الجمهورية الجزائرية الديمقراطية الشعبية
République algérienne démocratique et populaire
وزارة التعليم العالي والبحث العلمي
Ministère de l'enseignement supérieur et de la recherche scientifique
جامعة عين تموشنت بلحاج بوشعيب
Université - Ain Temouchent - Belhadj Bouchaib
Faculté des Sciences et de Technologie
Département des Mathématiques et Informatique

Projet de Fin d'Etudes

Pour l'obtention du diplôme de Master en : **Informatique**
Domaine : **Mathématique Et Informatique**
Filière : **Informatique**
Spécialité : **Réseaux Et Ingénierie Des Données**

Thème

Modélisation et simulation des services IoT dans un environnement de Fog computing

Présenté par :

1) Melle. BENAMAR YOUSRA
2) Melle. LAREDJ BOUCHRA

Devant le jury composé de :

M. BENARIBI Fethi Imad	MAA	UAT.B.B (Ain Temouchent) Président
M. MERAD BOUDIA Djalal	MAA	UAT.B.B (Ain Temouchent) Examinateur
M. MEDEDJEL Mansour	MCA	UAT.B.B (Ain Temouchent) Encadrant

Année Universitaire 2022/2023

Remerciements

Avant tout, nous tenons à remercier Allah le tout puissant qui nous a donné la force et la patience pour accomplir ce travail.

Nos remerciements les plus sincères s'adressent à nos jurys «M. BENARIBI Fethi Imad» et «M. MERAD BOUDIA Djalal » pour le grand honneur, leurs efforts et soins apportés afin de présider notre travail.

Nous souhaitons également exprimer nos vifs et sincères remerciements à notre encadreur «M. MEDEDJEL Mansour » pour son aide très précieux, et de nous avoir guidés à chaque étape pour réaliser ce mémoire.

Dédicaces

Je dédié ce modeste travail :
A celle qui m'a donné la vie, le symbole de tendresse , qui s'est sacrifiée pour mon bonheur et ma réussite, qui ma soutenu dans les moments les plus defficiles,
« ma chère Maman » je lui dis merci...
A mon père, l'école de mon enfance, qui a été mon ombre durant toutes les années d'études, et qui a veillé tout au long de ma vie à m'encourager , à me donner l'aide et me protéger.
A mon frère et ma soeur qui ont été toujours là pour moi, mes tantes et à toute la famille.
Que Allah garde et protège ma famille.
A tous mes amis, mes collègues de classe, a tous les personnes qui m'ont souhaité la réussite.

Benamar Yousra

Dédicaces

Je dédie ce travail à la personne qui représente toujours ce qu'il y a de meilleur dans ma vie, ma chère mère qui est un symbole de sacrifice, d'amour, de patience, et morale, mon modèle dans la vie à laquelle je veux ressembler, la personne qui a toujours cru en moi et m'a soutenu durant toutes mes années d'études.

A mon cher père, le meilleur père du monde qui est un symbole de sacrifices et de courage, toujours présent et faisant tout son possible pour m'offrir une belle vie. Je suis fière de lui et de tout ce qu'il a accompli.

Je suis reconnaissante envers mes parents pour les ailes qu'ils m'ont donné, pour les enseignements spirituels qu'aucune école au monde ne peut offrir. Qu'ALLAH les garde à mes côtés A mon chère frère Bachir et ma chère sœur Abir, qui ont été mes compagnons de vie, mes confidents et mes plus grands soutiens. Leur appui et leur encouragement constants ont été pour moi une
source de motivation et joie.

A mes meilleurs amies surtout Afaf, Fatna et Hafsa qui m'ont toujours soutenu. Je suis reconnaissante pour leur amitié sincère et pour les souvenirs précieux que nous avons partagés
ensemble.

A mes grands-pères et ma grand-mère et aussi toute la famille Laredj et Remini. A mes professeurs qui m'ont guidé et soutenu tout au long de mon parcours académique à tous les niveaux.

A ma précieuse amie et binôme Yousra. Votre travail acharné, et votre détermination ont été pour moi une source de motivation. Je suis reconnaissante pour la chance d'avoir travaillé avec vous et pour les souvenirs précieux que nous avons partagés dans la réalisation de ce travail. Je suis fière de
tout ce que nous avons accompli ensemble.

Et à tous ceux qui m'ont soutenu et encouragé.

Laredj Bouchra

TABLE DES MATIÈRES

TABLE DES FIGURES

LISTE DES TABLEAUX

LISTE DES ALGORITHMES

Résumé

L'IoT est une technologie révolutionnaire offrant des avantages considérables en matière d'efficacité opérationnelle, de prise de décision et de satisfaction client. Cependant, le fonctionnement efficace de l'IoT nécessite une infrastructure performante, telle que celle du Cloud computing, pour gérer les données et les services, ainsi que les exigences applicatives. Le Cloud computing révèle cependant certaines limites liées notamment à la latence, à la bande passante et aux coûts d'exploitation. Le Fog computing est donc proposé comme étant une solution à ce type de problèmes en offrant une infrastructure de calcul distribuée pour traiter les données et les services à proximité des utilisateurs et des objets connectés, offrant ainsi une alternative efficace au Cloud computing. Dans ce mémoire, nous étudions un modèle d'un système IoT dans environnement de Fog computing en réalisant une simulation d'une stratégie de placement des services IoT sur un ensemble de nœuds Fog. Cette stratégie est mise en œuvre en proposant un algorithme de placement "Best Fog" qui utilise une approche basée sur plusieurs critères, tels que la capacité de mémoire et de calcul, pour sélectionner le nœud fog le plus adapté à une tâche spécifique. Les résultats de simulation obtenus sont jugés intéressants puisqu'ils révèlent une meilleure utilisation des ressources disponibles et une affectation plus efficace des tâches. Cette évaluation est également validée par une analyse comparative entre l'algorithme proposé ("Best Fog") et un autre algorithme de type FCFS.

Mots-Clés : Cloud computing, Fog computing, Internet des objets (IoT), FognetSim++.

INTRODUCTION GÉNÉRALE

Les avancées technologiques actuelles ont donné naissance à trois concepts qui transforment la manière dont les services et les applications sont déployés, gérés et utilisés à savoir l'IoT, le Cloud computing et le Fog computing.

L'IoT est une technologie révolutionnaire offrant des avantages considérables en matière d'efficacité opérationnelle, de prise de décision et de satisfaction client. Cependant, pour que l'IoT fonctionne efficacement, une infrastructure performante est nécessaire pour gérer les données et les services. Le Cloud computing est souvent utilisé pour fournir cette infrastructure, mais il peut révéler des problèmes de latence, de bande passante ou même de sécurité, en plus des coûts d'exploitation. Le Fog computing est donc proposé comme étant une solution à ce type de problèmes en offrant une infrastructure de calcul distribuée pour traiter les données et les services à proximité des utilisateurs et des objets connectés, offrant ainsi une alternative efficace au Cloud computing.

Dans ce contexte, l'objectif de ce projet de fin d'études est de proposer un modèle d'un système IoT et de le simuler dans un environnement de Fog computing en prenant en compte certains paramètres tels que la capacité de stockage et de calcul qui peuvent être d'une grande importance dans le processus de placement de données et de services sur les nœuds de fog.

Ce manuscrit est composé des trois chapitres suivants :

Dans le premier chapitre, nous décrivons le Cloud computing et le Fog computing en présentant leurs concepts fondamentaux tels que les modèles de services, l'architecture de fonctionnement et les caractéristiques inhérentes à ces deux paradigmes. A la fin de ce chapitre, l'accent est mis également sur les avantages du Fog computing, tels que la latence réduite, la bande passante optimisée et la résilience aux pannes.

le deuxième chapitre met en exergue l'IoT et son rôle dans la transformation numérique. Il examine en détail les concepts fondamentaux de l'IoT ainsi que les différentes couches de son architecture. De plus, ce chapitre soulève les défis et les enjeux auxquels l'IoT est confronté. Une analyse de ces aspects permettra de mieux comprendre le rôle essentiel de l'IoT dans l'évolution du paysage technologique actuel.

Dans le troisième chapitre, nous présentons la partie implémentation et simulation de ce projet. Un modèle de simulation des services IoT est également proposé en implémentant une stratégie de placement de tâches sur les différents nœuds constituant la plateforme de Fog computing.

Cette stratégie est mise en œuvre à travers un algorithme qui vise à optimiser l'utilisation des ressources disponibles d'une part, et à améliorer les performances du système d'autre part. A la fin de ce chapitre, nous examinons les résultats obtenus en les comparant avec ceux de l'application d'un autre algorithme de type FCFS (First-Come, First-Served).

A la fin de ce manuscrit, une conclusion est donnée dans laquelle nous mettant l'accent sur les résultats obtenus et les performances atteintes tout en identifiant les avantages et les limites de la stratégie proposée dans ce travail, ainsi que les possibilités offertes pour optimiser le fonctionnement des systèmes IoT dans un environnement de Fog computing.

CHAPITRE I

CLOUD ET FOG COMPUTING

1 Introduction

L'informatique est un domaine en constante évolution depuis ses débuts. Au cours de ces dernières années, une nouvelle idée a émergé : celle de l'informatique en nuage (Cloud computing). Cette notion a été introduite depuis les années 1960 par **John McCarthy** [1], notamment avec le développement des premiers ordinateurs centraux et devenue aujourd'hui essentielle pour les applications et les dispositifs intelligents en offrant un accès distant à une infrastructure de calcul et de stockage évolutive, notamment des serveurs.

Cependant, l'avènement de l'internet des objets (ou IoT : Internet of Things) a accéléré la génération de grandes quantités de données qui nécessitent un stockage et un traitement avec des temps de réponse réduits. Cette exigence peut être limitée par la vitesse d'Internet et le stockage dans le cloud, en particulier dans quelques domaines (Systèmes à temps réel par exemple). Pour surmonter ces obstacles, Cisco a introduit le concept de l'informatique en brouillard (Fog computing) qui permet aux périphériques intelligents de stocker et de traiter les données localement avant de les envoyer périodiquement au Cloud. Depuis lors, la popularité du Fog computing a augmenté et cette technologie a été adoptée dans différents secteurs tels que la santé, les transports, l'industrie et les villes intelligentes.

Face à cette tendance, nous allons présenter dans ce chapitre les concepts de Cloud et de Fog computing, à savoir les caractéristiques, les services rendus, les éléments constitutifs, les modèles de déploiement et les architectures, ainsi que les avantages et les limites.

2 Le Cloud computing

2.1 Définition

Le Cloud computing est une infrastructure qui repose sur des serveurs distants ayant une grande capacité de calcul et de mémoire, et auxquels les utilisateurs se connectent via Internet pour utiliser des services ou exécuter des tâches [17].

1. John McCarthy : (née le 4 septembre 1927, à Boston, Massachusetts) était un chercheur, professeur et pionnier de l'intelligence artificielle suggéra que la technologie informatique partagée pouvait construire un bel avenir dans lequel la puissance de calcul pouvait être vendue comme un service public.

Une autre définition selon l'Institut national des normes et de la technologie (**NIST** [2]) est : « le Cloud computing est un modèle qui permet un accès réseau omniprésent à un pool partagé de ressources informatiques configurables (par exemple, réseaux, serveurs, applications et services de stockage) qui peuvent être rapidement réservées et libérées avec un minimum d'effort de gestion ou d'interaction avec le fournisseur de service» [32].

2.2 Caractéristiques du Cloud computing

Les principales caractéristiques du Cloud computing peuvent être résumées dans ce qui suit :

- **Accès à la demande :** les utilisateurs peuvent accéder aux ressources informatiques instantanément, sans avoir besoin de passer par un processus de commande ou d'approbation [19].
- **Évolutivité rapide :** les ressources et les capacités peuvent être déployées et mises à l'échelle de manière rapide et automatisée en fonction des besoins, à n'importe quelle quantité et à tout moment [19].
- **Modèle de paiement à l'utilisation :** le Cloud computing fonctionne généralement selon un modèle de paiement à l'utilisation, où les utilisateurs ne paient que pour les ressources qu'ils ont réellement utilisées [19].
- **Gestion centralisée :** les ressources informatiques sont gérées de manière centralisée, permettant aux utilisateurs de se concentrer sur leur utilisation plutôt que sur leur gestion [7].
- **Sécurité :** les fournisseurs de services Cloud investissent massivement dans la sécurité et la protection des données, offrant ainsi une protection plus élevée que ce que la plupart des entreprises peuvent assurer en interne [7].

2.3 Concepts de base

Certains services et modèles fonctionnent en arrière-plan, ce qui rend le Cloud computing réalisable et accessible aux utilisateurs finaux [27]. Dans cette sous-section, nous présentons les principaux modèles sur lesquels repose le fonctionnement du Cloud computing :

2.3.1 Modèles de déploiement

Il existe différents modèles de déploiement pour accéder au Cloud, chacun proposant différents niveaux de contrôle et de responsabilité pour la gestion des ressources.

2. NIST : (National Institute for Standards and Technology) : Institut national pour les standards et la technologie est une division du département américain du commerce qui promouvoit l'économie en développant des technologies, la métrologie et des normes de concert avec l'industrie.

— **Cloud public :**le Cloud public est une infrastructure accessible au grand public ou grand groupe industriel et est généralement détenue par une organisation qui vend des services Cloud, ce qui en fait le modèle le plus courant , tel qu'AWS, Azure ou Google Cloud [43].

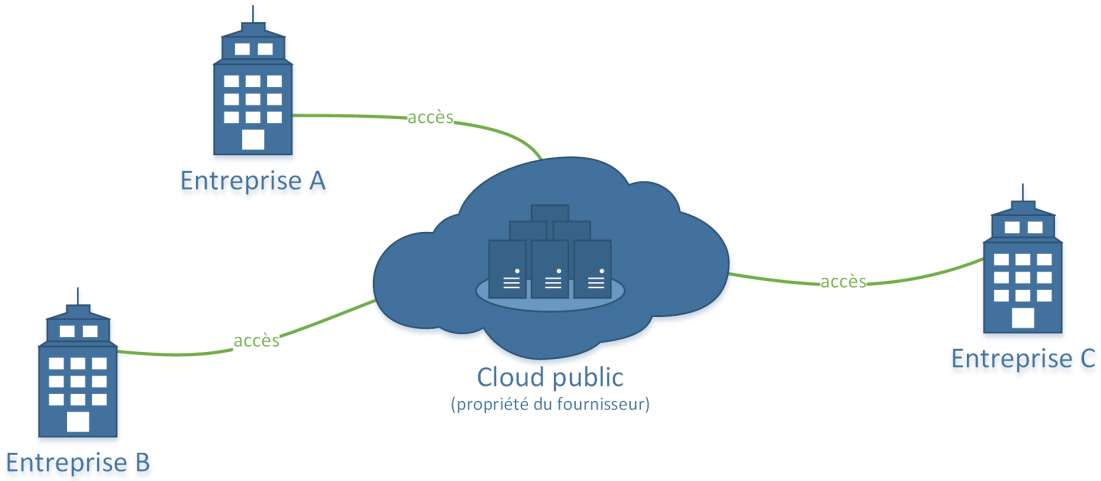

FIGURE 1 – Modèle de déploiement d'un Cloud public [5].

— **Cloud privé :** l'infrastructure Cloud est utilisée par une seule organisation ou une entreprise. Cette infrastructure peut être hébergée dans les locaux de l'organisation ou à l'extérieur [43]. Par exemple Window Server Hyper-V.

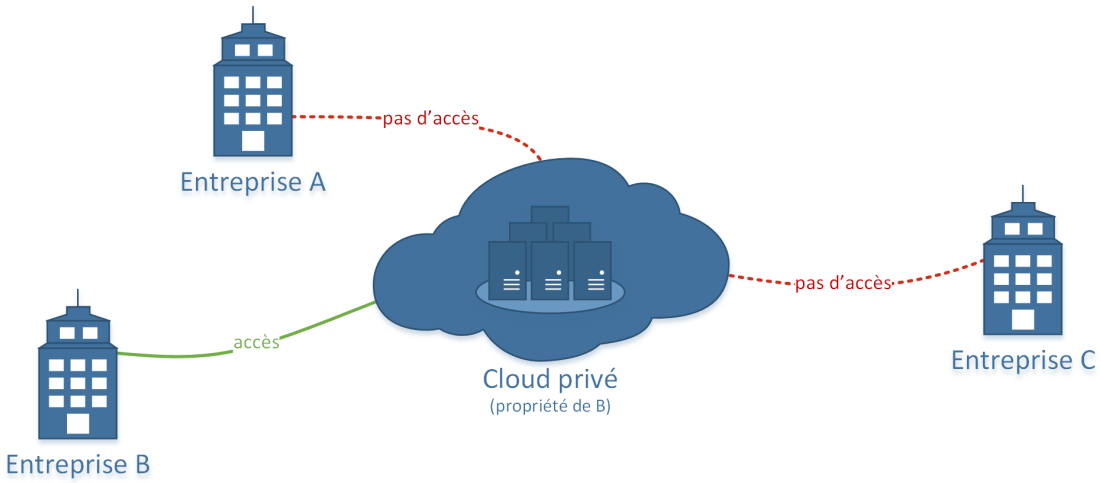

FIGURE 2 – Modèle de déploiement d'un Cloud privé [5].

