du délai :

$$\sum_{i=0}^{n} d_{i,j} * x_e \leq \mathrm{D}_j \quad \forall i \in \mathrm{O}, \forall j \in \mathrm{F}, (i,j) = e \in \mathrm{E} \tag{5.3}$$

où D_j est le seuil de délai du nœud fog j, défini comme suit :

$$\mathrm{D}_j = \min_{f \in \mathrm{F}} \max_{o \in \mathrm{O}} \{d_{o,f}\} \tag{5.4}$$

où $d_{o,f}$ indique le délai traitement de la requête de o dans nœud f

— Nous supposons que la capacité d'utilisation d'un nœud fog est limitée. De plus, on note par $\sum_{i=0}^{|O|} K_i * x_{i,j}$ la charge de travail affectée au nœud fog j. Soit χ_j^{max} la capacité maximale de calcul d'un nœud fog $j \in F$, nous avons :

$$\sum_{i=0}^{|O|} K_i \times x_{i,j} < \chi_j^{max} \forall j \in F \tag{5.5}$$

— Étant donné λ_j le taux d'arrivée maximal du nœud fog $j \in F$. Le nombre de requêtes entrant dans j ne doit pas dépasser ses ressources de calcul. Ainsi, nous avons :

$$\sum_{i=0}^{|O|} \frac{1}{D_{i,j}} \times x_{i,j} + \pi(j) < \lambda_j \tag{5.6}$$

le taux d'arrivée d'un nœud fog j lors du traitement exclusif d'une requête K_i (appartenant à l'objet i) peut être approximé comme l'inverse de sa latence $\frac{1}{D_{i,j}}$. Dans le cas d'objets multiples, il s'agit de l'inverse de la somme de leurs latences $\frac{1}{\sum_{i=0}^{|O|}(D_{i,j})}$. Toutefois, des frais généraux supplémentaires (overhead) seront probablement engendrés, en raison d'interférences entre les requêtes simultanées notées $\pi(j)$. Par conséquent, nous ne devrions placer une requête sur un nœud fog que si celui-ci ne reçoit pas un débit d'entrée supérieur à ce λ_j.

Notre solution doit trouver une affectation optimale $(x_e)_{e \in E}$ qui minimise la consommation totale d'énergie du système fog/cloud tout en respectant les contraintes décrites ci-dessus. La contrainte (5.2) est introduite pour garantir que chaque objet sera affecté à un seul nœud fog. La deuxième contrainte de (5.3) stipule que le temps de traitement d'une requête doit être inférieur au certain délai acceptable donné par D_j. Les troisième et quatrième contraintes décrites en (5.5) et (5.6) ont pour but de garantir que les nœuds fog ne dépassent pas leur charge de travail maximale.

5.2.2 Algorithmes d'assignation adaptatifs

L'algorithme proposé consiste à appliquer une stratégie de consolidation de charge, tout en mettant le maximum de nœuds fog à l'état inactif. Cette approche nécessite de gérer la consommation d'énergie et de trouver un compromis avec la latence, car cette dernière peut se dégrader en raison de la consolidation de la charge.

Dans un premier temps, nous allons proposer une heuristique d'assignation pour résoudre ce problème. Ensuite, nous étendrons cette solution au scénario dynamique en proposant deux heuristiques d'affectation. Les approches heuristiques contribuent à améliorer les performances de l'heuristique de base.

Algorithme d'optimisation génétique : Notre algorithme génétique est inspiré de [87] et il est décrit comme suit : chaque objet et chaque nœud fog est codé par son index respectif. Un chromosome correspond à une distribution de toutes les assignations possibles d'objets et de nœuds fog. L'aptitude (la fonction fitness) d'un chromosome représente la quantité totale d'énergie consommée par les nœuds fog utilisés dans la solution, qui respecte les contraintes décrites précédemment. L'algorithme IGA (Improved Genetic Algorithm) commence par coder les objets et les nœuds fog, de manière à pouvoir générer une population comprenant tous les gènes (toutes les assignations possibles objets-nœud fog qui respectent les contraintes (5.2)- (5.6)). Si la meilleure solution n'est pas remplacée après $l \leq |F| \times |O|^2$ itérations, le processus se termine ; la meilleure solution trouvée est choisie comme solution optimale. La sélection des membres parents s'effectue à l'aide de la méthode de la roulette. Un processus de croisement est ensuite appliqué aux parents. Nous remplaçons un gène dans les nouveaux membres avec une probabilité de mutation définie.

Ici, l'opérateur de croisement traditionnel est remplacé par un nouvel opérateur de génération. Cela permet à notre algorithme d'explorer de nouvelles solutions tout en conservant une partie des solutions précédemment découvertes. Nous notons également une amélioration du temps

d'exécution du processus. Le processus principal est décrit dans l'algorithme 4 : nous désignons l'ensemble des liens similaires dans les membres parents sous la forme E_{fix}, et l'ensemble des différents liens sous la forme E_{free}. Nous supprimons chaque lien de E_{free} et formons une nouvelle solution possible (brouillon)M_{draft} en prenant l'union de E_{free} et de E_{fix}. Nous calculons la fonction fitness de la solution actuelle. Si la solution possible indique la meilleure valeur de la fonction fitness, nous supprimons le chromosome de E_{free}. Nous répétons cela jusqu'à ce que le nombre de gènes (nombre d'éléments) dans l'ensemble union de E_{fix} et E_{free} soit égal à N. Le taux de croisement donne la probabilité qu'une paire de parents subisse un croisement. De plus, si une opération de croisement génère un descendant qui dépasse la longueur maximale autorisée du génome, le croisement ne se produit pas. Les parents qui ne se croisent pas sont transformés en progéniture inchangée. Cependant, ils peuvent toujours subir une mutation.

Algorithm 4 Opérateur de génération

Entrées : $member_1$, $member_2$
Sortie : $member_{new}$
$E_{fix} \leftarrow$ Même liens dans $member_1$, $member_2$
$E_{free} \leftarrow$ Différents liens dans $member_1$, $member_2$
length= size ($member_1 \cup member_2$)
while $length >$ |O| **do**
  m=0
  for Pour chaque lien $\in E_{free}$ **do**
    $M_{draft} \leftarrow E_{fix} \cup E_{free} - currentlink$
    $m_{draft} \leftarrow fitness(M_{draft})$
    if $m_{draft} \leq m$ **then**
      m $\leftarrow m_{draft}$
      $E_{worst} \leftarrow currentlink$
    end if
    Retirer E_{worst} de E_{free}
    length= length-1
  end for
  $member_{new} \leftarrow E_{fix} \cup E_{free}$
end while

Extensions aux méta-heuristiques GA : Dans la première contribution, nous avons modélisé le problème d'affectation de requêtes sous forme optimisation linéaire en nombres entiers et nous avons résolu le problème grâce à une stratégie basée sur les algorithmes génétiques. L'approche modélise un chromosome sous la forme d'une fonction d'affectation μ de longueur n correspondant au nombre de requêtes générées par les objets. L'algorithme IGA génère une population aléatoire de chromosomes et utilise les croisements et les mutations décrites dans l'algorithme 4 pour créer une nouvelle génération de la population. Notons que nous ne considérons pas les chromosomes qui violent les contraintes (5.2)- (5.6) et nous conservons le chromosome qui donne le meilleur makespan parmi toutes les solutions. L'algorithme se termine si aucune amélioration n'est constatée après les l itérations. Dans ce qui suit, nous introduisons deux extensions de IGA pour résoudre le problème du temps d'exécution et de la mise à grande échelle.

IGA-Incrémental (IGAI). L'objectif est de réduire le temps d'évolution et de mesurer en nombre de générations dans IGA. Ceci est particulièrement utile concernant les problèmes d'optimisation complexes et à grande échelle, qui nécessitent plusieurs générations et un temps d'exécution plus long. Cette approche a conduit à des solutions plus rapides. IGAI est basé sur deux phases.

— Dans la phase I, nous collectons des informations utiles lors de l'exécution d'IGA. Celles-ci comprennent les meilleurs chromosomes réalisables et ceux impossibles à réaliser, qui cor-

respondent aux meilleures et aux pires solutions dans chaque génération d'IGA. Il convient de noter que ces chromosomes ont la meilleure fonction de fitness dans leurs solutions réalisables et irréalisables. Pour assurer la densité des générations, nous devons empêcher la duplication des chromosomes possibles choisis et assurer leur diversité.

— Dans la phase II, nous exécutons IGAI à partir des chromosomes sélectionnés parmi les informations utiles sauvegardées, en plus des chromosomes générés de manière aléatoire.

La prévention de la duplication est réalisée par le calcul de la distance de Hamming entre un chromosome candidat et tous les chromosomes sauvegardés. Si celle-ci est différente de zéro, le chromosome est accepté. Si un chromosome ne se trouve pas dans les meilleurs chromosomes possibles sauvegardés depuis cinq générations, cela peut signifier que la liste des meilleurs chromosomes n'est pas diversifiée.

Pour que IGAI ne se retrouve pas avec une liste des meilleurs chromosomes réalisables trop petite, nous nous assurons d'ajouter des chromosomes réalisables aussi divers que possible à la fin d'IGA. Si aucun chromosome n'est ajouté pendant cinq générations successives, nous commençons par calculer la valeur de fitness de chaque chromosome candidat réalisable et sa distance de Hamming avec les meilleurs chromosomes jusqu'à présent. Nous calculons également la moyenne des valeurs de la fonction de fitness et des distances de Hamming des chromosomes possibles sauvegardés. Nous acceptons le chromosome qui satisfait le test de duplication et si la valeur de fitness est supérieure à la valeur fitness moyenne des meilleurs chromosomes possibles.

IGA-Global (IGAG). IGAG est une autre variante de l'IGA. Nous avons proposé une version plus rapide d'IGAI comportant plusieurs améliorations, notamment une évaluation efficace de la fonction de fitness. IGAG adresse l'état optimal local d'affectation d'une requête après que celle-ci a déjà été affectée à un nœud fog ou d'une requête existante non affectée par IGAI. Nous générons la population initiale en utilisant IGAI et en calculant la fonction fitness de chaque chromosome candidat de la génération, puis nous sélectionnons deux chromosomes candidats. Une sélection de classement élitiste avec échantillonnage stochastique des éléments restants sans remplacement est utilisée [88]. Pour obtenir deux enfants, nous appliquons à ceux-ci le croisement défini précédemment et des mutations. Nous remplaçons ensuite les parents par des enfants, en conservant la meilleure solution de la génération précédente. Enfin, nous sélectionnons la nouvelle génération en fonction des valeurs de la fonction fitness.

Cette démarche devrait apporter une meilleure solution au problème d'optimisation que IGAI, puisque toutes les requêtes sont prises en compte pour l'affectation. Cependant, une telle approche entraîne des coûts élevés, car l'attribution de toutes les requêtes, qu'elles soient nouvelles ou précédemment attribuées, change à chaque intervalle de contrôle.

5.3 Solution implémentée au niveau du fog

Dans cette section, nous proposerons tout d'abord une solution de collaboration fog/cloud, qui combine les ressources du fog et du cloud. Dans cette solution, le fog est la principale capacité de calcul en raison de sa proximité avec les utilisateurs finaux. Les serveurs cloud fournissent la capacité supplémentaire en période de pointe. Ensuite, nous présenterons et analyserons un jeu coopératif entre les nœuds fog. Des serveurs temporaires sont alloués pour libérer la charge totale de la couche fog. Ici, les nœuds fog collaborent pour traiter le total des requêtes.

Nous considérons l'architecture IoT-fog-cloud déjà présentée dans la section 3.2.2 et illustrée dans la figure 5.1, qui comprend : (a) la couche IoT, où sont situés les dispositifs IoT et les utilisateurs finaux; (b) la couche de fog, où sont placés des nœuds de fog; (c) la couche en cloud, où se trouve un DC. Les dispositifs IoT peuvent directement fonctionner et interagir avec la couche de fog en générant et en transmettant des requêtes à un nœud fog dans les mêmes zones fonctionnelles. Une zone fonctionnelle est une région dans laquelle une (ou plusieurs) application(s) IoT est (sont) déployée(s). Par exemple, si des nœuds fog sont placés à proximité d'un réseau intelligent, il convient de supposer que tous les objets IoT situés à proximité sont associés à l'application.

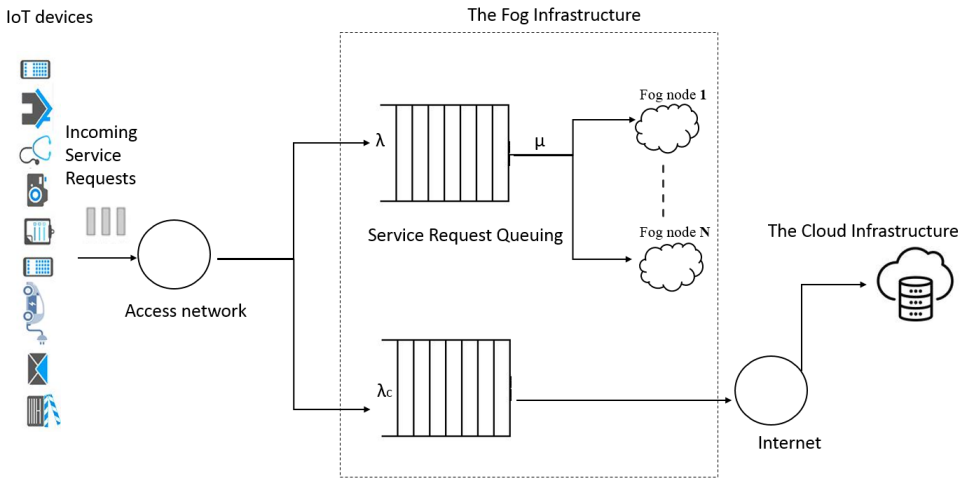


FIGURE 5.1 – L'architecture d'un système mixte fog/cloud

Ils sont considérés comme appartenant à une seule zone fonctionnelle. Le fog computing est déployé pour libérer la charge du cloud et apporter aux objets à proximité une latence de service plus faible. Cela implique que la plupart des requêtes reçues de la couche IoT seront traitées par les nœuds fog. Cependant, le fog peut transférer une requête au cloud lorsque celle-ci nécessite l'intervention de ce dernier, ou si le nœud fog est occupé à traiter d'autres requêtes.

Nous utilisons le modèle basé sur le temps décrit dans 3.3.1 et 3.3.2 pour mesurer les performances de notre système. Dans ce scénario, nous considérons chaque nœud fog comme un système de délai Erlang, au sein duquel un nœud fog $i \in \mathcal{N}$ est représenté par un modèle de file d'attente M/M/n_i, avec n_i serveurs identiques. Les requêtes arrivent selon un processus de Poisson avec un taux $\lambda_i > 0$. Les temps de service sont indépendants et répartis de manière exponentielle avec un taux de $\mu_i > 0$. Lorsqu'une requête entre dans un nœud et que tous les serveurs sont occupés, elle est placée dans une file d'attente à capacité infinie, jusqu'à ce qu'elle soit servie par le premier serveur disponible. Par ailleurs, nous considérons que le DC du cloud héberge des serveurs avec des ressources et des capacités infinies. Nous modélisons le cloud en tant que M/G/∞, disposant de ressources de calcul suffisantes pour traiter les requêtes entrantes avec un temps d'attente des requêtes négligeable.

5.3.1 Solution de collaboration Cloud-Nœud Fog

Solution proposée : Dans notre proposition, nous pouvons décomposer le délai de bout en bout en : (a) un délai de transmission sans fil ; (b) un délai de mise en file d'attente dans le fog ; enfin (c) un délai de calcul dans le nœud fog. Les valeurs de (a) et (c) sont données respectivement par l'équation (3.4) et $\frac{1}{\mu_{fog}}$. Pour satisfaire les exigences de la qualité de service, le temps d'attente de chaque requête doit être limité par une valeur maximale. Nous définissons la probabilité dans l'état stable (ou the steady state probability) qu'une requête doit attendre avant d'être servie dans le nœud fog i grâce à la **Formule de délai d'Erlang C** [89]. Elle peut être exprimée par :

$$\widehat{\mathrm{C}}(n_i, a) = \left(1 + \sum_{y=0}^{n_i-1} \frac{n_i!(1 - a/n_i)}{y!a^{n_i-y}}\right)^{-1} \tag{5.7}$$

$\forall a > 0$, $\forall n_i \in \mathbb{N}$, avec $n_i > a$.

Par conséquent, le temps de séjour d'une requête dans le nœud fog i à l'état stable est défini comme suit :

$$\widehat{\mathrm{R}}(n_i, \lambda_i, \mu_i) = \frac{\widehat{\mathrm{C}}(n_i, a)}{n_i\mu_i - \lambda_i} + \frac{1}{\mu_i} \tag{5.8}$$

$\forall \lambda_i > 0$, $\forall \mu_i > 0$ et $\forall n_i \in \mathbb{N}$, avec $n_i > \lambda_i/\mu_i$

L'extension de $\widehat{\mathrm{C}}(n_i, a)$ au nombre de serveurs (n_i) réel positif (cf. Jagers et van Doorn [90]) est donnée par :

$$\mathrm{C}(n_i, a) = \left(\int_0^\infty a e^{-at} (1+t)^{n_i - 1} t \mathrm{d}t \right)^{-1} \tag{5.9}$$

$\forall a > 0$, $\forall n_i \in \mathbb{R}$, avec $n_i > a$.

Par conséquent, la durée du séjour dans un système de file d'attente artificiel avec un nombre de serveurs non entier peut être exprimée par :

$$\mathrm{R}(n_i, \lambda_i, \mu_i) = \frac{\mathrm{C}(n, a)}{n_i \mu_i - \lambda_i} + \frac{1}{\mu_i} \tag{5.10}$$

$\forall \lambda_i > 0$, $\forall \mu_i > 0$ et $\forall n_i \in \mathbb{R}$, avec $n_i > \lambda_i / \mu_i$

Comme observé dans [90], nous avons $\widehat{\mathrm{C}}(n_i, a) = \mathrm{C}(n_i, a)$ pour tout $n_i \in \mathbb{N}$ et $a \in (0, n_i)$. Par conséquent, nous avons $\widehat{\mathrm{R}}(n_i, \lambda_i, \mu_i) = \mathrm{R}(n_i, \lambda_i, \mu_i)$ pour tout $n_i \in \mathbb{N}$ et $a \in (0, n_i)$. Nous étendrons notre analyse au nombre de serveurs non entier et utiliserons diverses propriétés structurelles de $\widehat{\mathrm{C}}$ et $\widehat{\mathrm{W}}$, également valables pour C et W. Cela nous permettra d'obtenir des résultats intéressants pour la solution de collaboration nœud fog-cloud.

Nous exploitons aussi la **formule de perte d'Erlang B** pour un modèle M/G/n/n défini par :(cf. Jagers and van Doorn [90])

$$\mathrm{B}(n_i, a) = \left(\int_0^\infty a e^{-at} (1+t)^{n_i} \mathrm{d}t \right)^{-1} \tag{5.11}$$

$\forall n_i \in [0, \infty[$ et $a > 0$.

Le lemme suivant montre la connexion entre la formule de délai d'Erlang et la formule de perte d'Erlang pour une valeur entière de n_i. Lorsque $\rho < 1$ et $n_i \in \mathbb{N}$, cette relation est connue [91], mais elle est valable aussi pour tous les $\rho > 0$ et n_i non entiers.

Lemme 8. *Pour une constante $a > 0$, $n_i \in \mathbb{R}$ et $n_i > a$, nous avons :*

$$\mathrm{B}(n_i, a) = \frac{(1-\rho)\mathrm{C}(n_i, a)}{1 - \rho \mathrm{C}(n_i, a)}, \quad \rho > 0, \rho \neq 1.$$

Démonstration. Soit $\mathrm{B}^{-1}(n_i, a) = 1/\mathrm{B}(n_i, a)$, $\mathrm{C}^{-1}(n_i, a) = 1/\mathrm{C}(n_i, a)$ et $\rho = a/n_i$. Dans ce cas, nous avons :

$$\begin{aligned} \mathrm{C}^{-1}(n_i, a) - \rho &= \int_0^\infty a e^{-at} (1+t)^{n_i - 1} t \mathrm{d}t - \rho \\ &= \rho / a = (1-\rho)\mathrm{B}^{-1}(n_i, a) \end{aligned}$$

Ce qui signifie que $\mathrm{C}(n_i, a) = \mathrm{B}(n_i, a)/[1 - \rho(1 - \mathrm{B}(n_i, a)]$. □

De plus, nous définissons une fonction de coût pour la violation du délai d'attente. Une violation de délai survient lorsque le temps d'attente d'une requête dans la file d'un nœud fog excède un certain seuil. Soit D le temps d'attente maximal qu'une requête peut tolérer. Nous notons note par θ_i la pénalisation par un bit d'une requête. Ainsi, le coût de la punition pour une requête de taille x et avec un temps d'attente W peut être exprimé comme suit :

$$Cost(x, \mathrm{W}) = \begin{cases} \theta_i x, \mathrm{W} > \mathrm{D} \\ 0, 0 \leq \mathrm{W} \leq \mathrm{D} \end{cases} \tag{5.12}$$

Puisque la probabilité d'attendre dans une file du nœud fog i est définie par $\mathrm{C}(n_i, a) = Pr\{Wait > 0\}$, la probabilité que le temps d'attente d'une requête dans la file dépasse D est alors exprimée par :

$$\Lambda_{\mathrm{D}}(n_i, a) = Pr\{Wait > \mathrm{D}\} = \mathrm{C}(n_i, a) e^{-(n_i \mu_i - \lambda_i)\mathrm{D}} \tag{5.13}$$

Dans cette solution, une capacité supplémentaire dans le cloud sera allouée aux requêtes des nœuds de fog pendant les périodes de pointe. Quand un objet o génère une requête r, celle-ci est d'abord attribuée au nœud de fog le plus proche pour être traitée. Le nœud de fog enregistre alors

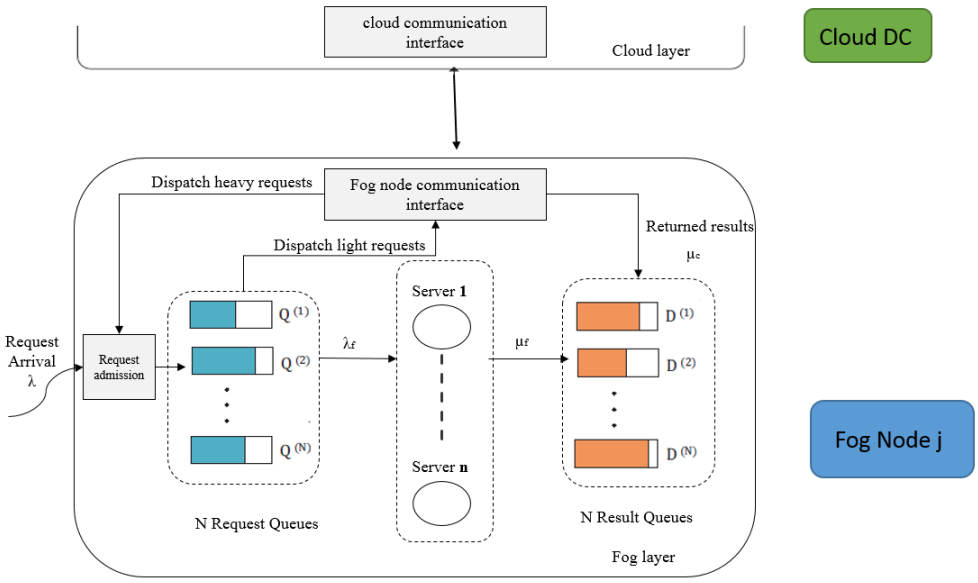


FIGURE 5.2 – Une illustration de l'admission, le traitement et la distribution des requêtes dans une solution de collaboration cloud- nœud fog

son temps d'attente dans la file. Si ce temps atteint D, le nœud fog transmet la requête au cloud pour que celui-ci la traite. La consommation d'énergie et les délais dus au problème de compromis du traitement de la requête sont pris en compte, et un problème d'optimisation est modélisé afin de minimiser l'overhead. La solution de collaboration entre le cloud et un nœud fog est illustré dans la figure 5.2.

Problème d'optimisation du overhead : Nous considérons qu'un nœud fog $i \in \mathcal{N}$ a n_i serveurs. Comme indiqué précédemment, une partie des requêtes de i doivent être transmises au cloud pour être traitées. Notons par $p_{cld}(\mathrm{D})$ la probabilité à l'état stable qu'une requête soit affectée au cloud, à savoir la fraction de requêtes dont le temps d'attente dépasse D. En général, $p_{cld}(\mathrm{D})$ est inférieure à $\Lambda_{\mathrm{D}}(n_i, a)$, où $\Lambda_{\mathrm{D}}(n_i, a)$ représente toutes les requêtes dans la file d'attente, que celles-ci aient dépassé le délai d'attente ou non. Nous définissons $p_{cld}(\mathrm{D})$ comme suit [92] :

$$p_{cld}(\mathrm{D}) = \frac{(1-\rho)(1-\Lambda_{\mathrm{D}}(n_i,a))}{1-\rho(1-\Lambda_{\mathrm{D}}(n_i,a))} \tag{5.14}$$

Dans une unité de temps, nous avons $\lambda_i p_{cld}(\mathrm{D})$ pour cent des requêtes de i qui doivent être traitées dans le cloud. Dans ce cas, nous pouvons définir O_{cld} le total des frais généraux (ou overhead) pour le traitement des requêtes du nœud fog i dans le cloud.

$$\mathrm{O}_{cld} = \lambda_i p_{cld}(\mathrm{D}) \left[\frac{1}{\mu_{wn}} (\epsilon \mathrm{P}_{i,c}) + \frac{1}{\mu_c} (\gamma + \delta \mathrm{P}_{cld}) \right] \tag{5.15}$$

où $\mathrm{P}_{i,c}$ est la puissance de transmission du nœud fog i au cloud ; celle-ci est déterminée par la station de base sans fil.

