ÉNERGIE ET RADIOSCIENCES

Mesure de la Consommation Énergétique des Systèmes Embarqués Communicants

Nadir Cherifi*, Gilles Grimaud**, Thomas Vantroys**, Alexandre Boé***

** Worldline e-payment services et IRCICA/Cristal, CNRS UMR 9189, Univ. Lille 1, nadir.cherifi@worldline.com*
*** IRCICA/CRIStAL, CNRS UMR 9189, Univ. Lille 1, {gilles.grimaud; thomas.vantroys}@univ-lille1.fr*
*** IRCICA/IEMN, CNRS UMR 8520, Univ. Lille 1, alexandre.boe@univ-lille1.fr

Mots-clefs : Consommation énergétique; Mesure énergétique; Systèmes embarqués; Internet des Objets.

Résumé

Les systèmes embarqués connectés autonomes sont en expansion permanente. De nos jours, nous en trouvons de plus en plus tout autour de nous. L'avènement et la croissance de l'Internet des Objets (IdO) ont permis à ces systèmes ou objets de faire un bond considérable, notamment concernant leur diversification de déploiement dans de multiples applications réelles de la vie quotidienne. Les prédictions font état d'un nombre colossal d'objets connectés pouvant être déployés dans les années à venir, dont nombre d'entre eux seront dépendants et contraints énergétiquement par une batterie. De ce fait, au niveau de ces petits objets, l'énergie représente une ressource critique qui limite un déploiement à très large échelle de ces derniers. De surcroît, cette contrainte énergétique complexifie grandement le développement du logiciel embarqué. Par conséquent, la possibilité de mesurer, tracer et profiler finement la consommation énergétique de tels objets puis de la corréler à l'exécution de l'application embarquée par ces derniers représente un vrai challenge dans l'optique d'améliorer le développement des applications embarquées pour l'IdO. Parmi les applications possibles, il en existe une sous partie décrivant au mieux l'idée de l'internet des objets, qui sont les applications du Web des Objets.

Dans ce papier, nous présentons nos travaux concernant la mesure de consommation énergétique des applications embarquées connectées. Plus précisément, nous nous concentrons sur la mesure d'une application web, consistant en un serveur web HTTP embarqué sur le populaire système d'exploitation Contiki. Pour ceci, nous utilisons deux méthodes, ou plutôt deux environnements de mesures : a) une mesure en environnement totalement simulé et b) une mesure en environnement réel. Nous quantifions et modélisons, l'écart existant entre ces deux environnements de mesure énergétique pour des objets communicants et contraints en énergie. Enfin, nous présentons nos intentions dans l'utilisation d'une approche de mesure énergétique matérielle en environnement réel.

Introduction

Nous pouvons définir l'Internet des Objets (IdO) comme l'idée d'un système où les objets environnants dans le monde physique sont connectés directement, sans passerelle, à l'Internet traditionnel via des connexions filaires ou non-filaires. Le web des objets (WdO) étend cette vision un peu plus loin, en incluant l'interopérabilité au niveau applicatif. Le WdO promeut l'utilisation de services web directement embarqués sur l'objet communicant, afin d'obtenir et tendre vers une intégration totale de ce dernier avec les infrastructures informatiques déjà existantes. Afin d'achever totalement cet objectif et assurer à l'objet une vraie interopérabilité au niveau réseau, il est donc nécessaire d'embarquer sur l'objet des piles logicielles réseaux IP et applicatives.

En se focalisant sur des objets communicants sans fil alimentés par batterie, de multiples exemples d'applications peuvent être cités où le remplacement ou rechargement de la batterie est contraignant, difficile voire impossible. Ainsi, la consommation et l'efficacité énergétique deviennent les préoccupations principales dans le développement logiciel embarqué de ces dispositifs communicants. En outre, cette question prend un rôle critique quand cela concerne l'IdO en raison de l'utilisation de protocoles internet traditionnels existants qui ont été originalement conçus sans contraintes énergétiques à l'esprit.

Il découle des précédentes affirmations que la conception de logiciels embarqués efficaces en énergie pour l'IdO nécessite d'avoir un profil de consommation énergétique fin grain des objets cibles dans le but de détecter les points chauds de consommation du logiciel et de les corriger. La mesure et la capacité d'estimer la consommation énergétique d'un logiciel embarqué sur un objet communiquant est basée sur plusieurs méthodes et approches. Nous proposons de notre côté, une méthode de mesure énergétique matérielle s'effectuant en conditions réelles, permettant de capturer la majeur partie de l'activité énergétique de l'objet en condition réelles d'exécution et de déploiement. Ce papier propose un profil de consommation énergétique du serveur web HTTP embarqué par le système d'exploitation Contiki OS. Nous décrivons et expliquons la consommation énergétique induite par l'utilisation d'une couche applicative telle que HTTP

qui repose sur les principes de garantie de communication du protocole TCP. Enfin, nous montrons expérimentalement, en utilisant plusieurs configurations, que l'écart existant entre une mesure énergétique en environnement réel et une autre en environnement simulé est non négligeable.