— **Cloud hybride :** le modèle de déploiement de Cloud hybride combine des Clouds publics et privés, et peut même inclure des serveurs traditionnels sur site. Une entreprise peut utiliser son Cloud privé pour certains services et son Cloud public pour d'autres, ou peut choisir de maintenir son Cloud public comme solution de secours en cas de défaillance de son Cloud privé [10]. Par exemple Cloud Bursting pour gérer les pics de demande et répartir la charge entre Cloud public et Cloud privé

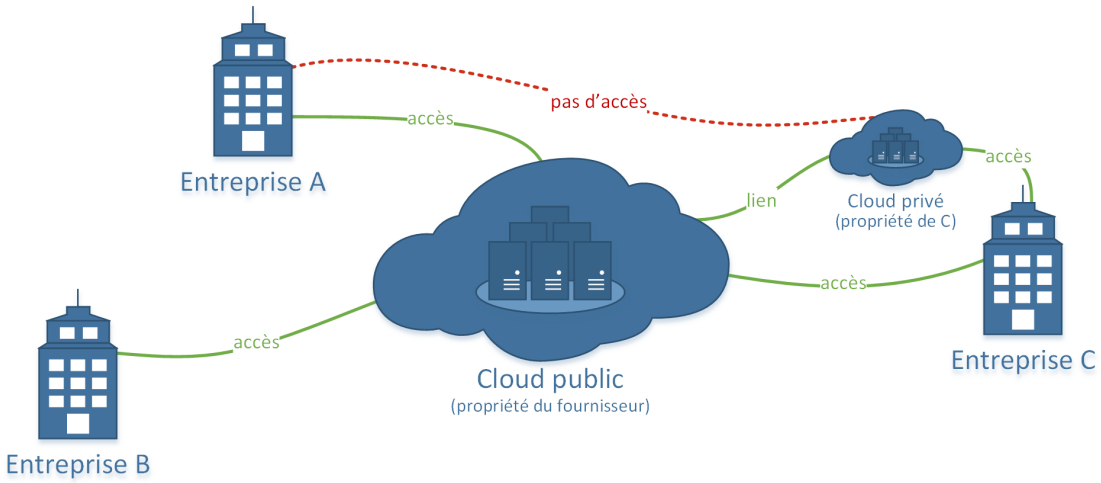

FIGURE 3 – Modèle de déploiement d'un Cloud hybride [5].

— **Cloud communautaire :** le modèle de Cloud communautaire implique le déploiement d'une infrastructure destinée à être utilisée exclusivement par un groupe d'entreprises ou d'organisations partageant des intérêts communs. Dans ce type d'architecture, l'administration du système peut être effectuée par une ou plusieurs des organisations qui partagent les ressources du Cloud [21].
La gestion et l'hébergement des ressources sont pris en charge par les membres de la communauté, ce qui permet d'optimiser leur utilisation tout en réduisant les coûts.

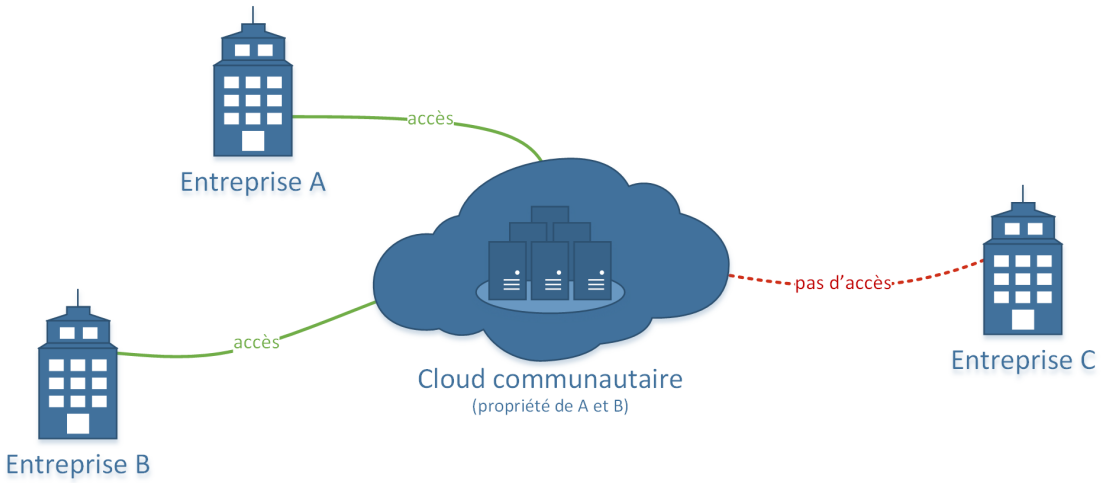

FIGURE 4 – Modèle de déploiement d'un Cloud communautaire [5].

2.3.2 Modèles de services

Les modèles de services Cloud font référence aux différents types de services proposés aux utilisateurs, tels que l'infrastructure en tant que service (IaaS), la plateforme en tant que service (PaaS) et le logiciel en tant que service (SaaS).

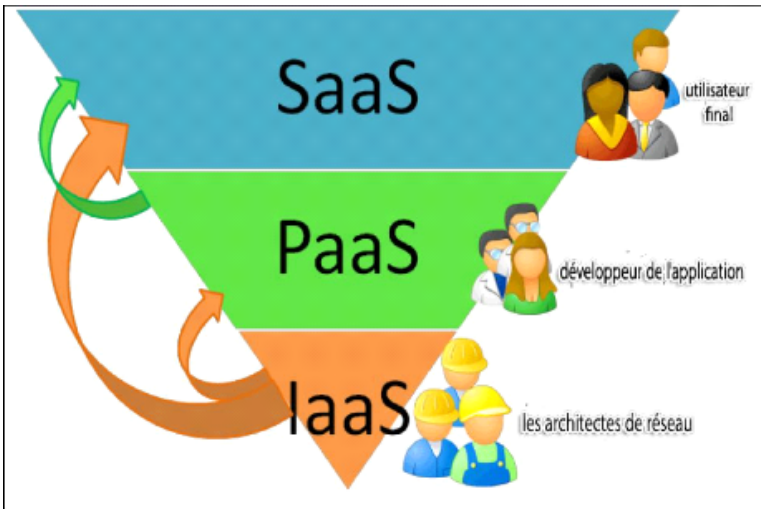

FIGURE 5 – Les modèles de services dans le Cloud [28].

La figure 5 présentée ci-dessus illustre les différentes couches du Cloud computing, classées selon leur niveau de visibilité pour les utilisateurs finaux.

- **Infrastructure en tant que service (IaaS) :** IaaS donne accès à des ressources fondamentales telles que des machines physiques, des machines virtuelles, du stockage virtuel, etc. [42]. Le consommateur peut déployer et exécuter des logiciels arbitraires.
  Par exemple : Amazon Web Services et Flexi scale.

- **Plateforme en tant que service (PaaS) :** le consommateur a la possibilité de déployer sur l'infrastructure Cloud des applications développées ou achetées, en utilisant les langages de programmation, bibliothèques, services et outils supportés par le fournisseur de services [40].
  Par exemple : Google App Engine.

- **Logiciel en tant que service (SaaS) :** le modèle SaaS permet à l'utilisateur d'accéder aux applications du fournisseur qui s'exécutent sur une infrastructure Cloud, via une interface client légère telle qu'un navigateur web ou une interface de programme, depuis différents dispositifs clients [40].
  Par exemple : Dropbox, Microsoft Office 365 et Salesforce.

2.4 Architecture du Cloud computing

L'architecture du Cloud computing est basée sur une combinaison d'architecture orientée services et d'architecture pilotée par les événements. Elle se divise en deux parties distinctes : le Front End et le Back-End comme est illustré dans la figure 6 ci-dessous :

- **Front-End :** le front-end est la partie du Cloud computing qui est accessible par le client et qui inclut les interfaces et les applications permettant d'accéder aux plates-formes de Cloud computing. Les éléments du front-end comprennent des services Web ainsi que des clients légers et lourds et des appareils mobiles, etc [9].

- **Back-End :** le back-end est utilisé par le fournisseur de services. Il gère toutes les ressources nécessaires pour fournir des services de Cloud computing. Il comprend une énorme quantité de stockage de données, un mécanisme de sécurité, des machines virtuelles, des modèles de déploiement, des serveurs, des mécanismes de contrôle du trafic, etc [9].

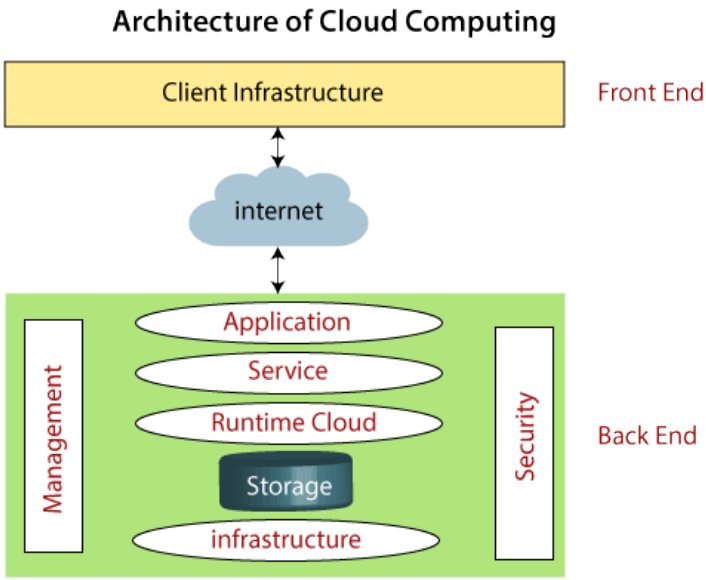

FIGURE 6 – Architecture du Cloud computing [9].

2.5 Composants de l'architecture du Cloud computing

Les différents composants de l'architecture du Cloud computing sont les suivants :

- **Infrastructure client :** ce composant permet aux utilisateurs d'interagir avec le Cloud à travers une interface graphique.

- **Applications :** tout logiciel ou plate-forme à laquelle un utilisateur souhaite accéder est considéré comme une application dans le Cloud.

- **Services :** les services Cloud gèrent les différents types de services auxquels un client peut accéder en fonction de ses besoins.

- **Environnement d'exécution (Runtime) :** cet élément fournit l'environnement d'exécution nécessaire aux machines virtuelles.

- **Stockage :** ce composant offre une grande capacité de mémoire dans le Cloud pour stocker et gérer les données.

- **Infrastructure :** elle fait référence aux composants matériels et logiciels tels que des serveurs, du stockage, des périphériques réseau et des logiciels de virtualisation.

- **Gestion :** cet élément permet de gérer les différents composants du Cloud computing tels que les applications, les services, l'environnement d'exécution, le stockage et l'infrastructure, ainsi que d'autres problèmes de sécurité dans le back-end.

- **Sécurité :** est un composant intégré du Cloud computing, qui implémente des mécanismes de sécurité dans le back-end.

- **Internet :** est une infrastructure qui permet la communication et l'interaction entre les différentes interfaces front-end et les systèmes back-end [9].

2.6 Avantages et inconvénients du Cloud computing

Le Cloud computing a en fait certains avantages et inconvénients que nous pouvons résumer dans le tableau suivant :

Tableau 1 : Les avantages et les limites du Cloud computing [42, 4]

Avantages	Inconvénients
— L'accès aux applications en tant que services via Internet — La possibilité de manipuler et de configurer les applications en ligne à tout moment — Ressources cloud disponibles pour un accès indépendant de la plateforme à tout type de client — Nécessité seulement d'une connexion Internet — Utilisation des ressources cloud en libre-service à la demande sans interaction avec le fournisseur de services — Équilibrage de charge pour une plus grande fiabilité	— Exigence de connexion Internet constante — Sécurité moindre — Temps de réponse lent — Coûts potentiels élevés — Limites de performances

3 Le Fog computing

3.1 Définition

Le Fog computing a été proposée en 2012 par Cisco comme étant une infrastructure décentralisée dans laquelle les données, les calculs et les applications sont distribués à la périphérie d'un réseau entre les sources de données (ou objets connectés) et le Cloud [35].

D'après le NIST, le Fog computing est défini comme un paradigme de ressources horizontales, physiques ou virtuelles qui résident entre les terminaux intelligents et les centres de données ou Cloud traditionnel. Ce paradigme prend en charge les applications verticalement isolées et sensibles à la latence en fournissant un service de calcul, de stockage et de connectivité omniprésent, évolutive fédérée et distribué [29].

L'objectif du Fog computing est de rapprocher les avantages du Cloud au bord du réseau, où la plupart des données sont générées et où des informations exploitables sont nécessaires en temps réel. Il prend en compte la grande quantité de données générées par l'IoT (Internet of Things) et la variation de la latence dans un réseau distribué, tout en offrant un meilleur contrôle sur les données transmises [6].

3.2 Caractéristiques du Fog computing

Les caractéristiques du Fog computing peuvent être résumées dans ce qui suit [6] :

- **Disponibilité :** le Fog computing peut assurer une disponibilité élevée grâce à son architecture décentralisée qui implique la distribution des nœuds de traitement sur le réseau. Cette répartition de la charge de traitement des données permet de réduire le risque d'avoir un seul point de défaillance, ce qui peut améliorer la disponibilité des services pour les utilisateurs finaux.

- **Localisation et faible latence :** le Fog computing prend en charge la connaissance de l'emplacement où les nœuds du Fog sont déployés et permet de réduire le temps de latence en rapprochant les traitements de données de leur source.

- **Évolutivité :** le Fog computing permet une évolution flexible des architectures IoT grâce à la capacité de traitement et de mémoire locale offerte par les nœuds Fog, ce qui permet une adaptation plus facile aux besoins variantes des applications.

- **Hétérogénéité :** l'hétérogénéité est également une caractéristique importante du Fog computing, car il permet la prise en charge de différents types de périphériques, de plateformes et de protocoles de communication. Cette caractéristique facilite l'intégration de différents systèmes et périphériques dans l'architecture du Fog .

- **Prise en charge de la mobilité :** le Fog computing prend en compte la mobilité des appareils connectés en permettant un traitement des données là où elles sont générées, ce qui permet une réduction de la latence et une utilisation plus efficace de la bande passante.

- **Scalabilité :** le Fog computing peut s'adapter aux besoins de charge en augmentant ou en diminuant la capacité de traitement et de mémoire de ses nœuds, ce qui permet une utilisation plus efficace des ressources et une meilleure gestion des coûts.

- **Décentralisation :** contrairement au Cloud computing, le Fog computing est une infrastructure informatique décentralisée, où le traitement est réparti sur le réseau. Cela permet d'améliorer la fiabilité du système et de réduire les risques liés à un point de défaillance unique.

- **Proximité des périphériques :** le Fog computing s'appuie sur des nœuds des systèmes informatiques à petite échelle, qui sont déployés à la périphérie du réseau, tels que des routeurs, des commutateurs et autres périphériques. La proximité de ces nœuds permet de réduire la latence et d'effectuer un traitement des données en temps réel.

- **Interopérabilité :** le Fog computing permet aux composants de différentes zones et de différents fournisseurs de services de communiquer et de fonctionner ensemble.

- **Sécurité :** le Fog computing permet de renforcer la sécurité des données en évitant de les envoyer sur des réseaux à distance d'une part, et d'autre part, il permet de mettre en place des mécanismes de sécurité locaux contrôlables.

