THÈSE

Pour l'obtention du grade de :

DOCTEUR EN SCIENCES de l'Université Mohammed 1er d'Oujda -Maroc-

DOCTEUR EN SCIENCES de l'Université de Lille -France-

Formation Doctorale : Mathématiques et Informatique / Informatique

Spécialité : Informatique

Présentée et soutenue par:

M. Khalil Ibrahim HAMZAOUI

Contribution à la modélisation de la consommation d'énergie dans un dispositif mobile

Le : 21/06/2018

Président :	
Mr. M. MOUSSAOUI	**Professeur à l'École Supérieure de Technologie - Oujda**
Rapporteurs :	
Mme. F. BENABBOU	**Professeur à la Faculté des Sciences- Casablanca**
Mr Y. BELLIK	**Professeur associé (HDR) à l'Université de Paris Sud**
Examinateurs :	
Mr. M. AZIZI	**Professeur à l'École Supérieure de Technologie - Oujda**
Mr. G. LIPARI	**Professeur à la Faculté des Sciences et Technologies - Lille**
Directeurs de Thèse :	
Mr. P. BOULET	**Professeur à la Faculté des Sciences et Technologies - Lille**
Mr. M. BERRAJAA	**Professeur à la Faculté des Sciences - Oujda**

Remerciements

J’adresse mes premiers remerciements à mes encadrants. J’exprime toute ma gratitude au Pr Pierre Boulet. Très présent, il a été central dans l’aboutissement de mon doctorat. J’ai apprécié son encadrement avisé et ses qualités humaines. Au Pr Mohammed Berrajaa pour son encadrement et ses conseils précieux. Au Pr Giuseppe Lipari pour sa rigueur scientifique et ses remarques pointues. Au Pr Mostapha Azizi que je remercie profondément pour son co-encadrement avisé, pour son soutien et sa sympathie.

Je remercie également les membres de mon jury. Mme Fouzia. BENABBOU, Professeur à la Faculté des Sciences de Casablanca, M Yacine. BELLIK, Professeur associé (HDR) à l’Université de Paris Sud qui ont accepté de consacrer de leur temps précieux à la relecture de mon travail.

Je garderai un très bon souvenir de mon passage au sein de l’équipe Emeraude. Je me rappellerai de la sympathiques de mes collègues, Houssam, Mahyar, Philippe, Pierre et Richard.
Un grand merci à mes camarades de l’IRCICA.

J’adresse toute ma gratitude et reconnaissance à ceux qui ont contribué à la correction, l’amélioration ainsi que la relecture de ce manuscrit.

Que l’ensemble des personnes qui ont contribué de près ou de loin à la réalisation de cette thèse, trouve ici l’expression de mon profond respect et gratitude.

Dédicace

À mon père : Un témoignage de ma profonde affection et ma grande reconnaissance pour tous les sacrifices que tu as fais pour moi.

À ma mère (Medecin) : Avec toute mon affection et ma profonde gratitude pour ton amour, ton dévouement. Tes sacrifices m'ont été un grand soutient durant toutes années d'études. Que dieu tout puissant vous garde ma chère maman. Accepte ce travail en témoignage de ma reconnaissance, mon amour et mon affection.

À mes sœurs (Laila et Hajar) : Avec mon grand respect, vous avez contribué largement à l'élaboration de mon parcours.

Pour ma Fille Lina : Tout mon amour.

Pour toute ma famille maternelle et paternelle : Tout mes respects.

À mon oncle Youssef (Pharmacien) : Tes conseils m'ont été précieux.

À mon oncle Abdel-Ilah (Chirurgien) : Un grand merci pour pour ta présence, ton suivi et ton encouragement permanent.

À mon oncle Ahmed (Medecin biologiste) : Merci pour tes conseils et ton suivi.

Pour mes meilleurs amis : Hamza Kasmi, Anas Mokhtari, Soufiane Dahmani et Mohammed Gabli : Merci pour les bons moments passés ensemble.

Pour tout mes amis.

Table des matières

Résumé

La consommation d'énergie est le résultat d'interactions entre le matériel, les logiciels, les utilisateurs et l'environnement d'application. L'optimisation de la consommation d'énergie est devenue cruciale, la mesure de l'énergie est considérée comme une mesure critique, il est donc important de savoir comment mesurer et comprendre comment l'énergie est consommée sur les appareils mobiles. Des connaissances précises nous permettront de proposer différentes solutions pour réduire la consommation d'énergie afin d'améliorer l'expérience utilisateur. L'objectif principal de cette thèse consiste à modéliser la consommation d'énergie d'une application particulière fonctionnant sur un appareil mobile. Dans ce manuscrit, nous proposons un modèle de suivi du comportement énergétique, nous décrivons également une méthodologie pour identifier les paramètres du modèle. À cette fin, nous avons analysé une collection de données expérimentales recueillies lors de mon tour de France en fauteuil roulant électrique. Ensuite, nous avons appliqué des outils statistiques pour obtenir les paramètres du modèle. Nous validons enfin le modèle en comparant les résultats avec d'autres données expérimentales.

Dans le but de réaliser des modèles basés sur des cas concrets, trois études de cas ont été développées et sont présentées dans cette thèse.

— La première étude de cas permet de comparer l'évolution du coût énergétique dans les environnements mobiles ainsi que le comportement énergétique des différents composants des smartphones en se basant sur plusieurs modèles énergétiques.

— La deuxième étude de cas traite l'évaluation et les mesures du coût d'énergie consommée, ainsi que les problèmes rencontrés dans les méthodes utilisées pour l'évaluation de la consommation d'énergie. Pour une meilleure évaluation, on a introduit le cas d'étude du comportement énergétique en utilisant les machines virtuelles.

— La troisième étude de cas est basée sur le traitement des résultats des mesures obtenues lors de mon tour de France en fauteuil roulant électrique connecté. L'objectif consiste en une gestion anticipée des ressources, en réalisant des mesures réelles, ensuite, de faire le suivi du comportement énergétique dans un environnement réel et diversifié. Le modèle peut être utilisé pour définir une

fréquence optimale en termes de consommation d'énergie pour des situations précises sans trop dégrader la qualité de service souhaitée par l'utilisateur. La collecte de données scientifiques a été effectuée sur la base de plusieurs états sur différents lieux en fixant des métriques auparavant. Ces comparaisons de mesures ont été réalisées dans un environnement réel et confrontées aux mesures de consommation préalablement réalisées dans un environnement contrôlé.

Mots clés : Informatique mobile, Système d'exploitation, Modélisation de la consommation d'énergie.

Abstract

Energy consumption is the result of interactions between hardware, software, users, and the application environment. Optimization of energy consumption has become crucial, energy metric is considered a critical metric, so it is important knowing how to measure and understand how energy is consumed on mobile devices. Accurate knowledge will allow us to propose different solutions to reduce energy consumption in order to improve the user experience. The main objective of this thesis is to model the energy consumption of a particular application running on a mobile device. In this manuscript, we propose a model of energy behavior monitoring, we also describe a methodology to identify model parameters. To this end, we analyzed a collection of experimental data collected during a "Tour de France" in a wheelchair. Then we applied statistical tools to obtain the parameters of the model. We finally validate the model by comparing results against other experimental data. In order to create models based on concrete cases, three case studies have been developed and presented in this thesis.

— The first case study compares the evolution of the energy cost in mobile environments as well as the energy behavior of the different components of smart phones based on several energy models.
— The second case study deals with the evaluation and measurement of the energy cost consumed, as well as the problems encountered in the methods used for the evaluation of energy consumption. For a better evaluation, we have introduced the study of energy behavior using virtual machines.
— The third case study is based on the treatment of the results of the measurements obtained during my tour of France in a connected electric wheelchair. The objective is to anticipate resources by making real measurements and then monitor energy behavior in a real and diversified environment. The model can be used to define an optimal frequency in terms of energy consumption for precise situations without degrading too much the quality of service desired by the user. Collection of scientific data was carried out on the basis of several states on different places by fixing metrics previ-ously. These measures comparisons were carried out in a real environment and confronted with the consumption measures carried out beforehand in a controlled environment.

Index Terms Mobile computing, Operating system, Energy consumption modeling.

Chapitre 1

Introduction générale

Les Technologies de l'Information et de la Communication (TIC) sont un ensemble de techniques de l'informatique, de l'audiovisuel, du multimédia, d'Internet et des télécommunications qui permettent aux utilisateurs de communiquer, d'accéder aux sources d'information, de stocker, de manipuler, de produire et de transmettre l'information sous toutes les formes (texte, musique, son, image, vidéo, ...). Selon le rapport « Technologies Clés 2015, le secteur des technologies de l'information et de la communication (TIC)» est devenu un segment majeur de l'économie des principaux pays industrialisés avec une contribution directe de 5,9% du PIB en Europe (et 7,5% aux États-Unis). Les TIC représentant en effet plus de 50% de la croissance de la productivité en Europe [1].

Leur impact s'étend sur de multiples domaines, notamment sur notre mode de vie et notre économie. Les secteurs de production et d'utilisation de ces nouvelles techniques acquièrent une part croissante du PIB des économies développées et émergentes, d'où le concept de « nouvelle économie » ou « économie du savoir ».

Les opérateurs proposent des services de plus en plus évolués. La technologie du futur vise à interconnecter le monde et créer des villes durables, intelligentes et entièrement connectées. Le réseau doit disposer d'une capacité suffisante afin de gérer le trafic et garantir une qualité de service (QoS) adaptée. En effet, la QoS offerte définit les performances globales d'un réseau, notamment d'un point de vue utilisateur. Elle est définie comme la satisfaction du client par les services du réseau, ce qui inclut, en particulier, la qualité de la communication (audio et/ou vidéo), la puissance du signal de transmission et le taux d'acceptation des appels [4].

On observe une croissance exponentielle du nombre d'utilisateurs ayant recours à des services et applications sur internet à l'aide des différents appareils portables.

Le smartphone est incontestablement l'un des appareils les plus utiles du 21ème

1. http://www.agence-nationale-recherche.fr/suivi-bilan/editions-2013-et-anterieures/sciences-et-technologies-de-l-information-et-de-la-communication/

siècle, mais ils continuent d'avoir un défaut critique : l'autonomie de la batterie. Les utilisateurs ont probablement besoin de recharger leurs téléphones une fois par jour, mais que se passerait-il si une charge unique pouvait durer une semaine d'utilisation intensive ?.

En conséquence, les fabricants s'intéressent plus au concept d'efficacité énergétique dans différents domaines (réseaux sans fil, autonomie des batteries ...) vu la demande croissante des équipements sans fil. Les utilisateurs dépendent de plus en plus d'appareils avec des ressources d'énergie à capacités limitées. L'optimisation de

Figure 1.1 – Forte dépendance aux appareils avec sources d'énergie à capacités limitées

la consommation d'énergie constitue une part de plus en plus importante des politiques énergétiques durables. Beaucoup d'acteurs sont impliqués dans ces innovations (constructeurs et équipementiers automobiles, industrie informatique et opérateurs de téléphonie mobile).

Dans le cadre de cette thèse, nous nous intéressons au problème de gestion des ressources et au contrôle de la consommation énergétique par le suivi de la consommation d'énergie en réagissant sur le comportement énergétique du mobile en question. Les dispositifs mobiles disposent de plusieurs capteurs qui permettent de déterminer par la suite les différentes informations nécessaires pour notre étude tout en veillant sur la qualité de service qui permettra une meilleure utilisation.

La qualité de service (QoS) est un concept de gestion qui consiste à donner des

priorités à certaines applications pour garder un certain niveau de performance. Du point de vue réseau, la qualité de service est la capacité que peut fournir un réseau pour atteindre un niveau garanti.

Les utilisateurs apprécient la QoS en se basant sur les critères suivants :

— Fiabilité : Capacité de réaliser le service promis de manière sûre et précise.

— Rapidité : Possibilité de fournir un service rapide et adapté.

On a une meilleure économie d'énergie quand on ne la consomme pas. Il faut donc identifier les problèmes à la source, une bonne supervision des performances permet la réalisation d'une analyse correcte qui permettra l'implémentation des solutions optimales pour la modélisation de consommation pour permettre un bon pilotage énergétique.

Dans ma thèse, je me suis basé sur une tablette mobile. La méthodologie proposée peut être réutilisée sur d'autres plate-formes en utilisant des outils équivalents.

L'objectif de cette thèse entre dans le cadre de modélisation énergétique dédiée aux smartphones. Dans le but de réaliser des modèles basés sur des cas concrets, trois études de cas ont été développées et présentées dans cette thèse.

— La première étude de cas permet de comparer l'évolution du coût énergétique dans les environnements mobiles ainsi que le comportement énergétique des différents composants des smartphones en se basant sur des plusieurs modèles énergétiques.

— La deuxième étude de cas traite l'évaluation et les mesures du coût d'énergie consommée, ainsi que les problèmes rencontrés dans les méthodes utilisées pour l'évaluation de la consommation d'énergie. Pour une meilleure évaluation, on a introduit le cas d'étude du comportement énergétique en utilisant les machines virtuelles.

— La troisième étude de cas est basée sur le traitement des résultats des mesures obtenues lors de mon tour de France en fauteuil roulant électrique connecté. L'objectif consiste en une gestion anticipée des ressources, en réalisant des mesures réelles, ensuite, de faire le suivi du comportement énergétique dans un environnement diversifié.
  Le contexte de réalisation de cette étude est détaillé dans le chapitre 6.

1.1 Objectifs de la thèse

Les travaux de recherche de ma thèse sont axés sur l'étude des performances et de la qualité des services (QoS) dans les environnements mobiles. Cette thématique de recherche envisage, notamment avec l'abondance de l'utilisation des appareils mobiles, de mesurer les performances de tels équipements en se fixant des critères et métriques adoptés dans le domaine du mobile. Des paramètres reliés aux ressources matérielles ou logicielles disponibles comme la fréquence CPU, la mémoire, l'état de

la batterie, la connectivité, la mobilité, etc, seront évoqués dans une vision d'une problématique d'optimisation ou de satisfaction de contraintes.

Les recherches présentées dans cette thèse entrent dans le cadre du développement d'une méthodologie expérimentale pour la modélisation de la consommation d'énergie dans les environnements mobiles et plus spécifiquement dans le système Android.

L'objectif principal de ce travail consiste à faire le suivi de la variation énergétique d'une application particulière fonctionnant sur un appareil mobile. Dans ce manuscrit, nous proposons une méthodologie qui sert à modéliser le comportement énergétique. Nous décrivons également une méthodologie pour identifier les paramètres du modèle. À cette fin, nous avons analysé une collection de données expérimentales recueillies lors de mon tour de France en fauteuil roulant électrique. Ensuite, nous avons appliqué des outils statistiques pour obtenir les paramètres du modèle. Nous validons enfin le modèle en comparant les résultats avec d'autres données expérimentales.

1.2 Organisation de la thèse

Notre étude est repartie sur deux parties composées de sept chapitres, dont deux sont consacrés à l'aspect expérimental du travail et des études de cas. Le chapitre 1 présente la méthodologie de travail dans la thèse, il comporte les contextes étudiés, les objectifs visés et l'organisation de ce travail.

Le chapitre 2 présente une description générale des sources de consommation énergétique. Ces dernières sont subdivisées en deux grandes classes :

— Sources de consommation énergétique liées aux architectures matérielles,

— Sources de consommation coté logiciel.

Des descriptions sont faites pour les deux classes et sur les techniques de surveillance de la consommation énergétique. Enfin, une présentation de quelques axes de recherches et d'amélioration de la consommation énergétique dédié aux smartphones.

Le chapitre 3 détaille la plate-forme de travail, ainsi que le choix du système d'exploitation qui sera utilisé pour notre étude afin de mieux connaître les principales sources de consommation énergétique en détaillant les architectures en questions.

Dans la deuxième partie on a développé une méthodologie expérimentale de modélisation de la consommation d'énergie des applications.

Le chapitre 4 présente des travaux effectués sur l'optimisation de l'énergie dans les smartphones. La première partie portera sur l'étude comparative sur les différents modèles d'optimisation de la consommation énergétique en traitant le cas des machines réelles et virtuelles.

Dans le chapitre 5, nous nous intéressons à la technique de collecte des données qui a été réalisée dans le cadre d'un projet de recherche qui a consisté à faire un tour de France en fauteuil roulant électrique pendant trente-trois jours. Le but consiste à effectuer des mesures énergétiques pour avoir des mesures dans le cadre d'un cas réel.

Le chapitre 6 présente nos contributions sous forme de deux études de cas. Cette partie concerne un problème d'évaluation énergétique qui entre dans le cadre du développement des modèles mathématiques permettant la modélisation et l'évaluation du coût énergétique dans le système Android ainsi que des études sur le comportement énergétique des différents composants d'une tablette qui a été utilisée pour ces expérimentations.

La première étude de cas est consacrée pour une modélisation de la consommation d'énergie en traitant le cas du mode déconnecté, ce mode consiste à désactiver toutes les sources liées à la connexion telle que le Wi-Fi, Bluetooth et le GPS. Le mode déconnecté est basé sur des données locales pour la modélisation énergétique, seul le réseau GSM sera actif.

La deuxième étude de cas a comme objectif l'évaluation et les mesures du coût de l'énergie consommée, dans cette étude on s'est basé sur le mode connecté, les sources relatives à internet sont activées complètement (3 G/ 4 G ou Wi-Fi) ou partiellement (GPS) selon les scénarios de l'expérimentation.

Première partie

Contexte et état de l'art

Chapitre 2

Aperçu sur la consommation énergétique

2.1 Introduction

Actuellement, les nouvelles technologies connaissent une rapidité d'évolution croissante, leur démocratisation connaît une réussite historique. Par conséquent, chaque individu/citoyen peut avoir facilement les outils qui permettent l'exploitation de ces technologies. Par exemple, chacun de nous a un smartphone qui rassemble un nombre considérable de fonctionnalités, ce qui l'a rendu comme outil technologique nécessaire. En plus, l'Internet des objets (IoT, Internet of Things) rendra notre entourage plein de capteurs interconnectés entre eux afin d'assurer des services qu'on n'imaginait pas avant. A partir de ce qui précède, on peut constater que ces technologies sont caractérisées par l'utilisation intensive d'appareils Mobiles alimentés par des sources d'énergie (batterie, pile,...) limitée. Cette limite représente un défi sur lequel les chercheurs travaillent afin de faire augmenter au maximum la durée de fonctionnement du mobile par :

— Augmentation de l'autonomie des batteries,
— Minimisation de la consomation d'énergie disponible tout en assurant les bonnes fonctionnalités du mobile.

Pratiquement, les améliorations dans la technologie des batteries n'ont pas pu suivre l'évolution de la demande d'énergie imposée par les nouvelles fonctionnalités, chose qui a motivé les chercheurs à améliorer l'efficacité énergétique par la deuxième méthode, à savoir l'optimisation de la consommation énergétique. Donc, on a un grand intérêt à diminuer l'énergie consommée par ces appareils à partir de l'identification des sources gourmandes en énergie. Parmi les sources de consommation d'énergie dans le mobile on peut citer :

— Applications qui s'exécutent dans le mobile,

— Transmission sans-fil, que ça soit par 2G/3G/4G, WiFi, bluetooth ou autres.

2.2 Historique et définition

La pénétration du marché des smartphones est forte. En fin d'année 2013, 1.4 milliards de smartphones [3] sont en circulation dans le monde. Ils sont équipés de processeurs et systèmes d'exploitations plus puissants et de plusieurs interfaces réseaux. Ils remplacent différents objets que nous avions dans notre poche et permettent à leurs utilisateurs d'interagir avec les équipements existants. Mais l'utilisation des smartphones est fortement limitée par la durée de vie de leurs batterie, qui est soumise aux contraintes de taille et de poids du terminal. De plus l'énergie est l'une des principales ressources qui lorsqu'elle est épuisée, rend toutes les applications inopérantes de l'appareil mobile, y compris celles qui sont essentielles telles que les demandes d'urgence. Finalement, la question que pose un utilisateur lorsqu'il exécute une application mobile est : « Combien de temps puis-je utiliser mon téléphone en exécutant cette application ? ». L'optimisation de la consommation d'énergie de millions d'applications pour les smartphones est donc d'une importance critique. Savoir mesurer la consommation de l'énergie des terminaux mobiles devient donc d'une importance cruciale pour la compréhension et le débogage de la consommation d'énergie des applications mobiles. La consommation énergétique d'un smartphone est définie par la quantité d'énergie utilisée par le smartphone afin de faire fonctionner les services proposés.

Chaque composant matériel nécessite un coût énergétique nécessaire pour réaliser ces activités. Chaque opération sur le matériel produit une charge de travail pendant un temps donné, cette activité est induite par les exécutions des logiciels et déclenche une charge de travail sur le matériel. L'influence sur la charge du travail peut avoir lieu également via l'environnement externe par exemple l'effet des interférences réseaux qui peut produire une variation de charge de travail sur l'accès des entrées sorties. Les terminaux sont équipés de plus en plus par des batteries qui permettent de plus en plus d'autonomie, mais les ressources restent limités en terme d'énergie et de puissance.

Il est important de comprendre la différence entre les termes énergie et puissance qui sont parfois utilisés de façon interchangeable.

Puissance = Travail/Temps[Watts]

Énergie = puissance × temps[Joules]

Un composant d'un smartphone peut avoir un ou plusieurs niveaux d'états de puissance :

L'état actif :

Dans cet état les composants sont actifs et le processeur d'application est opérationnel. La consommation énergétique dans l'état actif est élevée et peut atteindre jusqu'à 2000 mW[74], lors de l'écoute de la musique et en utilisant la Wi-Fi à titre d'exemple, elle varie considérablement selon l'utilisation.

L'état inactif : Idle

Dans l'état Idle, le processeur de communication réalise un faible niveau d'activité alors que le processeur d'application est inactif. Le processeur de communication doit rester connecté au réseau en mesure de recevoir des appels ou des SMS. L'appareil est en mode de faible puissance.

D'une manière générale, la puissance consommée à l'état inactif est nettement inférieure à l'état actif et moyennement invariable dans des usages normaux.

L'état de «queue»(Tail Energy) :

Dans ce cas, aucune application n'est active et le dispositif n'est pas à l'état inactif. Les composants tels que les cartes réseaux, carte SD et GPS et la Wi-fi sur de nombreux smartphones présentent ce phénomène de l'état de « queue ». Ils restent à un état de puissance élevé pendant un certain temps après leur arrêt d'activité. Comme les plates-formes multi-core actuelles ne fournissent pas de capacités de mesure de puissance de granularité fine, McCullough et al [51] affirment que les modèles de puissance sont la première étape vers l'activation d'une gestion dynamique de l'alimentation pour la proportionnalité de l'alimentation à tous les niveaux d'un système. La modélisation de puissance envisage souvent des techniques d'apprentissage, par exemple en fonction de l'échantillonnage [10] qui assument la proportionnalité des événements du système avec la consommation d'énergie. Les mesures d'un compteur de puissance matérielle sont recueillies et utilisées ultérieurement, ainsi qu'un ensemble de valeurs estimées normalisées, dans différents modèles de régression.

L'écart entre l'énergie stockée dans la batterie et l'énergie consommée par les composants principaux augmente à chaque génération. La consommation d'énergie d'un smartphone résulte de la consommation de ses composants parce que les logiciels, toujours plus nombreux, les sollicitent. Il est donc important de savoir mesurer et comprendre comment l'énergie est consommée sur les appareils mobiles. Cela permet aux chercheurs de travailler sur différentes solutions pour réduire la consommation d'énergie afin d'améliorer l'expérience de l'utilisateur. La consommation énergétique du smartphone n'est pas non plus sans impact sur la consommation d'énergie globale.

2.3 Sources matérielles de consommation énergétique

Les smartphones modernes sont équipés de plusieurs composants matériels intégrés en eux. Les composants typiques incluent le CPU, mémoire, carte SD, Wi-Fi, téléphone, Bluetooth, GPS, appareil photo, accéléromètre, écran LCD, écrans tactiles... Il est courant pour les applications de smartphones d'utiliser plusieurs composants simultanément afin d'offrir une meilleure expérience pour l'utilisateur. Les états d'alimentation de chaque composant consomment chacun une quantité d'énergie différente.

2.3.1 Processeur

La consommation d'énergie du processeur est fortement influencée par l'utilisation et la fréquence du CPU. Il consomme environ 12,7 %[74] de la puissance. Des techniques d'optimisation de l'énergie du CPU ont fait l'objet de nombreuses recherches et des techniques matures. Comme l'ajustement dynamique de la fréquence qui ont déjà été intégrées dans les smartphones. Les résultats mesurés dépendent aussi énormément du contexte d'utilisation. En effet, plus le processeur est utilisé, plus il consomme de l'énergie, Par exemple, en 2007, des mesures sur un HTC Wizard iMate KJam[5] ont montré que le processeur consommait jusqu'à 35 % de la puissance totale.

2.3.2 Capteurs

Le tableau 2.1 [64] montre les frais généraux produits par des types populaires de capteurs dans la consommation d'énergie d'un smartphone typique : le HTC Touch Pro qui fonctionne avec Windows Mobile 6.1. Le dépassement de puissance pour chaque capteur est exprimé en pourcentage de la puissance consommée par le téléphone HTC dans trois états d'alimentation représentatifs : actif, inactif et en sommeil. Les résultats obtenus sont présentés dans le tableau 2.1

2.3.3 Modèles de puissance de processeur

La conception des modèles de puissance CPU a été régulièrement examinée par la communauté de recherche [9, 19, 51, 41, 81]. Actuellement, l'approche la plus proche de la surveillance matérielle est RAPL (Running average power limit), introduit avec l'architecture Intel «Sandy Bridge» pour faire rapport sur la consommation d'énergie de l'ensemble du/des processeur(s). cette fonctionnalité n'est pas toujours exacte vu sa non disponibilité sur d'autres architectures [19], les métriques du système d'exploitation standard (CPU, mémoire, disque ou réseau), directement calculées par le noyau ont tendance à afficher un grand taux d'erreur en raison de leurs manques de précision [41, 81]. Les modèles d'alimentation CPU sont généralement conçus

Capteurs	**État actif (1680 mW)**	**État Idle (399 mW)**	**État Sommeil (7.56 mW)**
Accéléromètre	0.03 %	0.14 %	7.4 %
Tempérpature	0.012 %	0.053 %	2.78 %
Barometre	0.1 %	0.42 %	22.2 %
Boussole	0.13 %	0.56 %	29.63 %

Table 2.1 – Surcharge des différents types de capteurs populaires sur la consommation d'énergie globale d'un HTC Téléphone Touch Pro dans 3 états d'alimentation représentatifs

en fonction de métriques brutes. Contrairement aux statistiques d'utilisation, les compteurs de performance matérielle HPC peuvent être obtenus à partir du processeur (par exemple, nombre d'instructions, erreurs de cache, cycles non interrompus). Les processeurs modernes fournissent un nombre variable de HPC, selon les architectures et les générations. Comme le montrent Bellosa [9] et Bircher [12], certains HPC sont très corrélés avec la consommation d'énergie du processeur alors que les auteurs dans [69] concluent que plusieurs compteurs de performance ne sont pas utiles car ils ne sont pas directement liés à la puissance dynamique. Néanmoins, cette corrélation dépend de l'architecture du processeur.

Le modèle de puissance du processeur calculé à l'aide de certains HPC peut ne pas être porté à différents paramètres et architectures. En revanche, le nombre de HPC qui peuvent être surveillés simultanément est limité et dépend de l'architecture sous-jacente [38], ce qui limite également la capacité de transporter un modèle d'alimentation CPU sur une architecture différente. Par conséquent, trouver une approche pour sélectionner le HPC pertinent représente une tâche fastidieuse.

Les modélisations de puissance se basent souvent sur les mesures brutes afin d'appliquer les techniques d'apprentissage par des exemples basés sur l'échantillonnage. La modélisation de puissance s'appuie souvent sur ces mesures brutes pour appliquer des techniques d'apprentissage [11] afin de corréler les métriques avec le pouvoir matériel.

Les mesures basées sur les modèles de régression, sont jusqu'à présent principalement linéaires [51].

Notre approche (voir chapitre 6) se base également sur un modèle de régression qui est quasiment linéaire.

Bertran et al. [11] modélise la consommation d'énergie d'un Intel Core2 Duo en sélectionnant 14 HPC basés sur une connaissance a priori de l'architecture sous-jacente. Leur modèle se base sur une régression linéaire multivariée. Une version modifiée est utilisée pour collecter les valeurs HPC brutes. Leur solution est évaluée

avec un taux d'erreur de 5 % sur une architechture multicore.

Dolz, M. F., Kunkel, J., Chasapis, K. Et al. [23] ont proposé une méthodologie analytique pour construire des modèles avec un nombre réduit de fonctionnalités afin d'estimer la consommation d'énergie. Le but est de construire des modèles de puissance simples en effectuant une analyse par composant (CPU, mémoire, réseau, E/S) bien qu'ils soient exécutés, les informations de tous les compteurs matériels disponibles et les mesures d'utilisation des ressources fournies par le système sont collectées. Sur la base des corrélations entre les métriques enregistrées et leur corrélation avec la puissance instantanée, la méthodologie permet d'identifier les paramètres significatifs ; Et d'attribuer des poids aux paramètres sélectionnés afin de dériver des modèles réduits.

Y. Zhai, X. Zhang, S. Eranian, L. Tang, and J. Mars. [91] ont présenté un mécanisme d'exécution qui quantifie et attribue la consommation d'énergie à des tâches individuelles à granularité fine. Plus précisément, ils ont introduit un modèle de puissance hyperthread-aware qui différencie les états lorsque les threads matériels d'un noyau sont utilisés et lorsqu'un seul thread est utilisé. En capturant ces deux états différents, ils ont été en mesure d'attribuer précisément de l'énergie à chaque processeur logique dans les serveurs modernes. Ils ont pu mener des expériences avec plusieurs charges de travail de production de Google sur un serveur Intel Sandy Bridge. Par rapport à un modèle antérieur hyperthread-inconscient, HaPPy est nettement plus précis, réduisant l'erreur de prédiction de 20,5 % à 7,5 % en moyenne et de 31,5 % à 9,4 % dans le pire des cas. Pour évaluer leur estimation de puissance, ils ont utilisé le RAPL (Running Average Power Limit) comme interface accessible sur ce serveur. Ce modèle sera détaillé dans la section 4 En résumé, les modèles de puissance CPU existants dans la littérature ne peuvent pas être tous reproduits vu les détails des événements HPC sélectionnés qui ne sont pas fournis [82] ou suffisamment documentés [91]. Comme ils sont adaptés à une architecture de processeur spécifique et vu l'ensemble limité de fonctionnalités compatibles avec l'alimentation [48].

Les différents modèles sont construits sur des charges de travail privées qui ne peuvent être réutilisées pour évaluer d'autres modèles de puissance [91].

2.3.4 Modèles de puissance de disque

Dans [56], le modèle de puissance du disque dur (HDD) proposé utilise les statistiques d'E/S comme paramètres d'entrée. La technique consiste à utiliser le nombre d'octets lus / le nombre d'octets écrits sur le disque et de multiplier ces informations brutes par la puissance consommée par opération. Les utilisateurs finaux vont récupérer la consommation d'énergie pour les opérations de lecture et d'écriture dans la documentation, aucun apprentissage n'était proposé dans ce modèle.

Cas de mémoire de stockage

Une ScanDisk de 2 Go, montre une augmentation de puissance de 21.1 mW pour les lectures et de 2.2 mW pour les écritures [68]. Sa consommation d'énergie est négligeable.

Une bonne partie des modèles de puissance proposés par l'état de l'art utilisent principalement des paramètres de base qui peuvent être extraits du système d'exploitation. La bande passante, la lecture et écriture peuvent être facilement extrait de l'état de l'art comme mesure la plus utilisée lors de la modélisation de la consommation d'énergie du disque dur. Cependant, le nombre limité d'approches existantes extraites de l'état de la technique sont limitées parce qu'ils considèrent seulement les disques durs alors que les SSD sont largement répandus.

2.3.5 Affichage et 3D

L'écran est le dispositif de sortie primaire pour l'interaction avec l'utilisateur final. La majorité de la consommation d'énergie peut être attribuée au rétroéclairage (jusqu'à 414 mW), qui comprend l'écran LCD, l'écran tactile, l'accélérateur graphique, et le rétroéclairage. Le contenu affiché à l'écran peut également affecter la consommation d'énergie de 33.1 mW avec un écran blanc et 74.2 mW [15] pour un écran noir. En d'autres termes, la consommation de l'écran représente environ 35.5 % de la puissance active du smartphone [74], elle est composée de 19,2 % [74] à 19,5 % [75] due à la luminosité de l'écran, et 16,3 % due au contenu sur l'écran. Le rétroéclairage consomme une quantité d'énergie négligeable lorsque le smartphone est à l'état inactif (7,8 mW) [15]. Le matériel graphique dédié, avec un processeur graphique (GPU), disponible sur le haut de gamme des smartphones a permis de fournir à la fois une exécution plus rapide et une plus faible consommation électrique. De plus l'incorporation de graphismes 3D dans les smartphones ont posé plusieurs défis aux concepteurs de matériels. La résolution, le niveau de détail (LOD), l'éclairage, la texture et le taux de trames sont des facteurs qui jouent un rôle essentiel dans la détermination de la qualité des graphismes 3D. Par exemple, en comparaison, sur une simulation d'un jeu en 2D (Angry Birds) et d'un jeu en 3D (Need for Speed Most Wanted) pendant 110 secondes, la puissance consommée mesurée pour le jeu 2D est de 1516 mW, tandis que pour le jeu 3D, la puissance consommée est de 2425 mW [1] Très récemment, des écrans AMOLED ont commencé à remplacer les écrans LCD standard des smartphones grand public. Par rapport à l'écran LCD, AMOLED offre une meilleure qualité d'affichage et un meilleur rendement énergétique en raison de son mécanisme unique d'éclairage.

