Université Abou Bekr Belkaid
Tlemcen Algérie
جامعة أبي بكر بلقايد
تلمسان الجزائر

RépubliqueAlgérienneDémocratiqueetPopulaire
MinistèredelʼEnseignementSupérieuretdelaRecherche Scientifique

UNIVERSITE de TLEMCEN
Faculté des Sciences

Département d'informatique

Mémoire de fin d'études

En vue de l'obtention du Diplôme de MASTER en Informatique
Option : Réseaux et Systèmes Distribués (RSD)

Thème

Utilisation des mécanismes d'ordonnancement pour la gestion des ressources dans un réseau à base de Fog Computing

Présenté par :

M^elle Ghofiri amina
Mr Guettaia oussama

Soutenu le 04 /07 /2023, devant le jury composé de:

Président	**Mr BENMAMMAR Badr**	Université de Tlemcen
Examinateur	**Mr Belhocine Amine**	Université de Tlemcen
Encadrant	**Mme AMRAOUI Asma**	Université de Tlemcen
Co-Encadrant	**Mme ARICHI Bochra**	Université de Tlemcen

Année universitaire : 2023-2024.

Remerciements

Nous tenons tout d'abord à remercier Allah tout puissant et miséricordieux, de nous avoir donné la volonté et la force pour terminer ce projet de fin d'études.

Nous tenons aussi à remercier notre encadreur **Dr. AMRAOUI Asma** pour ses précieux conseils, encouragements, sa disponibilité, et son aide tout au long de la réalisation de ce travail.

Nous remercions également notre Co-encadreur de mémoire **ARICHI Bouchra**, pour nous avoir suivis et conseillés au quotidien durant ces travaux de mémoire.

Nos remerciements aux membres de jury pour l'honneur qu'ils nous ont fait en acceptant d'examiner ce travail.

Nos remerciements à Nos très chers parents de nous avoir toujours donné le courage et d'être l'une des raisons qui nous pousse à nos battre face à chaque obstacle.

Nos remerciements à nos enseignants qui ont contribué à notre formation et au personnel de notre département.

Nous remercions enfin l'ensemble des personnes qui ont contribué d'une manière ou d'une autre à l'élaboration de ce travail.

Merci

Dédicaces

Ce modeste travail est dédié à :

Mes chers parents qui m'ont soutenu tout au long de mon parcours et dont tous les sacrifices justifient à eux seuls ma ténacité à achever ce projet et à avoir travaillé avec autant d'acharnement.

A tous les enseignants qui m'ont transmis des connaissances depuis ma première année de Licence jusqu'à la deuxième année de Master.

A mes sœurs et mes frères pour leur Amour sans lequel aucun projet n'eut pu aboutir.

A tous mes professeurs.

A tous mes amis et mes collègues sans exception.

A tous la promo RSD 2022/2023

Et à toute personne qui ne cesse de se battre pour réaliser ses buts.

Oussama

Dédicaces

Avant tous je remercie le dieu le tout puissant de m'avoir donné le courage et la volonté pour réaliser ce travail ; je dédié :

A mes très chers parents pour tous les sacrifices qu'ils ont faits et pour tout le soutien qu'ils ont offert tout au long de mes études.

A mes sœurs :Hadjer, Ikram, Assia et à mon frère Abdelghafour

A la lumiere de ma vie mes enfants :Kamer, Ghizlane, Ahmed .

A toute la famille et mes amis en particulier : Asma, Hanane, Rihem et Samah .

A tous la promo RSD 2022/2023

Et à toute personne qui ne cesse de se battre pour réaliser ses buts.

Amina

Table des Matières

Liste des figures

Liste des tableaux

Liste des acronymes

Acronyme	Signification
FC	**F**og**C**omputing
IOT	**I**nternet **O**f **T**hings
RFID	**R**adio **F**requency**I**dentification
MIT	**M**assachusetts **I**nstituteof**T**echnology
LTE-M	**L**ong **T**erm**E**volution for **M**achines
NB-IoT	**N**arrow**b**and**I**nternet **O**f**T**hings
API	**A**pplication**P**rogramming **I**nterface
LISP	**L**ocation / **I**dentifier**S**eparation **P**rotocol
TIC	**T**echnologiesdel'**i**nformationet de la **C**ommunication
OMS	**O**rganisation **M**ondiale de la**S**anté
QoS	**Q**uality **O**f **S**ervice
UR	**U**tilisation des **R**essources
ICT	**I**nformation and **C**ommunications**T**echnology
OC	**C**out **O**pérationnel
RMA	**R**ate**M**onotonic**A**nalysis
DMA	**D**eadline**M**onotonic**A**nalysis
EDF	**E**arliest **D**eadline **F**irst
LLF	**L**east**L**axity **F**irst
CO	**O**ptimisation **C**ombinatoire
POC	**P**roblèmes d'**O**ptimisation **C**ombinatoire
PL	**P**rogramme **L**inéaire
SA	**R**ecuit **S**imulé
AG	**A**lgorithme **G**énétique
TOPSIS	**T**echnique for **O**rder **P**erformance by **S**imilarity to **I**deal **S**olution
AHP	**A**nalyse **H**iérarchique des **P**rocédés

Liste des Annexes

Introduction générale

Contexte et problématique

Nous vivons à une époque où la technologie évolue à un rythme effréné, avec une succession constante d'innovations visant à optimiser et maximiser les ressources disponibles. L'Internet des objets (IoT) cherche aujourd'hui à connecter tous les objets et individus via Internet, ce qui génère une quantité considérable de données à traiter. Dans ce contexte, il est recommandé d'utiliser toutes les ressources du réseau, et pas seulement les objets, pour répondre à ce défi. Le cloud computing (CC) basé sur des réseaux de capteurs traite déjà un volume important de données, mais cela entraîne des temps de réponse lents pour les services. Afin de remédier à cela, un nouveau paradigme appelé Fog Computing (FC) a été conçu pour soutenir les applications IoT.

Dans ce cadre, nous nous soucions du secteur de la santé qui a toujours besoin de technologies pour détecter et traiter les urgences en temps réel pour se renseigner à l'avance sur les situations critiques.

Afin de traiter ce problème, nous employons le fog computing qui fonctionne comme un médiateur qui vise à optimiser la communication entre de nombreux objets connectés et des services distants. Compte tenu de la grande quantité de données (big data) générée par cette architecture, le FC réduit les variations de retard dans les réseaux distribués tout en offrant un meilleur contrôle des données transmises. Concrètement, au lieu d'envoyer directement les données depuis les appareils intelligents vers le cloud, le FC permet de gérer localement ces données, offrant ainsi de nombreux services tels que le calcul, la sauvegarde et le réseau à la périphérie des appareils IoT.

Contribution

Avec la popularité croissante du Fog Computing, la théorie de l'ordonnancement des tâches devient de plus en plus importante dans les systèmes de FC. En général, l'ordonnancement des tâches consiste à attribuer des tâches aux ressources disponibles en fonction de leurs caractéristiques et des conditions spécifiques.

Dans ce contexte, notre travail se concentre sur la planification des tâches dans le Fog Computing. Nous avons choisi d'utiliser une méthode d'optimisation multicritère appelée TOPSIS (Technique for Order Preference by Similarity to Ideal Solution). Nous allons appliquer une topologie de FC qui effectue un traitement client en tenant compte de : la latence, l'énergie et utilisation de réseau dans le but d'apercevoir la différence entre le FC et le CC.

Organisation du manuscrit

Notre mémoire est structuré autour de trois chapitres, les deux premiers sont théoriques et le dernier est pratique, qui se répartissent comme suit :

Dans le premier chapitre, nous allons présenter l'internet des objets (Iot) et le fog computing ainsi nous allons exposer la santé numérique et le fog computing dans le soin de santé.

Dans le deuxième chapitre, nous allons aborder l'ordonnancement des tâches et les principaux algorithmes d'ordonnancement ainsi Méthodes de résolution d'optimisation

Dans le troisième chapitre, nous exposerons dans la méthode d'optimisation multicritères TOPSIS, l'environnement de développement et de simulation ainsi que les résultats obtenus.

Chapitre I

Le Fog Computing et l'internet des objets

I.1. Introduction

Les infrastructures de l'informatique en brouillard, en anglais Fog Computing (FC) constituent un élément majeur de l'Internet. Grâce à leur capacité de stockage et de calcul, ainsi que leur emplacement près des utilisateurs finaux, elles permettent de traiter les données à proximité de l'endroit où elles sont générées. Bien que le domaine de l'Internet des Objets, ou Internet of Things (IoT), soit largement adopté et déjà bien connu, nous commencerons ce chapitre par expliciter l'IoT. Par la suite, nous détaillerons l'infrastructure du Fog Computing, telle que nous la connaissons aujourd'hui. A la fin, nous allons parler sur l'intégration du Fog Computing avec l'IoT, nous donnerons des exemples d'applications IoT qui utilise le FC et nous allons expliquer l'e-santé (ou la santé électronique).

I. 2. Internet des Objets

I.2.1. Histoire de l'IoT

Dans la deuxième moitié du XXe siècle, les éléments fondamentaux qui permettront le développement de l'Internet des Objets sont en place. Voici donc une chronologie des événements, relativement récents, qui vont précipiter la naissance de l'IoT et son expansion.

En 1982, des étudiants de l'Université Carnegie Mellon de Pittsburgh (Etats unis) inventent un distributeur de Coca-Cola relié à ARPANET (l'ancêtre d'Internet). À l'époque, l'expression « Internet des objets » n'est pas encore inventée : on parle alors d'Internet intégré ou d'informatique omni présente.

En 1990, lors d'une exposition, un certain John Romkey présente un grille-pain connecté à Internet. Ce grille-pain est considéré comme le premier appareil de l'histoire de l'IoT. Il était connecté à un ordinateur via un protocole TCP/IP.

En 1994, Steve Mann crée une caméra portable connectée à Internet. La WearCam était une expérience de connectivité, elle transférait via Internet les images capturées vers l'ordinateur. C'est une évolution de la caméra portable sur laquelle il a travaillé depuis le début des années 80. **[1]**

Ce n'est qu'en 1999 que Kevin Ashton, informaticien chez Procter &Gamble, utilise pour la première fois l'expression « Internet des objets ». Cette appellation avait pour but de susciter l'intérêt des décideurs sur la technologie de radio-identification (RFID) ainsi que les autres capteurs qu'il souhaitait mettre en place sur les produits de leur chaîne d'approvisionnement.

Cette même année, le professeur à l'Institut de technologie du Massachusetts (MIT) Neil Gershenfeld publiait *WhenThings Start to Think*. Dans ce livre, il y décrit clairement la direction prise par l'interconnexion des machines, sans toutefois utiliser l'expression d'Ashton.

Les gens commencent à penser à l'avenir de l'IoT**.** Les gourous de la technologie, les innovateurs, les ingénieurs et les influenceurs se mettent à faire des prédictions sur un avenir dans lequel l'Internet des Objets prendra une place importante.

Ces dernières années, l'IoT est devenu un réseau de capteurs composé de milliards d'objets intelligents, connectant des personnes, des systèmes informatiques et des applications dans le but de partager et de collecter des données. Où en 2017, Les géants de la Tech étendent leurs offres dédiées à l'IoT. Par exemple : **[1]**

- Microsoft lance Azure IoTEdge pour permettre l'utilisation à grande échelle de services cloud pour les appareils connectés, même lorsqu'ils ne sont pas connectés au cloud.
- Amazon fournit des fonctions avancées de sécurité et de gestion des périphériques.
- Google lance Cloud IoTCore, un service qui permet de connecter et gérer plus facilement des appareils connectés.

I.2.2 Définition

L'Internet des objets désigne à la fois le processus de connexion d'objets physiques à Internet et le réseau qui relie ces objets.

Par « objets », on entend aussi bien des appareils de la vie quotidienne tels que domotique, montre de fitness, appareils portables, appareils intelligents, des appareillages médicaux, des machines agricoles, des chaînes d'approvisionnement, des robots industriels ou des feux de circulation routière.

Au cœur de l'internet des objets se trouve la capacité de l'objet à s'interconnecter et à interagir avec son environnement physique. Il regroupe donc :

- Les objets connectés directement à internet
- Le machine to machine -M2M- c'est-à-dire la communication entre machines et l'accès au système d'information sans intervention humaine que ce soit Bluetooth, RFID, Wifi, 4G et bientôt la 5G…
- Les « smart connected devices » tels que les tablettes ou Smartphones

En définitive, l'IoT relie tout élément capable de transférer des données sur un réseau. Et ce, sans nécessiter d'interactions entre humains ou entre un humain et un ordinateur. Cependant, l'interaction personne-machine est rendue possible, ne serait-ce que pour procéder au paramétrage, à la configuration ou simplement pour accéder aux informations.**[2]**

I.2.3. Fonctionnement de l'IoT

L'IoT fonctionne principalement avec des capteurs et objets connectés placés dans/sur des infrastructures physiques. Ces capteurs vont alors émettre des données qui vont remonter à l'aide d'un réseau sans fil sur des plateformes IoT. Elles pourront ainsi être analysées et enrichies pour en tirer le meilleur profit.

Ces plateformes de data management et de data visualisation sont les nouvelles solutions IoT permettant aux territoires, entreprises ou même usagers d'analyser les données et d'en tirer des conclusions pour pouvoir adapter pratiques et comportements.**[3]**

Vous l'aurez compris, l'IoT est étroitement lié aux objets connectés car ils ont la faculté de capter une donnée et de l'envoyer, via le réseau Internet ou d'autres technologies. Les objets connectés interagissent avec leur environnement par le biais de capteurs : température, vitesse, humidité, vibration… Dans l'IoT, un objet peut aussi bien être un véhicule qu'une machine industrielle ou encore une place de parking.

Cette notion d'internet des objets peut être expliquée grâce au concept de « Service Oriented Architecture » décomposé en quatre couches différentes appelées layer. Nous vous l'expliquons à l'aide du tableau ci-dessous : **[3]**

SENSING	NETWORKING	SERVICE	INTERFACE
Les « hardwares » physiques comme les capteurs intelligents	Connecte les équipements entre eux	Les technologies du « middleware » qui permet de faire communiquer entre eux « hardware » et « software »	Les plateformes qui présentent les applications aux utilisateurs finaux

Tableau I.1: Table explicatif la décomposition des quatre couches

I.2.4. Architecture de l'IoT

L'architecture IoT peut en fait varier considérablement en fonction de la mise en œuvre, elle doit être suffisamment ouverte avec des protocoles ouverts pour pouvoir prendre en charge plusieurs applications réseau.

Même s'il n'existe pas d'architecture IoT unique universellement acceptée, le format le plus basique et le plus largement accepté est une architecture IoT à quatre couches.

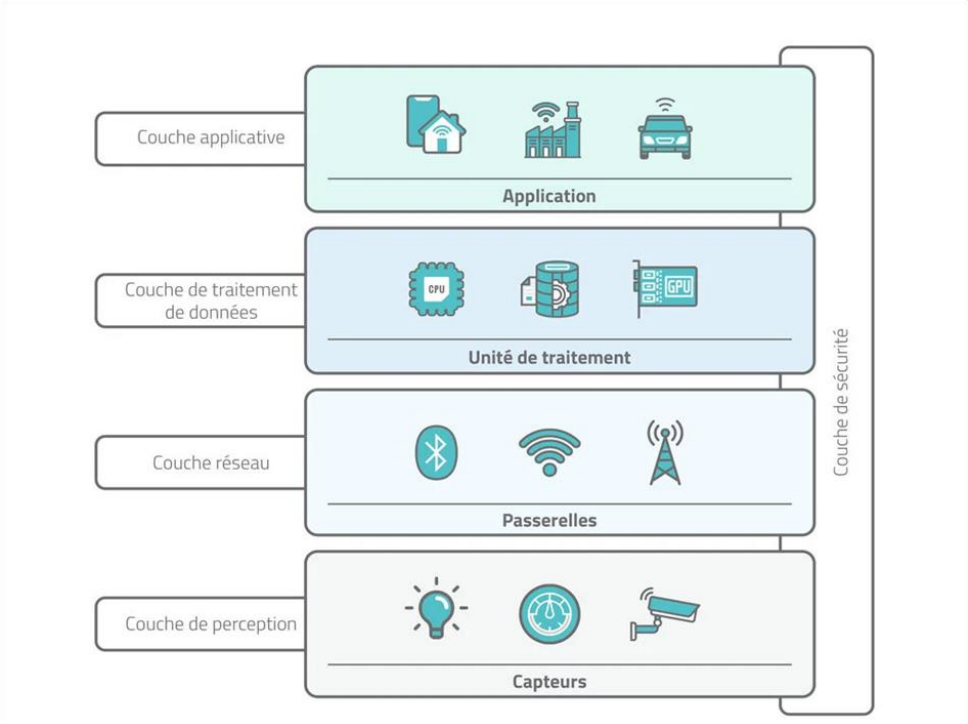


Figure I.1: les 4 couches de IoT. [4]

I.2.4.1. Couche de perception

La couche de perception joue un rôle essentiel dans la conversion des signaux analogiques en données numériques, et inversement. Elle existe sous différentes formes et tailles, allant des minuscules puces de silicium aux gros véhicules. Les objets de l'Internet des objets (IoT) peuvent être regroupés en fonction de leurs fonctions : **[5]**

- **Capteurs :** Ces capteurs, tels que les sondes, les jauges, les compteurs, et autres, recueillent des paramètres physiques tels que la température ou l'humidité. Ils les convertissent en signaux électriques et les transmettent au système IoT. Les capteurs IoT sont généralement petits et économes en énergie.
- **Actionneurs :** Ils traduisent les signaux électriques provenant du système IoT en actions physiques. Les actionneurs sont utilisés dans les contrôleurs de moteur, les lasers, les bras robotiques, et d'autres dispositifs.
- **Machines et dispositifs :** Ils peuvent être connectés à des capteurs et des actionneurs ou être des composants intégrés à l'ensemble du système.

I.2.4.2. Couche réseau

La couche réseau joue un rôle crucial dans la transmission et le traitement des données collectées par tous ces appareils. Elle est chargée de connecter ces appareils à d'autres objets intelligents, serveurs et appareils réseau, tout en gérant la transmission de toutes les données.

Les communications entre les appareils IoT et les services cloud ou les passerelles impliquent l'utilisation de différentes technologies : **[6]**

- Ethernet : Cette technologie permet de connecter des appareils IoT fixes tels que des caméras de sécurité, des équipements industriels installés en permanence, et d'autres dispositifs similaires.
- Réseaux cellulaires : Les technologies cellulaires les plus couramment utilisées sont la 5G et la 4G, offrant une transmission de données fiable et une couverture quasi mondiale. Il existe également deux normes cellulaires spécifiquement développées pour les objets IoT : LTE-M (Long Term Evolution for Machines), qui permet aux appareils de communiquer directement avec le cloud et d'échanger de grandes quantités de données, et NB-IoT (Narrow band IoT), qui utilise des canaux base fréquence pour l'envoi de petits paquets de données.
- LPWAN (Low-power Wide-area Network) : Cette technologie a été spécialement conçue pour les appareils IoT. Elle offre une connectivité longue portée avec une faible consommation d'énergie,

- WiFi : La technologie de réseau sans fil la plus populaire, le WiFi, convient parfaitement aux solutions IoT qui nécessitent une grande quantité de données et sont faciles à recharger. Elle est généralement utilisée dans des zones de petite taille.

Un bon exemple d'utilisation est celui des appareils domestiques intelligents connectés au réseau électrique.

Ces différentes technologies de communication permettent à la couche réseau de faciliter la transmission efficace des données dans l'écosystème de l'Internet des objets.

I.2.4.3. Couche de traitement de données

La couche de traitement de données est responsable de l'accumulation, du stockage et du traitement des données provenant de la couche précédente. Elle s'appuie généralement sur des plateformes IoT et comprend deux étapes principales.

- **Étape d'accumulation des données :**

Les données en temps réel sont capturées via une interface de programmation (API) et sont mises en attente pour répondre aux besoins des applications non temps réel. Cette étape de l'accumulation des données agit comme une plaque tournante entre la génération de données basée sur les événements et la consommation de données basée sur les requêtes. Elle détermine si les données sont pertinentes pour les besoins de l'entreprise et où elles doivent être stockées. Les données sont enregistrées dans une variété de solutions de stockage, notamment des data lakes capables de gérer des données non structurées telles que des images et des flux vidéo. L'objectif global est de trier et de stocker efficacement une grande quantité de données diverses. **[7]**

- **Étape d'abstraction des données :**

Dans cette étape, la préparation des données est finalisée afin que les applications grand public puissent les utiliser pour générer des informations. Le processus comprend les étapes suivantes :

  - Combinaison de données provenant de différentes sources, à la fois IoT et non-IoT.
  - Harmonisation de différents formats de données.
  - Agrégation des données en un seul emplacement rendant leur accès possible indépendamment de leur emplacement grâce à la virtualisation des données. De plus, les données collectées au niveau de la couche applicative sont reformatées ici pour être envoyées au niveau physique, afin que les appareils puissent les comprendre.

Ensemble, les étapes d'accumulation et d'abstraction des données masquent les détails matériels, ce qui améliore l'interopérabilité des appareils intelligents **[7].**

I.2.4.4. Couche Applicative :

La couche applicative est celle avec laquelle l'utilisateur interagit. Elle est chargée de fournir des services spécifiques de l'application à l'utilisateur. Actuellement, les applications peuvent être développées directement sur des plateformes IoT qui offrent une infrastructure logicielle de développement avec des outils prêts à l'emploi pour l'exploration de données, l'analyse avancée et la visualisation de données. Dans le cas contraire, les applications IoT utilisent des API pour s'intégrer à la couche précédente.

I.3. Fog Computing

L'internet des objets nécessite un calcul sensible à la latence pour le traitement des applications en temps réel. Dans les environnements IoT, les objets interconnectés produisent une énorme quantité de données. Les données générées par les dispositifs IoT sont généralement traitées dans une infrastructure cloud en raison des services à la demande et des caractéristiques d'évolutivité du paradigme de Cloud Computing. Cependant, l'intégration de l'IoT avec l'informatique en nuage (Cloud Computing) est confrontée à de nombreux défis. L'un de ces défis est la croissance des applications en temps réel et latentes et les limitations de bande passante du réseau.

Cela entraîne un fardeau de communication inutile pour le réseau central et le centre de données dans le nuage. Par conséquent, un nouveau paradigme dont le nom semble prometteur dans la littérature est le Fog Computing. Il est conçu pour prendre en charge les applications IoT caractérisés par des limitations de latence et des exigences de mobilité et de géo distribution.