3.3 Architecture du Fog computing

L'architecture du Fog est généralement composée de trois couches : la couche Fog, la couche IoT(ou périphérie du réseau) et la couche Cloud comme illustré dans la figure 7 ci-dessus

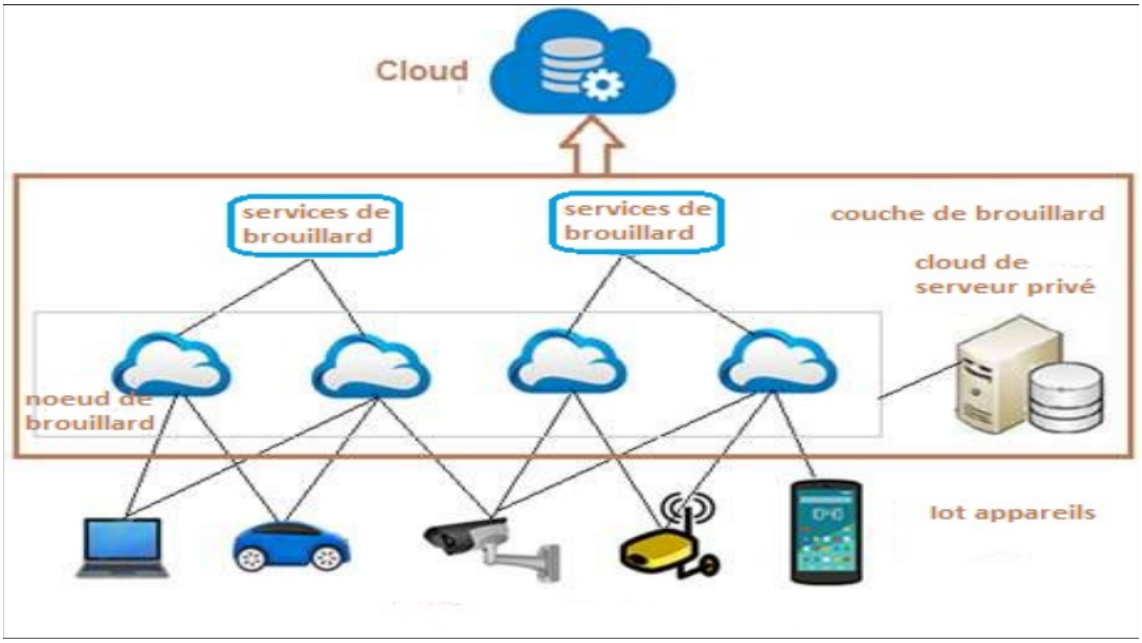

FIGURE 7 – Architecture du Fog computing[16].

— **La couche Cloud :** c'est la couche la plus haute, située dans le Cloud. Elle est composée de centres de données, de serveurs et de ressources de stockage qui fournissent des services de calcul et de stockage aux couches inférieures.

— **La couche Fog :** c'est la couche intermédiaire, située entre le Cloud et les terminaux des utilisateurs finaux. Elle est composée de nœuds de calcul et de stockage décentralisés, tels que des passerelles de périphériques, des routeurs, des commutateurs et des points d'accès Wi-Fi, qui fournissent des services de traitement de données et de communication de proximité.

— **La couche IoT :** c'est la couche la plus basse, située sur les terminaux des utilisateurs finaux, tels que les smartphones, les tablettes, les ordinateurs portables, et les capteurs.Cette couche est responsable de la collecte de données, du traitement local et de l'interaction avec les utilisateurs [6].

3.4 Avantages du Fog computing

Le Fog computing offre plusieurs avantages, notamment :

— **Faible latence :** l'un des principaux avantages du Fog computing est qu'il permet de réduire la latence en rapprochant le traitement et le stockage des données du périphérique ou de l'utilisateur final. Cela permet une réponse plus rapide et une meilleure expérience utilisateur.

- **Meilleure exploitation de la bande passante :** le Fog computing permet également de réduire la quantité de données qui doivent être envoyées au cloud pour le traitement, ce qui réduit la demande de bande passante et les coûts associés.

- **Sécurité améliorée :** en rapprochant les données du périphérique ou de l'utilisateur final, le Fog computing permet également de renforcer la sécurité des données, car celles-ci ne sont pas exposées sur des réseaux publics ou sur le Cloud.

- **Traitement temps réel :** en rapprochant les données des périphériques et des utilisateurs finaux, le Fog computing permet une analyse et une prise de décision plus rapide et plus précise en temps réel, ce qui offre un traitement de données en temps réel pour les applications qui nécessitent des réponses rapides.

- **Haute disponibilité :** grâce à une répartition de la charge de traitement et de stockage sur plusieurs nœuds, le Fog computing peut garantir une haute disponibilité des applications, assurant ainsi une continuité de service même en cas de panne de certains nœuds.

- **Efficacité énergétique :** le Fog computing permet une gestion efficace de la consommation d'énergie en répartissant de manière intelligente les tâches de traitement entre les périphériques et les serveurs, en optimisant les ressources de calcul et en évitant le transfert inutile de données vers des centres de données distants [15].

3.5 Défis du Fog computing

Le Fog computing est confronté à plusieurs défis qui nécessitent une attention particulière pour assurer son efficacité et son succès, notamment :

- **Complexité :** la diversité des dispositifs IoT provenant de différents fabricants complique le choix des composants optimaux, en raison des différentes configurations matérielles et logicielles ainsi que des exigences spécifiques.

- **Dynamicité :** les appareils IoT évoluent et changent leur flux de travail au fil du temps, ce qui nécessite une reconfiguration automatique et intelligente des ressources et de la structure topologique des nœuds Fog.

- **Hétérogénéité :** la gestion des réseaux IoT devient complexe en raison de la diversité des dispositifs provenant de différents fabricants, avec des capacités de communication, de calcul et de mémoire variables.

- **Latence :** le Fog computing vise à offrir une faible latence pour les applications sensibles au temps, mais plusieurs facteurs peuvent entraîner une latence élevée, ce qui peut entraîner une insatisfaction des utilisateurs.

- **Sécurité :** la sécurité de Fog computing est un défi en raison de la vulnérabilité inhérente à cet environnement, nécessitant des mesures de sécurité spécifiques adaptées à la mobilité, l'hétérogénéité et la distribution géographique du Fog.

- **Gestion des ressources :** les appareils fog ont des capacités de mémoire et de calcul limitées par rapport aux serveurs traditionnels, ce qui nécessite une gestion efficace des ressources pour un fonctionnement optimal de l'informatique Fog.

- **Consommation d'énergie :** le Fog computing implique de nombreux appareils de périphérie, et la répartition des calculs peut être moins économe en énergie que le modèle de calcul centralisé du Cloud. Par conséquent, réduire la consommation d'énergie dans le calcul Fog est un défi important à relever [3].

3.6 Applications du Fog computing

Le Fog computing peut être appliqué dans de nombreux domaines, parmi lesquels, nous pouvons citer [25] :

- **Villes intelligentes :** les villes intelligentes utilisent l'IoT pour collecter des données sur les conditions de circulation, la qualité de l'air, la consommation d'énergie, etc. Le Fog computing permet de traiter ces données en temps réel pour prendre des décisions plus rapidement et améliorer la qualité de vie des citoyens.

- **Santé connectée :** les dispositifs médicaux connectés, tels que les moniteurs de glucose et les tensiomètres, collectent des données sur la santé des patients. Le Fog computing permet de traiter ces données en temps réel et de les transmettre aux professionnels de santé pour améliorer les soins.

— **Industrie :** les usines intelligentes, notamment dans l'industrie 4.0, utilisent l'IoT pour collecter des données sur les machines et les processus de production. Le Fog computing permet de traiter ces données en temps réel pour détecter les pannes avant qu'elles ne se produisent et améliorer l'efficacité de la chaîne de production.

— **Agriculture intelligente :** en utilisant les capteurs IoT pour collecte des données sur la température, l'humidité et la qualité du sol dans les champs, le Fog computing permet de traiter ces données en temps réel pour optimiser les cultures, réduire les coûts et améliorer la productivité.

4 Conclusion

Le Cloud computing est une infrastructure offrant de nombreux avantages pour le stockage et le traitement de grandes quantités de données. Cependant, il peut s'avérer insuffisant et inefficace pour le traitement des données générées par l'IoT en raison des exigences en matières de certains critères tels que la latence, la disponibilité, la mobilité et d'autres. A cet effet, le Fog computing a été proposé comme une solution complémentaire du Cloud computing pour répondre aux besoins de traitement des données avec une faible latence, et d'autres fonctionnalités que les infrastructures de Cloud computing ne peuvent pas fournir ou assurer efficacement. Dans ce chapitre, nous avons présenté ces deux paradigmes en mettant l'accent sur le Fog computing comme étant une solution prometteuse à la technologie IoT.
Le chapitre suivant détaille le concept de la technologie IoT et son rapport avec le Fog computing.

CHAPITRE II

L'INTERNET DES OBJETS (IoT)

1 Introduction

L'Internet des objets (ou IoT : Internet of Things) est un concept technologique qui a émergé dans les années 1990 lorsque les premiers appareils connectés à Internet ont commencé à apparaître. Ces appareils étaient principalement utilisés pour surveiller les environnements industriels, tels que les pipelines et les machines. À l'époque, ces appareils étaient coûteux et ne pouvaient transmettre que de petites quantités de données. Mais, le terme IoT a été utilisé pour la première fois à la fin des années 90 par Kevin Ashton en proposant de mettre des puces RFID (Radio Frequency Identification) sur les produits pour les suivre à travers une chaîne d'approvisionnement [2].

L'IoT consiste à connecter des objets physiques à Internet et à leur permettre d'échanger des données sans intervention humaine. Cette technologie repose sur la combinaison de plusieurs éléments tels que des capteurs, des réseaux de communication, des plates-formes logicielles, des services cloud et des interfaces utilisateur.

Dans ce chapitre, nous allons explorer cette technologie, en présentant son architecture, y compris les modèles de communication, les protocoles et les technologies utilisées, ainsi que les avantages et les inconvénients de cette technologie. Ensuite, nous nous pencherons sur les domaines d'application de l'IoT et sur le rôle du Fog computing dans l'amélioration des capacités de cette technologie.

2 Définition

Le concept de l'IoT s'articule sur deux termes, "Internet" et "Objet". Le premier terme se réfère aux protocoles, aux services et aux réseaux, tandis que le second terme désigne les capteurs et les dispositifs intelligents. Il est caractérisé par des capacités de calcul, de mémoire et de traitement limitées.

L'IoT peut être défini comme un réseau de dispositifs intelligents interconnectés qui collectent et échangent des données en utilisant des protocoles de communication standards. Dans ce qui suit, quelques définitions courantes de l'IoT :

— Selon **Gartner**[3] l'IoT est "le réseau de dispositifs physiques, de véhicules, de maisons, de bâtiments et d'autres objets intégrés avec des capteurs, des logiciels et des réseaux qui permettent une collecte et un échange de données" [18].

3. Gartner : est une entreprise de conseil et de recherche spécialisée dans les technologies de l'information. Dans le domaine de l'IoT,

— Selon **Cisco** [4], l'IoT est "la connexion des objets physiques au réseau pour permettre une collecte de données, une analyse et une action en temps réel" [8].

3 Importance de l'IoT

Dans l'ère numérique actuelle, l'IoT est devenu un élément clé en raison de sa capacité à connecter les objets physiques à Internet et à faciliter l'échange de données sans intervention humaine. L'IoT permet également d'atteindre les objectifs suivants [23] :

— **Automatisation :** grâce à l'IoT, les processus peuvent être automatisés, ce qui permet d'augmenter l'efficacité et de réduire les coûts en éliminant les tâches manuelles.

— **Collecte de données en temps réel :** l'IoT permet de collecter des données en temps réel sur les objets connectés, ce qui permet aux entreprises de prendre des décisions plus précises et plus rapides.

— **Optimisation des ressources :** l'IoT permet une meilleure utilisation des ressources, telles que l'énergie, ce qui réduit les coûts et maximise les gains.

— **Innovation :** l'IoT offre de nouvelles opportunités d'innovation dans de nombreux domaines tels que la santé, l'agriculture, l'environnement et la sécurité.

4 Architecture de l'IoT

L'architecture d'IoT repose principalement sur les quatre couches suivantes : la couche Perception, la couche Réseau, la couche Middleware et la couche Application comme illustré dans la figure 8.

4. Cisco est une entreprise réputée spécialisée dans les solutions de réseau et de communications. Elle propose une large gamme de produits et de services novateurs pour connecter, sécuriser et accompagner la transformation digitale des entreprises, notamment en matière de gestion de l'Internet des objets (IoT).

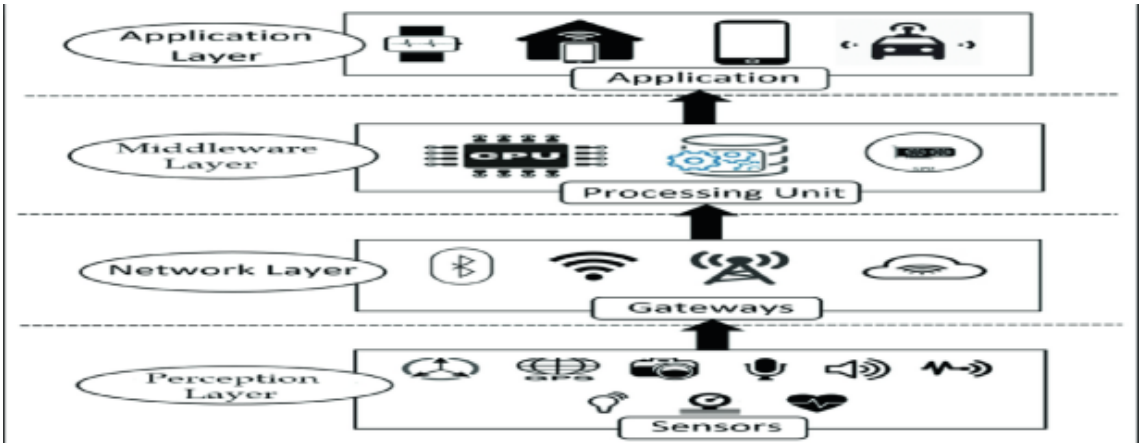

FIGURE 8 – Architecture de l'IoT [20].

— **Couche Perception :** cette couche est constituée des dispositifs physiques (capteurs, lecteurs, ...) qui permettent la détection et la collecte de données à partir de leur environnement. Ses principaux composants sont les :

  ◦ Capteurs : détectent les changements physiques, chimiques ou environnementaux.

  ◦ Actionneurs : permettent de contrôler les appareils connectés, des périphériques embarqués tels que des microcontrôleurs ou des SoC (System-on-Chip) qui traitent les données collectées

  ◦ Connecteurs : permettent de connecter les capteurs et les actionneurs au réseau IoT.

— **Couche Réseau :** cette couche est responsable de la transmission des données collectées de la couche Perception à la couche Middleware. Les composants de cette couche peuvent inclure [24] :

  ◦ Les protocoles et les technologies de communication : telles que Wifi, Sigfox, Zigbee, Bluetooth, LoRaWAN et les réseaux cellulaires (Voir tableau 1 pour plus de détails).

  ◦ Passerelles IoT : la passerelle IoT permet la communication entre les capteurs et le Cloud en agrégeant les données collectées par les capteurs et les envoyant à la plateforme Cloud [44].

Tableau 2 : les différentes protocoles et technologies de communications

Protocoles	Explication
Wifi	Wireless Fidelity, est une technologie de mise en réseau sans fil qui permet aux appareils de communiquer entre eux via des signaux sans fil. Il offre des vitesses de transmission de données élevées et permet une connectivité dans un rayon limité, généralement de quelques dizaines de mètres [26].
Sigfox	est une technologie à faible consommation d'énergie pour la communication sans fil d'une gamme variée d'objets à faible énergie tels que les capteurs et les applications M2M. Il permet le transport de petites quantités de données allant jusqu'à 50 kilomètres [39].
Zigbee	il s'agit d'un protocole de communication sans fil à courte portée, conçu pour les réseaux de capteurs sans fil à faible consommation d'énergie. Zigbee utilise une topologie en étoile ou en maillage pour connecter les objets connectés.
Bluetooth	il s'agit d'un protocole de communication sans fil à courte portée utilisé dans de nombreux dispositifs IoT, tels que les capteurs, les montres connectées et les objets de domotique.
LoRaWAN	(Long Range Wide Area Network) : il s'agit d'un protocole de communication sans fil à longue portée conçu pour les réseaux de capteurs à faible consommation d'énergie. LoRaWAN permet une communication bidirectionnelle entre les objets connectés et les passerelles, avec une portée allant jusqu'à plusieurs kilomètres.
Réseaux cellulaires	sont des infrastructures de communication sans fil adaptées aux exigences des appareils IoT, leur permettant de se connecter, de communiquer et d'échanger des données sur de vastes étendues géographiques, telles que : 4G, LTE [37].