Ce papier est structuré comme suit. Dans la Section 1, nous présentons les bases nécessaires pour appréhender les approches de mesure énergétique ainsi que les piles logicielles embarquées de communication. En Section 2, nous décrivons la méthodologie expérimentale de nos travaux. En Section 3, nous présentons et discutons les résultats expérimentaux. Enfin, en Section 4 nous donnons un aperçu de nos futurs travaux.

1. Bagage technique

Dans ce papier, nous décrivons le profilage énergétique d'un serveur web embarqué sur un objet communicant. Cela requiert évidement une bonne connaissance des approches de mesure énergétique. De plus, Nous pouvons supposer que dans le cas des petits objets communicants sans fil, la majeure partie de la consommation énergétique est destinée au module radio. Ainsi, la compréhension des points chauds de consommation énergétique dans les communications entre objets, nécessite une compréhension claire des concepts sous-jacents aux piles logicielle de communication pilotant l'activité du module radio.

1.1 Techniques de mesure énergétique

La consommation énergétique des petits systèmes embarqués communicants est une grandeur difficile à mesurer. Nous pouvons globalement diviser les méthodes de mesure en trois catégories distinctes : Les approches matérielles, les approches logicielles et les approches basées sur la simulation.

La majeure partie des techniques matérielles repose sur la mesure de la tension aux bornes d'une résistance de de faible valeur placée en série avec le dispositif à mesurer. L'acquisition des profils de consommation énergétique se fait grâce à l'utilisation d'un oscilloscope digital ou une plate-forme dédiée à l'instar de FlockLab [1]. Cette dernière consiste en un banc de test conçu pour le profilage temporel et énergétique de petits objets embarqués. La plate-forme tire avantage des GPIO existants sur la plupart des systèmes embarqués afin d'enregistrer des événements logiques, et de les corréler à la courbe énergétique mesurée. Des méthodes alternatives à l'utilisation d'une résistance de shunt existent. L'une d'entre elles, utilise des condensateurs spéciaux, nommés GoldCaps [2], ayant une très large capacité (Typiquement 1 Farad), dans le but d'alimenter le dispositif à mesurer. Cela permet de réaliser de courtes expérimentations, d'une grandeur équivalente à la capacité du GoldCaps, et de fournir au final une prédiction de la durée de vie de l'objet mesuré lorsque ce dernier effectue telle ou telle activité.

Certains dispositifs matériels de mesure peuvent être directement embarqués sur l'objet à mesurer, octroyant à ce dernier, la capacité de mesurer *in-situ* sa propre consommation. Cette capacité est hautement importante puisqu'elle permet à l'objet d'effectuer des décisions et activités selon le niveau d'énergie consommé et restant. ICount [3] est l'un de ces dispositifs. Il est utilisable sans ajout de matériel, sur toute plate-forme ayant un simple compteur matériel ainsi qu'un système d'alimentation équipé d'un convertisseur DC-DC. La mesure énergétique passe, par le comptage du nombre de cycles de commutation du convertisseur, les auteurs ayant prouvé que ces derniers sont linéairement dépendants du courant débité (et donc l'énergie) par le système d'alimentation. Quanto [4] tire avantage du dispositif ICount et instrumente les pilotes matériels de l'objet à mesurer afin de tracer et profiler la consommation énergétique des diverses activités effectuées par l'objet en question. Cependant, ces dispositifs de mesure pouvant être embarqués souffrent d'une précision insuffisante lorsqu'on s'intéresse à un profilage fin grain de la consommation énergétique.

Contrairement à la précédente catégorie, les méthodes de mesure logicielle ne requièrent aucun rajout de circuit matériel. Powertrace présenté en [5] mesure le temps durant lequel les composants de l'objet (CPU, radio, mémoire flash, etc.) sont dans différents états d'énergie (CPU Actif ou en veille, radio en transmission ou réception, etc.). Pour cela, les pilotes matériels sont instrumentés afin d'enregistrer une estampille temporelle lorsque le composant associé entre ou quitte un état énergétique prédéterminé. Ainsi, la durée de chaque état énergétique de chaque composant est enregistrée puis multipliée par les informations de puissance tirées des datasheets des composants afin de calculer l'énergie consommée. En plus, Powertrace permet de corréler à un niveau plus haut d'abstraction, les consommations enregistrées aux différentes activités effectuées par l'objet, en introduisant un nouveau concept qui est « les capsules ». Malgré des avantages certains, comparables avec ceux de Quanto, Powertrace accuse un manque de précision lié à son approche d'estimation majoritairement basé sur la datasheet qui n'est pas toujours représentative du monde réel [6].