Parmi toutes les requêtes entrantes dans le nœud i, $\lambda_i p_{cld}(\mathrm{D})$ pour cent ne seront pas traitées localement. Par conséquent, la consommation d'énergie due au traitement des requêtes dans le nœud fog i est définie par $\mathrm{P}_i = \rho(1-p_{cld}(\mathrm{D}))\kappa s^{\phi} + \mathrm{P}^*$. L'overhead des requêtes traitées par le nœud fog i en une unité de temps est exprimé par :

$$\mathrm{O}_i = n_i(\chi + \delta \mathrm{P}_i + \frac{1}{\mu_{wl}}(\tau \mathrm{P}_{o,i})) \tag{5.16}$$

où $\mathrm{P}_{o,i}$ est la puissance d'énergie nécessaire pour transmettre la requête de l'objet o au nœud fog i.

Le lemme suivant donne le coût attendu pour la violation du délai d'une requête.

Lemme 9. *Le coût prévu pour une violation du délai d'une requête est égal à* 0.

Démonstration. Étant donné que le temps d'attente d'une requête W est toujours inférieur ou égal à D, le coût attendu pour la violation concernant la requête avec une exigence d'exécution x est égal à 0. Soit x (la taille des requêtes) une variable aléatoire exponentielle avec la moyenne $\overline{x}$ et une fonction de densité de probabilité (dfp) est donnée par $f_x(y) = \frac{1}{\overline{x}} e^{-y/\overline{x}}$. Par conséquent, $\theta_i x$ est également une variable aléatoire. Le coût attendu pour la violation du délai d'une requête est donné par :

$$\begin{aligned}\int_0^\infty f_x(y)\mathrm{C}ost(x,y)\mathrm{d}y &= \int_0^\infty \frac{1}{\overline{x}} e^{-y/\overline{x}} 0 \mathrm{d}y \\ &= 0\end{aligned}$$

□

Selon (5.15) et (5.16), il est possible d'exprimer le total overhead du nœud fog i en termes d'énergie et de délai comme suit :

$$\begin{aligned}\mathrm{O}verhead_i &= \mathrm{V}iolation\_\mathrm{C}ost + \mathrm{O}_{cld} + \mathrm{O}_i \\ &= 0 + \lambda_i p_{cld}(\mathrm{D}) \left[\frac{1}{\mu_{wn}} (\epsilon \mathrm{P}_{i,c}) + \frac{1}{\mu_c} (\gamma + \delta \mathrm{P}_{cld}) \right] \\ &+ n_i (\chi + \delta \mathrm{P}_i + \frac{1}{\mu_{wl}} (\tau \mathrm{P}_{o,i}))\end{aligned} \tag{5.17}$$

où $\mathrm{V}iolation\_\mathrm{C}ost = \lambda_i \mathrm{C}ost(x, \mathrm{W})$, et il est égal à 0 selon le lemme 9.

Notre objectif est de choisir une solution optimale de sorte que le coût total $\mathrm{O}verhead_i$ d'un nœud fog i soit minimisé. Comme le montre la figure 5.3, l'overhead total d'un nœud fog varie en fonction de la taille du serveur et de la vitesse d'exécution du nœud fog. Nous avons donc fixé λ_i, $\overline{x}$, P_*, γ, δ, χ, ϵ et D et avons tenté de déterminer s_i la vitesse optimale du serveur et n_i le nombre optimal de serveurs du nœud fog i.

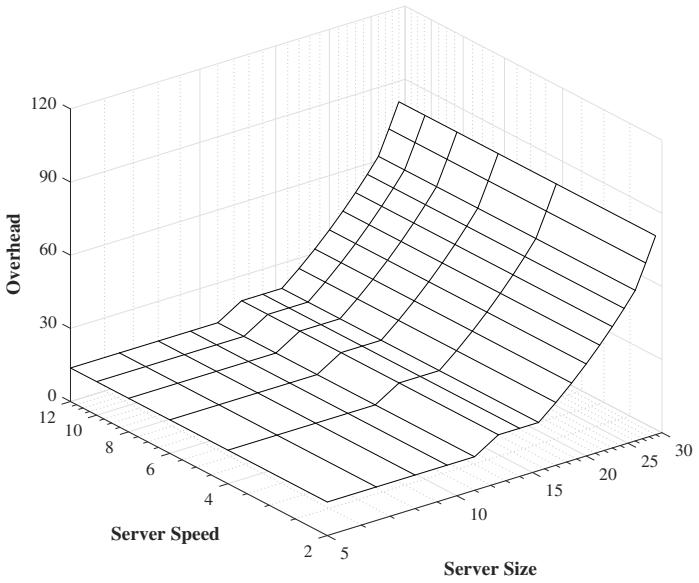


FIGURE 5.3 – La variation $\mathrm{O}verhead_i$ d'un nœud fog i en fonction de s et n

Dans ce qui suit, nous développerons dans un premier temps une méthode analytique permettant de résoudre notre problème d'optimisation (pour trouver n et s). Nous supposerons que n_i et s sont des variables continues, et nous résoudrons analytiquement les fonctions $\frac{\partial \mathrm{O}verhead}{\partial n_i} =$

0 et $\frac{\partial O\mathit{verhead}}{\partial s} = 0$. Dans un second temps, nous proposerons un algorithme de résolution basé sur l'analyse analytique.

Pour davantage de commodité, nous réécrivons $p_{cld}(\mathrm{D}) \approx \frac{(1-\rho)\mathrm{K}_1}{1-\rho\mathrm{K}_1}$, où $\mathrm{K}_1 = 1 - \mathrm{C}_\mathrm{D}(n_i, a)) = 1 - \left(\int_0^\infty ae^{-a\mathrm{D}}(1+\mathrm{D})^{n_i-1}\mathrm{DdD}\right)^{-1}$. En nous basant sur la formulation de $p_{cld}(\mathrm{D})$, nous résoudrons notre problème de minimisation.

Recherche du nombre optimal de serveurs : Étant donné λ_f, $\overline{x}$, P_*, γ, δ, χ, ϵ, D, ϕ, κ et s, notre objectif est de trouver n_i de manière à minimiser $\mathrm{O}\mathit{verhead}_i$ du nœud fog i.

$$\frac{\partial \mathrm{O}\mathit{verhead}}{\partial n_i} = \frac{\partial \mathrm{O}_{cld}}{\partial n_i} + \frac{\partial \mathrm{O}_i}{\partial n_i} = 0 \tag{5.18}$$

où

$$\frac{\partial \mathrm{O}_{cld}}{\partial n_i} = (\lambda_i \frac{1}{\mu_{wn}} \epsilon \mathrm{P}_{i,c} + \frac{1}{\mu_c}(\gamma + \delta(\kappa' s'^{\phi} + \mathrm{P}^{**}))) \frac{\partial p_{cld}(\mathrm{D})}{\partial n_i}$$

et

$$\frac{\partial \mathrm{O}_i}{\partial n_i} = \chi + \delta \mathrm{P}^* - \lambda_i \delta \rho \kappa s^{\phi} \frac{\partial p_{cld}(\mathrm{D})}{\partial n_i} + \frac{1}{\mu_{wl}} \tau \mathrm{P}_{o,i}$$

Puisque

$$\frac{\partial \mathrm{B}(n_i, a)}{\partial n_i} = -\frac{\mathrm{B}(n_i, a)^2}{2\sqrt{n_i}}\big(a_0 - a2\big) - \frac{n_i + a}{2n_i}\mathrm{B}(n_i, a)\{\frac{x}{a} - 1 + \mathrm{B}(n_i, a)\}$$

Le lemme 9 fournit la relation suivante :

$$\mathrm{C}(n_i, a) = \frac{\mathrm{B}(n_i, a)}{\big[1 - \rho(1 - \mathrm{B}(n_i, a))\big]}$$

Nous avons,

$$\frac{\partial \rho}{\partial n_i} = \frac{\lambda_i \bar{x}}{n_i^2 s} = -\frac{\rho}{n_i}$$

Ainsi,

$$\frac{\partial \mathrm{K}_1}{\partial n_i} = -\mu_i \mathrm{DK}_1$$

De plus, nous avons

$$\frac{\partial p_{cld}(\mathrm{D})}{\partial n_i} = \frac{1}{\rho \mathrm{K}_1^2}\Big(\frac{\rho}{n_i} + (\rho - 1)\mu_i \mathrm{DK}_1 \frac{(1+\rho)}{2n_i} + (1-\rho)\mathrm{K}_1(\ln \rho)\Big).$$

Bien que ne disposions que d'une expression semi-fermée (semi-closed) de n_i, nous pouvons obtenir une solution numérique à (5.18). En variant n_i, nous remarquons que $\partial \mathrm{O}\mathit{verhead}_i/\partial n_i$ ne diminue et n'augmente pas. Nous pouvons alors trouver n_i tel que $\mathrm{O}\mathit{verhead}_i$ décrit puis utiliser la méthode de bissection standard. S'il y a plus d'une valeur minimale du overhead, ces dernières sont comparées et le minimum est sélectionné. Lors de l'utilisation de la méthode de bissection pour trouver un point extrême, la précision de l'itération est définie sur une valeur unifiée 10^{-10}.

Dans la figure 5.4, nous démontrons le coût total en une unité de temps en fonction de n_i et λ_i, où $s = 1$, $\bar{x} = 1$; les autres paramètres sont les mêmes que ceux utilisés dans la figure 5.3. Nous remarquons qu'il existe un choix optimal de n_i, de sorte que la surcharge totale est réduite au minimum. En utilisant la méthode analytique, les valeurs optimales de n_i telles que $\partial \mathrm{O}\mathit{verhead}_i/\partial n_i = 0$ sont 10,88582, 11,8587, 12,8590 et 13,859 respectivement avec $\lambda_i = 0,3\mathrm{M}\mathit{bps}, 0,55\mathrm{M}\mathit{bps}, 0,85\mathrm{M}\mathit{bps}$, et $1,1\mathrm{M}\mathit{bps}$. Lorsque n_i le nombre de serveurs dans le nœud fog est inférieur à la valeur optimale, celui-ci doit envoyer au cloud davantage de requêtes avec un temps d'attente égal à D. Par conséquent, des coûts supplémentaires apparaissent, dépassant même les gains en termes de délai ou d'énergie en utilisant le fog computing. À mesure que n_i

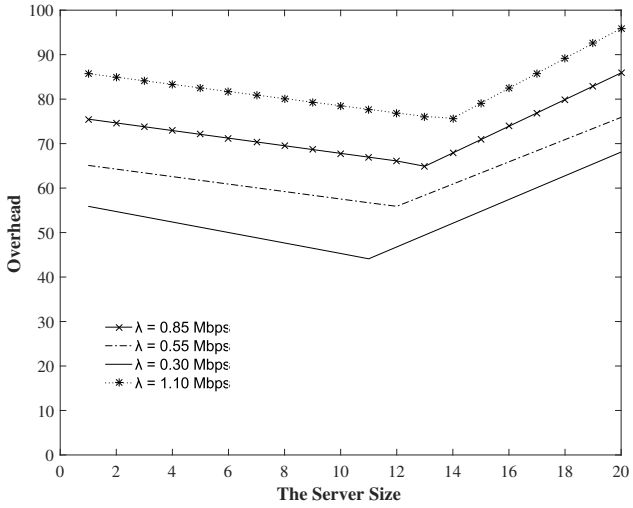
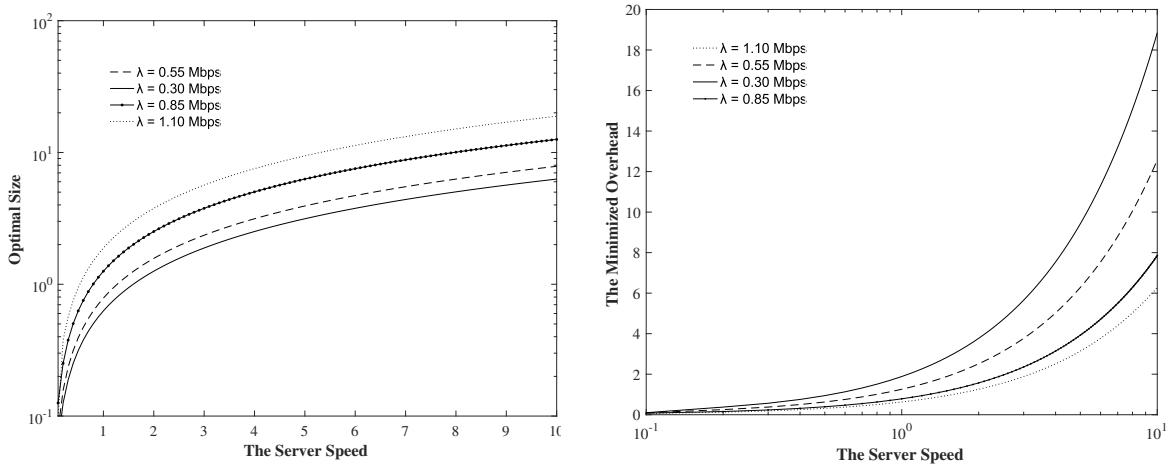


FIGURE 5.4 – La surcharge d'un nœud fog versus n et λ



(a) Nombre de serveurs optimal n_i par rapport à s et λ_i

(b) $Overhead_i$ minimal par rapport à s et λ_i

FIGURE 5.5 – Nombre de serveurs optimal n_i et surcharge minimale $Overhead_i$ par rapport à s et λ_i

augmente, les temps d'attente sont considérablement réduits, mais le coût en O_i augmente considérablement en raison de l'augmentation de P^{fog}. Par conséquent, un choix optimal de n_i minimisant le coût total existe.

Dans la figure 5.5, nous analysons la variation de $Overhead_i$ par rapport au nombre de serveurs optimal $n_i^\star$, la vitesse du serveur s et λ_i. Cela signifie que pour chaque combinaison de s et λ_i, les paramètres utilisés sont les mêmes que ceux indiqués dans la figure 5.4. Les figures montrent qu'une vitesse plus élevée entraîne un nombre moins important de serveurs des nœuds fog nécessaires pour des valeurs λ_i différentes. En outre, plus λ_i est élevé, plus la surcharge minimale peut être obtenue.

Vitesse optimale : Étant donné λ_i, $\overline{x}$, P_*, γ, δ, χ, ϵ, D, ϕ, κ et n_i, notre objectif est de déterminer s telle que $Overhead_i$ du nœud fog i soit minimisée. Nous devons donc trouver s tel que $Overhead_i$ est le minimum.

$$\frac{\partial Overhead_i}{\partial s} = \frac{\partial O_{cld}}{\partial s} + \frac{\partial O_i}{\partial s} = 0 \tag{5.19}$$

où

$$\frac{\partial \mathrm{O}_{cld}}{\partial s} = \lambda_i \frac{\partial p_{cld}(\mathrm{D})}{\partial s} \left[\frac{1}{\mu_{cld}} \epsilon \mathrm{P}_{i,c} + \frac{1}{\mu_c} (\gamma + \delta \mathrm{P}_{cld}) \right]$$

et

$$\frac{\partial \mathrm{O}_i}{\partial s} = \lambda_i \delta \kappa \bar{x} s^{\phi-2} \left[(\phi - 1)(1 - p_{cld}(\mathrm{D})) - s \frac{\partial p_{cld}(\mathrm{D})}{\partial s} \right]$$

Étant donné que

$$a = \frac{\lambda_i}{\mu_i} = \frac{\lambda_i \bar{x}}{s}$$

Ainsi que

$$\frac{\partial \mathrm{B}(n_i, a)}{\partial a} = \{ \frac{n_i}{a} - 1 + \mathrm{B}(n_i, a) \} \mathrm{B}(n_i, a)$$

Nous avons,

$$a = \frac{\lambda_i}{\mu_i} = \frac{\lambda_i \bar{x}}{s}$$

et

$$\frac{1}{a} \frac{\partial a}{\partial s} = n \left(1 - \frac{1}{\rho} \right) \frac{\partial a}{\partial s}$$

Alors, nous obtenons

$$\frac{\partial \mathrm{K}_1}{\partial s} = -\mathrm{D}(\mathrm{K}_1 - \mathrm{K}_2) \frac{n}{\bar{x}}$$

où

$$\mathrm{K}_2 = \sqrt{2\pi n}(\rho + n(1-\rho)^2)$$

De plus, nous avons

$$\begin{aligned} \frac{\partial p_{ext}(\mathrm{D})}{\partial s} = \frac{1}{(\mathrm{K}_2 - \rho \mathrm{K}_1)^2} \Big(& \frac{\rho}{n} \mathrm{K}_1 (\mathrm{K}_2 - \mathrm{K}_1) \\ & + (\rho - 1) \mathrm{K}_1 \sqrt{2\pi n} (\rho + n(1-\rho)^2) \frac{\varphi}{s} \\ & + (1 - \rho) \mathrm{D} \mathrm{K}_1 \mathrm{K}_2 \frac{n}{\bar{x}} \Big). \end{aligned}$$

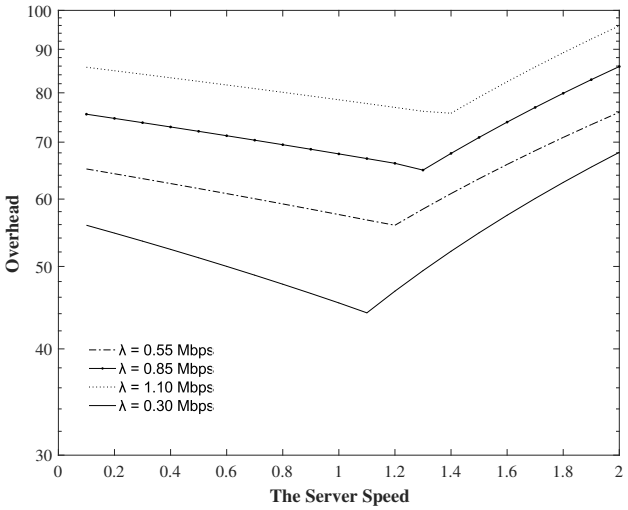


FIGURE 5.6 – La variation du overhead des nœuds fog en fonction de s et λ

De même que pour n_i, nous ne pouvons obtenir qu'une expression de forme semi-fermée proche de s. Nous pouvons donc utiliser la même méthode pour trouver la solution numérique de s. Dans la figure 5.6, nous montrons le coût total du nœud i en fonction de s et λ_i, où $n = 10$. Les

paramètres restants sont les mêmes que ceux indiqués dans la figure 5.4. Nous remarquons qu'il existe un choix optimal de s, grâce auquel la surcharge totale est réduite au minimum. Lorsque le nœud fog i fonctionne à une vitesse inférieure à la vitesse optimale, les temps d'attente des requêtes de service sont longs et dépassent le seuil D. Le coût de traitement dans le cloud O_{cld} est donc élevé, et la surcharge augmente. Lorsque s augmente, la consommation d'énergie du nœud fog augmente également, ce qui peut induire à une mauvaise utilisation du fog. Par conséquent, il existe un choix optimal de s, de manière à minimiser l'overhead.

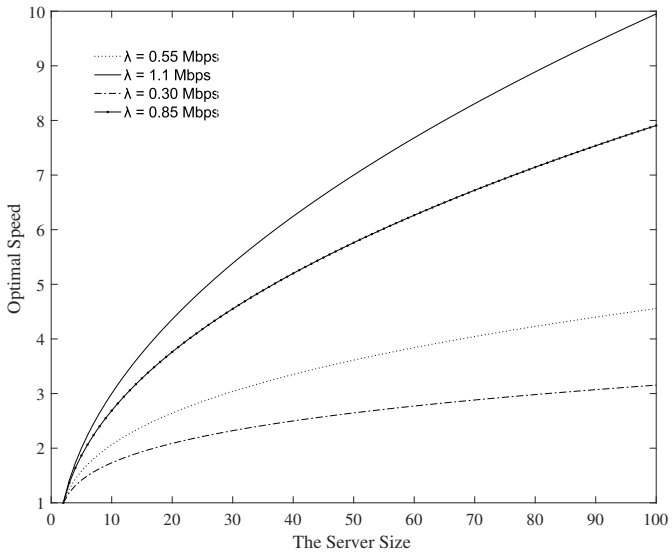


FIGURE 5.7 – La variation du overhead des nœuds fog en fonction de s et λ

Dans la figure 5.6, nous analysons la vitesse optimale et la surcharge minimale en fonction de n_i et λ_i. Les paramètres utilisés sont les mêmes que ceux présentés dans la figure 5.4. La figure montre que si le nombre de serveurs de nœuds fog fixes est important, le nœud fog doit fonctionner à une vitesse basse pour générer une surcharge optimale. En outre, la vitesse optimale du nœud fog ne doit pas dépasser 1,2, car le coût du traitement dans le fog ne devient supérieur que lors du transfert des requêtes vers le cloud. La figure montre également que différentes valeurs de λ_i produisent différentes combinaisons optimales de la vitesse et du nombre de serveurs.

Taille et vitesse optimales : Compte tenu du nœud fog i, λ_i, $\overline{x}$, P_*, γ, δ, χ, ϵ, D, ϕ et κ, notre objectif est de trouver n_i et s de sorte que $Overhead$ soit minimal. Par conséquent, nous devons trouver n_i et s tels que $\partial Overhead/\partial n_i$ et $\partial Overhead/\partial s$ ont été dérivés auparavant.

Ainsi, en nous basant sur les résultats obtenus précédemment, la solution optimale à notre problème peut être efficacement calculée par l'algorithme itératif proposé à l'aide de la méthode décente des coordonnées de bloc, détaillée dans l'algorithme 1. Notre problème étant conjointement convexe par rapport au nombre de serveurs n_i et à la vitesse du serveur s, le résoudre peut garantir la convergence vers une solution optimale. Étant donné le nœud fog i,λ_i, $\overline{x}$, P_*, γ, δ, χ, ϵ, D, ϕ et κ, la méthode est présentée dans l'algorithme 5.

5.3.2 Quelques propriétés du système

Considérons le système de délai d'Erlang décrit précédemment, à savoir une file d'attente M/M/n. Dans ce qui suit, nous analyserons $p_{cld}(d)$ et O_f, puis nous présenterons divers attributs de notre solution en utilisant les propriétés C et W.

Le lemme suivant indique que, lorsque la charge de travail d'un nœud fog est maintenue constante, p_{cld}(D) la probabilité de transmettre des requêtes au cloud est réduite par l'ajout de serveurs.

Algorithm 5 Trouver le nombre de serveurs et la vitesse optimale

$Overhead \leftarrow 0$
Trouver la taille du serveur n_l^* et n_r^* et sa vitesse en utilisant les méthodes analytiques
$n_l^* \leftarrow \lfloor n^* \rfloor; n_r^* \leftarrow \lceil n^* \rceil;$
Trouver la vitesse optimale s_l^* et s_r^* en utilisant les méthodes analytiques
$Overhead_l^* \leftarrow Overhead(n_l^*, s_l^*)$ et $Overhead_r^* \leftarrow Overhead(n_r^*, s_r^*)$
if $Overhead_l^* < Overhead_r^*$ **then**
  $n^* \leftarrow n_l^*$
  $s^* \leftarrow s_l^*$
else
  $n^* \leftarrow n_r^*$
  $s^* \leftarrow s_r^*$
end if

Lemme 10. *Soit le nœud fog i où $a > 0$ et $n_i \in \mathbb{R}$, avec $n_i > a$. Dans ce cas, $p_{cld}^t(\mathrm{D}) = \frac{(1-\rho)(1-\Lambda(tn_i,ta))}{1-\rho(1-\Lambda(tn_i,ta)}$ cela diminue en t, pour $t > 0$.*

Démonstration. Comme indiqué dans [91], $\mathrm{B}(tn_i, ta)$ diminue en t pour $t > 0$, sachant que $n_i > 0$. En combinant ce résultat avec le lemme 9, cela signifie que $\mathrm{C}(tn_i, ta)$ diminue aussi pour $t > 0$. Par conséquent, $\Lambda(tn_i, ta)$ et $p_{cld}^t(\mathrm{D})$ diminuent également en t pour $t > 0$. □

Le théorème suivant indique que, lorsque la charge de travail d'un nœud fog i par serveur est maintenue constante, O_i la surcharge attendue du nœud fog est réduite en ajoutant des serveurs.

Théorème 11. *Soit le nœud fog i où $a > 0$ et $n_i \in \mathbb{R}$, avec $n_i > a$. Ainsi, $\mathrm{O}_i(tn_i, \lambda_i, \mu_i)$ cela diminue en t, pour $t > 0$.*

Démonstration. La preuve du théorème 11 découle directement du lemme 10 □

Le théorème suivant stipule que O_i la surcharge attendue d'un nœud fog i est décroissante et strictement convexe en s.