2.3.6 Interface réseau

Les smartphones sont équipés de multiples interfaces réseaux sans fil qui leur permettent de répondre aux différentes demandes de communication et de réseau. Comme les smartphones s'en remettent de plus en plus aux connexions sans fil pour réaliser des fonctions, la consommation énergétique augmente d'une manière significative [84].

2.3.7 Wi-Fi

Le Wi-Fi est une norme de connexion sans fil de courte portée (100 mètres) et qui offre une bande passante de 11Mb/s pour la norme IEEE 802.11b ou encore 54Mb/s pour la norme IEEE 802.11 g [62].

L'utilisation du Wi-Fi produit un pic de consommation lors de la recherche de bornes Wi-Fi et lors de son association avec la borne, jusqu'à près de 5 fois plus d'énergie que le transfert de données. Cependant le coût du maintien de cette association est faible. La consommation d'énergie augmente quand le temps entre deux transferts augmente, cette augmentation se produit car l'interface Wi-Fi reste à l'état actif ce qui consomme 3 à 3,5 joules par minute [6].

Pour économiser de l'énergie, l'interface Wi-Fi fournit un mode « économie d'énergie » uniquement accessible lorsque le terminal reste connecté à la même bornes Wi-Fi. L'interface Wi-Fi peut se trouver dans 5 états : transmission, réception, inactif, sommeil et économie d'énergie.

L'état de sommeil est celui qui consomme le moins. La consommation est sensiblement réduite en mode « économie d'énergie ». L'état inactif peut être relativement élevé, car il correspond juste à l'état précédant l'état de sommeil. Les smartphones équipés d'une interface WiFi IEEE 802.11b (vitesse maximale de 11 Mbps), consomment moins d'énergie que les smartphones équipés d'une interface WiFi IEEE 802.11g (vitesse maximale de 54 Mbps). La figure 2.1 compare la variation energétique entre l'emission et la reception des données.

2.3.8 3G

La 3G permet des usages de transferts de données haut débit tels que la navigation web, la visualisation de vidéos en streaming... [47] Deux facteurs déterminent la consommation d'énergie due à l'activité du réseau dans un smartphone. Tout d'abord, l'énergie d'émission qui est proportionnelle à la longueur d'une transmission et au niveau de puissance d'émission, puis le protocole de contrôle de ressources radios qui est responsable de l'allocation de canal et de la mise à l'échelle de la puissance consommée par l'interface réseau sur la base de temporisateurs d'inactivité. Le contrôleur de ressources radios est composé de 5 états [80] :

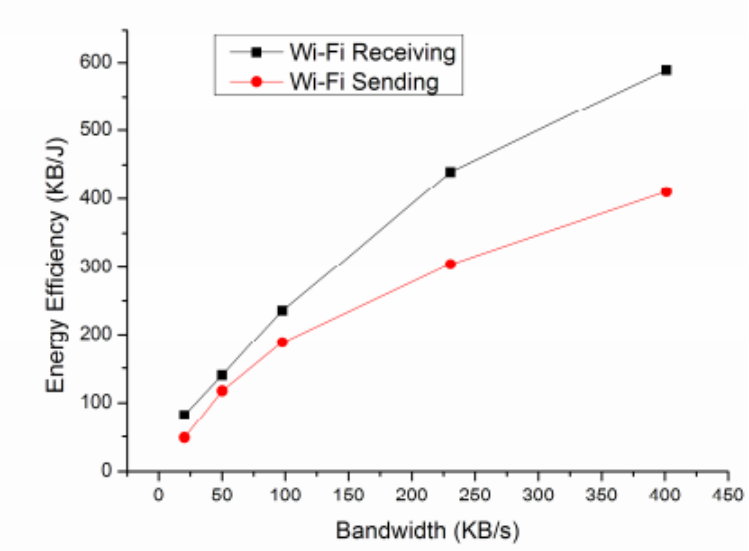

Figure 2.1 – Comparaison de la consommation d'energie entre l'emission et la reception des données pour un smartphone ZTE V880[5] doté d'une interface Wifi IEEE802.11 g

- Veille : Dans ce mode, l'interface réseau du smartphone ne communique pas avec le réseau même si elle est à l'écoute des messages de diffusion. C'est dans cet état que le smartphone consomme le moins d'énergie [92] [80].

- Cellule_DCH : Dans cet état, le smartphone est associé à un canal de transport dédié et est donc en mode émission et réception.

Il consomme le plus de ressources réseau et l'impact sur la batterie est à un niveau très élevé (800 mW) [80]. Quand il n'y a pas d'activité pendant une période de temps déterminée, l'interface réseau entre dans l'état Cellule_FACH

- Cellule_FACH : Dans cet état, le smartphone communique avec le réseau par l'intermédiaire d'un canal partagé. Quelques bits de données peuvent être transmis à un débit de données relativement faible. S'il existe un grand volume de données à émettre, l'interface réseau du smartphone passe à l'état Cellulle_DCH. La consommation est supérieure à l'état de veille. Elle est de 400 mW [80]

- Cellule_PCH : Dans cet état, aucun canal physique dédié n'est alloué à l'équipement utilisateur, de sorte qu'aucune activité d'émission ne soit possible, mais le smartphone est connu d'un canal partagé. La consommation d'énergie est faible, 30 mW [80].

Le tableau 2.2 [65] présente la consommation moyenne d'énergie radio mesurée sur un smartphone Android HTC TyTn II et un Google Nexus One pour chaque

état RRC et pendant chaque état pro-mouvement. En supposant que ces valeurs moyennes sont représentatives [92, 66]. Malgré d'autres facteurs tels que la force du signal qui a également un impact direct sur la consommation d'énergie [73].

États radios	**TyTn Carrier1**	**NexusOne Carrier1**	**ADPI [92] T-Mobile**
P(IDLE)	0	0	10
P(FACH)	460	450	460
P(DCH)	800	700	570
P(FACH→DCH)	700	550	N/A
P(IDLE→DCH)	550	530	N/A

Table 2.2 – Consommation moyenne d'énergie radio mesurée. Signification des colonnes : **État radio** : État des contrôleurs des ressources radios, **TyTn Carrier1, NexusOne Carrier1, ADPI T-Mobile** : Trois modèles differents de smartphone Android [37].

Un autre paramètre influe sur la consommation énergétique du téléphone, c'est la temporisation d'inactivité qui contrôle la transition entre états. La consommation d'énergie, à valeur de temporisation d'inactivité identique, est moindre pour la technologie WCDMA que pour la technologie CDMA2000 [47]. Le phénomène d'état de « queue » a une influence importante sur la 3 G. Par exemple, lors de la réception d'un fichier de 50 Ko, l'énergie consommée à l'état de « queue » représente plus de 60 % de l'énergie totale consommée alors qu'en début du transfert, seulement 14 % de l'énergie totale est consommée. La qualité du signal du réseau de téléphonie mobile est également une source de consommation d'énergie. En effet, plus la qualité du signal est mauvaise ou que le smartphone est éloigné d'une antenne-relais, plus la consommation d'énergie est importante [80].

2.3.9 4G

En 4G le smartphone peut avoir deux états : « RRC-connecté [1]» et « RRC-veille » Le smartphone transite de l'état « RRC-veille » vers l'état « RRC-connecté [20] [76]» lorsqu'il y a une réception ou émission de données. Après le dernier paquet transféré, la temporisation d'inactivité s'enclenche. Le smartphone retourne à l'état « RRC-veille » lorsque la temporisation d'inactivité du smartphone est écoulée [31]. Chaque changement d'état consomme de l'énergie. Le coût d'activation de la carte 4G est élevé, cela s'explique par le fait que plus la taille des transferts est importante, plus le réseau

1. `https://fr.wikipedia.org/w/index.php?title=Radio_Resource_Control&action=edit&redlink=1`

4G devient économe en énergie. Cependant, l'interface LTE consomme beaucoup d'énergie [20], à cause du phénomène de l'état de « queue » qui est particulièrement élevé (de 32.2 % à 62.2 % de la consommation totale) [92]. Le tableau 2.3 nous donne une idée précise sur la consommation énergétique d'un smartphone HTC en fonction des différents supports des réseaux.

Réseau	**Émission**	**Réception**	**Passage à l'état actif**
3G	868.98 mW	122.12 mW	817.88 mW
4G	438.39 mW	51.97 mW	1288.04 mW
WiFI	283.17 mW	137.01 mW	132.86 mW

Table 2.3 – Comparaison de puissance du mobile HTC en fonction du réseau. Signification des colonnes : **Réseau :** modes d'accès à internet, **Émission :** Paquets émis par le HTC, **Réception :** Paquets reçus par le HTC, **Passage à l'état actif :** Correspond au passage de l'état Idle à l'état actif.

2.3.10 Systèmes de localisation

Le GPS (Global Positioning System) est le système de géolocalisation le plus utilisé [7]. Il représente trois inconvénients majeurs vis-à-vis de la consommation énergétique qui se manifeste dans le temps important pour la première localisation qui peut être une vraie source de consommation.

La consommation d'un système de localisation basé sur la Wifi qui dépond fortement de la fréquence de recherche des bornes wifi, cette technique repose sur le fait de fournir l'adresse MAC du dispositif en question afin d'obtenir une correspondance avec une borne Wifi via une base de donnée. La précision de cette méthode est élevée en cas de présence de beaucoup de bornes, elle est gourmande en énergie dans le cas contraire. Le troisième type de localisation est la localisation basée sur un réseau cellulaire. À l'aide de cet identifiant unique un téléphone mobile peut retrouver la localisation, de la tour à laquelle il est connecté, via une base de données.

Le GPS est le système de localisation le plus consommateur d'énergie. Pour mettre en évidence la différence de consommation d'énergie des méthodes citées précédemment, des tests ont été réalisés à partir d'un smartphone Nokia N 95 qui effectue une localisation toutes les 30 secondes avec, au départ, une batterie chargée à 100 %. Il apparaît qu'avec le système GPS la batterie est déchargée au bout neuf heures, au bout de 40 heures avec le système Wi-Fi et au bout de 60 heures avec le système basé sur le réseau cellulaire.

2.3.11 GSM

La puissance de l'interface GSM est importante, consommant environ 45 % de la puissance globale (environ 30 mW [15]) quand l'appareil est à l'état inactif. Contrairement à la 3G, l'énergie à l'état de « queue » ne représente que 30 % de l'énergie de transfert. L'interface GSM s'expose à une petite quantité d'énergie de maintenance, entre 2 et 3 joules par minute pour garder l'interface éveillée. L'interface GSM a une consommation d'énergie relativement uniforme, malgré un débit qui est variable.

2.4 Sources logicielles de consommation énergétique

2.4.1 Les appels vocaux

Un appel vocal peut consommer en moyenne 800 mW. Une grande partie de cette énergie est due au rétroéclairage du smartphone. Une désactivation d'éclairage peut économiser jusqu'à 40 % [15] de la consommation prévue.

Le fait d'être connecté à un réseau GSM peut économiser 41 % d'énergie qu'être lié à l'UMTS(Universal Mobile Telecommunications System).

2.4.2 Le transfert des données

Le réseau GSM consomme de 40 à 70 % de moins d'énergie que le réseau 3G. En outre l'énergie de la queue est beaucoup plus présente en 3G qu'au GSM (6 secondes pour le GSM devant 12.5 secondes pour la 3G) [6]. Comparé à la 3G et au GSM, la Wifi reste plus économe énergétiquement parlant. Malheureusement la Wifi reste loin d'être répondue en comparant ses bornes au réseau GSM.

2.4.3 Multimédia

Les applications les plus populaires et plus précisément celles utilisées en diffusion (streaming) telle que YouTube, Dailymotion sont les plus gourmandes en énergie. Le téléchargement, le décodage et l'affichage sont les trois actions les plus consommatrices d'énergie [76] [37], le démarrage rapide durant lequel le serveur de streaming transmet des données au lecteur à un débit supérieur à celui du reste de la lecture. Ces données sont stockées dans la mémoire du tampon du lecteur. Pour la suite de la lecture plusieurs techniques sont possibles.

Mise en cache rapide

C'est le fait de télécharger complètement le fichier [37]. Cette méthode, utilisée par le lecteur YouTube HTML5 embarqué dans navigateurs, est la plus économe

en consommation d'énergie, les interfaces réseaux seront dans un état de basse consommation d'énergie pendant la lecture. Si le fichier n'est pas lu dans sa totalité, des données auront été téléchargées inutilement, ce qui va engendrer un gaspillage d'énergie.

Taux d'encodage

Le serveur tente d'envoyer plus de données que le lecteur ne peut en stocker dans sa mémoire tampon. Ensuite le contrôle de flux TCP autorise le serveur à transmettre le reste des données au débit auquel le lecteur les lit.

Cette technique, utilisée par le lecteur YouTube flash player haute définition embarquée dans les navigateurs [37] est la plus énergivore. En effet l'interface réseau est active plus longtemps car le débit est inférieur au débit maximum et l'intervalle de temps entre deux paquets n'est pas suffisant pour que l'interface passe à l'état inactif.

Étranglement

Le serveur transfère les données avec un débit moindre que pour le transfert en vrac, mais avec un débit plus élevé que pour le « taux d'encodage » le débit demandé par le lecteur est précisé dans la requête HTTP. Le débit pour le lecteur YouTube flash player du navigateur [37] du smartphone Samsung Galaxy S III est de 1,25 fois le débit du « taux d'encodage ». Cette technique, utilisée par le lecteur YouTube flash player standard qui est une technique embarquée dans les navigateurs qui effectuent la lecture plus rapidement que la technique « taux d'encodage ». Mais comme la technique « taux d'encodage », elle gaspille de l'énergie car le débit est inférieur au débit maximum.

2.4.4 Les jeux

Les jeux qui s'appuient sur des graphiques 3D, sont parmi les applications les plus populaires sur les smartphones. Mais, en raison du grand volume de calcul graphique sur le processeur et la GPU et de l'exigence de la qualité d'affichage, les jeux vidéo font partie des types d'applications pour smartphone consommant le plus d'énergie. La contribution du CPU à la consommation d'énergie totale peut aller jusqu'à 40 %. C'est l'étape des calculs de géométrie, où le processeur calcule les attributs de vertex et les positions au sein d'une scène 3D, qui utilise la plus grande partie de la puissance et de temps de calcul, consommant plus de 40 % de temps de calcul et plus de 35 % de la puissance totale. De plus, certains jeux vidéo nécessitent une interaction de l'utilisateur qui implique l'utilisation fréquente des différents capteurs, par exemple, les détecteurs d'inclinaison ou de déplacement des doigts sur

l'écran tactile, influant sur la consommation d'énergie. le tableau 2.4 présente les principales causes de consommation de cinq applications populaires.

Applications	**Durée(s)**	**nbthreads**	**sources energivores**
Navigateur Android	30	34	38 % pour les requêtes et réponse Http et 16 % du à un outil de traqueur d'utilisateur.
Angry Birds	28	47	45 % dus à un outil de traqueur d'utilisateur.
Fchess	33	37	50 % du à la publicité.
nytimes	41	29	65 % pour la construction de la base de données et 15 % dû à un outil de traqueur d'utilisateur.
mapquest	29	43	27 % pour le rendu du navigateur, 20 % sur le téléchargement de la carte d'utilisateur.

Table 2.4 – Les principales causes de consommation d'énergie de 5 applications populaires. Signification des colonnes : **Applications** : Application traitées, **Durée(s)** : prise en compte de la durée du test, **nbthreads** : prise en compte de nombre de thread en cours, **sources energivore** : principales sources de perte d'énergie.

2.4.5 Applications avec Géolocalisation

Les applications basées sur la localisation, telles que « trafic en temps réel », Facebook ou Myspace, sont utilisées pour être en contact permanent avec les réseaux sociaux, pour les besoins professionnels ou encore pour le divertissement. Ces applications, qui nécessitent des localisations en temps réel, consomment beaucoup d'énergie. Cette énergie est en partie consommée par des rafraîchissements inutiles de la localisation.

En effet, dans certain cas, comme lorsque le téléphone est statique ou que le signal GPS ou réseau cellulaire n'est pas disponible, les requêtes de localisation sont exécutées alors qu'elles sont infructueuses. Un autre cas est l'exécution simultanée de plusieurs applications basées sur la localisation. Celles-ci invoquent des rafraîchissements de localisation de manière indépendante au lieu de se synchroniser. Les deux paramètres qui influent sur la consommation d'énergie sont l'intervalle de temps et l'intervalle de distance qui produisent le rafraîchissement de la localisation. Les applications peuvent décider d'augmenter ces intervalles afin de réduire la consommation d'énergie, quand la batterie est faible par exemple. Ceci peut se traduire par

le paramétrage du rafraîchissement toutes les minutes ou tous les 20 mètres plutôt que toutes les 30 secondes et tous les 10 mètres.

2.5 Consommation énergétique dans les systèmes multiprocesseurs

Dans cette section on va présenter deux concepts fondamentaux pour la gestion de la consommation énergétique dans les systèmes multiprocesseurs vu que nos travaux de thèse seront effectués dans un système multiprocesseur. La réduction de la consommation d'énergie dans les systèmes multicœurs reste un problème critique, le but principal consiste à réduire la consommation, même lorsque les composants sont connectés au réseau électrique, nous devons conserver la consommation aussi faible que possible. Même avec une charge de calcul qui n'est pas très élevée, les processeurs multicœurs sont plus efficaces en énergie qu'un équivalent monocœur [88]. Auparavant, la gestion de l'alimentation sur les architectures multicœurs se résume à un principe simple : « Désactivez tout ce que vous n'utilisez pas » dans la logique de Dynamic Power Management (DPM). Dans les processeurs modernes, il est possible de changer dynamiquement la fréquence de fonctionnement pour réduire la consommation d'énergie, cette opération s'appelle Dynamic Voltage and Frequency Scaling : DVFS.

2.5.1 Gestion dynamique de l'alimentation

La quantité de puissance dissipée en raison du courant existant pendant les périodes de repos des dispositifs électroniques nommée courant de fuite est appelée puissance statique.

En 1996, Intel, HP et Microsoft équipent Toshiba et Phoenix d'une puissance statique standardisée en présentant la spécification ACPI. ACPI définit les enregistrements, les composants matériels qui devraient être disponibles et quelles informations devraient être proposées pour contrôler les états du processeur.

L'idée fondamentale derrière la gestion de l'alimentation basée sur ACPI consiste à mettre les dispositifs non utilisés dans des états de puissances inférieurs, l'ensemble du système peut être mis en état de faible puissance ou dans un état de sommeil dans la mesure de possible.

Les normes désignent deux familles d'états de processeur : états P (de performance) qui correspond à une certaine horloge, Vitesse et tension, et l'état C. Les états P sont décrits comme des états de performance ; Les états P peuvent également être appelés états de traitement, contrairement aux états C, un noyau dans un état P est en cours de traitement des instructions. À l'exception de C0, les états C correspondent aux états actifs / inactifs, il n'y a pas de traitement sur un noyau lorsqu'il se trouve dans

un état C. La norme ACPI ne définit que 4 états de puissance du CPU à partir de C0 à C3 : C0 est l'état où les transitions d'état P se produisent : le processeur est en cours de traitement. C1 Arrête le processeur. Il n'y a pas de traitement, mais la gestion du matériel du processeur détermine s'il y aura des économies d'énergie significatives. Tous les CPU compatibles ACPI doivent avoir un état C1. C2 est facultatif, également connu sous le nom de chronomètre. Alors que la plupart des CPU interrompent les signaux d'horloge en C1, la plupart des horloges sont arrêtées en C2. C3 est également connu comme dans un état de sommeil, ou complètement arrêtés.

Le résultat réel de chaque état biACPI n'est pas défini. Cela dépend de la gestion de l'alimentation et du matériel disponible sur le processeur. Les processeurs modernes n'arrêtent pas seulement à l'horloge C3, mais peuvent aussi passer à des états de sommeil plus profond C4, C5, C6 et laisser tomber la tension du CPU. La dissipation de puissance totale est la somme de la puissance dynamique et statique. L'intégrale de l'énergie dissipée au fil du temps définit la consommation d'énergie.

2.5.2 Mise à l'échelle dynamique de la tension et de la fréquence

L'augmentation de la fréquence d'un processeur implique la commutation rapide de ses transistors, et les transistors qui sont commutés plus rapidement dissipent plus de puissance. La puissance dissipée en raison de la commutation est appelée puissance dynamique. La mise à l'échelle dynamique de la tension et de la fréquence DVFS : Dynamic voltage and frequency scaling qui décrit l'utilisation de deux techniques d'économie d'énergie : mise à l'échelle dynamique de la fréquence et mise à l'échelle dynamique de la tension. L'avantage de ces mises à l'échelle de la tension et de la fréquence est la réduction de la consommation d'énergie du processeur et des périphériques connectés comme la mémoire. Une fréquence peut être réglée sur l'un des points de fonctionnement disponibles. Un point de fonctionnement est un couple tension, fréquence déterminés dans lesquel le processeur peut fonctionner. En règle générale, la tension est déterminée par la tension minimale qui peut supporter une fréquence de processeur déterminée, il n'est généralement pas logique d'avoir deux points de fonctionnement différents à la même fréquence, mais à différentes tensions. La variation de la fréquence dégrade les performances de synchronisation d'un système. Ainsi, l'étalonnage de la fréquence pour réduire la consommation d'énergie est toujours couplé à une qualité de service.

2.6 Surveillance de la consommation d'énergie

Pour bien connaître où l'énergie est dissipée, il faut être capable de créer un modèle de consommation d'énergie et/ou de concevoir une bonne mise en œuvre des logiciels. Il existe des méthodes pour mesurer et comprendre comment l'énergie est

consommée sur les smartphones : La mesure de la puissance et la modélisation de puissance.

Ces deux méthodes sont complémentaires et doivent être suffisamment calibrées pour ne pas avoir d'incidence sur la validité des résultats.

La modélisation de la puissance permet de décrire la façon dont l'énergie est consommée en utilisant des modèles mathématiques. D'un point de vue du logiciel, chaque composant matériel a plusieurs modes de fonctionnement, correspondant à des activités et des capacités de traitement différentes. Compte tenu d'un mode de fonctionnement, il est possible de dériver la consommation d'énergie de la composante matérielle et donc de construire son modèle. Il existe deux ensembles de méthodes pour établir la modélisation.

Méthodes déterministes

L'idée est d'estimer la puissance consommée par les composants matériels sur la base de leur activité. Plusieurs chercheurs ont créé des outils de profilage, qui permettent d'établir le profil de performance des applications, d'efficacité énergétique, et/ ou impact sur le réseau. Par exemple, l'outil Eprof [60], permet de tracer et comptabiliser l'énergie de chaque appel système.

Méthodes statistiques

Les méthodes statistiques ont pour but de trouver une relation entre la consommation de puissance et des variables de modèle basé sur des modèles statistiques comme la régression linéaire. L'idée générale est de trouver la relation entre les statistiques du système collectées et la consommation d'énergie. Par exemple, plusieurs outils permettent d'établir une corrélation entre la consommation d'énergie et les statistiques systèmes.

2.7 Axes d'amélioration et de recherche

2.7.1 Conception de la batterie

Les batteries lithium-ion sont très populaires pour les systèmes embarqués mobiles en raison du bon rapport énergie/poids, de leurs longues durées de vie, et leur faible auto-décharge [92]. La consommation de la batterie est plus élevée que la consommation d'énergie du smartphone seul. Il y a une différence substantielle (+27,6 % dans le pire des cas, 9,0 % en moyenne) entre la consommation de la batterie et la consommation d'énergie du smartphone à cause de pertes dans la batterie [43]. Les futures recherches d'économie d'énergie sur les batteries devraient, selon [43] être plus orientées sur la consommation de l'ensemble « batterie + smartphone » plutôt

que sur la consommation d'énergie du smartphone seul. D'autres pistes sont étudiées. Certains chercheurs, de l'Université de Stanford, utilisent les nanotechnologies pour concevoir des batteries capables de produire dix fois la quantité d'électricité des batteries lithium-ion actuelles. D'autres chercheurs tentent d'exploiter les mouvements de l'utilisateur afin de recharger la batterie du téléphone, mais ils n'en sont qu'au stade initial.

2.7.2 Systèmes d'exploitation et conception des applications

Les concepteurs d'applications sont incités à développer des logiciels pour les smartphones en prenant en compte l'efficacité énergétique.

L'obstacle principal réside dans la difficulté de déterminer l'impact des décisions de conception de logiciels sur la consommation d'énergie du système [92]. Du point de vue de l'informatique mobile, les développeurs de systèmes d'exploitation doivent être conscients de l'impact important du code sur la consommation d'énergie en raison du CPU.

2.7.3 Gestion et optimisation des interfaces réseaux

L'un des objectifs est de garder les interfaces réseaux à un niveau faible de puissance car pendant l'état inactif, il n'y a pas les mêmes exigences de performance que lorsqu'un utilisateur utilise activement le smartphone [52]. Réduire la consommation d'énergie à l'état de repos doit être une priorité pour améliorer la durée de vie de la batterie (en moyenne, le téléphone est à l'état inactif 89 % du temps, et cela représente 46,3 % de la consommation totale du système) [75].

Par conséquent, réduire les niveaux de puissance des interfaces pendant l'activité du réseau, si les modules d'entrées-sorties sont éteints, peut augmenter l'efficacité énergétique d'un smartphone. De plus, il est important de regrouper les phases d'activité d'accès au réseau autant que possible, bien qu'elles doivent rester dans un ordre séquentiel, ceci afin d'obtenir des périodes plus longues d'inactivité et de réduire le phénomène d'état de queue.

2.7.4 Techniques sur l'usage du smartphone

Il existe des paramètres dans les smartphones tels qu'Android qui limitent le nombre d'applications d'arrière-plan en cours d'exécution. Cependant, toutes les applications ne réduisent pas l'efficacité énergétique d'un smartphone. Par conséquent, au lieu de limiter le nombre d'applications, il devrait y avoir une information sur les applications et les types d'accès réseaux moins consommateurs en énergie. Ainsi, l'utilisateur pourrait agir en conséquence [52]. De plus les téléphones mobiles fournissent des interfaces utilisateur permettant de faire des compromis entre la durée de vie de la batterie et la facilité d'utilisation, tels que la luminosité de l'écran,

mais la plupart des utilisateurs n'utilisent pas les niveaux de luminosité multiples. Un réglage automatique de la luminosité doit être inclus dans les smartphones. Les interfaces utilisateurs permettent également aux utilisateurs de désactiver les composants énergivores du système, tels que les interfaces Bluetooth et Wi-Fi, afin d'économiser l'énergie.

2.8 Conclusion

Dans cette partie on a décrit d'une manière générale des sources de consommation énergétique, ensuite on les a classifiées en deux grandes classes composées des sources de consommation énergétique liées aux architectures matérielles, et les sources de consommation coté logiciel.

Le chapitre suivant détaillera le choix du système d'exploitation et la plate-forme de travail. L'objectif est de mieux connaître les principales sources de consommation énergétique en détaillant les architectures en questions.

Chapitre 3

Environnement d'expérimentation

3.1 Introduction

Un système d'exploitation mobile est un système d'exploitation conçu pour fonctionner sur un appareil mobile, une bonne partie de ce type de système d'exploitation est focalisée sur la gestion de la connectivité sans fil et sur la gestion des différents types d'interfaces.

Il existe plusieurs systèmes d'exploitations dédiés aux smartphones, le tableau 3.1 nous donne une idée sur les plates formes les plus dans le domaine du mobile.

Système d'exploitation	**Créateur**
Symbian OS	Nokia
IOS	Apple
BlackBerry OS	RIM
Windows Phone	Microsoft, Palm webOS
Android	Google
Bada	Samsung

Table 3.1 – Systèmes d'exploitation dédiés aux smartphones

3.2 Choix du système d'exploitation

La totalité de nos travaux est effectuée avec le système d'exploitation Android pour sa dominance sur la grande part du marché des smartphones et vu sa distribution

open source.

Android est basé sur un kernel linux, au-dessus du kernel il y a « l'hardware abstraction layer » qui permet de séparer la plateforme logique du matériel. Au-dessus de cette couche d'abstraction on retrouve les bibliothèques (library) C/C++ utilisées par un certain nombre de composants du système Android. À coté des bibliothèques on retrouve l'Android Runtime, cette couche contient les bibliothèque cœurs du Framework ainsi que la machine virtuelle exécutant les applications. Au-dessus la couche « Android Runtime » et des bibliothèques cœurs on retrouve le Framework permettant au développeur de créer des applications. Enfin au-dessus du Framework il y a les applications. La figure 3.1 détaille le système Android.

Android est un système d'exploitation ouvert dont le code source est librement accessible ce qui permet à n'importe quel fabricant de l'intégrer dans son système gratuitement. Ce système est basé sur cinq couches distinctes :

— Le noyau Linux avec les pilotes ;
— Des bibliothèques logicielles telles que WebKit, OpenGL, SQLite ou FreeType ;
— Une machine virtuelle et des bibliothèques permettant d'exécuter des programmes prévus pour la plate-forme Java ;
— Un framework - kit de développement d'applications ;
— Un lot d'applications standard parmi lesquelles il y a un environnement de bureau, un carnet d'adresses, un navigateur web et un téléphone.

Ce système, assez nouveau, a eu quatorze versions chacune portant un nom et un code spécifique. La dernière version se nomme Android 8.0 (Oreo). Sa vente est en progression permanente par rapport aux autres types de smartphones utilisant d'autres systèmes d'exploitation. Le graphique 3.2 indique l'augmentation des taux d'achat des différents smartphones avec les OS les plus répandus.

3.2.1 Composants principaux d'Android

Le principe d'Android s'appui sur des classes qui sont en quelque sorte les bases élémentaires sur lesquelles les futures applications reposent, elles sont indispensables pour la construction de toute application, dans cette partie on va décrire les principales classes du système Android.

Intents : Les intents sont des objets qui permettent de faire passer des messages contenant de l'information entre composants principaux. Cette notion peut être vue comme une demande de démarrage d'un autre composant, d'une action à effectuer.