La dernière catégorie de méthodes de mesure consiste à utiliser des simulateurs énergétiques qui prennent le choix d'estimer la consommation énergétique d'un objet en le simulant avec le logiciel embarqué sur un système hôte. PowerTOSSIM décrit en [7], est un simulateur qui étend TOSSIM, un simulateur pour les applications TinyOS. PowerTOSSIM instrumente les composants périphériques afin de détecter leurs changements d'états, et emploie une technique de transformation de code source afin d'estimer le nombre de cycles CPU effectué par l'objet, évitant ainsi l'utilisation d'un modèle énergétique lourd instruction par instruction. À l'opposé, E-Simu présenté en [8] estime la

consommation énergétique globale de l'objet en modélisant la consommation énergétique de chaque instruction ou de groupe d'instructions. E-Simu fournit aussi une caractérisation simple des composants périphériques d'un objet. Au final, les simulateurs énergétiques représentent de très bons outils pour une estimation énergétique rapide, leur permettant d'être intégrés aisément dans un cycle de développement. Néanmoins, ces derniers échouent lorsqu'il s'agit de fournir des estimations précises de ce que peut être la consommation de l'objet en conditions réel de déploiement. Enfin, les simulateurs sont totalement basés sur la fiabilité des modèles utilisés, qui sont dans la plus part du temps, très complexes à concevoir, développer et enfin valider.

1.2 Consommation énergétique et couche MAC

La couche MAC radio représente la couche logicielle la plus basse dans une pile de communication. Typiquement, elle dirige l'activité réelle du module radio dans le but d'effectuer les opérations nécessaires (écoute, transmission, etc.). Le module radio étant certainement le composant le plus énergivore dans un objet communicant [9], il est communément admis d'éteindre ce module le plus souvent possible afin d'économiser une quantité maximale d'énergie [10]. Ainsi, les protocoles MAC pour objets communicants sont conçus à cet effet, se réveillant uniquement périodiquement pour inspecter si un paquet radio leur est destiné. Afin d'être plus au moins sûr d'intercepter un objet lors de son réveil, un objet transmetteur utilise souvent un mécanisme d'envoi de paquets dit « *sondes* », en rafales.

Plusieurs protocoles MAC efficaces en énergie ont déjà été proposés dans la littérature [10]–[12]. ContikiMAC présenté en [9] est certainement l'un des plus populaires. Il utilise, à l'instar de ce qui a été dit, un protocole de réveil périodique ainsi qu'un mécanisme de transmission de sondes qui sont ici de simples copies du paquet réel à envoyer. Contiki MAC met en avant trois optimisations ou caractéristiques importantes : a) il utilise un mécanisme précis de réveil du module radio, reposant sur la librairie temps réel intégrée dans le système d'exploitation Contiki OS ; b) il implémente un mécanisme de mise en veille rapide du module radio, en permettant à ce dernier de différencier très rapidement entre un vrai paquet radio en cours de réception et du bruit ; c) Contiki MAC introduit un mécanisme de synchronisation de phases permettant d'enregistrer une liste de phase de réveil pour chaque objet voisin. Ainsi, un objet voulant transmettre à un voisin, connaissant le moment exact de réveil de ce dernier, pourra débuter sa communication juste avant le moment prévu où ce voisin se réveillera.

1.3 Consommation énergétique et IP

Les protocoles Internet traditionnels à l'image d'IP (*Internet Protocol*) n'ont pas été conçus avec des critères d'efficacité énergétique. Néanmoins, avec l'engouement de l'IdO, la possibilité d'utiliser ces protocoles pour connecter des objets communicants contraints en énergie, a été prouvée par de multiples travaux [13], [14]. uIP et lwIP [15] proposés par Adam Dunkels représentent les premiers travaux conséquents tirant avantage de l'utilisation d'IP sur des medium radio pour objets communicants, proposant de multiples approches pour limiter la surconsommation d'IP. Ainsi, le problème de lourdeur des entêtes IP par rapport à la faible bande passante du protocole radio 802.15.4 a été résolu en utilisant des techniques de compression d'en-tête. Enfin, lwIP et uIP proposent un mécanisme léger de routage IP, qui permet d'atténuer la charge réseau sur les objets hautement contraints d'un réseau.

Plus récemment encore, avec l'avènement du protocole IPv6, l'IETF (*Internet Engineering Task Force*) a proposé 6LoWPAN [16] comme protocole permettant de rendre possible le passage de larges paquets IPv6 par un médium radio contraint à l'instar de celui définit par la norme IEEE 802.15.4. Ainsi, plusieurs schémas de compression ont été définis ciblant plusieurs ratios et temps de compressions. De plus, IPv6 est caractérisé par une MTU (*Maximum transmission unit*) très large de 1280 octets. De ce fait, plusieurs mécanismes de fragmentation de paquets au niveau réseau ont été mis en place pour supporter, d'une manière optimale (en énergie et en temps), cette MTU. Enfin, concernant les implémentations existantes, SICSlowpan (conçu pour Contiki OS) [17] et b6loWPAN ou blip (conçu pour TinyOS) [18], représentent les travaux les plus matures en vue du support et de l'implémentation de 6LoWPAN. Ils incorporent ainsi les optimisations précédemment décrites en plus d'un mécanisme léger d'auto-assignation d'adresse IPv6 à l'intérieur d'un réseau d'objets.