I.3.1. Origine

Dès 2012, le concept de Fog Computingest apparu comme une généralisation de l'idée de Cloud Computingà la périphérie du réseau, notamment pour répondre aux exigences de performances liées à l'utilisation d'un grand nombre de capteurs. – Contexte de traitement du temps **[8]**.En novembre 2015, le Consortium Open Foga été formé, dirigé par ARM, Cisco, Intel, Microsoft et l'Université de Princeton (membres fondateurs) **[9]**. L'alliance, qui comptait 62 membres (industriels et universités) en avril 2018 **[10]**, vise à définir un cadre architectural et des standards ouverts pour le Fog Computing afin de faciliter l'interopérabilité de ses composants et la mise à l'échelle de capacités géographiquement

réparties. Le 31 janvier 2019, le consortium a fusionné avec l'Industriel Internet Consortium, qui a réuni Fog avec d'autres initiatives **[11].**

I.3.2. Définition

Le Fog Computing est un paradigme avec des capacités limitées telles que l'informatique, le stockage et les services de mise en réseau de manière distribuée entre les différents appareils finaux et le Cloud Computing classique. Il offre une bonne solution pour les applications IoT sensibles à la latence **[12].** Bien que le terme ait été inventé à l'origine par Cisco **[13],** le Fog Computing a été défini par de nombreux chercheurs et organisations à partir d'un certain nombre de perspectives différentes. Yi et al ont donné une définition générale de l'informatique par brouillard. Il est indiqué comme : « Fog Computing est une architecture informatique géographiquement distribuée avec un pool de ressources qui se compose d'un ou de plusieurs dispositifs hétérogènes connectés de façon omniprésente à la périphérie du réseau et non exclusivement soutenus par des services Cloud, fournir en collaboration des services de calcul, de stockage et de communication élastiques (et de nombreux autres nouveaux services et tâches) dans des environnements isolés à une grande échelle de clients à proximité » **[14].**

Perrera et al ont essayé de clarifier et d'élargir le concept du Fog Computing. « Le Fog Computing est un scénario dans lequel un nombre considérable de dispositifs omniprésents et décentralisés hétérogènes (sans fil et parfois autonomes) communiquent et coopèrent potentiellement entre eux et avec le réseau pour effectuer des tâches de stockage et de traitement sans

L'intervention de tiers. Ces tâches peuvent être destinées à prendre en charge des fonctions réseau de Base ou de nouveaux services et applications s'exécutant dans un environnement en bac à sable. Les utilisateurs qui louent une partie de leurs appareils pour héberger ces services sont incités à le faire. » **[15].**

L'informatique en brouillard est également définie par le consortium Open Fog comme « une architecture horizontale au niveau du système qui distribue des ressources et des services de calcul, de stockage, de contrôle et de mise en réseau partout dans le continuum, du nuage aux objets ». **[16]**

La figure 2 montre un modèle de base du Fog Computing qui joue un rôle de couche immédiate (couche de brouillard) entre la couche des objets et la couche de nuages.

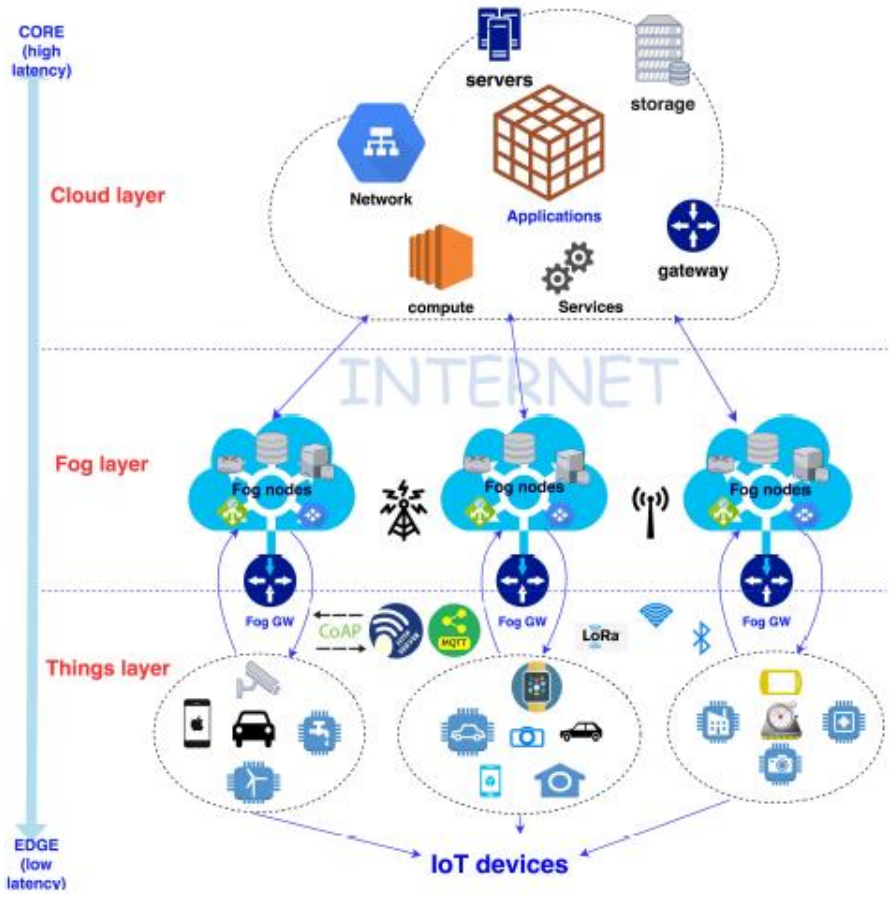


Figure I. 2: Modèle de Fog Computing.[17]

I.3.3. Caractéristiques du Fog Computing

Essentiellement, le Fog Computing est une extension du Cloud mais plus proche des choses qui fonctionnent avec les données IoT. Le Fog Computing agit comme un intermédiaire entre le Cloud et les périphériques finaux, ce qui rapproche les services de traitement, de stockage et de réseau des périphériques eux-mêmes. Ces dispositifs sont appelés nœuds de Fog. Ils peuvent être déployés n'importe où avec une connexion réseau. Tout appareil doté de l'informatique, du stockage et de la connectivité réseau peut être un nœud de Fog, comme les contrôleurs industriels, les commutateurs, les routeurs, les serveurs intégrés et les caméras de surveillance vidéo. **[18]**

Le Fog Computing est considérée comme l'élément constitutif du Cloud. Selon Ai et al **[19]**. Et Yi et al **[14],** les caractéristiques de Fog Computing peuvent se résumer comme suit :

- **Connaissance de l'emplacement et faible latence** : L'informatique de brouillard soutient la connaissance de l'emplacement dans lequel les nœuds de brouillard peuvent être déployés à différents endroits. En outre, comme le brouillard est plus

proche des dispositifs d'extrémité, il fournit une latence plus faible lors du traitement des données des dispositifs d'extrémité.

- **Distribution géographique** : Contrairement au cloud centralisé, les services et applications fournis par le brouillard sont distribués et peuvent être déployés n'importe où.
- **Évolutivité** : Il existe des réseaux de capteurs à grande échelle qui surveillent l'environnement environnant. Le brouillard fournit des ressources distribuées de calcul et de stockage qui peuvent fonctionner avec de tels dispositifs terminaux à grande échelle.
- **Soutien à la mobilité** : L'un des aspects importants des applications de brouillard est la capacité de se connecter directement aux appareils mobiles et donc de permettre des méthodes de mobilité, comme le protocole de séparation d'ID de localisation (LISP) qui a besoin d'un système de répertoire distribué.
- **Interactions en temps réel** : les applications de calcul du brouillard fournissent des interactions en temps réel entre les nœuds de brouillard plutôt que le traitement par lots utilisé dans le nuage.
- **Hétérogénéité** : Les nœuds de brouillard ou les dispositifs d'extrémité sont conçus par différents fabricants et se présentent donc sous différentes formes et doivent être déployés en fonction de leurs plates-formes. Le brouillard a la capacité de travailler sur différentes plates-formes.
- **Interopérabilité** : les composants de brouillard peuvent interagir et travailler avec différents domaines et différents fournisseurs de services.
- **Prise en charge de l'analyse en ligne et de l'interaction avec le nuage** : Le brouillard est placé entre le nuage et les dispositifs d'extrémité pour jouer un rôle important dans l'absorption et le traitement des données près des dispositifs d'extrémité.

I.3.4. Architecture du Fog Computing

Le Fog Computing est une approche qui apporte une partie des opérations d'un centre de données au bord du réseau. L'objectif principal du Fog Computing est de fournir une latence faible et prévisible pour les applications IoT sensibles au temps **[20].**

Selon Mukherjee et al **[21].**Aazam et Huh**[22]** et Muntjir et al **[23]**, l'architecture de Fog Computing contient 6 couches — physique et virtualisation, surveillance, prétraitement, stockage temporaire, sécurité et couche de transport — comme le montre la figure 3.

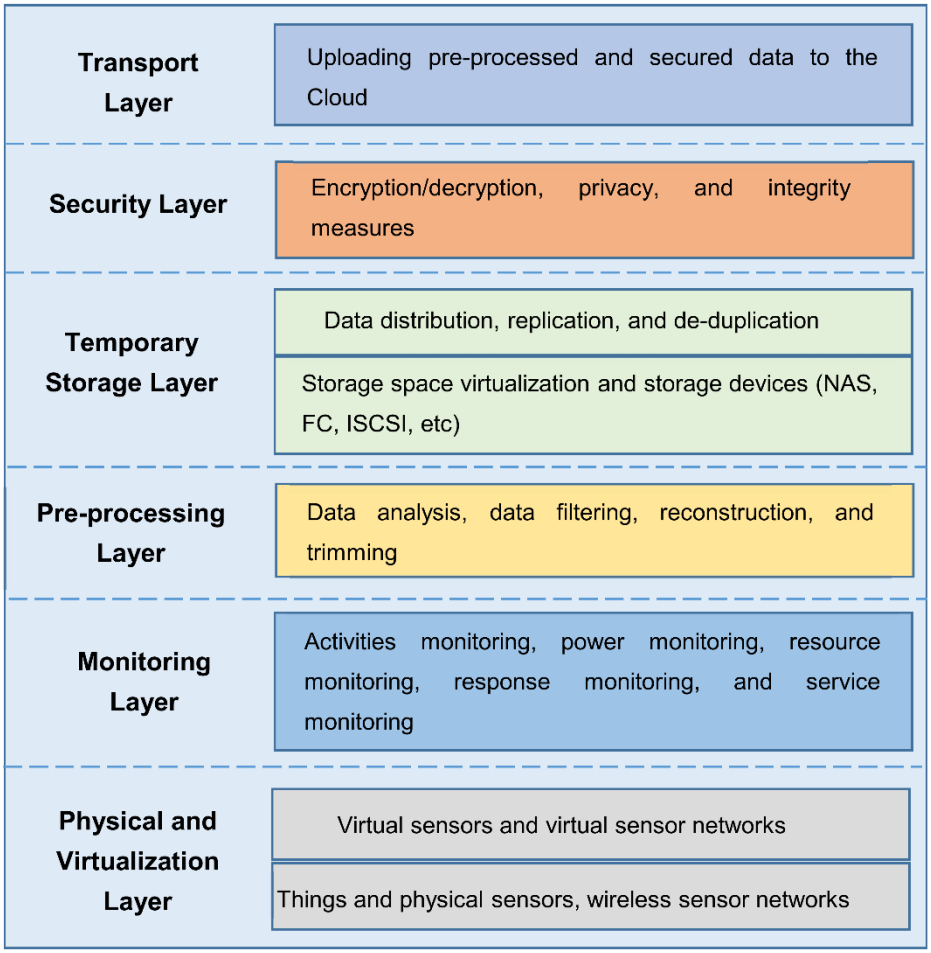


Figure I. 3: Architecture en couches de Fog Computing. [24]

La couche physique et de virtualisation englobe différents types de nœuds tels que les nœuds physiques, les nœuds virtuels et les réseaux de capteurs virtuels. Ces nœuds sont gérés et entretenus en fonction de leurs types et de leurs exigences de service **[25]**. **La couche de surveillance** surveille l'utilisation des ressources, la disponibilité des capteurs, des nœuds de Fog et des éléments du réseau. Elle permet de suivre les tâches effectuées par les nœuds, en identifiant quel nœud effectue quelle tâche, à quel moment et ce qui sera nécessaire par la suite **[21]**. **La couche de prétraitement** s'occupe de la gestion des données. Les données collectées sont analysées, filtrées et découpées afin d'extraire des informations significatives. Cette couche prépare les données pour un traitement ultérieur. Les données prétraitées sont ensuite stockées temporairement dans **la couche de stockage temporaire**. Lorsque les données sont transmises vers le cloud, elles ne nécessitent plus d'être stockées localement et

peuvent être supprimées du support de stockage temporaire **[23]**. **La couche de sécurité** joue un rôle essentiel dans le chiffrement/déchiffrement des données. Elle peut également appliquer des mesures d'intégrité pour protéger les données contre la falsification.

Enfin, dans **la couche de transport**, les données prétraitées sont téléchargées vers le cloud, permettant ainsi au cloud d'extraire et de créer des services plus utiles à partir de ces données **[23]**. Cette architecture du Fog Computing permet d'améliorer la latence et la performance des applications IoT en rapprochant une partie des opérations de traitement et de stockage des données du centre de données vers le bord du réseau.

I.4. Fog Computing avec IOT

Pour augmenter l'efficacité des applications IoT, la majorité des données générées par ces objets/dispositifs IoT doivent être traitées et analysées en temps réel **[26].** Le FogComputing amènera les capacités de réseau, d'informatique et de stockage à la périphérie du réseau, ce qui permettra de régler le problème en temps réel des dispositifs IoT et de fournir des applications IoT sûres et efficaces **[27].**

L'intégration du Fog Computing avec l'IoT apportera de nombreux avantages à diverses applications IoT. Le Fog permet des interactions en temps réel entre les appareils IoT pour réduire la latence, en particulier pour les applications IoT sensibles au facteur temps.

De plus, une caractéristique importante du Fog Computing est sa capacité à prendre en charge des réseaux de capteurs à grande échelle, ce qui est un gros problème pour le nombre croissant d'appareils IoT, qui atteindront bientôt des milliards. Le Fog Computing peut offrir de nombreux avantages pour diverses applications IoT, comme la montre la **Figure I.4**.

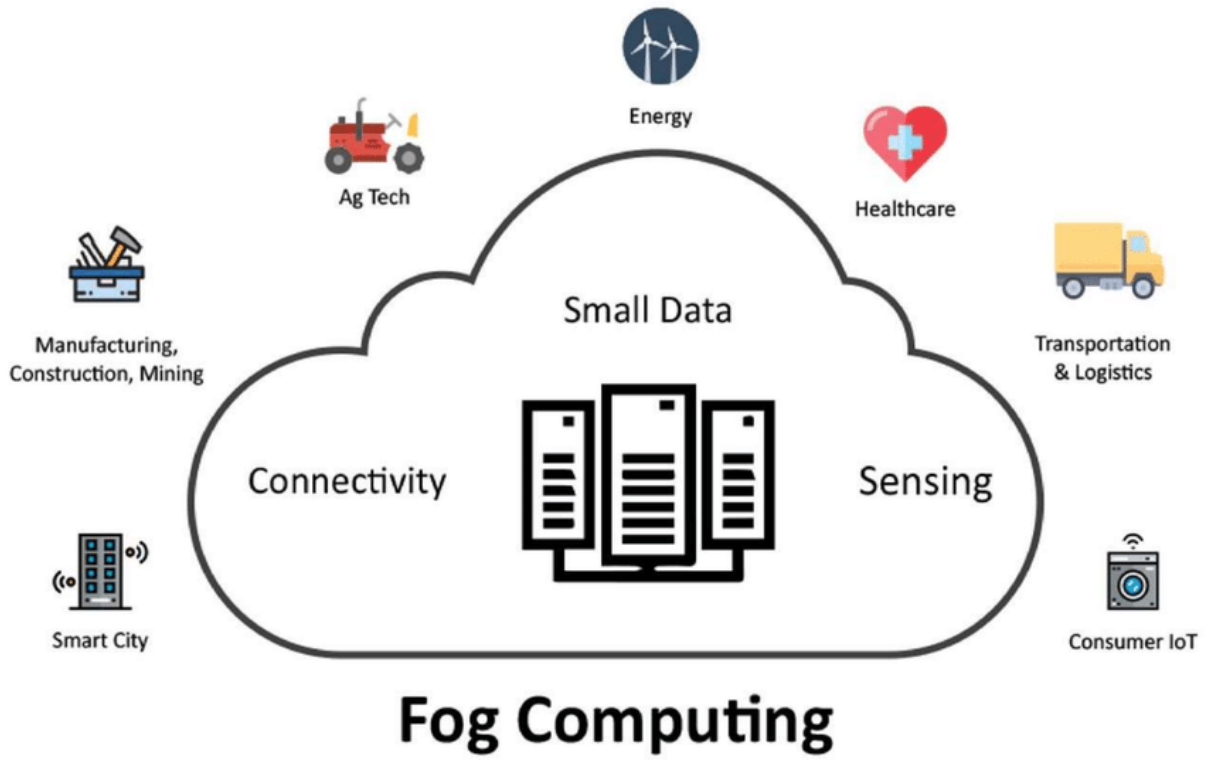


Figure I. 4: Le FC prend en charge de nombreuses applications IoT.[28]

I.5. Exemples d'applications du FC avec IoT

Il existe de nombreux domaines importants où le Fog Computing peut jouer un rôle vital dans différentes applications IoT. Cette section donne un aperçu des diverses applications IoT qui peuvent bénéficier de Fog Computing.

- **Feux de circulation intelligents**

  Le FC permet aux feux de circulation d'ouvrir les routes en fonction de la détection de feux clignotants. Il détecte la présence de piétons et de cyclistes et mesure la distance et la vitesse des véhicules à proximité. L'éclairage du capteur s'allume lorsqu'il identifie les mouvements et vice versa **[18].**

  Les feux de circulation intelligents peuvent être considérés comme des nœuds de brouillard qui sont synchronisés les uns avec les autres pour envoyer des messages d'avertissement aux véhicules à proximité. Les interactions du brouillard entre le véhicule et les points d'accès sont améliorées avec le WiFi, la 3G, les feux de circulation intelligents et les unités routières **[29].**

- **Maison intelligente**

  L'IoT dispose de nombreux capteurs et appareils connectés dans la maison. Cependant, ces appareils proviennent de différents fournisseurs et ont différentes

plateformes, ce qui rend difficile de les faire travailler ensemble. En outre, certaines tâches nécessitent une grande quantité de calcul et de stockage. Le FC résout beaucoup de ces problèmes. Il intègre toutes les différentes plates-formes et permet aux applications maison connectées de disposer de ressources flexibles **[14].**

Le FC a de nombreux avantages pour les applications de sécurité à domicile. Il fournit une interface unifiée pour intégrer tous les différents appareils indépendants. En outre, il fournit des ressources élastiques pour permettre le stockage, le traitement en temps réel et la faible latence **[14].**

- **Soins de santé et suivi des activités**

Le FC joue un rôle vital dans les soins de santé. Il fournit un traitement en temps réel et des réponses aux événements qui sont critiques dans les soins de santé **[30]**. En outre, l'interaction d'un grand nombre de dispositifs de santé pour le stockage à distance, le traitement et la récupération de dossiers médicaux à partir du nuage nécessite une connexion réseau fiable qui n'est pas disponible. Il aborde également les questions de connectivité et de trafic réseau **[31]**.

I.6. L'e-santé

Au cours des dernières années, l'industrie de la santé a constaté que les services Internet pouvaient contribuer à améliorer la qualité de vie du patient en offrant une analyse et un traitement des données en temps réel.

I.6.1. Défis pour les soins de santé

Dans la plupart des pays, les systèmes de soins de santé sont confrontés à d'énormes défis qui augmenteront en raison du vieillissement de la population et de l'augmentation des maladies chroniques. De nombreux pays connaissent également une pénurie croissante de personnel infirmier. En même temps, il y a une demande pour réduire les coûts tout en maintenant des soins de qualité aux patients. Par conséquent, l'industrie de la santé favorise un modèle de prestation de soins de santé axé sur l'information et basé sur la technologie d'IoT. Une partie de ce modèle de prestation permet la surveillance à distance des patients, ce qui permet d'accroître l'accessibilité, la qualité, l'efficacité et la continuité des soins pour les patients, et réduit également le coût global des soins de santé **[32]**.

I.6.2. Définition

L'e-santé (ou santé numérique) fait référence à « l'application des technologies de l'information et de la communication (TIC) à l'ensemble des activités en rapport avec la santé » **[33]**. Pour l'Organisation mondiale de la santé (OMS), l'e-santé est définie comme "les services numériques pour le bien-être de l'humanité". Elle se définit également comme « l'utilisation d'outils pour produire, transmettre, gérer et partager des informations numériques au profit de la médecine et de la pratique de la société médicale » **[34]**. Le terme e-santé regroupe ainsi un ensemble de techniques comme : la télé santé, la télémédecine, la m-santé (ensemble des objets connectés et des applications mobiles) et la robotique.

I.6.3. Internet des objets dans les soins de santé

Selon Carlos Jaime, la santé représente une formidable opportunité pour l'IoT. En obtenant une visibilité sur leurs différents flux et les activités, les hôpitaux peuvent optimiser leur gestion et améliorer les soins aux patients. IoT est également intéressant en termes de la traçabilité du circuit de médicaments en dehors de l'hôpital. Via des pilules connectées, ou un emballage médical connecté, les différents acteurs du cheminement des soins pourront savoir si un patient suit ou non le traitement prescrit **[35]**. L'Internet des objets peut résoudre un ensemble de défis dans le domaine de la santé comme la gestion des hôpitaux et des patients permanents surveillance.

- **Les avantages d'IoT dans le service de la santé**
  - améliorer la qualité de vie et la sécurité des patients grâce à une surveillance discrète et à des évaluations à distance.
  - permettre aux individus de gérer leur propre santé avec plus d'autonomie et de pro activité sur leur bien-être et leur pathologie.
  - améliorer la prévention grâce à une quantité incroyable de données et rendre le système de santé plus efficace tout en réduisant les coûts.
  - Optimisation des coûts des soins aux patients à l'hôpital et réduction des pénuries de stocks grâce à la surveillance à distance et à la gestion automatique du stock d'équipements.