Vues (Views) : La vue est le principal composant qui s'occupe de gérer les actions utilisateurs (appui sur l'écran, sur le clavier, etc.), les vues sont les composants de base de l'interface graphique. Elles permettent de réaliser l'interface utilisateur,

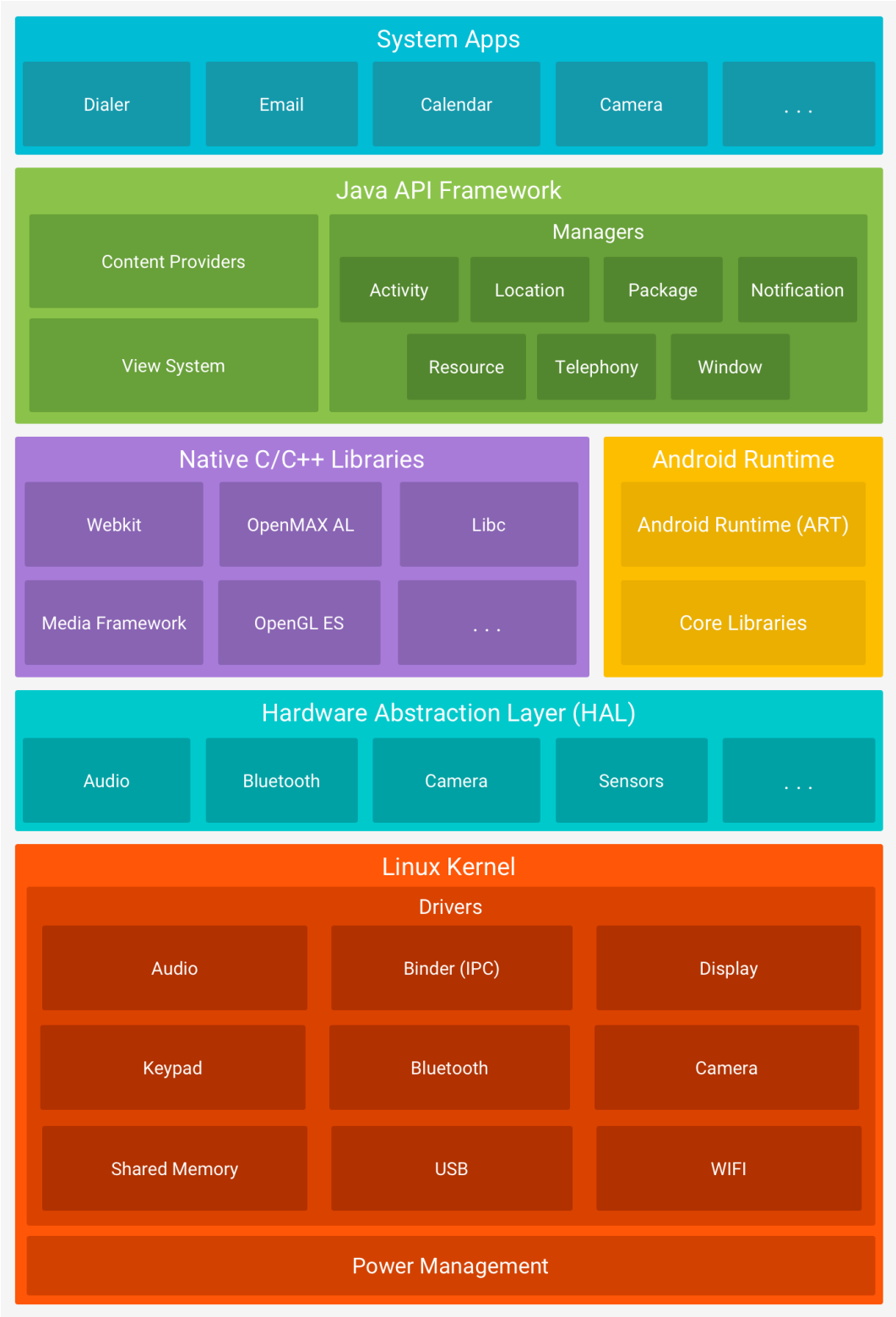

Figure 3.1 – Architecture du système d'exploitation Android (source : `https://developer.android.com/guide/platform/index.html`)

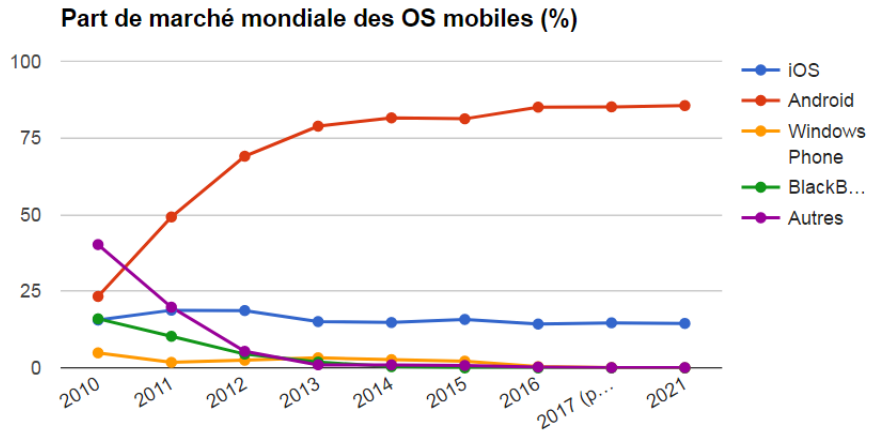

Figure 3.2 – Part de marché mondial réel des OS mobile ainsi que les prévisions (source : `http://rouhaud.over-blog.com/2016/10/apple-et-le-nfc.html`)

Activités : Une activité correspond à la fenêtre/écran qui sera affiché (e) à l'utilisateur, elle gère également les fonctionnalités relatives à l'appui sur la touche menu etc. Son concept repose sur la notion d'interaction utilisateur.

Services : Un service est un programme qui fait tourner une tâche de fond qui n'a pas d'interface graphique (exemple d'un lecteur Mp3 qui ne nécessite pas une interface graphique en permanence Le but est de laisser la possibilité aux autres applications de travailler. Les services constituent un point très fort du système Android vu que cette fonctionnalité n'est pas réalisable sur l'Ios.

ContentProvider : Les ContentProvider sont des gestionnaires de données qui permettent de partager les informations entre applications. Leur rôle consiste à stocker des informations relatives à une/ des application (s) pour qu'elles soient exploitées par d'autres applications (sans avoir la nécessité de repasser par les applications de base).

BroadcastReceiver : Un BroadcastReceiver est une application qui est à l'écoute des autres applications. Il consiste à répondre aux intents qui lui sont adressés. Son rôle unique est d'être à l'écoute des intents envoyés par d'autres composants applicatifs.

3.2.2 Machine virtuelle Android

Dalvik

Dalvik est une machine virtuelle destinée aux téléphones mobiles et tablettes tactiles, qui est incorporée dans le système d'exploitation Android 1, 2. Son rôle principal est de permettre l'exécution simultanée de plusieurs applications sur un appareil de faible capacité (peu d'espace mémoire et peu de puissance de calcul). Dalvik est un des composants clé d'Android2. Dalvik est remplacé par ART à partir de la version 5 d'Android sortie en novembre 2014.

ART : Android Runtime

Contrairement à Dalvik, ART utilise la compilation anticipée, en compilant l'application à son installation, sans besoin ultérieur d'interprétation. ART permet ainsi d'augmenter les performances, donc d'augmenter la durée de vie de la batterie. De plus, le ramasse-miettes et les allocations mémoires sont plus efficaces, avec plus d'options de débogage ou de profilage des applications.

Pour conserver la rétrocompatibilité ascendante, ART utilise des fichiers APK ou .dex, ainsi que du bytecode Dalvik. La figure 3.3 visualise la différence de performances des processeurs d'un Nexus 5.

3.3 Architecture materielle

3.3.1 Interaction entre les composants du système Android

Plusieurs facteurs qu'ils soient internes tel que les activités relatives au matériel, l'implémentation des logiciels ; ou externes comme l'environnement extérieur, le comportement utilisateur influencent la consommation énergétique dans un smartphone, donc la durée d'un cycle de vie d'une batterie. Afin d'optimiser la consommation énergétique, le choix des technologies ainsi que l'architecture sont primordiaux. La figure 3.4 nous montre les différentes interactions qui peuvent avoir un effet direct sur la consommation énergétique d'un smartphone.

3.3.2 La technologie big.LITTLE

ARM big.LITTLE est une architecture informatique hétérogène développée par ARM Holdings, couplant des cœurs de processeurs relativement plus économes en batterie et plus lents (LITTLE) avec des processeurs plus puissants et gourmands en énergie (big). D'une manière générale, un seul « côté » ou l'autre sera actif à la fois, mais puisque tous les cœurs ont accès aux mêmes régions de mémoire, les

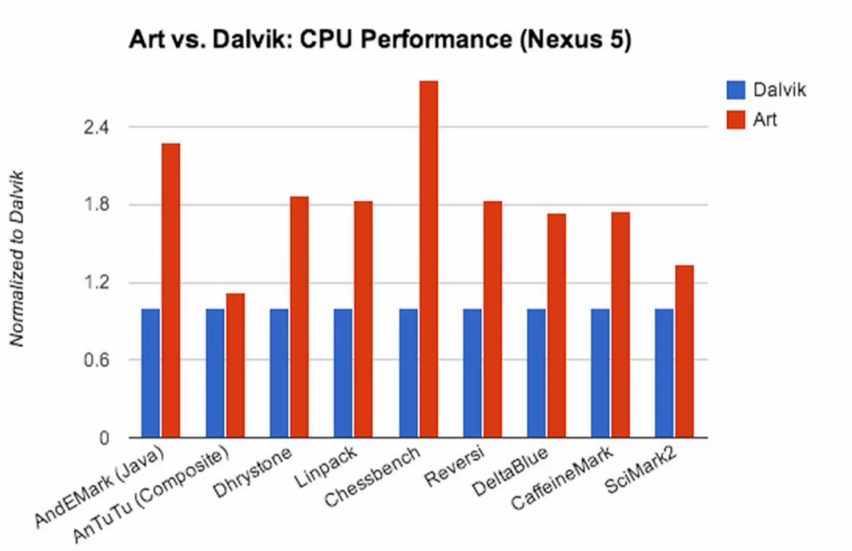

Figure 3.3 – Performance CPU ART vs Dalvik pour un Nexus 5 (source : `https://www.anandtech.com/show/8231/a-closer-look-at-android-runtime-art-in-android-l`)

charges de travail peuvent être permutées entre les cœurs big et LITTLE à la volée[1]. L'intention est de créer un processeur multi-cœurs qui peut mieux s'adapter aux besoins informatiques dynamiques et qui utilise moins de puissance que la mise à l'échelle de l'horloge seule. Le matériel de marketing d'ARM promet jusqu'à 75 % d'économie d'énergie pour certaines activités[2]. Ces modèles sont détaillés dans la figure 3.5.

3.3.3 Fonctionnement de big.LITTLE

Dans les smartphones et les tablettes actuelles, la mise à l'échelle dynamique de la tension et de la fréquence (DVFS) sont utilisées pour s'adapter aux fluctuations instantanées des performances requises. Les modes de migration de big.Little étendent ce concept en permettant une transition à des cœurs de processeur « big » au-dessus du point de fonctionnement de DVFS le plus élevé des petits cœurs. La migration dure environ 30 microsecondes. En revanche, le conducteur DVFS évalue la performance de l'OS et les noyaux individuels typiquement toutes les 50 millisecondes. Il faut environ 100 microsecondes pour modifier la tension et la fréquence. Parce que le temps

1. `http://www.ubergizmo.com/2013/01/what-is-arm-big-little/`
2. `https://developer.arm.com/technologies/big-little`

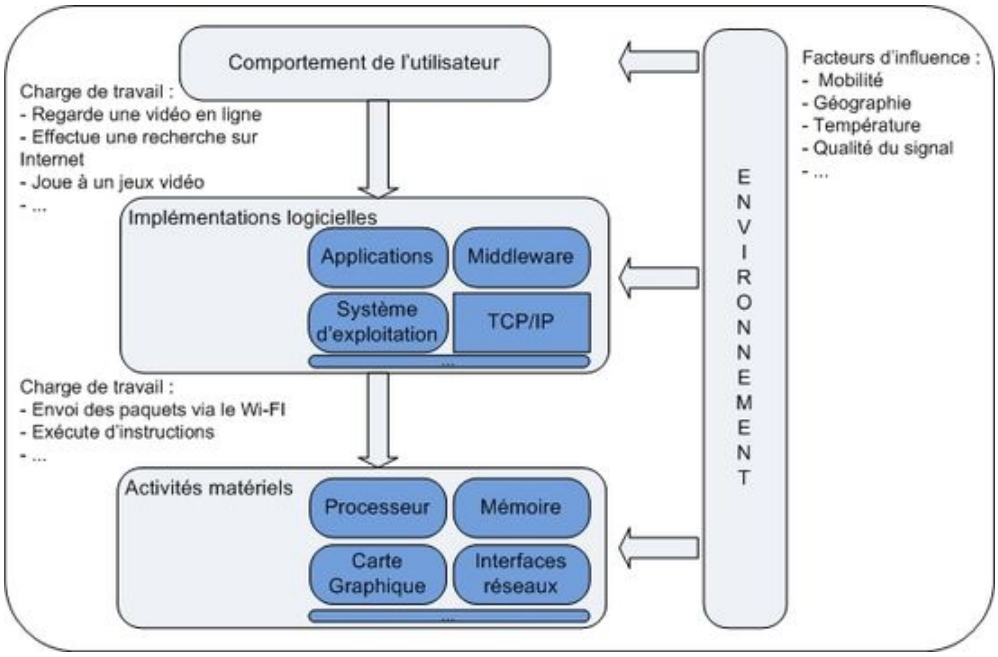

Figure 3.4 – Facteurs d'influence énergétique [85]

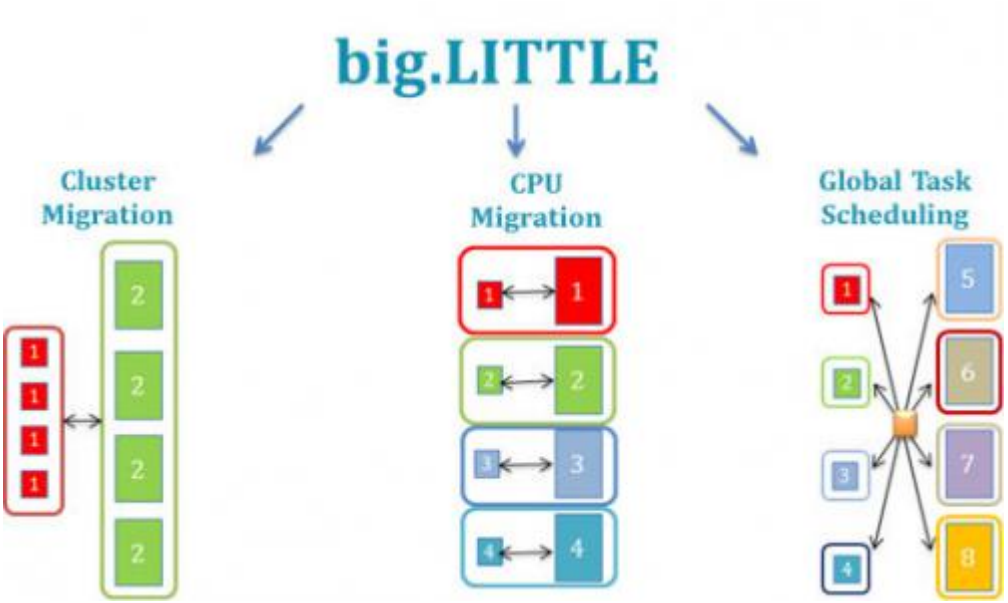

Figure 3.5 – Principaux modèles d'utilisation de l'architecture big.Little. Signification des modèles : **Cluster migration** : les CPU sont regroupés par type (big/Little) et la migration est effectuée entre les 2 blocs, **CPU Migration** : Chaque groupe de CPU (1 big et 1 Little) est regroupé à part, la migration ne peut se faire que dans le même groupe, **Global task scheduling** : Permet l'utilisation de tous les noyaux physiques en même temps, C'est le modèle d'utilisation le plus puissant de l'architecture big.Little. (Source : `https://community.arm.com/processors/b/blog/posts/ten-things-to-know-about-big-little`)

pris pour migrer un CPU ou un cluster est plus court que le temps de changement DVFS et un ordre de grandeur plus court que la période d'évaluation du système d'exploitation pour les changements DVFS, les transitions big.Little permettront aux processeurs d'exécuter des points de fonctionnement inférieurs, plus souvent et, en outre, être complètement invisible à l'utilisateur (voir figure 3.6).

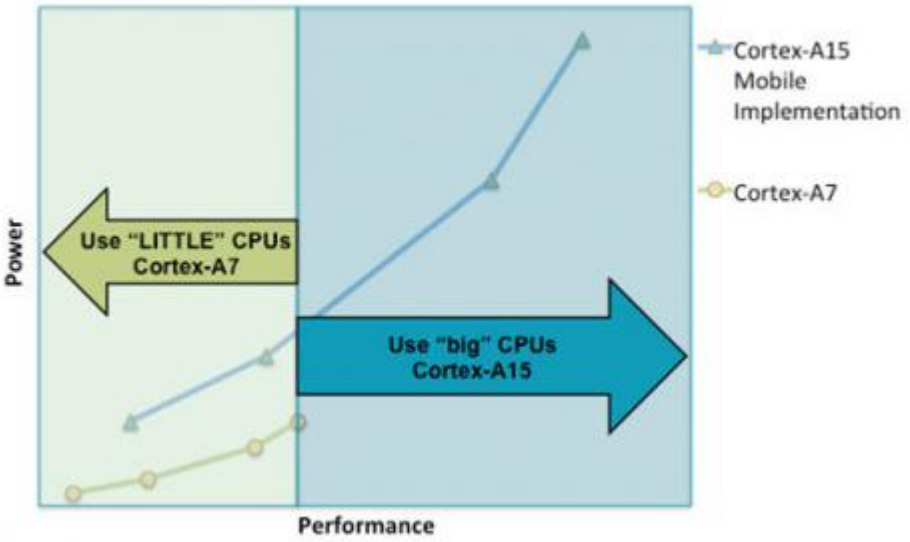

Figure 3.6 – Rapport énergie-performance. Signification : La migration vers le CPU de plus forte puissance (big) s'active en cas de demande importante en puissance de calcul. Cela permet d'économiser davantage d'énergie(source : `https://community.arm.com/processors/b/blog/posts/ten-things-to-know-about-big-little`)

3.3.4 SnapDragon

Snapdragon est une architecture de processeur fournie par Qualcomm. Scorpion, le processeur snapdragon original avait plusieurs caractéristiques similaires à l'ARM Cortex A8 basé sur le jeu d'instructions ARMv7, mais avec un avantage de performances par l'utilisation d'operations SIMD comme illustré dans la figure 3.7.

La totalité des mesures expérimentales effectuées dans notre travail sont réalisées avec une tablette dotée d'un processeur SnapDragon 800, ce dernier est conçu pour permettre un accès rapide aux applications tout en permettant une durée de vie de batterie importante. Le processeur Snapdragon 800 offre une connexion basée sur des

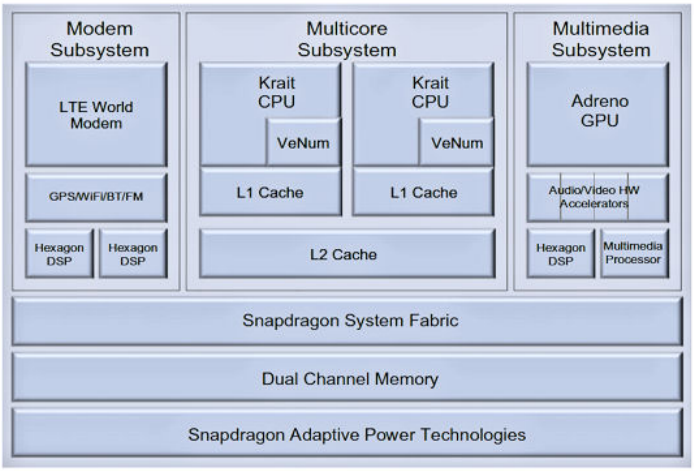

Figure 3.7 – Architecture SnapDragon, dotée d'un module suplémentaire(**SnapDragon Adapptive Power Technologie**) dédié à l'adaptaion de la consommation du système. (source : `https://www.anandtech.com/show/4940/qualcomm-new-snapdragon-s4-msm8960-krait-architecture`)

techniques avancées telles que la vidéo Ultra HD et l'imagerie avancée. La figure 3.8 présente l'architecture détaillée du processeur Qualcomm SnapDragon 800

Figure 3.8 – Modules intégrés dans l'architecture Qualcomm SnapDragon 800 (source : `https://techspirited.com/snapdragon800-processor-specs`)

3.4 Outils de mesure

L'estimation de la puissance logicielle des différentes composantes d'un dispositif mobile est une préoccupation centrale pour l'efficacité énergétique. Au cours de ces dernières années une dizaine de modèles de puissances ont été spécialement conçus pour faire face à la complexité croissante des architectures des CPU modernes, en revanche la quasi-totalité des modèles d'alimentation CPU dépendent d'une approche approfondie sur des architectures ciblées menant ainsi à la conception d'un matériel spécifique.

L'état de l'art dans ce domaine est confronté à plusieurs limitations, par exemple une architecture simplifiée du CPU, les métriques [13] sélectionnées et l'indisponibilité des benchmarks [91] et la diversité des charges de travail testées [11].

Ces limitations empêchent le bon déploiement des modèles de puissance et limitent ainsi la surveillance de la puissance du logiciel. La surveillance de l'alimentation est généralement réalisée par des équipements de mesure matérielle comme les compteurs de puissance ou des circuits intégrés spécialisés. Cette solution n'est pas totalement appropriée à une plus grande échelle, donc il faut avoir un niveau plus fin pour optimiser l'énergie, ce qui nécessite des investissements matériels coûteux.

Plusieurs solutions ont été proposées au cours des dernières années pour estimer la consommation d'énergie, au niveau du logiciel [26, 22, 79], et même au niveau du code [16, 57, 58]. Malgré ça la plupart des solutions existantes nécessitent des investissements matériels [8, 28, 46, 67]

3.5 Conclusion

L'un des principaux verrous technologiques reste la consommation énergétique des appareils avec des ressources d'énergie à capacités limitées. Il ne suffit pas seulement d'augmenter l'autonomie à l'aide de batteries toujours plus performantes, mais il faudrait une bonne gestion de l'énergie qui est confrontée à plusieurs limitations.

Dans le cas des environnements virtualisés, nous pouvons trouver des méthodes qui proposent des solutions grossières, l'inconvénient de ces techniques réside dans le fait que les machines virtuelles sont considérées comme des boîtes noires. Pourtant, les systèmes basés sur les machines virtuelles sont couramment utilisés pour héberger les applications multiples, pour économiser les coûts, et pour une meilleure utilisation de l'énergie en partageant des ressources communes. La partie suivante traite le problème de l'optimisation de la consommation énergétique à travers des modélisations basées sur des méthodes experimentales.

Deuxième partie

Optimisation de la consommation énergétique des applications : modélisation et méthodologie expérimentale

Chapitre 4

Optimisation de la consommation d'énergie dans les smartphones

4.1 Introduction

L'optimisation joue un rôle primordial en recherche opérationnelle, en informatique et dans différents domaines de l'industrie et de l'ingénierie. La difficulté de la résolution des problèmes d'optimisation, d'une part, et le nombre croissant des applications pratiques qui peuvent être modélisées sous la forme d'un problème d'optimisation, d'autre part, justifient l'importance de ce domaine. Les problèmes d'optimisation sont généralement difficiles à résoudre. Malgré le succès des smartphones sur le marché, leur utilité a été et restera fortement limitée par la durée de vie ou l'autonomie de leurs batteries [33]. C'est pourquoi l'optimisation de la consommation d'énergie des applications des smartphones est d'une importance critique. Cependant, les applications mobiles développées jusqu'ici ont été conçues sans prendre en compte le facteur énergétique [36]. La consommation énergétique d'un smartphone est définie par la quantité d'énergie utilisée par celui-ci afin de faire fonctionner ses services. Comme l'amélioration de la capacité de la batterie est très modérée par rapport à l'augmentation des nouveaux services et matériels [63], le contrôle du contexte et de la consommation d'énergie du smartphone devient un défi [85].

4.2 Enquête sur l'adaptation de la consommation énergétique d'un Smartphone

La consommation d'énergie d'un smartphone résulte de la consommation de ses composants qui participent à l'exécution des applications mises en œuvre. Il est donc important de savoir mesurer et comprendre comment l'énergie est consommée sur ces appareils mobiles.

Les terminaux sont alimentés par la batterie pour permettre le plus haut degré de liberté à l'utilisateur, mais cela limite les ressources en termes d'énergie et de puissance. Il est essentiel de comprendre la différence entre ces deux termes qui sont parfois utilisés de façon interchangeable [63]. Les travaux de recherche présentés dans cette partie s'inscrivent dans le cadre du développement des modèles qui permettent la modélisation et l'évaluation du coût énergétique dans les environnements mobiles et plus précisément dans les systèmes Android ainsi que des études sur le comportement énergétique des différents composants des smartphones. La section 4.2.1 portera sur l'évaluation et l'importance de la mesure des coûts énergétiques. Dans ce sens, nous allons mener notre enquête sur les modèles suivants : TailEnder, Cinder, Eprof, RAPL et AppScope. On procédera à l'analyse et à la comparaison des modèles cités auparavant, on finira par la synthèse des différentes méthodes.

4.2.1 Présentation du problème de modélisation

L'énergie sur les téléphones mobiles est une ressource précieuseà ; comme la totalité des téléphones mobiles sont équipés de multiples technologies sans fil tel que 3G, GSM et Wi-Fi, il est important de comprendre leurs caractéristiques relatives à la consommation d'énergie.

Importance de la précision d'évaluation des coûts énergétiques

Au moment du fonctionnement de chaque service dans un smartphone il existe une consommation énergétique relative à une quantité d'énergie dissipée. Il est donc important de savoir mesurer et comprendre comment l'énergie est consommée sur les appareils mobiles pour aboutir à des solutions qui permettent la réduction de la consommation d'énergie afin d'améliorer leurs performances et l'expérience utilisateur.

Les smartphones modernes sont équipés d'une grande variété de circuits intégrés. Ces derniers incluent entre autres le CPU [89], la mémoire, la carte SD, Wi-Fi, le téléphone, Bluetooth, GPS, l'appareil photo, l'accéléromètre, la boussole numérique, l'écran LCD [90] ou tactile, le microphone, le haut-parleur... Il est courant pour les applications des smartphones d'utiliser plusieurs composants simultanément afin d'offrir une expérience utilisateur plus riche.

La connaissance de la consommation énergétique des différents composants du smartphone constitue l'axe pivot pour connaître les parties essentielles qui consomment le plus d'énergie ; ces parties seront traitées en priorité pour optimiser le coût énergétique. Cette étape est très importante car toute erreur de classification empêchera de trouver une solution optimale.

4.2.2 Mesure des coûts énergétiques

Dans cette partie, on va mener notre enquête sur les techniques suivantes : TailEnder, Cinder, Eprof, RAPL et AppScope qui permettent de montrer les principales sources de consommation énergétique dans un but d'optimisation de l'énergie réservée pour les différents périphériques. Nous décrirons brièvement le principe de mise en œuvre et les limites de chacune des méthodes citées.

TailEnder

En 2009,Niranjan Balasubramanian, Aruna Balasubramanian et Arun Venkataramani ont conçu TailEnder [6] pour la mesure des caractéristiques de la consommation énergétique des différents équipements des smartphones.

TailEnder a pour but la minimisation de la consommation d'énergie pour les applications qui peuvent tolérer un léger retard tel que les E-mails. En particulier, l'étude des caractéristiques de consommation d'énergie de la 3G révèle des implications importantes et non intuitives pour la conception de l'application efficace de l'énergie.

Des travaux antérieurs tel que [47, 86, 17] ont étudié l'impact des différentes techniques d'économie d'énergie dans les réseaux 3G en utilisant des modèles analytiques. Sur la base de cette constatation, un modèle simple de la consommation d'énergie a été élaboré dans le but d'identifier les possibilités de réduction de la consommation d'énergie de l'activité réseau induite par les applications réseaux ordinaires pour chacune des trois technologies citées précédemment. Les expériences montrent que le protocole TailEnder donne la possibilité de télécharger 60 % de plus et de faire 50 % de plus de recherche web par comparaison à la même consommation avec une politique par défaut [6].TailEnder est évalué en utilisant une simulation axée sur le modèle d'expériences sur le téléphone, les expériences ont été réalisées en se basant sur Nokia N95. Le but de l'évaluation consiste à quantifier la réduction de l'utilisation de l'énergie lors de l'utilisation de TailEnder pour différentes applications, par rapport à un protocole par défaut. Pour montrer l'applicabilité générale de TailEnder, ses performances sont évaluées sur les e-mails, des nouvelles alimentations et recherche Web [6]. La recherche Web est une application interactive, mais peut bénéficier de préchargement. L'impact de TailEnder pour la minimisation de l'énergie dépend en grande partie du trafic de l'application et du comportement des utilisateurs. Si un utilisateur reçoit un e-mail toutes les heures, ou si de nouvelles alimentations sont

mises à jour une fois par heure, la probabilité que TailEnder apporte des avantages de gestion de l'énergie, est minime.

Cinder

En 2011 A. Roy, S. M. Rumble, R. Stutsman, P. Levis, D. Mazieres, et N. Zeldovich montrent dans [72] que la méthode Cinder permet de modéliser la consommation d'énergie et qu'elle est apte au partitionnement des applications à des bornes d'énergie, même avec des politiques complexes. Cinder est conçu pour les téléphones et appareils mobiles, il permet aux utilisateurs et aux applications de commander et gérer les ressources limitées de l'appareil tel que l'énergie. Ce mécanisme, contrairement aux approches antérieures introduit deux nouvelles abstractions qui ont pour but l'écoute avec précision des principaux responsables de la consommation des ressources. La tâche prioritaire consiste à visualiser le problème puis attribuer la consommation aux applications directrices. [89, 90, 27, 27, 26]. Le principe de Cinder [72] consiste à avoir des informations sur trois mécanismes : l'isolation, la subdivision et la délégation. L'isolation vis-à-vis du processeur assure que les processus ne vont pas mourir de faim, ce qui nous intéresse dans cette partie est l'isolation par rapport à la consommation. Par exemple, deux processus P1 et P2 qui consomment chacun une telle quantité d'énergie ne doivent pas avoir accès chacun aux ressources (même via les fils ou les tubes de communication) L'énergie réservée à l'appel d'urgence doit être isolée de manière en aucun cas un programme puisse l'utiliser [72]. Cinder est basé sur le noyau HISTAR de type ExoKernel. Cinder utilise le concept des réserves et des robinets. Une réserve décrit le droit d'une quantité donfnée d'une ressource telle que l'énergie qui a pour but l'empêchement d'exécution des applications qui ne disposent pas de ressources suffisantes. Cette méthode est efficace pour étrangler la consommation d'énergie. En théorie, les réserves sont suffisantes pour contrôler le système en terme d'utilisation des ressources, mais cette approche peut être susceptible d'être inefficace en pratique. La notion du robinet peut remédier à ce problème en faisant le transfert de ressources qui pourraient être mises en œuvre par des fils à usage spécial qui se déplacent de manière explicite entre les ressources et les réserves. La première réserve est le premier plan de réserve qui est relié à la batterie par l'intermédiaire d'un robinet à débit élevé. La seconde est une réserve à faible taux connecté à la batterie via un robinet à faible débit. La Figure 4.1 [72] explique ce mécanisme en donnant l'exemple d'un navigateur web configuré pour fonctionner pendant au moins six heures sur une batterie 15kJ. Le navigateur assure en outre que son Plug-in ne peut pas utiliser plus de 10 % de son énergie.0.1x (x étant l'énergie totale), les robinets proportionnels empêchent le navigateur et le Plug-in de thésaurisation de l'énergie [72]. Avec Cinder l'utilisateur peut faire un quota pour la navigation Web (réserve et robinet sur la page en question et le Plug-in) dans le but d'assurer l'exécution du Plug-in tout en étant sûr qu'il ne mourra pas de faim. Cinder divise la réserve de chaque application

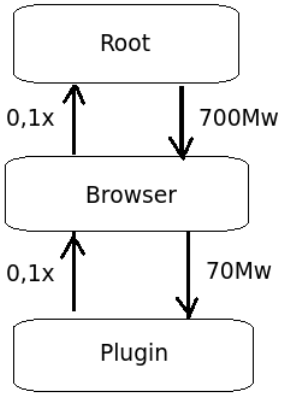

Figure 4.1 – Étranglement énergétique 1 : Notion de réserve et de robinet **Signification**, Le système empêche les Plug-in d'utiliser plus de 10 % de l'énergie totale dédiée au navigateur.

en deux autres sous réserves via les robinets. La première correspond au premier plan, elle est reliée à la batterie par l'intermédiaire d'un robinet à débit élevé. La seconde est une réserve à faible taux connecté à la batterie via un robinet à faible débit [72] Lorsque l'appel d'urgence est lancé, le dispositif ferme le robinet sur toutes les autres applications en cours (0mW), ces dernières n'auront pas d'accès aux ressources tant que l'appel d'urgence soit présent. La figure 4.2 montre comment conserver une valeur d'énergie (250 J) mise à côté pour un appel d'urgence :

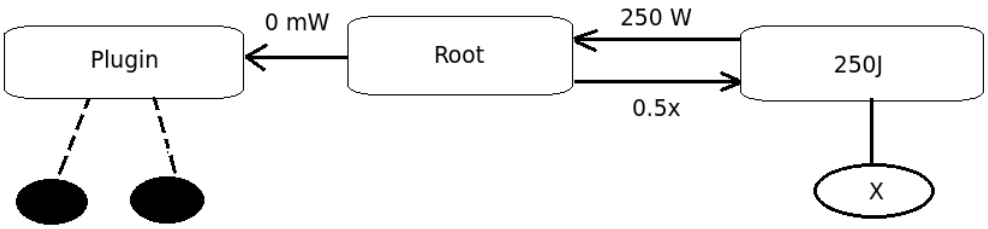

Figure 4.2 – Étranglement énergétique 2 : Notion de réserve et de robinet **Signification** : Pour augmenter la sécurité, le modèle permet de réserver 250J pour l'appel d'urgence. En cas de pénurie énergétique en bloquant l'accès à toutes les autres applications en cours.

Lorsque l'appareil détecte un appel d'urgence, il définit le robinet d'alimentation des applications normales à 0mW. Le robinet proportionnel empêche la tâche en

question (tâche X) de la capitalisation de l'énergie pendant le fonctionnement normal. Cinder peut contrôler le pouvoir par la subdivision, la délégation et l'isolement, il offre également une visibilité sur l'énergie et la puissance d'un système en cours d'exécution. En outre, en examinant comment les applications utilisent le chemin de données de téléphone, nous évaluons si Cinder peut améliorer l'efficacité énergétique d'un système de gestion des dispositifs complexes avec la consommation d'énergie non-linéaire.