1.4 Consommation énergétique et TCP

Lors de communications IP entre objets communicants, le protocole UDP est souvent préféré du fait de son faible surcoût. Néanmoins, afin d'être compatible avec d'historique protocoles applicatifs tels que HTTP ou SSH, ainsi que pour correspondre aux critères de divers types de cas d'utilisation sensibles (Medical, Militaire, etc.), une fiabilité est nécessaire concernant la délivrance des paquets émis. Dans ce but, l'adaptation de TCP (*Transmission Control Protocol*) a concentré de nombreux efforts. Dans [19] Adam Dunkels et al. présentent un schéma distribué de cache TCP afin de réduire le nombre de retransmission de bout en bout. Dans ce schéma, chaque objet du réseau maintient un cache pouvant accueillir un seul segment TCP, permettant ainsi une retransmission direct en cas de perte de celui-ci. Une autre approche présentée dans [20], tente de réduire le nombre d'acquittement (ACK) en tirant avantage du mécanisme d'acquittement retardé implémentée dans la plus part des pile TCP/IP traditionnelles. Sur le même thème, dans [21], les auteurs étudient l'impact des acquittements retardés sur l'ensemble d'un réseau d'objet, et proposent un mécanisme dynamique de retardement d'acquittements afin d'améliorer le rendement énergétique du réseau.

2. Méthodologie d'expérimentation

En utilisant deux techniques d'estimation énergétique, nous voulons quantifier les différences existantes entre simulation et monde réel et ainsi, montrer expérimentalement les faiblesses d'une estimation dans un environnement simulé. On présente dans cette section la méthodologie utilisée pour arriver à cet objectif.

2.1 Banc de test expérimental

Nous effectuons nos expérimentations sur des plate-formes Tmote-Sky. Elles embarquent un Microcontrôleur 16 bit et une radio CC2420 basse consommation fonctionnant selon la norme IEEE 802.15.4.

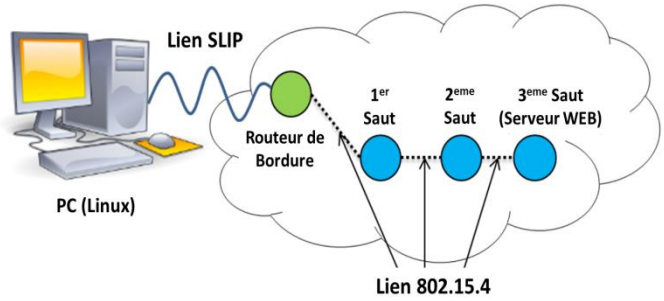

Figure 1 Banc de test expérimental

Le banc de test schématisé en Figure 1 consiste en un mini-réseau sans fil de 3 sauts avec des routes statiques, afin d'éviter d'involontairement mesurer les effets énergétiques causés par un protocole de routage dynamique. Une quatrième plate-forme qui joue le rôle d'un routeur de bordure IPv4/IPv6 est utilisée pour connecter le réseau sans-fil à un PC fonctionnant sur Linux. Cette plate-forme utilise le protocole SLIP (*Serial Line IP*) pour transmettre et recevoir les paquets IP radios. Les modules radios de toutes les plate-formes sont configurés pour utiliser le canal de communication 15 de la norme 802.15.4, qui est connu pour être le moins impacté par les interférences WiFi.

Afin de mesurer la consommation énergétique du serveur HTTP embarqué, nous effectuons divers expérimentations, utilisant notre banc de test. Plus précisément, le PC initie une requête HTTP demandant du serveur web localisé sur le dernier nœud, une charge effective allant de 1 à 1040 octets (Charge maximal pouvant être demandé en IPv6/6LoWPAN dans les limites des capacités de nos plate-formes). Les deux nœuds restants du réseau effectuent la transition des requêtes et réponses. Enfin nous utilisons la commande linux CURL afin d'initier les requêtes HTTP du PC.

2.2 Configurations du banc de test

Nous effectuons nos expérimentations sur deux configurations de la pile de communication de Contiki OS. Nous utilisons à cet effet, les piles de communications uIPv4 et uIPv6 implémentées dans Contiki OS. Concernant uIP4, nous utilisons une MSS (*Maximum Segment Size*) de 48 octets, défini par défaut sur Contiki. Dans le cas d'uIPv6 la MSS utilisée s'élève à 1220 octets, activant de ce fait le mécanisme de fragmentation de paquets réseaux implémenté dans cette pile. Dans un second cas, non documenté, mais uniquement discuté dans ce papier, nous avons choisi de passer outre ce mécanisme de fragmentation en fixant la MSS d'uIPv6 à 48 octets. Enfin, aucune optimisation TCP n'est incluse ou activée dans la version des piles de communication que nous utilisons. De ce fait, chaque segment TCP nécessite un acquittement.

2.3 Environnements de test

Nous effectuons nos tests sous deux environnements distincts : Dans le premier, nous déployons notre banc de test dans un environnement réel représenté par nos locaux. Dans le deuxième, nous utilisons le simulateur d'application Contiki, Cooja, qui offre un environnement d'expérimentation *idéal,* net de toutes interférences. Afin de caractériser la consommation énergétique de chaque objet lors du traitement d'une requête/réponse HTTP, nous utilisons le profileur énergétique précédemment décrit, Powertrace **[5]**. Particulièrement, chaque objet sauvegarde deux estampilles Powertrace, l'une au début et l'autre à la fin de la requête HTTP, lors des détections respectives des drapeaux TCP « SYN » et « le dernier FIN-ACK ». Les résultats présentés ci-dessous, sont moyennés sur 50 exécutions, et les bâtonnets gris représentent les écarts-type.