- **Les inconvénients d'IoT dans le service de la santé**
  - les informations recueillies par ces "objets" sont exposées à des cybers attaques de pirates à la recherche de ces précieuses données de santé.
  - les dispositifs non sécurisés vulnérables à toutes sortes de logiciels malveillants ou d'attaques peuvent être exploités avec des risques de dommages physiques.
  - constituent autant de brèches dans le système de soins auquel elles sont intégrées et augmentent le risque de sécurité des dispositifs médicaux.
  - la propagation de dispositifs médicaux bas de gamme qui contreviennent aux règles de protection et de sécurité des fonctionnalités de base dont dépend la vie des utilisateurs.

I.6.4. Le Fog Computing dans les soins de santé

Dans le domaine de la santé, l'une des mesures les plus critiques pour gérer les situations critiques est le temps. Compte tenu de l'état de santé des patients, le diagnostic rapide d'anomalies dans les paramètres de santé peut améliorer la réponse des médecins à ce problème et, par conséquent, sauver des vies. L'une des principales caractéristiques du Fog Computing est la nouvelle couche entre les capteurs de santé et les centres de données du Cloud qui traitent les données. Cette nouvelle couche est conçue pour être physiquement proche des capteurs, et elle fournit également le traitement des données pour des réponses rapides. Du point de vue réseau, cette infrastructure plus proche des capteurs permet une communication de données sur le réseau avec des niveaux de latence plus faibles. Par conséquent, en insérant cette nouvelle fonctionnalité dans l'infrastructure d'application d'e-santé, il est possible de réduire la réponse en temps du système et de gagner du temps, ce qui, comme mentionné précédemment, est important pour la santé des patients. Par conséquent, la principale contribution du Fog Computing fournit pour les applications d'e-santé est cette infrastructure de réseau améliorée pour le transfert et le traitement rapides des données **[36]**.

I.7. Conclusion

La croissance de l'IoT a créé un grand réseau avec capteurs et actionneurs qui fournissent une réponse sensible à la latence aux utilisateurs finaux. Par conséquent, l'informatique traditionnelle comme le Cloud Computing a besoin de nombreux changements. En outre, ce grand nombre de dispositifs IoT géographiquement distribués requiert plusieurs

réponses en temps réel telles que des services sensibles à la localisation, un soutien à la mobilité, la fiabilité, faible latence, etc. Le Cloud ne peut donc pas fournir ces exigences. Le Fog Computing a été introduit pour répondre à un besoin de traitement avec une faible latence que les infrastructures de Cloud Computing ne sont pas capables de fournir, mais également pour une question de passage à l'échelle. De fait, le cloud ne pouvait faire face à l'explosion du nombre d'utilisateurs. Dans le chapitre 2, nous allons détailler les différents mécanismes d'ordonnancements des tâches sur un réseau basé sur le Fog Computing.

Chapitre II
Les mécanismes d'ordonnancement des tâches et d'optimisation

II.1. Introduction

Le Fog Computing est sans aucun doute le complément du Cloud Computing et aide à fournir une solution efficace pour traiter diverses applications IoT (Internet of Things). Cependant, fournir une solution dans un tel environnement est un défi dans différentes applications basées sur l'IoT telles que les applications de soins de santé, le système de transport intelligent et les villes intelligentes.

L'ordonnancement des tâches et l'affectation des ressources sont des problèmes NP dans l'informatique distribuée. Chaque application se compose de plusieurs modules qui nécessitent des ressources à exécuter. Cependant, fournir une politique optimale de planification des tâches dans un système aussi hétérogène est un problème de classe NP et a été proposé par différentes méthodes comme Greedy, méta-heuristique et tout algorithme inspiré de la nature pour résoudre un problème NP-complet. Le problème de planification des tâches est un défi clé dans les systèmes informatique distribués.

Dans ce chapitre, nous allons expliquer les différents concepts d'ordonnancement et d'optimisation et nous allons présenter les mécanismes d'ordonnancement des taches et les méthodes de résolution de problème et d'optimisation dans les environnements Fog-IoT.

II.2. Ordonnancement des tâches et optimisation

II.2.1 Ordonnancement des tâches

Le problème d'ordonnancement consiste à organiser dans le temps l'exécution d'un ensemble de tâches, compte tenu des contraintes temporelle tel que le délai et d'autres contraintes portant sur l'utilisation et la disponibilité des ressources requises tout en satisfaisant un ou plusieurs objectifs (coût, qualité, délai, énergie, etc…). Dans un problème d'ordonnancement, plusieurs notions fondamentales interviennent. Parmi ces notions on trouve : les tâches, les ressources, les contraintes et les objectifs **[37]**.

- Tâche

Une tâche ou un job est une entité élémentaire localisée dans le temps par une date de début et une date de fin, et dont la réalisation nécessite une durée préalablement définie. Elle est constituée d'un ensemble d'opérations qui requiert pour son exécution certaines ressources et qu'il est nécessaires de programmer de façon à optimiser certains objectifs.

- Ressource

La ressource est un moyen technique ou humain destiné à être utilisé pour la réalisation d'une tâche et elle est disponible en quantité limitée.

- Contraintes

Les contraintes représentent les limitations imposées par l'environnement ou les ressources, ou les utilisateurs.

Exemple : le budget, l'échéance.

- Objectif

Ce sont les critères à optimiser, c'est un ou plusieurs fonctions à minimiser ou à maximiser dans l'étude.

Exemple : temps total d'exécution, cout monétaire, l'énergie consommée.

II.2.1.1. Type d'ordonnancement

Il existe deux types d'ordonnancement de tâches qui sont : **[38]**

- Ordonnancement statique (hors-ligne)

L'ordonnancement statique (hors-ligne) consiste à déterminer une séquence fixe d'exécution des tâches avant le lancement de l'application, en prenant en compte tous les paramètres des tâches. Cette séquence est ensuite enregistrée dans une table et exécutée en ligne par un automate. Le nombre de tâches est connu à l'avance et le schéma d'ordonnancement est définitif. Chaque tâche possède une date d'exécution fixée et constante.

- Ordonnancement dynamique (en ligne)

L'ordonnancement dynamique (en ligne) consiste à établir la séquence d'exécution des tâches de manière dynamique pendant la durée de vie de l'application, en fonction des événements qui se produisent. L'ordonnanceur choisit la prochaine tâche à exécuter en fonction d'un critère de priorité. Le nombre de tâches peut varier au fil du temps, ce qui nécessite une réévaluation en ligne du schéma d'ordonnancement.

II.2.1.2 Objectifs et finalités d'un ordonnancement de tâches

Un ordonnancement est guidé par un ou plusieurs objectifs à atteindre. Les solutions trouvées sont comparées entre elles selon un ou plusieurs indicateurs chiffrés représentant le

but à atteindre. Voici une liste non exhaustive des catégories d'objectifs trouvées dans la littérature.

Qualité de service (QoS) : peut être représentée par plusieurs mesures en fonction de l'application, du contexte et des exigences de l'utilisateur ou du système. Les métriques de temps de réponse sont l'une des métriques les plus couramment utilisées pour mesurer la qualité de service des applications. D'autres critères tels que la disponibilité, la robustesse, l'évolutivité, etc. sont des critères fonctionnels de qualité de service.

Utilisation des ressources (UR) (efficacité) : détermine si certaines ressources sont sous-utilisées ou sur utilisées. Cela permet d'établir des politiques d'équilibrage de la charge de travail entre les appareils afin que l'ensemble de l'infrastructure soit rentable. Les métriques de cette catégorie sont généralement une comparaison de la charge du processeur de l'appareil à la charge moyenne de l'infrastructure ou aux tâches par seconde.

Énergie : Comme nous l'avons vu précédemment, l'énergie est un élément important des ICT. Les métriques associées à cet objectif sont généralement l'efficacité énergétique des équipements (énergie utile/énergie consommée), l'énergie totale et la puissance moyenne consommée.

Coût opérationnel (OC) : le coût de mise en place et d'utilisation d'une politique de planification peut être défini par des mesures de temps, des ressources ou même de l'utilisation d'énergie.

II.2.1.3 Résolution d'un problème d'ordonnancement

La résolution d'un problème d'ordonnancement consiste à déterminer :

- Le placement des tâches dans l'espace, c'est-à-dire sur les processeurs (la machines).
- Le placement des travaux dans le temps, c'est-à-dire les dates de début d'exécution de chaque tâche.
- La dépendance des tâches d'une application qui permet de définir le moment où une tâche peut être lancée sur une machine; La dépendance de tâches à un impact crucial pour la conception des algorithmes d'ordonnancement, qu'on peut classer en deux catégories:**[38]**

i. Ordonnancement des tâches indépendantes

Dans ce type d'ordonnancement, toutes les informations sont connues à l'avance comme le temps d'exécution sur les différentes ressources. Dans ce cas, les tâches s'exécutent

en parallèle, indépendamment les unes des autres. On distingue deux catégories de taches indépendantes :

- Taches indépendantes périodiques: Quatre ordonnancements de base
- Rate MonotonicAnalysis (RMA)
- Deadline MonotonicAnalysis (DMA) :
- Earliest Deadline First (EDF)
- Least Laxity First (LLF)
- Taches indépendantes apériodiques: trois approches parmi d'autres
- Traitement en arrière-plan
- Serveur de tâches
- Slackstealer

ii. Ordonnancement des tâches dépendantes

Il existe maintenant une contrainte de priorité qui exprime les relations de synchronisation et de communication entre les tâches ou lorsque les ressources sont partagées sous exclusion mutuelle (ressources critiques).

II.2.1.4 Rôle de l'ordonnanceur

Pour réduire les couts et garantir les délais de production, les entreprises utilisent un ordonnanceur. Appelé parfois automate ou job scheduler, cet outil devra trouver la machine la plus appropriée pour traiter les tâches qui lui sont soumises. Les ordonnanceurs peuvent aller du plus simple (allocation de type round-robin) au plus compliqué (ordonnancement à base de priorités). Ainsi Les ordonnanceurs déterminent le facteur de charge pour chaque ressource afin de bien ordonnancer la tâche suivante. Ils peuvent être organisés en hiérarchie avec certaines, interagissant directement avec les ressources, et d'autres (méta ordonnanceurs) interagissant avec les ordonnanceurs intermédiaires. Il peut également surveiller la progression de la tâche jusqu'à son achèvement, la soumettre à nouveau si elle est brusquement interrompue et se terminer plus tôt si elle est bloquée dans une boucle infinie d'exécution.

Le problème d'ordonnancement est un problème majeur dans la littérature qui a poussé les auteurs à proposer différents algorithmes **[39]**.

II.2.1.5 Principaux algorithmes d'ordonnancement des taches

Nous présentons dans ce qui suit, les principaux algorithmes d'ordonnancement cités dans la littérature :

- ✓ Algorithme Min-min

Il s'agit d'un algorithme de planification des taches statique utilisé pour l'équilibrage de charge. Algorithme Min-Min trouve d'abord le temps d'exécution minimum de toutes les taches. Ensuite, il choisit la tâche avec le moins de temps d'exécution parmi toutes les taches. Cet algorithme procède en affectant la tâche à la ressource qui produit le temps d'exécution minimum. La même procédure est répétée par Min-Min jusqu'à ce que toutes les tâches soient planifiées **[40]**.

- ✓ Algorithme Max min

L'algorithme Max-Min suit les mêmes principes que l'algorithme Min-Min, à l'exception des propriétés suivantes. Après avoir calculé le temps d'exécution minimal, la valeur maximale représentant la durée maximale de toutes les tâches sur la ressource est sélectionnée. Les tâches sont alors planifiées en fonction de ce temps maximum sur les machines correspondantes. Les temps d'exécution de toutes les autres tâches sur cette machine sont ensuite mis à jour en ajoutant le temps d'exécution de la tâche affectée aux temps d'exécution des autres tâches sur les machines qui ont repris la tâche sélectionnée. La tâche affectée est supprimée de la liste des tâches. Le même processus est répété jusqu'à ce que toutes les tâches soient affectées aux ressources **[40]**.

- ✓ Round robin (RR)

L'algorithme de scheduling du tourniquet, appelé aussi Round Robin est un algorithme d'ordonnancement courant dans les systèmes d'exploitation et convient bien aux systèmes fonctionnant en temps partagé. Des petites unités de temps sont définies appelées quantum de temps. La file d'attente est gérée comme une file d'attente circulaire. L'ordonnanceur analyse cette file d'attente, allouant du temps processeur à chaque processus à des intervalles de temps allant jusqu'à un quantum.

- ✓ First come first served (FCFS)

L'algorithme FCFS est tout simplement la politique FIFO —First in First out, Il s'agit de l'un des algorithmes les plus simples, une stratégie qui met en file d'attente chaque tâche et ressource et exécute chaque tâche et ressource dans l'ordre dans lequel elle arrive **[40]**.

- ✓ Earliest Deadline First scheduling (EDF)

Dans le même ordre d'idée, on peut aussi choisir d'exécuter en premier la tâche qui nécessite d'être fini le plus rapidement. Cet algorithme est utilisé pour les systèmes temps réel. C'est un ordonnancement préemptif avec priorité dynamique : la tâche la plus prioritaire est celle dont la date de fin est la plus proche, c'est à dire que plus le travail doit être réalisé rapidement, plus elle est prioritaire. Cependant, il est peu utilisé car il est très complexe à mettre en œuvre et se comporte mal lorsque le système est surchargé **[40]**.

II.2.2 Le problème d'optimisation

Un problème d'optimisation est défini comme la recherche de la valeur minimale ou maximale (optimale) d'une fonction donnée. Les définitions de problèmes d'optimisation sont souvent complétées par des données de contraintes. Les paramètres proposés (ou variables de décision) devraient en tenir compte limite **[41]**.

II.2.2.1 L'optimisation combinatoire

L'optimisation combinatoire (CO) joue un rôle très important en recherche opérationnelle et en informatique. De nombreuses applications peuvent être modélisées comme des problèmes d'optimisation combinatoire (POC), tels que les problèmes de voyageur de commerce, **la planification des tâches** et les problèmes de coloration des graphes. Un problème d'optimisation combinatoire consiste à trouver dans un ensemble discret de solutions, une ou plusieurs solutions parmi les meilleures solutions réalisables du problème. La résolution de (POC) consiste à trouver la meilleure solution, définie comme la solution globalement optimale ou un optimum global. En théorie, résoudre un problème nécessite de tester toutes les solutions possibles et de comparer leur qualité pour déterminer la meilleure solution. Cependant, en pratique, la liste de toutes les solutions peut prendre beaucoup de temps. La théorie de la complexité fournit des outils pour mesurer ce temps de recherche [**42]**.

II.2.2.2 Classe et complexité d'un problème

Tout problème d'optimisation peut être réduit à un problème de décidabilité. Un problème est dit décidable s'il existe un algorithme pour le résoudre en temps fini. Ce dernier est dit indécidable si aucun algorithme ne peut le résoudre en temps fini. La théorie de la complexité **[43]** permet d'estimer le temps et les ressources de calcul nécessaires pour résoudre un problème. Les problèmes dits décidables peuvent être divisés en trois classes de complexité temporelle:

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

II.3. Méthodes de résolution d’optimisation

II.3.1 Classification des méthodes de résolution

Face à un problème d’optimisation combinatoire donné il se pose un problème qui doit être résolu. Ce dernier a généralement un certain nombre de solutions réalisable. La solution la plus évidente consiste à lister toutes les combinaisons possibles et à trouver celle qui fonctionne le mieux.

Les méthodes de résolution d’un problème d’optimisation combinatoire peuvent être classées en deux grandes familles de classes : les méthodes exactes et les méthodes approchées.

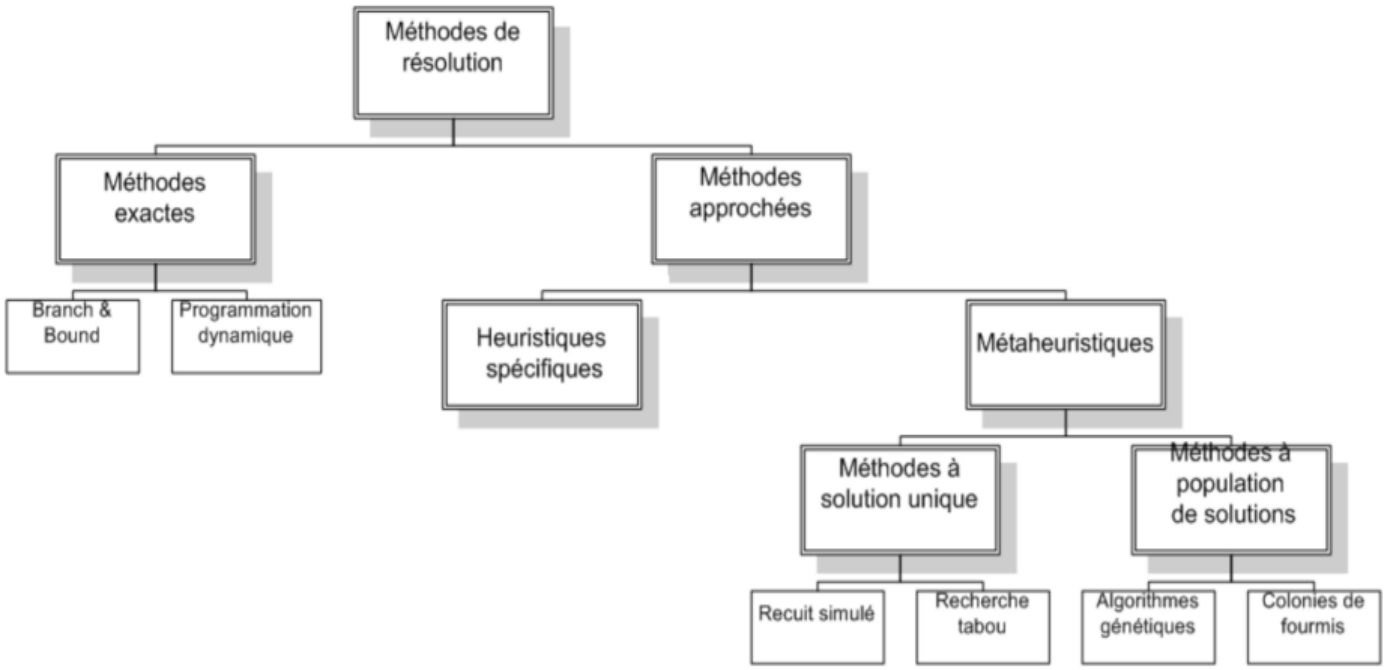


Figure II. 1: Classification des méthodes de résolution. [44]

II.3.2 Méthodes de résolutions exactes

Les méthodes exactes (appelées aussi complètes) produit la solution optimale suivante une instance de problème d'optimisation spécifique, ils sont généralement basés sur une recherche arborescente et sous-énumération de l'espace des solutions. Ces méthodes garantissent de trouver la solution optimale pour des instances de taille finie et de prouver son optimalité en un temps limité. Le principal inconvénient de ses méthodes est que le nombre de combinaisons augmente à mesure que la dimension du problème augmente. L'efficacité de ces algorithmes n'est prometteuse que pour les petites instances de problèmes. Les algorithmes exacts les plus connus sont la méthode de séparation et évaluation, la programmation dynamique, et la programmation linéaire **[45]**.

II.3.2.1 Méthode séparation et évaluation (Branch and Bound)

L'algorithme de séparation et évaluation est basé sur une méthode arborescente qui trouve la solution optimale par isolation et évaluation en représentant les états de la solution sous la forme d'un arbre d'état avec des nœuds et des feuilles. Le Branch and Bound repose sur trois axes principaux **[46]**.

L'évaluation :Cette évaluation peut réduire l'espace de recherche en excluant certains sous-ensembles qui ne contiennent pas la solution optimale. Le but est d'essayer d'évaluer l'intérêt d'explorer un sous-ensemble de l'arbre.

La séparation : elle consiste à diviser un problème en sous-problèmes. Son principe doit satisfaire aux trois règles, le nombre total de nœuds engendrés doit être fini, en conservant la meilleure solution trouvée et la règle d'arrêt.

La stratégie de parcours: Il est évident d'examiner la totalité des sommets de l'arborescence pour réaliser une énumération implicite efficace. Pour savoir quel sommet doit-on séparer on utilise des stratégies, on peut distinguer trois d'entre elles La largeur d'abord, La profondeur d'abord, Le meilleur d'abord.

II.3.2.2 Méthode de coupes planes (Cutting-Plane)

La méthode de coupes planes (cutting-plane) a été développée par (Schrijver 1986), elle est destinée à résoudre des problèmes d'optimisation combinatoire (POC) qui se formulent sous la forme d'un programme linéaire (PL). Si le POC est de grande taille représentée explicitement en mémoire, ou adapté à un solveur de programmation linéaire, on utilise une technique de résolution de problèmes relâchés (POCR) qui supprime certaines de ces contraintes. La solution optimale pour (PL) est incluse dans l'ensemble des solutions réalisables pour cette relaxation.

La méthode de coupe plane n'est pas très efficace, mais lorsqu'elle est combinée avec la méthode "Branch and Bound", elle améliore les performances **[47]**.

II.3.2.3 Programmation dynamique

La programmation dynamique est une méthode algorithmique pour résoudre des problèmes d'optimisation Le terme a été mentionné pour la première fois par le professeur Richard Bellman dans les années 1950.La programmation dynamique décompose le problème en sous problèmes plus faciles à résoudre puis combine les solutions. Il s'agit d'une méthode d'énumération implicite, on conserve ou on rejette des sous-ensembles de solution, mais on ne construit pas toutes les solutions. On rejette les solutions spécifiques sans configuration explicite si elles appartiennent à un sous-ensemble non intéressant **[45]**.

II.3.3 Méthodes de résolutions approchées

Contrairement aux méthodes exactes, les méthodes approchées permettent de trouver de manière rapide une solution réalisable à un problème donné. Cependant, cette solution ne fournit pas forcément une solution optimale, mais seulement une bonne solution dans un temps raisonnable.

II.3.3.1 Approche heuristique

En optimisation combinatoire, une heuristique est un algorithme approché qui permet de trouver la solution la plus proche possible de celle d'une méthode exacte tout en étant plus rapide, pas nécessairement optimale ou exacte pour un problème d'optimisation difficile. Donc Leur but est d'atteindre un optimum global tout en échappant les optima locaux.Les heuristiques peuvent être classées en deux catégories: **[48]**

- **Méthodes constructives :** l'idée de base des méthodes constructives est de produire une solution. On commence par une solution nulle et à chaque itération, on ajoute un élément (sommet, arc, job, tâche, etc.) jusqu'à l'obtention d'une solution complète.
- **Méthodes de fouilles locales :** ces méthodes utilisent une population de solutions à chaque itération jusqu'à l'obtention de la solution globale.