- **Couche Middleware :** cette couche fournit une plateforme logicielle responsable du traitement, et de la communication des données, ainsi que l'intégration entre les dispositifs et les applications. . Elle englobe les protocoles de communication suivants :

  - MQTT (Message Queuing Telemetry Transport) : il s'agit d'un protocole de messagerie léger conçu pour les appareils à faible bande passante et à faible puissance de traitement. MQTT permet une communication bidirectionnelle entre les objets connectés et les serveurs, avec une faible latence et une consommation d'énergie réduite.

- HTTP (Hypertext Transfer Protocol) : il s'agit du protocole de communication standard pour le World Wide Web, mais il est également utilisé pour l'IoT. HTTP permet une communication entre les objets connectés et les serveurs à l'aide de requêtes et de réponses.

- CoAP (Constrained Application Protocol) : il s'agit d'un protocole de communication destiné aux objets connectés à faible puissance de traitement et à faible consommation d'énergie, tels que les capteurs. CoAP utilise des messages HTTP simplifiés pour permettre la communication entre les objets connectés et les serveurs.

— **Couche Application :** cette couche est responsable de l'interface utilisateur et de la fourniture de services IoT aux utilisateurs finaux. Les composants de cette couche peuvent être des applications et des services destinés à fournir certains fonctionnalités ou à résoudre certains problèmes, parmi lesquels, on peut citer les services de l'analyse de données, de contrôle et d'automatisation, de sécurité, de localisation.

5 Modèles de communication de l'IoT

La communication dans l'IoT se repose sur plusieurs modèles, tels que :

— **Machine-to-Machine (M2M) :** est une communication directe entre les objets connectés sans intervention humaine ou intermédiaire.
Exemple : Communication entre véhicules autonomes pour le partage d'informations sur le trafic et les conditions routières.

— **Machine-to-Cloud (M2C) :** implique l'envoi des données des objets connectés à un serveur Cloud pour le traitement et le stockage, rendant les données accessibles aux utilisateurs via des applications.
Exemple : Surveillance de la santé avec collecte de données biométriques et suivi à distance via le Cloud.

— **Cloud-to-Device (C2D) :** permet aux utilisateurs de contrôler à distance les objets connectés via une application Cloud. Les commandes sont transmises au Cloud qui les relaie aux objets connectés.
Exemple : Contrôle domestique intelligent : recevoir des instructions du Cloud pour la gestion des dispositifs.

- **Device-to-Gateway (D2G) :** consiste à envoyer les données des objets connectés à une passerelle (Gateway) qui les traite et les transmet soit au Cloud, soit à d'autres objets connectés.
  Exemple : Transmission de flux vidéo de caméras de surveillance via une passerelle locale vers le stockage Cloud.

- **Gateway-to-Cloud (G2C) :** permet aux passerelles de transmettre les données collectées des objets connectés vers le Cloud pour le traitement et le stockage.
  Exemple : Optimisation agricole grâce à la collecte et à l'analyse de données en temps réel.

- **Device-to-Device (D2D) :** permet aux objets connectés de communiquer directement entre eux sans passer par un serveur ou une passerelle intermédiaire. Ce modèle est idéal pour les environnements où les objets sont proches les uns des autres.
  Exemple : Échange de données (l'humidité de sol, la météo) entre capteurs environnementaux pour optimiser l'irrigation agricole.

Ces modèles de communication dépendent dans leurs utilisations sur des besoins spécifiques de chaque application et des contraintes technologiques et opérationnelles. Les critères de choix peuvent varier en fonction des exigences particulières de chaque application IoT, tels que : la bande passante requise, la latence acceptable, la sécurité des communications, la prise en charge des protocoles appropriés et l'intégration avec les services Cloud [36] [34].

Tableau 3 : les différentes modèles de communications

Modèle de communication	Utilisation	Critères de choix
M2M	Communication autonome entre appareils IoT	Faible consommation d'énergie, sécurité des communications, protocoles basse consommation, évolutivité
M2C	Transmission des données des appareils IoT au cloud	Bande passante élevée, faible latence, intégration avec les services cloud, sécurité des données
C2C	Communication entre différents services cloud.	Interopérabilité, compatibilité des protocoles, sécurité des communications, capacité de traitement dans le cloud
D2G	Communication des appareils IoT avec une passerelle	Portée de communication, protocoles de communication pris en charge, faible latence, faible consommation d'énergie
G2C	Transmission des données de la passerelle vers le cloud.	Bande passante élevée, faible latence, intégration avec les services cloud, sécurité des données
D2D	Communication directe entre appareils IoT	Faible latence, faible consommation d'énergie, prise en charge des protocoles de communication à courte portée, sécurité des communications

6 Caractéristiques de l'IoT

L'IoT présente plusieurs caractéristiques clés qui peuvent être résumées dans ce qui suit [12] :

— **Connectivité :** grâce à l'IoT, chaque objet peut être connecté à une infrastructure mondiale de communication et d'information. Cette connectivité est essentielle pour assurer l'accessibilité et la compatibilité du réseau, ce qui facilite la consommation et la production de données partagées.

- **Services liés aux objets connectés :** l'IoT peut fournir des services associés aux objets tout en respectant les contraintes telles que la protection de la vie privée et la cohérence sémantique entre les objets physiques et leurs homologues virtuels. Les technologies pour les mondes physiques et virtuels doivent évoluer pour fournir ces services.

- **Hétérogénéité :** les appareils IoT sont hétérogènes en termes de plateformes matérielles et de réseaux et peuvent interagir avec d'autres dispositifs ou plateformes via différents réseaux.

- **Changements dynamiques :** l'état des dispositifs IoT peut changer dynamiquement, tels que se mettre en veille, se réveiller, se connecter ou se déconnecter, ainsi que leur contexte comme leur localisation et leur vitesse. Le nombre d'appareils peut également changer de manière dynamique.
- **Échelle Énorme :** le nombre de dispositifs IoT à gérer et qui communiquent entre eux sera au moins d'un ordre de grandeur supérieur à celui des dispositifs connectés à Internet. La gestion et l'interprétation des données générées à des fins d'application seront donc critiques, tout comme la sémantique des données et leur gestion efficace.

7 Avantages et inconvénients de l'IoT

7.1 Avantages

- **Efficacité :** les réseaux IoT permettent aux objets connectés de communiquer directement entre eux, ce qui peut améliorer l'efficacité des processus industriels ou des services.

- **Collecte de données :** les capteurs intégrés dans les objets connectés permettent de collecter des données précieuses sur l'utilisation et le fonctionnement de ces objets.

- **Automatisation :** les réseaux IoT peuvent être utilisés pour automatiser certaines tâches et processus, ce qui peut améliorer l'efficacité et réduire les coûts.

- **Amélioration de la qualité de vie :** l'IoT peut améliorer la qualité de vie en permettant la création de systèmes de soins de santé connectés, de villes intelligentes et de maisons intelligentes.

- **Innovation :** les réseaux IoT permettent de développer de nouvelles applications et services innovants [11].

7.2 Inconvénients

- **Sécurité :** les objets connectés peuvent représenter une menace pour la sécurité des données et des systèmes informatiques s'ils ne sont pas correctement protégés.

- **Coût :** Les objets connectés et les réseaux IoT peuvent être coûteux à mettre en place et à maintenir.

- **Complexité :** les réseaux IoT peuvent être complexes à configurer et à gérer, en particulier si de nombreux équipements sont impliqués.

- **Vie privée :** les objets connectés peuvent collecter des données personnelles sensibles, ce qui soulève des questions de vie privée et de sécurité.

- **Interopérabilité :** les équipements connectés de différents fabricants peuvent utiliser des protocoles et des technologies différents, ce qui peut rendre difficile l'interopérabilité entre ces équipements [1].

8 Défis de l'IoT

L'IoT présente plusieurs défis, notamment [34] :

- **Sécurité :** l'IoT implique la collecte et la transmission de grandes quantités de données sensibles, ce qui rend les objets connectés vulnérables aux attaques de sécurité. Les fabricants d'objets connectés doivent donc mettre en place des mesures de sécurité efficaces pour protéger les données contre les piratages.

- **Interopérabilité :** l'IoT comprend de nombreux dispositifs et réseaux différents, ce qui peut rendre difficile l'interopérabilité entre eux. Les normes et les protocoles de communication doivent donc être harmonisés pour permettre une intégration transparente.

- **Coûts :** les objets connectés peuvent être coûteux, ce qui peut limiter leur adoption à grande échelle. Les fabricants doivent donc travailler à la réduction des coûts de production pour rendre les objets connectés plus accessibles.

- **Vie privée :** l'IoT collecte des données personnelles, ce qui peut soulever des questions de vie privée. Les fabricants doivent donc être transparents quant à la manière dont les données sont collectées, stockées et utilisées, et doivent se conformer aux réglementations de protection de la vie privée.

- **Énergie :** les objets connectés nécessitent une alimentation en énergie, ce qui peut limiter leur autonomie et leur portabilité. Les fabricants doivent donc travailler à l'amélioration de l'efficacité énergétique des dispositifs pour prolonger leur durée de vie.

Par conséquent, ces défis doivent être levés pour permettre une adoption plus large et plus efficace de l'IoT. Ainsi pour garantir que les avantages et les objectifs de l'IoT soient atteints de manière durable.

9 Domaines d'application de l'IoT

L'IoT ouvre la voie à de multiples domaines d'application où les objets physiques peuvent être connectés à Internet pour collecter, échanger et analyser des données. Cette technologie trouve ses applications dans divers secteurs (voir la figure 9). Dans ce qui suit, nous présentons quelques exemples d'applications [13] :

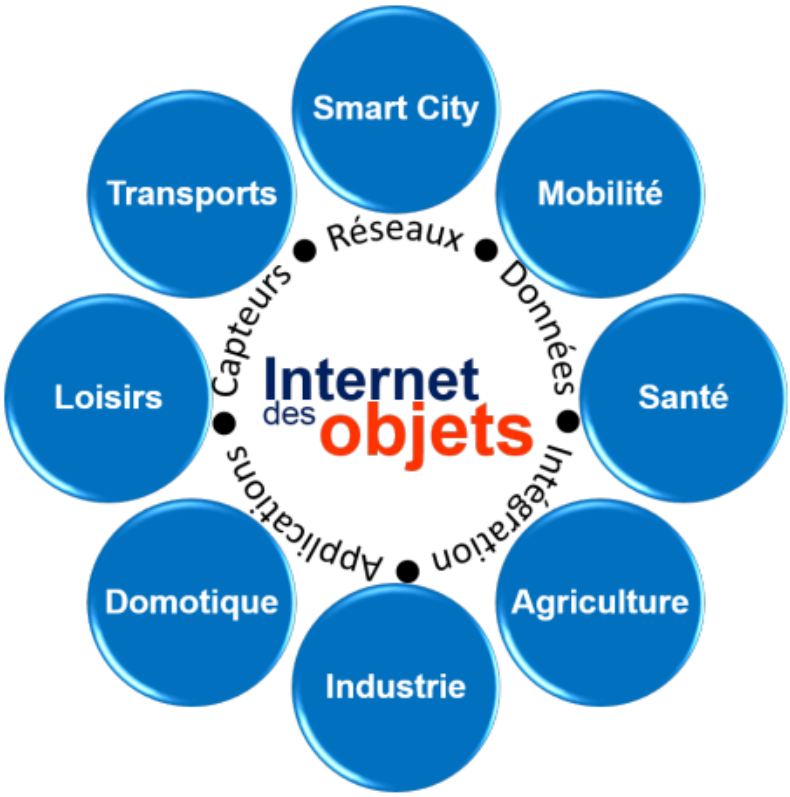

FIGURE 9 – Les domaines d'application de l'IoT [30].

— **Domotique :** l'IoT permet aux utilisateurs de contrôler leur maison à distance via des applications mobiles, de surveiller la sécurité, la température, l'éclairage et les appareils électroménagers.

— **Santé connectée :** l'IoT offre des solutions pour la surveillance à distance des patients, la collecte de données de santé en temps réel, la détection de problèmes de santé et la gestion des médicaments.

— **Industrie :** l'IoT peut être utilisé dans les usines pour surveiller les machines, réduire les temps d'arrêt, optimiser les processus de production et améliorer la qualité des produits.

— **Transport :** l'IoT est utilisé pour la gestion du trafic, la surveillance de la qualité de l'air, le suivi des véhicules, la gestion de flotte et la sécurité routière.

— **Agriculture intelligente :** l'IoT permet la collecte de données sur les cultures, les sols, les conditions météorologiques et les conditions de croissance, aidant les agriculteurs à optimiser les rendements et à réduire les coûts.

— **Ville intelligente :** l'IoT peut être utilisé pour surveiller la qualité de l'air, la consommation d'énergie, la gestion des déchets, la surveillance de la sécurité et l'amélioration de la mobilité urbaine.

— **Sécurité :** l'IoT est utilisé pour la surveillance vidéo, la détection d'intrusion, les systèmes de contrôle d'accès et les systèmes d'alerte.

— **Énergie :** l'IoT est utilisé pour la gestion de l'énergie, la surveillance des compteurs intelligents, la production d'énergie verte et la réduction des coûts énergétiques.

— **Commerce et chaîne d'approvisionnement :** l'IoT peut être utilisé pour la gestion des stocks, la surveillance des rayons, la surveillance des comportements d'achat des clients et la personnalisation de l'expérience d'achat.

— **Sport :** l'IoT est utilisé pour la surveillance de la condition physique, la gestion de l'entraînement, l'analyse des performances et la prévention des blessures.

10 Le Fog computing et les applications IoT

L'architecture du Cloud computing centralisée est confrontée à de grands défis pour les applications IoT. Par exemple, le Cloud ne prend pas en charge les applications IoT sensibles au délai, telles que le vidéo streaming, les jeux en ligne, les véhicules connectés. En outre, l'environnement de Cloud ne supporte pas totalement la localisation des nœuds puisqu'il est centralisé. Le Fog computing est donc conçu pour être en mesure de relever ces défis. Le tableau suivant résume quelques différences entre le Cloud et le Fog [31] :

Tableau 4 : Comparaison entre le Cloud et le Fog computing

critères	Cloud computing	Fog computing
Déploiement	Centralisé	Distribué
Localisation des nœuds	Via internet	Au bord du réseau local
Latence	Elevée	Faible
Evolutivité	Moyenne	Elevée
Coût de déploiement	Elevé	Faible
Matériel	Stockage et puissance de calcul immense (évolutif)	Stockage et puissance de calcul limitée
Distance entre le client et le serveur	Plusieurs sauts	Un saut

Le Fog Computing est une technologie qui permet aux objets connectés de communiquer entre eux et de partager des données en temps réel, tout en offrant une meilleure fiabilité des services et une plus grande efficacité. Les données sont traitées à la périphérie du réseau, ce qui permet aux objets connectés de fonctionner plus rapidement et de fournir des services plus avancés.

De plus, le Fog Computing facilite la communication entre les objets connectés, ce qui permet une meilleure gestion des données et une plus grande flexibilité dans leur utilisation.

11 Conclusion

L'évolution actuelle de l'IoT permet d'améliorer considérablement notre mode de vie grâce à l'utilisation et la vulgarisation des objets intelligents. Ces derniers permettent une interaction accrue entre les différents éléments de notre environnement.
Dans ce chapitre, nous avons présenté un état de l'art sur l'internet des objets, nous avons discuté son architecture, ses domaines d'application ainsi que les problèmes ou les défis qui peuvent ralentir son développement.
A la fin, nous avons mis l'accent sur le Fog Computing et le rôle qu'il peut exercer dans le développement des applications IoT[17].

Cependant, malgré les avancées de l'IoT, la problématique de l'emplacement des tâches dans le Fog reste un défi majeur à relever.
Dans le chapitre suivant, nous aborderons cette problématique en posant solutions efficaces pour optimiser l'emplacement des tâches dans le Fog,afin d'améliorer les performances globales des applications IoT

CHAPITRE III

SIMULATION ET IMPLÉMENTATION

1 Introduction

Nous avons vu dans le premier chapitre (Section 3) que le Fog computing est une infrastructure décentralisée qui étend les fonctionnalités du Cloud computing en rapprochant les ressources de calcul, de stockage et les services des utilisateurs et des objets connectés. Cette approche offre de nombreux avantages, notamment en termes de latence, de bande passante et de sécurité, mais soulève également des défis importants en matière de gestion des ressources et de placement des tâches.