3. Expérimentations

Dans cette section, nous proposons d'évaluer expérimentalement l'écart existant entre le monde réel et la simulation (comprendre aussi le modèle de simulation) pour mettre en évidence les différences causées par une estimation énergétique en environnement simulé.

3.1 Simulation

Nous effectuons l'expérience décrite, dans un environnement simulé à l'aide de Cooja. En outre, les 3 objets de la route implémentent un protocole MAC efficace en énergie qui est Contiki MAC avec une fréquence de réveil de 8 Hz. Les figures 2 et 3 montrent la consommation énergétique des 3 objets connectés en fonction de la charge utile. Le 3eme saut illustré en vert représente le serveur web. Les 1er et 2eme sauts illustrés respectivement en rouge et bleu correspondent aux routeurs.

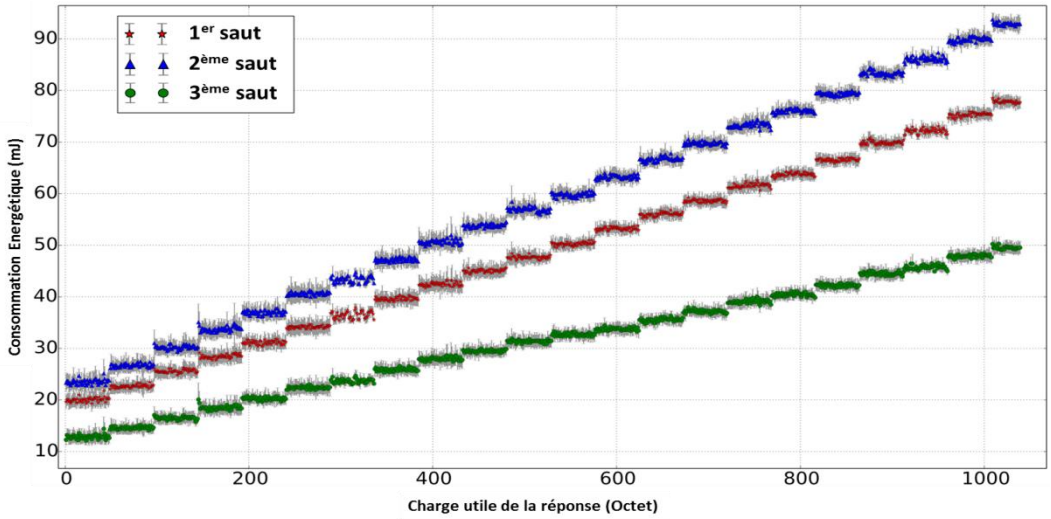

Figure 2 Consommation énergétique des 3 objets en environnement simulé en IPv4

En premier lieu, nous pouvons apercevoir sur les deux expériences, une différence entre la consommation des 3 objets. Ainsi, les deux objets routeurs (1er et 2ème sauts) consomment plus que l'objet serveur, du fait de leurs communications dans les deux sens de la route réseau. De plus, le premier routeur (1er saut) est caractérisé par une consommation énergétique moindre. Cela est explicable car le routeur de bordure, voisin de ce dernier, a son module radio toujours actif. De ce fait, le premier routeur, contrairement au deuxième routeur, ne perd pas d'énergie dans la transmission de plusieurs paquets sondes, en attente du réveil du routeur de bordure.

Sur l'expérimentation en IPv4, nous pouvons noter une incrémentation en escalier de la consommation énergétique de tous les nœuds à mesure que le nombre de paquets augmentent (dans notre configuration, tous les 48 octets). De plus, nous remarquons que l'écart-type symbolisant la variation sur toutes les itérations, est faible (~ 1,5 mJ). Ainsi, la consommation énergétique étant très stable et directement reliée au nombre de paquets dans la réponse HTTP, il est aisé de construire le modèle énergétique sous-jacent. On note $\boldsymbol{E_{v4_1}}$, la consommation énergétique en mJ en IPv4 du i^{eme} objet sur la route. La variable « $\boldsymbol{x}$ » représente dans toutes les équations, la taille de la charge utile de la réponse HTTP en octet. Nous donnons comme exemple, l'équation énergétique du premier objet de la route (eq 1). Concernant les deux autres objets restants, nous trouvons par rapport aux paramètres d'équations : $\boldsymbol{\alpha_2 = 3.5}$, $\boldsymbol{\beta_2 = 22.7}$, $\boldsymbol{\alpha_3 = 2, \beta_3 = 12}$ $\boldsymbol{et\ std_{v4} \sim 1.5}$.

$$\boldsymbol{E_{v4_1} = \alpha_1 \left\lfloor \frac{x}{48} \right\rfloor + \beta_1 \pm std_{v4} = 3.5 \left\lfloor \frac{x}{48} \right\rfloor + 19 \pm 1.5} \qquad (1)$$