II.3.3.2 Approche méta-heuristique

Une méta-heuristique décrit un algorithme qui résout un problème d'optimisation en un temps de calcul acceptable sans aucune garantie d'optimalité. Les approches méta-heuristiques utilisent des stratégies générales indépendantes du problème étudié et guident la recherche dans l'espace des solutions. Ils sont donc applicables à tout problème d'optimisation.

Les méta-heuristiques sont des méthodes généralement inspirées de la nature, ces algorithmes sont plus complets et complexes qu'une simple heuristique, elles s'appliquent à plusieurs problèmes de nature différentes. Les méta-heuristiques sont en général non déterministes et ne donnent aucune garantie d'optimalité.

Les méta-heuristiques regroupent des méthodes qui peuvent se diviser en deux catégories suivant le nombre de solutions générées :

Des méta-heuristiques à solution unique et des méta-heuristiques à population de solutions **[49]**.

i. **Méta-heuristiques à solution unique**

Les méta-heuristiques à solution unique sont celles qui ne manipulent qu'une solution à la fois, ces méthodes ont plusieurs noms : elles peuvent être appelées méthodes de recherche locale ou bien méthodes de trajectoire.

Les méta-heuristiques sa solution unique commence avec une seule solution initiale et s'en éloignent progressivement, en construisant une trajectoire dans l'espace de recherche.

Nous présentons les méthodes les plus connues qui sont : la méthode de descente, le recuit simulé **[50]**.

- **Méthode de descente**

Appelée aussi l'amélioration itérative ou même le HillClimbing est probablement la méthode méta-heuristique la plus ancienne et la plus simple. L'algorithme commence par sélectionner aléatoirement une solution, puis à chaque itération la meilleure solution dans le voisinage de la solution courante est sélectionnée. La recherche arrête lorsque tous les voisins candidats sont pires que la solution actuelle, ce qui signifie qu'un optimum local est atteint. Donc le principe de la méthode consiste à partir de la solution s et à choisir une solution s'améliore la recherche (généralement telle que f(s') <f(s)).

On peut décider soit d'examiner toutes les solutions du voisinage et prendre la meilleure de toutes, soit d'examiner un sous-ensemble du voisinage. Le principal inconvénient de la descente est qu'elle s'arrête au premier minimum local rencontré. L'algorithme peut être démarré plusieurs fois à partir de nombreuses solutions initiales différentes pour améliorer les résultats, mais les performances de cette technique se dégradent rapidement.

- **Recuit Simulé (simulatedannealing)**

Le recuit simulé est basé sur les principes de la mécanique statistique, où le processus de recuit chauffe un matériau puis le refroidit lentement, ce qui donne une structure cristalline solide. Cet algorithme simule le changement d'énergie d'un système à travers un processus de refroidissement jusqu'à ce que le système converge vers un état stable. Le recuit simulé (SA) a été introduit comme méthode de recherche locale régulière avec une stratégie d'évitement minimum local **[47]**.

ii. Méta-heuristiques à population de solutions

Contrairement aux algorithmes qui commencent par des solutions singulières, les méta-heuristiques qui utilisent des populations de solutions améliorent la population de solutions au cours des itérations. Dans cette catégorie on distingue les algorithmes évolutionnaires qui représentent une famille d'algorithmes issus de la théorie de l'évolution par sélection naturelle et les algorithmes d'intelligence en essaim qui, de la même manière que les algorithmes évolutionnaires. Le principe de ces algorithmes est d'utiliser un ensemble de solution au lieu d'une seule. L'utilisation des procédures de sélection, génération et évaluation des individus, et change la population à chaque itération.

Nous présentons les méthodes les plus connues qui sont: les algorithmes génétiques, les algorithmes de colonies de fourmis et l'optimisation par Essaim de Particules **[51]**.

- **Algorithmes génétiques**

Un algorithme génétique (AG) est un algorithme d'optimisation stochastique basé sur les mécanismes de la sélection naturelle et de la génétique.

Leur fonctionnement est extrêmement simple, on part d'un ensemble de solutions potentielles (chromosomes) et on choisit arbitrairement. Évaluer leur performance relative (fitness). Sur la base de ces propriétés, une nouvelle population de solutions potentielles a été créée à l'aide d'opérateurs évolutifs simples : sélection, croisement et mutation. Certains individus se reproduisent, d'autres disparaissent et seuls les plus aptes survivent. Nous répétons ce cycle jusqu'à ce qu'une solution satisfaisante soit trouvée. En effet, l'hérédité de génération en génération permet à la population de s'adapter pour répondre à des critères d'optimisation **[47]**.

L'algorithme génétique est résumé par le pseudo-code suivant :

```
Début
    T=0
    Initialiser P(t)
    Evaluer la population P(t)
    Tant que condition de terminaison non satisfaite faire
    Début
        T = t+1
        Sélectionner P(t) à partir P(t-1)
        Mutations
        Croisements
        Evaluations
    Fin
Fin
```

Figure II.2: pseudo-code de l'algorithme. [52]

- **Algorithmes de colonies de fourmis (AntColonyoptimization)**

L'algorithme de AntColony est né d'un constat simple. Les insectes sociaux, en particulier les fourmis, résolvent naturellement des problèmes complexes. Un tel comportement est possible car les fourmis communiquent indirectement en déposant sur le sol des substances chimiques appelées phéromones. Ce type de communication indirecte est connu sous le nom de stigmatisation. Cet algorithme est de population itérative, dans lesquels tous les individus partagent des connaissances communes qui peuvent guider les choix futurs et indiquer aux autres individus ce qu'il faut suivre ou éviter. La principale explication de ce constat est présentée dans la figure ci-dessous.

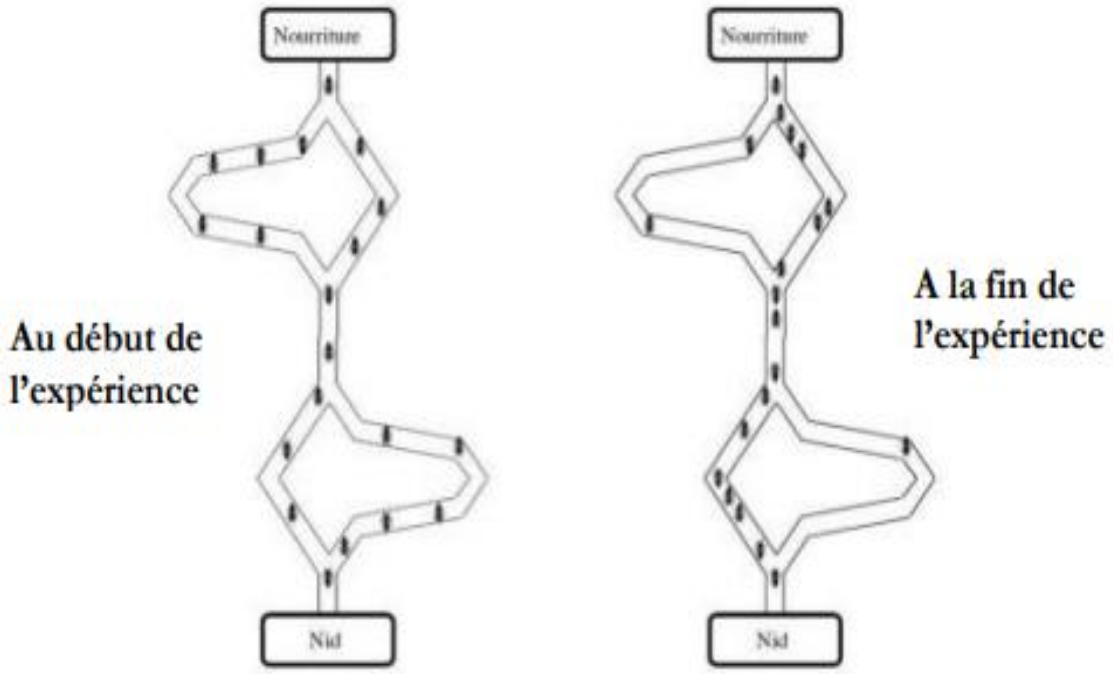


Figure II.3: L'expérience du pont à double branche. [53]

On voit sur cette figure que Les fourmis se dispersent pour trouver de la nourriture (elles n'ont qu'une vision locale de leur environnement), et après l'avoir trouvée, retournent au nid en laissant derrière elles des phéromones qui marquent le passage. Les fourmis attirées diffusent à leur tour des phéromones, renforçant les voies entre le nid et la zone de nourriture. Cependant, grâce au phénomène d'évaporation de la traînée de phéromones (si le chemin n'est pas utilisé ou peu de phéromones tend à disparaître), le chemin très balisé qui est le chemin le plus court par rapport aux autres directions assez rapidement disparaître **[45]**.

- **Optimisation par Essaim de Particules**

L'optimisation par Essaim de Particules est une approche stochastique basée sur la population pour résoudre les problèmes d'optimisation continue et discrète. Dans l'optimisation par essaim de particules, des agents logiciels simples, appelés particules, se déplacent dans l'espace de recherche d'un problème d'optimisation. La position d'une particule représente une solution possible au problème d'optimisation qui se pose. Chaque particule recherche de meilleures positions dans l'espace de recherche en changeant sa vitesse selon des règles inspirées à l'origine de modèles comportementaux de flocage d'oiseaux **[54]**.

Cet algorithme est inspiré par l'observation des relations grégaires d'oiseaux migrateurs, qui pour parcourir de « longues distances » (migration, quête de nourriture, etc.), doivent optimiser leurs déplacements en termes d'énergie dépensée, de temps.

L'optimisation par essaim de particules appartient à la classe des techniques **d'intelligence des essaims** utilisées pour résoudre les problèmes d'optimisation.

Figure II.4: Essaim de particules [55]

II.4. Conclusion

Dans ce chapitre, nous avons vu une définition du concept d'ordonnancement des tâches, élaborer les problèmes d'ordonnancement, étudier le problème d'optimisation combinatoire (POC) et nous avons cité les différents algorithmes dédiés à la résolution de ce dernier. Dans le prochain chapitre, nous allons présenter notre système distribué qui dépend sur l'infrastructure Fog en impliquant un des mécanismes d'ordonnancement des tâches.

Chapitre III
Simulation et évaluation des résultats obtenus

III.1. Introduction

Cette partie constitue le dernier volet de ce mémoire. Après avoir présenté l'IOT et l'intégration du Fog Computing avec l'IOT, ainsi expliquer le problème d'ordonnancement de tâches et d'optimisation et élaborer les méthodes de résolutions, nous allons dans ce chapitre définir la méthode d'optimisation multicritère TOPSIS, ses principes et donner un exemple qui l'explique, ensuite on va parler sur l'environnement de développement et de simulation qui inclus le langage Java, Eclipse et le simulateur IFogSim, a la fin nous allons présenté notre topologie de réseau, notre simulation et les résultats obtenus de simulation.

III.2. Méthode d'optimisation multicritères TOPSIS

III.2.1. Définition

« **Technique for Order Performance by Similarity to Ideal Solution** » (TOPSIS) est une des méthodes les plus utilisée d'aide multicritère à la décision pour le classement et la sélection parmi un certain nombre d'alternatives via la distance euclidienne, elle a été initialement développée par Ching-Lai Hwang et Yoon en 1981 **[56]** avec des développements ultérieurs par Yoon en 1987 **[57]**.

Le but des méthodes d'aide à la décision n'est pas seulement de faire du profit autant que possible, mais aussi d'éviter autant que possible les risques. Elle est basée sur le choix d'une solution qui se rapproche le plus de la solution idéale (meilleure sur tous les critères) et s'éloigne le plus de la pire solution (qui dégrade tous les critères).

III.2.2. Principe de TOPSIS

- **Solutions Idéale et Anti-Idéale**

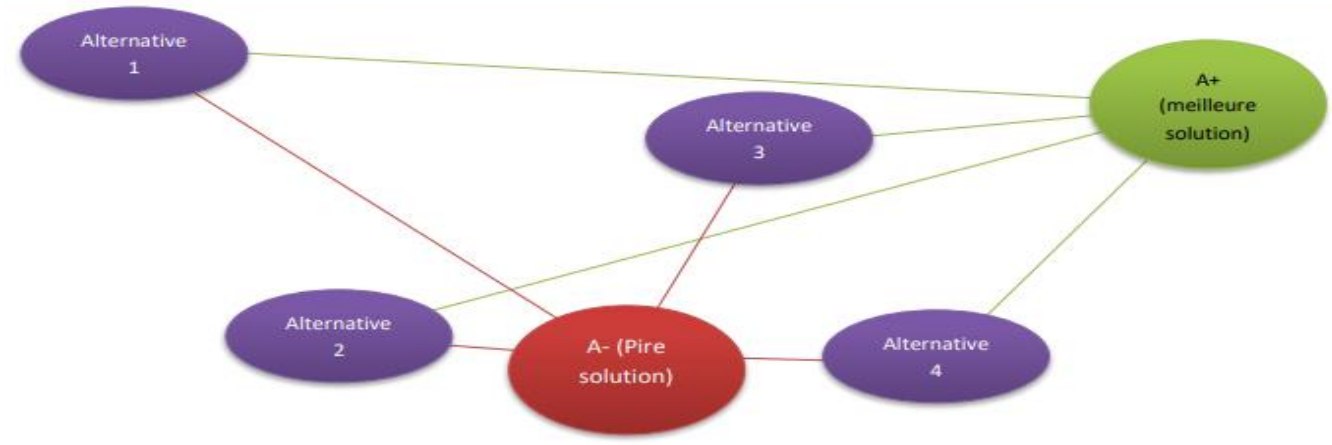


Figure III.1 : schéma d'illustration pour l'idéal et anti-idéal.

Le principe de TOPSIS est basé sur la distance des alternatives par rapport à l'idéal positif et l'idéal négatif. La meilleure alternative est celle qui est la plus lointaine de la solution A- (dite pire solution ou solution idéale négative) et la plus proche de la solution A+ (dite meilleure solution ou solution idéale positive). Les solutions (meilleure et pire) ne sont pas des alternatives, mais uniquement deux repères qui nous permettent de déterminer la meilleure alternative.

- **Matrice de décision**

Elle se compose de m alternatives (solutions) et de n attributs (critères). Soit xij le poids de l'option i par rapport au critère j.

$$D = \begin{array}{c|cccccc} & \boldsymbol{x1} & \boldsymbol{x2} & \boldsymbol{x3} & \cdots & \boldsymbol{xn} \\ \boldsymbol{A1} & \boldsymbol{x11} & \boldsymbol{x12} & \boldsymbol{x13} & \cdots & \boldsymbol{x1n} \\ \boldsymbol{A2} & \boldsymbol{x21} & \boldsymbol{x22} & \boldsymbol{x23} & \cdots & \boldsymbol{x2n} \\ \boldsymbol{A3} & \boldsymbol{x31} & \boldsymbol{x32} & \boldsymbol{x33} & \cdots & \boldsymbol{x3n} \\ \vdots & \vdots & \vdots & \vdots & \ddots & \vdots \\ \boldsymbol{Am} & \boldsymbol{xm1} & \boldsymbol{xm2} & \boldsymbol{xm3} & \cdots & \boldsymbol{xmn} \end{array}$$

III.2.3. Algorithme TOPSIS

Apres la construction de la matrice des scores attribuent à chaque critère relativement à chaque alternative, dans qui suit je vous présente les grandes étapes de cette méthode :

a. Calculer la matrice de décision normalisée : on calcul les préférences normalisées pour obtenir une nouvelle matrice R d'élément **rij** tel que

$$rij = \frac{xij}{\sqrt{\sum_{j=1}^{m} xij^2}} \quad i = 1\ldots\ m.\ j = 1\ldots\ n$$

b. Calculer la matrice normalisée pondérée : Dans cette étape, on multiplie simplement toutes les entrées (nij) de la matrice normalisée par la pondération associée à chaque critère, avec $\sum_{i=1}^{n} \boldsymbol{wi}$

$vij = wi * rij,\ j = 1 \ldots\ldots m,\ i = 1 \ldots\ldots$

c. Définir les solutions idéales et anti-idéales :

$V^+ = \{V_1^+, \cdots, V_n^+\} = \{(max_j * V_{ij} \mid i \in I), (min_j * V_{ij} \mid i \in I),\}$

$V^- = \{V_1^-, \cdots, V_n^-\} = \{(max_j * V_{ij} \mid i \in I), (min_j * V_{ij} \mid i \in I),\}$

d. Calcul les distances de séparation : dans cette partie on va calculer pour chaque alternative, la distance euclidienne entre l'idéal positif et l'idéal négatif, notées d_j^+ et d_j^- respectivement :

$$d_j^+ = \sqrt{\sum_{i=1}^{n} (V_{ij} - V_i^+)^2}$$

$$d_j^- = \sqrt{\sum_{i=1}^{n} (V_{ij} - V_i^-)^2}$$

e. Calculer la proximité relative de la solution idéale : on calcule le degré de proximité avec le positif idéal. Plus est important, plus l'alternative est proche de l'idéal positif et loin de l'idéal négatif :

$$D_j = \frac{d_j^-}{d_j^- + d_j^+}, \quad j = 1, \ldots\ldots m$$

Ou $d_j^- \geq 0$ et $d_j^+ \geq 0, D_j \in [0,1]$.

III.3. Environnement de développement et de simulation

III.3.1. Java

Java est à la fois un langage de programmation informatique orienté objet et un environnement d'exécution informatique portable créé par James Gosling et Patrick Naughton employés de Sun Microsystems avec le soutien de Bill Joy (cofondateur de Sun Microsystems en 1982), présenté officiellement le 23 mai 1995 au SunWorld**[58].**

Parmi les caractéristiques du langage java, il y a :

- **Interprété :** La source est compilé en pseudo code puis exécuté par un interpréteur Java (JVM).
- **Portable :** Il est indépendant de toute plate-forme.
- **Orienté objet :** Chaque fichier source contient la définition d'une ou plusieurs classes qui sont utilisées les unes avec les autres pour former une application.
- **Simple :** Le choix de ses auteurs a été d'abandonner des éléments mal compris ou mal exploités des autres langages.

III.3.2. Eclipse

Eclipse est un environnement de développement intégré libre dont le but est de fournir une plate-forme modulaire pour permettre de réaliser des développements informatiques.

I.B.M. est à l'origine du développement d'Eclipse, il permettant potentiellement de créer des projets de développement mettant en œuvre n'importe quel langage de programmation. Eclipse IDE est principalement écrit en Java (à l'aide de la bibliothèque graphique SWT, d'IBM), et ce langage, grâce à des bibliothèques spécifiques, est également utilisé pour écrire des extensions **[59]**.

Eclipse possède de nombreux points forts qui sont à l'origine de son énorme succès dont les principaux sont :

- Une plate-forme ouverte pour le développement d'applications et extensible grâce à un mécanisme de plug-ins
- Support de plusieurs plates-formes d'exécution : Windows, Linux, Mac OS X, ...
- Possibilité d'utiliser des outils open source : CVS, Ant, Junit

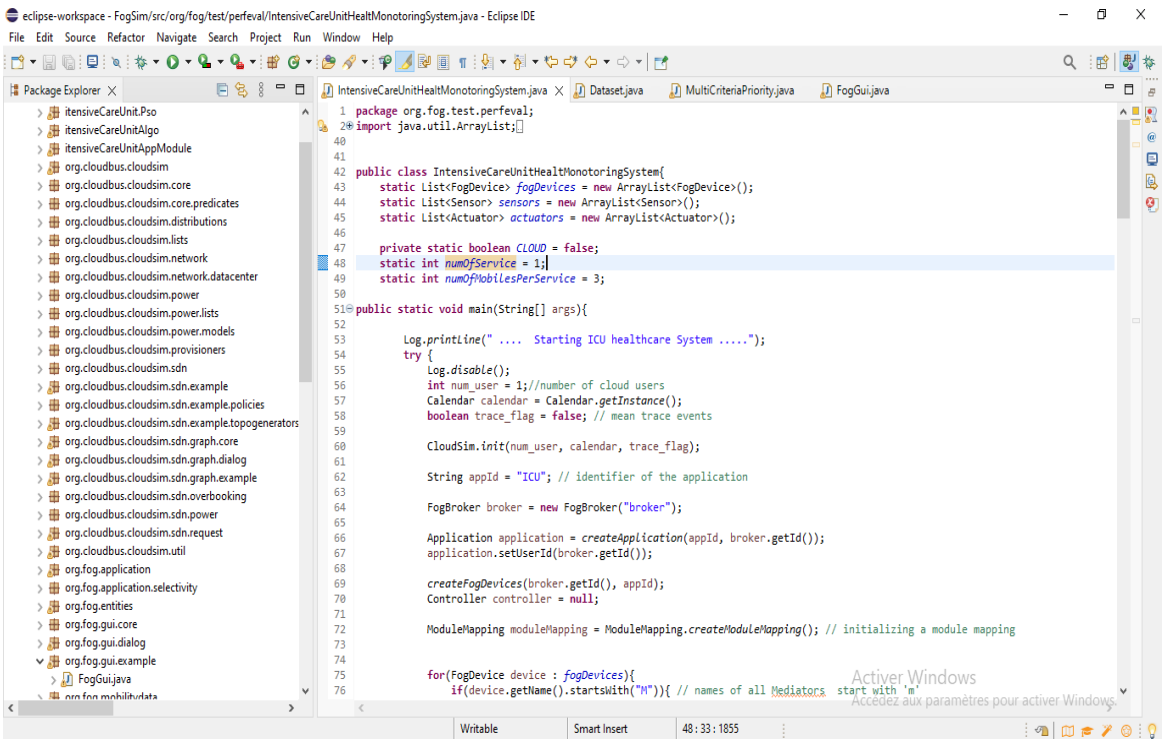


Figure III.2: L'environnement de développement Eclipse.

III.3.3. IFogSim

III.3.3.1. Définition

IFogSim est une boîte à outils open source hautes performances pour le Fog Computing et l'IoT, il s'agit d'une extension du simulateur d'environnements de Cloud : CloudSim **[60]**. Il a été développé pour évaluer les stratégies de gestion des ressources dans les environnements de Fog et IoT en termes d'impact sur la latence, Consommation d'énergie, congestion du réseau, coûts d'exploitation. IFogSim permet de simuler un environnement de Fog et d'IoT incluant des capteurs, des actionneurs, des nœuds de Fog et des centres de données de Cloud. Les applications d'IoT simulées dans IFogSim sont basées sur le modèle Perception Traitement-Action. Dans ce modèle, un ensemble de capteurs collecte d'abord des informations sur l'environnement physique de l'application et les publie de manière périodique ou événementielle. Ces informations sont capturées et traitées par le Fog et un ensemble d'instances de service IoT déployées dans le Cloud. En conclusion, les résultats du traitement sont des messages sont envoyés aux acteurs et agissent sur l'environnement physique de l'application **[61].**

III.4. Implémentation de simulation

III.4.1 La topologie du réseau

Pour ce projet de fin d'étude on a réalisé une application « **HealthcareMonotoringSystem**» qui permet de surveillé et traité les patients d'un seul service d'urgence par faire l'ordonnancement entre eux et afficher leur niveau d'urgence, Dans ce qui suit nous allons présenter la topologie du notre réseau.