Théorème 12. *Soit le nœud fog i où $\lambda_i, \mu_i > 0$ et $n_i \in \mathbb{R}$. Ensuite $\mathrm{O}_i(n_i, t\lambda_i, \mu_i)$est une fonction décroissante et strictement convexe de n_i, pour $n_i \in \mathbb{R}$ où $n_i > \frac{\lambda_i}{\mu_i}$.*

Démonstration. Soit le nœud i, pour un choix fixe de $\overline{x}$, P^*, δ, τ, χ, D, ϕ, κ, λ_i et μ_i, nous notons par $a = \lambda_i/\mu_i$et nous définissons, pour chaque $n_i \in \mathbb{R}$ avec $n_i > a$, $g(s) = \delta\rho(1 - p_{cld}(\mathrm{D}))\kappa s^{\phi} + \mathrm{P}^*$ et $f(s) = n_i$. Ainsi, pour chaque $n_i \in \mathbb{R}$ avec $n_i > a$, on $\mathrm{O}_i(n_i, t\lambda_i, \mu_i) = f(n_i)(g(n_i) + b)$ où $b = \chi + \frac{1}{\mu_{wl}}(\tau \mathrm{P}_{o,i})$. Comme montré par Jagers et van Doorn (1991), l'extension continue de la fonction de délai d'Erlang est positive, différentiable, non croissante et convexe en n_i, f a les mêmes propriétés. En outre, g est positif, différentiable, décroissant et strictement convexe. Par conséquent, laisser un nombre premier dénote une différenciation en ce qui concerne n_i, nous trouvons que $\mathrm{O}_i'((n_i, t\lambda_i, \mu_i) = f'(n_i)(g(n_i) + b) + f(n_n)g'(n_i) \leq f(n_i)(g(n_i) < 0$ et $\mathrm{O}''(n_i, t\lambda_i, \mu_i) = f''(n_i)(g(n_i) + b) + 2f'(n_i)g(n_i) + f(n_i)g''(n_i) \geq f''(n_i)(g(n_i) + b) + f(n_i)g''(n_i) > 0$. Ces inégalités tiennent par les signes des dérivées de f et g décrites dans le théorème suivant. □

Le dernier théorème indique que la combinaison de deux files d'attente M/M/n distinctes avec des taux de service communs en un seul système réduira toujours le coût total des deux systèmes.

Théorème 13. *Soit les nœuds fog $i, j \in \mathcal{N}$ où $\lambda_i, \lambda_j, \mu_i > 0$, $n_i \in \mathbb{R}$ avec $n_i > \lambda_i/\mu_i$ et $n_j \in \mathbb{R}$ avec $n_j > \lambda_j/\mu_i$. Il tient que $\mathrm{O}_{i,j}(n_i + n_j, \lambda_i + \lambda_j, \mu_i) \cdot (\lambda_i + \lambda_j) < \mathrm{O}_j(n_i, \lambda_i, \mu_i) \cdot \lambda_i + \mathrm{O}_j(n_j, +\lambda_j, \mu_i) \cdot \lambda_j$.*

Démonstration. La propriété de sous-additive découle de :

$$\begin{aligned} O_{i,j}(n_i+n_j,\lambda_i+\lambda_j,\mu_i) &< \frac{\lambda_i}{\lambda_i+\lambda_j}O_i(\frac{\lambda_i+\lambda_j}{\lambda_i}n_i, \\ &\quad \lambda_i+\lambda_j,\mu) + \frac{\lambda_j}{\lambda_i+\lambda_j}O_j(\frac{\lambda_i+\lambda_j}{\lambda_j}n_j, \\ &\quad \lambda_i+\lambda_j,\mu) \\ &< \frac{\lambda_i}{\lambda_i+\lambda_j}O_i(n_i,\lambda_i,\mu_i) \\ &\quad + \frac{\lambda_j}{\lambda_i+\lambda_j}O_j(n_j,\lambda_j,\mu_i) \end{aligned}$$

où la première inégalité tient par la propriété de convexité du théorème 12, et la seconde par le théorème 11. Multiplier les deux côtés avec $\lambda_i+\lambda_j>0$ complète la preuve. □

5.3.3 Solution de coalition entre nœuds fog

Comme indiqué précédemment, les nœuds fog sont encouragés à unir leurs forces si cela améliore leurs performances. À l'aide de notre système, des serveurs temporaires sont alloués pour libérer la charge totale. Ici, les nœuds fog d'une même instance fog peuvent collaborer, pour traiter ensemble le nombre total des requêtes entrantes. Dans cette partie, nous présenterons et analyserons ce jeu coopératif. Nous proposerons une solution pour trouver des coûts équitables répartis entre tous les nœuds fog appartenant à une même instance fog dans un système en pool.

Pour analyser une coalition entre des nœuds fog appartenant à une même instance, nous analyserons d'abord une coalition avec un nombre fixe de serveurs. Par la suite, nous considérerons le cas où chaque coalition peut optimiser le nombre de serveurs afin de minimiser la somme des coûts en ressources.

Nombre de serveurs fixe

Préliminaires de notre jeu coopératif : Nous considérons une instance fog $\mathscr{FI}$ qui contient m nœuds fog, plus précisément des joueurs. Chacun d'entre eux est témoin d'un processus de Poisson des requêtes entrantes. Les processus d'arrivée des joueurs sont indépendants. Chaque joueur a un nombre de serveurs donné de manière exogène et non ajustable. Les temps de service pour une requête arbitraire sont exponentiels, indépendants et identiquement distribués (i.i.d). Les requêtes trouvant tous les serveurs occupés à l'arrivée attendent dans une file d'attente, ce qui entraîne des coûts de délais proportionnels à la durée de leur séjour. Ces coûts de délais, qui sont symétriques d'un joueur à l'autre, pourraient représenter le mécontentement de l'utilisateur final. Ces derniers sont à la charge du nœud fog auquel appartient l'utilisateur final. Les joueurs s'intéressent à leurs coûts moyens à long terme par unité de temps.

Pour analyser les paramètres de notre jeu, nous définissons d'abord une situation de files d'attente avec un nombre fixe de serveurs (FIX-queueing) sous forme de tuple $(\mathscr{FI},\lambda,\mu,n,d)$, où

- $\mathscr{FI}$ est l'ensemble fini de joueurs non vide, c'est-à-dire les membres d'une instance fog ;
- $\lambda=(\lambda_k)_{k\in\mathscr{FI}}$, où $\lambda_k>0$ est le taux d'arrivée des requêtes du joueur $k\in\mathscr{FI}$;
- $\mu>0$ est le taux de service ;
- $n=(n_k)_{k\in\mathscr{FI}}$ avec $n_k>\lambda_k/\mu$ représente le nombre de serveurs appartenant aux joueurs $k\in\mathscr{FI}$;
- $d>0$ est le coût en délai induit par toute requête dans la file d'attente d'une unité de temps dans le système.

Un jeu de coût coopératif avec utilité transférable est une paire (F,c) , avec F un ensemble non vide fini de joueurs et $c:2^F_-\to\mathbb{R}$ une fonction de coût caractéristique. Ici, $2^F_-=\{F\subset\mathscr{FI}|F\neq\varnothing\}$

représente l'ensemble des coalitions (non vides). La valeur $c(\mathrm{F})$ représente le coût total encouru lorsque tous les joueurs unissent leurs forces et forment la grande coalition $\mathscr{F}\mathscr{I}$. Pour chaque coalition $\mathrm{F} \in 2^{\mathrm{FI}}_{-}$, on note $\lambda_{\mathrm{F}} = \sum_{k\in\mathrm{F}} \lambda_k$ et $n_{\mathrm{F}} = \sum_{k\in\mathrm{F}} n_k$. Enfin, notre problème consiste à allouer $c(\mathscr{F}\mathscr{I})$ de manière juste aux différents joueurs. Une allocation pour une partie $(\mathscr{F}\mathscr{I}, c)$ est un vecteur $y \in \mathbb{R}^{\mathscr{F}\mathscr{I}}$ satisfaisant $\sum_{k\in\mathscr{F}\mathscr{I}} y_k = c(\mathscr{F}\mathscr{I})$. La valeur y_k est interprétée comme les coûts attribués au joueur k. Ainsi, l'allocation est dite stable si f $\sum_{k\in\mathscr{F}} y_k \leq c(\mathscr{F})$ pour tout $\mathrm{F} \in 2^{\mathscr{F}\mathscr{I}}_{-}$. Dans une répartition stable, chaque groupe de joueurs ne doit pas payer plus collectivement que ce à quoi il devrait faire face s'il se séparait et agissait de manière indépendante. L'ensemble de toutes les allocations stables s'appelle le noyau [93]. Une solution d'allocation pour un jeu $(\mathscr{F}\mathscr{I}, c)$ est un vecteur $y = (y_{i,\mathrm{F}})_{i\in\mathrm{F},\mathrm{F}\in 2^{\mathscr{F}\mathscr{I}}_{-}}$, avec $\sum_{i\in\mathrm{F}} y_{i,\mathrm{F}} = c(\mathrm{F})$ pour toutes les coalitions $\mathrm{F} \in 2^{\mathscr{F}\mathscr{I}}$, qui précise comment attribuer les coûts de chaque coalition à ses membres. Un tel système est appelé schéma d'allocation monotone de population, si le montant qu'un joueur doit payer n'augmente pas lorsque la coalition à laquelle il appartient se développe. Ainsi, $y_{i,\mathrm{F}} \geq y_{i,\mathrm{L}}$ pour tout $\mathrm{F},\mathrm{L} \in 2^{\mathscr{F}\mathscr{I}}$ avec $\mathrm{F} \subseteq \mathrm{L}$ et $i \in \mathrm{F}$. Si une partie $(\mathscr{F}\mathscr{I}, c)$ admet un schéma d'allocation monotone de population y, l'allocation attribuant $y_{i,\mathrm{F}}$ à chaque $i \in \mathscr{F}\mathscr{I}$ est stable.

Le jeu : Nous considérons une situation de file d'attente FIX $\Psi = (\mathscr{F}\mathscr{I}, \lambda, \mu, n, d)$ et une coalition $\mathrm{F} \in 2^{\mathscr{F}\mathscr{I}}_{-}$. Les joueurs de cette coalition collaborent, en regroupant leurs flux et serveurs d'arrivée respectifs dans un système commun avec n_{F} serveurs et avec λ_{F} taux d'arrivés. Nous supposons que chaque serveur peut traiter toutes les requêtes, que tous les services sont déjà implémentés dans les serveurs et qu'une discipline de service non biaisée est utilisée.

Sur la base de ces hypothèses, le système en pool se comporte comme un système de délai d'Erlang. Le temps de séjour attendu qu'un client arbitraire passe dans le système est égal à $\mathrm{R}(n_{\mathrm{F}}, \lambda_{\mathrm{F}}, \mu)$. Par conséquent, $\lambda_{\mathrm{F}}\mathrm{R}(n_{\mathrm{F}}, \lambda_{\mathrm{F}}, \mu)\left(d + \mathrm{P}^{fog}\right)$ représentent sont les coûts par unité de temps en régime établi. Nous appelons le jeu $(\mathscr{F}\mathscr{I}, c^{\Psi})$, avec :

$$c^{\mathrm{P}si}(\mathrm{F}) = \lambda_{\mathrm{F}}\mathrm{R}(n_{\mathrm{F}}, \lambda_{\mathrm{F}}, \mu)\left(d + \mathrm{P}^{fog}\right) \tag{5.20}$$

pour tout $\mathrm{F} \in 2^{\mathscr{F}\mathscr{I}}_{-}$ le jeu FIX-M/M/n est associé à Ψ. En nous basant sur le théorème 13, nous pouvons facilement prouver que les jeux FIX-M/M/n sont strictement sous-additifs. Ainsi, la coopération entraîne toujours une réduction des coûts, et la collaboration entre les joueurs est bénéfique. Cependant, cela n'implique pas immédiatement l'existence d'une répartition stable des coûts.

Coûts d'allocation stable : Le théorème suivant montre que nos jeux FIX-M/M/n permettent toujours une allocation stable.

Théorème 14. *Soit $\Psi = (\mathscr{F}\mathscr{I}, \lambda, \mu, n, d)$ une situation de FIX-queueing. Si $|\mathscr{F}\mathscr{I}| > 1$, il y a une infinité d'allocations de base pour le jeu $(\mathscr{F}\mathscr{I}, c^{\Psi})$.*

(La preuve du théorème est détaillée dans la section A.3 de l'annexe.)

Enfin, nous montrons que si le rapport entre le nombre de serveurs d'un nœud fog et le taux d'arrivée est le même pour tous les membres d'une instance fog, le délai diminue à mesure que la coalition grandit. Cela implique que le fait de laisser les joueurs payer uniquement les coûts de leurs propres requêtes permet d'obtenir une allocation stable.

Théorème 15. *Soit $\Psi = (\mathscr{F}\mathscr{I}, \lambda, \mu, n, d)$ est une situation de FIX-queueing et $n_i/lambda_i = n_j/\lambda_j$ pour tous les joueurs $i, j \in \mathscr{F}\mathscr{I}$. En considérant le jeu* FIX − M/M/n, $(\mathscr{F}\mathscr{I}, c^{\Psi})$ *:*

1. *Pour tous $\mathrm{F},\mathrm{L} \in 2^{\mathscr{F}\mathscr{I}}_{-}$ avec $\mathrm{F} \subset \mathrm{L}$, il tient que $\mathrm{R}(n_{\mathrm{F}}, \lambda_{\mathrm{F}}, \mu) > \mathrm{R}(n_{\mathrm{L}}, \lambda_{\mathrm{L}}, \mu)$;*
2. *L'allocation qui attribue $\mathrm{R}(n_{\mathrm{F}}, \lambda_{\mathrm{F}}, \mu) \cdot \lambda_i d$ pour tout $i \in \mathscr{F}\mathscr{I}$ est une allocation stable.*

Démonstration. (1). Soit $\mathrm{F},\mathrm{L} \in 2_{-}^{\mathscr{F}\mathscr{I}}$ avec $\mathrm{F} \subset \mathrm{L}$. Nous avons :

$$\mathrm{R}(n_\mathrm{F},\lambda_\mathrm{F},\mu) < \mathrm{R}(n_\mathrm{F}\cdot\frac{n_\mathrm{L}}{n_\mathrm{F}},\lambda_\mathrm{F}\cdot\frac{n_\mathrm{L}}{n_\mathrm{F}},\mu) = \mathrm{R}(n_\mathrm{L},\lambda_\mathrm{L},\mu)$$

où l'inégalité peut être déduite du théorème 14 et l'égalité tient parce que $\frac{n_\mathrm{F}}{\lambda_\mathrm{F}} = \frac{n_\mathrm{L}}{\lambda_\mathrm{L}}$.

(2) résulte immédiatement de (1). □

Nombre de serveurs optimisé

Préliminaires de notre jeu coopératif : Nous considérons le cas où chaque coalition peut optimiser le nombre de serveurs (afin de minimiser la somme des coûts en ressources). Il s'agit d'un modèle raisonnable pour les situations dans lesquelles le nombre de serveurs peut facilement être ajusté.

Si l'optimisation est permise, la coalition ne sera pas moins pire que dans le cas où le nombre de serveurs est fixe; cela peut élargir le noyau. Cependant, les sous-coalitions peuvent aussi réduire leurs coûts; cela peut réduire le noyau. Compte tenu de ces deux effets opposés, un jeu avec un nombre optimisé de serveurs présente une structure différente de celle d'un jeu avec un nombre fixe de serveurs.

Formellement, une situation de file d'attente ayant un nombre optimisé de serveurs est représentée par un tuple $(\mathscr{F}\mathscr{I},\lambda,\mu,h,d)$, où $\mathscr{F}\mathscr{I}$,λ,μ et d sont identiques à ceux définis précédemment. $h = (h_\mathrm{F})_{2_{-}^{\mathrm{F}}}$ tel que $h_f \geq h_\mathrm{L} > 0$ pour $\mathrm{M},\mathrm{L} \in 2_{-}^{\mathscr{F}\mathscr{I}}$ avec $\mathrm{F} \subseteq \mathrm{L}$. Ici, h_F représente le coût des ressources et des communications engagées par unité de temps par serveur pour la coalition F.

Le jeu : Soit $\vartheta = (\mathscr{F}\mathscr{I},\lambda,\mu,h,d)$ une situation de file d'attente optimale; considérons une coalition $2_{-}^{\mathscr{F}\mathscr{I}}$. Cette coalition servira des requêtes entrantes selon un processus de Poisson avec un taux combiné $\lambda_\mathrm{F} = \sum_{k\in\mathrm{F}} \lambda_k$. Nous supposons que cette coalition choisirait des serveurs communs $n > \lambda_\mathrm{F}/\mu$. Ensuite, les nœuds fog de cette coalition se comportent comme un système de délai d'Erlang et les coûts escomptés par unité de temps à l'état stationnaire supportés par la coalition F sont égaux à :

$$\mathrm{M}_\mathrm{F}^{\vartheta} = h_\mathrm{F} n + \lambda_\mathrm{F}\mathrm{R}(n,\lambda_\mathrm{F},\mu)\left(d + \mathrm{P}^{fog}\right) \tag{5.21}$$

Nous modélisons un jeu correspondant à la situation optimale de file d'attente ϑ, où toute coalition F optimise $\mathrm{M}_\mathrm{F}^{\vartheta}$ sur un nombre entier de serveurs, c'est-à-dire sur le domaine $\mathfrak{N}_\mathrm{F}^{\vartheta} = \{n \in \mathbb{N}\ n > \lambda_\mathrm{F}/\mu\}$. Nous soulignons que le principal objectif de ceci réside dans un jeu où n est un entier, car il représente le problème d'optimisation discret exact. Néanmoins, il est possible de considérer une situation où n n'est pas un entier.

Nous appelons le jeu $(\widetilde{\mathscr{F}\mathscr{I}},\widetilde{c}^{\vartheta})$, avec :

$$\widetilde{c}^{\vartheta}(\mathrm{F}) = \min_{n\in\mathfrak{N}_\mathrm{F}^{\vartheta}} \mathrm{M}_\mathrm{F}^{\vartheta}(n) \tag{5.22}$$

pour tout $\mathrm{F} \in 2_{-}^{\mathscr{F}\mathscr{I}}$ est le jeu optimal associé $\mathrm{M}/\mathrm{M}/n^{\mathfrak{N}}$. Considérons une coalition $\mathrm{F} \in 2_{-}^{\mathscr{F}\mathscr{I}}$ sur le domaine $\mathfrak{N}_\mathrm{F}^{\vartheta}$; la fonction du coût $\mathrm{M}_\mathrm{F}^{\vartheta}$ est strictement convexe (elle peut être déduite du théorème 15), et elle atteint un minimum (puisque les coûts augmentent de manière illimitée à mesure que le nombre tend vers l'infini). Par conséquent, un nombre entier optimal de serveurs est donné par le plus petit $n \in \mathfrak{N}_\mathrm{F}^{\vartheta}$ satisfaisant $\mathrm{M}_\mathrm{F}^{\vartheta}(n+1) \geq \mathrm{M}_\mathrm{F}^{\vartheta}(n)$, et nous désignons ce nombre optimal par $\widetilde{n}_\mathrm{M}^{*}$.

Allocation de coûts stables : Nous nous concentrons sur la règle qui consiste à attribuer les coûts de toute coalition proportionnellement aux taux d'arrivée, c'est-à-dire que nous définissons $\widetilde{\varkappa}_{i,\mathrm{F}}(\vartheta) = \widetilde{c}^{\varkappa}(\mathrm{F})\lambda_i/\lambda_\mathrm{F}$ et $\varkappa_{i,\mathrm{F}}(\vartheta) = c^{\varkappa}(\mathrm{F})\lambda_i/\lambda_\mathrm{F}$ pour toute situation de file d'attente optimale $\vartheta = (\mathscr{F}\mathscr{I},\lambda,\mu,h,d)$, coalition $\mathrm{F} \in 2_{-}^{\mathscr{F}\mathscr{I}}$, et le joueur $i \in \mathrm{F}$. Le théorème suivant énonce des conditions suffisantes pour que les jeux optimaux $\mathrm{M}/\mathrm{M}/n^{\mathfrak{N}}$ permettent de traiter une allocation de base et admettent un schéma d'allocation monotone de la population.

Théorème 16. *Soit* $\vartheta = (\mathscr{F}\mathscr{I}, \lambda, \mu, h, d)$ *une situation de file d'attente optimale.*

1. *si* $n^*_{\mathrm{F}} \in \mathbb{N}$ *pour tout* $\mathrm{F} \in 2^{\mathscr{F}\mathscr{I}}_{-}$, *ainsi* $\widetilde{\varkappa}_{i,\mathrm{F}}(\vartheta)$ *est un schéma d'allocation monotone de la population pour l'optimum* $(\mathscr{F}\mathscr{I}, \widetilde{c}^{\vartheta})$;
2. *si* $n^*_{\mathrm{F}} \in \mathbb{N}$, *then* $(\widetilde{\varkappa}_{i,\mathrm{F}}(\vartheta))_{i\in\mathrm{F}}$ *est une allocation stable pour l'optimum* $(\mathscr{F}\mathscr{I}, \widetilde{c}^{\vartheta})$.

Démonstration. (1) Soit $\mathrm{M}, \mathrm{L} \in 2^{\mathscr{F}\mathscr{I}}_{-}$ avec $\mathrm{F} \subseteq \mathrm{L}$, et $i \in \mathrm{F}$. Supposons que $n^*_{\mathrm{F}}, n^*_{\mathrm{L}} \in \mathbb{N}$. Nous avons :

$$\begin{aligned}\widetilde{\varkappa}_{i,\mathrm{L}}(\vartheta) = \widetilde{c}^{\varkappa}(\mathrm{L})\frac{\lambda_i}{\lambda_{\mathrm{L}}} = c^{\varkappa}(\mathrm{L})\frac{\lambda_i}{\lambda_{\mathrm{L}}} = \varkappa_{i,\mathrm{L}}(\vartheta) \geq \varkappa_{i,\mathrm{F}}(\vartheta)\\ = c^{\varkappa}(\mathrm{F})\frac{\lambda_i}{\lambda_{\mathrm{F}}} = \widetilde{c}^{\varkappa}(\mathrm{F})\frac{\lambda_i}{\lambda_{\mathrm{F}}} = \widetilde{\varkappa}_{i,\mathrm{F}}(\vartheta)\end{aligned}$$

nous concluons que $\widetilde{\varkappa}(\vartheta)$ est en effet un schéma d'allocation monotone pour $(\mathscr{F}\mathscr{I}, \widetilde{c}^{\vartheta})$.

(2) Soit $\mathrm{M} \in 2^{\mathscr{F}\mathscr{I}}_{-}$ et supposons que $n^*_{\mathrm{F}} \in \mathbb{N}$. Ensuite, nous savons que $\widetilde{c}^{\varkappa}(\mathrm{F}) = c^{\varkappa}(\mathrm{F})$ et $\widetilde{c}^{\varkappa}(\mathrm{F}) \geq c^{\varkappa}(\mathrm{F})$. Ainsi, $(\widetilde{\varkappa}_{i,\mathscr{F}\mathscr{I}}(\vartheta))_{i\in\mathscr{F}\mathscr{I}} = (\varkappa_{i,\mathscr{F}\mathscr{I}}(\vartheta))_{i\in\mathscr{F}\mathscr{I}} \in Core(\mathscr{F}\mathscr{I}, c^{\varkappa}) \subseteq Core(\mathscr{F}\mathscr{I}, \widetilde{c}^{\varkappa})$. □

Dans la section suivante, nous allons examiner le problème d'allocation de ressources et de distribution de requêtes du côté de l'utilisateur final. Nous modéliserons ce problème comme un problème de Bandit manchot, dans lequel la fonction de coût de traitement des requêtes est minimisée grâce un algorithme d'apprentissage basé sur le machine learning.

5.4 Solution implémentée au niveau des utilisateurs finaux

Considérons l'architecture fog/cloud déjà présentée dans la section 3.2.2, composée d'un ensemble $\mathscr{N}$ de N objets IoT répartis uniformément sur la zone réseau, d'un ensemble $\mathscr{M}$ de M nœuds fog et d'un cloud distant. Chaque objet $n \in \mathscr{N}$ est connecté à un nœud fog via son point d'accès (AP) le plus proche, noté p^*_n, tandis que les nœuds fog sont connectés au DC du cloud via la fibre optique. De plus, un dispositif $n \in \mathscr{N}$ n'a accès qu'à un sous-ensemble de nœuds fog dans sa couverture $\mathscr{M}_n \subset \mathscr{M}$, où la surface de couverture est calculée en fonction des distances attendues entre l'objet et les nœuds fog, supposée être inférieure à un seuil d_n. Chaque objet IoT peut exécuter plusieurs instances d'applications localement, et génère des requêtes de manière aléatoire. Ces requêtes sont réparties sur des nœuds fog ou le cloud.

Une requête générée par l'objet $n \in \mathscr{N}$, est décrite par $\mathrm{R}_n = \{\mathrm{S}_n, App_n, \kappa, \gamma^{max}_n\}$, où S_n la taille de la requête (en bits). Chaque requête nécessite κ cycles de CPU par bit, γ^{max}_n est le délai maximal tolérable (en secondes), et App_n le type de l'application en cours d'exécution sur l'objet qui a généré la requête.

5.4.1 Modèle de coût

Notre objectif est de minimiser le coût total de traitement des requêtes en distribuant efficacement ces dernières. Nous définissons une fonction de coût du traitement de la requête R_n comme la somme pondérée de la consommation d'énergie et de la latence due au traitement :

$$Cost_n = \theta \mathrm{E}_{fog} + (1-\theta)\mathrm{T}_{fog} \tag{5.23}$$

où θ est un paramètre qui ajuste l'impact de la consommation d'énergie et de la latence, avec $\theta \in [0,1]$.

La consommation d'énergie E_{fog} et le délai T_{fog} pour le traitement d'une requête R_n dans le fog sont donnés par :

$$\mathrm{E}_{fog} = \mathrm{E}^{fog}_{\mathrm{Tr},\mathrm{R}_n} + \mathrm{E}^{fog}_{\mathrm{Pr},\mathrm{R}_n} \tag{5.24}$$

$$\mathrm{T}_{fog} = \mathrm{T}^{fog}_{\mathrm{Tr},\mathrm{R}_n} + \mathrm{T}^{fog}_{\mathrm{Pr},\mathrm{R}_n} \tag{5.25}$$

Le délai et l'énergie consommée lorsqu'un objet envoie une requête R_n sont définis par : (a) l'énergie consommée et le délai généré par la transmission de la requête, c'est-à-dire $\mathrm{E}^{fog}_{\mathrm{Tr,R}_n}$ et $\mathrm{T}^{fog}_{\mathrm{Tr,R}_n}$. Cela inclut l'énergie consommée et le délai, généré par la technologie d'accès sur le réseau de périphérie (routeurs et commutateurs entre l'objet et le fog) ; et (b) l'énergie consommée et délai pour traiter et stocker la requête de l'objet ($\mathrm{E}^{fog}_{\mathrm{Pr,R}_n}$, respectivement $\mathrm{T}^{fog}_{\mathrm{Pr,R}_n}$).