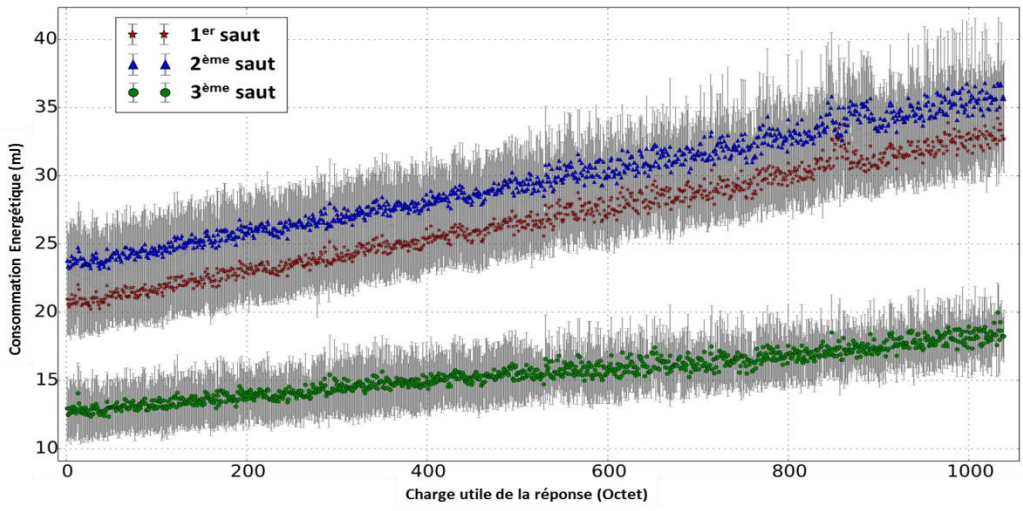

Figure 3 Consommation énergétique des 3 objets en environnement simulé en IPv6

Dans l'expérimentation IPv6, la consommation énergétique n'est plus vraiment liée au nombre de paquets transitant sur la route, mais plus directement au nombre d'octets. De plus, la consommation est nettement moins grande en termes de valeur global, et d'incrémentation en fonction de la taille de la charge utile. Ceci est dû au fait qu'IPv6 permet d'utiliser une large MSS de 1220 octets. Ainsi, via l'utilisation du mécanisme de fragmentation implémenté dans la pile IPv6-6LoWPAN et décrit dans la RFC de la couche d'adaptation 6LoWPAN, le serveur web peut délivrer tout le contenu

demandé en un seul et unique paquet IP. Enfin, l'écart-type reste aussi, dans ce cas, assez faible et constant (~ 2,5 mJ). A l'instar de l'expérimentation précédente nous pouvons modéliser la consommation énergétique sous forme d'une application linéaire. Nous notons $\boldsymbol{E_{v6_1}}$, la consommation énergétique en mJ, en IPv6 du i^{eme} objet sur la route. Nous donnons comme exemple, l'équation énergétique du premier objet (eq 2) et nous trouvons concernant les deux autres objets restants : $\boldsymbol{\gamma_2 = 0.012,\ \varphi_2 = 23.2,\ \gamma_3 = 0.006, \varphi_3 = 12.7\ et\ std_{v6} \sim 2.5}$.

$$\boldsymbol{E_{v6_1} = \gamma_1 x + \varphi_1 \pm std_{v6} = 0.010\, x + 20.3 \pm 2.5} \qquad (2)$$

3.2 Environnement réel

Afin de montrer l'impact d'un environnement réel de déploiement sur la consommation énergétique d'un réseau d'objets communicants, nous réalisons les mêmes expériences que celles décrites précédemment en utilisant de vraies plate-formes embarquées Tmote-Sky déployées dans nos bureaux. Les figures 4 et 6 décrivent la consommation énergétique des 3 objets sur la route, en fonction de la taille de la charge utile dans les cas d'IPv4 et IPv6 respectivement.