Nous relevons que Cinder simplifie l'utilisation des politiques qui permettent une utilisation efficace des périphériques coûteux. Cependant, il présente certaines limites :

— Cinder est conçu dans un cadre professionnel (il utilise les unités de mesure comme le Watt et le Joule) ce qui est inadéquat pour l'utilisateur final/développeur d'application qui n'ont pas l'habilité d'interpréter ces valeurs (durée ou quantités restantes à télécharger ...)
— Cinder et Android sont développés sous Linux. Comme les fonctionnalités et applications Android se basent sur la machine virtuelle Dalvik, Cinder doit aussi être adapté avec Dalvik pour qu'il puisse supporter toutes les applications Android.
— Cinder n'est pas en mesure de donner une partie de la capacité à consommer via les ressources à un rythme élevé, (tâches à réaliser par les utilisateurs et les applications), ce qui entraînera un octroi non forcément opportun des ressources.
— Le contrôle de la concurrence et la sécurité d'accès doit être mieux géré(le stockage des privilèges dans les robinets doit être plus sécurisé, et doit permettre aux robinets de transférer des ressources entre les réserves).

Eprof

En 2012 Abhinav Pathak, Y. Charlie Hu et Ming Zhang ont modélisé EPROF [60] qui est un « Profiler » d'énergie fine pour les applications des smartphones. Il aide directement un développeur d'application dans le cycle d'optimisation énergétique de l'application. Basé sur le modèle de puissance FSM [36], EPROF a la capacité d'analyser l'état de la demande d'énergie asynchrone, la modélisation des caractéristiques énergétiques, l'état de la queue des composants avec une granularité de niveau routine. L'objectif principal de cette solution consiste à déterminer où est passée l'énergie à l'intérieur des applications ; l'identification précise de l'entité responsable d'une telle consommation à un tel instant est difficile à avoir, cela est dû au comportement asynchrone de l'énergie (la puissance instantanée tirée d'un composant qui ne peut être liée au courant) [60].

Plusieurs composants, tels que le disque, Wi-Fi, 3G, GPS, dans les smartphones, présentent le comportement de la queue de puissance [61], où les activités en une

seule entité (une routine par exemple) peuvent déclencher un composant à entrer dans un état de forte puissance et rester dans cet état d'énergie au-delà de la fin de la routine. Le point pivot de cette approche est l'évaluation de l'énergie consommée par les entrées/ sorties (E/S), et de l'énergie de la queue (Tail energy). Les E/S et la queue sont jugées comme étant les grands consommateurs d'énergie. Un tel comportement est asynchrone. Plusieurs éléments (GPS, Wi-Fi, Carte SD, 3G) ont des états d'alimentation de la queue [61, 6]

Les systèmes d'exploitation des smartphones, peuvent passer à un état inactif après un certain délai de repos, mais les applications doivent rester éveillées [50].

Les Wakelocks (des processus qui peuvent forcer le CPU à rester activé) présentent un autre exemple de l'aspect asynchrone tenue en puissance des smartphones. L'énergie consommée (à cause du réveil des verrous) est particulièrement importante car elle peut aider à traquer les bogues Wakelocks [59] (par exemple, Facebook bugs [39], E Mail bug, Android [59, 24], et Lieu d'écoute bugs [54]).

Le programme peut être optimisé par restructuration du code dans les processus au niveau des thread et sous programmes. Pour remédier au problème de l'énergie de la queue on peut faire la séparation de l'énergie de la queue du reste, le développeur doit comprendre un tel comportement électrique pour la bonne gestion de l'énergie. L'énergie consommée par les E/S peut être optimisée en utilisant les paquets.

Le Tableau 4.1 résume la consommation d'énergie de cinq applications populaires [60].

Applications	**Temps d'exécution(s)**	**Conso. de la batterie (%)**
Browser	30	35
Angrybirds	28	37
Fchess	33	60
Nytimes	41	75

Table 4.1 – La consommation énergétique de cinq applications populaires

Remarque : Les E/S consomment beaucoup d'énergie. Les expériences ont montré les observations clés sur l'énergie consommée dans les applications les plus populaires :

- La majorité de l'énergie est consommée par l'affichage graphique.
- La majorité de l'énergie consommée est dissipée par un petit nombre d'Entrés/Sortie.
- Le processeur consomme très peu de puissance même lors de l'exécution de milliers d'opérations.
- La minimisation du nombre des paquets minimise la consommation énergétique.

Le Tableau 4.2 Présente un résumé sur l'énergie consommée relative à quelques applications vis à vis des E/S [60] :

Applications	**Paquets**	**(E/S utilisées) / (E/S totales) (%)**
Browser	30s	35
Handset:tytn2 runningWM6.5		
pslide	3 (3 Disk)	9.52
pup	3 (3 NET)	9.37
Handset:magic running Android		
syncdroid	4 (1 NET, 3DISK)	0.88
streamer	3 (3 NET)	0.36
Handset:passion running Android		
Browser	3 (2 Net, 1GPS)	0.14
Angrybirds	4 (3 NET, 1GPS)	0.22
Fchess	2 (2 NET)	0.18
Nytimes	2 (1 NET, 1GPS)	0.2
Mapquest	3 (2NET,1GPS)	0.19
pup	1 (1 NET)	0.27

Table 4.2 – Résumé sur l'énergie consommée vis à vis des E/S **Signification** : **Applications** : Nom des applications traitées, **Paquets** : Type des paquet actifs, **(E/S utilisées) / (E/S totales) (%)** : Rapport entre le nombre de paquets utilisés et le nombre de paquets total.

Le profilage des performances est un sujet longtemps étudié. L'exécution du profilage de temps a été proposée au niveau de l'application [29, 55, 78] pour suivre la trace de l'appel graphique et estimer le temps d'exécution des routines, des langages orientés objet [21, 30] et au niveau du noyau [14].

On peut citer les limites suivantes d'Eprof :

— Eprof est un modèle de puissance fondé sur l'utilisation.

— Eprof ne reflète pas le comportement de puissance asynchrone de l'énergie (la puissance instantanée tirée d'un composant qui ne peut être liée au courant) [60] trouvée dans les smartphones modernes.

RAPL

En 2012, Marcus Hahnel, Bjorn Dobel, Marcus Volp et Hermann Hartig,ont élaboré l'approche RAPL (Running Average Power Limit : Limite de la puissance moyenne glissante) qui permet de faire une comparaison des applications mobiles de deux secteurs et montre qu'elles révèlent une consommation d'énergie différente, tout en offrant des services similaires.

la notion de RAPL est introduite dans [32] vu que l'instrumentation manuelle est grossière et se fait souvent au niveau de l'alimentation de l'appareil ; cette dernière reste inexacte pour le calcul de l'énergie consommée par les applications de la queue. Les auteurs de l'article [77] ont tenté de trouver des approches pour l'efficacité énergétique, de créer des modèles qui adaptent le comportement des applications, de mesurer l'énergie nécessaire pour décoder une image ainsi que le choix du chemin pour le résultat d'une requête.

L'apport de ce travail consiste à calculer l'énergie consommée par les pilotes des appareils, car un pilote non optimisé utilisera plus de ressources CPU, donc il y aura plus de consommation d'énergie. Les dispositifs matériels, tels que des disques durs ou interfaces réseau consomme de l'énergie, L'idée proposée consiste à mesurer non seulement l'énergie consommée du dispositif, mais de mesurer également l'énergie que le pilote de cet appareil consomme en terme de puissance de calcul. L'utilisation de (RAPL) est basée sur des capteurs d'énergie disponibles dans les processeurs Intel récents pour mesurer la consommation d'énergie et pour tenir compte de la consommation d'énergie dans les composantes logicielles. L'utilisation de l'infrastructure RAPL sert pour caractériser les coûts d'énergie et décoder les tranches vidéo et le choix du chemin pour le résultat d'une requête.

Exemple1 : Décodage d'une image

Le codeur livre deux versions : une de haute qualité (choix1) et une autre version de moindre qualité qui peut être décodée à faible consommation (choix2), le décodeur décidera en fonction du budget de batterie entre le choix 1 ou 2.

Exemple2 : Choix du chemin pour le résultat d'une requête

Le SGBD a généralement plusieurs choix pour aboutir au résultat d'une requête. Le but est de permettre au SGBD de sélectionner non seulement la combinaison qui permet d'obtenir un résulat le plus rapidement que possible, mais aussi celui qui consomme le moins d'énergie [32].

On peut citer les limites suivantes de RAPL :

— RAPL fourni des capteurs qui permettent de mesurer la consommation électrique au niveau du CPU et la mémoire mais le problème qui réside dans cette technique est l'impossibilité de calcul de la consommation des périphériques d'E/S.
— RAPL doit avoir des informations précises sur la vidéo pour la minimisation du coût énergétique ce qui rend la tache difficile à mettre en pratique.

Appscope

Apocope [87] est un système qui évalue automatiquement la consommation d'énergie des applications fonctionnant sur les smartphones Android. Sa conception est basée sur le suivi du noyau Android à un niveau microscopique.

L'objectif du modèle AppScope est de pouvoir mesurer l'énergie d'application pour le système d'applications qui utilise des modèles d'alimentation de matériel et les statistiques d'utilisation pour chaque composant matériel.

L'outil AppScope estime avec précision la consommation d'énergie des applications Android. Le système analyse les traces d'une application système puis AppScope recueille des informations d'utilisation basées sur une approche événementielle [87]. AppScope estime avec précision et en temps réel, l'utilisation des composants matériels à un niveau microscopique.

AppScope a été développé avec le noyau Linux 2.6.35.7. La version 1.3 de SystemTap [70] utilise également K probes et la collecte de données aux fins de l'évaluation. Toutes les évaluations sont effectuées sur HTC Google Nexus One (N1 ; Qualcomm QSD 8250 Snapdragon 1GHz, écran Super LCD de 3,7 pouces) [45] avec Android version de te forme 2.3.

AppScope peut être utilisé sur un périphérique basé sur Android sans modifier le logiciel système. La limitation de la méthode en ligne peut éventuellement être surmontée en utilisant une approche en ligne qui utilise une unité de contrôle de la batterie (BMU) [25, 40], qui est intégré dans les smartphones.

AppScope permet la détection des événements qui sont pertinents pour le fonctionnement d'un composant matériel comme le changement fréquence CPU, les paquets de transmission et les entrées / Sortie (E/S).

En se référant aux K probes (K probes [42] qui est utilisé pour surveiller le comportement des appels système), AppScope est compilé comme un module du noyau et contrôlé dynamiquement.

Le tableau 4.3 explique la manière d'interaction d'AppScope avec les différents composants [87].

Remarques :

— Lors de l'activation du GPS, AppScope compte les demandes de localisation de Location Manager. Le comptage est alors utilisé pour estimer la consommation d'énergie pour chaque processus d'application.
— Pour les expériences vidéo, l'instrumentation effectuée est adaptée par Roitzsch dans [71].

Composants	Méthodes utilisées par Appscope
CPU	AppScope détecte la commutation de processus par le suivi d'un événement de réveil par sched_switch()
Wi-Fi	La consommation d'énergie du réseau local varie en fonction du débit de paquets (par exemple, des paquets transmis par seconde)
3G	interface La Consommation d'énergie 3G dépend de l'état de la CRR (CRR : The Radio Resource Control [87])
LCD	La consommation d'énergie d'un écran LCD est proportionnelle à l'afficheur, la luminosité et la durée d'affichage.
GPS	AppScope surveille (appels LocationManager) et calcule la durée d'activation du GPS

Table 4.3 – Type d'interaction de AppScope avec les différents composants **Signification** : **Composants** : Composant traité par Appscope, **Méthodes utilisées par Appscope** : Manière d'interaction d'Appscope pour la détection des événements pertinents pour le fonctionnement du composant.

Limites :

— AppScope génère des résultats moins précis lors de la présence de l'énergie de la queue qui influence sur la précision des résultats générés par AppScope.
— La précision d'AppScope dépend de la caractéristique électrique de l'appareil en question.
— AppScope est opérationnel dans un nombre limité d'architectures de processeur et ne prends pas en considération l'architecture cœurs multiples.

Analyse et comparaison

En 2009 le modèle TailEnder a pour but l'étude et la minimisation de la consommation énergétique de 3G, GSM et le Wi-fi pour les applications qui tolèrent un léger retard, la communication est une cause principale de la consommation énergétique. En 2011, le modèle Cinder était mis en place pour permettre aux utilisateurs de gérer les ressources limitées de l'appareil tel que l'énergie. Cinder a modélisé la consommation d'énergie ainsi que son partitionnement (en utilisant le principe des réserves et des robinets).

La différence entre Cinder et TailEnder réside dans le fait que TailEnder gère la consommation énergétique des trois équipements cités précédemment (3G, GSM, Wi-fi) vis-à-vis aux applications qui tolèrent un léger retard alors que Cinder est un système qui gère la consommation énergétique dans un cadre plus général et envers

toute source de consommation qui peut être optimisée, le modèle Cinder empêche la persistance des applications malveillantes et permet de maximiser l'état actif du smartphone pour le cas d'urgence.

En 2012, [72] a modélisé Eprof qui a permis d'avoir une identification précise sur le coût énergétique passé à l'intérieur des applications ainsi que de déterminer l'entité responsable d'une telle consommation à un tel instant.

Les taches effectuées par TailEnder et Eprof sont complémentaires, ils se focalisent sur l'énergie consommée par les entrées/ sorties (E/S), et de l'énergie de la queue, contrairement au modèle Cinder, Eprof entre à l'intérieur des applications, en contrepartie Eprof ne gère pas le côté sécurité (telle que la gestion de consommation des applications malveillantes ainsi que l'appel d'urgence gérée par Cinder).

L'exécution limite de la puissance glissante (RAPL) était définie en 2012. Cette méthode permet de faire une comparaison des applications mobiles de deux secteurs et montre qu'elles révèlent une consommation d'énergie différente, tout en offrant des services similaires.

Le modèle RAPL a pour but la minimisation de l'énergie de la queue en proposant plusieurs possibilités afin de permettre le choix le moins coûteux en termes de l'énergie de la queue. Son fonctionnement est complémentaire au modèle Eprof qui permet d'identifier là où l'énergie est passée. La méthode RAPL n'a pas beaucoup traité la partie sécurité paradoxalement à Cinder.

En juin 2012, AppScope était mis en pratique pour évaluer automatiquement la consommation d'énergie des applications fonctionnant sur les smartphones Android.

L'efficacité de la méthode proposée était justifiée et juge que les Entrées/Sorties sont les principales sources qui influencent sur la qualité du service d'AppScope

Dans cette présentation on a fait recourt à deux grands types de modèle

Des modèles qui gèrent les ressources telles que : TailEnder , Cinder et R.A.P.L .

Le deuxième type consiste à avoir des évaluations ainsi que des informations sur l'identification précise du chemin énergétique telles que : Eprof et AppScope.

Nous nous intéresserons par la suite à l'estimation de l'énergie consommée dans un appareil mobile et c'est la raison pour laquelle la suite de nos travaux de recherche portera sur AppScope et Eprof

Le travail effectué dans cette partie consiste à mesurer et comprendre comment l'énergie est consommée sur ces appareils mobiles. Pour ce faire, plusieurs composantes sont prises en considération. L'importance des critères est liée au degré de consommation ainsi que l'évaluation du coût énergétique.

Le tableau 4.4 englobe les critères pertinents vis-à-vis des différents modèles étudiés dans notre enquête :

	Qualité	**Optim. hard**	**Optim. appli.**	**QoS**	**Sécurité**	**Dev.**
TailEnder	+	++	NA	+	NA	NA
Cinder	+	++	NA	++	+++	NA
Eprof	++	+	+	+	NA	++
RAPL	+	++	NA	+	+	+
AppScope	+	NA	+++	+	NA	+

Table 4.4 – Type d'interaction des modèles d'énergie avec les différents composants. Signification des colonnes : **Qualité** : qualité du modèle énergétique, **Optim. hard** : prise en compte des optimisations matérielles, **Optim. appli.** : prise en compte des optimisations logicielles, **QoS** : impact sur la qualité de service, **Sécurité** : prise en compte de la sécurité, **Dev.** : aide au développeur.

Résultats et Synthèse

On remarque que les modèles du premier type (TailEnder, Cinder et R.A.P.L) ont réalisé des résultats pertinents pour les trois critères suivants : performance des optimisations hard, impact sur la qualité des services et la sécurité. Toutefois, les modèles du deuxième type (Eprof et AppScope) ont permis d'évaluer et d'identifier précisement l'acheminement énergétique. Ils ont montré leurs efficacités dans les critères : qualité du modèle énergétique, performance des optimisations des applications et aide au développeur. Le modèle Cinder est le seul qui a réalisé trois fois le meilleur résultat pour les critères 2, 4 et 5 (voir le tableau 4.4). Le modèle AppScope a montré une grande efficacité pour le critère performance des optimisations applications.

4.2.3 Conclusion et perspective

Dans cette partie, nous avons étudié le problème de l'optimisation énergétique d'un smartphone.

Dans un premier temps, nous avons présenté l'évaluation et les mesures du coût énergétique, ainsi que les problèmes rencontrés dans les méthodes utilisées pour l'optimisation énergétique. Pour ce faire, nous avons étudié cinq modèles : TailEnder, Cinder, Eprof, RAPL et AppScope.

Ensuite, nous avons essayé d'analyser et de comparer ces modèles selon plusieurs critères.

Les résultats présentés permettent de connaître les principales sources de consommation énergétique. Dans les prochains travaux, on continuera à évaluer ces modèles en essayant de traiter le cas d'optimisation d'énergie en utilisant les machines virtuelles.

4.3 Estimation et optimisation de la consommation d'énergie sur les smartphones

4.3.1 Introduction

La plupart des applications mobiles développées jusqu'à présent ont été conçues sans prendre en considération leur consommation d'énergie [63]. La consommation d'énergie dans un smartphone correspond au coût necéssaire pour réaliser une activité pour faire fonctionner un équipement ou un composant. En d'autres termes, la consommation d'énergie pourrait résulter de l'exécution de différentes interactions entre le matériel, les logiciels et les utilisateurs qui, par leurs comportements, déclenchent une charge de travail sur les composants matériels.

La recherche présentée fait partie du développement de modèles. La deuxième section détaillera les principales sources de consommation d'énergie et leurs différents impacts sur la durée de vie de la batterie. La section 4.3.3 se concentrera sur les pricipales sources d'évaluation ainsi que la mesure du coût de l'énergie. Nous menons notre travail sur les modèles suivants : Kalimucho, AppScope, RAPL et BITWATTS. Nous allons faire une évaluation de différentes méthodes puis on finira par une conclusion et perspectives. [34]

4.3.2 Composants matériels

Les smartphones modernes sont équipés d'une variété de composants matériels intégrés. Ces composants ont des impacts différents sur la consommation d'énergie des appareils mobiles. Les modèles de consommation d'énergie ne doivent pas seulement tenir compte des principaux composants matériels de l'appareil, mais aussi de ses caractéristiques. Chaque composant matériel peut être dans de nombreux états, chacun consommant une quantité d'énergie différente.

CPU

La consommation d'énergie du processeur est fortement liée à sa fréquence, la consommation d'énergie pour un CPU unique est calculée en fonction de la formule précisée dans DevScope [40] :

$$P_{\text{composant}} = P_{\text{mesurée}} + P_{\text{basique}}$$

où P_{basique} représente la consommation d'énergie totale d'un composant. CPU à deux états : inactif et actif [40].

L'affichage

Les applications multimédias, telles que la diffusion d'image, les outils de capture vidéo et les jeux constituent maintenant une partie considérable de l'utilisation quotidienne des smartphones. La consommation d'énergie de ces applications dépend fortement du type de technologie d'affichage utilisée, qui joue également un rôle important dans l'interaction homme-machine. Le modèle d'alimentation pour l'affichage dépend de la technologie d'affichage utilisée. Le type d'écran du smartphone est LCD, OLED et Active Matrix OLED (AMOLED). Par exemple, la consommation d'énergie pour l'écran LCD est liée à sa luminosité. Dans la figure 4.3, la gamme de luminosité de l'écran est de 0 à 255. Les résultats montrent que la consommation d'énergie de l'écran LCD varie avec le niveau de luminosité de l'écran et montre aussi que la consommation d'énergie de l'affichage n'est pas linéaire avec le niveau de luminosité.

Le pixel varie avec la composition de couleur chromatique affichée et les couleurs plus intenses nécessitent plus d'énergie pour être affiché [40].

L'affichage d'un téléphone intelligent se compose de plusieurs pixels individuels. La couleur d'un pixel est construite grâce à la combinaison de couleurs RVB de base qui sont elles-mêmes les seules couleurs directement émises par l'unité d'affichage par petits sous-pixels. L'unité d'affichage est définie par l'agencement de ces sous-pixels.

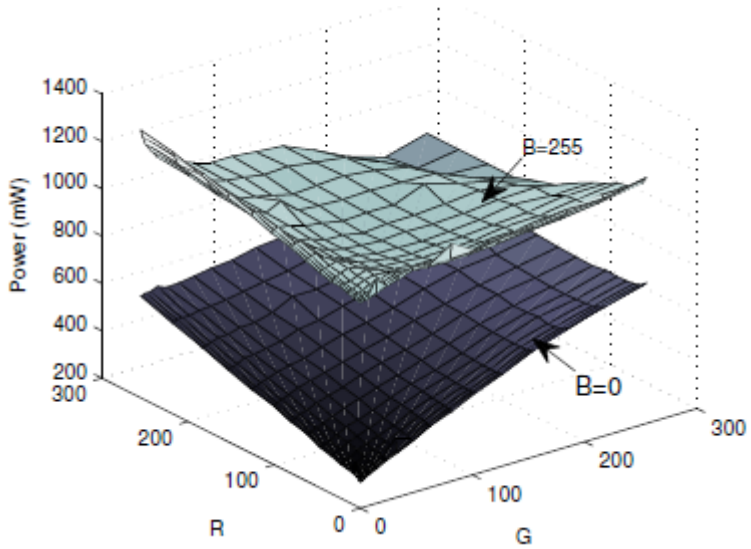

Figure 4.3 – Mesures de puissance pour différentes couleurs sur un affichage AMOLED [53]

Les mesures de puissance pour AMOLED et OLED [53] montrent que lorsque l'écran est de bas niveau jaune (rouge + vert), il y a moins d'énergie consommée. Contrairement à la technologie LCD, la consommation d'énergie d'un écran AMOLED dépend fortement de la couleur.

L'écran LCD peut être modélisé en utilisant simplement le niveau de luminosité, car la variation avec la couleur n'est pas significative.

4.3.3 Interface réseaux

Wifi

Le Wi-Fi est court pour la norme de connexion sans fil et offre une bande passante de 11 Mo/s pour la norme IEEE 802.11b. Ce composant possède également des états inactifs. En état chargé, la consommation d'énergie est conforme à diverses fréquences de sa livraison de paquets de données.

En mode veille dans les statuts de fréquence basse et haute, il y a la même consommation d'énergie car elles sont en état de repos.

3G

Apparu en 2000, la troisième génération (3G) désigne une génération de normes de téléphonie mobile. Il est représenté principalement par le système de télécommunication mobile universel (UMTS) et CDMA2000, permettant un débit de 2-42 Mb/s, beaucoup plus rapide qu'avec la génération précédente.

La consommation d'énergie de 3G dépend de l'état de fonctionnement.

4G

4G est la quatrième génération de normes pour la téléphonie mobile. En réunissant 2G et 3G. La 4G permet la transmission de données « très haute vitesse mobile » à des vitesses théoriques supérieures de 100 Mb/s ou supérieure à 1 Gb/s. Le téléphone intelligent 4G peut avoir deux états : RRC_IDLE et RRC_CONNECTED. Le smartphone transite du RRC_IDLE vers le RRC_CONNECTED lorsqu'il y a réception/ transmission de données [83]. Le tableau 4.5 détaille les différents états existants. L'énergie consommée par l'interface réseau 3G lors de la réception et de la transmission est minimale par rapport à l'énergie consommée par l'interface 3G. Cependant, pendant la transition de l'état Idle à l'état actif, l'interface 4G consomme plus d'énergie que la 3G.

4G+

LTE-Advanced est une norme de réseau de téléphonie mobile de 4e génération définie par l'organisme de normalisation 3GPP qui fait partie (avec le Gigabit WiMAX)

Interface (3G/4G)	Émission (mW)	Réception (mW)	Passage vers l'état actif (mW)
4G	438,39	51,97	1 288,04
3G	868,98	122,12	817,88

Table 4.5 – Consommation électrique d'un smartphone HTC

des technologies réseaux retenues par l'Union internationale des télécommunications (UIT) comme norme 4G IMT-Advanced ; il représente la « vraie » 4G. LTE signifie Long Term Evolution [1].

Le LTE-Advanced sera capable de fournir des débits pics descendants (téléchargement) supérieurs à 1 Gb/s à l'arrêt et à plus de 100 Mb/s pour un terminal en mouvement, grâce aux technologies réseaux intelligentes [2] qui permettent de maintenir des débits plus élevés en tout point de la cellule radio [3]

5G

La technologie 5G est une « technologie clé » [4] qui pourrait permettre des débits de télécommunication mobile, de plusieurs gigabits de données par seconde, soit jusqu'à 1 000 fois plus rapides que les réseaux mobiles en 2010 [5] et jusqu'à 100 fois plus rapide que la 4G à l'horizon 2020 [6]

4.3.4 Section expérimentale

Kalimucho

Kalimucho est une plate-forme de reconfiguration qui met en œuvre un déploiement de contexte heuristique pour trouver une configuration satisfaisant les conditions de contexte et de qualité de service (QoS).

La plate forme kalimucho se concentre sur la gestion de la qualité des services dans les systèmes mobiles contraints par la reconfiguration dynamique de ces applications [60]. Le principe de Kalimucho consiste à saisir les éléments contextuels

1. https://www.nextinpact.com/archive/52582-lte-advanced-europe-mobile-internet.htm/
2. http://www.artizanetworks.com/resources/tutorials/accelera_tech.html/
3. http://www.3gpp.org/technologies/keywords-acronyms/97-lte-advancedl/
4. https://www.entreprises.gouv.fr/files/files/directions_services/politique-et-enjeux/innovation/technologies-cles-2020/technologies-cles-2020.pdf/
5. http://www.lemonde.fr/technologies/article/2014/02/25/la-5g-devrait-vous-faire-rever_4372801_651865.html/
6. https://www.lesechos.fr/19/05/2015/lesechos.fr/02176466182_alcatel-et-le-coreen-kt-partenaires-dans-la-5g-mobile.htm/

nécessaires à la prise de décision (figure 4.4).

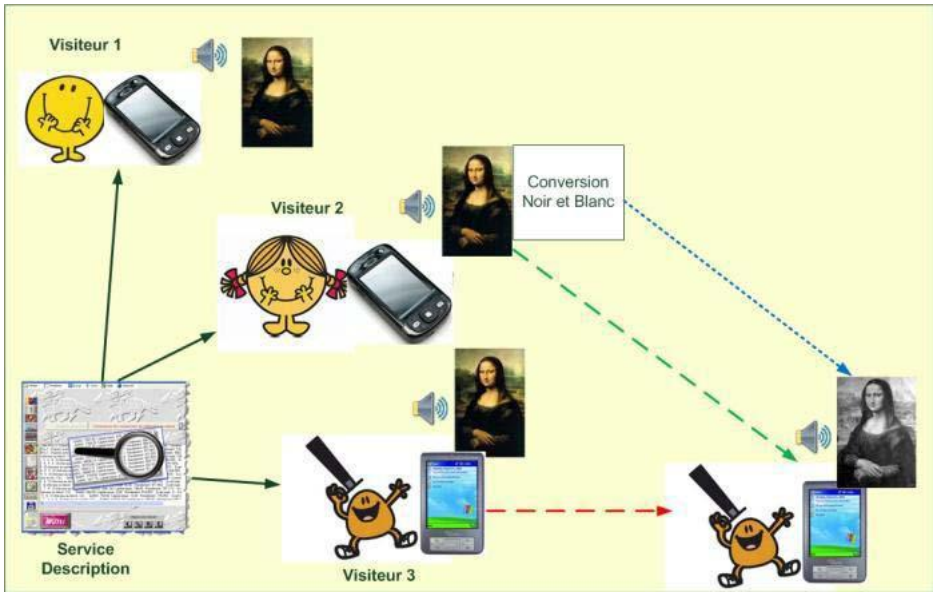

Figure 4.4 – Kalimucho [60] **Signification** : **Cas 1** : Réduction de nombre de données en réduisant le nombre des pixels pour minimiser le coût, **cas 2** : Fournir des informations aux nouveaux visiteurs en se basant sur le dispositif des premiers arrivés pour une meilleure Qos avec un coût minimal.

L'exemple présenté dans [49], détaille une application de musée qui offre aux visiteurs de voir sur leurs smartphones une description des travaux présents dans le musée 4.4. La meilleure QoS est de fournir la vidéo en couleur. Un scénario possible d'utilisation de l'application est lorsque le visiteur 3 se déplace. Deux cas sont présentés :

Cas 1 Faible débit et l'appareil se situe à la portée du serveur. Pour assurer le service, la solution consiste à réduire le nombre de données à transmettre par une conversion de la vidéo en noir et blanc ou en réduisant la résolution de l'image.

Cas 2 L'appareil n'est pas dans la portée du serveur. La solution consiste à utiliser le dispositif du visiteur 2 ou 1 (figure 4.4) pour établir un relais pour le dispositif 3 sous condition que le périphérique 1 ou 2 du visiteur dispose de l'énergie disponible.

Pour un bon déploiement d'une configuration appropriée, Kalimucho se réfère à quatre critères pour assurer une bonne qualité de service : énergie, CPU, mémoire et capacité réseau.

AppScope et RAPL

Ces Modèles ont été traités avec précision dans la section 4.2.2.

On s'est référé à ces derniers également dans cette contribution vu leurs efficacité et cohérence avec ce qui suit.

BITWATTS

En 2014, Colmant et al, ont développé le BITWATTS, une technique de supervision à grain fin qui fournit une estimation en temps réel de la puissance énergétique consommée. L'idée de BITWATTS repose sur l'utilisation des processus et la déduction de l'énergie consommée. En outre, cette conception peut être utilisée dans des environnements distribués [19].

Les approches actuelles, comme JOULEMETER, peuvent superviser pratiquement une application à la fois (chaque machine virtuelle supervise une seule application). Dans ce cas, les machines virtuelles (VM) sont considérées comme des boîtes noires. BITWATTS est développé avec Scala [18], une extension de l'outil d'action Power API tool [19] qui permet une comparaison des mesures physiques effectuées avec une puissance pour justifier son compteur d'efficacité. Les résultats indiquent que BITWATTS fournit une estimation de puissance digne de confiance dans quelques pourcent des mesures réelles lorsqu'elles sont configurées avec le modèle d'alimentation approprié pour le matériel sous-jacent [44].

L'idée est basée sur l'utilisation du processus et déduit la consommation d'énergie dans un environnement virtuel. Par exemple, nous n'avons pas accès au CPU physique, mais nous pouvons observer le processeur émulé par les machines virtuelles. L'architecture de BITWATTS est décrite dans la figure 4.5.

Estimation de la consommation dans les canaux de communication

L'échange de données entre les instances de BITWATTS nécessite un échange de données entre hôte et VM pour estimer la consommation d'un Processus dans la machine virtuelle :

- Dans une configuration distribuée, pour estimer la consommation d'énergie BITWATTS doit utiliser un autre serveur d'évaluation du modèle d'approvisionnement.
- Pour démontrer que BITWATTS peut gérer des applications avec une charge variée, nous commençons l'observation avec une expérience de base sur Intel Core I3.
- Dans cette expérience, les résultats sont comparés non seulement en ce qui concerne la performance limite de la puissance moyenne glissante (RAPL), mais aussi avec un outil de mesure externe (Power Spy) [44] qui permet une

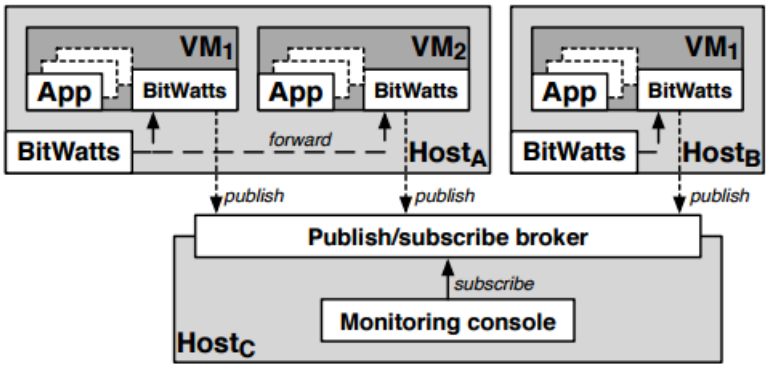

Figure 4.5 – Architecture de BITWATTS [44]. **Signification** : BITWATTS assure l'échange des données entre la machine hôte et machine virtuelle, ça permet d'estimer la consommation d'un processus dans une machine virtuelle.

consommation précise de l'information. De plus, pour cette expérience, la fréquence du CPU a été portée à 1,6 GH, figure 4.6.