Selon les équations (3.1)- (3.3) et (3.6)- (3.8) , on peut simplifier les (5.24)- (5.25) comme suit :

$$\mathrm{E}_{fog} = \mathrm{E}^{m}_{\mathrm{Tr,R}_n} + \mathrm{E}^{m}_{\mathrm{Pr,R}_n} x_{m,\mathrm{R}_n} + \mathrm{E}^{cld}_{\mathrm{R}_n}(1 - \sum_{j \in \mathscr{M}_n} x_{j,\mathrm{R}_n}) \tag{5.26}$$

$$\mathrm{T}_{fog} = \mathrm{T}^{m}_{\mathrm{Tr,R}_n} + \mathrm{T}^{m}_{\mathrm{Pr,R}_n} x_{m,\mathrm{R}_n} + \mathrm{T}^{cld}_{\mathrm{R}_n}(1 - \sum_{j \in \mathscr{M}_n} x_{j,\mathrm{R}_n}) \tag{5.27}$$

où x_{m,R_n} est une variable binaire ; si $x_{m,\mathrm{R}_n} = 1$, cela signifie que R_n sera traitée par le fog m. Si $\sum_{j \in \mathscr{M}_n} x_{j,\mathrm{R}_n} = 0$, alors R_n sera traitée dans le cloud.

5.4.2 Modélisation du problème

Nous avons défini une fonction de coût pour le traitement des requêtes du système, dans le fog ou dans le cloud. Nous cherchons minimiser cette fonction, en distribuant efficacement les requêtes générées par les objets IoT. En utilisant la fonction de coût définie par (5.23), ainsi que la répartition des requêtes sur les nœuds fog et les matrices d'allocation de ressources de calcul $\mathrm{X} = [x_{m,\mathrm{R}_n}]$ et $f^{fog} = [f^m_n]$, le problème de minimisation des coûts est défini comme suit :

$$(\mathrm{P1}) : \min_{\mathrm{X}, f^m_n} \max_{n \in \mathscr{N}} \mathrm{C}ost_n \tag{5.28}$$

$$\text{s.t.} \quad (\mathrm{C1}) : \mathrm{T}^{fog}_n \leq \gamma^{max}_n \quad \forall n \in \mathscr{N} \tag{5.29}$$

$$(\mathrm{C2}) : \sum_{n \in \mathscr{N}} f^m_n \leq \mathrm{F}_{max} \quad \forall m \in \mathscr{M} \tag{5.30}$$

$$(\mathrm{C3}) : \frac{1}{\mathrm{T}} \sum_{t=1}^{\mathrm{T}} \sum_{n=1}^{\mathrm{N}} \mathrm{E}^{m,t}_n \leq \mathscr{Q} \tag{5.31}$$

$$(\mathrm{C4}) : \sum_{n \in \mathscr{N}} x_{m,\mathrm{R}_n} \leq 1 \quad \forall m \in \mathscr{M} \tag{5.32}$$

$$\tag{5.33}$$

où F_{max} est la capacité totale de calcul d'un nœud fog. Ainsi, le problème (P1) dans (5.28) vise à minimiser le coût maximum de traitement des requêtes de tous les objets, tout en garantissant une équité entre eux du point de vue du coût du système.

Nous avons quatre contraintes, décrites comme suit :

— (C1) permet de garantir que la latence de traitement d'une requête est limitée par un seuil ;

— (C2) indique que les ressources de calcul allouées ne peuvent pas dépasser la capacité de calcul maximale d'un nœud fog ;

— (C3) est la contrainte énergétique à long terme pour chaque nœud fog, ce qui nécessite que la consommation totale d'énergie à long terme ne dépasse pas une limite $\mathscr{Q}$;

— (C4) assure la correspondance une à une de la distribution des nouvelles requêtes sur les nœuds fog.

Notons que (P1) n'est pas convexe en raison de la modélisons min-max et des variables binaires X. C'est un problème de programmation non linéaire à nombres entiers mixtes, qui peut être complexe en termes de calcul [90]. Pour rendre ce problème traçable et réduire la complexité du calcul, nous simplifions l'expression min-max dans P1.

Nous définissons une variable de jeu telle que $\varphi_{\mathrm{R}_n} = \max_{\mathrm{R}_n \in \mathscr{R}} \mathrm{C}ost_n$, où $\mathscr{R}$ est un ensemble de R requêtes générées par les objets, c'est-à-dire que toutes les requêtes pouvant être générées à partir

des dispositifs sont dans $\mathscr{R}$. Soit $y_{\mathrm{R}_n} = 1 - \sum_{j \in \mathscr{M}_n} x_{j,\mathrm{R}_n}$, y_{R_n} représente les requêtes qui sont dans le cloud. Nous avons alors $Cost_n \leq \varphi_{\mathrm{R}_n}$, c'est-à-dire :

$$\Big(\theta \mathrm{E}^m_{\mathrm{Tr},\mathrm{R}_n} + (1-\theta)\mathrm{T}^m_{\mathrm{Tr},\mathrm{R}_n}\Big) + \Big(\theta \mathrm{E}^m_{\mathrm{Pr},\mathrm{R}_n} + (1-\theta)\mathrm{T}^m_{\mathrm{Pr},\mathrm{R}_n}\Big) x_{m,\mathrm{R}_n} + \Big(\theta(\mathrm{E}^{cld}_{\mathrm{Tr},\mathrm{R}_n} + \mathrm{E}^{cld}_{\mathrm{Pr},\mathrm{R}_n}) + (1-\theta)(\mathrm{T}^{cld}_{\mathrm{Tr},\mathrm{R}_n} + \mathrm{T}^{cld}_{\mathrm{Pr},\mathrm{R}_n})\Big) y_{m,\mathrm{R}_n} \leq \varphi_{\mathrm{R}_n}$$

Il est établi que les serveurs du cloud surpassent les ressources dans le fog en termes de capacités de calcul et de stockage [12]. Nous supposons alors que $\mathrm{T}^{cld}_{\mathrm{Tr},\mathrm{R}_n}$ et $\mathrm{T}^{cld}_{\mathrm{Pr},\mathrm{R}_n}$ sont des variables constantes. Étant donné que les ressources du cloud sont réparties équitablement entre les utilisateurs finaux, nous supposons également que f^{cld}_n est constant. Selon [76], $\mathrm{E}^m_{\mathrm{Tr},\mathrm{R}_n}$ et $\mathrm{T}^m_{\mathrm{Tr},\mathrm{R}_n}$ sont également constants. Soit $\max_{n \in \mathscr{N}} \theta \mathrm{E}^m_{\mathrm{Pr},\mathrm{R}_n} x_{m,\mathrm{R}_n} = \mathrm{E}^m_n$ et $\max_{n \in \mathscr{N}} (1-\theta)\mathrm{T}^m_{\mathrm{Pr},\mathrm{R}_n} x_{m,\mathrm{R}_n} = \mathrm{T}^m_n$.

Selon les définitions ci-dessus, (P1) peut être transformé en (P2) comme suit :

$$(\mathrm{P2}) : \min_{\mathrm{X}, f^m_n, \varphi_{\mathrm{R}_n}} \quad \varphi_{\mathrm{R}_n} \tag{5.34}$$

$$\text{s.t.} \quad (\mathrm{C1}) \text{ - } (\mathrm{C4}) \tag{5.35}$$

$$(\mathrm{C5}) : \sum_{\mathrm{R}_n \in \mathscr{R}} x^{m,\mathrm{R}_n} \leq 1 \forall m \in \mathscr{M} \tag{5.36}$$

$$(\mathrm{C6}) : Cost_n \leq \varphi_{\mathrm{R}_n} \tag{5.37}$$

$$(\mathrm{C7}) : \theta \mathrm{E}^m_{\mathrm{Pr},\mathrm{R}_n} x_{m,\mathrm{R}_n} \leq \mathrm{E}^m_n \tag{5.38}$$

$$(\mathrm{C8}) : (1-\theta)\mathrm{T}^m_{\mathrm{Pr},\mathrm{R}_n} x_{m,\mathrm{R}_n} \leq \mathrm{T}^m_n \tag{5.39}$$

$$\tag{5.40}$$

où $\Omega = [e_1, \ldots, e_{\mathrm{N}}]$, $e_n = [\mathrm{E}^m_n, \mathrm{T}^m_n]$. (C5) assure également une correspondance une à une lors de la distribution de nouvelles requêtes sur des nœuds fog.

5.4.3 Distribution des requêtes d'un objet

Pour résoudre le problème décrit en (P2), nous le remodelions comme un problème Bandit manchot dans lequel un objet tente de maximiser ses gains. Nous proposons aussi un algorithme d'apprentissage en ligne contextuel inspiré par [94]. L'algorithme apprend le coût prévu pour le traitement de l'ensemble de requêtes d'un objet sous différents états de nœuds fog (utilisation de la CPU, état d'inactivité, état de la file d'attente de nœuds fog, etc.). L'algorithme fonctionne en supposant que, pour des états de nœuds fog similaires, le coût des requêtes générées par des objets sera similaire en moyenne.

Modélisation du problème contextuel de Bandit manchot :

Dans un problème de Bandit manchot, un ensemble de joueurs essaie d'acquérir un ensemble limité de ressources de manière à maximiser le gain escompté des joueurs. Chaque choix dépend de propriétés qui ne sont que partiellement connues au moment de l'acquisition des ressources et qui peuvent être mieux comprises au fil du temps. Dans notre cas, les joueurs sont les objets ; chaque objet essaie d'acquérir les ressources du système fog/cloud qui minimisent le coût total du traitement de ces requêtes. Un objet IoT n peut utiliser un ensemble fini ($\mathscr{M}_n \subset \mathscr{M}$) de M_n nœuds de fog présents dans sa couverture. À chaque période t, l'objet distribuera ces requêtes sur le sous-ensemble sélectionné de nœuds fog B^t_n minimisant le coût de traitement de ces requêtes, où $\mathrm{B}^t_n < \mathrm{M}_n$ représente un nombre fixe. Par conséquent, l'objet devrait prendre en compte l'état de chaque nœud fog dans sa liste de couverture, car le coût dépend de l'état de ces derniers. Comme l'objet ne dispose pas de connaissances préalables sur son environnement (état des nœuds fog, nombre

de nœuds actifs, etc.), il doit apprendre au fil du temps quelles sont les meilleures distributions, compte tenu de son environnement.

Nous considérons un temps discret ; la génération des requêtes suit un processus de Poisson, où chaque objet met à jour la distribution de ses requêtes en une liste de nœuds fog à chaque nouvelle génération de requêtes. À chaque période $t = 1, \ldots, \mathrm{T}$:

— Un objet n recueille des informations sur le contexte (l'état d'utilisation des nœuds dans sa zone de couverture). Pour chaque $m \in \mathscr{M}_k$, nous notons $\delta^t_{m,n}$ le contexte collecté par n à t. Ce contexte est un vecteur ;

— L'objet n sélectionne un sous-ensemble $\mathscr{B}^t_n$;

— Nous associons chaque nœud $m \in \mathscr{B}^t_k$ avec un ensemble de requêtes de l'objet n, de telle sorte que la fonction de correspondance est définie comme $\Lambda(m,n,t) = \mathrm{R}^t_{m,n} \subset \mathscr{R}$. Ensuite, les nœuds fog sélectionnés sont informés de la distribution des requêtes ;

— Les nœuds fog peuvent rejeter le choix de l'objet si une requête ne vérifie pas la contrainte (C3) dans (5.31). Ensuite, l'objet trouvera des substituts à ces requêtes parmi $\mathscr{B}_n$;

— Quand une requête R_n est transmise à son nœud fog m, nous définissons par $cost_{\delta^t_{m,n}}$ le coût mesuré pour le traitement de R_n en m. Nous répartissons toutes les requêtes générées sur les nœuds fog sélectionnés et observons le coût total.

En raison de la contrainte (C2) dans (5.30), nous savons que $cost_{\delta^t_{m,n}}$ est limité à la fois par un coût minimal et maximal entre $[\mathrm{C}ost_{min}, \mathrm{C}ost_{max}]$. Nous définissons $\sigma_{\delta^t_{m,n}}$ comme la valeur estimée de la variable $cost_{\delta^t_{m,n}}$. Pour un objet n, nous désignons le sous-ensemble optimal par $\mathscr{B}^{t*}_n \subseteq \mathscr{B}^t_n = \{m^*\}_{m^* \in \mathrm{B}^t_n}$, qui dépend de $\chi_t = \{\delta^t_{m,n}\},\ m \in \mathscr{B}^t_n$.

Les nœuds fog B^t_n sélectionnés doivent satisfaire la condition :

$$m^* \in \underset{m \in \mathscr{B}^t_n \setminus \mathscr{B}^{t*}_n}{\operatorname{argmin}} \sigma_{\delta^t_{m,n}} \tag{5.41}$$

Si un objet n connaît le coût attendu $\sigma_{\delta^t_{m,n}}$ pour chaque état du nœud fog $\delta \in \chi$ et chaque $m \in \mathscr{M}_n$, il peut simplement choisir les nœuds les plus performants pour distribuer ses requêtes. Pour un intervalle $t = 1, \ldots, \mathrm{T}$, nous définissons le coût total de toutes les requêtes générées dans l'intervalle t comme suit :

$$\sum_{t=1}^{\mathrm{T}} \sum_{m^* \in \mathrm{B}^{t*}_n} \mathrm{E}[cost_{\delta^t_{m^*,n}}] = \sum_{t=1}^{\mathrm{T}} \sum_{m^* \in \mathrm{B}^{t*}_k} \sigma_{\delta^t_{m^*,n}} \tag{5.42}$$

Cependant, l'objet ne connaît pas l'état du fog en raison de la présence d'autres objets ; il doit donc connaître le coût attendu de traitement de ces requêtes au fil du temps σ_δ. Afin de connaître les valeurs de coût, l'objet essaie plusieurs combinaisons de nœuds fog/requêtes correspondant à différents états des nœuds fog au fil du temps. De la même manière, l'objet devrait trouver la sélection optimale des nœuds fog. Par conséquent, chaque objet doit trouver un compromis entre l'exploration des nœuds fog, sur lesquels il dispose de peu de connaissances, et l'exploitation des nœuds fog les plus performants. Pour chaque période de temps t, dans laquelle un nouvel ensemble de requêtes d'un objet n et χ_t est généré, notre algorithme d'apprentissage distribue l'ensemble de ces requêtes sur un sous-ensemble de nœuds $\mathscr{B}^t_n$. Soit n, $\mathscr{B}^t_n$ et $t = 1, \ldots, \mathrm{T}$, nous avons :

$$\sum_{t=1}^{\mathrm{T}} \sum_{m \in \mathrm{B}^t_k} \mathrm{E}[cost_{\delta^t_{m,n}}] = \sum_{t=1}^{\mathrm{T}} \sum_{m \in \mathrm{B}^t_k} \mathrm{E}[\sigma_{\delta^t_{m,n}}] \tag{5.43}$$

Le paramètre regret : Le regret d'apprentissage attendu peut être exprimé par la différence entre le montant du coût observé et le coût estimé du traitement des nouvelles requêtes générées par l'objet dans un intervalle $t = 1, \ldots, \mathrm{T}$, et peut être défini par :

$$\mathrm{R(T)} = \mathrm{E}\Big[\sum_{t=1}^{\mathrm{T}} \sum_{m^* \in \mathrm{B}^{t*}_n} cost_{\delta^t_{m^*,n}} - \sum_{t=1}^{\mathrm{T}} \sum_{m \in \mathrm{B}^t_n} \sigma_{\delta^t_{m,n}}) \Big] \tag{5.44}$$

L'algorithme d'apprentissage : Pour qu'un objet puisse choisir le sous-ensemble le plus approprié de nœuds fog en fonction des informations de contexte relatives à leur état, l'objet doit connaître le coût de traitement d'une requête spécifique au contexte. En raison de la modélisation du problème formel ci-dessus (problème de Bandit manchot), nous pouvons adapter et étendre un algorithme d'apprentissage [94] à notre environnement. Notre algorithme est basé sur l'hypothèse, selon laquelle nous attendons les mêmes performances des nœuds fog avec un état similaire. Si cette hypothèse est vérifiée, le contexte collecté peut être exploité par l'objet pour être pris en compte lors de futures décisions de répartition des requêtes. À cette fin, notre algorithme décrit dans 6 commence par la collecte du contexte. Ensuite, l'objet apprend indépendamment les meilleures distributions de requêtes. Pour chaque intervalle de temps, l'algorithme observe d'abord les contextes du sous-ensemble de nœuds fog $\mathscr{B}_n$. L'algorithme sélectionne alors le nœud de distribution pour les requêtes de l'objet dans cet intervalle de temps. Basé sur une certaine fonction de contrôle, notre algorithme d'apprentissage est soit en phase d'exploration, soit en phase d'exploitation. Dans la première phase, il choisit un ensemble aléatoire de nœuds fog pour traiter les requêtes de l'objet. Ces phases sont nécessaires pour connaître les coûts de traitement non utilisés auparavant. En phase d'exploitation, les requêtes de l'objet sont transmises aux nœuds fog qui étaient, en moyenne, les plus efficaces des créneaux précédents dans des contextes similaires. Après avoir réparti l'ensemble des requêtes de l'objet sur les nœuds fog, l'algorithme observe le coût de traitement. De cette manière, au fil du temps, l'algorithme apprend les performances spécifiques des nœuds fog dans certains contextes.

L'algorithme d'apprentissage en ligne (Online Learning Algorithm OLA) est décrit dans l'algorithme 5 comme suit : tout d'abord, chaque objet n couvre un sous-ensemble de nœuds fog et collecte des informations sur leurs contextes; $\Lambda(m, \delta^t_{m,n})$ est la fonction de correspondance du noeud fog m et des requêtes de n et $\bar{\sigma}_{\delta^t_{m,n}}$ est le coût estimé des requêtes appariées (avec m) traitées dans le contexte donné $\delta^t_{m,k}$. Pour chaque l'intervalle t, nous sélectionnons B^t_n noeuds fog où $\mathscr{B}^t_n \subset \mathscr{M}_n$; puis l'algorithme recueille toutes les observations faites par l'objet n pour construire $\chi_t = \{\delta^t_{m,n}\}_{m \in \mathscr{M}_n}$, où $\delta^t_{m,n}$ est l'état de $m \in \mathrm{B}^t_k$. Pour chaque $\delta^t_{m,n}$, nous recueillons toutes les observations sur les nœuds fog χ_t. Ensuite, l'algorithme calcule l'ensemble des nœuds fog sous-exploités $\mathrm{UE}_{\chi_t} = \cup_{m \in \mathrm{B}^t_n} \{m \in \mathscr{M}_k \quad s.t. \quad |\Lambda(m, h^t_{m,k})| < \mathrm{F}_t\}$, où $\mathrm{F} \in \mathbb{N}$ est une valeur prédéterminée.

Soit $u^t = |\mathrm{UE}_{\chi_t}|$, où chaque nœud fog sous-exploité contient r_{min}, le nombre minimal de requêtes correspondantes; si $u^t > \mathrm{B}^t_n$ l'algorithme sélectionne B^t_n nœuds fog dans UE_{χ_t}. Sinon, il sélectionne tous les u^t nœuds de UE_{χ_t}. Ensuite, l'algorithme sélectionne un ensemble de $\mathrm{B}^t_n - u^t$ nœuds fog dans $\mathscr{M}_n$. Les nœuds sélectionnés sont placés dans un ensemble appelé $\mathscr{B}^t_n$ qui dépend de χ_t. Chaque $m \in \mathscr{B}^t_n$ vérifie :

$$m \in \underset{m \in (\mathrm{UE}_{\chi_t} \cup \mathscr{M}_n) \setminus \mathscr{B}^t_n}{\arg\min} \bar{\sigma}_{\delta^t_{m,n}} \tag{5.45}$$

Nous appelons cette phase une phase d'exploration. Une fois que $\mathscr{B}^t_n$ a été formé, l'algorithme entre dans une phase d'exploitation. Il répartit les requêtes générées sur un sous-ensemble $\mathscr{B}^{t*}_n \subseteq \mathscr{B}^t_n$ tel que :

$$m^* \in \underset{m \in \mathscr{B}^t_n}{\arg\min} \bar{\sigma}_{\delta^t_{m,n}} \tag{5.46}$$

Algorithm 6 Online Learning Algorithm n

Input : T, F^t
$regret = \infty$
Initialiser le match : pour chaque $m \in \mathscr{B}_n^t$ et tout χ_t, et tout $\Lambda(m, \delta_{m,n}^t) = 0$
Initialiser les estimations : pour chaque $m \in \mathscr{B}_n^t$ et tout $\delta_{m,n}^t \in \chi_t$, mettre $\bar{\sigma}_{\delta_{m,n}^t} = 0$
while $regret \geq \varepsilon$ **do**
  for pour chaque $t = 1, \cdots, T$ **do**
    Observer le contexte du fog $\chi_t = \{\delta_{m,n}^t\}_{n \in C_k}$
    Calculer l'ensemble des nœuds fog sous-explorésUE_{χ_t}
    if $UE_{\chi_t} \neq \varnothing$ **then**
      $u^t \leftarrow |UE_{\chi_t}(t)|$
      if $u^t \geq B_n^t$ **then**
        Sélectionner $m_1, \ldots, m_{B_n^t}$ de UE_{χ_t}
      else
        Sélectionner $m_1, \ldots, m_{u^t}$ comme le u^t nœud fog de UE_{χ_t}
        Sélectionner $m_{u^t+1}, \ldots, m_{B_n^t}$ comme les $B_k^{t+1} - u^t$ nœuds fog en utilisant l'équation (5.45)
      end if
    else
      Déterminer B_k^{t*} en utilisant l'équation (5.46)
    end if
    Observer les données reçues $cost_{m,k}$ du nœud fog $m_j \in B_k^{t*}$
    for pour chaque $m \in B_n^{t*}$ **do**
      Construire $\mathscr{H}_t = \mathscr{H}_t \cup \{h_{m,k}\}$
      Calculer le $regret$
    end for
  end for
end while

5.5 Évaluation des performances

Dans cette section, nous effectuerons des simulations approfondies pour évaluer les performances des approches proposées. Dans un premier temps, nous étudierons l'approche centralisée, c'est-à-dire les trois contributions proposées (IGA, IGAI et IGAG). Dans un deuxième temps, nous évaluerons les approches implémentées dans le fog (FNC, FN-FIX et FN-OPT). Dans un troisième temps, afin d'évaluer la validité et l'efficacité de l'approche implémentée dans les objets, nous simulerons l'algorithme OLA. Nos scénarios d'évaluation sont caractérisés par une taille de problème variable, permettant d'analyser l'évolutivité de nos approches avec diverses contraintes d'utilisation afin d'étudier leur comportement dans des situations plus difficiles.

5.5.1 Configuration de la simulation

Nous supposons que notre système comprend dix nœuds fog, un DC cloud et un nombre fixe d'objets. Les requêtes (Location Based Services) sont générées à partir d'un jeu de données de trajectoire T-Drive [95]. L'ensemble de données T-Drive a collecté les trajectoires GPS de 10 357 taxis opérant dans la ville de Beijing au cours d'une période d'une semaine, en 2008. Nous sélectionnons de manière aléatoire des taxis dans T-Drive et regroupons leurs requêtes d'arrivée, ce qui simule la collecte de données par les nœuds de fog à partir des objets. Avec l'étude intensive des modèles d'arrivée des requêtes, l'arrivée est approximée à un processus de Poisson avec un ratio d'arrivée λ (requêtes par seconde). Pour notre configuration du système nous supposons que les dispositifs fog sont Raspberry Pi 2 modèle B v1.1, avec un processeur ARM A7 à quatre cœurs à 900 MHz, 1 Go de RAM et une carte réseau de 100 Mbps. Pour le DC du cloud, nous définissons la configuration suivante : un processeur Intel Xeon E5 à deux cœurs, 2 GHz, 7 Go de RAM et une carte réseau Gigabit. Deux configurations de ressources sont utilisées pour les périphériques IoT : une petite, une avec 96 périphériques Pi et quatre machines virtuelles Azure pour un total de 100 ressources, et une grande capacité avec 960 périphériques Pi et des machines virtuelles 40 Azure. La taille des requêtes générées à partir des objets IoT est uniformément distribuée dans $[1;8]$ Mbits. La latence de transmission d'un paquet de données est basée sur le temps d'aller-retour (rtt) entre les objets et les nœuds fog. Par ailleurs, le délai de transmission entre le fog et le cloud est calculé en divisant leur distance par la vitesse de la lumière sur un câble à fibres optiques. Toutes les informations relatives à la consommation d'énergie dans le DC ou les nœuds fog ont été recueillies à partir de [96].

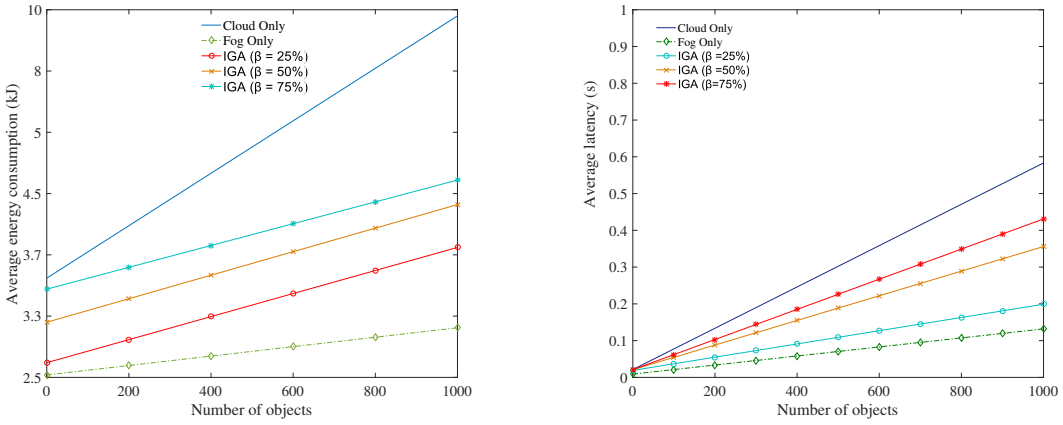
En raison de la nature non déterministe de notre système, nous avons considéré 20 exécutions, chacune avec un état de file d'attente aléatoire initialisé différemment, et avons choisi la moyenne des solutions trouvées.