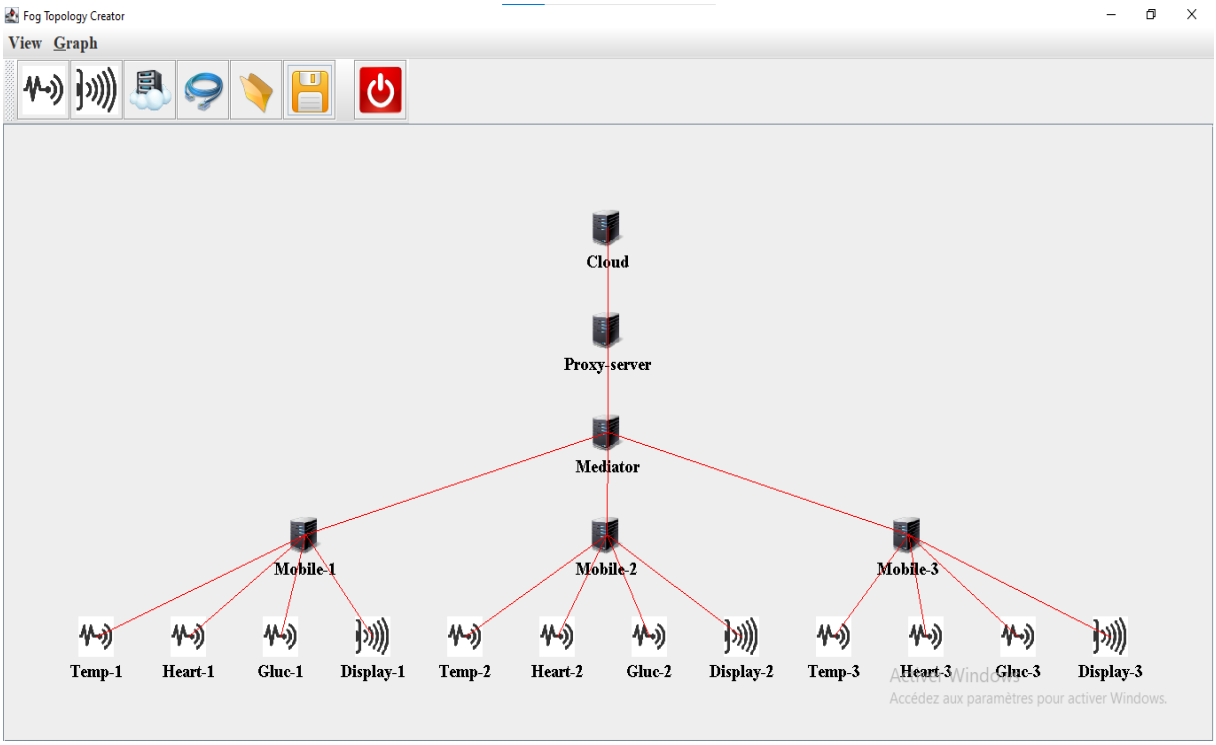


Figure III.3: La topologie du réseau.

D'après la topologie représentée dans la figure précédente (**Figure III.3**), les nœuds du Fog est exprimé par le Médiator et les 3 mobiles qui sont la couche intermédiaire entre le Cloud et les IoT capteurs.

- **Les nœuds du Fog**
  - **Le Médiator :** L'architecture de notre réseau se base sur le Médiator qui effectue le traitement-Client, il relie tous les mobiles et regroupe leurs données pour faire l'ordonnancement des taches, ensuite retourne une valeur qui indique le niveau d'urgence d'un patient à chaque mobile.

- **Le Mobile :** est attaché avec des capteurs et un actionneur, il représente un patient dans le service d'urgence, il reçoit les signaux de chaque capteurs, ensuite il transfère les données obtenu au Médiator pour le traitement, ainsi il va expédier la réponse de traitement vers le Display en cas d'urgence.

➢ **La couche inférieure d'IoT**

- **Capteurs (Sensors) :**

Les capteurs sont des dispositifs qui transforment l'état d'une grandeur physique observée à une grandeur utilisable **[62]**, Ils sont appliqués dans plusieurs segments comme les segments de soins pour surveiller un patient. Dans notre application Ils permettent de mesurer les données telles que la température (**Ex : Temp-1**), la tension artérielle (**Ex : Heart-1**), le taux de glucose (**Ex : Gluc-1**). **[Figure III.3]**

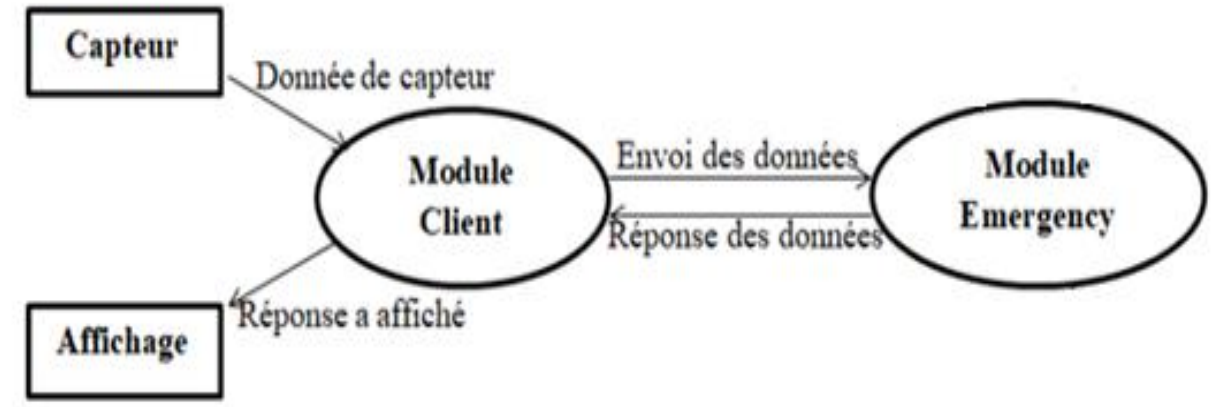
Ces capteurs sont placés sur un patient, ils sont particularisés pour transférer les données capturées vers le mobile.

- **Actionneur (Actuators) :**

L'actionneur est un dispositif matériel pour transformer une information digitale en un phénomène physique **[63]**, dans notre cas d'affichage nous permet d'afficher la réponse de traitement reçus qui est chargé en cas d'urgence (**Ex : Display-1**). **[Figure III.3]**

III.4.2.Les modules de l'application

Notre application est composée de 2 Modules (**Emergency** et **Client**) qui sont placé sur les nœuds du Fog comme ce suit: Emergency Module est hébergé sur le Médiator du service, pour le Client Module**,** il est placé sur chaque mobile du patient. La **Figure III.4** montre les relations et les échanges de données entre les modules.



2

Figure III.4: Modèle de l'application «HealthcareMonotoringSystem»

- **Module Client :** Ce module regroupe tous les données de patient, il collecte les données des capteurs et les envoyer vers Emregency module, ainsi il transfère le niveau d'urgence qui le reçoit vers l'actionneur.
- **Module Emergency** : Gérer l'ordonnancement des taches qui a été mis en œuvre sur la base de la priorité (selon le niveau d'urgence) et la méthode d'optimisation multicritères « **TOPSIS** », les résultats sont renvoyés vers le module Client pour les afficher dans l'actionneur.

III.4.3. Les algorithmes implémentés dans la simulation

On a impliqué sur notre simulation un ordonnancement des taches en utilisant la méthode d'optimisation multicritères et le niveau d'urgence d'un patient, on a les calculé a partir des données de data-set qui est intégré dans notre simulation. On a un seul data-set concerne les patients cardiologies, cette data-set est un fichier csv « **heart(1).csv** » **[64]** qui contient toutes les informations des patients cardiologies comme : (**Age**, **RestingBloodPressure**, **Cholesterol**, etc.). La **figure III.5** montre le data-set qu'on a utilisé et les données de chaque patient.

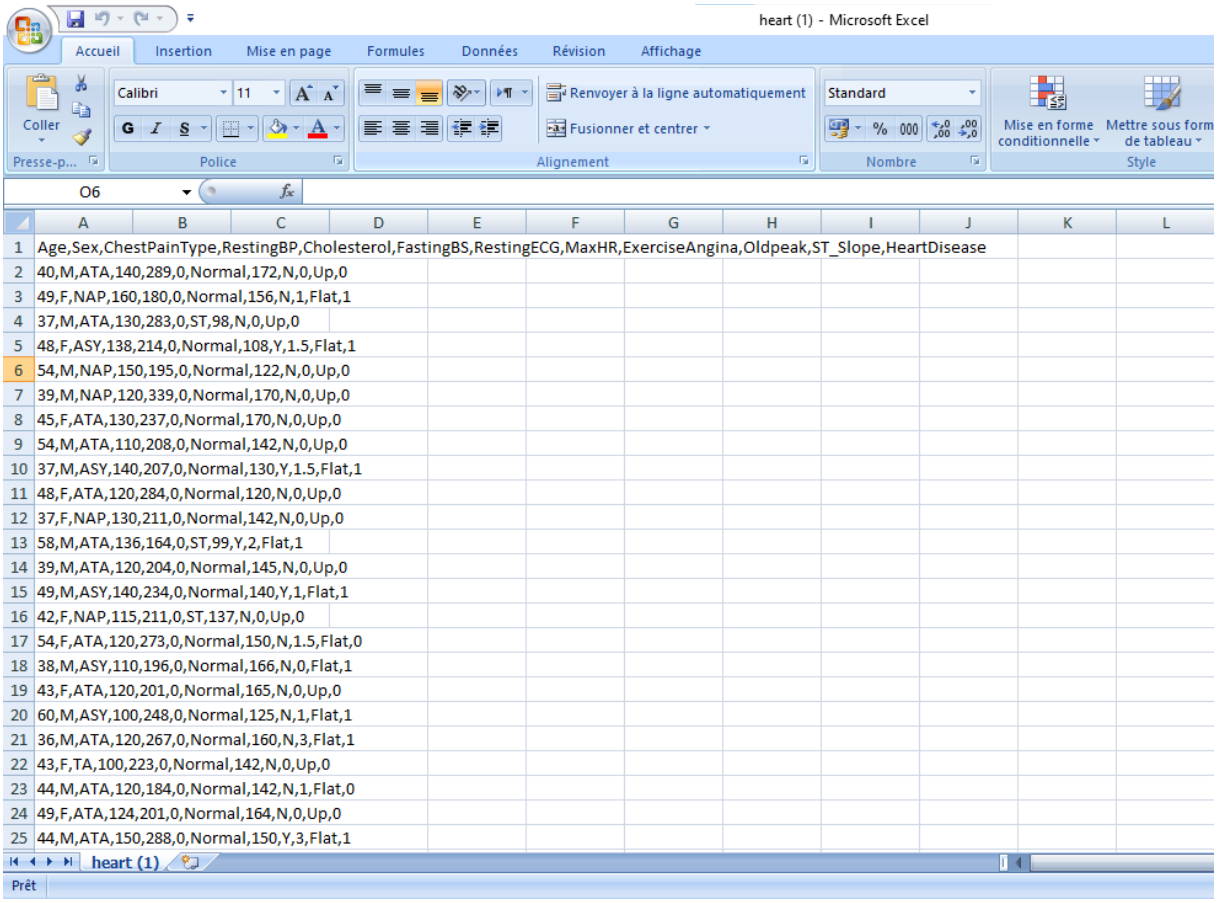


Figure III.5: Le data-set utilisé dans la simulation.

Pour garantir l'efficacité de l'ordonnancement des taches on a implémenté deux méthodes qui sont dépendent l'un de l'autre :

i. Calculer le niveau d'urgence

On a considère 3 niveau d'urgence qui sont : **Critique**, **Medium**, **Stable** avec les entier (**3, 2, 1**) respectivement, le calcul du niveau d'urgence est basée sur les données de la colonne « **RestingBP** », on a récupérer les données de ce dernier et on a calculé le niveau d'urgence par suivi les conditions suivant : **[65]**

- Si **RestingBP**>139 le niveau d'urgence de patient sera **3.**
- Si **RestingBP**>120 le niveau d'urgence de patient sera **2.**
- Sinon le niveau d'urgence de patient sera **1**.

Le tableau suivant montre le niveau d'urgence calculé de 5 patients

	RestingBP	Niveau d'urgence
Patient 01	**140**	**3**
Patient 02	**110**	**1**
Patient 03	**138**	**2**
Patient 04	**100**	**1**
Patient 05	**130**	**2**

Tableau III.2: Le niveau d'urgence des patients

ii. Utiliser la méthode d'optimisation multicritères «TopSis»

L'utilisation de l'algorithme **TopSis** et dépend de niveau d'urgence de patient qu'on a calculé dans la première méthode, on a considéré les critères suivant sur notre algorithme : **Niveau d'urgence**, **Age** et **Cardiopathie,** donc on a récupéré de data-set de la **figure III.15** les données des colonnes « **Age** » et « **HeartDisease** ». Ensuite, on a récupéré le « **Niveau d'urgence »** de chaque patient qu'on a calculé récemment et on a intégré ces données dans le **TopSis** pour calculer la solution idéale et anti idéale, le tableau suivant montre un exemple sur les critères utilisés dans la méthode pour déterminer l'ordonnancement des taches :

	Niveau d'urgence	Age	HeartDisease
Patient 01	**3**	**40**	**0**
Patient 02	**1**	**54**	**0**
Patient 03	**2**	**48**	**1**
Patient 04	**1**	**60**	**1**
Patient 05	**2**	**28**	**0**

Tableau III.3: Les critères utilisés dans le TopSis pour faire l'ordonnancement des taches.

Concernant les données de la colonne «**HeartDisease**», la valeur **0** ca veut dire que le patient n'a aucune maladie cardiaque, par contre, la valeur **1** montre que le patient souffre d'une maladie cardiaque.

Pour l'implémentation de la méthode TOPSIS, nous avons utilisé les poids mentionnés dans le **tableau 4** et fournis à l'aide de la méthode **AHP [66].**

Niveau d'urgence	Age	HeartDisease
0.6	**0.15**	**0.25**

Tableau III.4: Les pondérations associées à chaque critère.

Le niveau d'urgence a le poids fort car il représente l'état actuel de patient, donc il est un facteur important dans la décision, Ainsi, dans la solution idéale, nous allons essayer de le maximiser. Pour les autres critères on a donné le paramètre HeartDisease un poids supérieur à l'âge car il montre si le patient a un problème de santé cardiaque, Ainsi, dans la solution idéale, nous allons essayer de les maximiser.

On a traité les résultats obtenus de l'algorithme par faire un ordonnancement descendent de la solution maximale vers la solution minimale, donc on a donné la priorité à les patients qui ont une valeur maximale.

III.4.4. Présentation de simulation

III.4.1.1. Les classes de la simulation

Pour réaliser la simulation nous avons intégré 4 classes essentielles qui sont : **HealthHandler.Java**, **PFE_FOG.Java, DataParser.Java, HealthcareController.Java**, ces classes permet de gérer et manager notre simulation.

- **La classeHealthHandler.Java**

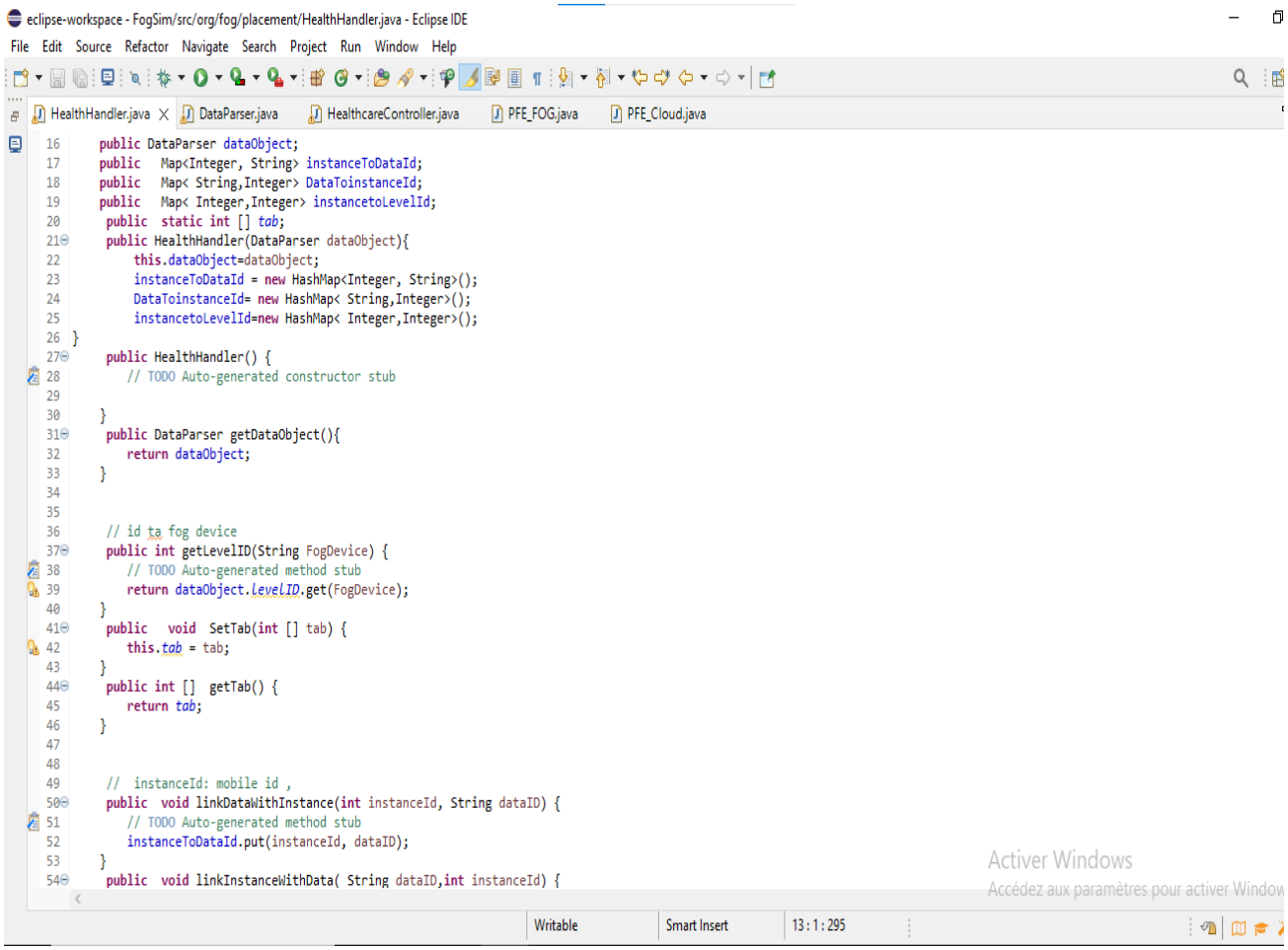


Figure III.6: La classe HealthHandler.

Cette classe contient toutes les méthodes que nous avons utilisées pour faire l'ordonnancement des taches dans la simulation comme : le TopSis, la méthode qui calcule le niveau d'urgence et autres méthodes qui sont relié avec les autres classes.

- **La classePFE_FOG.Java**

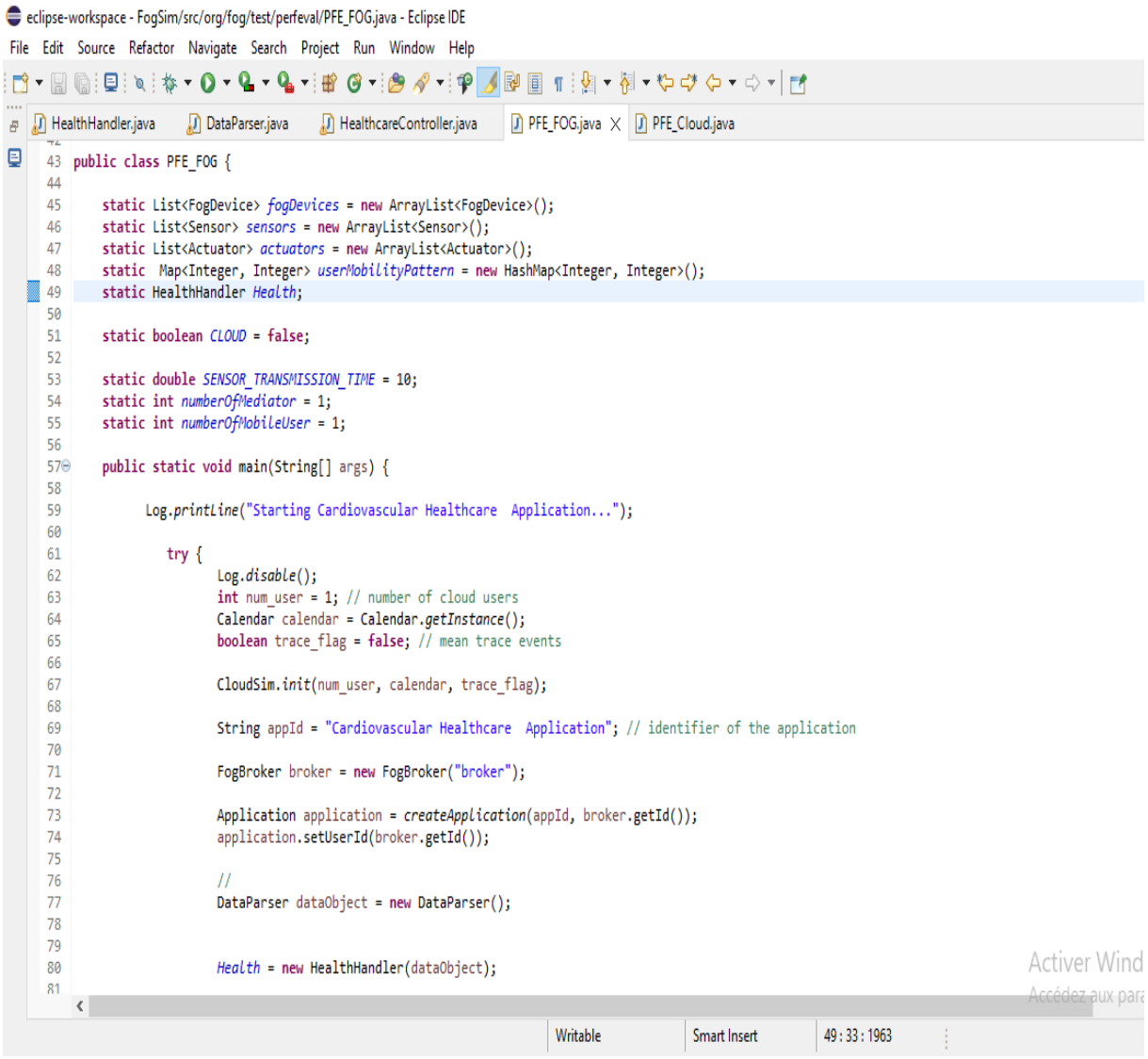


Figure III.7: La classe PFE_FOG.