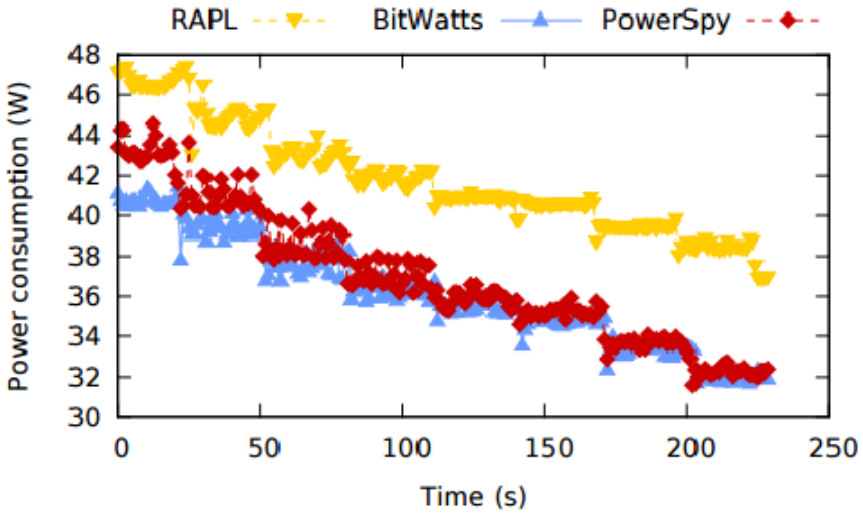

Figure 4.6 – Comparaison de 3 modèles de consommation énergétique [44]. **Signification** : BITWATTS effectue une estimation précise sur le système testé vu l'absence de l'energie de la queue dans les environnements virtuels.

Nous constatons que l'énergie est surestimée par RAPL, BITWATTS est proche de PowerSpy. Cela indique que BITWATTS effectue une estimation précise sur le système testé.

4.3.5 Conclusion

Dans cette partie, nous avons étudié le problème d'estimation et d'optimisation de l'énergie consommée dans un smartphone. Tout d'abord, nous avons présenté l'évaluation et la mesure du coût de l'énergie, puis les problèmes rencontrés dans les méthodes utilisées pour l'optimisation de l'énergie. Précisément, quatre modèles majeurs sont pris en considération : Kalimucho, AppScope, RAPL et BITWATTS. En outre, nous avons exposé les coûts énergétiques de divers matériels en essayant de les corréler avec les modèles cités. Les résultats présentés ont permis d'identifier les principales sources responsables de la consommation d'énergie. Nous avons remarqué que les derniers modèles ne tiennent pas compte de l'énergie de la queue qui influe de manière permanente sur les activités du logiciel, contrairement à BITWATTS qui estime la consommation énergétique dans un environnement virtuel.

Dans le chapitre suivant, on va effectuer une évaluation expérimentale sur des données réelles afin d'élaborer un modèle mathématique relatif à une étude expérimentale. Des tests seront effectués pour mener à bien la suite de notre travail.

Chapitre 5

Campagne de mesures lors de mon tour de France

5.1 Introduction

Dans le chapitre précédent, on a traité le cas des modèles de consommation énergétiques qui se basent sur des mesures effectuées dans des environnements protégés. Dans ce chapitre on va détailler un projet qui a consisté à faire le tour de France en fauteuil roulant électrique en passant par les différentes universités françaises tout en effectuant des mesures sur le terrain.

5.2 Description générale du projet

Le but du projet consiste à réaliser un tour de France en fauteuil roulant électrique sur une durée de 33 jours pour un parcours de plus de 3000 km en passant par différents pôles universitaires.

Ce projet a été réalisé en collaboration avec le centre national de la recherche scientifique (CNRS), l'université de Lille, le centre de recherche en informatique, signal et automatique de Lille (CRIStAL), l'institut de recherche sur les composants logiciels et matériels pour l'information et la communication avancée de Lille (IRCICA), l'université Mohammed Premier d'Oujda (Maroc) et l'association des paralysés de France (APF).

Les principaux objectifs de ce projet sont liés à mon activité de doctorant en informatique :

— Effectuer des mesures scientifiques liées à mon sujet de thèse tout au long du trajet :
— Réaliser dans un cadre scientifique un record de distance en fauteuil roulant entre le 02 mai et le 03 juin 2016.

— Collecter des données scientifique en se basant sur plusieurs scénarios.
— Rencontrer des personnes en situation de handicap des différentes universités et les sensibiliser à l'importance des études supérieures ,

Afin de réaliser un tel parcours (3006 km en 33 jours) il a fallu parcourir en moyenne environ 100 km/jour. L'itinéraire était étudié d'une manière exacte, le projet était réalisé par un fauteuil électrique tout terrain capable de fournir une vitesse et des performances compatibles aux besoins, les spécificités du matériel seront détaillées dans la partie 5.4.1 La figure 5.1 détaille le parcours du projet.

5.3 Objectifs scientifiques détaillés

Vu la capacité limitée du matériel et les demandes des utilisateurs, il faut gérer le conflit d'intérêts existant entre la consommation énergétique et la qualité de service voulue.

La première partie scientifique du projet consiste à comparer la consommation énergétique d'une tablette en fixant plusieurs métriques auparavant. Ces mesures comparatives ont été réalisées dans un environnement réel et confrontées aux mesures de consommation réalisées au préalable dans un environnement contrôlé. Cette partie consiste à Mesurer les critères suivants :

— Consommation énergétique ;
— Qualité des services ;
— Connectivité.

5.4 Outils d'expérimentation

5.4.1 Le fauteuil tout terrain : Quadrix

Afin de réaliser le tour, un fauteuil tout terrain était conçu pour pouvoir atteindre une vitesse de 25 km/heure et une autonomie qui assurer 100 km par jour, les étapes effectuées étaient d'environ 8 heures/jour avec environ 2 heures de pause dans la journée et pour garder une marge en cas d'imprévu.

Des essais ont été effectués sur un terrain écologique (terrain accidenté, ville...), le matériel était validé pour tous les besoins du projet (Vitesse, maniabilité, confort, franchissement des bordures<17 cm...). Pour prévenir des aléas de la route une batterie supplémentaire (soit 3 au total) pour assurer 140 km d'autonomie. La figure 5.2 présente le fauteuil tout terrain (FTT) qui a servi à la réalisation de l'expérimentation. Le FTT était équipé d'un coussin anti-escarre et d'un capteur position qui permet d'une part d'éviter les escarres, de détecter ma présence et d'alerter en cas de chute ou de vol. Les différentes forces (points d'appuis) exercées sur le coussin étaient enregistrées et envoyées en temps réels dans le but d'exploiter

Figure 5.1 – Parcours du projet de recherche

Figure 5.2 – Fauteuil d'expérimentation utilisé pour la collecte des données

les informations obtenues pour concevoir des matelas anti-escarres connectés qui seront exploitée par la suite dans un contexte médical.

Le fauteuil était également doté d'un boîtier équipé d'une connectique Bluetooth et d'une carte SIM 2G. Il intègre un accéléromètre, qui lui permet d'enregistrer les éventuels chocs subis, et un système de géolocalisation, pour déterminer son emplacement à tout moment avec précision. Le but scientifique est de déterminer la qualité de signal dans les différentes zones parcourues afin de définir les zones blanches...

5.4.2 Outil matériel

L'outil utilisé pour toutes les expérimentations est une tablette de type HP Pro Slate 8 qui a été rootée auparavant, on a opté pour ce type vu la capacité de sa batterie qui permet un grand nombre de tests pour chaque cycle de charge ainsi que l'architecture de processeur de type (Snapdragon 800) qui représente un bond en avant en performance et en efficacité énergétique. Les caractéristiques [1] de la tablette sont détaillés dans le tableau 5.1.

5.4.3 Outils logiciels

Trepn Profiler

Trepn Profiler est un outil de diagnostic qui permet de profiler les performances et la consommation d'énergie des applications Android. Tous les tests de cette expérimentation étaient traités par la version V6.2s.

On peut avoir les informations sur l'état du système, l'état du réseau, les graphes de performance, la vitesse, la fréquence des processeurs... La figure 5.3 illustre un exemple de présentation.

CPU Frequency

CPU Fréquency est un outil conçu pour afficher des informations de fréquence CPU. Cet outil permet de changer le réglage de la fréquence CPU afin d'économiser l'énergie ou d'obtenir de meilleures performances.

Plusieurs possibilités peuvent avoir lieu :

— Fixation des fréquences des CPU : 300, 652, 1000...
— Basses fréquences : choix automatiques des basses fréquences afin de baisser la consommation
— Fréquence optimale : choix automatique d'une fréquence qui minimise la consommation sans influencer dégradation de la qualité de services (QoS)

1. `https://support.hp.com/fr-fr/product/hp-pro-slate-8/7398584/model/7398585/product-info/`

Type de Produit	Tablette
Système d'exploitation	Android 4.4.4 (KitKat)
Type d'affichage	7.86" IPS TFT
Résolution	2048 x 1536 (326 ppi)
Caractéristiques d'écran	Multi-touches à 10 points, technologie Direct Bonding, verre Corning Gorilla Glass 4, Grand angle de visualisation
Processeur	QUALCOMM Snapdragon 800
Fréquence CPU	2.3GHz
Nombre de coeurs	Quadricœur
Mémoire de stockage	32 Go eMMC
RAM	2 Go - LPDDR3 SDRAM
Connectivité sans fil	802.11a/b/g/n/ac, NFC, Bluetooth 4.0 LE
Protocoles de sécurité	Wi-Fi Miracast
Résolution Camera	8 Mégapixels (arrière), 2 Mégapixel (avant)
Vidéo HD	1080p
Lieu et localisation	GPS
Quantité de Batteries	1
Technologie	Lithium-polymère
Capacité	21 Wh
Durée de fonctionnement	Jusqu'à 13.75 heures
Autonomie	Lecture vidéo jusqu'à 13.75 heures
Capteurs présents	Accéléromètre, capteur de lumière ambiante, Capteur de proximité, boussole numérique, baromètre, capteur gyroscopique, capteur à effet Hall
Temp de fonctionnement mini	0 C
Temp de fonctionnement maxi	40 C
Temp de stockage mini	-20 C
Temp de stockage maxi	60 C

Table 5.1 – Caractéristiques du matériel d'expérimentation

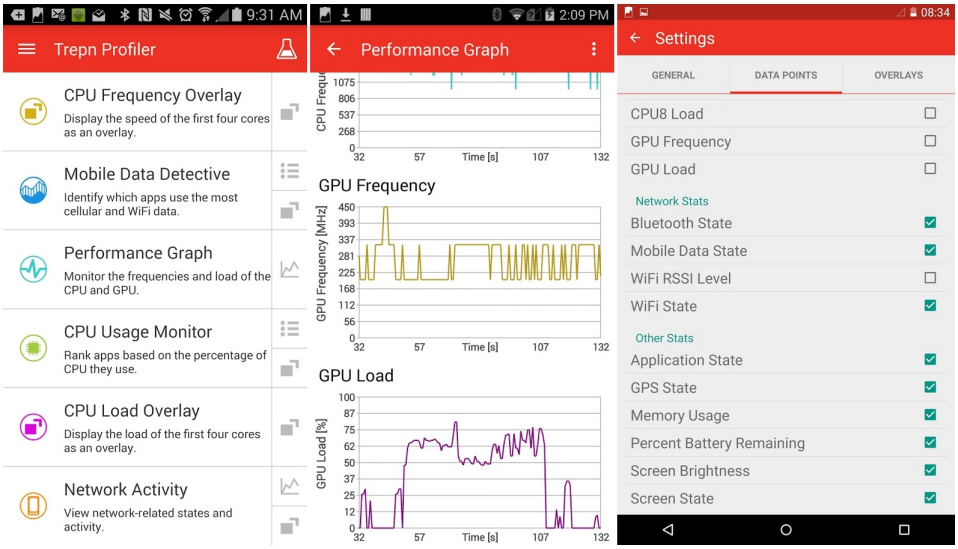

Figure 5.3 – Présentation du trepn Profiler

CPU Fréquency permet également de donner des informations sur l'état des processeurs (Idle, sleep ou active state)(voir figure 5.4).

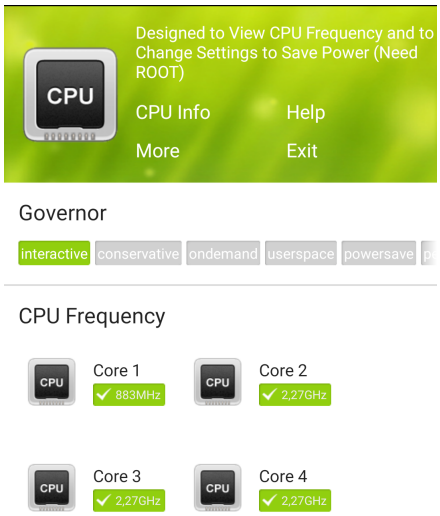

Figure 5.4 – Présentation du CPU Frequency

Cronoid

Coronoid est un outil d'automatisation qui permet d'effectuer les tâches sur une base régulière comme cron. Il permet également la possibilité d'effectuer des tâches automatiquement lorsque l'état du terminal a changé. La version utilisée pour l'élaboration de ce travail est Cronoid-3.5.1.

Le point fort de l'outil consiste à minimiser le contact avec l'utilisateur lors de l'exécution d'une ou plusieurs taches, ce qui permettra d'avoir plus de précision pour les mesures de consommation. Parmi les taches réalisées on peut citer l'envoi automatique des SMS, simulation d'appels, déclenchement d'une action à un niveau précis de batterie pour arrêter le test à titre d'exemple...

La section suivante détaillera la manière par laquelle la collecte des données était effectuée.

5.5 Scénarios d'expérimentation

Le but de cette étude est de faire un suivi détaillé sur les principales sources de la consommation énergétique, pour ce faire on s'est basé sur plusieurs scénarios différents afin de mieux suivre et comprendre l'aspect énergétique. Dans la suite de nos traitements on se basera sur deux types de scénarios :

— Le mode déconnecté dans lequel toutes les mesures ont été effectuée en présence du réseau GSM uniquement.
— Le mode connecté où la totalité des techniques de connexion étaient activées (3g, 4g, Wi-Fi, GPS..) selon le scénario choisi.

5.5.1 Scénarios du mode déconnecté

Le mode déconnecté consiste à effectuer des mesures sans être connecté à internet (3G/Wifi). Dans ce type de mesure, deux scénarios sont à présenter :

Vidéo Locale avec Fréquence Fixe (VLFF)

Le scenario VLFF : Vidéo locale avec Fréquence fixe sert à suivre l'état du système en mesurant la consommation énergétique globale en lançant une vidéo locale sous une fréquence fixée à l'avance.

Vidéo Locale avec Fréquence Variable (VLFV)

Dans Le mode VLFV (Vidéo Locale avec Fréquence Variable) la prise des mesures est faite en lançant une vidéo locale sans fixer les fréquences des différents processeurs.

Parmi les modes existants on peut citer :

— Mode interactif 1 : la fréquence des CPU correspond à la fréquence par défaut fourni par le système.
— Mode powersave : la fréquence choisie consiste à garder le maximum d'énergie.
— ONdemand : la fréquence est fixée par l'utilisateur.

5.5.2 Scenarios mode connecté

Le mode connecté est le mode le plus utilisé et le plus gourmand en énergie. Dans ce mode toutes les expérimentations seront effectuées en étant connecté au Wi-Fi. Pour des raisons techniques, nous avons utilisé un téléphone connecté en 4G comme point d'accès Wi-Fi. En plus des modes VLFF et VLFV, le mode connecté contient les scénarios suivants.

Navigation avec Fréquence Fixe (NFF)

Le mode NFF : navigation avec fréquence Fixe permet de suivre le comportement du système en activant la navigation tout au long de l'expérimentation en se fixant la fréquence des CPU.

Navigation avec Fréquence Variable (NFV)

Le mode NFV : Navigation avec fréquence variable est similaire au mode NFF, mais en se basant sur des fréquences CPU variables ou limitées dans des intervalles fixés à l'avance.

Vidéo Distante avec Fréquence Fixe (VDFF)

Le VDFF : Vidéo distante avec fréquence fixe est un mode qui consiste à lancer une vidéo via le net (YouTube etc.) et de suivre le comportement énergétique en fixant la fréquence des CPU.

Vidéo Distante avec Fréquence Variable (VDFV)

Le mode VDFV : Vidéo distante avec fréquence variable est similaire au VDFF en utilisant une fréquence variable.

Audio Distante avec Fréquence Variable (ADFV)

Le mode ADFF : audio distante avec une fréquence fixe, dans ce mode le comportement du système est suivi en lançant une audio via le net.

Audio Distante avec Fréquence Fixe (ADFF)

Le mode ADFV : audio distante avec une fréquence variable est similaire au ADFF mais avec une fréquence CPU variable. Le Tableau 5.2 présente les différentes caractéristiques des scénarios utilisés pour l'évaluation de notre étude.

Exemple de fichier de mesure

Après chaque expérimentation, les données obtenues sont stockées dans un fichier. Vu le nombre important des paramètres et des données récupérées, le fichier de mesure est représenté sous forme de deux tableaux 5.3 et 5.4 qui présentent les données mesurées d'une expérimentation qui a duré 45 minutes dans laquelle 27000 mesures étaient effectuées. Le tableau 5.3 représente la charge relative aux différents CPU ainsi que leurs fréquences respectives. Pour mesurer les différentes fréquences des quatre CPU et leurs charges utilisées, nous avons utilisé les outils suivants :

- TrepnProfiler (voir page 81) pour effectuer les différentes mesures.
- CPU Frequency (voir page 81) pour la gestion des fréquences des CPU.
- Cronoid (voir page 84) pour automatiser le lancement et l'arrêt d'expérimentation et pour veiller sur la fiabilité du dispositif d'expérimentation et pour minimiser le contact utilisateur.

Le tableau 5.6sente les différentes applications actives et les classes selon leurs fréquences d'utilisation (nombre sollicitation par expérimentation)

Le tableau 5.5 indique le pourcentage d'utilisation des ressources CPU par toutes les applications présentes lors de l'expérimentation. Les ressources relatives à la mémoire virtuelle (la moyenne et le maximum utilisés) ainsi que pour la mémoire physique. Comme il s'agit d'un cas de test relatif à une vidéo distante, on remarque la forte demande de ressources CPU via YouTube (20.51 %) qui correspond à la plus grande valeur par rapport aux autres applications relatives à la même expérimentation.

Nom du modèle	Signification	Mode	Description
VLFF	Vidéo Locale avec Fréquence Fixe	Déconnecté	Suivi du comportement énergétique d'une vidéo locale sous une fréquence fixe
VLFV	Vidéo locale avec fréquence variable	Déconnecté	Suivi du comportement énergétique d'une vidéo locale sous une fréquence variable
VDFF	Vidéo distante avec fréquence fixe	Connecté	Suivi du comportement énergétique d'une vidéo distante (YouTube) sous une fréquence fixe
VDFV	Vidéo distante avec fréquence variable	Connecté	Suivi du comportement énergétique d'une vidéo distante (YouTube) sous une fréquence variable
ADFF	Audio distante avec fréquence fixe	Connecté	Suivi du comportement énergétique d'une piste audio distante (Mp3...) sous une fréquence fixe
ADFV	Audio distante avec fréquence variable	Connecté	Suivi du comportement énergétique d'une piste audio distante (Mp3...) sous une fréquence variable
NFF	Navigation avec fréquence fixe	Connecté	Suivi du comportement énergétique en lançant une navigation via le GPS sous une fréquence fixe
NFV	Navigation avec fréquence variable	Connecté	Suivi du comportement énergétique en lançant une navigation via le GPS sous une fréquence variable

Table 5.2 – Caractéristiques des scénarios de référence.

Time (ms)	**Cpu1 Freq (kHz)**	**Cpu1 Load %**	**Cpu2 Freq (kHz)**	**Cpu2 Load %**	**Cpu3 Freq (kHz)**	**Cpu3 Load %**	**Cpu4 Freq (kHz)**	**Cpu4 Load %**
0	652800	85	729600	80	729600	60	729600	71
100	652800	88	729600	82	729600	66	729600	80
200	652800	73	729600	81	729600	83	729600	60
300	652800	100	729600	82	729600	75	729600	75
400	652800	80	729600	83	729600	75	729600	83
500	652800	83	729600	88	729600	83	729600	80
600	652800	92	729600	86	729600	100	729600	80
700	729600	87	729600	85	729600	100	729600	87
800	729600	66	729600	90	729600	87	729600	75
900	729600	80	729600	91	729600	71	729600	83
1000	729600	80	729600	82	729600	83	729600	75
...	...	...	...	...	...	...	...	...
...	...	...	...	...	...	...	...	...
27000	300000	90	883200	90	729600	80	729600	80

Table 5.3 – Fichier de mesures (partie 1). Signification des colonnes : **Time** : Intervalles des mesures, **Cpu3 Freq (kHz), Cpu2 Freq (kHz), Cpu3 Freq (kHz), Cpu4 Freq (kHz)** : Les fréquences respectives des CPU 1, CPU 2, CPU 3 et CPU 4. **Cpu1 Load, Cpu2 Load, Cpu3 Load, Cpu4 Load** : Pourcentages des charges respectives des CPU 1, CPU 2, CPU 3 et CPU 4.

Time (ms)	**Memory Usage (Kb)**	**Screen Brightness**	**Battery Power (uW)**	**Battery Remaining**	**GPU Load (%)**	**CPU Load (%)**
0	1800004	255	875708	3665025	45	89
100	1800376	255	866186	3665076	45	86
200	1800516	255	831271	3665121	45	83
300	1800648	255	880999	3665165	45	76
400	1800896	255	868302	3665230	45	76
500	1801152	255	890521	3665275	45	77
600	1801504	255	948712	3665334	45	82
700	1801636	255	920146	3665387	45	65
800	1802140	255	891579	3665439	50	82
900	1802404	255	882057	3665495	50	68
1000	1802660	255	893862	3665440	50	75
...	....	...	...	...	...	...
...	...	...	...	...	...	...
27000	1802410	255	883200	3665400	47	82

Table 5.4 – Fichier de mesures (partie 2). Signification des colonnes : **Time** : l'intervalle des mesures, **Memory Usage (Kb)** : Mémoire utilisée par intervalle de mesure, **Screen Brightness** : État de luminosité de l'écran, **Battery Power (uW)** : La puissance consommée par intervalle de mesures (en micro Watt), **Battery Remaining (%)** : Batterie restante en %, **GPU Load (%)** : La charge totale du GPU, **CPU Load (%)** : La charge totale des CPU.

Sensors Application	**Type Application**	**Average**
200	Mobile Data State	0.0
205	Battery Remainin (%)	47.63441680465886
206	Battery Status	0.0
328	Memory Usage	1772465.436876518
331	Screen Brightness	247.38371006912013
332	Battery Power	989798.485167196
400	GPU Frequency	226874.2200635158
401	GPU Load	32.80052307117504
600	CPU Load	75.21363721277788
1000	CPU1 Frequency	856941.4085559499
1001	CPU2 Frequency	664897.7395852793
1002	CPU3 Frequency	812255.0943396227
1003	CPU4 Frequency	860962.047450028
1096	CPU1 Load	76.48867924528302
1097	CPU2 Load	68.41748552213711
1098	CPU3 Load	73.45888286941901
1099	CPU4 Load	74.58380347468709

Table 5.5 – Statistiques sur les capteurs. Signification des colonnes : **Sensors Applications** : identifiant du capteur d'application, **Type Applications** : corresponds au type d'application active pendant l'expérimentation, **Average** : nombre moyen d'accès par application.

Applications	**CPU** [%]	**Average Virtual Memory Size** [MB]	**Max Virtual Memory Size** [MB]	**Average Resident Set Size** [MB]	**Max Resident Set Size** [MB]
Facebook	0.18	1784.41	1794.03	26.99	34.0
GsmaService	0.0018	1483.33	1485.60	5.79	6.09
Horloge	0.006	257.32	1493.86	1.42	8.37
CPU Frequency	5.51E-5	606.2	1489.42	2.79	7.64
Google Play Store	0.004	15.80	1545.80	8.33	9.59
Launcher3	0.004	1589.49	1589.49	12.08	13.789
Services Google Play	0.34	4096.29	6540.66	34.89	60.18
Configuration des partenaires Google	5.04E-4	24.55	1486.86	0.09	5.97
Play-Fi	0.0017	1483.88	1483.88	5.05	5.69
Dictionnaire personnel	0.007	99.57	1491.23	0.52	7.94
YouTube	20.51	1966.10	1985.42	42.51	48.83
Cron Tasker Free	0.06	1330.37	1492.48	7.004	8.63
Gmail	0.050	893.47	1834.24	12.28	33.15
Appli Google	0.032	3084.76	3091.49	16.81	19.653
Cronoid	0.62	1507.74	1520.84	7.97	8.94
Power Battery	0.47	1682.2	3114.73	10.07	29.05
Hangouts	0.05	19.82	1665.26	0.18	21.22
Google+	0.031	64.793	1545.664	0.487	12.356
Clavier Google	0.078	1548.13	1607.44	11.58	14.71
Configuration du réseau mobile	0.28	3034.45	3035.91	14.26	16.59
SmartcardService	0.006	2987.34	2991.18	13.007	14.622
Interface du système	0.25	1727.72	1805.57	41.59	51.07
Messenger	0.14	1871.18	1910.15	46.07	56.89
SVI Settings	3.46	9172.51	9217.14	55.79	64.84
Stockage multimédia	0.002	277.25	1495.03	1.28	7.613

Table 5.6 – Répartition des ressources système. Signification des colonnes : **Application** : Les applications présentes pendant l'expérimentation, **CPU** [%] : Pourcentage d'utilisation des ressources CPU, **Average Virtual Memory Size** [MB] : Utilisation moyenne de la memoire virtuelle par application, **Max Virtual Memory Size** [MB] : Utilisation maximale de la memoire virtuelle par application, **Average Resident Set Size** [MB] : Utilisation moyenne de la memoire réelle par application, **Max Resident Set Size** [MB] : Utilisation maximale de la memoire réelle par application.

5.6 Conlusion

Le projet du tour de France m'a permis de concevoir une méthodologie capable de faire le suivi de la consommation énergétique dans un dispositif mobile pour un certain nombre d'applications sous certaine condition. Pour mener à bien le projet, une étude de besoin a nécessité un ans et demi de préparation afin de prendre en compte tous les cotés (scientifique, logistique, financier...) qui a nécessite un travail sur du long terme à l'aide de l'équipe.

Les principales difficultés rencontrés dans mon périple étaient de :

— Trouver le adéquat bon le bon déroulement des expérimentation (fauteuil tout terrain avec une grande autonomie, dispositif mobile avec une capacité maximale...)

— Chercher les sources de financements potentiels pour couvrir les frais du voyage Automatiser aux maximum les méthode de collecte des données pour gérer les test tout en étant en mouvement

— Gérer les moyens humains pour la conduite du véhicule d'assistance, planifier des séances de kiné...

— Gérer les moyens de communication : page Facebook, publications régionale, TV, presse...

Le Tableau 5.7 présente d'une manière globale le budget total qui était nécessaire pour l'élaboration du projet :

Produit / service	**Coût (en €)**
Fauteuil tout terrain adapté avec équipements	13000
Matériel d'expérimentation	1500
Location de véhicule d'assistance	2500
Carburant	1000
Intervention / dépannage	5000
Hébergement	3000
Nourriture	3000
Communication	4000
Materiel annexe	3000
Divers	3000

Table 5.7 – Budget estimatif du projet

Chapitre 6

Modélisation de la consommation d'énergie

Les travaux de recherche présentés dans cette partie entrent dans le cadre du développement des modèles mathématiques, ces derniers permettent la modélisation et l'évaluation du coût énergétique dans les environnements mobiles. Notre étude a porté sur le système Android.

L'objectif principal de ce travail est de modéliser la consommation d'énergie d'une application particulière fonctionnant sur un appareil mobile. Ce travail consiste à élaborer un modèle mathématique de consommation énergétique spécifique pour le suivi de consommation énergétique d'une vidéo locale, vidéo distante (en streaming) ainsi que dans le cas de navigation. en agissant sur les paramètres suivants :

— Fréquence des processeurs,
— Niveau initial de la batterie,
— Energie dissipée.

Dans cette étude on s'est focalisé sur le déroulement et la mesure des coûts énergétiques, le travail a porté sur le suivi de consommation énergétique en se basant sur plusieurs scénarios d'expérimentations.

Pour le traitement des données, nous avons divisé l'analyse en deux modes :

— Le mode déconnecté
— Le mode connecté.

6.1 Cas du mode déconnecté

6.1.1 Éléments de base

Dans cette étude [35], nous proposons un modèle dans lequel nous décrivons également une méthodologie pour identifier les paramètres qui modélisent le suivi de

la consommation énergétique.

À cette fin, nous avons analysé une collection de données expérimentales recueillies lors de mon tour de France en fauteuil roulant électrique qui avait pour but la collecte des données selon plusieurs modes et en se basant sur différents scénarios présentés dans le chapitre 5.

Pour réaliser notre étude le fauteuil électrique tout terrain (FTT) était équipé de plusieurs capteurs (Vitesse, accélération, position...) (voir § 5.4.1). Les mesures ont été effectuées en se basant sur une tablette équipée de caractéristiques d'écran : Multipoint de 10 points, technologie Direct Bonding, Quad-Core, Type de processeur QUALCOMM Snapdragon 800 2,3 GHz, mémoire de stockage 32 Go eMMC, RAM 2 Go LPDDR3 SDRAM, technologie de batterie Lithium-polymère 21 Wh avec une autonomie qui peut atteindre jusqu'à 13 heures et 45 minutes en cas de lecture vidéo, un système Android (version 4.4.4 - KitKat)(voir le tableau 5.1) qui a été rooté pour avoir l'accés au Gouverneur ("scaling governor") qui est un module du kernel qui a pour rôle la gestion de la fréquence du processeur en fonction de la demande. Par exemple, au lancement d'une application, le gouverneur va faire grimper la fréquence du processeur, puis la faire diminuer lors d'une mise en veille [2].

Pour la fixation des fréquences, on a utilisé l'outil CpuFrequency (voir § 5.4.3 page 81) et l'outil Cronoid (voir § 5.4.3 page 84) pour l'automatisation des tâches afin de minimiser l'interaction avec l'utilisateur (pour avoir un maximum de précision).

6.1.2 Description du mode déconnecté

Le mode déconnecté consiste à désactiver toutes les sources liées à la connexion telle que le Wi-Fi, Bluetooth et le GPS. Dans ce mode, on s'est basé sur des données locales pour la modélisation énergétique. Seul le réseau GSM était actif.

Notre étude a porté sur les scénarios liés à la vidéo vu la taille de séquences qui permet une meilleure analyse énergétique. En plus, le choix d'étude était basé sur deux scénarios à savoir :

— Vidéo locale avec fréquence fixe **VLFF**
— Vidéo locale avec fréquence variable **VLFV**

Les scénarios cités précédemment ont été choisis vu la diversité des cas possibles, pour le cas des fréquences fixes, on s'est basé sur les fréquences suivantes : 1 GHz, 1.5 GHz, 1.7 GHz et 2.2 GHz.

Dans toutes les expérimentations de ce modèle on s'est basé sur une vidéo de 720*302 de résolution avec 25000 Trames en utilisant le codec MPEG-4 Vidéo sur une durée de 45 min. On a répeté la même expérimentation avec différentes fréquences de processeur (fixées au préalable) et avec des niveaux de batterie différents.

Dans la section 6.1.4 on a procédé à l'analyse et à la génération des équations mathématiques correspondantes afin de définir le modèle de consommation énergétique. Après le test du modèle, on a élaboré un classement avec des modèles déjà

existants qui traitent la même problématique.

6.1.3 Description et déroulement des expérimentations

Les mesures de cette expérimentation ont été effectuées en se basant sur le scenario d'une Vidéo Locale avec Fréquence Fixe (VLFF). Les informations recueillies sont présentées dans le tableau 6.1.

Les variables utilisées pour établir les prédictions relatives au modèle VLFF sont présentées en gras, les mesures ont été effectués toutes les 100 millisecondes.

Grandeurs mesurées
Charge totale par CPU
Mémoire utilisée (Memory usage)
Degré de luminosité
Fréquence CPU
Niveau batterie (Battery remaining)
Puissance de batterie (Batterie Power)

Table 6.1 – Grandeurs visées pendant les expérimentations

Les applications présentes lors de ces mesures sont présentées dans le tableau 6.2.

Applications	
Facebook	GsmaService
Appli Google	Horloge
Avast Mobile Security	Cronoid
CPU Frequency	Google Play Store
Launcher3	Power Battery
Services Google Play	Hangouts
Stockage multimédia	Google Play Musique
Play-Fi	Clavier Google
MusicFX	Configuration du réseau mobile
SmartcardService	Interface du système
Messenger	Paramètres
VLC	Cron Tasker Free
Configuration des partenaires Google	

Table 6.2 – Applications présentes durant les expérimentations

6.1.4 Résultats des expérimentations

Chaque expérimentation s'est déroulée en mode déconnecté en fixant la fréquence des quatre CPU. Les fréquences utilisées pour les différentes mesures sont : 1 GHz, 1.5 GHz, 1.7 GHz et 2.2 GHz.

L'objectif est de déterminer le type de corrélation existante entre le nombre d'opérations totales et l'énergie élémentaire dissipée sous une fréquence fixée auparavant.

Le but de cette partie consiste à évaluer l'énergie dissipée par cycle d'horloge qu'on nommera par la suite « énergie élémentaire dissipée ». Un cycle d'horloge correspond au battement d'un microprocesseur, chaque instruction nécéssite au moins un cycle d'horloge pour s'exécuter.

Pour ce faire, on calcule le nombre d'opérations totales en effectuant la somme des cycles actifs de chaque processeur par intervalle de temps (100 ms), on effectue par la suite une regression linéaire de la somme des cycle actifs par rapport à l'énergie élémentaire dissipée.