L'affectation optimale des tâches dans un environnement Fog représente un défi majeur en raison des ressources limitées et hétérogènes des nœuds Fog, ainsi que des différentes exigences des applications telles que la latence, la localisation, etc. Pour garantir un placement efficace, il est essentiel de prendre en compte les caractéristiques du réseau, les capacités de traitement des nœuds, les contraintes de communication et les préférences des utilisateurs.

Notre objectif est de concevoir un modèle approprié d'un système IoT et le simuler dans un environnement de Fog computing en mettant en œuvre une stratégie de placement qui sera détaillée dans les sections qui suivent. Ce système IoT est en mesure de prendre en charge des services avec plusieurs contraintes, telles que la capacité de mémoire et de calcul , ainsi que le temps de communication et de traitement.

2 Placement des tâches

Dans un environnement de fog computing où les tâches de calcul sont réparties entre les nœuds fog, une stratégie de placement de tâches peut être conçue en fonction de certains facteurs tels que la disponibilité des nœuds, l'utilisation des ressources et la proximité de la source de données.

L'objectif est de faire correspondre la tâche avec une ressource qui peut fournir les capacités de calcul nécessaires et répondre aux exigences de cette tâche.
Supposons que nous disposions d'une infrastructure de Fog computing composée de plusieurs nœuds Fog dont les capacités de calcul et la disponibilité des ressources varient. Chaque nœud Fog dispose de capacités différentes en termes de CPU, de mémoire et de stockage. En outre, chaque tâche nécessite des ressources spécifiques en termes de ces facteurs.

Lorsqu'une tâche doit être exécutée, la mise en correspondance des ressources consiste à trouver le nœud Fog qui correspond le mieux aux besoins en ressources de la tâche. A cet effet, les étapes suivantes sont nécessaires :

- **Besoins en ressources de la tâche :** la tâche qui doit être placée spécifie ses besoins en ressources, tels que la capacité de CPU et de mémoire dont elle a besoin pour être exécutée. Ces exigences peuvent être définies en fonction de la nature de la tâche, de sa complexité et de l'utilisation prévue des ressources.

- **Informations sur les ressources des nœuds Fog :** les nœuds Fog fournissent des informations sur leurs ressources disponibles, y compris les ressources de calcul, de traitement et de stockage. Ces informations peuvent être obtenues par le biais de mécanismes de surveillance ou de gestion des ressources dans l'environnement de Fog compuitng.

- **Algorithme de correspondance (Matching algorithm) :** un algorithme de correspondance est utilisé pour comparer les besoins en ressources de la tâche avec les ressources disponibles des nœuds Fog. L'algorithme prend en compte les ressources requises tels que l'utilisation de l'unité centrale, la mémoire disponible et la capacité de stockage avec celles disponibles sur les nœuds Fog.

- **Meilleure correspondance de ressources :** sur la base de l'algorithme de correspondance, le nœud Fog qui correspond le mieux aux ressources nécessaires de la tâche est sélectionné. Ce nœud doit fournir les ressources nécessaires pour une exécution efficace de la tâche qui y placée.

Le processus de mise en correspondance des ressources garantit que les tâches sont placées sur des nœuds Fog qui peuvent les traiter efficacement sans contraintes de ressources. En associant les tâches aux ressources appropriées, il est possible d'optimiser les performances globales, l'efficacité et la qualité du service dans l'environnement de Fog computing.
Ainsi, différents algorithmes et techniques, tels que les algorithmes de satisfaction des contraintes, d'apprentissage automatique ou d'optimisation, peuvent être appliqués pour effectuer une mise en correspondance efficace des ressources [22].

2.1 Stratégies de placement

Dans cette section, nous survolons quelques algorithmes et techniques de placement couramment utilisés :[14]

2.1.1 Les algorithmes traditionnels

Sont des approches classiques simples à comprendre et à mettre en œuvre. Ils sont statiques ce qui signifie que toutes les informations sur les tâches et les ressources sont connues à l'avance. Parmi ces algorithmes :

- **Premier arrivé, premier servi (FCFS) :** les tâches sont exécutées dans l'ordre de leur arrivée. Lorsqu'une tâche est reçue, elle est placée à la fin de la file d'attente pour être exécutée en fonction de son ordre d'arrivée.

- **Round Robin (RR) :** les tâches sont également exécutées dans l'ordre d'arrivée, mais en leur affectant à un intervalle de temps de processeur appelé quantum. Si une tâche ne termine pas son exécution pendant le quantum de temps défini, elle est interrompue et la tâche suivante dans la file d'attente est exécutée.
- **Min-Min :** cet algorithme sélectionne et exécute d'abord la tâche la plus petite sur les machines disponibles, puis attribue une ressource à la tâche qui permettra d'obtenir le temps d'achèvement minimum.
- **Max-Min :** cet algorithme sélectionne et exécute d'abord les tâches volumineuses sur les machines disponibles, ce qui peut entraîner des problèmes de famine pour les petites tâches.
- **Temps d'achèvement minimum (MCT) :** cet algorithme sélectionne et exécute la tâche dont l'heure d'achèvement est la plus proche.
- **Algorithme de priorité :** chaque tâche se voit attribuer une priorité et est exécutée en fonction de cette priorité. Les tâches ayant la même priorité sont exécutées selon le principe du premier arrivé, premier servi.

Il convient de noter que les performances de ces algorithmes peuvent varier en fonction des applications et des environnements dans lesquels sont utilisés.

2.1.2 Les heuristiques :

Les algorithmes heuristiques sont des approches de résolution de problèmes de planification et allocation des ressources qui utilisent des heuristiques pour orienter la recherche vers des solutions acceptables et réalisables. Ces méthodes se concentrent sur la découverte d'une solution réalisable en utilisant une stratégie de recherche locale pour explorer l'ensemble des solutions possibles.

2.1.3 Les métaheuristiques :

Les algorithmes méta-heuristiques sont populaires pour résoudre des problèmes complexes de planification des tâches et d'allocation des ressources dans des environnements en explorant un grand nombre de solutions possibles.

2.1.4 La programmation linéaire en nombres entiers :

La programmation linéaire en nombres entiers (ou ILP : Integer linear programming) et l'une des techniques mathématiques utilisées pour résoudre des problèmes d'optimisation où certaines variables de décision doivent prendre des valeurs entières ou binaires.
Dans le domaine de l'allocation des ressources, des problèmes tels que le placement de services de Fog ont été résolus à l'aide de l'ILP. Ces problèmes visent à allouer des nœuds de Fog aux services IoT tout en respectant des contraintes de qualité de service (QoS).
L'objectif est souvent de maximiser l'utilisation des ressources des nœuds de Fog . De même, dans le cadre de l'ordonnancement dans un environnement Fog ou Cloud, des approches basées sur ILP et MILP (Mixed ILP) ont été utilisées pour minimiser conjointement le délai de calcul et de communication.

2.1.5 L'apprentissage automatique :

Ces algorithmes tirent parti des techniques d'apprentissage automatique pour améliorer la précision et l'adaptabilité des décisions de placement des tâches dans l'environnement Fog computing. L'objectif est d'optimiser les performances globales du système en prenant en compte divers facteurs tels que les préférences des utilisateurs, les contraintes de communication et les caractéristiques spécifiques des tâches à exécuter[4].
Ce type d'algorithmes repose sur l'utilisation de modèles d'apprentissage automatique préalablement construits, tels que des réseaux de neurones, des arbres de décision ou des machines à vecteurs de support. Ces modèles sont formés sur des données historiques ou générées par simulation, afin d'apprendre les relations entre les caractéristiques des tâches, les capacités des nœuds Fog et les performances de traitement attendues.
Une fois que les modèles d'apprentissage automatique sont entraînés, l'algorithme peut prédire la performance de traitement des tâches sur les nœuds Fog. Ces prédictions sont ensuite utilisées pour prendre des décisions sur le placement des tâches. L'algorithme cherche à trouver la combinaison optimale de placement des tâches qui maximise les performances du système tout en respectant les contraintes spécifiques.

Il convient de noter que la performance de ces algorithmes dépend de la qualité et de la représentativité des données d'entraînement utilisées pour les modèles d'apprentissage automatique.

2.1.6 La théorie des jeux :

Les algorithmes utilisant la théorie des jeux pour prendre des décisions de placement de tâches dans l'environnement Fog, prennent en compte différents critères tels que le coût de traitement, la latence de transmission et la capacité de stockage des nœuds Fog. Ils utilisent des modèles de jeu pour modéliser l'interaction entre les nœuds Fog et les tâches à placer et optimisent la répartition des tâches en prenant en considération les ressources disponibles sur ces derniers [3]. L'objectif principal de l'algorithme basé sur la théorie des jeux est d'optimiser la répartition des tâches de manière équilibrée et efficace en prenant en compte les ressources disponibles sur les nœuds Fog. Cela permet de maximiser les performances du système, en minimisant les coûts de traitement, en réduisant la latence de transmission et en optimisant l'utilisation des ressources.

2.1.7 Algorithmes gloutons :

Les algorithmes gloutons constituent une approche simple et intuitive utilisée dans le placement des tâches dans l'environnement Fog. Ils permettent de choisir la meilleure option à chaque étape, en se basant uniquement sur l'information disponible localement. Cela peut être basé sur des critères tels que la disponibilité des ressources, la capacité de traitement, la proximité géographique, ou d'autres contraintes spécifiques. Il convient de noter que le choix de l'algorithme de placement dépend de plusieurs facteurs tels que la configuration système requise, l'évolutivité, la complexité de calcul et les objectifs d'optimisation. Des approches hybrides combinant plusieurs algorithmes peuvent également être utilisées pour améliorer l'efficacité du placement des tâches dans les environnements de fog [Réf02].

Quant à notre travail, nous appliquons, dans ce qui suit, les principes d'un algorithme glouton pour résoudre le problème du placement des tâches dans un environnement de Fog computing. Ceci est réalisé en adoptant une approche simple et intuitive qui consiste à prendre localement la meilleure décision à chaque étape.

3 Outils de simulation

3.1 Simulateur OMNET++

3.1.1 Définition

OMNeT++ [5] (Objective Modular Network Testbed in C++) est un simulateur de réseau à évènements discrets, modulaire orienté objet écrit en C++.

OMNeT++ fournit une bibliothèque de modules prêts à l'emploi qui permettent de modéliser différents aspects d'un réseau, tels que les nœuds, les canaux de communication, les protocoles de routage, les applications, etc. Les utilisateurs peuvent également développer leurs propres modules personnalisés pour répondre aux besoins spécifiques de leur système.

L'avantage d'utiliser OMNeT++ réside dans sa flexibilité et son extensibilité. L'architecture modulaire d'OMNeT++ permet de combiner différents modules pour créer des modèles de réseau complexes. De plus, la simulation événementielle d'OMNeT++ permet de capturer les interactions entre les composants du réseau et de modéliser leur comportement dynamique [33].

Plusieurs exemples de modèles peuvent être simulés avec OMNeT++ ; tels que les algorithmes de routage, les modèles de trafic, files d'attente, etc.

3.1.2 Caractéristiques d'OMNET++

OMNeT++ est un framework de simulation qui présente plusieurs caractéristiques distinctives :

- **Noyau de simulation :** OMNeT++ fournit un noyau de simulation qui permet d'exécuter des simulations de réseaux et de systèmes complexes. Il offre un environnement de modélisation et de simulation puissant pour étudier et analyser le comportement des systèmes.
- **Éditeur graphique de réseau :** OMNeT++ propose un éditeur graphique de réseau pour les fichiers NED (Network Description). Cet éditeur permet de créer des modèles de réseaux en définissant des modules, des connexions et des paramètres.
- **Outils de traçage de données :** OMNeT++ offre des outils intégrés qui permettent de tracer et d'analyser les données générées pendant la simulation. Ces outils facilitent la visualisation des résultats, la compréhension du comportement du système et l'identification des problèmes éventuels.

5. https ://omnetpp.org

- **Compilation de langages de description de topologie NED :** OMNeT++ prend en charge la compilation de langages de description de topologie NED. Cela signifie que vous pouvez utiliser différents langages pour décrire la topologie des réseaux dans vos modèles de simulation, ce qui offre une flexibilité dans le choix des langages de modélisation.
- **Interfaces utilisateur :** OMNeT++ propose deux types d'interfaces utilisateur pour exécuter les simulations. Il offre une interface en ligne de commande, qui permet d'exécuter les simulations en utilisant des scripts et des commandes, ainsi qu'une interface graphique conviviale qui facilite l'interaction avec les modèles et l'observation des résultats de simulation [38].

3.1.3 Avantages d'OMNeT++

- Flexibilité dans la modélisation des systèmes.
- Approche modulaire permettant la réutilisation des composants.
- Capacité à exécuter des simulations parallèles pour une meilleure efficacité.
- Prise en charge de plusieurs langages de programmation.
- Large éventail de fonctionnalités pour la modélisation et la simulation.
- Communauté active offrant un soutien et des ressources supplémentaires.

3.2 Le framework INET

INET est un package de simulation spécialement conçue pour le simulateur OMNeT++. Il fournit une collection complète de modèles et de protocoles de réseau pré-implémentés, ce qui facilite grandement le processus de modélisation et de simulation.

INET couvre une large gamme de protocoles et de technologies de réseau, tels que les réseaux sans fil, les réseaux ad- hoc, les réseaux de capteurs, etc. Il est souvent utilisé comme point de départ pour les simulations de réseaux dans OMNeT++ [10].

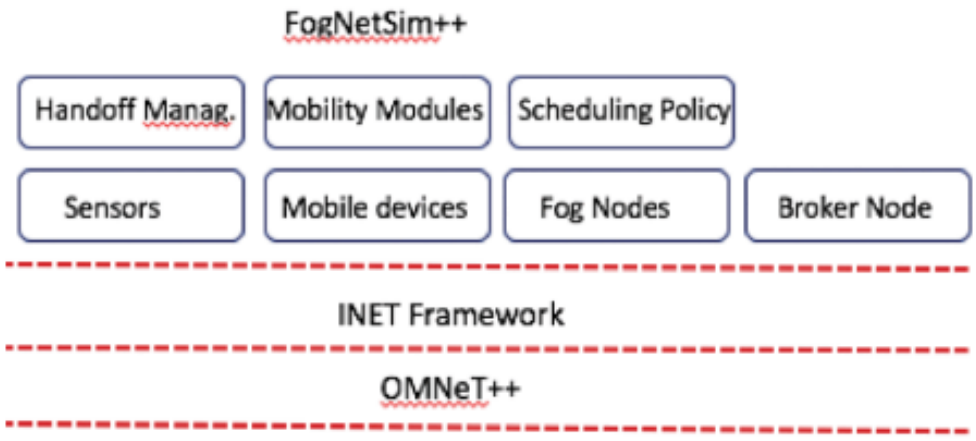

FIGURE 10 – Architecture du framework INET [41].

3.3 Le simulateur FogNetSim++

FogNetSim++ est un simulateur qui permet de modéliser et simuler des réseaux de fog computing. Elle étend les fonctionnalités d'OMNeT++ en fournissant des modules spécifiques pour représenter les composants du fog computing tels que les serveurs de périphérie, les dispositifs IoT et les services de traitement des données.

L'utilisation conjointe de FogNetSim++ avec INET dans OMNeT++ permet une modélisation détaillée et réaliste des réseaux de fog computing, offrant ainsi aux chercheurs et aux ingénieurs un outil puissant pour évaluer les performances et concevoir des systèmes innovants basés sur le Fog Computing.

4 Installation et configuration de l'environnement de travail

L'environnement de travail utilisé est présenté dans le tableau ci dessous :

Tableau 5 : environnement de travail

Hardware	Processeur	Intel Core i3-4005 CPU @.1.70GHz
	RAM	4 Go
Software	Système d'exploitation	Ubuntu 16.04.7 LTE
	OMNET++	4.6
	INET	3.3.0

4.1 Pré-configuration

Pour installer OMNeT++, il est primordial d'effectuer quelques configuration qui sont détaillées dans les étapes mentionnées ci-dessous :

— Tout d'abord, nous vous recommandons de mettre à jour votre système Linux en tapant la commande suivante dans le terminal :

```
$ sudo apt-get update
```

— Une fois le système mis à jour, vous pouvez installer les packages nécessaires en exécutant la commande suivante :

```
$ sudo apt-get install build-essential gcc g++ bison
    flex perl python python3 qt5-default
   libqt5opengl5-dev tcl-dev tk-dev libxml2-dev
   zlib1g-dev default-jre doxygen graphviz
   libwebkitgtk-1.0
```

— Si vous souhaitez utiliser la visualisation des objets et flux de réseau dans OMNeT++ sur Ubuntu 16.04, vous pouvez installer le package suivant :

```
$ sudo add-apt-repository ppa:ubuntugis/ppa
$ sudo apt-get update
$ sudo apt-get install openscenegraph-plugin-
   osgearth libosgearth-dev
```

— Si vous souhaitez activer la prise en charge de la simulation parallèle (optionnelle), vous pouvez exécuter la commande suivante :

```
$ sudo apt-get install openmpi-bin libopenmpi-dev
```

— Ensuite, vous pouvez télécharger la version 4.6 d'OMNeT++ à partir du site officiel (https ://omnetpp.org/) et placer l'archive à l'emplacement souhaité.