Cette classe représente la classe Main de la simulation qui contient les entités physique qui représentent la topologie de réseau, ainsi, les modules qui construisent notre application.

- **La classeDataParser.Java**

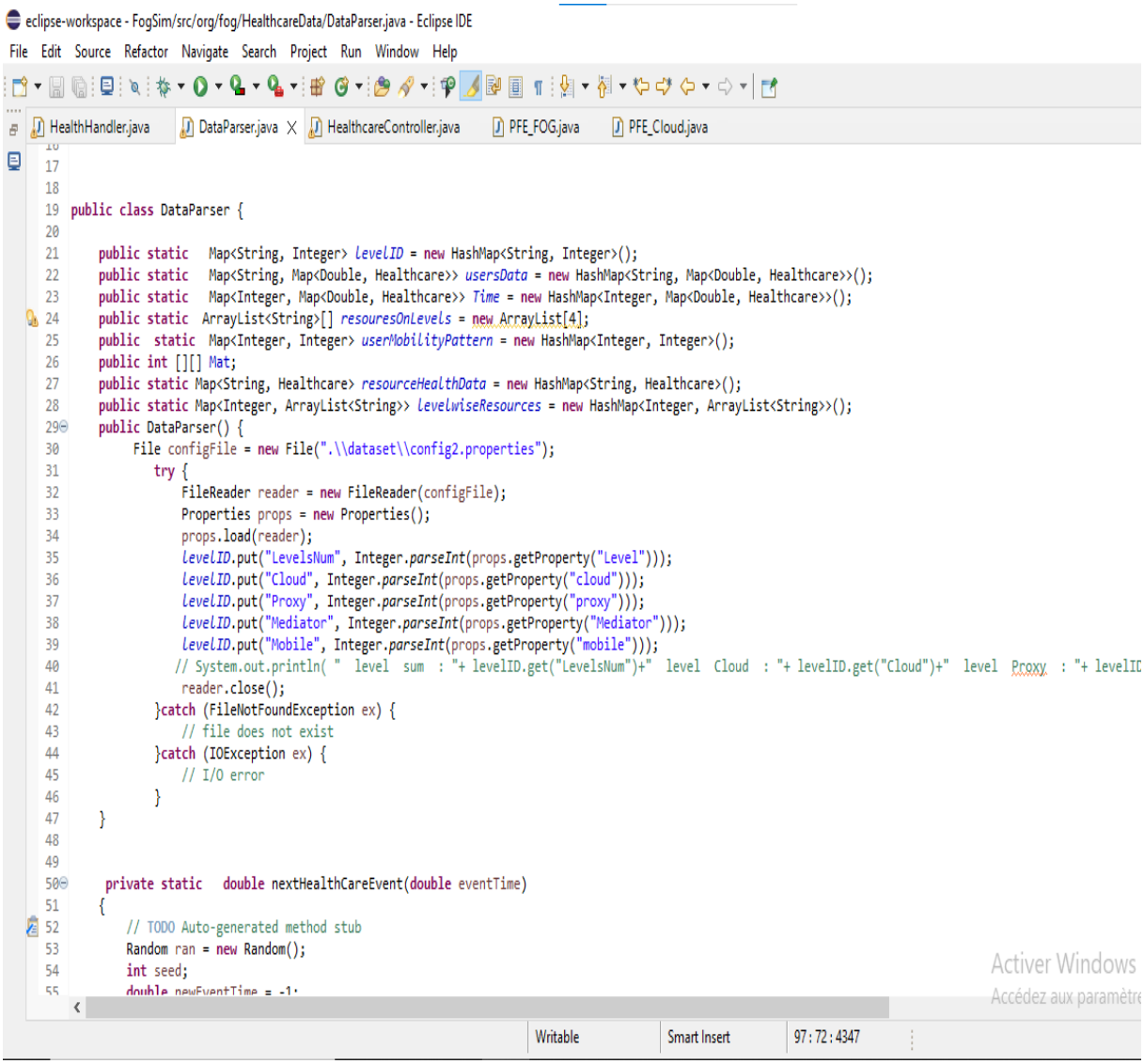


Figure III.8 : La classe DataParser.

Cette classe permet de récupérer les données des patients de data-set « **heart(1).csv** » et les transférer vers la classe **HealthHandler.Java**pour faire l'ordonnancement entre les patients.

- **La classeHealthcareController.Java**

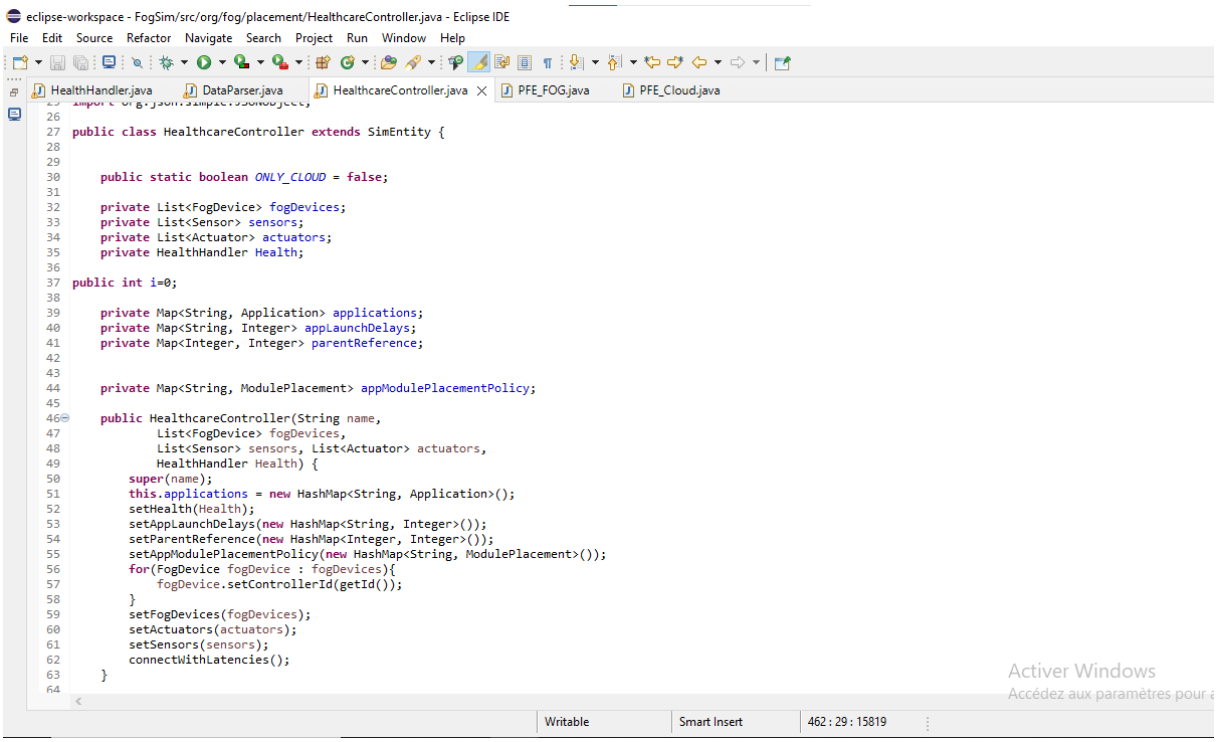


Figure III.9: La classe HealthcareController.

Cette classe est considère comme le cœur de simulation car elle permet de manager et contrôler les entités physiques (FogDevices, Sensors, Actuators) et logiques(les modules de l'application), elle permet aussi de manager les événements du Fog, ainsi, elle permet de soumettre l'application dans les nœuds du Fog, elle relié toutes les précédentes classes entre eux.

III.4.4.2. Evaluation des résultats obtenus

A. Les résultats obtenus de la simulation

Dans ce qui suit nous allons montrer des captures d'écrans des résultats obtenus de notre simulation.

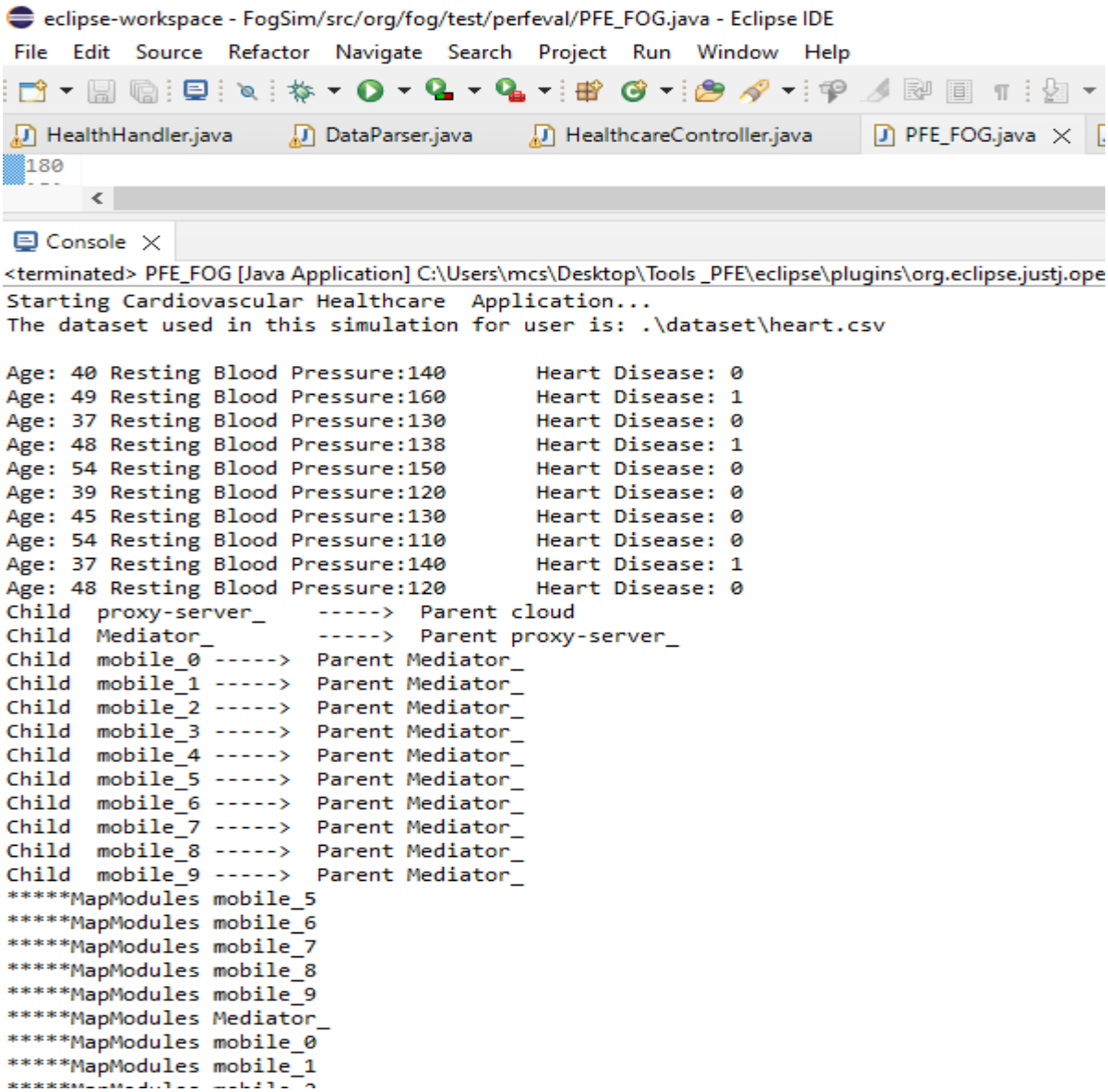


Figure III.10: Le début de la simulation.

Dans cette figure on observe que la simulation au début montre le data-set utiliser dans la simulation et les données des patients récupérer, on a récupérer les données de 10 patients. La figure montre aussi les appareils créés et la relation parent-fils entre eux.

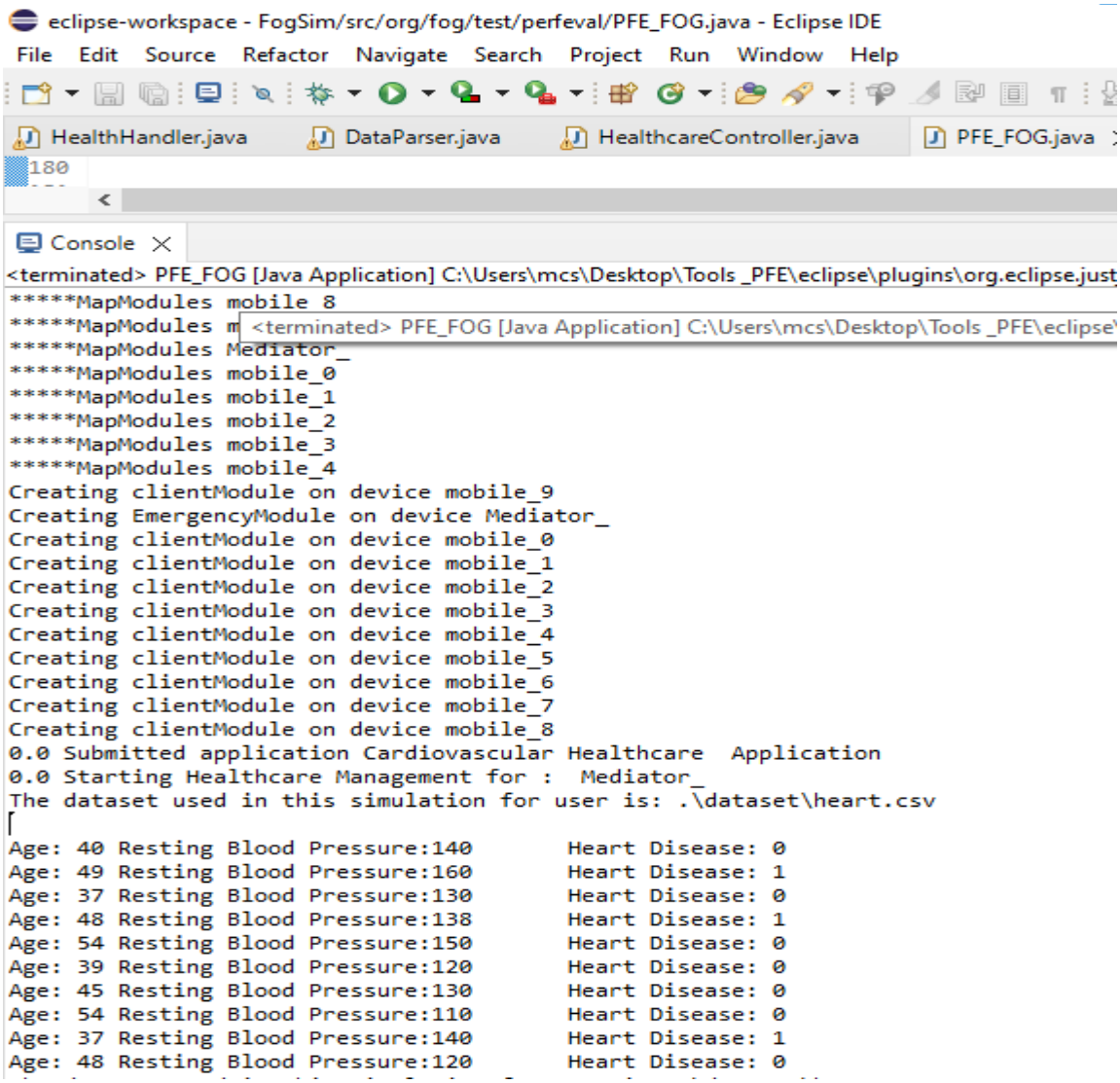


Figure III.11: La création des modules de l'application.

Cette figure montre la création des modules « Emergency » et « Client » qui construit notre application et le placement de chaque module sur les appareils du Fog.

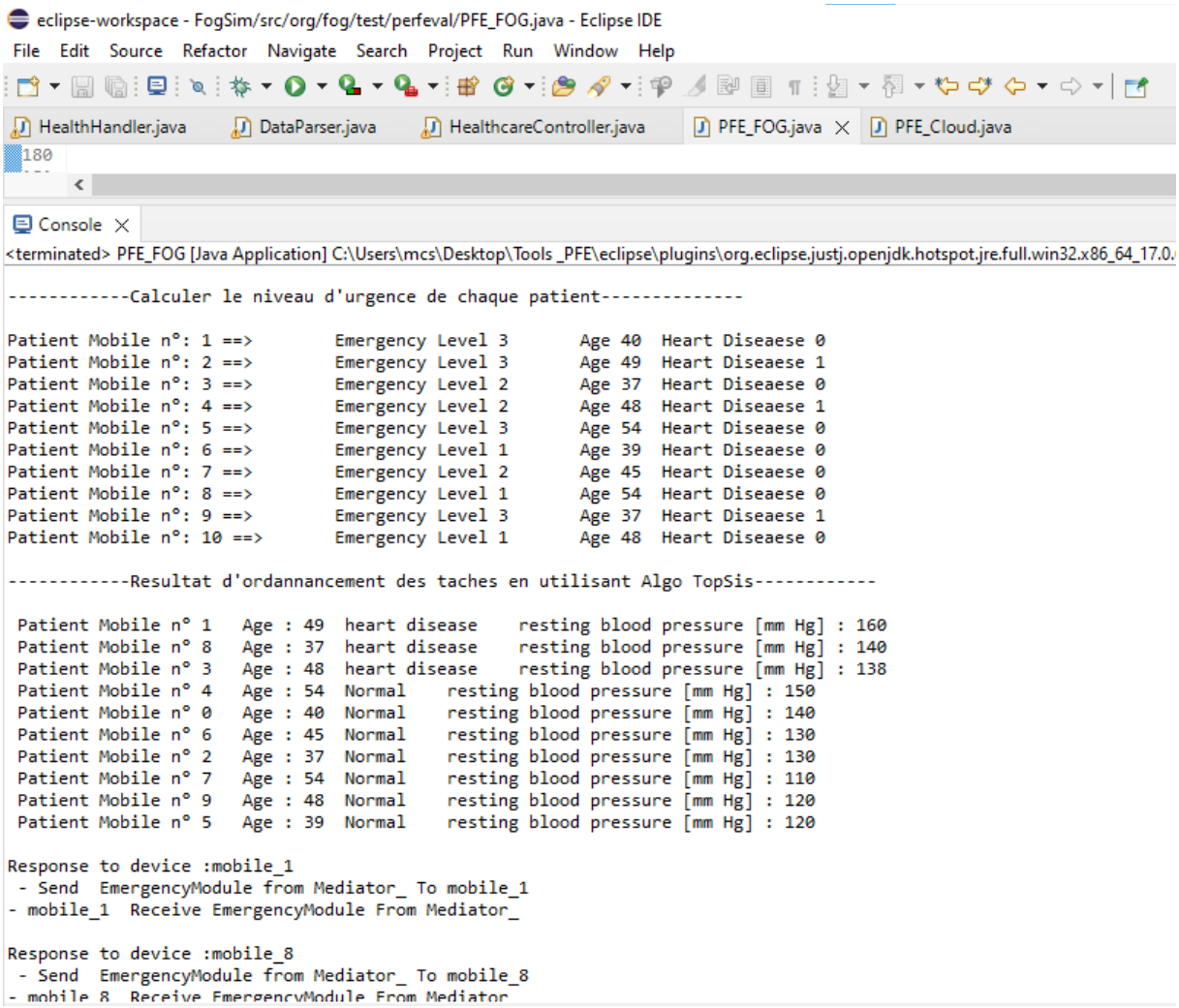


Figure III.12: Le résultat de l'ordonnancement des taches.

La figure montre la première étape de notre ordonnancement des taches qui est le calcul de niveau d'urgence de chaque patient à partir de leurs RestingBP, ensuite, elle montre la deuxième étape qui est l'utilisation de TopSis pour fournis le résultat de l'ordonnancement des taches.