6.1.5 Description du modèle mathématique proposé

Pour qualifier la qualité du modèle on s'est basé sur la valeur du coefficient de détermination qui représente le carré du coefficient de corrélation linéaire R qui se calcule comme suit :

$$R^2 = \frac{\sum_{i=1}^{n}(\hat{y}_i - \bar{y})^2}{\sum_{i=1}^{n}(y_i - \bar{y})^2} = 1 - \frac{\sum_{i=1}^{n}(y_i - \hat{y}_i)^2}{\sum_{i=1}^{n}(y_i - \bar{y})^2} \in [0, 1]$$

avec y_i pour chaque i $\in [1, 4]$ y etant la valeur des mesures, $\hat{y}_i$ les valeurs prédites et $\bar{y}$ la moyenne des mesures.

Les expérimentations ont été effectuées à plusieurs reprises sous différentes fréquences. La consommation moyenne par cycle est obtenue en appliquant une régression linéaire sur le nombre d'opérations totales des différents CPU et l'énergie dissipée correspondante.

La consommation moyenne par cycle peut s'exprimer ainsi :

$$E_{\text{cycle}} = \text{Regression\_Linéaire}(\sum_{i=1}^{4} \text{ca}_i, E_{\text{dissipé}})$$

où E_{cycle} représente l'énergie totale par cycle, ca_i le nombre de cycles où le processeur i est actif sur l'intervalle de mesure et $E_{\text{dissipé}}$ correspond à l'énergie dissipée mesurée expérimentalement sur ce même intervalle.

Le récapitulatif des résultats relatifs à l'énergie moyenne par cycle est présenté dans le tableau 6.3. qui montre que l'énergie moyenne consommée par cycle augmente proportionnellement à la fréquence. Pour ce mode d'expérimentation (mode déconnecté) on constate que la charge initiale de la batterie n'influence pas la consommation

d'énergie (ou l'influence d'une manière négligeable), c'est la raison pour laquelle on pourra l'ignorer dans la suite.

Fréquence (GHz)	**Niveau de batterie initial (%)**	**Énergie consommée / cycle (J)**
1	78	1.959e-11
1.5	20	4.236e-11
1.5	35	4.156e-11
1.5	89	4.232e-11
1.7	7	5.061e-11
1.7	12	5.061e-11
1.7	26	5.103e-11
1.7	28	5.070e-11
2.2	58	7.616e-11
2.2	95	7.014e-11
2.2	80	7.014e-11

Table 6.3 – Consommation énergétique par cycle d'horloge

Le tableau 6.3 montre que la consommation d'énergie par cycle ne dépend pas du niveau initial de la batterie mais uniquement de la fréquence du CPU.

Les différentes mesures relatives à l'énergie dissipée ont été effectuées par rapport au nombre d'opérations totales sous une fréquence fixe, le graphe 6.1 nous montre un exemple d'expérimentation en fixant la fréquence du processeur à 1.5GHz.

Les graphes correspondant aux fréquences 1 GHz, 1.7 GHz et 2.2 GHz se présentent d'une manière similaire (figure 6.2).

6.1.6 Pertinance du modèle

En se basant sur les résultats expérimentaux obtenus, on peut déduire l'allure correspondante à la décharge énergétique par unité de temps (100 ms) lors d'un lancement d'une vidéo locale avec une fréquence fixée au préalable. Le comportement de décharge énergétique est quasi linéaire, l'energie dynamique relative au modèle VLFF est présentée dans le graphe 6.3.

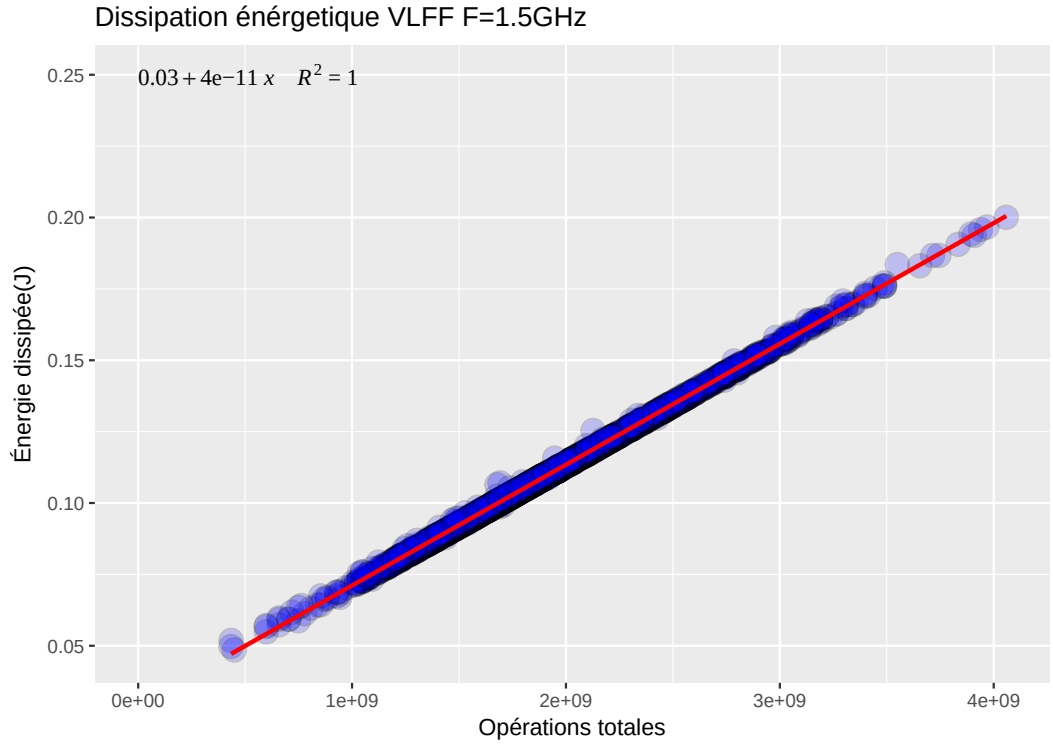

Figure 6.1 – Test VLFF avec fréquence = 1.5GHz. Sigification : La consommation d'énergie croît linéairement avec le nombre d'oprérations par cycle, les deux variables sont directement proportionnelles.

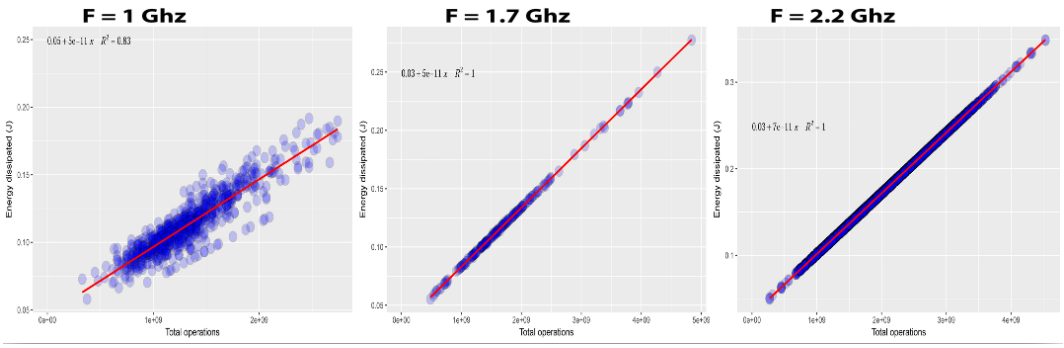

Figure 6.2 – VLFF sous fréquences = 1.0 GHz, 1.7GHz, 2.2 GHz

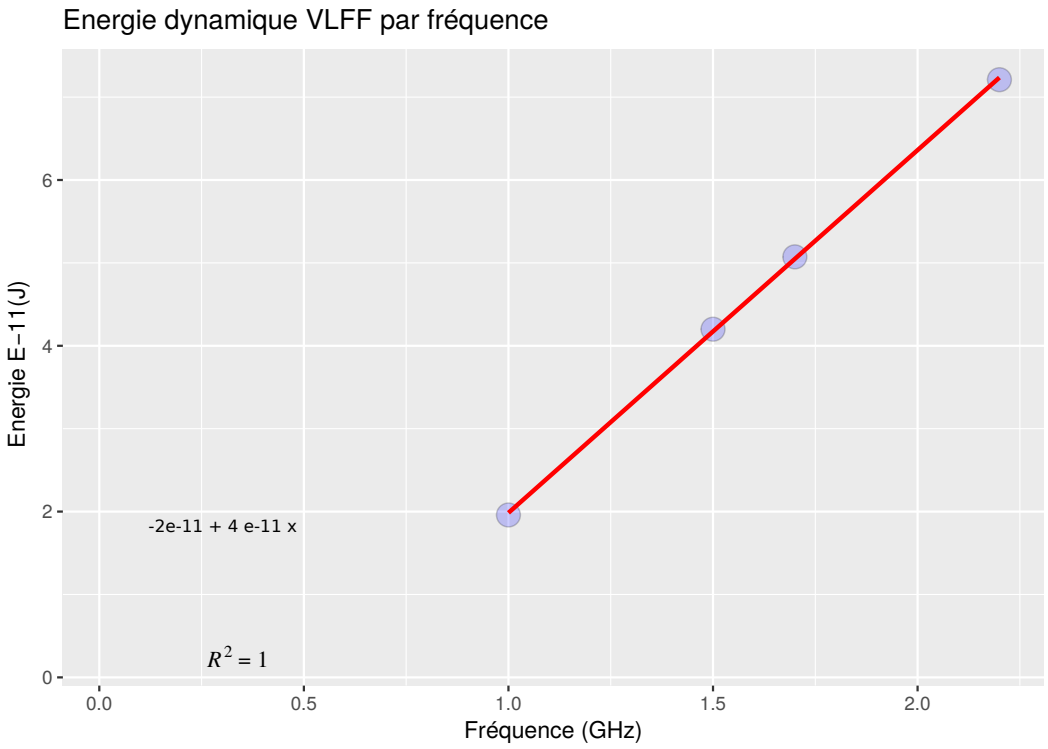

Figure 6.3 – Énergie dynamique du modèle VLFF

Modèle proposé

Le modèle mathématique qui opère le comportement énergétique du scénario VLFF correspond aux coefficients relatifs au graphe obtenu qui se présentent ainsi :

$$E(F, Op) = (A \times F + b) \times Op + (C \times F + d)$$

où E correspond à l'énergie totale, F la fréquence selectionnée, Op le nombre d'operation par intervalle de mesure (dans notre cas 100 ms), $A \times F + b$ correspond à l'énergie dynamique, $C \times F + d$ à l'énérgie statique.

La variation de l'enérgie statique par rapport aux différentes fréquences traitées dans le scénario VLFF est directement proportionnelle à la fréquence, elle croît d'une manière quasiment linéaire.

La figure 6.4 présente la moyenne relative à l'énergie statique relative au modèle VLFF.

Pour de notre cas d'étude, les coefficients relatifs au scénario VLFF via la régression linéaire entre la moyenne d'énergie dissipée par rapport au nombre d'opérations totales sont définies par l'équation suivante :

$$E(F, Op) = (4,008.10^{-11} \times F - 2,101.10^{-11}) \times Op + (0,0291 \times F - 0,00899)$$

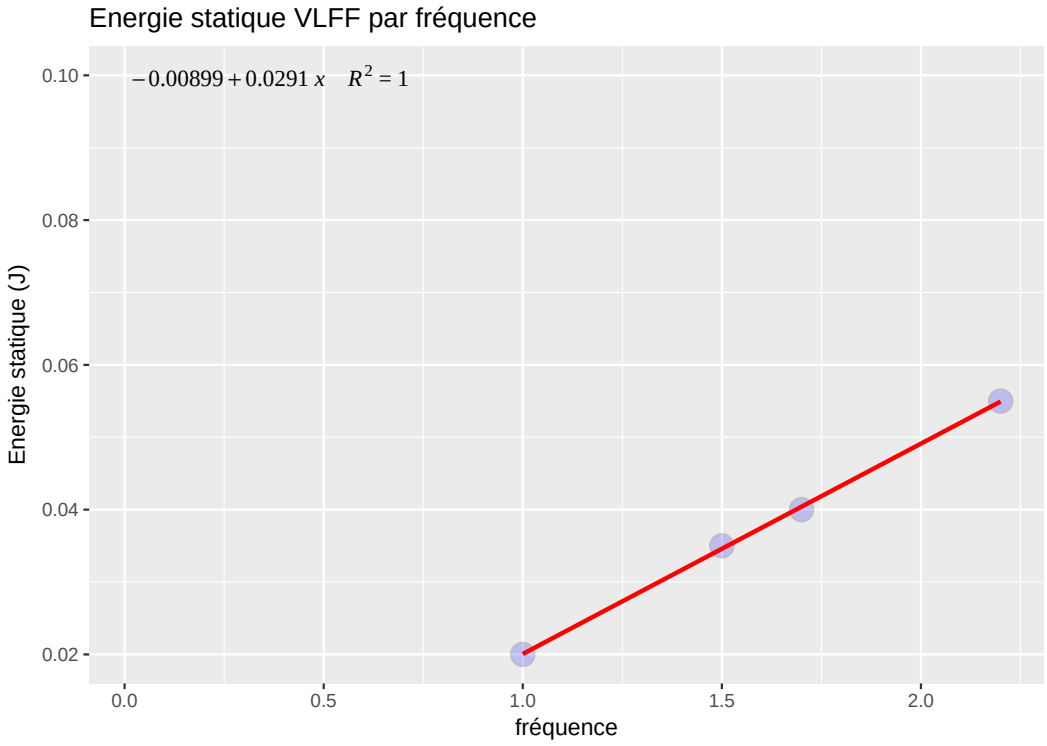

Figure 6.4 – Variation de l'énergie statique du modèle VLFF. Sigification : La consommation d'énergie statique est directement proportionnelle à la fréquence. Elle croît quasi-linéairement en passant d'une fréquence donnée à une fréquence superieure.

6.1.7 Validation du modèle proposé

L'objectif de cette étude consiste à faire une confrontation du modèle avec la variation dynamique de la fréquence pour voir si le modèle VLFF peut être étendu vers le cas d'une fréquence variable. Pour une meilleure évaluation, les expérimentations sont effectuées dans les même contexte du modèle VLFF (Déroulement d'expérimentation, Environnement, mesures...).

Pour vérifier le modèle construit précédemment, on se basera sur une expérimentation qui n'a pas servi pour l'élaboration du modèle, on dispose des résultats relatifs à l'énergie dissipée (mesurées expérimentalement) puis on applique les coeficients de l'équation relative au modèle proposé pour le scénarios VLFF.

Les expérimentations qui serviront au test sont composées de 45945 mesures. La comparaison des grandeurs prédites par un modèle mathématique et les mesures obtenue expérimentalement permettent de définir la précision du modèle prédit.

Dans un premier temps, nous avons lancé une mesure sous une fréquence variable (figure 6.5) puis nous avons comparé les résultats obtenus en utilisant le modèle réalisé pour les fréquences fixes pour la validation du modèle :

$$E(F, Op) = (4,008.10^{-11} \times F - 2,101.10^{-11}) \times Op + (0,0291 \times F - 0,00899)$$

Toutes les expérimentations étaient effectuées en mode déconnecté et sans fixer la fréquence des quatre CPU, afin d'obtenir des résultats concrets, on s'est basé sur le mode par défaut de fréquence proposé par le système. Le but de cette partie est de déterminer le type de corrélation existante entre la consommation énergétique d'une opération élémentaire. Pour ce faire, on s'est basé sur un scenario avec une fréquence variable. Les résultats obtenus expérimentalement sont comparés avec ceux obtenus par le modèle proposé.

Pour tester le modèle VLFF, on s'est referé aux résultats expérimenteaux obtenus avec le cas d'étude VLFV. Par la suite on a appliqué les parametres du moddèle (obtenus mathématiquement) dans le scénario VLFF (voir § 6.1.6).

La différence enérgétique existante entre l'énergie mesurée experimentalement et celle prédite par le modèle correspond à l'erreur relative du modèle, elle est présentée dans la figure 6.6. Après comparaison, on constate qu'il y a une erreur relative relativement petite.

Pour une meilleur évaluation, nous avons calculé le rapport entre l'énergie mesurée expérimentalement et celle prédite par le modèle mathématique proposé. La figure 6.7 montre qu'il y a une bonne concordance entre le résultat prédit et le résultat expérimental ce qui prouve l'efficacité du modèle énergétique.

Au début de l'expérimentation, on remarque qu'il y a un pic énergétique qui se stabilise après une dizaines de secondes. La partie agrandie (contournée en rouge) montre que les mesures prédites par le modèle sont très proches du cas idéal qui

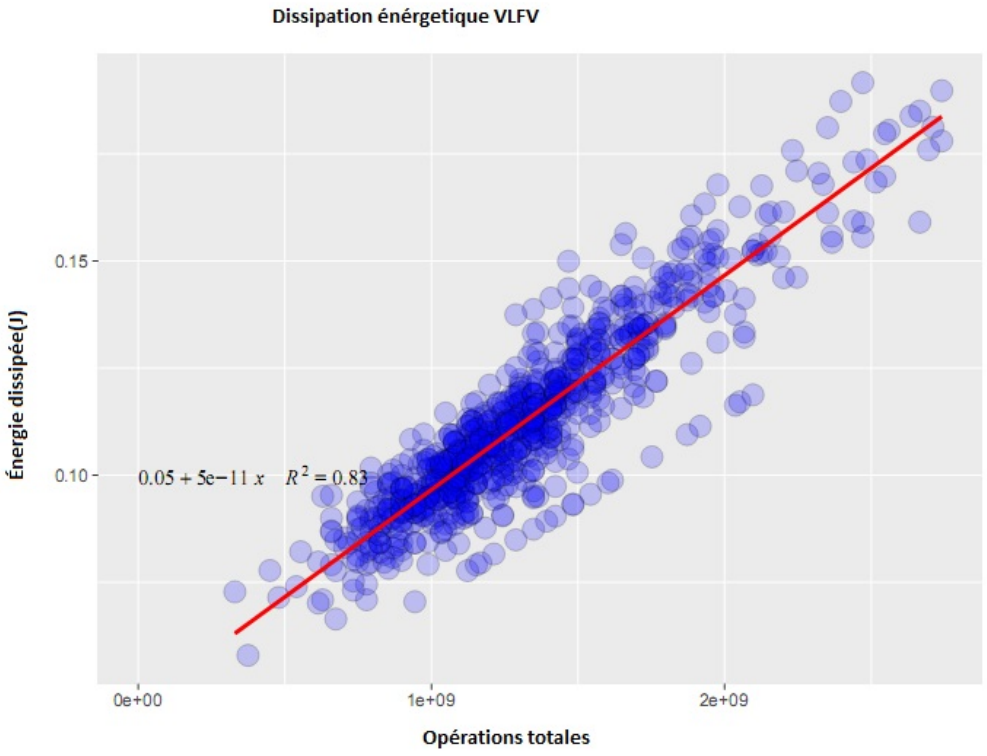

Figure 6.5 – Test VLFV. Signification : Les opérations totales sont calculées en se basant sur la fréquence optimale évaluée par le système, elle est relative au scénario, pour le cas du VLFV, la fréquence optimale correspond à la valeur : 1.7GHz.

est un rapport égale à 1 (cas dans lequel les mesures obtenues experimentalement sont identiques à celles prédites par le modèle proposé) ce qui prouve l'efficacité du modèle proposé.

L'erreur relative moyenne entre notre modèle et les mesures expérimentales est de 1,767 % avec un écart-type de 3,652 %. En outre, 93,9 % des mesures obtenues à partir d'un taux d'erreur inférieur à 8 % et 64,73 % des mesures obtenues ont un taux d'erreur inférieur à 5 %. Pour montrer l'efficacité du modèle nous avons effectué la somme cumulée des différentes quantités d'énergies obtenue pour les mesures effectuées sur le terrain ainsi que celles prédites via le modèle dans le but de bien visualiser la différence (l'erreur) existante entre les deux mesures. Pour mieux présenter notre modèle, on a calculé le ratio entre l'énergie générée par le modèle et l'énergie mesurée expérimentalement afin de connaître le comportement du modèle

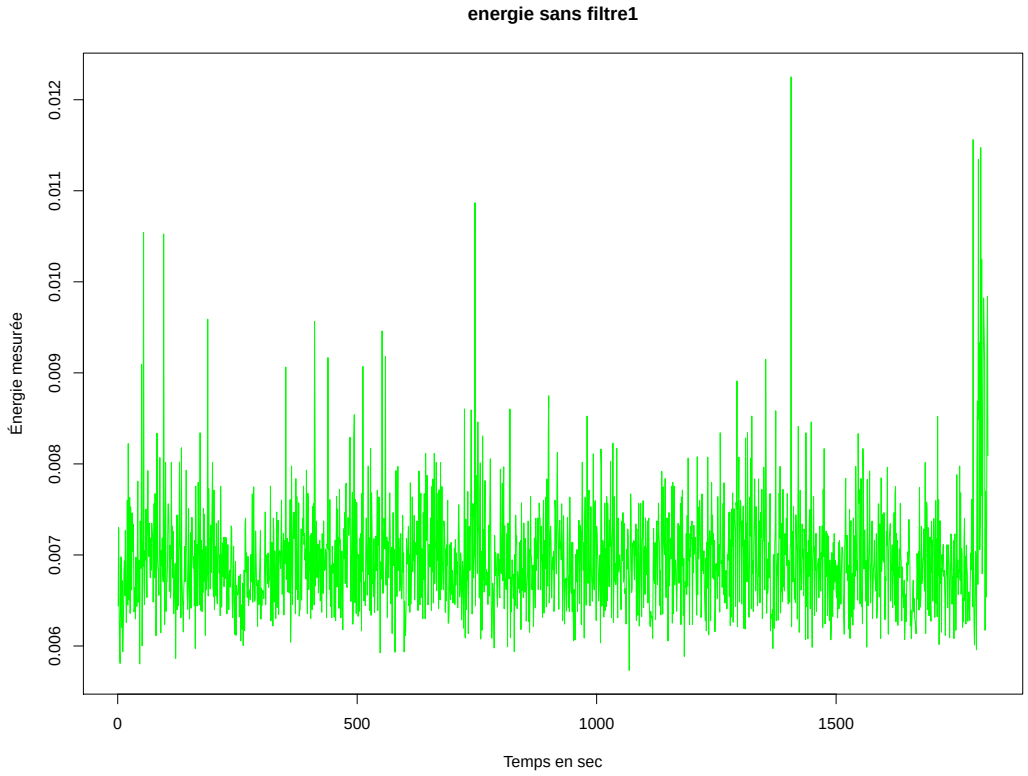

Figure 6.6 – Différence entre l'enérgie mesurée et l'energie prédite par le modèle

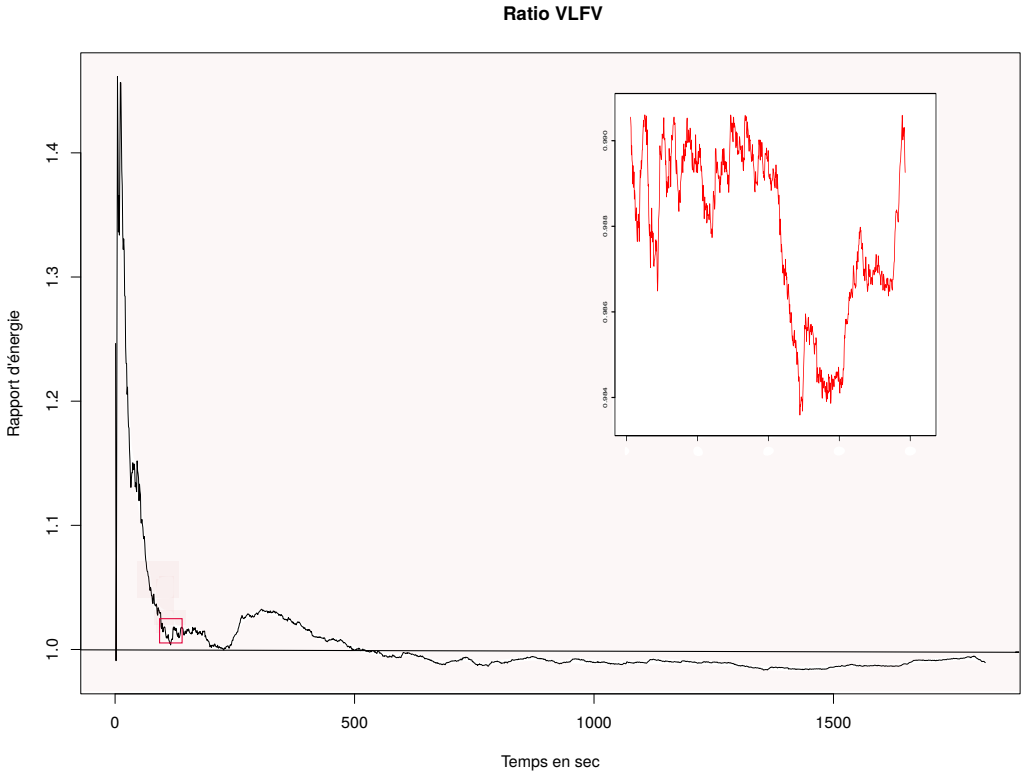

Figure 6.7 – Rapport énergétique du Modèle VLFV. Signification : Le rapport entre l'énergie mesurée experimentalement et celle prédite par le modèle proposé montre l'existance d'une forte proportionnalité, la partie présentée en rouge correspond à un échantillon de mesure qui montre que le rapport existant est très proche du cas idéal.

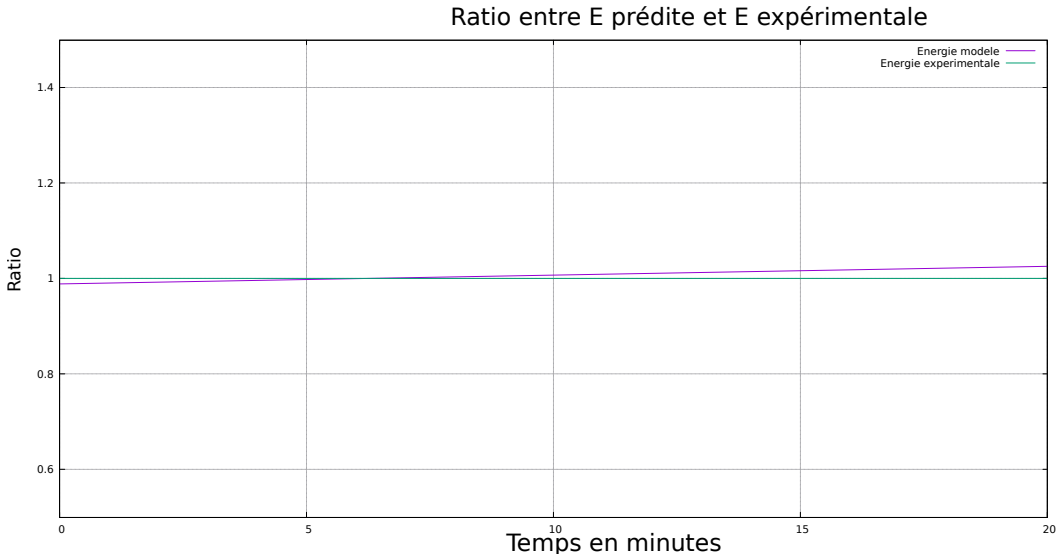

Figure 6.8 – Rapport entre l'énergie mesurée et l'énergie estimée. Signification : l'énergie du modèle est légèrement surestimée.

(voir figure 6.8).

D'après les résultats obtenus, on constate que notre modèle présente une sous estimation qui atteint 0.98 (1 étant le cas idéal des mesures) dans la première partie des expérimentations et une sur estimation après le 1/3 du temps total des mesures qui atteint un maximum de 1.1.

Pour mieux concrétiser notre étude, nous avons élaboré la présentation de distribution des sommes cumulées des erreurs relatives obtenues dans le modèle. Le résultat obtenu affichera les différentes classes d'erreur, l'axe vertical est normalisé selon le nombre de classe existantes, 1 correspond à la plus grande classe (celle qui contient le plus grand nombre d'appartenance). La figure 6.9 montre la corrélation existante entre l'énergie mesurée et celle prédite par le modèle en se basant sur l'erreur relative cumulée des differentes mesures effectuées, on obtient une courbe en cloche dans la quelle on remarque que le taux d'erreur relatif est faible ce qui justifie le choix des parametres de notre modèle. D'après les résultats obtenus précédemment, on peut déduire que le modèle proposé pour le cas d'une vidéo locale avec fréquence fixe (VLFF) peut être étendu pour évaluer le cas d'une vidéo locale avec fréquence variable (VLFV) tout en étant dans le mode déconnecté. Dans la section suivante on se focalisera sur le cas du mode connecté dans lequel on fera recours à la 3G, 4G, le GPS et le Wi-Fi afin de traiter leurs impact sur l'efficacité énergétique.

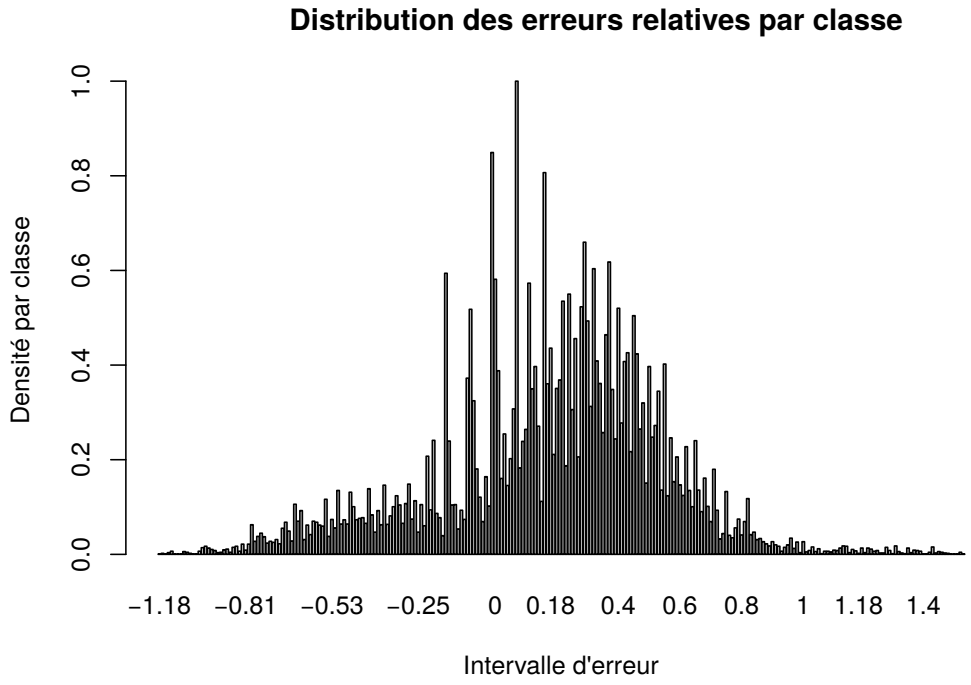

Figure 6.9 – Fréquence de présence par classe

6.2 Cas du mode connecté

6.2.1 Description du mode connecté

Dans le mode connecté, les sources relatives à internet sont activées complètement ou partiellement selon les scénarios en question. Le but de ce mode est de faire le suivi de la consommation énergétique dans un état connecté, ainsi que la comparaison avec des scénarios déjà traités en mode déconnecté afin de déduire la variation énergétique relative à plusieurs scénarios similaires dans des modes (connecté/déconnecté) différents. Notre étude a porté sur les scénarios liés à la vidéo (en mode déconnecté et connecté), et les scénarios relatifs à la navigation en mode connecté vu leur influence importante sur l'autonomie de la batterie :

— Vidéo distante avec fréquence fixe **VDFF**.
— Navigation avec fréquence fixe **NFF**.
— Navigation avec fréquence variable **NFV**.

Les scénarios cités précédemment ont été choisis vu l'existence de différents cas possibles, pour le cas des fréquences fixes nos expérimentations ont portées sur les fréquences suivantes : 1 GHz, 1.5 GHz, 1.7 GHz et 2.2 GHz.

L'objectif de cette étude est d'exploiter la variation des mesures dans les différents scénarios afin d'analyser l'impact de la connectivité ainsi que les conditions environnementales (météo, température, pression, connectivité...) sur la consommation énergétique selon les différentes conditions.

6.2.2 Cas d'une vidéo Distante avec fréquence fixe : VDFF

Le but de cette partie consiste à faire l'étude de consommation énergétique tout en activant le Wi-Fi. Pour rester dans le même contexte, la vidéo distante était identique à celle utilisée en mode déconnecté. Nous avons utilisé YouTube qui est l'un des serveurs vidéo en ligne les plus populaires. En 2011, YouTube a été reporté au compte pour 10 % du trafic Internet en Amérique du Nord. Ce trafic est livré sur TCP en utilisant le téléchargement HTTP progressif. La vidéo est livrée just-in-time au lecteur vidéo, alors lorsque l'utilisateur annule une vidéo, seule une quantité limitée de données est ignorée.