— Une fois l'archive téléchargée, vous pouvez extraire son contenu en utilisant la commande suivante :

```
$ tar xvfz omnetpp-4.6-src.tgz
```

— Après l'extraction, vous devez accéder au répertoire OMNeT++ en utilisant la commande suivante :

```
cd omnetpp-4.6/
```

— Ensuite, vous pouvez ouvrir le fichier .bashrc en utilisant un éditeur de texte, par exemple : gedit /.bashrc

— Ajoutez la ligne suivante dans le fichier .bashrc, puis enregistrez-le :

```
export PATH=/chemin/vers/omnetpp-4.6/bin:$PATH
```

— Assurez-vous de remplacer "/chemin/vers" par le chemin réel vers le répertoire où vous avez extrait OMNeT++.

— Après avoir enregistré le fichier .bashrc, vous pouvez fermer l'éditeur de texte.

4.2 Installation d'OMNET++

Une fois la phase de préconfiguration est achevée, nous installons OMNeT++à travers les commandes suivantes :

— Nous lançons le script ". /configure" à partir de la ligne de commande pour configurer OMNeT++ en fonction de notre système Linux.

— Après la configuration réussie, nous utilisons la commande "make" pour compiler OMNeT++. La durée de cette étape peut varier en fonction des performances de notre machine.

— Une fois la compilation terminée, nous exécutons la commande omnetpp dans le terminal pour exécuter OMNeT++sur notre système Linux.Si l'interface graphique d'OMNeT++ s'ouvre sans erreur, cela confirme que l'installation a été réussie.

L'étape suivante consiste à télécharger le package INET 3.3.0 depuis le site web d'OMNeT++ et à l'ajouter à notre installation en tant que projet. Nous avons deux méthodes pour lier INET à OMNeT++ :

Méthode 1 : Téléchargement direct depuis le site web d'OMNeT++ :

— Nous nous rendons sur le site inet.omnetpp.org et téléchargeons la version INET 3.3.0.

— Nous extrayons le contenu de l'archive téléchargée.

— Dans l'interface graphique d'OMNeT++, nous allons dans le menu 'Aide → Modèles de simulation'.

— Nous repérons INET dans la liste et l'installons à partir de là en suivant les instructions fournies.

— Une fois l'installation terminée, nous pourrons exécuter les exemples existants d'INET.

Méthode 2 : Importation en tant que projet dans OMNeT++ :

- Nous téléchargeons le package INET 3.3.0 depuis le site inet.omnetpp.org.
- Dans l'interface graphique d'OMNeT++, nous créons un nouveau projet.
- Nous sélectionnons l'option d'importation du projet et indiquons le répertoire où nous avons téléchargé INET.
- Nous suivons les instructions pour importer le projet INET dans OMNeT++.
- Une fois l'importation terminée, nous pourrons exécuter les exemples existants d'INET.

4.3 Installation de FogNetSim++

FogNetSim++ est disponible sur la plateforme GitHub (github.com/ rtqayyum/-fognetsimpp) sous forme de deux archives ces fichiers téléchargés, nous suivons les étapes suivantes :
Étape 1 : Extraire le contenu de 'mqttapp.zip' et placez-le dans le répertoire approprié d'INET. Pour ce faire, nous ouvrons le projet INET dans OMNeT++, puis développons les dossiers suivants : src → applications. Nous collons le dossier 'mqttapp' à cet emplacement. Veuillez noter que le répertoire peut varier selon la configuration de votre projet INET.

Étape 2 : Importer le projet 'fognetsimpp.zip' dans OMNeT++. Nous sélectionnons l'option d'importation de projet et indiquons le chemin du fichier 'fognetsimpp.zip'. Une fois le projet importé, nous effectuons un clic droit sur le projet et ajoutons une référence à INET. Cela permettra à FogNetSim++ de tirer parti des fonctionnalités fournies par INET.

5 Architecture de simulation

Dans cette section, nous décrivons l'architecture modulaire sur laquelle est basé le simulateur OMNeT++ et qui sera utilisée pour notre simulation.

Un modèle de simulation est constitué de modules qui communiquent entre eux en échangeant des messages. Les modules actifs sont appelés "modules simples" et sont écrits en C++ en utilisant la bibliothèque de classes de simulation. Les modules simples peuvent être regroupés pour former des "modules composés", et ainsi de suite, permettant un niveau de hiérarchie illimité.

L'ensemble du modèle, appelé "réseau" dans OMNeT++, est lui-même un module composé. Les messages peuvent être envoyés à d'autres modules soit via des connexions qui traversent plusieurs modules, soit directement.

La figure ci-dessous, illustre un exemple de modules simples et modules composés. Les flèches représentent les connexions entre ces modules à travers les ports.

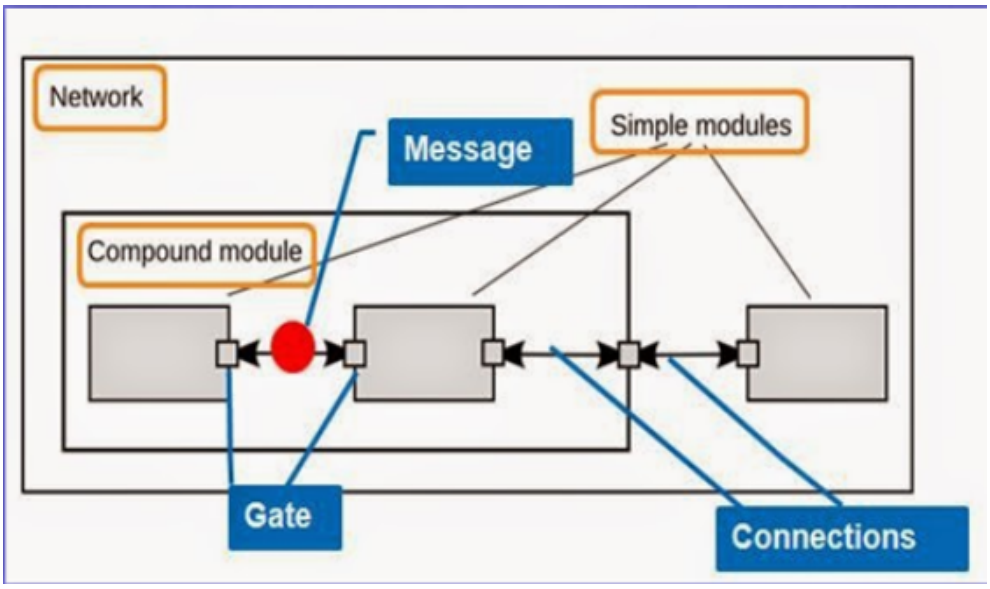

FIGURE 11 – Architecture du simulateur OMNET++ [45].

L'architecture d'OMNeT++ repose sur l'échange de messages entre les modules, qui peuvent contenir des données de manière flexible. Les modules simples envoient généralement des messages via des ports,mais il est également possible de les envoyer directement aux modules destinataires.

Les ports agissent comme des interfaces d'entrée et de sortie des modules : les messages sont émis par les ports de sortie et reçus par les ports d'entrée. Les connexions sont établies entre une port d'entrée et une port de sortie, ce qui permet le transfert des messages. Cependant, les connexions ne peuvent être établies qu'à l'intérieur d'un seul niveau de hiérarchie de module.

À l'intérieur d'un module composite, il est possible d'établir des correspondances entre les sous-modules ou entre un sous-module et le module composite. Cependant, les connexions ne peuvent pas couvrir plusieurs niveaux de hiérarchie. En raison de la structure hiérarchique du modèle, les messages suivent généralement une chaîne de connexions, en commençant et en aboutissant aux modules simples. Cette architecture permet une représentation modulaire et flexible des systèmes dans OMNeT++, favorisant ainsi une modélisation précise et efficace des réseaux et des systèmes de communication.

Les principaux types de fichiers utilisés dans OMNeT++ sont :

- **Fichier .ini (Initialization) :** il s'agit du fichier de configuration principal d'un projet OMNeT++. Il spécifie les paramètres de simulation tels que la durée de la simulation, les modules à utiliser,les paramètres de chaque module, etc. Le fichier .ini est utilisé pour définir les conditions initiales de la simulation [**ref45**].

- **Fichier .ned (Network Description) :**lLe fichier NED est utilisé pour décrire la structure du réseau et les composants qui le composent. Il utilise une syntaxe similaire au langage C++ pour définir les modules, leurs paramètres, leurs connexions et leurs hiérarchies. Le fichier NED est utilisé pour construire la topologie du réseau [**ref45**].

- **Fichier .cc (C++ source code) :** les fichiers .cc contiennent le code source en langage C++ qui implémente la logique et le comportement des modules définis dans le fichier NED. Ces fichiers contiennent les fonctions et les méthodes qui définissent le comportement des modules pendant la simulation. Ils sont compilés avec le simulateur OMNeT++ pour exécuter la simulation .

- **Fichier .msg :** les modules communiquent entre eux en échangeant des messages. Cela peut être déclaré dans un fichier avec une extension (.msg) ou vous pouvez ajouter des champs de données. OMNeT ++ traduira les définitions de message en classes C ++ [**ref45**].
- **Fichiers de résultats :** pendant l'exécution de la simulation, OMNeT++ génère des fichiers de résultats qui enregistrent les mesures et les données collectées pendant la simulation. Ces fichiers de résultats peuvent inclure des données de performance, des statistiques de réseau, des événements de simulation, etc.

6 Scénario de simulation et résultats

6.1 Modèle

Le modèle présenté dans la figure 12 est un exemple d'environnement de Fog computing qui est composée de nombreux modules tels que :

- **Module Fog :** c'est le module noyau constituant l'environnement de Fog Computing. Dans le cas de ce scénario, ce module est instancié en un ensemble de noeuds (fog1, fog2, ...,fog7) .

Chaque nœud Fog possède des caractéristiques spécifiques telles que la capacité de mémoire (MB ou GB), la capacité de traitement (MIPS) et un identifiant unique sur le réseau (ID).

— **Module Broker :** c'est un module intermédiaire qui est responsable de la gestion des ressources disponibles et de la coordination entre les nœuds utilisateurs et les nœuds Fog. Dans le modèle simulé, chaque instance (BaseBroker) gère un ou plusieurs nœuds Fog.

— **Module User :** il s'agit des utilisateurs ou des dispositifs terminaux du système IoT. Les instances de ce module (user1, user2,....,user6) sont responsables de la création et de l'envoi des requêtes de services (tâches à exécuter) au niveau des nœuds Fog.

D'autres modules sont aussi nécessaires pour l'interconnexion et le fonctionnement du réseau, tels que :

— **Module Router :** ce module est responsables du routage des paquets de données à travers le réseau.

— **Module AccessPoint (AP) :** il s'agit des points d'accès sans fil IEEE 802.11.Ils sont utilisés pour connecter les utilisateurs ou les objets aux autres module du réseau. Chaque point d'accès à des propriétés spécifiques, telles que le délai de transmission et le débit de communication.

Ces composants interagissent les uns avec les autres à travers des connexions définies dans le modèle, simulé sous Fognetsim++.

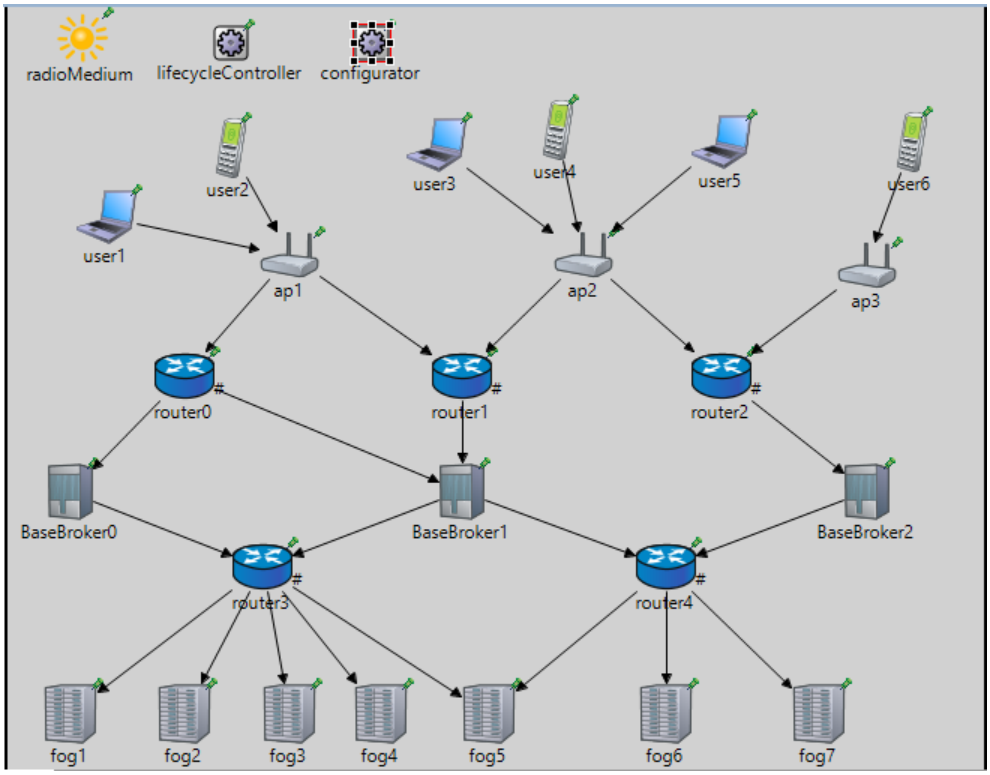

FIGURE 12 – La topologie du modèle simulé.

6.2 Algorithme de placement proposé

Pour garantir un placement de tâches efficace dans le modèle de simulation présenté ci-dessus, nous proposons dans ce travail une stratégie de placement basée sur le choix de meilleur nœud fog selon des critères donnés. Cette stratégie est mise en œuvre à travers l'algorithme présenté dans ce qui suit.