5.5.2 Analyse des performances de la solution implémentée dans le cloud

Pour l'évaluation des performances de nos contributions, nous étudions deux scénarios. En plus d'IGA, nous simulons également deux approches dans le premier scénario : (a) un algorithme décentralisé dans lequel les dispositifs IoT forment des clusters, puis envoient toutes les requêtes au nœud fog le plus proche - cette approche est inspirée de la forme [97] ; (b) un algorithme de déchargement fixe qui est une version améliorée de l'état de la technique développée dans [98], appelé GBD, qui repose sur une approche approximative pour résoudre le problème primordial (PP) par décomposition et formulation de trois sous-problèmes (SP) et de trois sous-systèmes correspondants (dispositifs IoT, nœuds fog et cloud). Plus précisément, l'algorithme de pointe [98] est centralisé, l'offloading des tâches est confiné dans un seul saut et les tâches doivent être offloader, traitées et retournées dans un seul intervalle de temps. Au cours des simulations, nous définissons β le taux des requêtes transmises des nœuds fog au cloud, dans la plage $[0.25-0.75]$. Nous comparons aussi les résultats de IGA avec deux stratégies de placement simple : Fog-Only et Cloud-Only. Dans l'approche Fog-Only, nous considérons que le fog computing dispose de ressources infinies sans surcharge de réseau. Dans Cloud-Only, nous traitons toutes les requêtes sur le DC, en analy-

Approche	250 (users)	500 (users)	750 (users)	1000 (users)
IGA	2.17 (s)	76.32 (s)	107.65 (s)	458.86 (s)
IGAI	0.052 (s)	0.79 (s)	1.89 (s)	5.55 (s)
IGAG	0.31 (s)	0.94 (s)	2.716 (s)	5.75 (s)

TABLEAU 5.1 – Comparaison des temps de calcul entre IGA, IGAI, et IGAG.



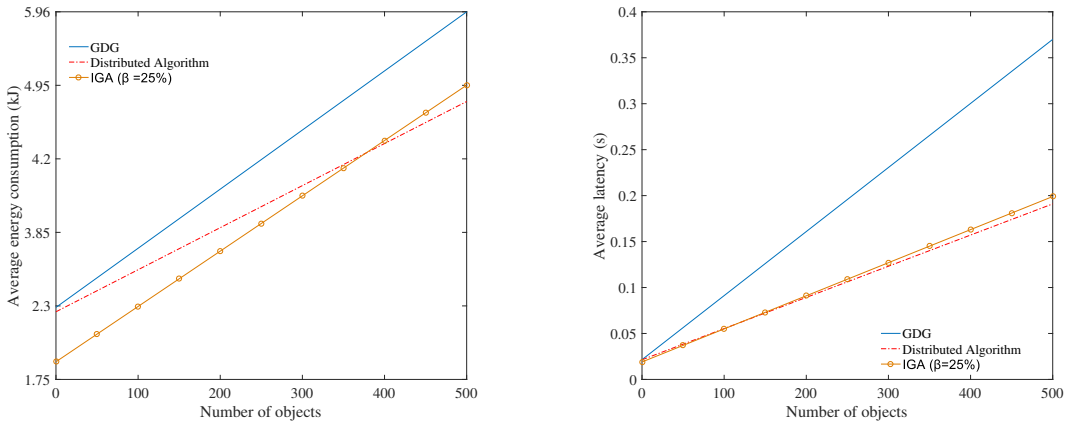
(a) (b)

FIGURE 5.8 – (a) Énergie moyenne consommée par rapport au nombre d'objets et à β. (b) Délai moyen par rapport au nombre d'objets et à β.

sant à nouveau la latence du réseau une seule fois, du bord au cloud. Bien que cela soit impossible dans des scénarios réels, ces approches de base permettent de fixer des limites à nos solutions lorsqu'elles ne disposent pas de ressources limitées et de contraintes de latence réseau. Pour la deuxième étude de cas, nous évaluons les performances système des méta-heuristiques des trois contributions proposées (IGA, IGAI et IGAG).

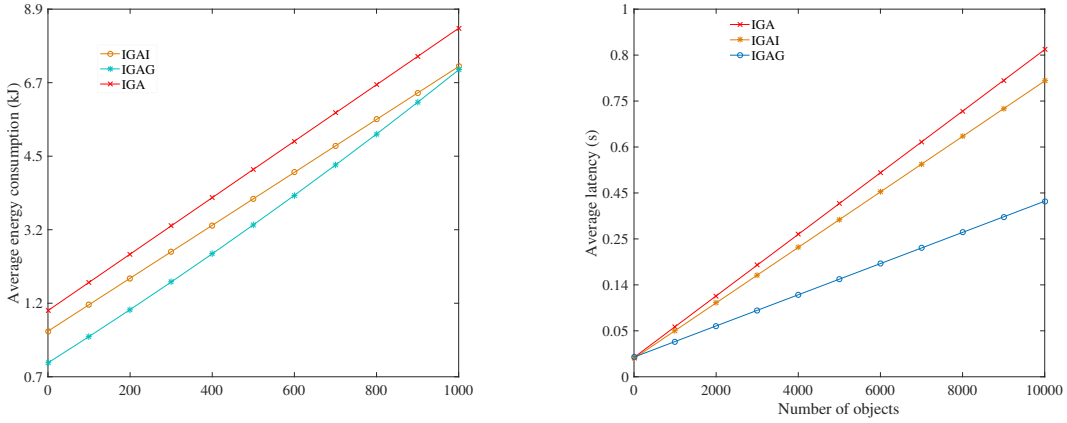
Le temps de calcul de l'approche optimale (IGA) et les algorithmes heuristiques (IGAI et IGAG) sont comparés dans le tableau 5.1 avec $\beta = 50\,\%$. Bien que l'algorithme heuristique fournisse la solution sous-optimale en quelques secondes, le temps de calcul de l'approche optimale augmente rapidement avec le nombre de requêtes. Cela tient à la complexité d'IGA comparée à celle d'IGAI et IGAG, ce qui est en accord avec notre discussion précédente. Nous soulignons aussi que le temps d'exécution d'IGA grandit de manière exponentielle avec le nombre de requêtes et que le temps d'exécution d'IGAI et d'IGAG grandit de manière polynomiale.

Pour analyser les performances de l'algorithme IGA proposé, nous comparons l'énergie consommée et le délai moyen dans trois cas de fog ($\beta = 25\,\%, 50\,\%$ et $75\,\%$) de la figure 5.8a et 5.8b. IGA est utilisé pour créer une liste de préférences pour le couplage entre les nœuds fog et les objets. Dans la première série d'expériences, nous comparons les résultats IGA à ceux des solutions Cloud-Only et Fog-Only. En comparant le délai moyen des figures 5.8b pour les trois modes principaux ($\beta = 25\,\%, 50\,\%$ et $75\,\%$), il est clair que lorsque $\beta = 25\,\%$, IGA offre le délai le plus bas. Ainsi, les objets IoT tirent pleinement parti des ressources fog. À mesure que le nombre d'objets augmente, l'utilisation du fog augmente. L'architecture de Fog-Only améliore également la consommation d'énergie et le délai par rapport aux performances de l'approche Cloud-Only, comme illustrée dans la figure 5.8b. Ceci illustre une relation intéressante entre la consommation d'énergie et le délai avec β. Les deux figures montrent les performances d'IGA en matière de réduction de l'énergie consommée (ou le délai) lorsque la probabilité d'envoyer des requêtes aux nœuds fog varie. Même si la simulation ne le montre pas, nous supposons que, lorsque l'utilisation du fog atteint



(a) (b)

FIGURE 5.9 – Analyse de l'énergie consommée et de la latence du service de IGA contre GDG et Distributed Algorithm.



(a) (b)

FIGURE 5.10 – Analyse de l'énergie consommée et de la latence du service de IGA contre IGAI et IGAG.

un seuil, elle ne peut pas répondre à toutes les requêtes. Ainsi, une fois qu'un certain nombre de requêtes a été généré, il est préférable d'augmenter β.

Les figures 5.9a et 5.9b illustrent les performances d'IGA (avec β = 25%) par rapport à la solution distribuée et à l'algorithme GBD. Nous pouvons voir que sous une faible charge, les comportements de IGA et de Distributed Algorithm sont très similaires en termes de délai et d'énergie consommée. Cela tient à ce que la plupart des requêtes des objets sont envoyées aux nœuds fog les plus proches capables de gérer toutes les requêtes. On constate également qu'avec l'augmentation du nombre d'objets IoT, IGA dépasse les algorithmes GDG et Distributed Algorithm en termes de consommation d'énergie moyenne et de délai. Cela est dû au fait que notre solution a tendance à scinder toutes les requêtes adressées aux nœuds fog adjacents, au lieu d'inonder le nœud fog le plus proche, comme le font les deux autres solutions. Ainsi, nous constatons une meilleure performance en matière de consommation d'énergie. Toutefois, lorsque le nombre d'objets et le nombre de requêtes dans le système augmentent, les performances IGA deviennent légèrement inférieures à celles de l'algorithme distribué. En effet, le délai de transmission des requêtes entre les objets IoT et les nœuds fog est plus important dans l'IGA, ce qui augmente l'énergie moyenne et le délai.

Dans les figures 5.10a et 5.10b, nous comparons les performances d'IGA avec IGAI et IGAG en termes de consommation d'énergie et de délai lorsque β = 50%. On remarque que les performances des deux solutions sont assez proches de celle de l'approche IGA. Les figures 5.10a et

5.10b montrent également qu'IGAG fonctionne mieux que IGAI en termes de délai et d'énergie consommée, quand nous nous attendions à ce que l'algorithme incrémental fonctionne mieux que l'algorithme global. Le grand nombre de requêtes qui doit être envoyé au cloud peut entraîner une explosion de l'espace de recherche et une solution sous-optimale. Avec une plus grande configuration (nombre croissant d'objets), cette explosion spatiale sera visible pour IGAI, IGAG, ainsi que pour IGA.

5.5.3 Analyse des performances de la solution implémentée dans le fog

Sur la base de la configuration déjà détaillée précédemment, les deux principaux paramètres de configuration matérielle que nous modifierons au cours de notre évaluation seront les suivants : n le nombre de serveurs et s la vitesse des serveurs. Dans cette expérience, nous considérons trois échelles de scénario différentes, déterminées par le nombre de nœuds fog dans une FI T et le nombre total de nœuds fog K pour chaque nœud fog avec T = 6, 12, 24 et K = 12, 24, 24.

Pour évaluer les performances des trois solutions implémentées dans le fog, nous les comparons avec des propositions de la littérature. Le premier algorithme proposé s'appelle Fog Node-Cloud Collaboration (FNC), les deuxième et troisième propositions sont Fog Node-Fog Node Coalition avec un nombre fixe de serveurs (FN-FIX) et avec un nombre optimal de serveurs (FN-OPT). Nos adaptations des travaux dans [93] et [98] portent l'appellation all fog processing (AFP) et l'autre mode possible est appelé light fog processing (LFP). En plus de (AFP) et (LFP), nous utiliserons l'algorithme IGA qui porte l'appellation Best Fit (BF) proposée dans la section 5.2, qui est une des approches les plus simples qui ne tient. BF fonctionne simplement en choisissant la meilleure combinaison de requêtes et nœuds fog; un utilisateur n'effectue pas dans ce cas de réservation de ressources spécifique, mais fournit plutôt une limite de temps de réponse pour chacune de ses requêtes. BF servira de base à notre évaluation.

Nos scénarios d'évaluation sont caractérisés par une taille de problème variable permettant d'analyser l'évolutivité de nos approches, ainsi que par différentes configurations ayant des contraintes d'utilisation variables pour étudier leur comportement dans des situations plus difficiles. Pour chaque scénario, nous soulignons le nombre de nœuds actifs requis et l'amélioration de nos approches par rapport à notre base BF et aux approches inspirées de la littérature.

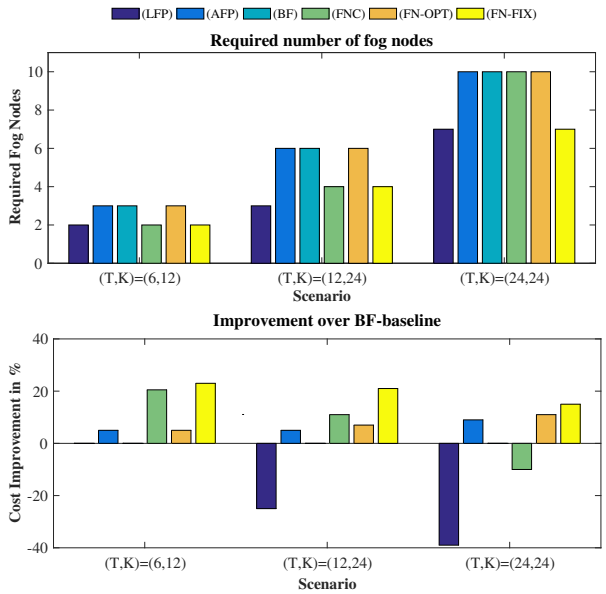


FIGURE 5.11 – Résultats de la simulation lors de la variation de l'échelle du scénario (T,K) - charge légère.

En analysant les figures 5.11, 5.12 et 5.13, nous pouvons voir que l'augmentation du nombre de nœuds fog par FI entraîne une augmentation proportionnelle du nombre de nœuds fog actifs nécessaires pour une solution optimale dans FN-OPT. Plus intéressant, cependant, les performances

de notre approche FN-FIX sont nettement supérieures à celles des autres méthodes. Les améliorations en coûts énergétiques réalisées par FN-FIX par rapport à BF et les autres solutions varient entre 6 % et 32 %, selon le scénario. La méthode FN-OPT, en revanche, ne semble trouver des solutions adéquates que dans des scénarios avec une charge de travail plus élevée (figure 5.13). Notre hypothèse est la suivante : dans ces scénarios, la marge de manœuvre est plus grande pour réduire le nombre de nœuds fog actifs. Gardons à l'esprit que, dans FN-OPT, la recherche d'un nombre optimal de serveurs guide la recherche pour le partage de ressources, mais elle est limitée par la FI en termes d'attribution de serveur. De plus, nous observons que FNC peut concurrencer FN-FIX, bien que cela ne semble être le cas que pour les scénarios à petite échelle et uniquement avec une charge légère. Nous attribuons cela au nombre plus élevé de contraintes dans des scénarios plus vastes, difficiles à satisfaire. Nos solutions (FN) évitent généralement ce problème, car FN-FIX et FN-OPT sont susceptibles de trouver une solution réalisable à chaque itération. Nous remarquons qu'AFP indique une amélioration constante par rapport à BF, qui reste toutefois comprise entre 2 % et 6 %. Comme prévu, l'approche LFP produit les plus mauvais résultats. Dans quelques cas, le LFP surmonte BF avec un faible pourcentage, ce qui est négligeable dans le scénario à lourde charge. Enfin, nous constatons que généralement, BF nécessite plus de nœuds fog actifs que LFP. Si nous observons le nombre de nœuds fog alloué par LFP il est possible d'en comprendre. En effet, LFP assigne le serveur fog le plus proche à la requête de l'utilisateur, ce qui d'une part, permet de réduire le délai de transmission, mais ne garantit pas, d'autre part une bonne répartition de la charge de travail et n'augmente pas le coût total en énergie et ni les délais. Nous nous attendons à ce que cet effet diminue avec les scénarios comportant un nombre plus élevé d'utilisateurs.

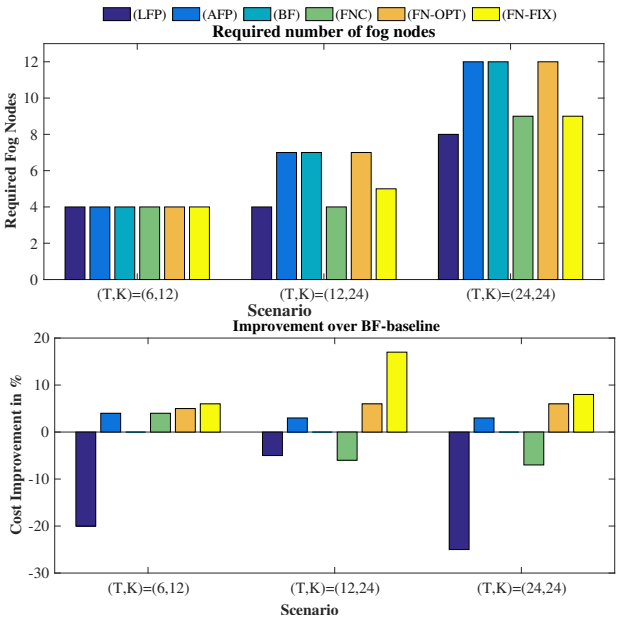


FIGURE 5.12 – Résultats de la simulation lors de la variation de l'échelle du scénario (T, K) - charge moyenne

Avec nos prochaines séries de simulations, nous souhaitons appréhender les performances de nos méthodes lorsque nous modifions certaines contraintes de ressources. Nous considérons différents choix de vitesse du serveur, les évaluons pour les différents scénarios et fournissons les résultats correspondants dans la figure 5.14. Nous pouvons constater que la méthode FN-OPT concurrence alors FN-FIX. Comme tous les nœuds fog ont la même configuration, il n'existe pas de différence entre les résultats de LFP et ceux de BF. Néanmoins, l'approche FNC peut être grandement améliorée. Comme le montre la figure 5.14, nous pouvons observer que FN-OPT a la meilleure performance lorsque la vitesse du serveur est fixe ou choisie aléatoirement dans l'intervalle [1.2-1.4], comme c'est le cas pour le dernier groupe de colonnes. Il est intéressant de noter que les performances de FN-FIX se détériorent lorsque nous utilisons une vitesse variable. Une telle variable exacerbe la recherche d'un nombre adéquat de serveurs pour une coalition ap-

TABLEAU 5.2 – L'amélioration de FN-OPT (%) par rapport à FNC, en variant (T,K) et l'intensité de la charge de travail

Charge	(6,12)	(12,24)	(24,24)
(FN-OPT) Amélioration-Charge Moyenne	10.49	10.38	10.49
(FN-OPT) Amélioration-Charge élevée	20.84	35.12	40.51

propriée. Nous remarquons des effets similaires lorsqu'on utilise différentes configurations pour chaque fog. Nous omettons les résultats pour la vitesse de 1,2, car aucune de nos méthodes ne pourrait trouver d'amélioration dans ce cas. Cela signifie que la vitesse des serveurs n'est pas suffisante pour exécuter les charges de travail.

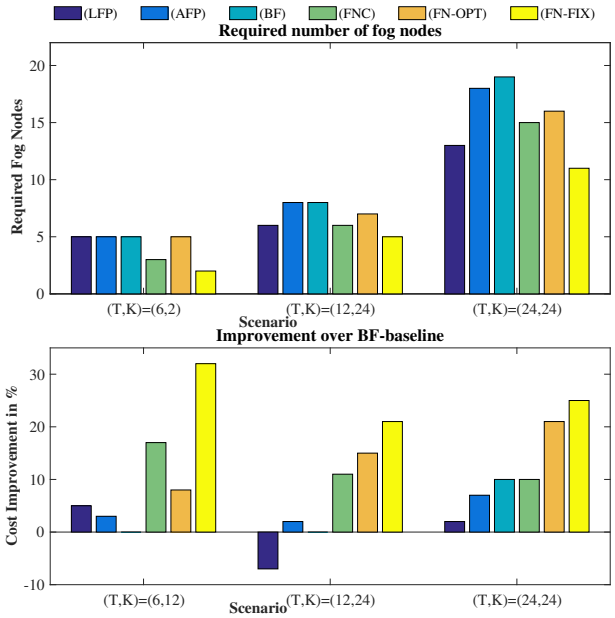


FIGURE 5.13 – Résultats de la simulation lors de la variation de l'échelle du scénario (T,K) - charge élevée

Enfin, nous cherchons à comparer les approches FNC et FN-OPT. Pour une valeur optimale de s et n_i pour chaque méthode, les résultats correspondants sont donnés dans le tableau 5.2 et la figure 5.15. Tout d'abord, nous remarquons que généralement, les performances de FN-OPT surpassent celles de FNC de 10,38 % à 10,49 %. Ensuite, déterminer le temps de recherche des valeurs optimales de s, n_i pour les deux méthodes se révèle peu pertinent pour notre scénario (temps d'exécution des deux algorithmes). Nous pouvons le constater pour un exemple de scénario de forte charge dans la figure 5.15, où les deux méthodes ont trouvé une solution proche de leur solution finale respective dans environ 12 000 évaluations de fonctions, ce qui correspond à une optimisation du temps d'une heure sur notre système de test.

5.5.4 Analyse des performances de la solution implémentée dans les objets

Pour évaluer l'approche discutée dans 5.4, nous examinons la consommation d'énergie et le délai dans différents scénarios pour l'algorithme 6 sans cloud et complet (avec un serveur cloud). Par la suite, nous comparons les performances de l'algorithme 6 et celles de l'algorithme IGA proposé dans la section 5.2 ainsi que la solution FN-OPT. Nous basons nos simulations sur la configuration déjà détaillée ; les paramètres de simulation que nous modifions au cours de notre évaluation sont les suivants : N le nombre d'utilisateurs, M le nombre de nœuds fog, λ_n le taux d'arrivée des requêtes et f_n^m la fréquence du CPU.

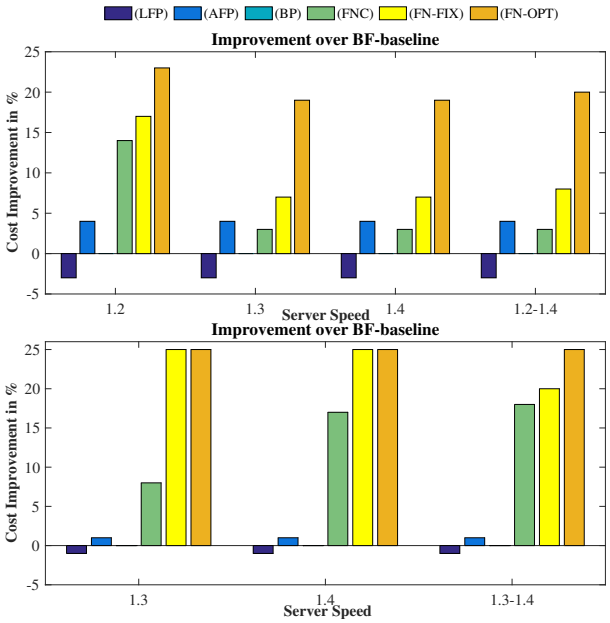


FIGURE 5.14 – Résultat des simulations lors de la variation de la vitesse du serveur dans le scénario (T, K) = (12,24). Charge moyenne, charge élevée.

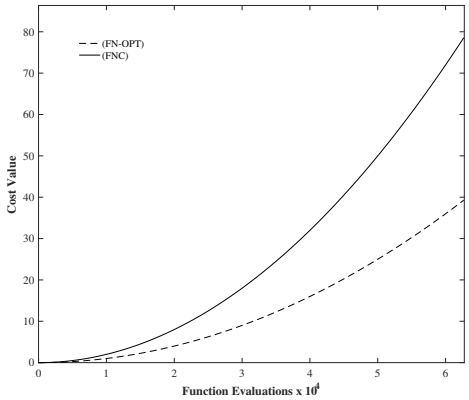
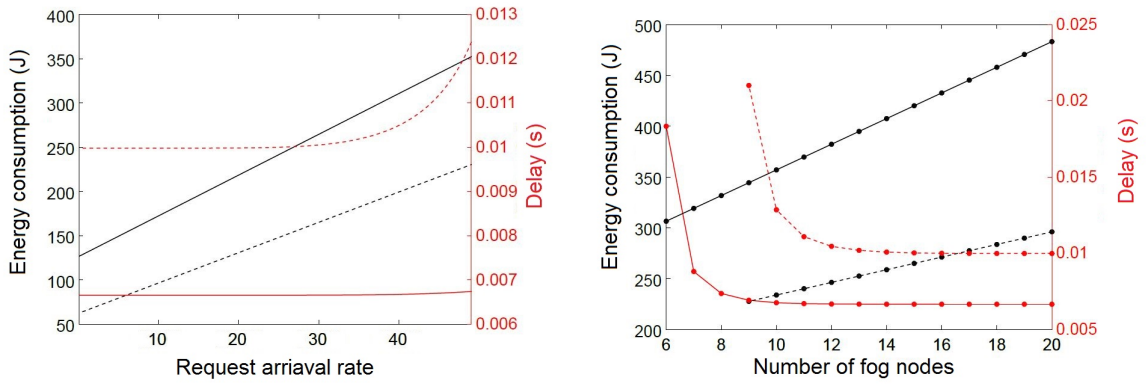


FIGURE 5.15 – Valeur du coût par rapport à la fonction d'évaluation, avec (T, K) = 24

OLA sans cloud :

Tout d'abord, nous considérons un réseau avec dix nœuds fog et aucune possibilité de déchargement sur le DC. Chaque requête a une taille de 1 Mo et une valeur $\theta = 0.3$. La durée de la période examinée T est fixée à 60 s. Le délai et la consommation d'énergie du système sont comparés au nombre de requêtes dans la figure 5.16a. Cela montre que diminuer la fréquence du processeur diminue la consommation d'énergie mais augmente le délai. La figure 5.16b représente la consommation d'énergie et le délai par rapport au nombre de nœuds fog pour un nombre de requêtes $\lambda_n = 50$. Nous pouvons voir que plus le nombre de nœuds fog diminue, plus le délai est long. Les retours diminuent, car le délai causé par la mise en file d'attente disparaît rapidement et le délai de traitement ne change pas avec le nombre de nœuds fog. De plus, si l'on augmente le nombre de nœuds fog, la consommation d'énergie augmente également.