Les différentes expérimentations se sont déroulées en mode connecté en lançant la vidéo distante via YouTube et en fixant la fréquence des quatre CPU. Les fréquences utilisées pour les différentes mesures sont : 1 GHz, 1.5 GHz, 1.7 GHz, 2.2 GHz.

Toutes les expérimentations étaient effectuées à partir de la fréquence 1 GHz vu que la qualité de service des mesures effectuées sous les fréquences basses (652 MHz, 730 MHz) étaient très dégradées (taille limitée des ressources octroyées aux expérimentations).

L'objectif de cette partie consiste à déterminer le type de corrélation existante entre la consommation énergétique du nombre d'opérations totales (par intervalle de

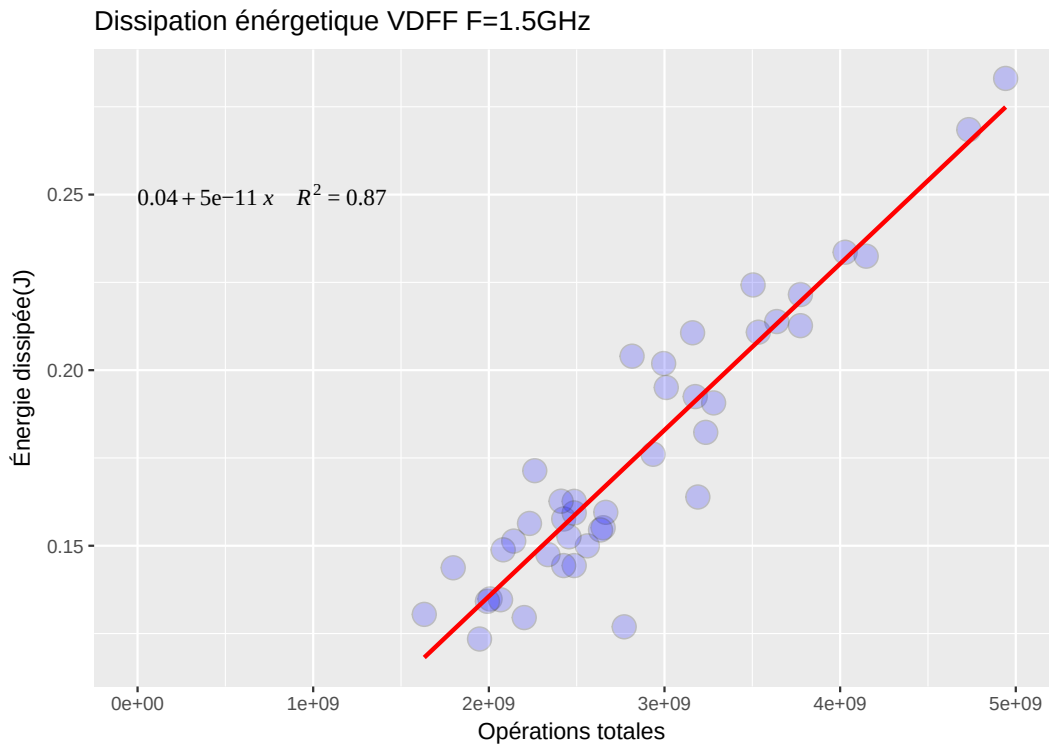

Figure 6.10 – Test VDFF avec fréquence = 1.5GHz

mesure) et l'énergie dissipée correspondante.

Pour le cas d'une vidéo distante avec fréquence fixe (VDFF), les expérimentations étaient effectuées en activant la Wi-Fi tout en étant en mouvement (déplacement dans zones qui n'ont pas forcément la même force du signal). Le graphe 6.10 montre que la modélisation du scénario VDFF en fixant la fréquence à 1.5 GHz peut être également prédite via une régression linéaire.

Les figures 6.11 et 6.12 montrent respectivement que l'énergie statique et l'énergie dynamique du modèle VDFF varient linéairement par rapport à la fréquence. On remarque que les scénarios du modèle VDFF peuvent également être modélisés par une régression linéaire, l'équation suivante définit la proportionnalité existante entre la variation énergétique et la fréquence :

$$E(F, Op) = (5,002.10^{-11} \times F - 2,104.10^{-11}) \times Op + (0,0297 \times F - 0,0295)$$

Figure 6.11 – Variation de l'énergie statique du modèle VDFF. Signification : La consommation d'énergie statique est directement proportionnelle à la fréquence, elle croît quasi-linéairement en passant d'une fréquence donnée à une fréquence plus élevée.

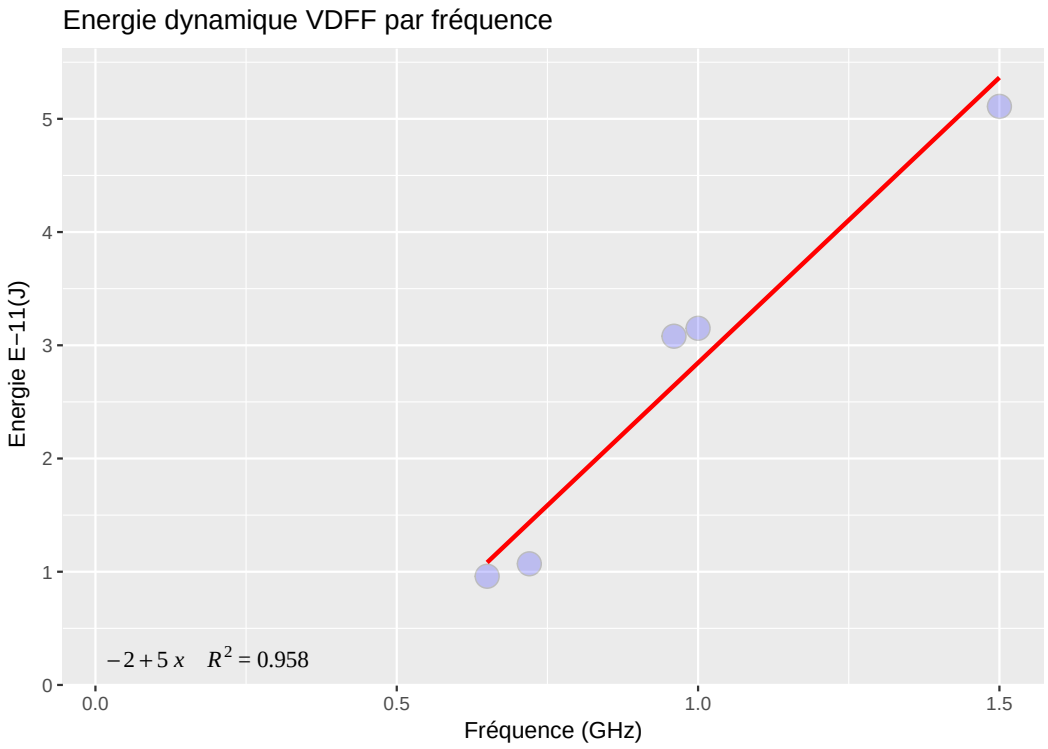

Figure 6.12 – Énergie dynamique du modèle VDFF

Étude comparative entre les scénarios VLFF et VDFF

Dans cette partie on traitera l'écart énergétique existant entre le scénario VLFF et VDFF réalisés dans des conditions similaires (même fréquence).

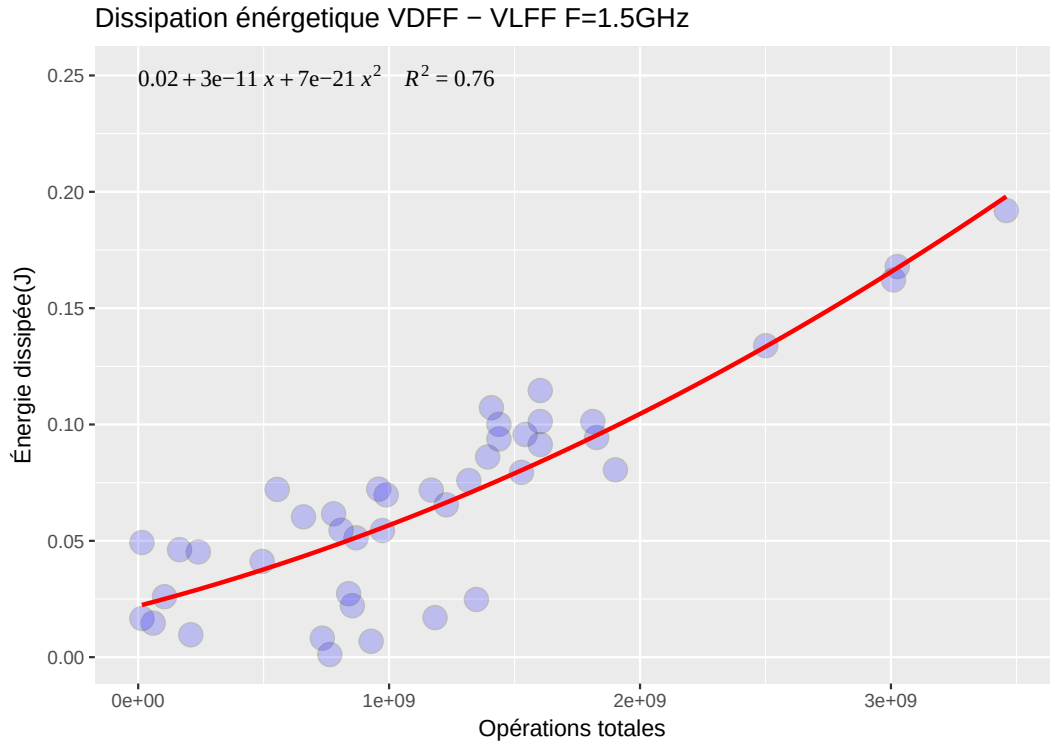

Figure 6.13 – Variation énergétique entres les modèles VDFF et VLFF avec une fréquence = 1.5 GHz due au coût de communication

La variation énergétique mesurée expérimentalement entre les modèles VLFF et VLFV est détaillée dans la figure 6.13.

On remarque que le taux de dissipation énergétique relatif au scénario VDFF est légèrement supérieur à celui relatif au scénario VLFF dans le cas du mode déconnecté vu la charge supplémentaire existante dans le cas connecté. La consommation liée à la communication est bien modélisée selon une regression quadratique qui donne un meilleur resultat en terme de pertinence et de simplicité.

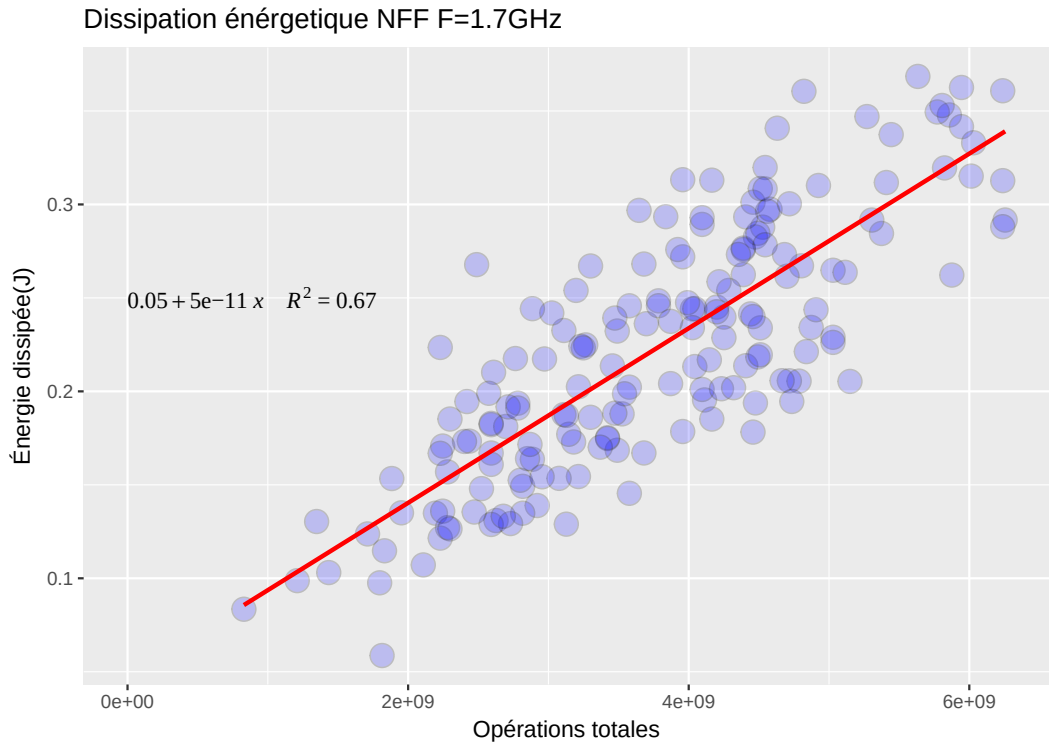

Figure 6.14 – Test NFF avec fréquence fixe, F = 1.7 GHz

6.2.3 Cas de navigation avec fréquence fixe : NFF

Le cas de navigation consiste à activer la Wi-Fi tout en activant le GPS afin de suivre le comportement énergétique dans une période donnée. Le premier cas consiste à fixer la fréquence et de faire le suivi de l'énergie dissipée totale des applications actives séparément dans un premier temps, puis la consommation globale du scénario NFF. Le tableau 6.4 montre que la navigation utilise énormément de ressources CPU (19.14 %) qui sont directement proportionnelles à la consommation totale contrairement aux autres applications du système qui varient entre 0.0000038 % et 3.93 %.

Les mesures étaient effectuées par rapport au nombre total d'opérations sous une fréquence fixe. Le graphe 6.14 montre un exemple d'expérimentation relatif au modèle NFF en fixant la fréquence du processeur à 1.7 GHz.

L'énergie statique (figure 6.15) et l'énergie dynamique (figure 6.16) sont modélisées selon une variation linéaire par rapport à la fréquence. La consommation d'énergie en fonction de l'activité des traitements pour les fréquences fixes est modélisée selon une

Applications	**CPU [%]**	**Average Virtual Memory Size [MB]**	**Max Virtual Memory Size [MB]**	**Average Resident Set Size [MB]**	**Max Resident Set Size [MB]**
Maps	**19.14**	**1994.1**	**2030.89**	**59.35**	**73.02**
Paramètres	3.93	10292.46	12446.72	72.63	104.7
Messenger	0.75	1801.79	1844.93	36.94	46.2
Interface du système	0.45	1704.438	1770.12	34.41	40.764
Services Google Play	0.43	4961.427	6588.196	44.232	56.596
Power Battery	0.4	1975.87	3121.44	13.468	28.587
Cronoid	0.38	1508.997	1522.89	7.92	8.562
Configuration du réseau mobile	0.24	3033.38	3049.16	16.254	24.316
Google Play Musique	0.23	1640.54	1640.8	13.263	13.28
Avast Mobile Security	0.19	1522.28	1543.37	11.502	13.21
Facebook	0.09	1807.975	1867.42	29.93	35.51
Synthèse vocale Google	0.075	1545.23	1546.46	15.43	17.945
Gmail	0.072	84.705	1666.868	0.646	17.82
Cron Tasker Free	0.03	1375.01	1491.5	7.36	8.695
Hangouts	0.022	11.909	1667.38	0.124	21.194
Google+	0.021	157.86	1544.60	1.13	11.647
Clavier Google	0.011	1551.848	1597.836	11.872	13.362
Stockage multimédia	0.008	235.45	1500.34	1.18	8.63
Drive	0.005	1.48	1537.04	0.011	12.33
Google Play Store	0.004	15.80	1545.80	8.33	9.59
Launcher3	0.004	1591.679	1760.28	14.196	26.402
Appli Google	0.004	3094.838	3098.244	16.064	17.605
HP Touchpoint Manager	0.003	6.341	1506.628	0.037	9.13
SmartcardService	0.003	2977.31	2990.26	13.763	14.187
Horloge	0.002	495.5	1495.87	2.69	8.339
Keep	0.002	0 2.83	1514.86	0.017	9.426
Google Play	0.001	3.533	1558.828	0.029	12.974
Agenda	0.001	43.74	1524.6	0.272	9.48
Play-Fi	4.41E-4	1486.06	1486.06	5.42	5.52
Stockage de l'agenda	3.82E-4	1.858	1485.504	0.008	6.63
GsmaService	3.63E-4	1469.999	1484.568	5.638	5.8
Lanceur d'applications	1.48E-4	0 1.68	1499.21	0.007	7.43
CPU Frequency	5.51E-5	606.2	1489.42	2.79	7.64
Firefox	5.38E-5	4.63	1505.31	0.023	7.642

Table 6.4 – Répartition des ressources systeme pour le scénario NFF. Signification des colonnes : **Application** : Les applications présentes pendant l'experimentation, **CPU**[%] : Pourcentage d'utilisation des ressources CPU, **Average Virtual Memory Size** [**MB**] : Utilisation moyenne de la memoire virtuelle par application, **Max Virtual Memory Size** [**MB**] : Utilisation maximale de la memoire virtuelle par application, **Average Resident Set Size**[**MB**] : Utilisation moyenne de la memoire réelle par application, **Max Resident Set Size**[**MB**] : Utilisation maximale de la memoire réelle par application.

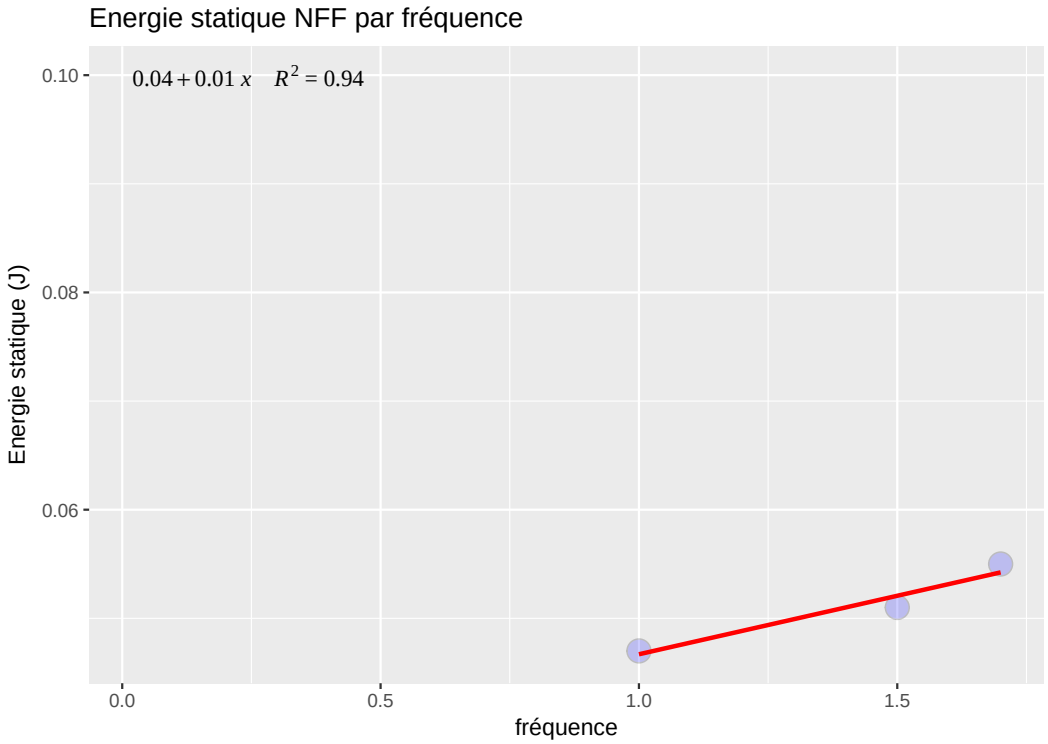

Figure 6.15 – Variation de l'énergie statique du modèle NFF. Signification : La consommation d'énergie statique est directement proportionnelle à la fréquence. Elle croît quasi-linéairement en passant d'une fréquence donnée à une fréquence plus élevée.

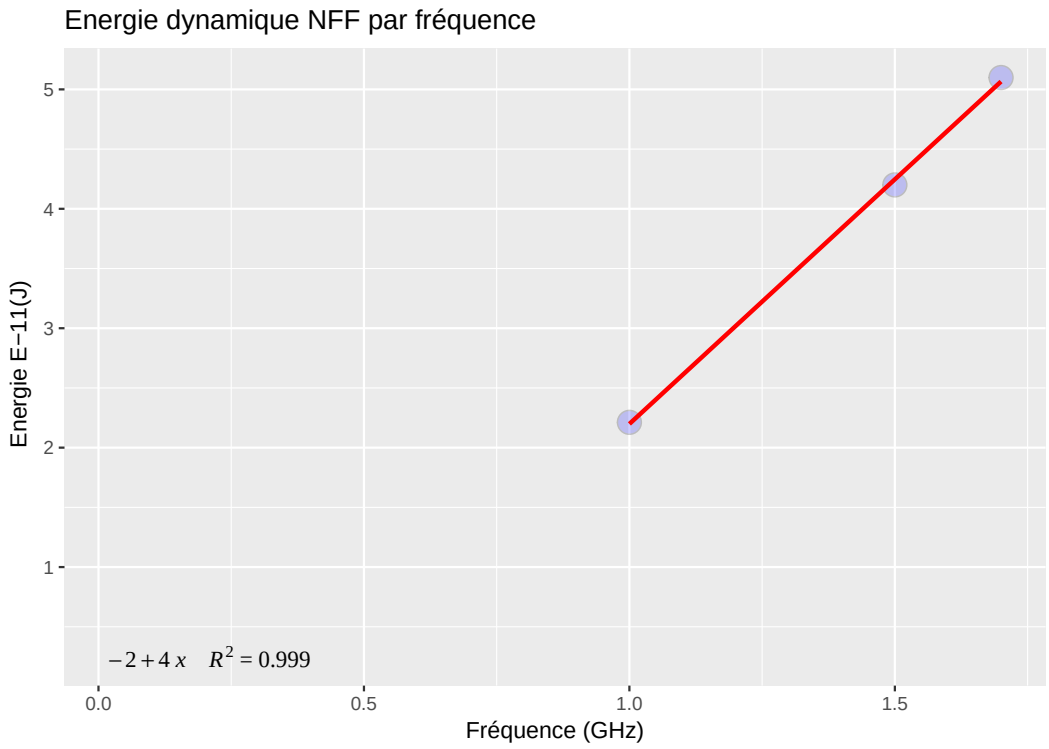

Figure 6.16 – Énergie dynamique du modèle NFF

régression (linéaire), elle est présentée suivant l'équation suivante qui est composée de l'énergie statique et l'énergie dynamique.

$$E_{\text{Statique}}(F) = 0.098 \times F - 0,0391, R^2 = 0.940$$
$$E_{\text{Dynamique}}(F) = 4.002.e^{-11} \times F - 2.103.e^{-11}, R^2 = 0.999$$

Pour notre étude de cas, les coefficients liés au scénario NFF sont donc définis par l'équation suivante :

$$E(F, Op) = (4,002.10^{-11} \times F - 2,103.10^{-11}) \times Op + (0,098 \times F - 0,0391)$$

6.2.4 Cas de navigation avec fréquence variable : NFV

Cette partie traite le cas de navigation avec fréquence variable (fréquence attribuée par le système) la figure 6.17 présente les résultats relatifs aux expérimentations dédiées au modèle NFV. On remarque l'existence d'un flot de données qui dépend fortement de la fréquence à chaque prise de mesure, le nombre élevé des opérations et le taux de charge expliquent la forme du graphe obtenue.

Étude comparative entre les scénarios NFF et NFV

Le but de cette section consiste à évaluer l'écart d'énergie existant entre les scénario NFF et NVF. l'énergie dissipée relative à la navigation en utilisant la géolocalisation pour les modèles NFF et NFV. Pour ce faire on s'est basé sur les exemples cités précédemment, en se basant sur le scénario NFF en fixant la fréquence à 1.7 GHz, et sur la fréquence variable gérée par le système pour le scénario NFV (figures 6.14 et 6.17). Pour comparer la variation d'énergie entre les deux scénarios, nous avons calculé la différence entre les mesures obtenues expérimentalement avec le scénario NVF et le scénario NFF afin de déterminer le coût dû à la fréquence variable.

La figure 6.18 illustre cette évaluation. La différence énergétique obtenue expérimentalement (en se basant sur les équations du modèle) 6.18 est très proche de celle calculée mathématiquement, elle se presente ainsi :

— NFF : $0.05 + 5.e^{-11}x$
— NFV : $0.07 + 2.e^{-11}x + 5.e^{-21}x^2$
— NFV - NFF : $0.02 - 3.e^{-11}x + 5.e^{-21}x^2$

Le résultat obtenu par le modèle est proche de celui obtenu experimentalement. Il s'exprime sous la forme : $0.03 + 2.e^{-11}x + 2.e^{-21}x^2$

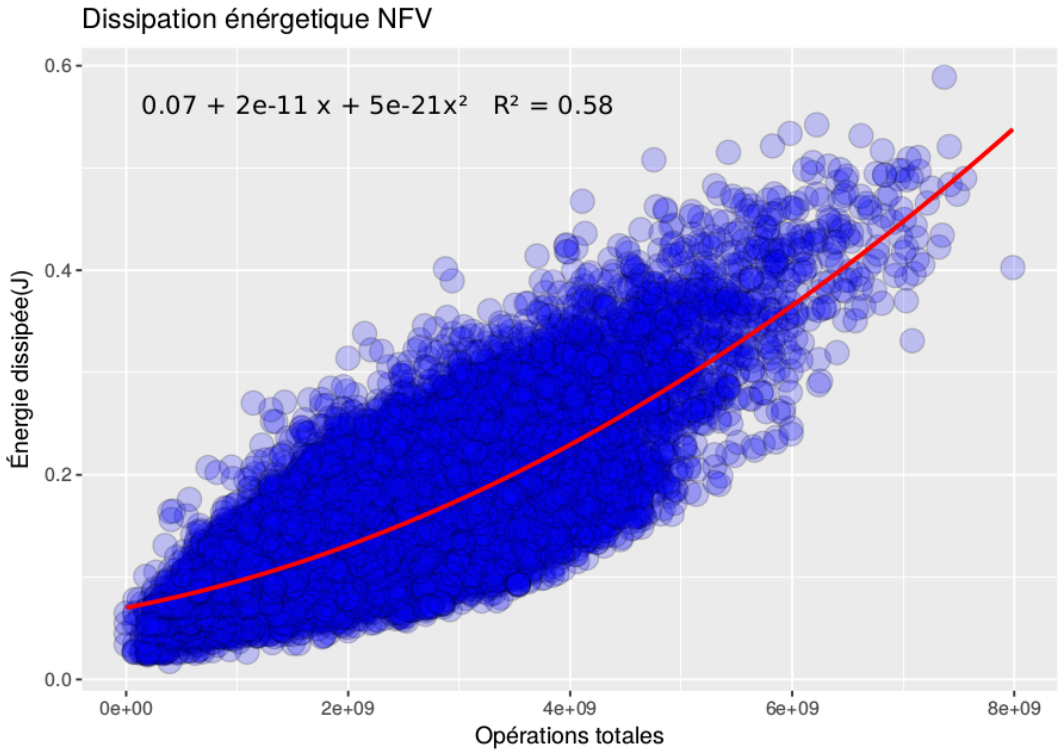

Figure 6.17 – Navigation avec fréquence variable

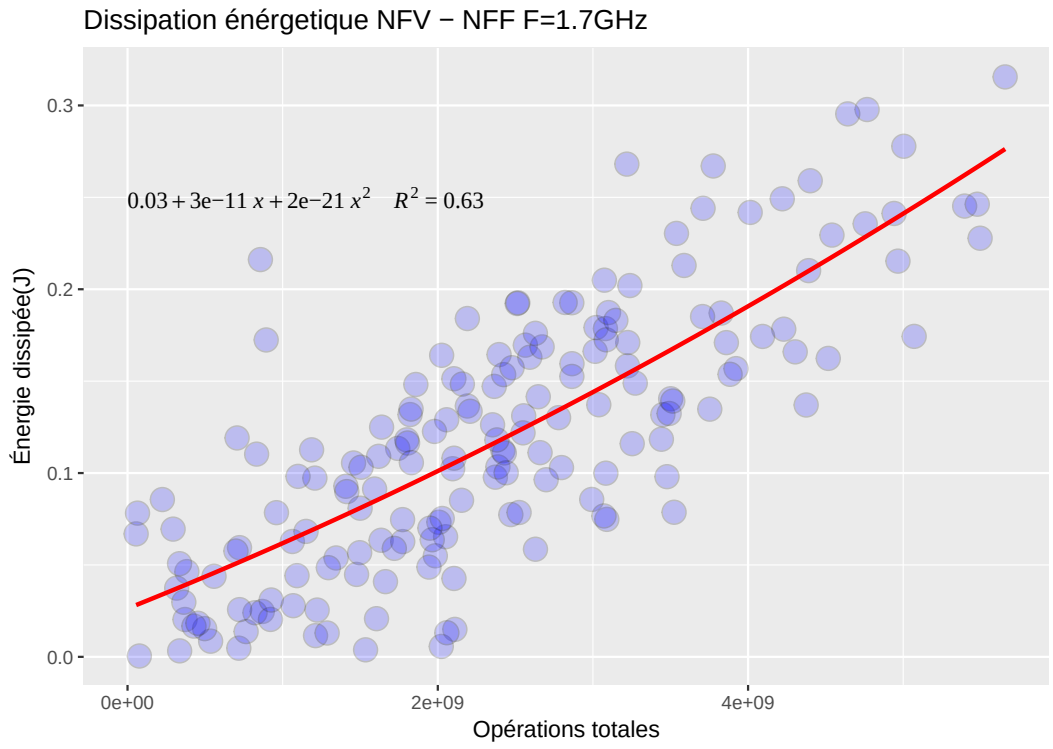

Figure 6.18 – Différence énergétique entre les scénarios de navigation (NFF et NFV)

La différence obtenue est due à la variabilité de la fréquence (énergie dynamique) plus la présence de l'énergie de la queue qui est relative aux dispositifs (Wi-Fi, GPS...) qui ne sont pas forcément actifs en permanence tout au long des expérimentations.

Dans cette étude on a traité les différents types de dépendances existantes entre la dissipation énergétique et la fréquence de plusieurs scénarios dans le mode déconnecté (VLFF) et le mode connecté (VDFF, NFF, NFV). La différence existante entre les différents scénarios peut s'expliquer d'une part à la variabilité de la force du signal reçu pour le GPS et la Wi-Fi (Pour le mode connecté) qui force l'activation/désactivation du/des services qui va générer une consommation supplémentaire due à l'énergie de la queue, d'autre part les scénarios basés sur le mode connecté (VDFF, NFF, NFV) nécessite l'activation de/des dispositifs supplémentaires telle-que le GPS, ce qui augmente le taux des entrées/sorties qui sont responsables partiellement du surcoût énergétique.

6.3 Étude comparative entre le modèle proposé et les modèles de référence

Dans cette section on va faire appel aux différents modèles de consommation énergétique afin de se positionner par rapport aux modèles les plus courants dans le domaine.

6.3.1 Rappel des modèles de référence

— **TailEnder** (voir page 57) a pour but la minimisation de la consommation d'énergie pour les applications qui peuvent tolérer un léger retard tels que les E-mails.
— **Cinder** (voir page 58) est conçu pour les téléphones et appareils mobiles, il permet aux utilisateurs et aux applications de commander et gérer les ressources limitées de l'appareil tel que l'énergie.
— **EPROF** (voir page 60) est un « Profiler » d'énergie fine pour les applications des smartphones. EPROF a la capacité d'analyser l'état de la demande d'énergie asynchrone, la modélisation des caractéristiques énergétiques, l'état de la queue des composants.
— **RAPL** (Running Average Power Limit, voir page 62) est un modèle qui permet de faire une comparaison des applications mobiles des deux secteurs et montre qu'elles révèlent une consommation d'énergie différente, tout en offrant des services similaires.
— **AppScope** (voir page 64) a pour objectif de mesurer l'énergie d'application pour les systèmes applications qui utilisent des modèles d'alimentation de matériel et les statistiques d'utilisation pour chaque composant matériel. Il

permet d'évaluer automatiquement la consommation d'énergie des applications fonctionnant sur les smartphones Android.

6.3.2 Positionnement du modèle proposé

La recherche présentée propose une méthodologie qui permet la modélisation et l'évaluation des coûts énergétiques dans les environnements mobiles. Le but de ce travail est de faire le suivi de la consommation d'énergie en agissant sur les paramètres suivants :

— Qualité du modèle énergétique,
— Prise en compte des optimisations matérielles,
— Prise en compte des optimisations logicielles,
— Impact sur la qualité de service,
— Aide au développeur.

Le modèle proposé peut être utilisé pour définir une fréquence optimale en termes de consommation d'énergie pour des situations précises sans trop dégrader la qualité de service souhaitée par l'utilisateur. La collecte des données liée à la variation énergétique était effectuée sur la base de plusieurs états sur différents lieux en fixant les métriques citées précédemment.