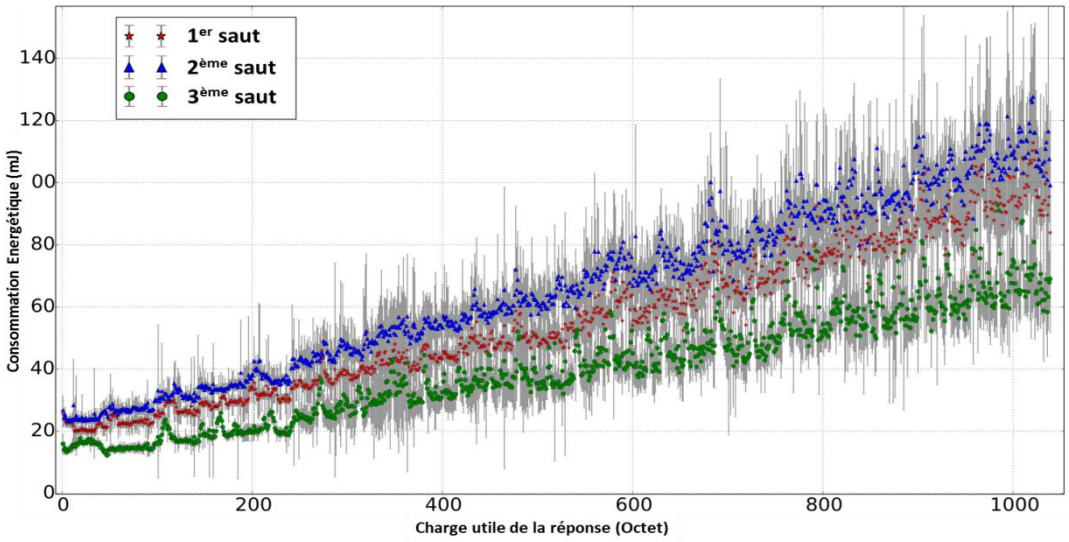

Figure 4 Consommation énergétique des 3 objets en environnement réel en IPv4

En se focalisant dans un premier temps sur le cas IPv4 en environnement réel, nous pouvons remarquer que la consommation énergétique, contrairement au cas simulé est directement liée à la taille de la charge utile transmise, plutôt qu'au nombre de paquets. Cette consommation ainsi que l'écart-type associé sont largement supérieure en comparaison du cas simulé, suivant une incrémentation non constante. Sans prendre en compte pour l'instant cet écart-type, nous pouvons approximer la consommation énergétique par une fonction linéaire. Nous donnons en guise d'exemple l'équation du premier objet de la route (eq 3). Pour les objets restants, nous trouvons : $\boldsymbol{\lambda_2 = 0.084,\ \psi_2 = 23.8,\ \lambda_3 = 0.049, \psi_3 = 13.6\ et\ std_{v4_i}}$

$$\boldsymbol{E_{v4_1} = \lambda_1 x + \psi_1 \pm std_{v4_1} = 0.073x + 21 \pm std_{v4_1}} \qquad (3)$$

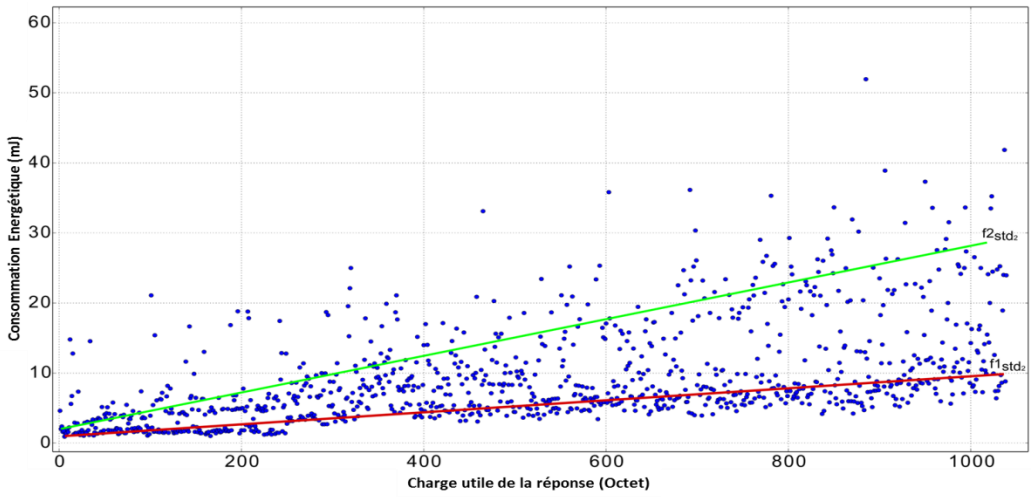

Figure 5 Écart-type de la consommation énergétique de l'objet du 2ème saut en environnement réel sur IPv4

À cause de l'écart-type élevé, le pouvoir prédictif potentiel de notre modèle linéaire n'est pas assez probant en comparaison avec le cas simulé. La figure 5 représente l'écart type de la consommation énergétique sur le deuxième nœud de la route. Nous illustrons ce dernier car il représente parmi les 3 nœuds de la route, le cas le plus souvent rencontré dans un réseau d'objet. L'écart-type augmente d'une manière non stable, en fonction de l'augmentation de la taille de la charge utile demandée, pour atteindre une valeur haute d'approximativement 36 mJ à 1000 octets. De surcroît, l'augmentation de la dispersion des valeurs est elle aussi de plus en plus importante. Enfin, nous pouvons constater, que l'écart-type est d'une sorte divisé en deux ensembles mis en évidence par les droites $\boldsymbol{F1_{std_2}}$ $\boldsymbol{et}$ $\boldsymbol{F2_{std_2}}$ sur la figure 6. Nous prenons en compte ce comportement pour approximer l'écart type de la consommation énergétique. Nous notons $\boldsymbol{std_{v4_2}}$ l'écart type en mJ du 2[eme] objet de la route (eq 4).

$$\boldsymbol{std_{v4_2}} = \begin{cases} \boldsymbol{F1_{std_2} = \eta_a x + \mu_a = 0.01x + 1} \\ \boldsymbol{F2_{std_2} = \eta_b x + \mu_b = 0.028x + 1.6} \end{cases} \quad (4)$$