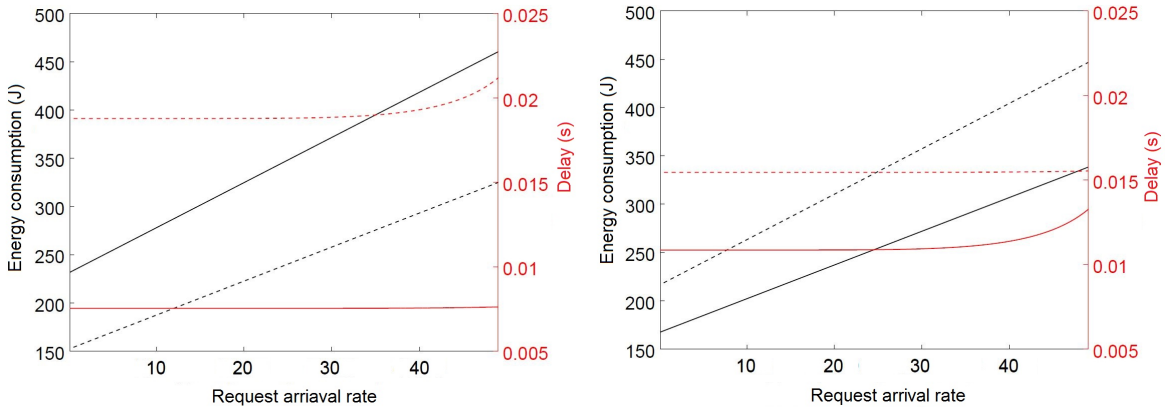
Les figures 5.17a et 5.17b détaillent la consommation totale d'énergie (issue des calculs et de la transmission) et les délais dans un système fog sans cloud avec un CPU et une bande passante variables. Nous remarquons que le délai moyen est faible - oscillant autour de 10 à 20 ms. Il existe également un lien évident entre la consommation d'énergie et le délai : une fréquence d'horloge



(a) Impact du taux d'arrivée des requêtes λ_n sur l'énergie consommée et le délai, avec 10 nœuds fog.

(b) Impact du nombre des nœuds fog sur l'énergie consommée et le délai, avec $\lambda_n = 50$.

FIGURE 5.16 – L'énergie consommée et le délai par rapport à λ_n et M.



(a) Impact du taux d'arrivée des requêtes λ_n sur l'énergie consommée et le délai, avec M = 10.

(b) Impact du nombre des nœuds fog sur l'énergie consommée et le délai, avec $\lambda_n = 50$.

FIGURE 5.17 – L'énergie consommée et le délai en fonction de λ_n et M (ligne pointillée - 2 GHz, ligne continue - 3 GHz).

est plus élevée et une largeur de bande plus courte permet de réduire le délai et la consommation d'énergie. Compte tenu de cela, la performance de la combinaison des nœuds fog qui fonctionnent à une fréquence de 2 GHz et à une liaison frontale de 10 Gbit surpasse celle des nœuds fog qui fonctionnent à une bande passante de 3 GHz et 1 Gbit dans les deux métriques estimées dans des conditions de trafic données.

OLA :

Nous considérons maintenant une architecture fog/cloud avec dix nœuds fog et un DC. La consommation d'énergie et le délai moyen sont représentés sur la figure 5.18 par rapport à θ, le poids de l'énergie dans notre fonction d'optimisation. Quelques observations intéressantes peuvent être formulées. Premièrement, la charge (lourde, médium ou légère) joue un rôle clé dans la détermination de l'utilité du transfert des requêtes au DC. Dans un scénario à charge lourde, l'utilisation de cloud réduit à la fois la latence et la consommation d'énergie. Nous remarquons aussi que la variation de θ influence légèrement le délai et la consommation d'énergie du système ; cela peut s'expliquer par le fait que le système dispose de ressources suffisantes pour traiter toutes les requêtes dans la couche fog dans ce scénario. Des résultats plus intéressants peuvent être observés dans le scénario à charge moyenne : effectuer des calculs dans le cloud réduit la consommation d'énergie, tandis que les effectuer dans le fog entraîne une latence inférieure. Dans ce cas aussi, la variation de θ influence peu la consommation d'énergie et le délai du système. Dans le

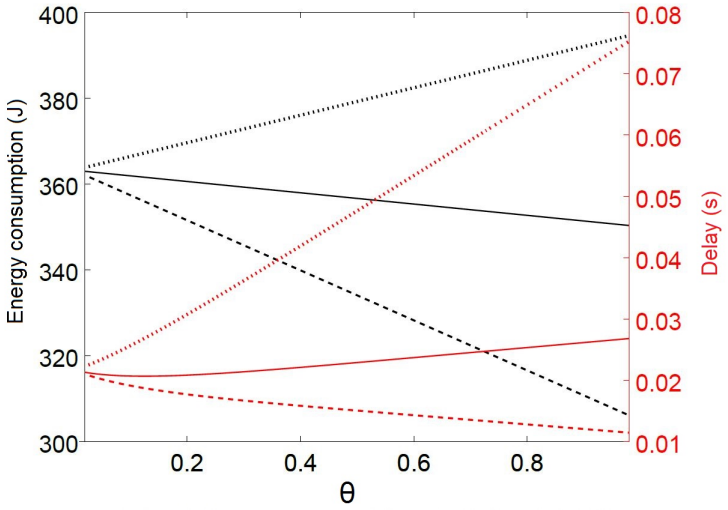


FIGURE 5.18 – L'énergie consommée et le délai en fonction de θ, avec M = 10 (lignes pointillées, pleines et pointillées - scénarios à charge légère, moyenne et lourde).

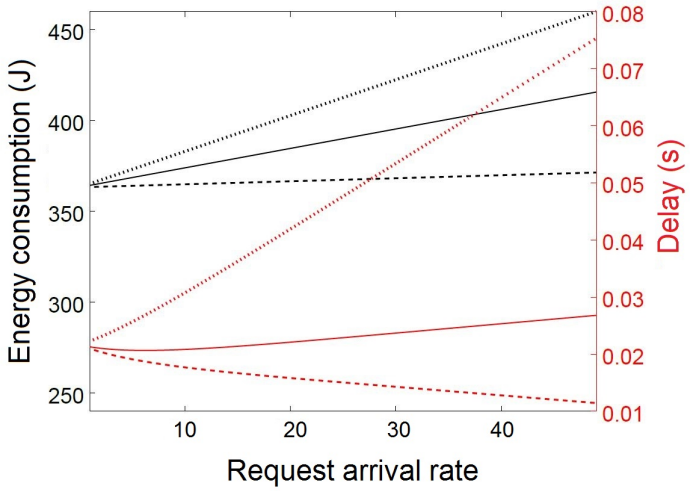


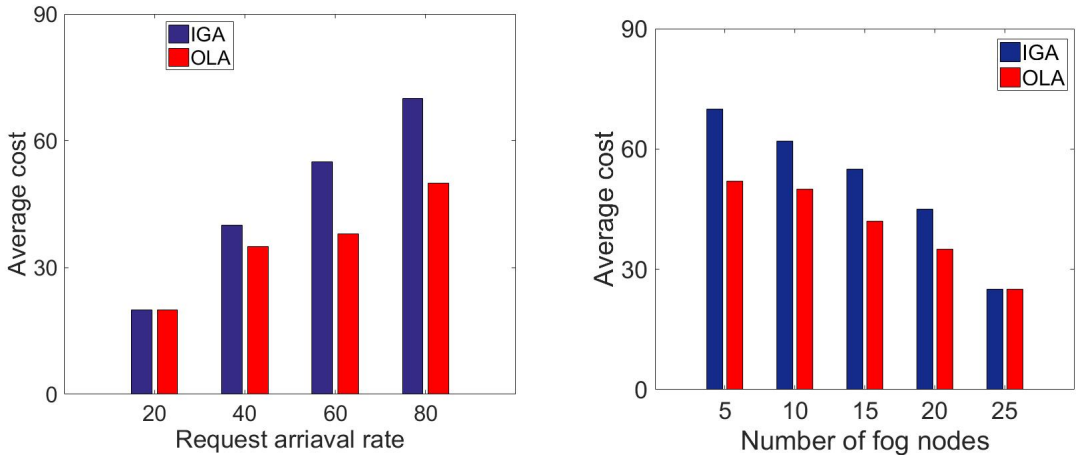
FIGURE 5.19 – L'énergie consommée et le délai en fonction de λ_n, avec M = 10 (Lignes pointillées, pleines et pointillées - scénarios à charge légère, moyenne et lourde).

dernier scénario (charge lourde), on remarque que l'énergie consommée du système augmente de manière linéaire contrairement au délai, qui augmente de manière exponentielle.

L'impact de λ_n sur les performances du réseau est également étudié dans le cas d'un système fog/cloud. Les résultats pour M = 10 et θ = 0.3 sont illustrés dans la figure 5.19. La part correspondante au délai total et de la consommation d'énergie consacrée au calcul et à la communication est représentée dans la figure. Les requêtes traitées dans les nœuds fog sont plus rapides et consomment moins d'énergie que celles traitées dans le cloud. Les délais et la consommation d'énergie causés par le transport des requêtes l'emportent sur les ressources de calcul de serveur cloud. Au contraire, lorsque les requêtes déchargées nécessitent des calculs lourds, il est avantageux d'envoyer ces requêtes au cloud, car un niveau élevé de charges signifie un coût de communication inférieur au coût de calcul.

Comparaison entre OLA, IGA et FN-OPT :

Les figures 5.20a et 5.20b illustrent les valeurs coûts dans OLA et IGA en fonction de λ_n le taux d'arrivée des requêtes et M le nombre de nœuds fog; comme montre la figure 5.20a, OLA surpasse de 20 % IGA . Nous faisons en outre varier le nombre de nœuds fog et étudions les performances



(a) Impact du taux d'arrivée des requêtes λ_n sur le coût dans IGA et OLA, avec M = 10.

(b) Impact du nombre des nœuds fog sur le coût dans IGA et OLA, avec $\lambda_n = 50$.

FIGURE 5.20 – Le coût dans OLA et IGA en fonction de λ_n et M.

d'OLA. Comme montre la figure 5.20b, le coût dans OLA est inférieur de 18 % par rapport à celui dans IGA. De plus, la figure 5.20b montre que le coût diminue considérablement lorsque le nombre de nœuds fog augmente.

Nous comparons dans cette partie les trois solutions présentées dans ce chapitre (IGA , OLA et FN-OPT) dans un scénario à forte charge. La figure 5.21 illustre le coût total dans les différentes approches. Le coût dans FN-OPT est toujours inférieur au coût dans IGA en raison du comportement stratégique des nœuds fog dans ce scénario. Cependant, comme on peut le constater, le coût total dans OLA est proche du coût FN-OPT, bien que le coût de ce dernier soit toujours supérieur. Cela montre que OLA est proche de la solution optimale. Cependant, même s'il n'est pas mentionné, le temps d'exécution d'OLA est clairement supérieur à celui de FN-OPT.

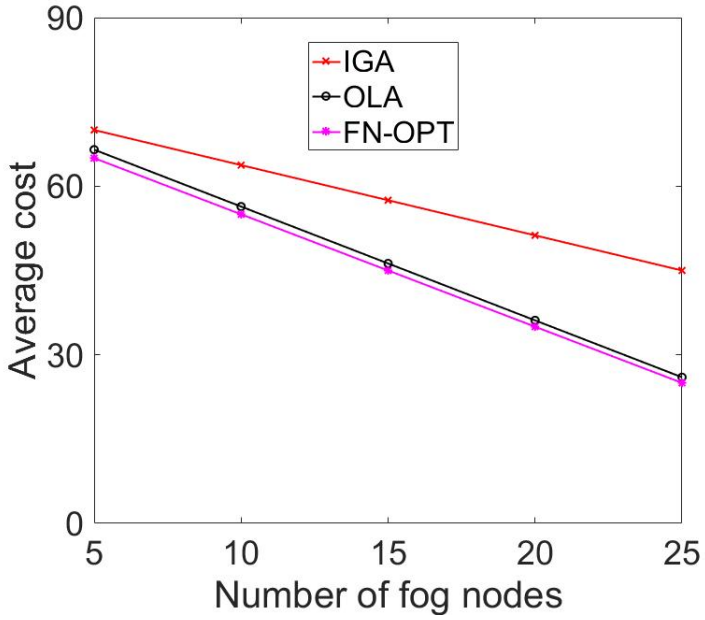


FIGURE 5.21 – Le coût dans OLA , IGA et FN-OPT en fonction de M.

5.6 Conclusion

Dans ce chapitre, nous avons mené une étude formelle du problème de la mise en œuvre des applications IoT à contenu dynamique dans un système fog/cloud. Nous avons caractérisé les principaux défis de la communication, du calcul en termes de délai et d'énergie consommée avec un modèle de coût total. Nous avons d'abord proposé une solution centralisée basée sur un algorithme génétique. Ensuite, nous avons proposé deux heuristiques pour améliorer les performances de celui-ci. Dans un second temps, nous avons présenté une solution décentralisée basée sur les modèles de files d'attente, pour laquelle nous avons mis en avant une stratégie de collaboration entre les nœuds fog et le cloud et les nœuds fog entre eux. Enfin, nous avons introduit une solution décentralisée basée sur les algorithmes de machine learning, qui met l'accent sur le comportement égoïste des utilisateurs. Chaque utilisateur essaie de minimiser le coût de traitement de ses requêtes en apprenant la stratégie des autres. Nous avons aussi comparé les performances obtenues par nos approches proposées avec les performances des solutions de base et des solutions inspirées de la littérature.

Plusieurs questions de recherche restent en suspens, notamment le développement de la collaboration des utilisateurs, ou encore l'incorporation de technologies comme la mise en cache qui peuvent être considérées dans un travail futur.

Conclusion générale et Perspectives

5.7 Conclusion et Discussion

Dans ce travail, nous avons proposé des stratégies permettant une allocation de ressources et une distribution de requêtes optimales dans un système fog/cloud. Notre objectif était de concevoir des approches à la fois, hors ligne et en ligne, et qui permettent de minimiser la latence et l'énergie consommée, avec une information à la volée de l'état du système (sous incertitude), tout en prenant en compte un modèle de délai et d'énergie réaliste. La première partie de ce manuscrit a été consacrée à l'état de l'art et à la définition de certaines notions utilisées dans la thèse tandis que nos contributions ont été détaillées dans la seconde partie.

Le chapitre 1 a introduit une définition de l'IoT, une courte discussion sur les quelques applications de cette technologie, ainsi qu'une présentation des nombreuses évolutions qui ont été nécessaires pour aboutir au fog compting. Il décrit également les propriétés et les caractéristiques des infrastructures de fog computing, permettant de fournir une puissance de calcul accessible aux objets connectés situés au bord du réseau. Dans ce contexte, les algorithmes présentés dans le chapitre 2 couvrent le calcul, le stockage, la distribution et la consommation d'énergie dans l'infrastructure fog. Nous avons fourni une évaluation critique de chaque contribution examinée à la lumière de critères bien définis et bien motivés. Pour chaque article, nous avons décrit son champ d'application, l'approche suivie, la méthodologie d'évaluation, son apport majeur ainsi que les critères auxquels il répond.

Dans le chapitre 3, nous avons présenté les caractéristiques de notre système fog/cloud dans le contexte des applications IoT. Après avoir détaillé notre problématique et formulé nos hypothèses de travail, nous avons présenté : (1) l'architecture fog/cloud ainsi que les composants de ce système ; (2) deux modèles pour mesurer l'énergie consommée et le délai pour l'approvisionnement d'une application IoT.

Notre première solution a été présentée dans le chapitre 4, où nous avons caractérisé les principaux défis de la communication et de calcul dans un système IoT-Fog-Cloud avec un modèle de coût complet. De même, nous avons proposé un nouvel algorithme basé sur la résolution itérative, Iterative Request Assignation Algorithm (IRAA). La performance de l'algorithme proposé était confirmée par de nombreuses simulations. Bien que le passage à large échelle ne soit pas évident dans ce cas, les simulations démontrent que la fonction objective aide à diminuer le coût de traitement des requêtes des utilisateurs d'applications IoT. Dans un second temps, nous avons proposé un algorithme décentralisé Decentrilized Request Distribution algorithm (DRD) basé sur la théorie des jeux. Dans notre modèle, les objet sont en concurrence pour maximiser l'efficacité de leur utilisation des ressources et satisfaire les exigences de qualité de service. Afin d'atteindre leurs objectifs, nous avons modélisé le problème de la gestion des ressources et de la distribution des requêtes des utilisateurs en tant que problème d'équilibre de Nash généralisé, puis nous avons modélisé et appliqué une méthode de Newton semi-lisse pour résoudre le problème. Nous avons également démontré que la solution qui correspond à l'équilibre de Nash converge vers un équilibre efficace après plusieurs itérations. L'efficacité de notre approche est démontrée par la simulation. Nous avons comparé les performances obtenues par l'algorithme proposé avec deux algorithmes simples et de deux solutions inspirées de la littérature. Nos résultats numériques montrent que l'algorithme proposé permet d'obtenir de bonnes performances.

Les approches proposées sont spécifiquement conçues pour des applications IoT à contenu statique dans les systèmes fog/cloud. En comparaison avec d'autres méthodes d'offloading et de distribution de contenu, ce travail a traité d'un large éventail de contraintes (processeur, propriétés des périphériques, bande passante et latence du réseau), qui garantissent que les requêtes des utilisateurs peuvent être exécutées correctement dans une infrastructure fog hétérogène. En considérant que les applications IoT sont en pleine expansion, les propositions IRAA et DRD permettent respectivement le passage à moyen et large échelle. En prenant en compte l'existence de liens entre les nœuds fog, IRAA permet la gestion de mobilités des utilisateurs.

Une deuxième classe d'applications IoT a été considérée dans le chapitre 5, à savoir les applications à contenu dynamique. Pour ce genre d'applications, les algorithmes doivent s'exécuter en ligne, et la distribution des requêtes se fait dès leurs générations par les objets. En particulier, un premier algorithme génétique centralisé a été proposé pour résoudre le problème du compromis entre le délai et l'énergie consommée. Des algorithmes heuristiques basés sur la solution initiale sont dérivés pour résoudre ce problème dans un temps d'exécution raisonnable. Les solutions proposées ont montré la pertinence de nos approches avec un grand nombre d'applications IoT en temps réel et à faible temps de latence. Dans la deuxième contribution présentée dans ce chapitre, nous avons proposé deux algorithmes de collaboration nœud fog-cloud (FNC) et nœud fog- nœud fog (FN-FIX et FN-OPT), avec comme objectif de minimiser le coût total de l'exécution d'applications IoT. Ici, une nouvelle conception de système est proposée, dans laquelle différents caractéristiques et théorèmes de file d'attente sont appliqués à notre système. Des simulations approfondies confirment la performance de l'algorithme proposé. Dans la dernière partie du chapitre 5, nous avons proposé un schéma de distribution des requêtes des objets dans les systèmes fog/cloud afin de minimiser les délais et l'énergie nécessaires au calcul des requêtes. Dans ce but, nous avons proposé un algorithme d'apprentissage distribué (Online Learning Algorithm ou OLA) en ligne pour répartir les requêtes de chaque objet sur le fog et le cloud de manière à minimiser le coût total de traitement et à respecter les contraintes du système. Les résultats des simulations ont montré que le schéma proposé minimise considérablement la consommation d'énergie et le délai de traitement, tout en garantissant des limites de latence des coûts.

5.8 Perspectives future

Concernant nos travaux futurs, nous espérons améliorer nos solutions en :

— *Expérimentations :* pour comparer différentes décisions de distribution, les temps de réponse des applications et l'énergie totale consommée sont simulés dans ce travail. Une expérience sur des bancs d'essai industriels (comme le banc d'essai interne d'Orange Labs introduit dans [99]) permet de mesurer les temps de réponse des applications dans des environnements réels. Une nouvelle évaluation basée sur l'expérimentation pourrait être mise en œuvre à l'avenir.

— *Contraintes plus fines :* ce travail modélise le cloud comme étant un serveur à ressources infinies, avec une large bande passante par une liaison constante avec le fog. Étant donné que les ressources des serveurs cloud varient périodiquement [100], notre modèle peut être amélioré avec des exigences de ressources temporelles.

— *Sélection de l'application :* sachant que les exigences des requêtes de services pour traiter celles-ci dans un nœud fog varient d'une application à une autre, une amélioration possible est de considérer que lorsqu'un fog nœud ne dispose pas de suffisamment de ressources pour héberger toutes les applications à placer, l'approche proposée renvoie *échec* pour indiquer que ces applications ne peuvent pas être satisfaites simultanément. Pour mieux traiter ce cas, la proposition peut être étendue avec un sélecteur d'applications, qui sélectionne un sous-ensemble d'applications à placer.

— *objectifs d'optimisation multiples :* un problème de d'offloading peut avoir plusieurs objectifs d'optimisation. Ce travail tente uniquement de réduire les délais de traitement et

l'énergie consommée pour l'approvisionnement des applications IoT. Pour améliorer les approches proposées, d'autres objectifs d'optimisation (par exemple, maximiser la disponibilité des applications déployées) peuvent être simultanément pris en compte dans des travaux futurs.

Bibliographie

[1] Neil Gershenfeld, Raffi Krikorian, and Danny Cohen. The internet of things. *Scientific American*, 291(4) :76–81, 2004. 3, 7

[2] Anne H Ngu, Mario Gutierrez, Vangelis Metsis, Surya Nepal, and Quan Z Sheng. Iot middleware : A survey on issues and enabling technologies. *IEEE Internet of Things Journal*, 4(1) :1–20, 2017. 3, 7

[3] George Suciu, Victor Suciu, Alexandru Martian, Razvan Craciunescu, Alexandru Vulpe, Ioana Marcu, Simona Halunga, and Octavian Fratu. Big data, internet of things and cloud convergence–an architecture for secure e-health applications. *Journal of medical systems*, 39(11) :141, 2015. 3, 7

[4] Jayavardhana Gubbi, Rajkumar Buyya, Slaven Marusic, and Marimuthu Palaniswami. Internet of things (iot) : A vision, architectural elements, and future directions. *Future generation computer systems*, 29(7) :1645–1660, 2013. 3, 7

[5] Adila Mebrek, Leila Merghem-Boulahia, and Moez Esseghir. Efficient green solution for a balanced energy consumption and delay in the iot-fog-cloud computing. pages 1–4, 2017. 5

[6] Adila Mebrek, Moez Esseghir, and Leila Merghem-Boulahia. Energy-efficient solution based on reinforcement learning approach in fog networks. 2019. 5

[7] Adila Mebrek, Leila Merghem-Boulahia, and Moez Esseghir. Energy-efficient solution using stochastic approach for iot-fog-cloud computing. pages 1–6, 2019. 5

[8] Daniele Miorandi, Sabrina Sicari, Francesco De Pellegrini, and Imrich Chlamtac. Internet of things : Vision, applications and research challenges. *Ad hoc networks*, 10(7) :1497–1516, 2012. 7

[9] Harald Sundmaeker, Patrick Guillemin, Peter Friess, and Sylvie Woelfflé. Vision and challenges for realising the internet of things. *Cluster of European Research Projects on the Internet of Things, European Commision*, 3(3) :34–36, 2010. 7

[10] Loic Letondeur, François-Gaël Ottogalli, and Thierry Coupaye. A demo of application lifecycle management for iot collaborative neighborhood in the fog. 2017. 7

[11] Dave Evans. The internet of things : How the next evolution of the internet is changing everything. *CISCO white paper*, 1(2011) :1–11, 2011. 8

[12] Flavio Bonomi, Rodolfo Milito, Preethi Natarajan, and Jiang Zhu. Fog computing : A platform for internet of things and analytics. pages 169–186, 2014. 8, 76

[13] Yun Chao Hu, Milan Patel, Dario Sabella, Nurit Sprecher, and Valerie Young. Mobile edge computing—a key technology towards 5g. *ETSI white paper*, 11(11) :1–16, 2015. 9

[14] Flavio Bonomi, Rodolfo Milito, Jiang Zhu, and Sateesh Addepalli. Fog computing and its role in the internet of things. pages 13–16, 2012. 9

[15] Luis M Vaquero and Luis Rodero-Merino. Finding your way in the fog : Towards a comprehensive definition of fog computing. *ACM SIGCOMM Computer Communication Review*, 44(5) :27–32, 2014. 9

[16] Rajesh Krishna Balan, Darren Gergle, Mahadev Satyanarayanan, and James Herbsleb. Simplifying cyber foraging for mobile devices. 2007. 9

[17] Tim Verbelen, Pieter Simoens, Filip De Turck, and Bart Dhoedt. Cloudlets : Bringing the cloud to the mobile user. pages 29–36, 2012. 9

[18] Michael Till Beck, Martin Werner, Sebastian Feld, and S Schimper. Mobile edge computing : A taxonomy. pages 48–55, 2014. 9

[19] Charles C Byers. Architectural imperatives for fog computing : Use cases, requirements, and architectural techniques for fog-enabled iot networks. *IEEE Communications Magazine*, 55(8) :14–20, 2017. 9