La qualité du modèle énergétique est évaluée par rapport à la précision de l'évaluation énergétique pour des applications spécifiques dans des appareils mobiles. Le symbole '+' correspondra aux modèles qui traitent la variation énergétique avec un simple degré de précision, '++' pour les modèles qui ont une précision plus élevée. La prise en compte des optimisations matérielles consiste à tirer davantage de performance en se basant sur les bons réglages du dispositif en question. La prise en compte des optimisations logicielles consiste à améliorer les performance et limiter sa consommation de ressources, '+' correspondra aux modèles qui veillent au bon suivi énergétique, '++' pour les modèles qui offrent des meilleurs performances. L'impact sur la qualité de service est exprimée par '++' pour les modèles qui affectent significativement la qualité de service et de '+' sinon. La prise en compte de la sécurité, concerne la conservation d'une quantité d'énergie pour les actions d'urgences (appel d'urgence...). L'aide au développeur est exprimé par '++' pour les modèles qui permettent aux développeurs de prendre des décisions pertinantes sur la gestion énergétique, '+' permet d'avoir des informations plus générales mais qui ne permettent pas la prise de décision.

Le tableau 6.5 englobe les critères pertinents vis-à-vis aux différents modèles qui traitent la problématique de la modélisation énergétique ainsi que la position de notre modèle par rapport aux modèles existants.

HBBLA (Hamzaoui, Boulet, Berrajaa, Lipari, Azizi) corresponds à notre modèle proposé pour la modélisation énergétique liées aux dispositifs mobiles.

Eprof et notre modèle HBBLA réalisent de meilleurs résultats pour la qualité

	Qualité	**Optim. hard**	**Optim. appli.**	**QoS**	**Sécurité**	**Dev.**
TailEnder	+	++	NA	+	NA	NA
Cinder	+	++	NA	++	+++	NA
Eprof	++	+	+	+	NA	++
RAPL	+	++	NA	+	+	+
AppScope	+	NA	+++	+	NA	+
HBBLA	*++*	*+*	*++*	*+*	*NA*	*+*

Table 6.5 – Comparaison des modèles d'estimation d'énergie. Signification des colonnes : **Qualité** : qualité du modèle énergétique, **Optim. hard** : prise en compte des optimisations matérielles, **Optim. appli.** : prise en compte des optimisations logicielles, **QoS** : impact sur la qualité de service, **Sécurité** : prise en compte de la sécurité, **Dev.** : aide au développeur.

du modèle énergétique et l'optimisation des performances. Eprof permet d'analyser l'état des demandes d'énergie et d'estimer le taux d'énergie de queue de certains composants. Notre modèle HBBLA permet de modéliser la consommation d'énergie afin qu'elle soit optimisée pour de meilleures performances.

TailEnder, Cinder, RAPL satisfont le critère Optim. TailEnder effectue une minimisation de la consommation d'énergie pour les applications qui tolèrent un léger retard en quantifiant l'utilisation de l'énergie, Cinder en allouant des ressources aux applications de guidage en visualisant le taux des ressources limitées. RAPL fait une comparaison énergétique d'un service similaire et montre que la consommation n'est pas identique.

Appscope et notre modèle donnent la possibilité d'optimiser des applications existantes. Appscope évalue automatiquement la consommation d'énergie des applications à un niveau microscopique, Il permet de détecter les événements pertinents au fonctionnement d'un composant pour une meilleure optimisation. Notre modèle permet de définir les paramètres optimaux pour une meilleure performance qui justifie leurs pertinences dans le critère Optim Application.

La qualité du service est bien gérée par Cinder car il gère automatiquement l'allocation des ressources énergétiques. Cinder satisfait également le côté sécurité, il permet d'octroyer plus de ressources aux appels d'urgence en cas de pénurie de ressources énergétiques.

Eprof satisfait le critère Dev car il peut aider les développeurs en analysant les différents états d'énergie des dispositifs (caractéristiques, énergie de la queue, etc). Nous avons déjà publié une étude plus détaillée sur les modèles de l'état de l'art dans le chapitre 4.

Afin de réaliser notre modèle, on a présenté une méthodologie basée sur des

mesures précises, en se basant sur la gestion du comportement énergétique observé dans les différents scénarios expérimentaux afin d'identifier les paramètres du modèle. Néaumoins, notre proposition présente des avantages et des limites.

Avantages du modèle proposé

Le modèle proposé présente une efficacité dans les critères suivants :

— Notre modèle est basé sur l'entité la plus élémentaire (cycle d'horloge), ce qui permet d'avoir un degré élevé de précision lors de l'évaluation des mesures.
— L'identification des paramètres du modèle est pertinente car elle est basée sur des mesures effectuées dans un environnement réel et diversifié.
— La méthodologie proposée permet une optimisation par suivi des sources de consommation d'énergie.
— La méthodologie proposée peut être utilisée pour prendre des décisions pour une performance souhaitée.

Limites du modèle proposé

— La précision du modèle dépend des caractéristiques électriques de l'appareil en question.
— La méthodologie proposée n'est pas automatisée, elle nécessite une installation manuelle des outils utilisés pour développer le modèle.
— Le type de régression n'est pas détecté automatiquement.

6.4 Conclusion

Afin de construire un modèle de consommation d'énergie des applications sur les appareil mobiles, nous avons présenté dans cette partie une methodologie composée de quatre étapes :

— La première étape consiste à mesurer précisement l'énergie consommée par les applications en cours d'utilisation.
— La deuxième étape consiste à établir les differentes prédictions relatives aux differents scénarios pour l'élaboration du modèle.
— Dans la troisième étape, nous nous sommes basé sur ces mesures pour construire un modèle de consommation énergétique en fonction de la fréquence CPU, et l'énergie dissipée.
— En fin, dans la quatrième étape nous avons validé le modèle proposé.

Les expérimentations liées à cette méthodologie on été effectuées sur différents scénarios basés sur les cas suivant :

— Cas de vidéo locale,
— Cas d'une vidéo distante,

— Cas de navigation.

Comme résultat, nous avons constaté que dans les scénarios de test qui n'impliquent pas de communication, la capacité de prédiction de notre modèle est plutôt bonne.

Pour les cas relatifs au mode connecté, on remarque que la prédiction est moins précise vu le nombre de paramètres impliquées et la diversité des situations rencontrées.

La qualité principale de notre méthodologie en ce qui concerne les travaux connexes est la pertinence des données enregistrées qui conduit à un modèle précis.

Chapitre 7

Conclusion générale et perspectives

Plusieurs domaines sont confrontés aux problèmes de modélisation énergétique dans les environnements mobiles. Les recherches déployées dans ce sens ont généré plusieurs méthodes de résolution approximatives ou exactes selon le problème traité.

Dans cette thèse, nous nous sommes intéressés aux modélisations mathématiques qui sont des méthodes approximatives vu le comportement électrique des différents composants électroniques du dispositif mobile utilisé pour les différentes expérimentations.

Les objectifs de notre travail consistent à :

— modéliser la consommation énergétique pour les smartphones ;
— proposer un modèle mathématique pour le suivi du comportement énergétique ;
— élaborer une méthodologie pour identifier les parametres du modèle.

Ce travail sert en premier lieu à modéliser la consommation énergétique d'une ou plusieurs applications particulières dans un dispositif mobile. Il permet, d'une part, de faire le suivi de la consommation d'énergie en fonction de la fréquence et du nombre d'opérations totales dans une période donnée ; et d'autre part, à mettre en pratique ces méthodologies proposées à travers plusieurs études de cas concrets.

Les premiers chapitres sont consacrés à une présentation générale des sources de consommation énergétique coté architecture et logiciel et à un état de l'art sur la consommation énergétique des différents composants matériels et logiciels.

Notre méthodologie peut être utilisée pour résoudre le problème d'utilisation excessive des ressources en agissant sur la fréquence des processeurs. Le rôle de notre proposition consiste à faire le suivi du comportement énergétique afin de définir une fréquence optimale pour une situation fixée pour offrir une meilleure expérience utilisateur avec une consommation minimale.

La première contribution de ce travail concerne l'adaptation de la consommation

énergétique d'un smartphone. Dans cette partie, nous avons mené une enquête sur des techniques qui permettent de montrer les principales sources de consommation énergétique dans un but d'optimisation de l'énergie consommée par les différents périphériques.

La deuxième contribution consiste, d'une part, à estimer la consommation d'énergie sur les smartphones, et d'autre part, à faire l'évaluation de la consommation d'énergie en introduisant le cas d'étude du comportement énergétique en utilisant les machines virtuelles.

La troisième contribution présentée dans ce document est le développement d'une méthodologie expérimentale pour modéliser l'énergie. La consommation dans les environnements mobiles et plus spécifiquement les plateformes Android. L'objectif principal de cette méthodologie est de permettre la modélisation de la consommation d'énergie d'un appareil mobile particulier lors d'un sénario d'usage particulier.

Dans cette partie nous avons développé une méthodologie de conception de modèle à base des paramètres de la fréquences des processeurs, de l'énergie dissipée ainsi que le niveau initial de la batterie. À cette fin, nous avons analysé par des outils statistiques une collection de données expérimentales que nous avons recueillies lors de mon tour de France en fauteuil roulant. Pour valider ce modèle, nous avons collecté d'autres données expérimentales et nous les avons comparées avec les prédictions de notre modèle, ceci nous a permis de le valider. Dans ce document nous avons présenté une méthodologie pour construire un modèle de la consommation d'énergie des applications sur des dispositifs mobiles. Cette méthodologie commence par enregistrer des mesures précises de consommation d'énergie lors de l'utilisation des applications. En utilisant ces mesures, on construit un modèle de consommation d'énergie en fonction de l'activité des traitements des cœurs pour des fréquences fixes par régression linéaire, et par régression quadratique pour les fréquences variables.

Nous avons démontré cette méthodologie sur un cas de test expérimental sur la lecture d'une vidéo locale, vidéo distante et pour le cas de navigation. Les capacités de prédiction du modèle sont plutôt bonnes pour le cas déconnecté. Notre modèle était moins précis dans le cas connecté en raison de la variabilité de la communication et de la connectivité du lien sans fil.

Le point fort de notre méthodologie réside dans la technique du traitement des données enregistrés qui mène à un modèle précis.

La solution proposée peut être utilisée pour définir une fréquence optimale pour une ou plusieurs applications afin d'offrir une meilleure expérience aux utilisateurs avec une consommation d'énergie réduite.

Les perspectives de ce travail sont nombreuses, nous comptons étendre cette méthodologie pour traiter d'autres cas de test dans le mode connecté. Une telle méthodologie peut aider les développeurs à comprendre le comportement de la consommation énergétique de leurs applications tout en veillant sur la qualité de service, les perspectives de cette étude consistent à :

— Améliorer et développer notre méthodologie pour atteindre un résultat similaire à celui obtenue en mode déconnecté pour chaque paramètre traité dans le mode connecté (Wi-Fi, 4G, 5G, GPS).
— Optimiser notre méthodologie afin de pouvoir prédire des résultats pertinents en fusionnant les différents paramètres liés au mode connecté (Wi-Fi et GPS, 4G/5G et GPS) puis sur les différents systèmes d'exploitations tels que IOS, Tizen...
— Développer des méthodes pour automatiser le type de régression envisageable pour les différents scénarios traités pour une prédiction plus précise et efficace.
— Identifier les fréquences optimales des différents scénarios traités dans les deux modes traités (déconnecté et connecté) selon les souhaits des utilisateurs.
— Développer la gestion l'énergie de la queue des différents dispositifs afin de définir des modèles aussi appropriés que possible pour assurer une meilleure utilisation.

Les pistes de recherche envisageables pour les perspectives consistent à :

— refaire le même travail sur d'autres type de système d'exploitation telle que IOs, Tizen...
— réaliser d'autres études sur d'autres types de scénarios vu le taille importante des données mesurées pendant le tour France pour une meilleure prédiction,
— mettre le lien pour les scénarios basés sur la géolocalisation avec les données récupérées avec le boîtier équipé de système de géolocalisation pour une évaluation plus évoluée,
— élaborer une étude sur la variation de la consommation énergétique constatée dans les zones blanches.

Bibliographie

[1] *3D*. https ://fr.wikipedia.org/w/index.php ?title=Trois_dimensions.

[2] *Gouverneur*. http ://www.phonandroid.com/forum/qu-est-ce-qu-un-gouverneur-cpu-t26062.html.

[3] *marché des smartphones*. http ://www.zdnet.fr/actualites/chiffres-cles-les-os-pour-smartphones-39790245.htm.

[4] Sasan Adibi. *Quality of Service Architectures for Wireless Networks : Performance Metrics and Management : [?]Performance Metrics and Management*. IGI Global, 2010.

[5] Arjun Anand, Constantine Manikopoulos, Quentin Jones, and Cristian Borcea. A quantitative analysis of power consumption for location-aware applications on smart phones. In *Industrial Electronics, 2007. ISIE 2007. IEEE International Symposium on*, pages 1986–1991. IEEE, 2007.

[6] Niranjan Balasubramanian, Aruna Balasubramanian, and Arun Venkataramani. Energy consumption in mobile phones : a measurement study and implications for network applications. In *Proceedings of the 9th ACM SIGCOMM conference on Internet measurement conference*, pages 280–293. ACM, 2009.

[7] Ulrich Bareth and Axel Kupper. Energy-efficient position tracking in proactive location-based services for smartphone environments. In *Computer Software and Applications Conference (COMPSAC), 2011 IEEE 35th Annual*, pages 516–521. IEEE, 2011.

[8] Daniel Bedard, Min Yeol Lim, Robert Fowler, and Allan Porterfield. Powermon : Fine-grained and integrated power monitoring for commodity computer systems. In *IEEE SoutheastCon 2010 (SoutheastCon), Proceedings of the*, pages 479–484. IEEE, 2010.

[9] Frank Bellosa. The benefits of event : driven energy accounting in power-sensitive systems. In *Proceedings of the 9th workshop on ACM SIGOPS European workshop : beyond the PC : new challenges for the operating system*, pages 37–42. ACM, 2000.

[10] Ramon Bertran, Yolanda Becerra, David Carrera, Vicenç Beltran, Marc Gonzàlez, Xavier Martorell, Nacho Navarro, Jordi Torres, and Eduard Ayguadé. Energy accounting for shared virtualized environments under dvfs using pmc-based power models. *Future Generation Computer Systems*, 28(2) :457–468, 2012.

[11] Ramon Bertran, Marc Gonzalez, Xavier Martorell, Nacho Navarro, and Eduard Ayguade. Decomposable and responsive power models for multicore processors using performance counters. In *Proceedings of the 24th ACM International Conference on Supercomputing*, pages 147–158. ACM, 2010.

[12] W Lloyd Bircher and Lizy K John. Complete system power estimation : A trickle-down approach based on performance events. In *Performance Analysis of Systems & Software, 2007. ISPASS 2007. IEEE International Symposium on*, pages 158–168. IEEE, 2007.

[13] William Lloyd Bircher, Madhavi Valluri, Jason Law, and Lizy K John. Runtime identification of microprocessor energy saving opportunities. In *Low Power Electronics and Design, 2005. ISLPED'05. Proceedings of the 2005 International Symposium on*, pages 275–280. IEEE, 2005.

[14] Ata E Husain Bohra and Vipin Chaudhary. Vmeter : Power modelling for virtualized clouds. In *Parallel & Distributed Processing, Workshops and Phd Forum (IPDPSW), 2010 IEEE International Symposium on*, pages 1–8. Ieee, 2010.

[15] Aaron Carroll and Gernot Heiser. An analysis of power consumption in a smartphone. 2010.

[16] Jie Chen and Guru Venkataramani. endebug : A hardware–software framework for automated energy debugging. *Journal of Parallel and Distributed Computing*, 96 :121–133, 2016.

[17] M Chuah, Wei Luo, and Xiaowei Zhang. Impacts of inactivity timer values on umts system capacity. In *Wireless Communications and Networking Conference, 2002. WCNC2002. 2002 IEEE*, volume 2, pages 897–903. IEEE, 2002.

[18] Maxime Colmant, Mascha Kurpicz, Pascal Felber, Loïc Huertas, Romain Rouvoy, and Anita Sobe. Bitwatts : a process-level power monitoring middleware. In *Proceedings of the Posters & Demos Session*, pages 41–42. ACM, 2014.

[19] Maxime Colmant, Mascha Kurpicz, Pascal Felber, Loïc Huertas, Romain Rouvoy, and Anita Sobe. Process-level power estimation in vm-based systems. In *Proceedings of the Tenth European Conference on Computer Systems*, page 14. ACM, 2015.

[20] Shuo Deng and Hari Balakrishnan. Traffic-aware techniques to reduce 3g/lte wireless energy consumption. In *Proceedings of the 8th international conference on Emerging networking experiments and technologies*, pages 181–192. ACM, 2012.

[21] Mikhail Dmitriev. Profiling java applications using code hotswapping and dynamic call graph revelation. In *ACM SIGSOFT Software Engineering Notes*, volume 29, pages 139–150. ACM, 2004.

[22] Thanh Do, Suhib Rawshdeh, and Weisong Shi. ptop : A process-level power profiling tool, 2009.

[23] Manuel Francisco Dolz Zaragozá, Julian Kunkel, Konstantinos Chasapis, and Sandra Catalán Pallarés. An analytical methodology to derive power models based on hardware and software metrics. 2015.

[24] Danilo Dominguez. *Analyzing Android applications for specifications and bugs*. Rochester Institute of Technology, 2013.

[25] Mian Dong and Lin Zhong. Self-constructive high-rate system energy modeling for battery-powered mobile systems. In *Proceedings of the 9th international conference on Mobile systems, applications, and services*, pages 335–348. ACM, 2011.

[26] Jason Flinn and Mahadev Satyanarayanan. Powerscope : A tool for profiling the energy usage of mobile applications. In *Mobile Computing Systems and Applications, 1999. Proceedings. WMCSA'99. Second IEEE Workshop on*, pages 2–10. IEEE, 1999.

[27] Rodrigo Fonseca, Prabal Dutta, Philip Levis, and Ion Stoica. Quanto : Tracking energy in networked embedded systems. In *OSDI*, volume 8, pages 323–338, 2008.

[28] Rong Ge, Xizhou Feng, Shuaiwen Song, Hung-Ching Chang, Dong Li, and Kirk W Cameron. Powerpack : Energy profiling and analysis of high-performance systems and applications. *IEEE Transactions on Parallel and Distributed Systems*, 21(5) :658–671, 2010.

[29] Susan L Graham, Peter B Kessler, and Marshall K Mckusick. Gprof : A call graph execution profiler. In *ACM Sigplan Notices*, volume 17, pages 120–126. ACM, 1982.

[30] David Grove, Greg DeFouw, Jeffrey Dean, and Craig Chambers. Call graph construction in object-oriented languages. *ACM SIGPLAN Notices*, 32(10) :108–124, 1997.

[31] Pankaj K Gupta, RV Rajakumar, C Senthil Kumar, and Goutam Das. Analytical evaluation of signalling cost on power saving mechanism in mobile networks. In *TENCON Spring Conference, 2013 IEEE*, pages 376–380. IEEE, 2013.

[32] Marcus Hähnel, Björn Döbel, Marcus Völp, and Hermann Härtig. Measuring energy consumption for short code paths using rapl. *ACM SIGMETRICS Performance Evaluation Review*, 40(3) :13–17, 2012.

[33] K. I. Hamzaoui, G. Grimaud, M. Azizi, M. Berrajaa, and A. Betari. Survey on adaptation techniques of energy consumption within a smartphone. In *2014 Science and Information Conference*, pages 247–253, August 2014.

[34] Khalil Ibrahim Hamzaoui, Wiame Benzekri, Gilles Grimaud, Mohammed Berrajaa, and Mostafa Azizi. Estimation and Optimization of Energy Consumption on Smartphones. *SpringerLink*, pages 333–342, 2016.

[35] Khalil Ibrahim Hamzaoui, Mohammed Berrajaa, Mostafa Azizi, Giuseppe Lipari, and Pierre Boulet. An experimental methodology for modeling the energy consumption of mobile devices. In *The First Conference on Embedded and Distributed Systems, EDiS'2017*. IEEE, 2017.

[36] Linda Hasman. An introduction to consumer health apps for the iphone. *Journal of Consumer Health On the Internet*, 15(4) :322–329, 2011.

[37] Junxian Huang, Feng Qian, Alexandre Gerber, Z Morley Mao, Subhabrata Sen, and Oliver Spatscheck. A close examination of performance and power characteristics of 4g lte networks. In *Proceedings of the 10th international conference on Mobile systems, applications, and services*, pages 225–238. ACM, 2012.

[38] Intel Intel. and ia-32 architectures software developer's manual, 2011. *Intel order Number*, 64, 64.

[39] Abhilash Jindal, Abhinav Pathak, Y Charlie Hu, and Samuel Midkiff. On death, taxes, and sleep disorder bugs in smartphones. In *Proceedings of the Workshop on Power-Aware Computing and Systems*, page 1. ACM, 2013.

[40] Wonwoo Jung, Chulkoo Kang, Chanmin Yoon, Donwon Kim, and Hojung Cha. Devscope : a nonintrusive and online power analysis tool for smartphone hardware components. In *Proceedings of the eighth IEEE/ACM/IFIP international conference on Hardware/software codesign and system synthesis*, pages 353–362. ACM, 2012.

[41] Aman Kansal, Feng Zhao, Jie Liu, Nupur Kothari, and Arka A Bhattacharya. Virtual machine power metering and provisioning. In *Proceedings of the 1st ACM symposium on Cloud computing*, pages 39–50. ACM, 2010.

[42] Dongwon Kim, Wonwoo Jung, and Hojung Cha. Runtime power estimation of mobile amoled displays. In *Design, Automation & Test in Europe Conference & Exhibition (DATE), 2013*, pages 61–64. IEEE, 2013.

[43] Minyong Kim, Young Guen Kim, Cheol Hong Kim, and Sung Woo Chung. Battery consumption != energy consumption : A case study with a smartphone. *Computer*, (1) :1, 2013.

[44] Avi Kivity, Yaniv Kamay, Dor Laor, Uri Lublin, and Anthony Liguori. kvm : the linux virtual machine monitor. In *Proceedings of the Linux symposium*, volume 1, pages 225–230, 2007.

[45] Deqian Kong, Tao Qi, Tan Yang, and Yidong Cui. A dynamic computation offloading framework for android. In *Broadband Network & Multimedia Technology (IC-BNMT), 2013 5th IEEE International Conference on*, pages 134–138. IEEE, 2013.

[46] James H Laros, Phil Pokorny, and David DeBonis. Powerinsight-a commodity power measurement capability. In *Green Computing Conference (IGCC), 2013 International*, pages 1–6. IEEE, 2013.

[47] Chi-Chen Lee, Jul-Hung Yeh, and Jyh-Cheng Chen. Impact of inactivity timer on energy consumption in wcdma and cdma2000. In *Wireless Telecommunications Symposium, 2004*, pages 15–24. IEEE, 2004.

[48] Min Yeol Lim, Allan Porterfield, and Robert Fowler. Softpower : fine-grain power estimations using performance counters. In *Proceedings of the 19th ACM International Symposium on High Performance Distributed Computing*, pages 308–311. ACM, 2010.

[49] Christine Louberry, Philippe Roose, and Marc Dalmau. Kalimucho : Plateforme d'adaptation des applications mobiles. In *NOTERE 201-Conférence Internationale sur les NOuvelles Technologies de la REpartition*, pages 83–90, 2011.

[50] Frank Maker and Y-H Chan. A survey on android vs. linux. *University of California*, pages 1–10, 2009.

[51] John C McCullough, Yuvraj Agarwal, Jaideep Chandrashekar, Sathyanarayan Kuppuswamy, Alex C Snoeren, and Rajesh K Gupta. Evaluating the effectiveness of model-based power characterization. In *USENIX Annual Technical Conf*, volume 20, 2011.

[52] Grace Metri, Abhishek Agrawal, Ramesh Peri, and Weisong Shi. What is eating up battery life on my smartphone : A case study. In *Energy Aware Computing, 2012 International Conference on*, pages 1–6. IEEE, 2012.

[53] Radhika Mittal, Aman Kansal, and Ranveer Chandra. Empowering developers to estimate app energy consumption. In *Proceedings of the 18th annual international conference on Mobile computing and networking*, pages 317–328. ACM, 2012.

[54] Henry Muccini, Antonio Di Francesco, and Patrizio Esposito. Software testing of mobile applications : Challenges and future research directions. In *Proceedings of the 7th International Workshop on Automation of Software Test*, pages 29–35. IEEE Press, 2012.

[55] Gail C Murphy, David Notkin, William G Griswold, and Erica S Lan. An empirical study of static call graph extractors. *ACM Transactions on Software Engineering and Methodology (TOSEM)*, 7(2) :158–191, 1998.

[56] Adel Noureddine. *Towards a better understanding of the energy consumption of software systems*. PhD thesis, Université des Sciences et Technologie de Lille-Lille I, 2014.

[57] Adel Noureddine, Syed Islam, and Rabih Bashroush. Jolinar : analysing the energy footprint of software applications. In *The International Symposium on Software Testing and Analysis*, pages Pages–445, 2016.

[58] Adel Noureddine, Romain Rouvoy, and Lionel Seinturier. Monitoring energy hotspots in software. *Journal of Automated Software Engineering*, 22(3) :291–332, 2015.

[59] Abhinav Pathak, Y Charlie Hu, and Ming Zhang. Bootstrapping energy debugging on smartphones : a first look at energy bugs in mobile devices. In *Proceedings of the 10th ACM Workshop on Hot Topics in Networks*, page 5. ACM, 2011.

[60] Abhinav Pathak, Y Charlie Hu, and Ming Zhang. Where is the energy spent inside my app ?: fine grained energy accounting on smartphones with eprof. In *Proceedings of the 7th ACM european conference on Computer Systems*, pages 29–42. ACM, 2012.

[61] Abhinav Pathak, Y Charlie Hu, Ming Zhang, Paramvir Bahl, and Yi-Min Wang. Fine-grained power modeling for smartphones using system call tracing. In *Proceedings of the sixth conference on Computer systems*, pages 153–168. ACM, 2011.

[62] Trevor Pering, Yuvraj Agarwal, Rajesh Gupta, and Roy Want. Coolspots : reducing the power consumption of wireless mobile devices with multiple radio interfaces. In *Proceedings of the 4th international conference on Mobile systems, applications and services*, pages 220–232. ACM, 2006.

[63] Gian Paolo Perrucci. *Energy saving strategies on mobile devices*. Aalborg University, Department of Electronics Systems, Multimedia Information and Signal Processing, 2009.

[64] Bodhi Priyantha, Dimitrios Lymberopoulos, and Jie Liu. Littlerock : Enabling energy-efficient continuous sensing on mobile phones. *IEEE Pervasive Computing*, 10(2) :12–15, 2011.

[65] Feng Qian, Zhaoguang Wang, Alexandre Gerber, Zhuoqing Mao, Subhabrata Sen, and Oliver Spatscheck. Profiling resource usage for mobile applications : a cross-layer approach. In *Proceedings of the 9th international conference on Mobile systems, applications, and services*, pages 321–334. ACM, 2011.

[66] Feng Qian, Zhaoguang Wang, Alexandre Gerber, Zhuoqing Morley Mao, Subhabrata Sen, and Oliver Spatscheck. Characterizing radio resource allocation for 3g networks. In *Proceedings of the 10th ACM SIGCOMM conference on Internet measurement*, pages 137–150. ACM, 2010.

[67] Mohammad Rashti, Gerald Sabin, David Vansickle, and Boyana Norris. Wattprof : A flexible platform for fine-grained hpc power profiling. In *Cluster Computing (CLUSTER), 2015 IEEE International Conference on*, pages 698–705. IEEE, 2015.

[68] Nishkam Ravi, James Scott, Lu Han, and Liviu Iftode. Context-aware battery management for mobile phones. In *Pervasive Computing and Communications, 2008. PerCom 2008. Sixth Annual IEEE International Conference on*, pages 224–233. IEEE, 2008.

[69] Suzanne Rivoire, Parthasarathy Ranganathan, and Christos Kozyrakis. A comparison of high-level full-system power models. *HotPower*, 8(2) :32–39, 2008.

[70] Harald Röck. Survey of dynamic instrumentation of operating systems, 2007.

[71] Michael Roitzsch. Slice-balancing h. 264 video encoding for improved scalability of multicore decoding. In *Proceedings of the 7th ACM & IEEE international conference on Embedded software*, pages 269–278. ACM, 2007.

[72] Arjun Roy, Stephen M Rumble, Ryan Stutsman, Philip Levis, David Mazières, and Nickolai Zeldovich. Energy management in mobile devices with the cinder operating system. In *Proceedings of the sixth conference on Computer systems*, pages 139–152. ACM, 2011.

[73] Aaron Schulman, Vishnu Navda, Ramachandran Ramjee, Neil Spring, Pralhad Deshpande, Calvin Grunewald, Kamal Jain, and Venkata N Padmanabhan. Bartendr : a practical approach to energy-aware cellular data scheduling. In *Proceedings of the sixteenth annual international conference on Mobile computing and networking*, pages 85–96. ACM, 2010.

[74] Alex Shye, Benjamin Scholbrock, and Gokhan Memik. Into the wild : studying real user activity patterns to guide power optimizations for mobile architectures. In *Microarchitecture, 2009. MICRO-42. 42nd Annual IEEE/ACM International Symposium on*, pages 168–178. IEEE, 2009.

[75] Alex Shye, Benjamin Scholbrock, Gokhan Memik, and Peter A Dinda. Characterizing and modeling user activity on smartphones : summary. In *ACM SIGMETRICS Performance Evaluation Review*, volume 38, pages 375–376. ACM, 2010.

[76] Matti Siekkinen, Mohammad Ashraful Hoque, Jukka K Nurminen, and Mika Aalto. Streaming over 3g and lte : How to save smartphone energy in radio access network-friendly way. In *Proceedings of the 5th Workshop on Mobile Video*, pages 13–18. ACM, 2013.

[77] David C Snowdon, Stefan M Petters, and Gernot Heiser. Accurate on-line prediction of processor and memoryenergy usage under voltage scaling. In *Proceedings of the 7th ACM & IEEE international conference on Embedded software*, pages 84–93. ACM, 2007.

[78] J Michael Spivey. Fast, accurate call graph profiling. *Software : Practice and Experience*, 34(3) :249–264, 2004.

[79] George Terzopoulos and Helen D Karatza. Power-aware bag-of-tasks scheduling on heterogeneous platforms. *Cluster Computing*, 19(2) :615–631, 2016.

[80] Anna Ukhanova, Evgeny Belyaev, Le Wang, and SøRen Forchhammer. Power consumption analysis of constant bit rate video transmission over 3g networks. *Computer Communications*, 35(14) :1695–1706, 2012.

[81] Daniel Versick, Ingolf Waßmann, and Djamshid Tavangarian. Power consumption estimation of cpu and peripheral components in virtual machines. *ACM SIGAPP Applied Computing Review*, 13(3) :17–25, 2013.

[82] Shinan Wang, Hui Chen, and Weisong Shi. Span : A software power analyzer for multicore computer systems. *Sustainable Computing : Informatics and Systems*, 1(1) :23–34, 2011.

[83] Feng Xia, Ching-Hsien Hsu, Xiaojing Liu, Fangwei Ding, and Wei Zhang. The power of smartphones. *arXiv preprint arXiv :1312.6740*, 2013.

[84] Feng Xia, Ching-Hsien Hsu, Xiaojing Liu, Haifeng Liu, Fangwei Ding, and Wei Zhang. The power of smartphones. *Multimedia Systems*, 21(1) :87–101, 2015.

[85] Yu Xiao et al. Modeling and managing energy consumption of mobile devices. 2011.

[86] Jui-Hung Yeh, Jyh-Cheng Chen, and Chi-Chen Lee. Comparative analysis of energy-saving techniques in 3gpp and 3gpp2 systems. *IEEE transactions on vehicular technology*, 58(1) :432–448, 2009.

[87] Chanmin Yoon, Dongwon Kim, Wonwoo Jung, Chulkoo Kang, and Hojung Cha. Appscope : Application energy metering framework for android smartphone using kernel activity monitoring. In *USENIX Annual Technical Conference*, volume 12, pages 1–14, 2012.

[88] Houssam-Eddine Zahaf, Abou El Hassen Benyamina, Richard Olejnik, and Giuseppe Lipari. Energy-efficient scheduling for moldable real-time tasks on heterogeneous computing platforms. *Journal of Systems Architecture*, 74 :46–60, 2017.

[89] Heng Zeng, Carla S Ellis, Alvin R Lebeck, and Amin Vahdat. Ecosystem : Managing energy as a first class operating system resource. In *ACM Sigplan Notices*, volume 37, pages 123–132. ACM, 2002.

[90] Heng Zeng, Carla Schlatter Ellis, Alvin R Lebeck, and Amin Vahdat. Currentcy : A unifying abstraction for expressing energy management policies. In *USENIX Annual Technical Conference, General Track*, pages 43–56, 2003.

[91] Yan Zhai, Xiao Zhang, Stephane Eranian, Lingjia Tang, and Jason Mars. Happy : Hyperthread-aware power profiling dynamically. In *USENIX Annual Technical Conference*, pages 211–217, 2014.

[92] Lide Zhang, Birjodh Tiwana, Robert P Dick, Zhiyun Qian, Z Morley Mao, Zhaoguang Wang, and Lei Yang. Accurate online power estimation and automatic battery behavior based power model generation for smartphones. In *Hardware/Software Codesign and System Synthesis (CODES+ ISSS), 2010 IEEE/ACM/IFIP International Conference on*, pages 105–114. IEEE, 2010.