Algorithm 1 BestFog

processMessage

```
1: function processMessage(msg : MyMessage)
2:     bestFogList ← []
3:     messageSize ← msg.getmessageSize()
4:     mips ← msg.getmips()
5:     messagesAffected ← 0
6:     messagesRemaining ← 0
7:     messageWaitingList ← []
8:     messagesInWaitingList ← 0
9:     bestFogListNotEmpty ← false
10:    for foginfogList do
11:        fog.capacityDiff ← fog.capacity − messageSize
12:        fog.mipsDiff ← fog.mips − mips
13:        if fog.capacityDiff ≥ 0andfog.mipsDiff ≥ 0 then
14:            bestFogList.push_back(fog)
15:            bestFogListNotEmpty ← true
16:    if bestFogListNotEmptyandmessageSize ≤ bestFogList[0].capacity then
17:        bestFog ← getBestFog()
18:        if bestFogexistsandbestFog.capacityDiff ≥ 0andbestFog.mipsDiff ≥ 0 then
19:            bestFog.capacity ← bestFog.capacity − messageSize
20:            bestFog.mips ← bestFog.mips − mips
21:            bestFog.messageCount ← bestFog.messageCount + 1
22:            messagesAffected ← messagesAffected + 1
23:            msg.setDestination(bestFog.id)
24:            send(msg,"outToRouter")
25:            updateTotalMessagesAffected()
26:        else
27:            addedToWaitingList ← false
28:            for foginbestFogList do
29:                if fog.capacityDiff ≥ 0andfog.mipsDiff ≥ 0 then
30:                    fog.capacity ← fog.capacity − messageSize
31:                    fog.mips ← fog.mips − mips
32:                    fog.messageCount ← fog.messageCount + 1
33:                    messagesAffected ← messagesAffected + 1
34:                    msg.setDestination(fog.id)
35:                    send(msg,"outToRouter")
36:                    addedToWaitingList ← true
37:                    exitloop
38:            if addedToWaitingListisfalse then
39:                if messageSize > getMaxFogCapacity() then
40:                    deletemsg
41:                    messagesRemaining ← messagesRemaining + 1
42:                else
43:                    messageWaitingList.push_back(msg)
44:                    messagesInWaitingList ← messagesInWaitingList + 1
45:                updateTotalMessagesAffected()
```

Cette fonction prend en entrée un message et cherche le meilleur Fog disponible pour le traiter. Si aucun Fog ne peut traiter le message immédiatement, le message est mis en attente jusqu'à ce qu'un Fog soit disponible. Si la taille du message est supérieure à la capacité maximale d'un Fog, alors le message est supprimé. La fonction met également à jour le nombre total de messages traités.

processWaitingMessages

```
1:  function processWaitingMessages(fog : FogInfo)
2:      waitingList ← fog.waitingList
3:      messagesToProcess ← []
4:      for msginwaitingList do
5:          messageSize ← msg.getmessageSize()
6:          mips ← msg.getmips()
7:          if fog.capacityDiff ≥ messageSize and fog.mipsDiff ≥ mips then
8:              fog.capacityDiff ← fog.capacityDiff − messageSize
9:              fog.mipsDiff ← fog.mipsDiff − mips
10:             fog.messageCount ← fog.messageCount + 1
11:             messagesAffected ← messagesAffected + 1
12:             msg.setDestination(fog.id)
13:             send(msg,"outToRouter")
14:             deletemsgfromwaitingList
15:             messagesToProcess.push_back(msg)
16:     for msginmessagesToProcess do
17:         processWaitingMessages(fog)
18:     updateTotalMessagesAffected()
```

Cette fonction prend en entrée un Fog et traite les messages en attente qui peuvent être traités par ce Fog. Les messages traités sont supprimés de la liste d'attente et ajoutés à une liste de messages à traiter plus tard. La fonction répète le processus pour les nouveaux messages à traiter jusqu'à ce qu'il n'y ait plus de messages à traiter. La fonction met également à jour le nombre total de messages traités.

getBestFog

```
1:  function getBestFog
2:      bestFog ← null
3:      minCapacityDiff ← MAX_INT
4:      for foginbestFogList do
5:          if fog.capacityDiff < minCapacityDiff then
6:              bestFog ← fog
7:              minCapacityDiff ← fog.capacityDiff
8:      return bestFog
```

Cette fonction prend une liste de Fog en entrée et renvoie le Fog ayant la capacité restante la plus proche de la taille du message à traiter. La fonction itère sur tous les Fog de la liste et conserve le Fog ayant la plus petite différence de capacité restante avec la taille du message. La fonction renvoie ce Fog.

getMaxFogCapacity

```
1: function GETMAXFOGCAPACITY
2:     maxCapacity ← 0
3:     for foginfogList do
4:         if fog.capacity > maxCapacity then
5:             maxCapacity ← fog.capacity
6:     return maxCapacity
```

Cette fonction prend une liste de Fog en entrée et renvoie la capacité maximale parmi tous les Fog de la liste. La fonction itère sur tous les Fog de la liste et conserve la capacité maximale. La fonction renvoie cette capacité maximale.

Algorithm 2 FCFS (First-Come, First-Served)

Nous avons utilisé l'algorithme FCFS qui permet d'attribuer les messages aux nœuds de fog dans l'ordre de leur arrivée. Les messages sont traités par les nœuds de fog disponibles, tandis que les messages qui ne peuvent pas être traités immédiatement sont placés en attente jusqu'à ce qu'un nœud de fog approprié devienne disponible.Si la taille d'un message dépasse la capacité maximale de tous les nœuds de fog disponibles, le message est supprimé.

Dans notre modèle de simulation, nous avons pris en considération trois paramètres clés, à savoir :

- **La capacité de mémoire :** se réfère à la quantité de données qu'un nœud Fog peut stocker. Ce paramètre représente aussi la quantité par les dispositifs IoT et qui peut être traitée en temps réel, elle est exprimé en MB (Mega Byte). Il convient de noter ici que les nœuds Fog qui disposent d'une grande capacité de mémoire peuvent stocker davantage de données localement, ce qui peut réduire la latence et les coûts de transmission des données vers le cloud.

- **La capacité de calcul :** la capacité d'un nœud Fog en termes de puissance de calcul.

— **La charge de travail des nœuds Fog :** fait référence à la quantité de travail ou de tratement qu'ils doivent effectuer pour exécuter les taches attribuées.

6.3 Résultats et interprétation

Dans l'optique d'évaluer les résultats obtenus, nous présentons dans ce qui suit une comparaison entre deux scénarios de simulation. Le premier scénario applique une stratégie de placement basée sur l'algorithme que nous avons proposé, tandis que le second applique une stratégie de type FCFS.
La figure 13 et 14 représentent le nombre total des tâches affectés correspondants respectivement aux scénarios 1 et 2.

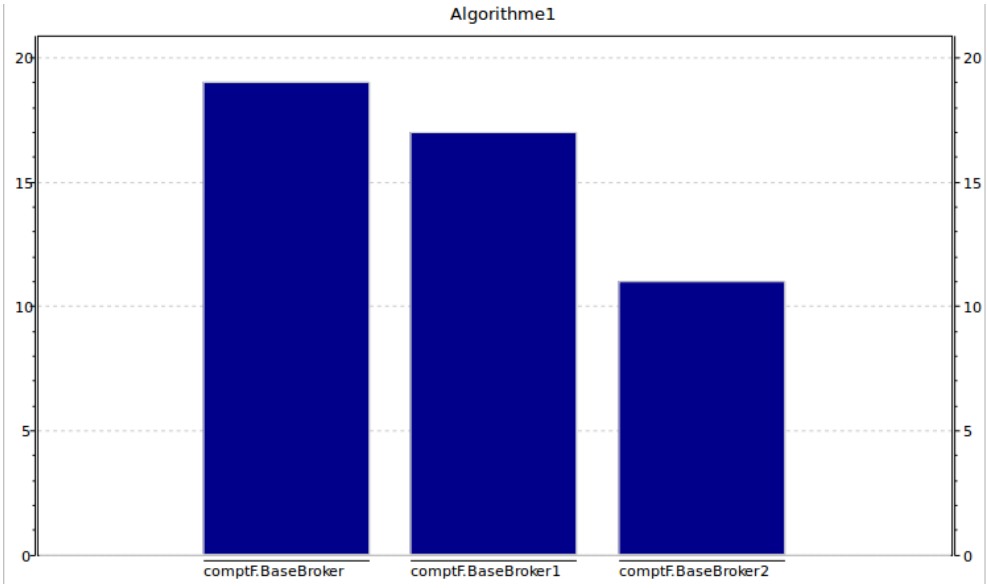

FIGURE 13 – Scénario 1 -Tâches affectées par l'algorithme BestFog

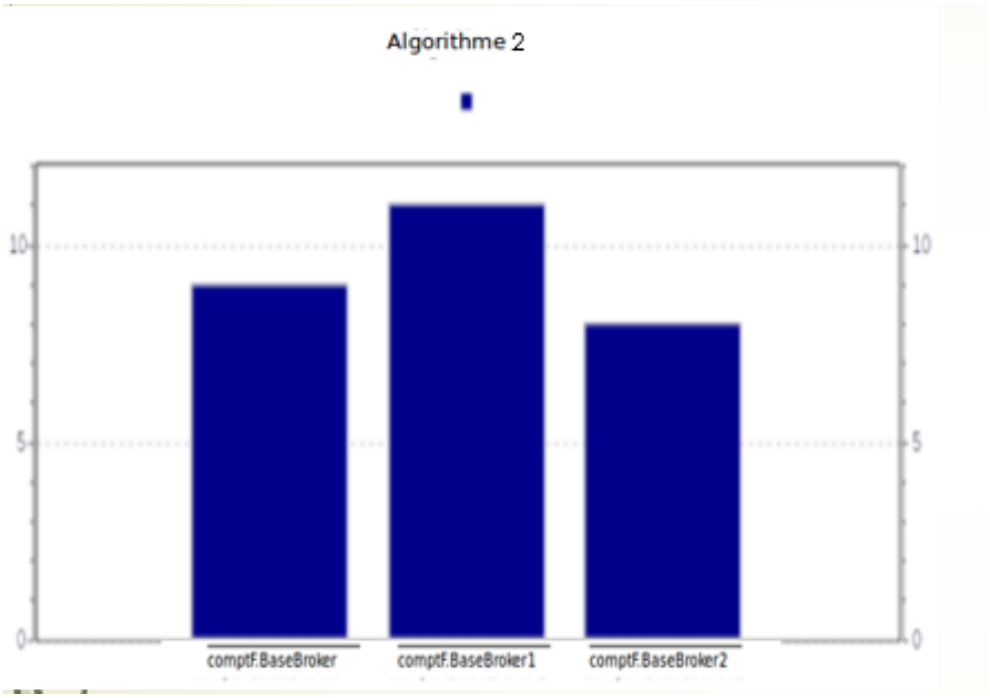

FIGURE 14 – Scénario 2 -Tâches affectées par l' algorithme FCFS

Nous constatons que l'algorithme Best Fog permet d'atteindre un taux de placement plus élevé par rapport à l'algorithme FCFS.En sélectionnant le meilleur nœud Fog pour chaque message en fonction des ressources disponibles. En revanche, l'algorithme FCFS traite les messages dans l'ordre de leur arrivée, ce qui peut entraîner une utilisation restreinte des ressources disponibles.

La figure 15 représente une variation des nombres de messages affectés en fonction de nombre de nœuds Fog.

Nous remarquons que lorsque le nombre de nœuds fog augmente, le nombre de messages pouvant être traités augmente également, car il y a plus de ressources disponibles pour affecter les messages entrants. Donc une planification appropriée du nombre et de la répartition des nœuds Fog peut permettre d'optimiser l'utilisation des ressources et d'améliorer les performances globales du système notamment en termes d'évolutivité.

Cependant, il est important de noter que les deux algorithmes ne parviennent pas à affecter la totalité des messages générés.Les figures 16 et 17 illustre respectivement, le nombre de messages non traité et traités par l'algorithme Best-Fog et l'algorithme FCFS.

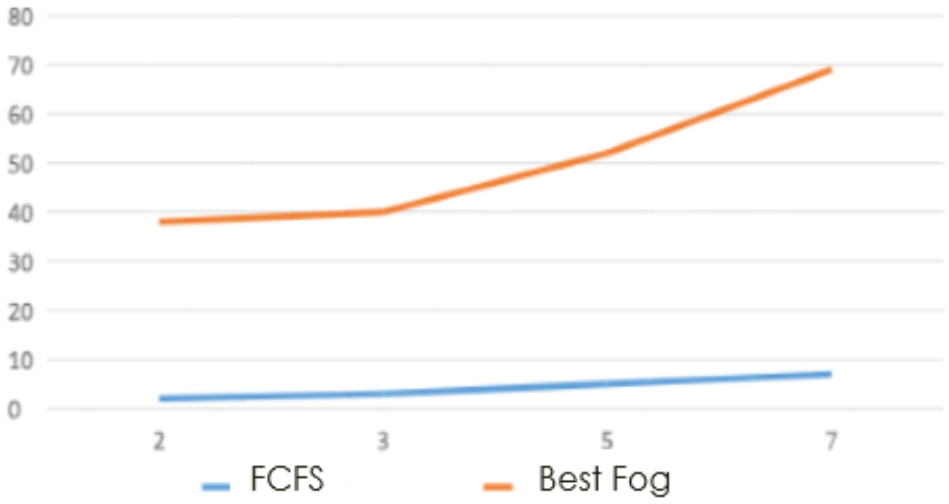

FIGURE 15 – Tâches affectées en fonction du nombre de nœuds Fog

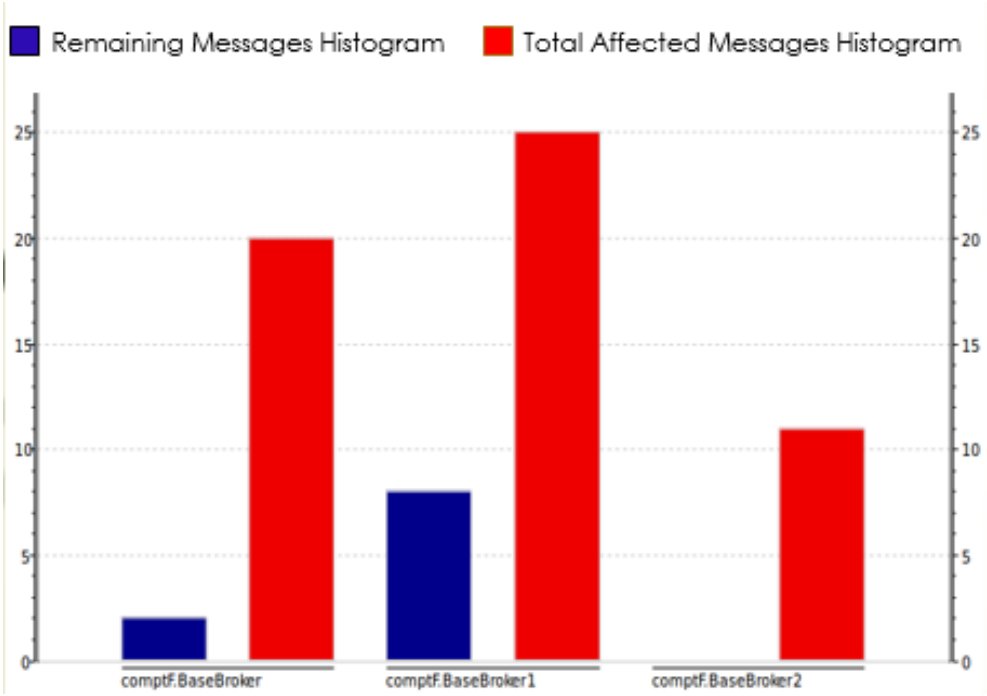

FIGURE 16 – Total des messages affectés/non affectés (Scénario 1).

Le rejet ou le non traitement de certaines tâches est dû au fait que les exigences en termes de mémoire et/ou de calcul étaient plus grands que la capacité des nœuds Fog, Cette limitation est commune aux deux algorithmes mais avec une différence remarquable qui illustre une efficacité de placement plus élevée de l'algorithme BestFog.

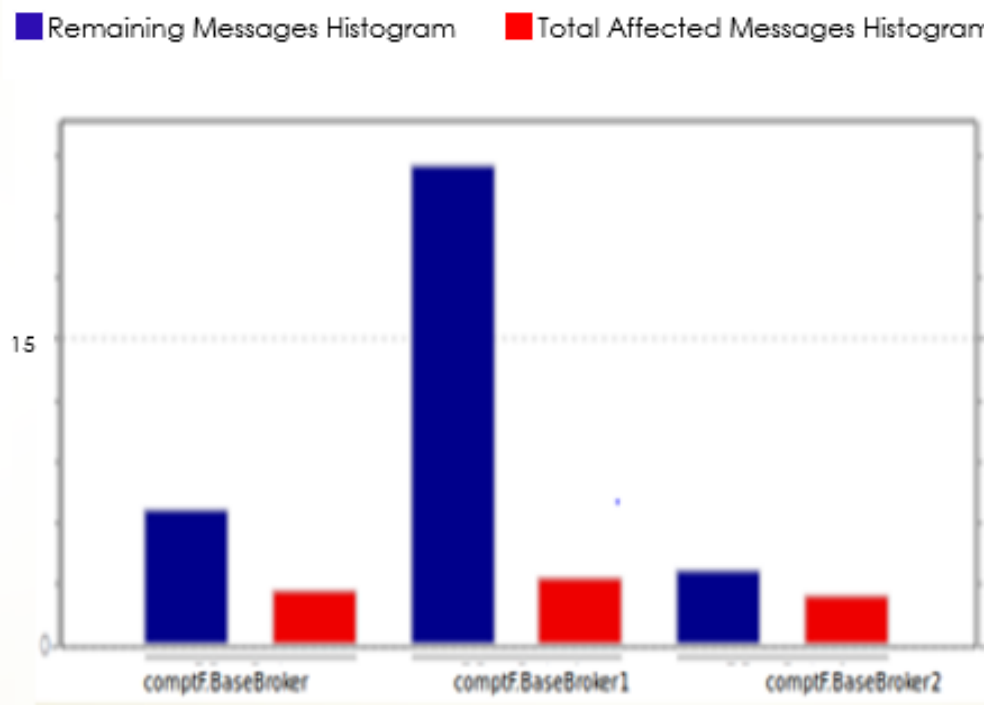

FIGURE 17 – Total des messages affectés/non affectés (Scénario 2).