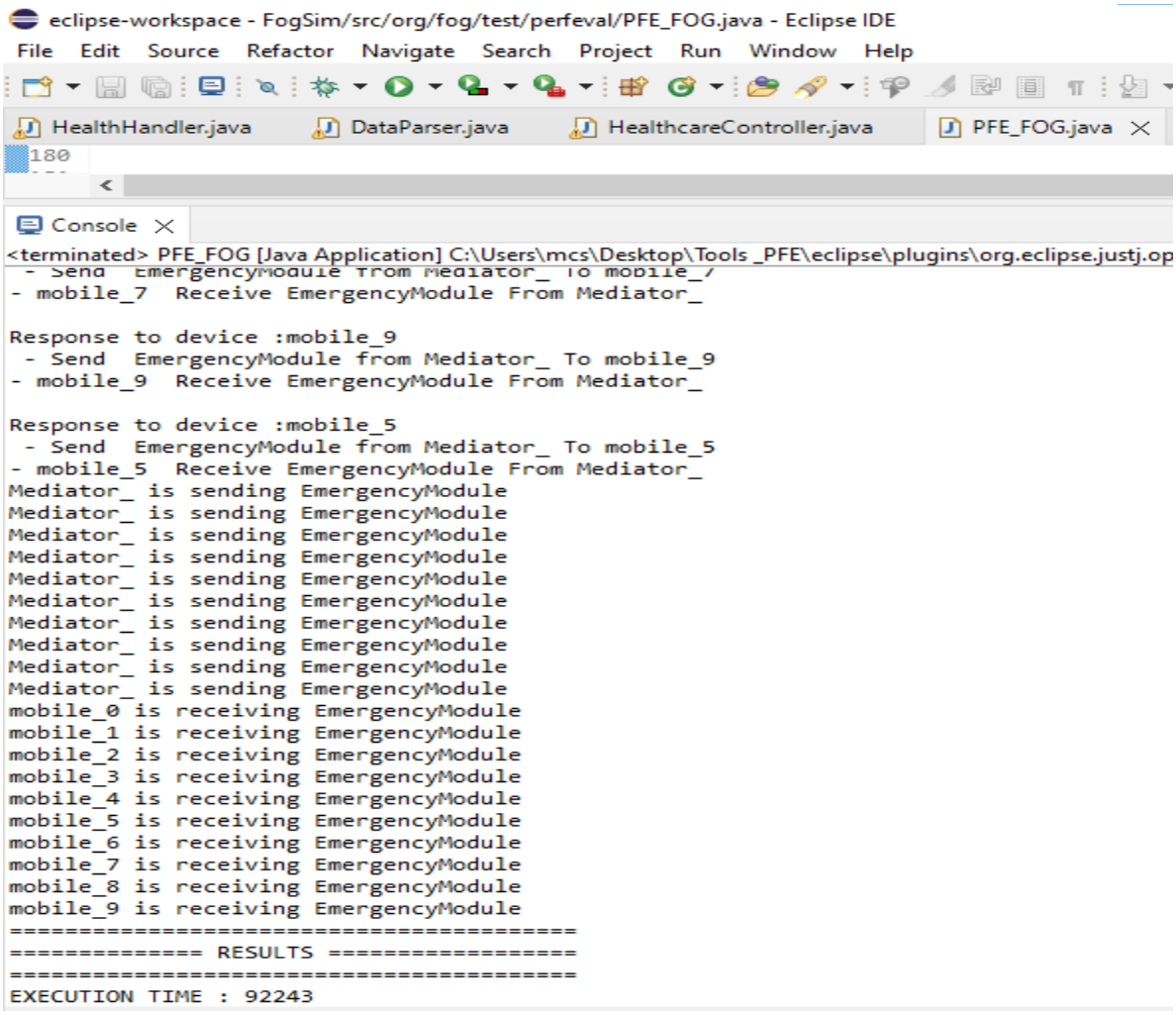


Figure III.13: Le transfert du niveau d'urgence de Médiator vers les mobiles.

Cette figure montre l'envoi de niveau d'urgence fournis par le module Emergency de médiator vers les mobiles, le médiator envoie le niveau d'urgence des patients en considérant l'ordonnancement qu'on a obtenu, chaque mobile reçois leur niveau d'urgence pour le afficher dans l'actionneur.

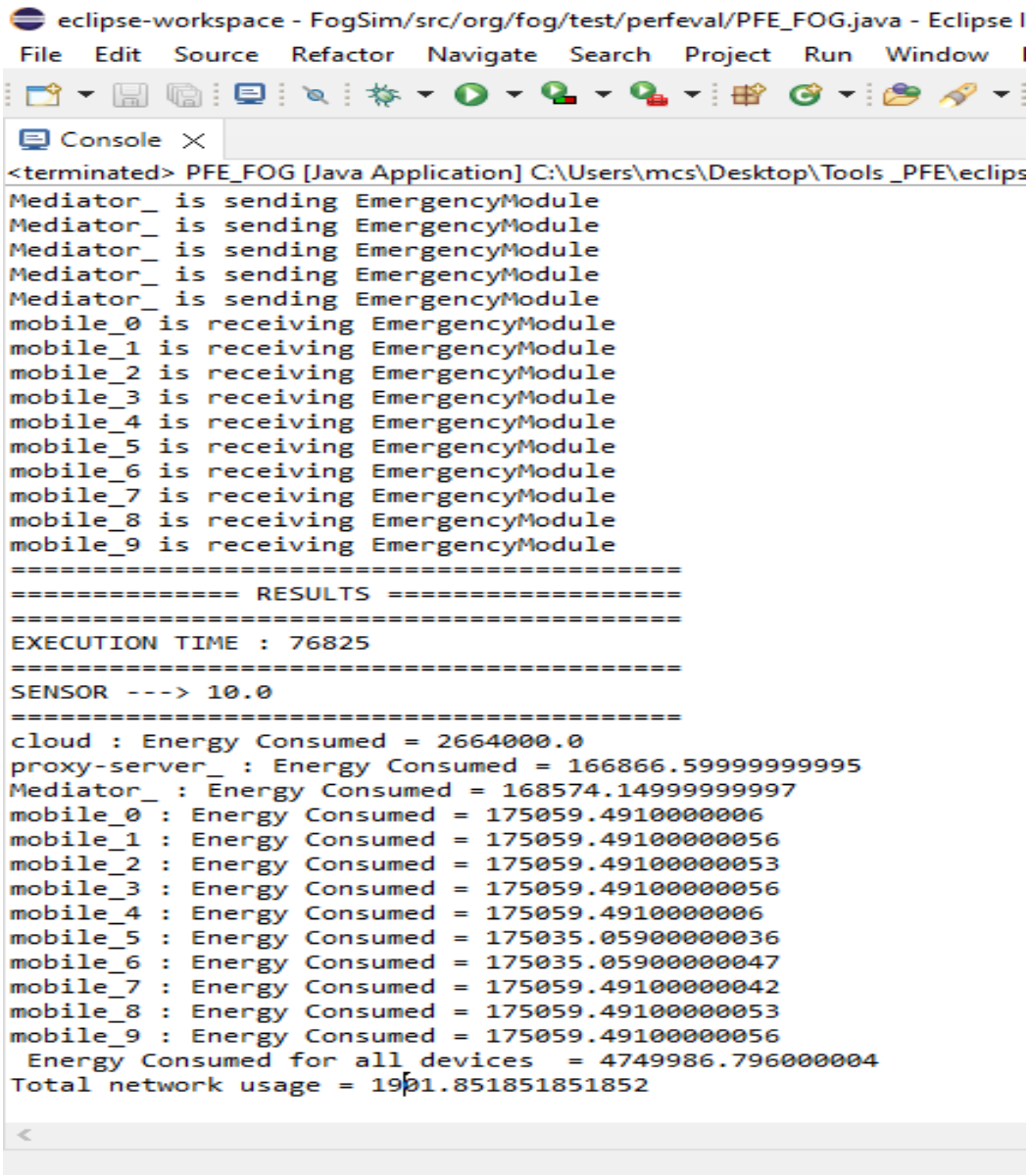


Figure III.14: les résultats de la simulation.

La figure montre les résultats de la simulation qu'on a obtenue comme le temps d'exécution de simulation, l'énergie consommée par chaque appareil et l'utilisation totale du réseau.

B. Discussion des résultats de simulation

- **Energie totale des appareils**

Pour étudier l'impact du nombre des tâches sur l'énergie totale, nous avons varié le nombre des tâches de 5 à 25 pour voir si la valeur de l'énergie reste stable ou elle augmente.

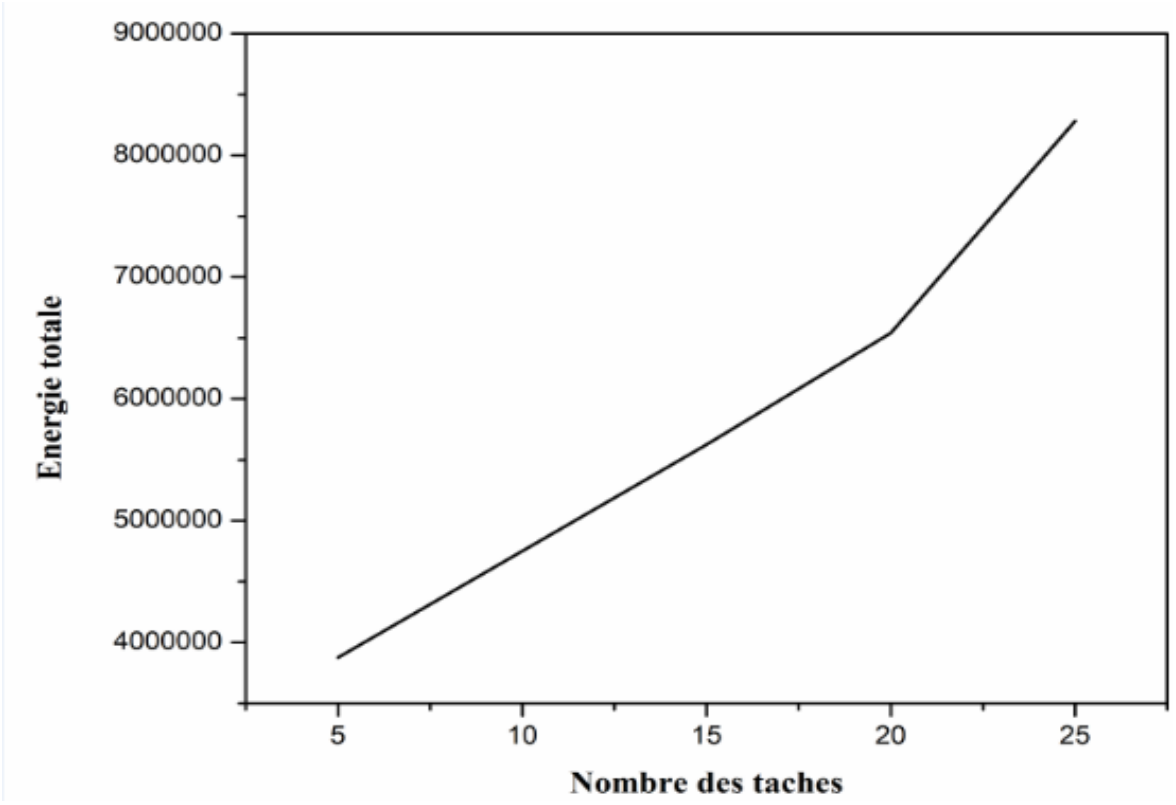


Figure III.15 :L'impact de nombre des tâches sur l'énergie totale des appareils.

Selon la **figure III.15**, nous remarquons que l'énergie totale augmente de manière proportionnelle avec le nombre des tâches car dans notre cas une tâche représente un capteur.

- **Utilisation totale de réseau**

La **figure III.16** montre l'impact du nombre des tâches sur l'utilisation totale de réseau. Nous remarquons clairement que l'utilisation de réseau augmente de manière proportionnelle avec le nombre des tâches.

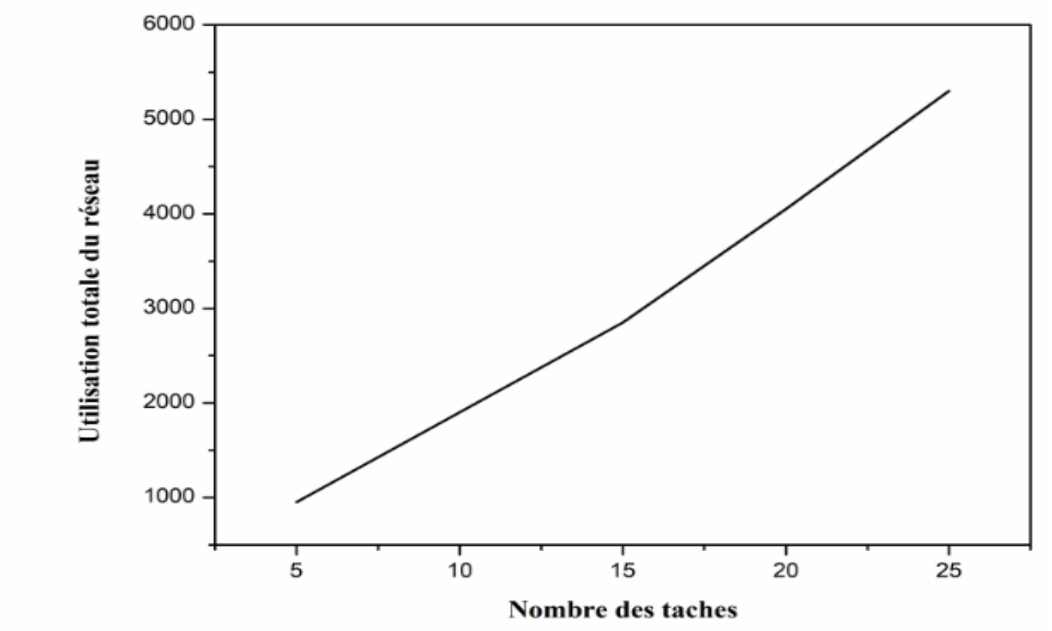


Figure III.16 :L'impact de nombre des tâches sur l'utilisation totale du réseau.

- **Le déploiement Cloud uniquement VS le déploiement Cloud-Fog**

Dans ce qui suit, nous allons présenter une comparaison entre un environnement cloud et un environnement cloud-fog concernant l'énergie et l'utilisation du réseau. Notre objectif est d'analyser ces résultats et de voir la différence entre ces deux déploiements.

- **Energie Totale des appareils**

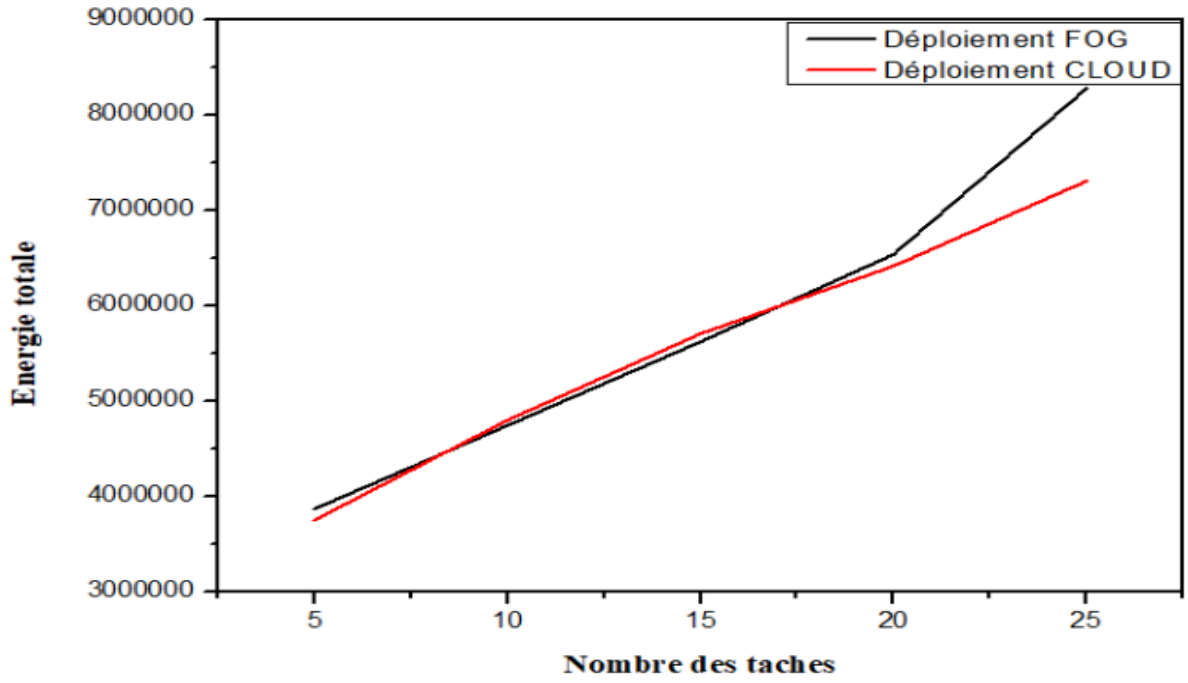


Figure III.17 :Une comparaison sur l'énergie totale entre le déploiement Cloud-Fog et le déploiement Cloud uniquement

La figure montre l'énergie consommée par tous les dispositifs dans la simulation. Le déploiement de l'application sur les dispositifs du Fog a été comparé au déploiement uniquement sur les centres de données en Cloud, l'emergency module effectuer l'ordonnancement des tâches qui consomme une grande quantité d'énergie. Par conséquent, comme le montre la **figure III.17**, lorsque le nombre des tâches augmente, la consommation d'énergie de ces dispositifs augmente aussi, en outre, la consommation d'énergie dans le centre de données en Cloud diminue lorsque les modules sont poussés vers des dispositifs de Fog.

- **Utilisation totale de réseau**

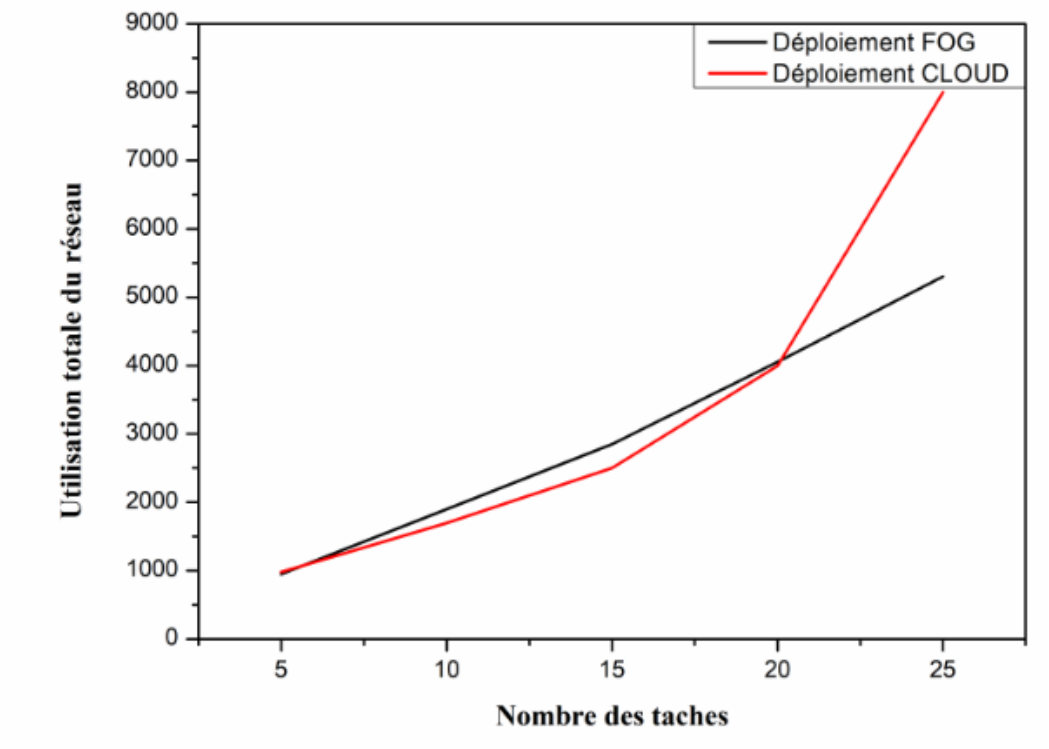


Figure III.18 :Une comparaison sur l'utilisation du réseau entre le déploiement Cloud-Fog et le déploiement Cloud uniquement.

Lorsque le nombre de dispositifs connectés à l'application (nombre des tâches) augmente, la charge sur le réseau s'accroît de manière significative dans le cas d'un déploiement Cloud uniquement, contrairement à un déploiement Cloud-Fog. Cette observation peut être attribuée au fait que dans l'exécution basée sur le Fog, la plupart des communications à forte intensité de données s'effectuent par le biais de liaisons à faible latence. Par conséquent, des modules tels que le module Emergency et les modules Clients sont placés sur les dispositifs périphériques, ce qui réduit considérablement le volume de données envoyées à un centre de données centralisé dans le Cloud.

III.5. Conclusion

Nous avons présenté dans ce chapitre notre contribution dans le cadre de ce PFE. Notre travail est basé sur l'implémentation d'un ordonnancement des tâches sur le service de cardiovasculaire en considérants le niveau d'urgence des patients et d'autres critères et en implémentant la méthode d'optimisation multicritère **TOPSIS** afin d'avoir un résultat optimal en fonction de la pondération attribuée à chaque critère.

Les résultats obtenus dans la partie étude comparative montrent la différence entre les deux déploiements sur l'énergie consommée et l'utilisation du réseau.

Conclusion générale

Ce mémoire a abordé la problématique de l'ordonnancement des tâches de maintenance dans le fog computing. L'objectif principal était de concevoir une architecture de services pour prendre en charge un ensemble d'appareils IoT hétérogènes utilisant différentes technologies basées sur le paradigme du fog computing. Le fog computing est considéré comme une solution adaptée pour accompagner l'expansion des services issus du cloud computing, car il permet de filtrer les données à la périphérie des appareils intelligents avant de les envoyer vers le cloud. Cela permet une utilisation plus efficace des ressources et des processus plus rapides.

Pour atteindre cet objectif, la méthode TOPSIS a été utilisée pour introduire plusieurs critères de gestion de la qualité de service (QoS). Cette méthode consiste à choisir une solution qui se rapproche le plus de la solution idéale (meilleure sur tous les critères) et s'éloigne le plus de la pire solution (qui dégrade tous les critères). Le simulateur iFogSim, largement adopté dans le domaine de l'informatique Edge/Fog, a été utilisé pour réaliser ce travail

Dans le cadre de ce mémoire, une architecture coopérative comprenant un service d'urgence pour surveiller la pression artérielle d'un client a été proposée. Les résultats obtenus ont démontré l'efficacité de l'approche adoptée. L'ordonnancement des tâches dans le fog a permis d'atteindre les résultats souhaités, en particulier pour le niveau d'urgence cardiaque, en utilisant un ordonnancement statique. Une comparaison entre le fog et le cloud a été réalisée en appliquant le même traitement dans le cloud, ce qui a permis d'observer la différence souhaitée.

Pour les travaux futurs, il est envisagé d'améliorer les solutions en intégrant d'autres capteurs tels que la température et le glucose, ainsi que plusieurs appareils mobiles, afin de réduire les délais de traitement et la consommation d'énergie pour l'approvisionnement des applications IoT. L'utilisation d'un ordonnancement dynamique sera également explorée pour obtenir de meilleurs résultats.