[20] Charith Perera, Yongrui Qin, Julio C Estrella, Stephan Reiff-Marganiec, and Athanasios V Vasilakos. Fog computing for sustainable smart cities : A survey. *ACM Computing Surveys (CSUR)*, 50(3) :32, 2017. 9

[21] Redowan Mahmud, Ramamohanarao Kotagiri, and Rajkumar Buyya. Fog computing : A taxonomy, survey and future directions, 2018. 9

[22] Pengfei Hu, Sahraoui Dhelim, Huansheng Ning, and Tie Qiu. Survey on fog computing : architecture, key technologies, applications and open issues. *Journal of network and computer applications*, 98 :27–42, 2017. 10

[23] Yifan Wang, Tetsutaro Uehara, and Ryoichi Sasaki. Fog computing : Issues and challenges in security and forensics. 3 :53–59, 2015. 10, 13

[24] Tom H Luan, Longxiang Gao, Zhi Li, Yang Xiang, Guiyi Wei, and Limin Sun. Fog computing : Focusing on mobile users at the edge. *arXiv preprint arXiv :1502.01815*, 2015. 10

[25] Madiha H Syed, Eduardo B Fernandez, and Mohammad Ilyas. A pattern for fog computing. page 13, 2016. 10

[26] Shanhe Yi, Zijiang Hao, Zhengrui Qin, and Qun Li. Fog computing : Platform and applications. pages 73–78, 2015. 10

[27] Mung Chiang and Tao Zhang. Fog and iot : An overview of research opportunities. *IEEE Internet of Things Journal*, 3(6) :854–864, 2016. 10, 13

[28] Niroshinie Fernando, Seng W Loke, and Wenny Rahayu. Mobile cloud computing : A survey. *Future generation computer systems*, 29(1) :84–106, 2013. 11

[29] Yaser Jararweh, Ahmad Doulat, Omar AlQudah, Ejaz Ahmed, Mahmoud Al-Ayyoub, and Elhadj Benkhelifa. The future of mobile cloud computing : integrating cloudlets and mobile edge computing. pages 1–5, 2016. 11

[30] Alexander Klemm, Christoph Lindemann, and Oliver P Waldhorst. A special-purpose peer-to-peer file sharing system for mobile ad hoc networks. 4 :2758–2763, 2003. 11

[31] Shanhe Yi, Zijiang Hao, Zhengrui Qin, and Qun Li. Fog computing : Platform and applications. pages 73–78, 2015. 11

[32] Shanhe Yi, Zijiang Hao, Zhengrui Qin, and Qun Li. Fog computing : Platform and applications. pages 73–78, 2015. 12

[33] Yuyi Mao, Changsheng You, Jun Zhang, Kaibin Huang, and Khaled B Letaief. A survey on mobile edge computing : The communication perspective. *IEEE Communications Surveys and Tutorials*, 19(4) :2322–2358, 2017. 12

[34] Yaser Jararweh, Ahmad Doulat, Omar AlQudah, Ejaz Ahmed, Mahmoud Al-Ayyoub, and Elhadj Benkhelifa. The future of mobile cloud computing : integrating cloudlets and mobile edge computing. pages 1–5, 2016. 12

[35] Yaser Jararweh, Ahmad Doulat, Omar AlQudah, Ejaz Ahmed, Mahmoud Al-Ayyoub, and Elhadj Benkhelifa. The future of mobile cloud computing : integrating cloudlets and mobile edge computing. pages 1–5, 2016. 12

[36] Athanasios V Vasilakos, Zhe Li, Gwendal Simon, and Wei You. Information centric network : Research challenges and opportunities. *Journal of network and computer applications*, 52 :1–10, 2015. 12

[37] Ben Zhang, Nitesh Mor, John Kolb, Douglas S Chan, Ken Lutz, Eric Allman, John Wawrzynek, Edward Lee, and John Kubiatowicz. The cloud is not enough : Saving iot from the cloud. 2015. 12

[38] Mahadev Satyanarayanan. Cloudlets : At the leading edge of cloud-mobile convergence. pages 1–2, 2013. 12

[39] Pengfei Hu, Huansheng Ning, Tie Qiu, Yanfei Zhang, and Xiong Luo. Fog computing based face identification and resolution scheme in internet of things. *IEEE transactions on industrial informatics*, 13(4) :1910–1920, 2017. 12

[40] Henrik Madsen, Bernard Burtschy, G Albeanu, and FL Popentiu-Vladicescu. Reliability in the utility computing era : Towards reliable fog computing. pages 43–46, 2013. 15

[41] Sam Meredith. Here's everything you need to know about the cambridge analytica scandal, 2018. 15

[42] Mohammed Al-khafajiy, Thar Baker, Hilal Al-Libawy, Zakaria Maamar, Moayad Aloqaily, and Yaser Jararweh. Improving fog computing performance via fog-2-fog collaboration. *Future Generation Computer Systems*, 100 :266–280, 2019. 17, 24

[43] Ashkan Yousefpour, Genya Ishigaki, Riti Gour, and Jason P Jue. On reducing iot service delay via fog offloading. *IEEE Internet of Things Journal*, 5(2) :998–1010, 2018. 17, 24, 27, 28

[44] Yong Xiao and Marwan Krunz. Qoe and power efficiency tradeoff for fog computing networks with fog node cooperation. pages 1–9, 2017. 17, 24, 27, 28

[45] Lina Ni, Jinquan Zhang, Changjun Jiang, Chungang Yan, and Kan Yu. Resource allocation strategy in fog computing based on priced timed petri nets. *ieee internet of things journal*, 4(5) :1216–1228, 2017. 18, 24, 27

[46] Soumya Kanti Datta, Christian Bonnet, and Jerome Haerri. Fog computing architecture to enable consumer centric internet of things services. pages 1–2, 2015. 18, 24

[47] Abbas Kiani and Nirwan Ansari. Toward hierarchical mobile edge computing : An auction-based profit maximization approach. *IEEE Internet of Things Journal*, 4(6) :2082–2091, 2017. 18, 19, 24

[48] Cássio Prazeres and Martin Serrano. Soft-iot : Self-organizing fog of things. pages 803–808, 2016. 18, 19, 20, 24

[49] Mohammad Hossein Yaghmaee Moghaddam and Alberto Leon-Garcia. A fog-based internet of energy architecture for transactive energy management systems. *IEEE Internet of Things Journal*, 5(2) :1055–1069, 2018. 18, 19, 24, 27, 28

[50] Ruilong Deng, Rongxing Lu, Chengzhe Lai, and Tom H Luan. Towards power consumption-delay tradeoff by workload allocation in cloud-fog computing. pages 3909–3914, 2015. 18, 20, 24

[51] Vasileios Karagiannis, Stefan Schulte, Joao Leitao, et al. Enabling fog computing using self-organizing compute nodes. pages 1–10, 2019. 19, 24

[52] Ye Yu, Xiangyuan Bu, Kai Yang, Zhikun Wu, and Zhu Han. Green large-scale fog computing resource allocation using joint benders decomposition, dinkelbach algorithm, admm, and branch-and-bound. *IEEE Internet of Things Journal*, 2018. 20, 21, 24

[53] Hamed Shah-Mansouri and Vincent WS Wong. Hierarchical fog-cloud computing for iot systems : A computation offloading game. *IEEE Internet of Things Journal*, 5(4) :3246–3257, 2018. 20, 21, 24, 27, 29

[54] Slađana Jošilo and György Dán. Decentralized algorithm for randomized task allocation in fog computing systems. *IEEE/ACM Transactions on Networking*, 27(1) :85–97, 2018. 20, 21, 24

[55] Xinchen Lyu, Chenshan Ren, Wei Ni, Hui Tian, and Ren Ping Liu. Distributed optimization of collaborative regions in large-scale inhomogeneous fog computing. *IEEE Journal on Selected Areas in Communications*, 36(3) :574–586, 2018. 20, 21, 24

[56] Junbin Liang, Yuxuan Long, Yaxin Mei, Tian Wang, and Qun Jin. A distributed intelligent hungarian algorithm for workload balance in sensor-cloud systems based on urban fog computing. *IEEE Access*, 7 :77649–77658, 2019. 20, 22, 24

[57] Jianbo Du, Liqiang Zhao, Jie Feng, and Xiaoli Chu. Computation offloading and resource allocation in mixed fog/cloud computing systems with min-max fairness guarantee. *IEEE Transactions on Communications*, 66(4) :1594–1608, 2018. 20, 22, 24

[58] Xiaoyi Tao, Kaoru Ota, Mianxiong Dong, Heng Qi, and Keqiu Li. Performance guaranteed computation offloading for mobile-edge cloud computing. *IEEE Wireless Communications Letters*, 2017. 23, 50, 52, 53

[59] Jianbo Du, Liqiang Zhao, Jie Feng, and Xiaoli Chu. Computation offloading and resource allocation in mixed fog/cloud computing systems with min-max fairness guarantee. *IEEE Transactions on Communications*, 66(4) :1594–1608, 2018. 23, 24, 50, 52, 53

[60] Ravi Tandon and Osvaldo Simeone. Harnessing cloud and edge synergies : Toward an information theory of fog radio access networks. *IEEE Communications Magazine*, 54(8) :44–50, 2016. 23, 27, 40

[61] Seok-Hwan Park, Osvaldo Simeone, and Shlomo Shamai. Joint cloud and edge processing for latency minimization in fog radio access networks. pages 1–5, 2016. 23, 25, 27

[62] Shao-Chou Hung, Hsiang Hsu, Shao-Yu Lien, and Kwang-Cheng Chen. Architecture harmonization between cloud radio access networks and fog networks. *IEEE Access*, 3 :3019–3034, 2015. 23, 25, 27

[63] Hongyu Xiang, Mugen Peng, Yuanyuan Cheng, and Hsiao-Hwa Chen. Joint mode selection and resource allocation for downlink fog radio access networks supported d2d. pages 177–182, 2015. 23, 25, 27

[64] Cuong T Do, Nguyen H Tran, Chuan Pham, Md Golam Rabiul Alam, Jae Hyeok Son, and Choong Seon Hong. A proximal algorithm for joint resource allocation and minimizing carbon footprint in geo-distributed fog computing. pages 324–329, 2015. 23, 26, 27

[65] Jingtao Su, Fuhong Lin, Xianwei Zhou, and Xing Lu. Steiner tree based optimal resource caching scheme in fog computing. *China Communications*, 12(8) :161–168, 2015. 23, 26, 27

[66] Matthew Malensek, Sangmi Lee Pallickara, and Shrideep Pallickara. Hermes : Federating fog and cloud domains to support query evaluations in continuous sensing environments. *IEEE Cloud Computing*, 4(2) :54–62, 2017. 23, 27

[67] Subhadeep Sarkar and Sudip Misra. Theoretical modelling of fog computing : a green computing paradigm to support iot applications. *Iet Networks*, 5(2) :23–29, 2016. 27, 28, 29

[68] Subhadeep Sarkar, Subarna Chatterjee, and Sudip Misra. Assessment of the suitability of fog computing in the context of internet of things. *IEEE Transactions on Cloud Computing*, 6(1) :46–59, 2015. 27, 28, 29

[69] Fatemeh Jalali, Kerry Hinton, Robert Ayre, Tansu Alpcan, and Rodney S Tucker. Fog computing may help to save energy in cloud computing. *IEEE Journal on Selected Areas in Communications*, 34(5) :1728–1739, 2016. 27, 28, 29

[70] Yu Cao, Songqing Chen, Peng Hou, and Donald Brown. Fast : A fog computing assisted distributed analytics system to monitor fall for stroke mitigation. pages 2–11, 2015. 27, 28, 29

[71] Di Chen, Stephan Schedler, and Volker Kuehn. Backhaul traffic balancing and dynamic content-centric clustering for the downlink of fog radio access network. pages 1–5, 2016. 27, 29

[72] Song Ningning, Gong Chao, An Xingshuo, and Zhan Qiang. Fog computing dynamic load balancing mechanism based on graph repartitioning. *China Communications*, 13(3) :156–164, 2016. 29

[73] Gilsoo Lee, Walid Saad, and Mehdi Bennis. An online secretary framework for fog network formation with minimal latency. pages 1–6, 2017. 29

[74] Anton Beloglazov, Jemal Abawajy, and Rajkumar Buyya. Energy-aware resource allocation heuristics for efficient management of data centers for cloud computing. *Future generation computer systems*, 28(5) :755–768, 2012. 31

[75] Subhadeep Sarkar, Subarna Chatterjee, and Sudip Misra. Assessment of the suitability of fog computing in the context of internet of things. *IEEE Transactions on Cloud Computing*, 6(1) :46–59, 2015. 34

[76] Antti P Miettinen and Jukka K Nurminen. Energy efficiency of mobile clients in cloud computing. *HotCloud*, 10 :4–4, 2010. 35, 76

[77] Ning Lu, Nan Cheng, Ning Zhang, Xuemin Shen, and Jon W Mark. Connected vehicles : Solutions and challenges. *IEEE internet of things journal*, 1(4) :289–299, 2014. 41

[78] Yuri Boykov and Vladimir Kolmogorov. An experimental comparison of min-cut/max-flow algorithms for energy minimization in vision. *IEEE Transactions on Pattern Analysis & Machine Intelligence*, (9) :1124–1137, 2004. 41

[79] KRA Jakob and Peter Mark Pruzan. The simple plant location problem : survey and synthesis. *European journal of operational research*, 12 :36–81, 1983. 41

[80] KRA Jakob and Peter Mark Pruzan. The simple plant location problem : Survey and synthesis. *European journal of operational research*, 12 :36–81, 1983. 42

[81] Yair Censor, Aviv Gibali, and Simeon Reich. The subgradient extragradient method for solving variational inequalities in hilbert space. *Journal of Optimization Theory and Applications*, 148(2) :318–335, 2011. 46

[82] Jong-Shi Pang and Masao Fukushima. Quasi-variational inequalities, generalized nash equilibria, and multi-leader-follower games. *Computational Management Science*, 2(1) :21–56, 2005. 46

[83] Francisco Facchinei, Andreas Fischer, and Veronica Piccialli. On generalized nash games and variational inequalities. *Operations Research Letters*, 35(2) :159–164, 2007. 47

[84] Francisco Facchinei and Jong-Shi Pang. *Finite-dimensional variational inequalities and complementarity problems*. Springer Science & Business Media, 2007. 48

[85] Tian Guo, Prashant Shenoy, KK Ramakrishnan, and Vijay Gopalakrishnan. Latency-aware virtual desktops optimization in distributed clouds. *Multimedia Systems*, 24(1) :73–94, 2018. 48

[86] Liqing Liu, Zheng Chang, and Xijuan Guo. Socially-aware dynamic computation offloading scheme for fog computing system with energy harvesting devices. *IEEE Internet of Things Journal*, 2018. 50, 52

[87] Kok-Hua Loh, Bruce Golden, and Edward Wasil. Solving the maximum cardinality bin packing problem with a weight annealing-based algorithm. In *Operations Research and Cyber-Infrastructure*, pages 147–164. Springer, 2009. 59

[88] David E Goldberg and John H Holland. Genetic algorithms and machine learning. *Machine learning*, 3(2) :95–99, 1988. 61

[89] AK Erlang. *Løsning af nogle Problemer fra Sandsynlighedsregningen af Betydning for de automatiske Telefoncentraler*. 1917. 62

[90] AA Jagers and Erik A van Doorn. Convexity of functions which are generalizations of the erlang loss function and the erlang delay function. *SIAM review*, 33(2) :281, 1991. 63, 75

[91] BY Gnedenko and I Kovalenko. *Introduction to Queuing Theory. Mathematical Modeling*. Birkhaeuser Boston, Boston, 1989. 63, 70

[92] Nam Kyoo Boots and Henk Tijms. Anm/m/c queue with impatient customers. *Top*, 7(2) :213–220, 1999. 64

[93] Xu Chen. Decentralized computation offloading game for mobile cloud computing. *IEEE Transactions on Parallel and Distributed Systems*, 26(4) :974–983, 2015. 72, 83

[94] David E Goldberg and John H Holland. Genetic algorithms and machine learning. *Machine learning*, 3(2) :95–99, 1988. 76, 78

[95] Jing Yuan, Yu Zheng, Xing Xie, and Guangzhong Sun. Driving with knowledge from the physical world. pages 316–324, 2011. 80

[96] Xiaohui Gu, Klara Nahrstedt, Alan Messer, Ira Greenberg, and Dejan Milojicic. Adaptive offloading for pervasive computing. *IEEE Pervasive Computing*, 3(3) :66–73, 2004. 80

[97] Subhadeep Sarkar and Sudip Misra. Theoretical modelling of fog computing : A green computing paradigm to support iot applications. *Iet Networks*, 5(2) :23–29, 2016. 80

[98] Ruilong Deng, Rongxing Lu, Chengzhe Lai, Tom H Luan, and Hao Liang. Optimal workload allocation in fog-cloud computing toward balanced delay and power consumption. *IEEE Internet of Things Journal*, 3(6) :1171–1181, 2016. 80, 83

[99] Loic Letondeur, François-Gaël Ottogalli, and Thierry Coupaye. A demo of application lifecycle management for iot collaborative neighborhood in the fog. 2017. 92

[100] Ye Xia, Xavier Etchevers, Loïc Letondeur, Thierry Coupaye, and Frédéric Desprez. Combining hardware nodes and software components ordering-based heuristics for optimizing the placement of distributed iot applications in the fog. pages 751–760, 2018. 92

[101] Murat Karakus and Arjan Durresi. A survey : Control plane scalability issues and approaches in software-defined networking (sdn). *Computer Networks*, 112 :279–293, 2017.

[102] Md Hasanul Ferdaus, Manzur Murshed, Rodrigo N Calheiros, and Rajkumar Buyya. Network-aware virtual machine placement and migration in cloud data centers. In *Emerging research in cloud distributed computing systems*, pages 42–91. IGI Global, 2015.

[103] Songlin Sun, Michel Kadoch, Liang Gong, and Bo Rong. Integrating network function virtualization with sdr and sdn for 4g/5g networks. *IEEE Network*, 29(3) :54–59, 2015.

[104] Vitor Barbosa C Souza, Wilson Ramírez, Xavier Masip-Bruin, Eva Marín-Tordera, G Ren, and Ghazal Tashakor. Handling service allocation in combined fog-cloud scenarios. pages 1–5, 2016.

[105] Mohammad Aazam and Eui-Nam Huh. Fog computing : The cloud-iot\/ioe middleware paradigm. *IEEE Potentials*, 35(3) :40–44, 2016.

[106] Xavi Masip-Bruin, Eva Marín-Tordera, Ghazal Tashakor, Admela Jukan, and Guang-Jie Ren. Foggy clouds and cloudy fogs : a real need for coordinated management of fog-to-cloud computing systems. *IEEE Wireless Communications*, 23(5) :120–128, 2016.

[107] Salvador Melendez and Michael P McGarry. Computation offloading decisions for reducing completion time. pages 160–164, 2017.

[108] Wanghong Yuan and Klara Nahrstedt. Energy-efficient soft real-time cpu scheduling for mobile multimedia systems. 37(5) :149–163, 2003.

[109] Mike Jia, Jiannong Cao, and Lei Yang. Heuristic offloading of concurrent tasks for computation-intensive applications in mobile cloud computing. pages 352–357, 2014.

[110] S Eman Mahmoodi, RN Uma, and KP Subbalakshmi. Optimal joint scheduling and cloud offloading for mobile applications. *IEEE Transactions on Cloud Computing*, 2016.

[111] Xavi Masip-Bruin, Eva Marín-Tordera, Ghazal Tashakor, Admela Jukan, and Guang-Jie Ren. Foggy clouds and cloudy fogs : a real need for coordinated management of fog-to-cloud computing systems. *IEEE Wireless Communications*, 23(5) :120–128, 2016.

[112] Ruilong Deng, Rongxing Lu, Chengzhe Lai, and Tom H Luan. Towards power consumption-delay tradeoff by workload allocation in cloud-fog computing. pages 3909–3914, 2015.

[113] Krittin Intharawijitr, Katsuyoshi Iida, and Hiroyuki Koga. Analysis of fog model considering computing and communication latency in 5g cellular networks. pages 1–4, 2016.

[114] Gilsoo Lee, Walid Saad, and Mehdi Bennis. An online secretary framework for fog network formation with minimal latency. pages 1–6, 2017.

[115] Dimitri P Bertsekas, Robert G Gallager, and Pierre Humblet. *Data networks*, volume 2. Prentice-Hall International New Jersey, 1992.

[116] Ejder Bastug, Mehdi Bennis, and Mérouane Debbah. Living on the edge : The role of proactive caching in 5g wireless networks. *IEEE Communications Magazine*, 52(8) :82–89, 2014.

[117] Seung-Woo Ko, Kaibin Huang, Seong-Lyun Kim, and Hyukjin Chae. Live prefetching for mobile computation offloading. *IEEE Transactions on Wireless Communications*, 16(5) :3057–3071, 2017.

[118] Mohammed S Elbamby, Mehdi Bennis, and Walid Saad. Proactive edge computing in latency-constrained fog networks. pages 1–6, 2017.

[119] Vytautas Valancius, Nikolaos Laoutaris, Laurent Massoulié, Christophe Diot, and Pablo Rodriguez. Greening the internet with nano data centers. pages 37–48, 2009.

[120] Fatemeh Jalali, Kerry Hinton, Robert Ayre, Tansu Alpcan, and Rodney S Tucker. Fog computing may help to save energy in cloud computing. *IEEE Journal on Selected Areas in Communications*, 34(5) :1728–1739, 2016.

[121] Thomas D Burd and Robert W Brodersen. Processor design for portable systems. *Journal of VLSI signal processing systems for signal, image and video technology*, 13(2-3) :203–221, 1996.

[122] Wanghong Yuan and Klara Nahrstedt. Energy-efficient cpu scheduling for multimedia applications. *ACM Transactions on Computer Systems (TOCS)*, 24(3) :292–331, 2006.

[123] Karel De Vogeleer, Gerard Memmi, Pierre Jouvelot, and Fabien Coelho. The energy/frequency convexity rule : Modeling and experimental validation on mobile devices. pages 793–803, 2013.

[124] Weiwen Zhang, Yonggang Wen, Kyle Guan, Dan Kilper, Haiyun Luo, and Dapeng Oliver Wu. Energy-optimal mobile cloud computing under stochastic wireless channel. *IEEE Transactions on Wireless Communications*, 12(9) :4569–4581, 2013.

[125] Xiaobo Fan, Wolf-Dietrich Weber, and Luiz Andre Barroso. Power provisioning for a warehouse-sized computer. 35(2) :13–23, 2007.

[126] Ching-Chi Lin, Pangfeng Liu, and Jan-Jan Wu. Energy-efficient virtual machine provision algorithms for cloud systems. pages 81–88, 2011.

[127] Mayank Mishra, Anwesha Das, Purushottam Kulkarni, and Anirudha Sahoo. Dynamic resource management using virtual machine migrations. *IEEE Communications Magazine*, 50(9) :34–40, 2012.

[128] Robert Kleinberg. A multiple-choice secretary algorithm with applications to online auctions. pages 630–631, 2005.

[129] Feyza Yildirim Okay and Suat Ozdemir. A fog computing based smart grid model. pages 1–6, 2016.

[130] Ricard Vilalta, Victor López, Alessio Giorgetti, Shuping Peng, Vittorio Orsini, Luis Velasco, Rene Serral-Gracia, Donal Morris, Silvia De Fina, Filippo Cugini, et al. Telcofog : A unified flexible fog and cloud computing architecture for 5g networks. *IEEE Communications Magazine*, 55(8) :36–43, 2017.

[131] Cyril Cecchinel, Matthieu Jimenez, Sébastien Mosser, and Michel Riveill. An architecture to support the collection of big data in the internet of things. pages 442–449, 2014.

[132] Weiwen Zhang, Yonggang Wen, Kyle Guan, Dan Kilper, Haiyun Luo, and Dapeng Oliver Wu. Energy-optimal mobile cloud computing under stochastic wireless channel. *IEEE Transactions on Wireless Communications*, 12(9) :4569–4581, 2013.

[133] Francisco Facchinei and Jong-Shi Pang. *Finite-dimensional variational inequalities and complementarity problems*. Springer Science & Business Media, 2007. IV

[134] Roger A Horn and Charles R Johnson. Matrix analysis cambridge university press. *New York*, 37, 1985. V

[135] Olga N Bondareva. Some applications of linear programming methods to the theory of cooperative games. *Problemy kibernetiki*, 10 :119–139, 1963. V, VI

[136] Lloyd S Shapley. On balanced sets and cores. *Naval research logistics quarterly*, 14(4) :453–460, 1967. VI

[137] Jeanette P Schmidt, Alan Siegel, and Aravind Srinivasan. Chernoff–hoeffding bounds for applications with limited independence. *SIAM Journal on Discrete Mathematics*, 8(2) :223–250, 1995.

Annexe A

Annexes

A.1 Preuve du théorème 2

Afin de prouver que notre GNEP a au moins une solution, nous devons d'abord prouver qu'il est conjointement convexe.