7 Conclusion

La problématique de l'affectation des tâches dans un réseau de Fog représente un défi majeur dans les systèmes IoT. L'affectation des tâches consiste à déterminer quel nœud Fog sera chargé de traiter chaque tâche provenant des dispositifs IoT connectés.

Afin de résoudre cette problématique,nous avons présenté et évalué dans ce travail une stratégie basée sur l'algorithme « Best fog » pour déterminer le nœud Fog le plus approprié pour exécuter une tâche en prenant en compte plusieurs critères tels que la capacité de mémoire, la capacité de calcul et la charge de travail actuelle du nœud.

Les résultats obtenus montrent que l'algorithme "Best Fog" permet une meilleure utilisation des ressources disponibles et une affectation plus efficace des tâches par rapport à l'algorithme FCFS.

Cependant, il ne garantit pas une solution optimale globale, car il prend des décisions locales à chaque étape. Il est donc important de considérer les caractéristiques spécifiques du réseau de Fog et des tâches à traiter pour choisir l'algorithme le mieux adapté à chaque situation.

En conclusion, bien que la stratégie proposée améliore considérablement l'efficacité de l'affectation des tâches, il reste encore des défis à relever pour traiter de manière optimale les requêtes de grande taille.

CONCLUSION GÉNÉRALE

L'évolution rapide de la technologie IoT a ouvert de nouvelles perspectives en connectant des milliards d'appareils intelligents pour collecter, traiter et échanger des données. Cependant, le défi majeur dans l'architecture IoT réside dans le traitement efficace de ces données générées par les appareils connectés. Les approches traditionnelles basées sur le Cloud computing peuvent présenter des limitations en termes de latence, de bande passante et de capacité de stockage. Cela a donné lieu à l'émergence du Fog computing, une infrastructure qui permet de traiter les données des applications plus près des utilisateurs finaux, réduisant ainsi les délais de transmission et répondant aux exigences de temps réel des applications IoT.

Néanmoins, le déploiement du Fog computing pose également des défis. La gestion et le placement des tâches est l'un des défis majeurs, car il est crucial de déterminer le nœud fog le plus approprié pour exécuter une tâche spécifique. Pour relever ce défi, nous avons proposé l'algorithme "Best Fog" qui utilise une approche basée sur plusieurs critères pour sélectionner le nœud Fog le plus adapté à une tâche spécifique. L'évaluation des résultats de simulation ont montré que l'algorithme "Best Fog" permet une meilleure utilisation des ressources disponibles et une affectation plus efficace des tâches. Cette évaluation est également validée par une analyse comparative entre l'algorithme "Best Fog" et l'algorithme FCFS qui est plus simple mais moins performant.

Cependant, il est important de souligner que la stratégie proposée dans ce travail n'est pas à l'écart de certaines limites telles que la difficulté de traitement des requêtes de grande taille au niveau de la couche Fog, en plus de la gestion de la mobilité des objets qui n'est pas pris en charge dans ce travail. De plus,l'algorithme "Best Fog" pourrait être optimisé en prenant compte les critères supplémentaires tels que la taille et l'urgence des requetés, afin d'améliorer encore plus l'affectation des taches.

Finalement et comme perspective, nous proposons de mettre en œuvre un scénario plus réaliste en tenant compte d'autres paramètres tels que la disponibilité temporelle des nœuds Fog, le temps de traitement des requêtes, le transfert des tâches entre les nœuds Fog et leur affectation au Cloud. Nous prévoyons également d'améliorer le modèle proposé en développant des stratégies plus sophistiquées qui prennent en compte les diverses contraintes et préférences en appliquant des métaheuristiques à titre d'exemple. L'intégration de telles solutions permettra certainement d'atteindre un niveau plus élevé en termes d'efficacité de traitement et de gestion des services IoT dans les environnement de type Cloud-Fog computing.

BIBLIOGRAPHIE DES DOCUMENTS

[2] K. Ashton et al. “That "Internet of Things."”. In : *RFID Journal* (2009).

[3] H. F. Atlam, R. J. Walters et G. B. Wills. “Fog Computing and the Internet of Things : A Review”. In : *Journal of Information and Communication Technologies* (2018).

[4] S. Bhattacharjee et J. Bora. “Cloud Computing : A Survey on Cloud Computing”. In : *International Journal of Advanced Research in Computer Science* (2021).

[6] E. H. Bourhim. “La communication et le placement de conteneurs Docker dans le Fog Computing”. Mémoire de maîtrise. Université du Québec à Montréal, 2020.

[7] R. Buyya et al. “Cloud Computing : Concepts, Architecture and Challenges”. In : *Journal of Cloud Computing* (2010).

[12] L. Djebri. “Prévision de stationnement dans un environnement IoT”. Mémoire de fin d’études. Université Akli Mohand Oulhadj-Bouira, 2019.

[13] L. Djebri et M. Lamraoui. “Prévision de stationnement dans un environnement IoT”. Mémoire de fin d’études. Université Akli Mohand Oulhadj-Bouira, 2019.

[14] I. Djouklafi et A. Benslimane. “L’adaptation d’un protocole IoT dans une architecture IoV”. Mém. de mast. Université Kasdi Merbah – Ouargla, 2019.

[19] N. Hamdani, S. Kerroum et N. Ouallouche. “Etude et comparaison des failles de sécurité d’OpenStack et OpenNebula”. Mémoire fin d’étude master. Université Mouloud Mammeri de Tizi-Ouzou, 2018/2019.

[22] B. Jamil et al. “Resource allocation and task scheduling in fog computing and internet of everything environments : A taxonomy, review, and future directions”. In : *ACM Computing Surveys (CSUR)* (2022).

[24] S. A. Korade, V. Kotak et A. Durafe. “A Review Paper on Internet of Things (IoT) and its Applications”. In : *International Research Journal of Engineering and Technology* (2020).

[26] S. Madakam, R. Ramaswamy et S. Tripathi. “Internet of Things (IoT) : A Literature Review”. In : *Journal of Computer and Communications* (2015).

[27] P. Mateti. “CLOUD COMPUTING OVERVIEW : What is Cloud Computing ?” In : *The Internet Protocol Journal* (2014).

[28] M. Mimoune. “Etude sur la sécurité du cloud computing”. Mémoire fin d’étude master. Université DE MSILA - Technologie de l’Information et de la Communication (TIC), 2014.

[31] Z. M. Nayeri, T. Ghafarian et B. Javadi. “Application placement in Fog computing with AI approach : Taxonomy and a state of the art survey”. In : *Journal of Network and Computer Applications* (2021).

[34] C. Perera et al. “A survey on Internet of Things from industrial market perspective”. In : *IEEE Access* (2014).

[36] S. Raza, L. Wallgren et T. Voigt. “A primer on 3GPP Narrowband IoT”. In : *IEEE Communications Magazine* (2017).

[37] C. Rese. “Cellular IoT Networks : Evolution, Applications, and Challenges”. In : *IEEE Communications Magazine* (2021).

[39] S. Al-Sarawi et al. “Internet of Things (IoT) Communication Protocols : Review”. In : *International Journal of Computer Trends and Technology* (2017).

[44] H. Tschofenig et al. “Architectural Considerations in Smart Object Networking”. In : *International Journal of Computer Trends and Technology* (2015).

[45] A. Yousefpour et al. “A survey on algorithms for task placement in the fog computing environment : Taxonomy and open research issues”. In : *Journal of Network and Computer Applications* (2018).

BIBLIOGRAPHIE DES SITES WEB

[1] *Advantages and Disadvantages of IoT*. Consulté en mars 2023. URL : https://www.aplusto%20pper.com/advantages-and-disadvantages-of-iot.

[5] Blaise Thirard Blog. *Qu'est-ce que le Cloud Computing ?* Consulté en mars 2023. URL : https://blog.blaisethirard.com/qu-est-ce-que-le-cloud-computing/.

[8] Cisco. *What is IoT*. Cisco. Consulté en mars 2023. URL : https://www.cisco.com/c/en/us/solutions/internet-of-things/what-is-iot.html.

[9] *Cloud Computing Architecture*. Consulté en mars 2023. URL : https://www.javatpoint.com/cloud-computing-architecture.

[10] Cloudflare. *What is the Cloud ?* Consulté en mars 2023. URL : https://www.cloudflare.com/fr-fr/learning/cloud/what-is-the-cloud/.

[11] Digi. *The Benefits of IoT : Real-World Examples*. Consulté en avril 2023. URL : https://fr.digi.com/blog/post/the-benefits-of-iot-real-world-examples.

[15] *Fog Computing - Une explication complète*. Consulté en mars 2023. URL : https://geekflare.com/fr/fog-computing/#:~:text=Dans%20le%20fog%20computing%2C%20l,%C3%A9galement%20moins%20cher%20aux%20entreprises.

[16] *Fog Computing Architecture - Working, Advantages and Disadvantages*. Consulté en mars 2023. URL : https://www.rfwireless-world.com/Terminology/Fog-Computing-Architecture-working.html.

[18] Gartner. *Internet of Things*. Gartner. Consulté en mars 2023. URL : https://www.gartner.com/en/information-technology/glossary/internet-of-things.

[20] *Image from Springer*. Consulté en avril 2023. URL : https://media.springernature.com/lw685/springer-static/image/chp%3A10.1007%2F978-981-13-9330-3_30/MediaObjects/469650_1_En_30_Fig1_HTML.png.

[21] Institut Numérique. *Chapitre 1 : Les concepts du Cloud Computing*. Consulté en mars 2023. URL : https://www.institut-numerique.org/chapitre-1-les-concepts-du-cloud-computing-51c0279ca534a.

[23] KINSTA. *Qu'est-ce que l'IoT*. Kinsta. Consulté en avril 2023. URL : https://kinsta.com/fr/base-de-connaissances/qu-est-ce-que-iot/.

[25] LE BIG DATA. *Fog Computing : Guide Complet*. Consulté en mars 2023. URL : https://www.lebigdata.fr/fog-computing-guide-complet.

[30] NAXOO. *L'Internet des Objets (Internet of Things - IoT)*. Consulté en mars 2023. URL : https://www.naxoo.ch/linternet-des-objets-internet-of-things-iot/.

[32] NIST. *Definition of Cloud Computing*. Consulté en mars 2023. URL : https://nvlpubs.nist.gov/nistpubs/Legacy/SP/nistspecialpublication800-145.pdf.

[33] OMNET++ DOCUMENTATION. *OMNeT++ Simulation Manual - User Interfaces*. Consulté en mai 2023. URL : https://doc.omnetpp.org/omnetpp/manual/#sec:ui.

[35] B. POSEY. *TechTarget IoT Agenda, Fog Computing (Fogging)*. Consulté en mars 2023. URL : https://www.techtarget.com/iotagenda/definition/fog-computing-fogging.

[38] RESEARCHGATE. *Proposed iNET Framework Architecture*. Consulté en mai 2023. URL : https://www.resea%20rchgate.net/figure/Proposed-iNET-framework-architecture_fig2_337275596.

[40] T. SASIKALA et M. JAGADEESWARI. *Cloud Computing : Cloud Computing Overview*. Consulté en mars 2023. URL : https://www.institut-numerique.org/chapitre-1-les-concepts-du-cloud-computing-51c0279ca534a.

[41] SLIDEPLAYER. *SlidePlayer - Slide 17440817*. Consulté en mars 2023. URL : https://slideplayer.fr/slide/17440817/.

[42] A. SRIVASTAVA. *Cloud Computing Overview : What is Cloud Computing ?* Consulté en mars 2023. URL : https://www.institut-numerique.org/chapitre-1-les-concepts-du-cloud-computing-51c0279ca534a.

[43] TECHNIQUES DE L'INGÉNIEUR. *Les quatre modèles de déploiement du Cloud Computing*. Consulté en mars 2023. URL : https://www.techniques-ingenieur.fr/base-documentaire/technologies-de-l-information-th9/gestion-de-contenus-numeriques-42311210/cloud-computing-h6020/les-quatre-modeles-de-deploiement-h6020niv10005.html.

Résumé

L'IoT est une technologie révolutionnaire offrant des avantages considérables en matière d'efficacité opérationnelle, de prise de décision et de satisfaction client. Cependant, le fonctionnement efficace de l'IoT nécessite une infrastructure performante, telle que celle du Cloud computing, pour gérer les données et les services, ainsi que les exigences applicatives. Le Cloud computing révèle cependant certaines limites liées notamment à la latence, à la bande passante et aux coûts d'exploitation. Le Fog computing est donc proposé comme étant une solution à ce type de problèmes en offrant une infrastructure de calcul distribuée pour traiter les données et les services à proximité des utilisateurs et des objets connectés, offrant ainsi une alternative efficace au Cloud computing. Dans ce mémoire, nous étudions un modèle d'un système IoT dans environnement de Fog computing en réalisant une simulation d'une stratégie de placement des services IoT sur un ensemble de nœuds Fog. Cette stratégie est mise en œuvre en proposant un algorithme de placement "Best Fog" qui utilise une approche basée sur plusieurs critères, tels que la capacité de mémoire et de calcul, pour sélectionner le nœud fog le plus adapté à une tâche spécifique. Les résultats de simulation obtenus sont jugés intéressants puisqu'ils révèlent une meilleure utilisation des ressources disponibles et une affectation plus efficace des tâches. Cette évaluation est également validée par une analyse comparative entre l'algorithme proposé ("Best Fog") et un autre algorithme de type FCFS.

Mots-Clés : Cloud computing, Fog computing, Internet des objets (IoT), FognetSim++.

Abstract

The IoT is a revolutionary technology offering significant benefits in operational efficiency, decision-making and customer satisfaction. However, the effective operation of the IoT requires a high-performance infrastructure, such as that of cloud computing, to manage data and services, as well as application requirements. Cloud computing, however, reveals certain limits related in particular to latency, bandwidth and operating costs. Fog computing is therefore proposed as a solution to this type of problem by offering a distributed computing infrastructure to process data and services close to users and connected objects, thus offering an effective alternative to Cloud computing. In this thesis, we study a model of an IoT system in a Fog computing environment by performing a simulation of IoT services placement strategy on a set of Fog nodes. This strategy is implemented by proposing a "Best Fog" placement algorithm that uses an approach based on several criteria, such as memory and computational capacity, to select the most suitable fog node for a specific task. The simulation results obtained are considered interesting since they reveal a better use of available resources and a more efficient assignment of tasks. This evaluation is also validated by a comparative analysis between the proposed algorithm ("Best Fog") and another FCFS algorithm.

Keywords: Cloud Computing, Fog Computing, Internet of Things (IoT), FognetSim++.

ملخص

انترنت الأشياء هي تقنية ثورية تقدم فوائد كبيرة في الكفاءة التشغيلية وصنع القرار ورضا العملاء. ومع ذلك، فإن التشغيل الفعال لإنترنت الأشياء يتطلب بنية تحتية عالية الأداء، مثل الحوسبة السحابية، لإدارة البيانات والخدمات، فضلاً عن متطلبات التطبيق. ومع ذلك، تكشف الحوسبة السحابية عن حدود معينة تتعلق على وجه الخصوص بزمن الانتقال وعرض النطاق الترددي وتكاليف التشغيل. لذلك، يُقترح استخدام حوسبة الضباب كحل لهذا النوع من المشكلات من خلال تقديم بنية أساسية للحوسبة الموزعة لمعالجة البيانات والخدمات القريبة من المستخدمين والأشياء المتصلة، وبالتالي تقديم بديل فعال للحوسبة السحابية. في هذه الأطروحة، ندرس نموذجًا لنظام إنترنت الأشياء في بيئة حوسبة الضباب عن طريق إجراء محاكاة لاستراتيجية وضع خدمات إنترنت الأشياء على مجموعة من عقد الضباب. يتم تنفيذ هذه الاستراتيجية من خلال اقتراح خوارزمية وضع "أفضل ضباب" التي تستخدم نهجًا يعتمد على عدة معايير، مثل الذاكرة والقدرة الحسابية، لاختيار عقدة الضباب الأكثر ملاءمة لمهمة معينة. تعتبر نتائج المحاكاة التي تم الحصول عليها مثيرة للاهتمام لأنها تكشف عن استخدام أفضل للموارد المتاحة وتعيين أكثر كفاءة للمهام. يتم التحقق من صحة هذا التقييم أيضًا من خلال تحليل مقارن بين الخوارزمية المقترحة ("أفضل ضباب") وخوارزمية اخرى وهي الترتيب الزمني الاولي.

الكلمات الرئيسية: حوسبة السحابة. حوسبة الضباب. انترنت الأشياء. فوجنتسيم++.