Références bibliographiques

1. https://actualiteinformatique.fr/internet-of-things-iot/quest-ce-que-iot-internet-of-things-internet-des-objets . Consulté le 10-12-2022.
2. «digora,» qu'est-ce que l'iot et pourquoi mener une stratégie d'iot ?, [en ligne]. available: https://www.digora.com/fr/blog/definition-iot-et-strategie-iot#. Consulté le 10-12-2022.
3. NAIT-SIDI-MOH, Ahmed, DURAND, David, FORTIN, Jérôme, et al. « Internet des objets et interopérabilité des flux logistiques: état de l'art et perspectives ». Université de Picardie Jule Verne, 2015.
4. https://images.app.goo.gl/7krfrmecsuv4ve237. Consulté le 10-12-2022
5. VIKAS HASSIJA, VINAY CHAMOLA, VIKAS SAXENA, DIVYANSH JAIN, PRANAV GOYAL, AND BIPLABSIKDAR « a survey on iotsecurity: application areas, security threats, and solution architectures. » ieee access, 7:82721–82743, 2019.
6. N. C. guillaume plouin, «blog.octo,» [en ligne]. available: blog.octo.com/modeles-architectures-internetdes-objets/.
7. SHANHE YI ET AL. "fog computing : platform and applications". Third ieee workshop on hot topics in web systems and technologies (hotweb). ieee. 2015
8. FLAVIO BONOMI, RODOLFO MILITO, JIANG ZHU ET SATEESH ADDEPALLI, « fog computing and its role in the internet of things », mcc '12 proceedings of the first edition of the mcc workshop on mobile cloud computing, acm, 17 août 2012, p. 13–16.
9. JANAKIRAM MSV, « is fog computing the next big thing in internet of things? », forbes, 18 avril 2016.
10. « membership information | openfog consortium » [archive],sur www.openfogconsortium.org. Consulté le 12-12-2022.
11. « the industrial internet consortium and openfog consortium join forces | industrial internet consortium » [archive], sur www.iiconsortium.org, communiqué de presse, 31 janvier 2019. Consulté le 12-12-2022.
12. VERMA, M., BHARDWAJ, N., &YADAV, A. K. « real time efficient scheduling algorithm for load balancing in fog computing environment » int. j. inf. technol. comput. sci, 8(4), 1-10. (2016).
13. BONOMI, F.; MILITO, R.; ZHU, J.; ADDEPALLI, S. « fog computing and its role in the internet of things ». in proceedings of the first edition of the mcc workshop on mobile cloud computing-mcc '12, helsinki, finland, ; pp. 13–15., 17 august 2012.

14. Yi, S.; Hao, Z.; Qin, Z.; Li, Q. « fogcomputing: platform and applications ». in proceedings of the 3rd workshop on hot topics in web systems and technologies, hotweb 2015, washington, dc, usa, pp. 73–78, 24–25 october 2016.
15. CHARITHPERERA, YONGRUI QIN, JULIO C ESTRELLA, STEPHAN REIFF-MARGANIEC, AND ATHANASIOS V VASILAKOS. « fog computing for sustainable smart cities : a survey. Acm computing surveys (csur) », 50(3) :32, 2017.
16. definition of fog computing available online : https://www.openfogconsortium.org/#definition-of-fog-computing. Consulté le 15 décembre 2022
17. https://tel.archives-ouvertes.fr/tel-03555562/document. Consulté le 15-12-2022
18. Fog computing and the internet of things: extend the cloud to where the things are. White paper. 2016. available online: http://www.cisco.com/c/dam/en_us/solutions/trends/iot/docs/computing-overview.pdf. Consulté le 15-12-2022.
19. AI, Y.; PENG, M.; ZHANG, K. « edge cloud computing technologies for internet of things : a primer. digit. commun. Network ». 2017.
20. SHI, Y.; DING, G.; WANG, H.; ROMAN, H.E.; LU, S. « the fog computing service for healthcare ». in proceedings of the 2015 2nd international symposium on future information and communication technologies for ubiquitous healthcare (ubi-healthtech), beijing, china, pp. 1–5, 28–30 may 2015;
21. MUKHERJEE, M.; SHU, L.; WANG, D. « survey of fog computing: fundamental, network applications, and research challenges ». ieee commun. surv. tutor. 2018.
22. AAZAM, M.; HUH, E.N. « fogcomputing micro data center based dynamic resource estimation and pricing model for iot ». proc. int. conf. adv. inf. netw. appl. aina 2015, 2015.
23. MUNTJIR, M.; RAHUL, M.; ALHUMYANI, H.A. « an analysis of internet of things (iot): novel architectures, modern applications, security aspects and future scope withlatest case studies ». int. j. eng. res. technol. , 6, 422–447, 2017.
24. https://www.researchgate.net/figure/layered-architecture-of-fog-computing_fig2_324280213. Consulté le 18-12-2022
25. LIU, Y.; FIELDSEND, J.E.; min, g. « a framework of fog computing: architecture, challenges and optimization.», 2017.
26. ATLAM, Hany F., ALASSAFI, Madini O., ALENEZI, Ahmed, *et al.* « XACML for Building Access Control Policies in Internet of Things ». In : *IoTBDS*. 2018. p. 253-

260.KETEL, M. « fog-cloud services for iot ». in proceedings of the south east conference . (2017, april).

27. PETER, N. « fog computing and its real time applications ». int. j. emerg. technol. adv. Eng, 266–269. 2015
28. Fog computing and the internet of things: a review - scientific figure on researchgate. Available from: https://www.researchgate.net/figure/fog-computing-supports-many-iot-applications-to-provide-better-service-to-customers_fig3_324280213. Consulter le 20-01-2023.
29. NIKOLOUDAKIS, Y.; MARKAKIS, E.; MASTORAKIS, G.; PALLIS, E.; SKIANIS, C. « an nf v-powered emergency system for smart enhanced living environments ». in proceedings of the 2017 ieee conference on network function virtualization and software defined networks (nfv-sdn), berlin, germany, 6–8 november, pp. 258–263. 2017;
30. DASTJERDI, A.V.; BUYYA, R. « fog computing: helping the internet of things realizeits potential ». ieee comput. soc. , 112–116,2016.
31. pour la solidarite .eu/fr/publication/quels-defis-pour-la-sante-en-europe extrait du « petit guide d'exploration de la santé numérique » publié en 2015 parla fondation de l'avenir 2015_petitguide_sante_numerique.pdf (fondationdelavenir.org). Consulté le 20-12-2022.
32. la e-santé, qu'est-ce que c'est ? | agence régionale de santé bourgogne-franche-comté (sante.fr). Consulté le 22-12-2022.
33. FATIMA ZAHRA, EL HABIB BEN LAHMAR, AND SANAA ELFILALI « internet of things: statistical study on research evolution » international journal of advances in electronics and computer science. (2019).
34. DE MOURA COSTA, H. J., DA COSTA, C. A., DA ROSA RIGHI, R., & ANTUNES, R. S. « Fog computing in health: a systematic literature review ». health and technology, 10, 1025-1044. (2020).
35. M. A. ALOULOU, « introduction aux problèmes d'ordonnancement », lamsade université paris dauphine, 2005.
36. H. SABBAH, « modélisation et dimensionnement d'une plate-forme hétérogène de service », thèse pour obtenir le grade de docteur, l'université de feranche-comte, 2009.
37. R. F. BERMAN, « adaptive computing on the grid using apples ». ieee transactions on parallel and distributed system, 2002.

38. D. E. INSAF, « optimisation d'ordonnancement et d'allocation de ressources dans le cloud computing », thèse de doctorat, 2016.
39. Y. C. -. P. SIARRY, « optimisation multi objectif », bld saint germain, 75240 paris cedex 05, 2002.
40. DR. DABBA ALI- « chapitre 6 : méthodes de résolution exactes heuristiques et métaheuristiques ».
41. INGO WEGENER ET R. pruim. « Complexity theory : exploring the limits of efficient algorithms ». springer-verlag, berlin, heidelberg, 2005. 35. informatique, université mohamed khider biskra. 2016.
42. BOUKEF BEN OTHMAN H., « sur l'ordonnancement d'ateliers job-shop flexibles et flow shop en industries pharmaceutiques optimisation par algorithmes génétiques et essaims particulaires», thèse de doctorat, ecole centrale de lille, 2009.
43. Baptiste P. And Le Pape C., « a constraint based branch and bound algorithm for preemptive job sop scheduling », proceedings of the international workshop on production planning and control, mons, belgium, 1996.
44. de, cryptographie et sécurité. cours des méthodes de résolution exactes heuristiques et métaheuristiques. Consulté le 05-02-2023
45. BAPTISTE MILLE, « méthodes approchées pour la résolution d'un problème d'ordonnancement avec travaux interférants ». rapport final de pfe, université de tours, 2014.
46. ALAOUI ABDIYA, « application des techniques des métaheuristiques pour l'optimisation de la tâche de la classification de la fouille de données ». thèse magistère, université d'oran, 2012.
47. SEBASTIEN, CAHON. PARADISEO : « une plate-forme pour la conception et le déploiement de métaheuristiques parallèles hybrides sur clusters et grilles ». thèse de doctorat, 2005.
48. DEFFAS ZINEB, « métaheuristiques parallèles à solution unique pour la résolution du problème du q3ap sur grille de calcul ». mémoire magister. université oum el bouaghui, 2010.
49. y. COOREN, « perfectionnement d'un algorithme adaptatif, d'optimisation par essaim de particules », thèse pour obtenir le grade de, l'université paris 12 val de marne, 2005.
50. https://images.app.goo.gl/vafhtsv7otfez6m6a. Consulté le 22-02-2023.

51. :https://www.google.com/url?sa=i&url=http%3a%2f%2fdevezeb.free.fr%2fdownloads%2fprojets%2ftspants.pdf&psig=aovvaw3ocdj1m0isl-zwwqesymvl&ust=1687344886800000&source=images&cd=vfe&ved=0cbmqjhxqfwotcmdrznbx0f8cfqaaaaadaaaaabai. Consulté le 24-04-2023
52. A.TOUMI, « restauration adaptative d'images par methodes intelligentes », thése pour obtenir le grade de docteur en science spécialité génie électrique, 2013.
53. https://images.app.goo.gl/jks497gpup7hbw3z5. Consulté le 18-06-2023.
54. HWANG, C. L., & YOON, K. « multiple attribute decision making: methods and applications, a state of the art survey ». 2012.
55. HWANG, C. L., LAI, Y. J., & LIU, T. Y. « a new approach for multiple objective decisions making. computers & operations research », 20(8), 889-899. (1993).
56. java (langage) : https://www.techno-science.net/glossaire-definition/javalangage.html . consulté le 21-06-2023.
57. introduction à l'utilisation d'eclipse (labri.fr). Consulté le 21-06-2023.
58. R. N. CALHEIROS, R. RANJAN, A. BELOGLAZOV, C. A. F. DE ROSE, AND R. BUYYA, « CLOUDSIM: a toolkit for modeling and simulation of cloud computing environments and evaluation of resource provisioning algorithms, software: practice and experience », 41(1): 23-50, 2011.
59. MAHMUD, R., & BUYYA, R. « modelling and simulation of fog and edge computing environment susing ifogsim toolkit. fog and edge computing: principles and paradigms », 1-35. (2019).
60. capteurs - analogiques - numériques - tor - photoélectriques - f2school Consulté le 25-06-2023.
61. capteurs et actionneurs - maxicours. Consulté le 25-06-2023.
62. physionet. Consulté le 20-04-2023.
63.
64. la tension artérielle et ses valeurs expliquées simplement | hirslanden. Consulté le 25-06-2023.
65. le résultat de la méthode ahp fournis par le co-encadrant arichi bouchra le 18-06-2023.
66. l'architecture d'ifogsim. | downloadscientificdiagram (researchgate.net). Consulté le 25-06-2023.

Annexes

Annexe A

Exemple d'utilisation de l'algorithme TOPSIS

Nous allons utiliser un exemple pour expliquer l'utilisation de la méthode TOPSIS. Dans cet exemple, nous construirons une matrice de décision basée sur l'analyse de la sélection de modèles de voitures (alternatives) en fonction des critères suivants :

- Style
- Fiabilité
- Consommation
- Cout

Qui respectivement ont le poids suivant : Poids (Wi) = {0.1, 0.4, 0.3, 0.2}.

	$style$	$fiabilité$	$Consommation$	$cout$
GOLF	5	8	8	6
PEUGEOT	9	6	5	7
PICASSO	7	7	5	9
SATURN	8	6	5	8

Etape 1 : On normalise tous les scores de la matrice des niveaux attribués aux critères on applique la formule de rij

	$style$	$fiabilité$	$Consommation$	$cout$
GOLF	0.34	0.59	0.68	0.40
PEUGEOT	0.61	0.44	0.42	0.46
PICASSO	0.47	0.52	0.42	0.59
SATURN	0.54	0.44	0.42	0.53

Etape 2 : Dans cette étape, on multiplie simplement toutes les entrées ($r\ i\ j^{¨}$) de la matrice normalisée par le poids associé wj pour avoir leVij.

	$style$	$fiabilité$	$Consommation$	$cout$
GOLF	0.034	0.236	0.204	0.08
PEUGEOT	0.061	0.176	0.126	0.092
PICASSO	0.047	0.208	0.126	0.118
SATURN	0.054	0.176	0.126	0.106

Etape 3 :

a) Déterminer la solution idéale V^+

V^+= {0.061, 0.236, 0.204, 0.118}

	$style$	$fiabilité$	$Consommation$	$cout$
GOLF	0.034	**0.236**	**0.204**	0.08
PEUGEOT	**0.061**	0.176	0.126	0.092
PICASSO	0.047	0.208	0.126	**0.118**
SATURN	0.054	0.176	0.126	0.106

b) Déterminer la solution idéale V^-

V^- = {0.034, 0.176, 0.126, 0.08}

	$style$	$fiabilité$	$Consommation$	$cout$
GOLF	**0.034**	0.236	0.204	**0.08**
PEUGEOT	0.061	**0.176**	**0.126**	0.092
PICASSO	0.047	0.208	**0.126**	0.118
SATURN	0.054	**0.176**	**0.126**	0.106

Étape 4:

a) Distance idéale

V^+= {0.061, 0.236, 0.204, 0.118}

	$style$	$fiabilité$	$Consommation$	$cout$
GOLF	$(0.034-0.061)^2$	$(0.236-0.236)^2$	$(0.204-0.204)^2$	$(0.080-0.118)^2$
PEUGEOT	$(0.061-0.061)^2$	$(0.176-0.236)^2$	$(0.126-0.204)^2$	$(0.092-0.118)^2$
PICASSO	$(0.047-0.061)^2$	$(0.208-0.236)^2$	$(0.126-0.204)^2$	$(0.118-0.118)^2$
SATURN	$(0.054-0.061)^2$	$(0.176-0.236)^2$	$(0.126-0.204)^2$	$(0.106-0.118)^2$

- On utilisant la formule de calcul les distances de séparation d_j^+ (idéal positif) on aura les calculs suivant :

Distance idéale	
GOLF	0.00147
PEUGEOT	0.10176
PICASSO	0.08404
SATURN	0.09938

b) Distance anti-idéal

V^- = {0.034, 0.176, 0.126, 0.08}

	$style$	$fiabilité$	$Consommation$	$cout$
GOLF	$(0.034-0.034)^2$	$(0.236-0.176)^2$	$(0.204-0.126)^2$	$(0.080-0.08)^2$
PEUGEOT	$(0.061-0.034)^2$	$(0.176-0.176)^2$	$(0.126-0.126)^2$	$(0.092-0.08)^2$
PICASSO	$(0.047-0.034)^2$	$(0.208-0.176)^2$	$(0.126-0.126)^2$	$(0.118-0.08)^2$
SATURN	$(0.054-0.034)^2$	$(0.176-0.176)^2$	$(0.126-0.126)^2$	$(0.106-0.08)^2$

- On utilisant la formule de calcul les distances de séparation d_j^- (idéal négatif) on aura les calculs suivant :

Distance anti-idéale	
GOLF	0.09840
PEUGEOT	0.02954
PICASSO	0.05135
SATURN	0.03280

Étape 5: Nous voilà à la dernière étape dans laquelle nous calculons la proximité relative de la solution idéale utilisons la fonction D_j .

La solution idéale	
GOLF	**0.98528**
PEUGEOT	**0.22522**
PICASSO	0.37927

SATURN	0.24815

Donc la meilleure solution est : **0.98528**

Annexe B

Architecture initiale d'IFogSim

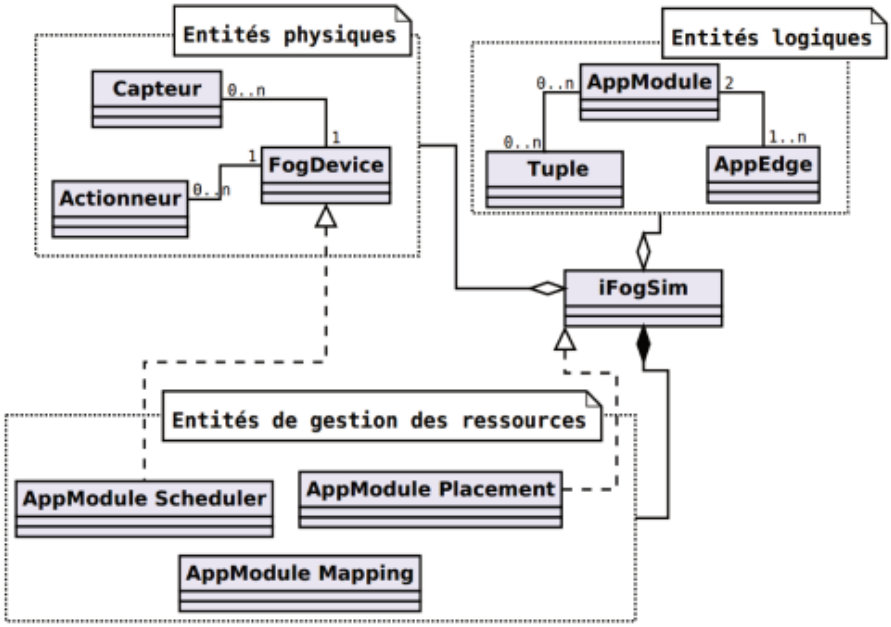


Figure B1 : **l'architecture initiale d'IFogSim [67].**

L'architecture de base d'iFogSim est comprend trois entité principale **[61]** :

Entités physiques : ce sont les modèles des équipements physiques présents dans une infrastructure Fog tels que les serveurs, les capteurs, les actionneurs etc.

✓ **FogDevice :** modélise les nœuds de Fog tels que les switchs, les routeurs, les stations de base, les passerelles, etc.). il Spécifie les caractéristiques matérielles de chaque nœud comme le modèle de processeur, la taille de la RAM, la capacité de stockage et la bande passante.

✓ **Capteur (Sensor) :** modélise les capteurs déployés dans l'infrastructure simulée, il spécifie les attributs de chaque capteur tels que la connectivité réseau et les données quil peut capturer.

✓ **Actionneur :** modélise les actionneurs et leurs effets sur l'environnement simulent. il spécifie les attributs de chaque actionneur tels que la connectivité, la latence et la passerelle associée.

Entités logique : ce sont des modèles représentent les éléments logiques dans un environnement informatique (**Ex**: données, services, traitements, etc.)

✓ **Tuple :** représente l'unité fondamentale des communications entre les entités physiques. Chaque tuple est caractérisé par son type, le nœud destinataire, la capacité du traitement de données demandes et la taille des donnes encapsulées.

✓ **AppModule :** représente les unités de traitement de scenario simule qui sont les instances de services de l'IOT. ces instances de services sont déployées dans les nœuds de Fog et les centres de données. chaque AppModule traite les donnée d'entrée (Tuple) et génère des données de sortie qui peuvent être envoyées a d'autres AppModule.

✓ **AppEdge :** représente les dépendances de données existantes entre deux AppModules (i.e. un producteur et un consommateur de données). De plus, une AppEdge spécifie le mode de production de données parmi les deux modes possibles : périodique ou événementiel.

Entités de gestion de ressources : ce sont des modèles représentant les stratégies de gestion de ressources physiques et logiques dans iFogSim (ex. allocation, ordonnancement, migration, etc.).

✓ **AppModule Placement :** gère le placement des AppModules dans l'infrastructure simulée.il offre des méthodes permettant aux utilisateurs de définir leur stratégie de placement afin d'optimiser un ou plusieurs critères tels que la latence, la consommation d'énergie, la congestion du réseau, les couts opérationnels.

✓ **AppModuleScheduler :** gère l'ordonnancement des AppModules au sein des nœuds de Fog ou au centre de données. Il propose des méthodes permettant aux utilisateurs de définir leur stratégie d'ordonnancement pour optimiser les performances.

✓ **AppModuleMapping :** cette entité définit pour chaque instance de service d'IoT, le nœud de Fog ou le centre de données qui l'héberge.

Résumé :

Ce projet de fin d'études se concentre sur l'utilisation du FogComputing dans le secteur de la santé pour détecter et traiter les urgences en temps réel. Le FogComputing permet de gérer localement les données générées par les objets connectés, réduisant ainsi les temps de réponse et offrant un meilleur contrôle des données. L'objectif est d'optimiser la planification des tâches dans le FogComputing en utilisant une méthode d'optimisation multicritère appelée TOPSIS. Le rapport présente une architecture de services pour les appareils IoT dans le domaine de la santé, en utilisant le FogComputing et en prenant en compte plusieurs critères de qualité de service. Des expérimentations ont été réalisées à l'aide d'un simulateur appelé iFogSim. Les résultats montrent l'efficacité de cette approche.

Mots clés :

Fogcomputing, e-santé, qualité de service, iFogSim, Topsis, multicritère

Abstract :

This graduation project focuses on the use of Fog Computing in the health sector to detect and treat emergencies in real time. Fog Computing makes it possible to locally manage the data generated by connected objects, thus reducing response times and offering better data control. The objective is to optimize task planning in Fog Computing using a multi-criteria optimization method called TOPSIS. The report presents a service architecture for IoT devices in healthcare, using fog computing and taking into account several quality of service criteria. Experiments were carried out using a simulator called iFogSim. The results show the effectiveness of this approach.

Keywords:

Fog computing, healthcare, quality of service, iFogSim, Topsis, multicriteria

ملخص:

يركز مشروع التخرج هذا على استخدام حوسبة الضباب في القطاع الصحي لاكتشاف حالات الطوارئ وعلاجها في الوقت الفعلي. تتيح حوسبة الضباب إمكانية إدارة البيانات التي تم إنشاؤها بواسطة الكائنات المتصلة محليًا ، وبالتالي تقليل أوقات الاستجابة وتوفير تحكم أفضل في البيانات. الهدف هو تحسين تخطيط المهام في الحوسبة الضبابية باستخدام طريقة تحسين متعددة المعايير تسمى TOPSIS. يقدم التقرير بنية خدمة لأجهزة إنترنت الأشياء في مجال الرعاية الصحية ، باستخدام حوسبة الضباب ومراعاة العديد من معايير جودة الخدمة. أجريت التجارب باستخدام جهاز محاكاة يسمى iFogSim. تظهر النتائج فعالية هذا النهج.

الكلمات المفتاحية :

حوسبة الضباب ، الصحة الإلكترونية ، جودة الخدمة ، iFogSim، Topsis ، معايير متعددة