A.1.1 Preuve de la convexité conjointe de notre GNEP

Par souci de simplicité, nous définissons :

$$\alpha_j = \sum_{\hat{u} \in \mathcal{N}} \lambda_{\hat{u}} x_{\hat{u},j} \, {}^2\overline{\mathrm{D}}^{\mathrm{P}r}_{\hat{u},j}$$

$$\beta_{u,j} = (1 - \rho^{\mathrm{P}}_j)$$

$$\overline{\mathrm{D}}_{u,j} = \overline{\mathrm{D}}^{\mathrm{T}}_u + \overline{\mathrm{D}}^{\mathrm{P}r}_u$$

et,

$$\overline{\mathrm{E}}_{u,j} = \overline{\mathrm{E}}^{\mathrm{T}}_u + \overline{\mathrm{E}}^{\mathrm{P}r}_u$$

En utilisant ces notations, nous exprimons le coût dans (4.16) comme suit :

$$\overline{\mathrm{D}}_u = x_{u,0}\overline{\mathrm{D}}_{u,0} + \sum_{j \in \mathcal{M}} x_{u,j}\Big(\overline{\mathrm{D}}^{\mathrm{T}}_{u,j} + \frac{\alpha_j}{2\beta_{u,j}} + \overline{\mathrm{D}}^{\mathrm{P}r}_{u,j}\Big) \tag{A.1}$$

$$\overline{\mathrm{E}}_u = x_{u,0}\overline{\mathrm{E}}_{u,0} + \sum_{j \in \mathcal{M}} x_{u,j}\Big(\overline{\mathrm{E}}^{\mathrm{T}}_{u,j} + \mathrm{P}^{idle}_j \times \frac{\frac{\alpha_j}{2\beta_{u,j}}}{\frac{\alpha_j}{2\beta_{u,j}} + \overline{\mathrm{D}}^{\mathrm{P}r}_{u,j}} + \mathrm{P}^{max}_j \times \frac{\overline{\mathrm{D}}^{\mathrm{P}r}_{u,j}}{\frac{\alpha_j}{2\beta_{u,j}} + \overline{\mathrm{D}}^{\mathrm{P}r}_{u,j}}\Big) \tag{A.2}$$

$$\begin{aligned} \mathrm{C}_u(x_u, x_{-u}) = \frac{\mathrm{E}^{max}_u}{\mathrm{D}^{\mathrm{SLA}}_u}\Big[\Big(x_{u,0}\overline{\mathrm{D}}_{u,0} + \sum_{j \in \mathcal{M}} x_{u,j}\Big(\overline{\mathrm{D}}^{\mathrm{T}}_{u,j} + \frac{\alpha_j}{2\beta_{u,j}} + \overline{\mathrm{D}}^{\mathrm{P}r}_{u,j}\Big)\Big)\Big/ \\ \Big(x_{u,0}\overline{\mathrm{E}}_{u,0} + \sum_{j \in \mathcal{M}} x_{u,j}\Big(\overline{\mathrm{E}}^{\mathrm{T}}_{u,j} + \mathrm{P}^{idle}_j \times \frac{\frac{\alpha_j}{2\beta_{u,j}}}{\frac{\alpha_j}{2\beta_{u,j}} + \overline{\mathrm{D}}^{\mathrm{P}r}_{u,j}} + \mathrm{P}^{max}_j \times \frac{\overline{\mathrm{D}}^{\mathrm{P}r}_{u,j}}{\frac{\alpha_j}{2\beta_{u,j}} + \overline{\mathrm{D}}^{\mathrm{P}r}_{u,j}}\Big)\Big)\Big] \end{aligned} \tag{A.3}$$

Pour prouver la convexité de notre GNEP, nous exprimons les dérivées du premier ordre comme étant $h_{u,j} = \frac{\partial \lambda_u \mathrm{C}_u(x_u, x_{-u})}{\partial x_{u,j}}$, avec $j \in \mathcal{M} \cup \{0\}$. On a donc,

$$h_{u,0} = \lambda_u \frac{\mathrm{E}^{max}_u}{\mathrm{D}^{\mathrm{SLA}}_u}\big[\big(\overline{\mathrm{D}}_{u,0}\overline{\mathrm{E}}_u - \overline{\mathrm{E}}_{u,0}\overline{\mathrm{D}}_u\big)/(\mathrm{E}_u)^2\big] \tag{A.4}$$

Ainsi que,

$$h_{u,j}|_{j\neq 0} = \lambda_u \frac{\mathrm{E}_u^{max}}{\mathrm{D}_u^{\mathrm{SLA}}}\Big[\overline{\mathrm{E}}_u\Big(\overline{\mathrm{D}}_{u,j}^{\mathrm{T}} + \delta_{u,j} + \epsilon_{u,j}\Big) - \Big(\overline{\mathrm{E}}_{u,j}^{\mathrm{T}} + \mathrm{P}_j^{max}\frac{\overline{\mathrm{D}}_{u,j}^{Pr}}{\delta_{u,j}}(1-\frac{\epsilon_{u,j}}{\delta_{u,j}}) + \mathrm{P}_j^{idle}\frac{1}{\delta_{u,j}} \times (\frac{\alpha_j}{2\beta_{u,j}} + \frac{\overline{\mathrm{D}}_{u,j}^{Pr}}{\delta_{u,j}})\Big)\overline{\mathrm{D}}_u \Big/ (\mathrm{E}_u)^2\Big] \tag{A.5}$$

où $\delta_{u,j} = \frac{\alpha_j}{2\beta_{u,j}} + \overline{\mathrm{D}}_{u,j}^{Pr}$, $\epsilon_{u,j} = \lambda_u x_{u,j}(\frac{^2\overline{\mathrm{D}}_{u,j}^{Pr}}{2\beta_{u,j}} + \frac{\overline{\mathrm{D}}_{u,j}^{Pr}\alpha_j}{2\beta_{u,j}})$

Nous pouvons maintenant exprimer la matrice de hian sous la forme :

$$\mathrm{H}_u(x_u, x_{-u}) = \begin{pmatrix} h_{u,0}^u & 0 & \dots & 0 \\ 0 & h_{u,1}^u & \dots & 0 \\ \vdots & \dots & \ddots & \vdots \\ 0 & 0 & \dots & h_{u,\mathrm{M}}^u \end{pmatrix} \tag{A.6}$$

avec $h_{u,j}^u = \frac{\partial^2 \lambda_u \mathrm{C}_u(x_u, x_{-u})}{\partial x_{u,j}^2}$ et $j \in \mathscr{M} \cup \{0\}$.

Nous observons que tous les éléments diagonaux de $\mathrm{H}_u(x_u, x_{-u})$ ne sont pas négatifs, ainsi la matrice de hian $\mathrm{H}_u(x_u, x_{-u})$ est positif semi-défini sur $\mathscr{X}$, ce qui implique la convexité du GNEP.

A.1.2 Preuve théorème 2

Nous commençons par prouver que l'ensemble des solutions de VI(K, F) est compact et non vide. On peut constater que F est continue sur K, puisque ses dérivés du premier ordre $h_{u,j}$ sont clairement des fonctions rationnelles. Par conséquent, l'ensemble K est compact et convexe. Ainsi, comme le prouve le corolaire 2.2.5 dans [133], le VI(K, F) a une solution. Aussi, le GNEP définit dans (4.19) a une solution.

Nous démontrons ensuite que toute solution du problème VI(K, F) est une solution d'équilibre. La fonction F est continue sur K, on a donc $\lambda_u \mathrm{C}_u(x_u, x_{-u})$ est continuellement différentielle sur K. De plus, $\lambda_u \mathrm{C}_u(x_u, x_{-u})$ est convexe. Par conséquent, toute solution de VI(K, F) a une solution et, par conséquent, notre GNEP (4.19) a aussi une solution.

A.2 Résolution du GNEP (preuve du théorème 3)

Afin de prouver la monotonie de la fonction F présentée dans 3, nous montrons que JF le Jacobien de F est positif semi-défini sur K.

Le Jacobian JF a la structure suivante :

$$\mathrm{JF} = \begin{pmatrix} h_{1,0}^1 & 0 & \dots & 0 & 0 & 0 & \dots & 0 & \dots & 0 & 0 & \dots & 0 \\ 0 & h_{1,1}^1 & \dots & 0 & 0 & h_{2,1}^1 & \dots & 0 & \dots & 0 & h_{\mathrm{N},1}^1 & \dots & 0 \\ \vdots & \vdots & \ddots & \vdots & \vdots & \vdots & \ddots & \vdots & \ddots & \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \dots & h_{1,\mathrm{M}}^1 & 0 & 0 & \dots & h_{2,\mathrm{M}}^1 & \dots & 0 & 0 & \dots & h_{\mathrm{N},\mathrm{M}}^1 \\ \vdots & & & & & & & & & & & & \\ 0 & 0 & \dots & 0 & 0 & 0 & \dots & 0 & \dots & h_{\mathrm{N},0}^{\mathrm{N}} & 0 & \dots & 0 \\ 0 & h_{1,1}^{\mathrm{N}} & \dots & 0 & 0 & h_{2,1}^{\mathrm{N}} & \dots & 0 & \dots & 0 & h_{\mathrm{N},1}^{\mathrm{N}} & \dots & 0 \\ \vdots & \vdots & \ddots & \vdots & \vdots & \vdots & \ddots & \vdots & \ddots & \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \dots & h_{1,\mathrm{M}}^{\mathrm{N}} & 0 & 0 & \dots & h_{2,\mathrm{M}}^{\mathrm{N}} & \dots & 0 & 0 & \dots & h_{\mathrm{N},\mathrm{M}}^{\mathrm{N}} \end{pmatrix} \tag{A.7}$$

où $h_{u,j}^u$, $u \in \mathscr{N}$ et $j \in \mathscr{M}$ sont les mêmes variables décrites précédemment.

Nous avons alors :

$$\mathrm{JF} = \begin{pmatrix} \mathrm{A} & 0 & \dots & 0 \\ 0 & \mathrm{B}_1 & \dots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \dots & \mathrm{B}_\mathrm{M} \end{pmatrix}, \ \mathrm{A} = diag(h_{u,0}^u)_{u=1}^{\mathrm{N}} \tag{A.8}$$

Enfin,

$$\mathrm{B}_j = \begin{pmatrix} h^1_{1,j} & h^1_{2,j} & \dots & h^1_{\mathrm{N},j} \\ \vdots & \vdots & \ddots & \vdots \\ h^{\mathrm{N}}_{1,j} & h^{\mathrm{N}}_{2,j} & \dots & h^{\mathrm{N}}_{\mathrm{N},j} \end{pmatrix} \tag{A.9}$$

Sachant que A est positif défini par $\left(\prod_{u\in\mathcal{N}} \widetilde{\mathcal{K}_u}\right)$, pour prouver la monotonie de JF, nous devons montrer d'abord que B_i est positif semi-défini. En plus, nous devons prouver la semi-finitude de sa partie symétrique. On doit prouver que $\mathrm{B}^s_i = \frac{1}{2}(\mathrm{B}^{\mathrm{T}}_i + \mathrm{B}_i)$ est positif semi-d éfini.

Les éléments diagonaux de B_j sont donnés par ${}^d h^s_{j,u} = h^j_{j,u}$, et les éléments hors diagonale sont définis par ${}^o h^s_{u,j}|_{j\neq 0} = \frac{1}{2}(h^u_{j,u} + h^j_{u,u}$.

Nous définissons la matrice ψ_j comme suit :

$$\psi_j = \begin{pmatrix} diag(h^u_{u,j})_{u\in\mathcal{N}} & 0 \\ 0 & 0 \end{pmatrix} \tag{A.10}$$

Enfin, définissons la matrice ϕ_j par :

$$\phi_j = \begin{pmatrix} \delta_{1,j}x_{1,j} & \frac{\delta_{1,j}x_{1,j}+\delta_{2,j}x_{2,j}}{2} & \dots & \frac{\delta_{1,j}x_{1,j}+\delta_{\mathrm{N},j}x_{\mathrm{N},j}}{2} \\ \frac{\delta_{2,j}x_{1,j}+\delta_{1,j}x_{1,j}}{2} & \delta_{2,j}x_{2,j} & \dots & \frac{\delta_{1,j}x_{1,j}+\delta_{\mathrm{N},j}x_{\mathrm{N},j}}{2} \\ \vdots & \vdots & \ddots & \vdots \\ \frac{\delta_{\mathrm{N},j}x_{\mathrm{N},j}+\delta_{1,j}x_{1,j}}{2} & \frac{\delta_{\mathrm{N},j}x_{\mathrm{N},j}+\delta_{2,j}x_{2,j}}{2} & \dots & \delta_{\mathrm{N},j}x_{\mathrm{N},j} \end{pmatrix} \tag{A.11}$$

Après simplification, nous pouvons réécrire B_j comme suit :

$$\mathrm{B}_j = \left(\frac{\lambda_u}{\beta_j}\right)^2 \times \frac{\mathrm{E}^{max}_u}{\mathrm{D}^{\mathrm{SLA}}_u} \times \left(\psi_j\psi^{\mathrm{T}}_j \circ \left(\mathrm{I} + \mathrm{E} + \frac{2\lambda_u}{\beta_j}\phi_j\right)\right) \tag{A.12}$$

où $\circ$ désigne le produit Hadamard de deux matrices, c'est-à-dire que la matrice a pour éléments$(\mathrm{A}\circ \mathrm{B})_{ij} = \mathrm{A}_{ij}\mathrm{B}_{ij}$.

Nous savons que la matrice d'identité I et la matrice d'unité E sont définies positives. La matrice $\psi_j\psi^{\mathrm{T}}_j$ est clairement semi-définie positive. Nous observons également que ϕ_j est semi-défini positif, puisqu'il s'agit d'une matrice diagonale avec des éléments non négatifs. Le théorème du produit de Schur [134] indique que le produit de Hadamard de deux matrices semi-définies positives est également positif semi-défini.

Maintenant, nous prouvons que la matrice $\left(\mathrm{I} + \mathrm{E} + \frac{2\lambda_u}{\beta_j}\phi_j\right)$ est positive semi-définie. Cela nous amène à démontrer que la valeur propre minimale de la matrice $\frac{2\lambda_u}{\beta_j}\phi_j$ est supérieure ou égale à -1. D'après [135], la matrice $\phi_j = \frac{1}{2}(t_j e^{\mathrm{T}} + e(t_j)^{\mathrm{T}})$ a des caractéristiques polynomiales présentées comme suit :

$$\eta^{\mathrm{N}-2}\left(\eta^2 - 2(e^{\mathrm{T}}t_j)\eta + (e(t_j)^{\mathrm{T}})^2 - \mathrm{N}||t_j||^2\right) \tag{A.13}$$

Nous pouvons observer que $(t_j e^{\mathrm{T}} + e(t_j)^{\mathrm{T}})$ a une valeur propre non négative et une valeur propre non positive. Ces derniers sont donnés par : $\eta_+ = e(t_j)^{\mathrm{T}} + \sqrt{\mathrm{N}}||t_j||/2$ et $\eta_- = e(t_j)^{\mathrm{T}} - \sqrt{\mathrm{N}}||t_j||/2$. Ainsi, une condition suffisante pour que B^s_j soit semi-définie est :

$$\frac{\lambda_u}{\beta_j}\left(\sqrt{\mathrm{N}}||t_j|| - e(t_j)^{\mathrm{T}} + \right) \leq 1 \tag{A.14}$$

Puisque t_j est un vecteur avec des éléments non négatifs, nous avons $t_j e^{\mathrm{T}} \geq ||t_j||$ et$||t_j|| \leq \sqrt{\mathrm{N}}$. Et sachant que $\rho \mathrm{P}_j \leq \mathrm{U}_{max}$ et $\beta_j \geq (1 - \mathrm{U}_{max})$, on peut en déduire que :

$$\frac{\lambda_u}{\beta_j}\left(\sqrt{\mathrm{N}}||t_j|| - e(t_j)^{\mathrm{T}} + \right) \leq \frac{\lambda_u}{\beta_j}\left(\sqrt{\mathrm{N}}\max_{u\in\mathcal{N}} t_{u,j}(\sqrt{\mathrm{N}} - 1)\right) \leq \frac{\mathrm{N}\lambda_u}{1 - \mathrm{U}_{max}}\max_{u\in\mathcal{N}} \overline{\mathrm{D}}^{\mathrm{P}r}_{u,j} \tag{A.15}$$

Par conséquent, (A.15) est certainement satisfait si (3) est vrai.

A.3 Preuve du théorème 14

La preuve de ce théorème est basée sur la caractérisation de jeux équilibrés [135, 136]. Une map $\nu : 2_{-}^{\mathcal{F}\mathcal{I}} \to [0,1]$ est dite symétrique pour $\mathcal{F}\mathcal{I}$ si $\sum_{F \in 2_{-}^{\mathcal{F}\mathcal{I}}; k \in F} \nu = 1$ pour tout $k \in \mathcal{F}\mathcal{I}$. Pour chaque map équilibrée ν, nous avons défini $\mathbb{N}(\nu) = \{F \in 2_{-}^{\mathcal{F}\mathcal{I}} | \nu(F) > 0\}$. Toute collection $\mathbb{N} \supset 2_{-}^{\mathcal{F}\mathcal{I}}$ de coalitions est appelée équilibrée pour $\mathcal{F}\mathcal{I}$, s'il existe une map équilibrée ν pour $\mathcal{F}\mathcal{I}$ tel que $\mathbb{N} = \mathbb{N}(\nu)$.

Une collection équilibrée est appelée minimalement équilibrée pour $\mathcal{F}\mathcal{I}$, si $\mathcal{F}\mathcal{I} \notin \mathbb{N}$ et il n'existe pas d'autre collection équilibrée pour $\mathcal{F}\mathcal{I}$ qui soit un sous-ensemble approprié de $\mathbb{N}$. De la même manière, une map équilibrée ν i est appelée minimalement équilibrée (ou minimally balanced) pour $\mathcal{F}\mathcal{I}$. Chaque collection minimally balanced est associée à une map unique minimally balanced. Soit $\mathfrak{N}^{\mathcal{F}\mathcal{I}}$ le jeu de mappes minimally balanced,pour $\mathcal{F}\mathcal{I}$; chaque $\nu \in \mathfrak{N}^{\mathcal{F}\mathcal{I}}$ satisfait $\sum_{F \in 2_{-}^{\mathcal{F}\mathcal{I}}} \nu(F) \cdot \sum_{k \in F} f(k) = \sum_{k \in \mathcal{F}\mathcal{I}} f(k)$ pour toutes les fonctions $f : \mathcal{F}\mathcal{I} \to \mathbb{R}$.

Pour prouver la deuxième partie de ce théorème, nous allons exploiter le lemme suivant.

Lemme 17. *Soit $(\mathcal{F}\mathcal{I}, c)$ un jeu avec $|\mathcal{F}\mathcal{I}| > 1$ et $c(\mathcal{F}\mathcal{I}) < \sum_{F \in \mathbb{N}(\nu)} \nu(F) c(F)$ pour tout $\nu \in \mathfrak{N}^{\mathcal{F}\mathcal{I}}$. Ce jeu a donc une infinité d'allocations de base.*

Démonstration. Soit $\nu^* \in \arg\min_{\nu \in \mathfrak{N}^{\mathcal{F}\mathcal{I}}} \{\nu(F) c(F)\}$, tel que ν^* existe parce que le nombre de collections non équilibrées pour $\mathcal{F}\mathcal{I}$ n'est pas supérieur au nombre de sous-ensembles de $2_{-}^{\mathcal{F}\mathcal{I}}$. Soit $\iota^* = \sum_{F \in \mathbb{N}(\nu^*)} \nu^*(F) c(F) - c(F)$; notons que $\iota^* > 0$. Nous cherchons à identifier deux allocations de base différentes. Pour ce faire, nous définissons un jeu auxiliaire $(\mathcal{F}\mathcal{I}, c^*)$ par $c^*(F) = c(F)$ pour tout $F \in 2_{-}^{\mathcal{F}\mathcal{I}}$, $F \neq \mathcal{F}\mathcal{I}$ et $c^*(\mathcal{F}\mathcal{I}) = c(\mathcal{F}\mathcal{I}) + \iota^*$. Ensuite, pour tout $\nu \in \mathfrak{N}^{\mathcal{F}\mathcal{I}}$, on obtient :

$$\begin{aligned} c^*(\mathcal{F}\mathcal{I}) = c(\mathcal{F}\mathcal{I}) + \iota^* &= \sum_{F \in \mathbb{N}(\nu)} \nu^*(F) c(F) - \iota^* + \iota^* \\ &\leq \sum_{F \in \mathbb{N}(\nu)} \nu(F) c(F). \end{aligned}$$

où l'inégalité tient grâce au choix de ν^* comme minimiseur. Ainsi, le jeu $(\mathcal{F}\mathcal{I}, c^*)$ est équilibré, c'est-à-dire qu'il existe une allocation stable. Soit $\varsigma(N, c^*)$ une allocation stable pour (N, c^*).

De plus, soit $i, j \in \mathcal{F}\mathcal{I}$, $i \neq j$, deux joueurs différents. Nous définissons deux allocations pour le jeu$(\mathcal{F}\mathcal{I}, c) : n^{(i)}$ et $n^{(j)}$. La première allocation $n^{(i)}$ est définie par $n_k^{(i)} = \varsigma_k$,r pour tout $k \in \mathcal{F}\mathcal{I}/\{i\}$ et $n^{(i)} = \varsigma(\mathcal{F}\mathcal{I}, c) - \iota^*$. La deuxième allocation n^j est définie de manière analogue, ce qui réduit l'allocation du joueur j de ι^*. On peut voir clairement que $n^{(i)} \neq n^{(j)}$. Nous constatons aussi que $n^{(i)}$ est une allocation efficace pour le jeu $(\mathcal{F}\mathcal{I}, c)$, puisque :

$$\begin{aligned} \sum_{k \in \mathcal{F}\mathcal{I}} n_k^{(i)}) &= \sum_{k \in \mathcal{F}\mathcal{I}/\{i\}} \varsigma_k(\mathcal{F}\mathcal{I}, c^*) + \varsigma_i(\mathcal{F}\mathcal{I}, c^*) - \iota^* \\ &= c^*(\mathcal{F}\mathcal{I}) - \iota^* = c(\mathcal{F}\mathcal{I}). \end{aligned}$$

où la deuxième égalité tient parce que $\varsigma(\mathcal{F}\mathcal{I}, c^*)$ est une allocation pour le jeu $(\mathcal{F}\mathcal{I}, c^*)$. De plus, $n^{(i)}$ est une allocation stable pour $(\mathcal{F}\mathcal{I}, c)$, puisque pour tout $F \in 2_{-}^{\mathcal{F}\mathcal{I}}$, $F \neq \mathcal{F}\mathcal{I}$, on a :

$$\sum_{k \in F} n_k^{(i)}) \leq \sum_{k \in F} \varsigma_k(\mathcal{F}\mathcal{I}, c^*) \leq c^*(F) = c(F)$$

où la deuxième inégalité est valide parce que $\varsigma(\mathcal{F}\mathcal{I}, c^*)$ est une allocation stable pour le jeu $(\mathcal{F}\mathcal{I}, c^*)$. Nous concluons que $\varsigma^{(i)}$ est dans le noyau de $(\mathcal{F}\mathcal{I}, c^*)$. L'argument pour le noyau de l'inclusion de $\varsigma^{(j)}$ va de manière analogue. Enfin, comme le noyau est un ensemble convexe, l'existence de deux allocations de base différentes implique qu'il existe une infinité d'affectations de base. □

Supposons que $|\mathcal{F}\mathcal{I}| > 1$. L'inégalité $c^{\Psi}(\mathcal{F}\mathcal{I}) < \sum_{F \in \mathbb{N}(\nu)} \nu(F) c^{\Psi}(F)$ est stricte pour chaque $\nu \in \mathfrak{N}^{\mathcal{F}\mathcal{I}}$, comme indiqué précédemment. Ainsi, par le lemme 17, le jeu $(\mathcal{F}\mathcal{I}, c^{\Psi})$ a une infinité d'allocations de base. Ceci complète la preuve.

Adila MEBREK

Doctorat : Systèmes SocioTechniques

Année 2020

Fog Computing pour l'Internet des objets

Le *Fog Computing* constitue une approche prometteuse dans le contexte de l'Internet des Objets (IoT) car il fournit des fonctionnalités et des ressources à l'extrémité du réseau, plus près des utilisateurs finaux. Cette thèse étudie les performances du *Fog Computing* dans le cadre des applications IoT sensibles à la latence. La première problématique traitée concerne la modélisation mathématique d'un système IoT-*fog-cloud*, ainsi que les métriques de performances du système en termes d'énergie consommée et de latence. Cette modélisation nous permettra par la suite de proposer diverses stratégies efficaces de distribution de contenu et d'allocation des ressources dans le *fog* et le *cloud*. La deuxième problématique abordée dans cette thèse concerne la distribution de contenu et de données des objets dans des systèmes *fog/cloud*. Afin d'optimiser simultanément les décisions d'*offloading* et d'allocation des ressources du système, nous distinguons entre deux types d'applications IoT : (1) applications IoT à contenu statique ou avec des mises à jour peu fréquentes ; et (2) applications IoT à contenu dynamique. Pour chaque type d'application, nous étudions le problème d'*offloading* de requêtes IoT dans le *fog*. Nous nous concentrons sur les problèmes d'équilibrage de charge afin de minimiser la latence et l'énergie totale consommée par le système.

Mots clés : informatique dans les nuages – informatique mobile – internet des objets – traitement réparti – technologies de l'information et de la communication.

Fog Computing for the Internet of Things

Fog computing is a promising approach in the context of the Internet of Things (IoT) as it provides functionality and resources at the edge of the network, closer to end users. This thesis studies the performance of fog computing in the context of latency sensitive IoT applications. The first issue addressed is the mathematical modeling of an IoT-fog-cloud system, and the performance metrics of the system in terms of energy consumed and latency. This modeling will then allow us to propose various effective strategies for content distribution and resource allocation in the fog and the cloud. The second issue addressed in this thesis concerns the distribution of content and object data in fog / cloud systems. In order to simultaneously optimize offloading and system resource allocation decisions, we distinguish between two types of IoT applications: (1) IoT applications with static content or with infrequent updates; and (2) IoT applications with dynamic content. For each type of application, we study the problem of offloading IoT requests in the fog. We focus on load balancing issues to minimize latency and the total power consumed by the system.

Keywords: cloud computing – mobile computing – internet of things – distributed processing – information and communication.

Thèse réalisée en partenariat entre :

Aube en Champagne LE DÉPARTEMENT

utt UNIVERSITÉ DE TECHNOLOGIE TROYES

Ecole Doctorale "Sciences pour l'Ingénieur"