En résumé, un environnement de déploiement réel, à l'opposé d'un simulé, requiert de la part d'un développeur d'applications IdO/WdB de faire face à une hausse de consommation énergétique a priori imprédictible, en fonction de l'augmentation de la taille de la charge utile transmise. Nous pouvons expliquer la haute valeur de l'écart-type (i.e. erreur) et son caractère imprédictible par le fait de l'existence de nombreuses interférences en environnement réel. En fait, ces expérimentations ont été effectuées dans nos locaux où nous pouvons dénoter la présence de hautes interférences WiFi et d'autres protocoles fonctionnant sur la même bande de fréquence que le 802.15.4 comme le Bluetooth. Ceci impacte hautement le médium radio en causant des variations importantes dans la qualité du lien de communication radio. Cela, à son tour, augmente le taux de perte de paquets durant l'expérimentation, causant dans le cas de la pile uIPv4, une haute pénalité énergétique dû à la retransmission de bout en bout du paquet perdu (HTTP utilise le protocole de garantie de livraison TCP). En plus des perturbations radios, nous pouvons supposer l'existence d'interférences matérielles à l'intérieur de la plate-forme mesurée, causant ainsi, par exemple, une dérive d'horloge qui casse la synchronisation temporelle entre les objets, et donc au final une surconsommation énergétique. Tous ces types d'interférences sont pour le moins imprédictible mais surtout hautement variable sur une longue durée d'expérimentation. Elles sont donc de ce fait, difficilement modélisables pour être prises en compte par un simple simulateur d'objet connecté.

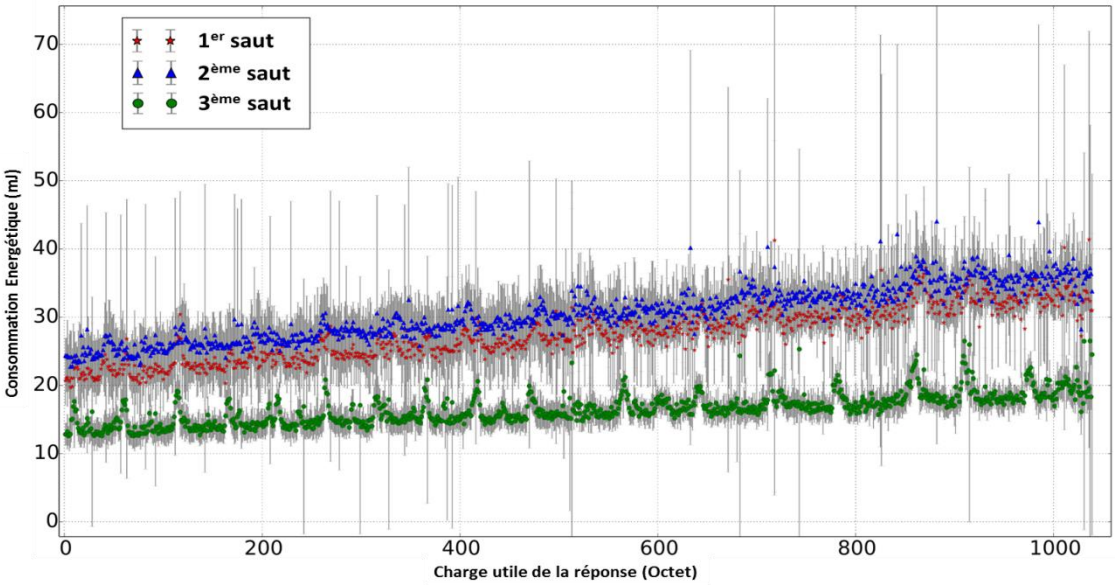

Figure 6 Consommation énergétique des 3 objets en environnement réel en IPv6

Toujours en environnement réel, mais dans l'expérience IPv6, le comportement de la courbe de consommation énergétique des objets illustré en Figure 6 est tout aussi imparfait que l'expérimentation précédente en IPv4. Néanmoins, les valeurs restent plutôt proches du cas simulé en IPv6, et peuvent être prédites et modélisées en utilisant un modèle presque similaire à celui utilisé dans le cas simulé. L'écart-type, illustré aussi en Figure 7 comme exemple pour l'objet de deuxième saut est caractérisé par plusieurs valeurs hautes approchant les 20 mJ ainsi qu'une dispersion des valeurs aux alentours des 10 mJ. Néanmoins, au global, l'écart type et donc la variation reste en pratique constante. La cohérence et ressemblance entre les résultats des cas simulé et réel peuvent être expliqués par l'efficacité énergétique de la pile IPv6, et plus précisément la couche d'adaptation 6LoWPAN. En utilisant une caractéristique de la norme 802.15.4 permettant de multiples envois en rafales, cette couche permet la transmission de larges paquets IP, réduisant de ce fait, les effets néfastes des interférences sur la perte de fragments radios. Plus précisément, la possibilité d'envoyer de larges paquets à travers le médium radio réduit logiquement la probabilité de perte de paquets IP, et donc la probabilité de retransmission de bout en bout du réseau. De plus, la possible perte d'un fragment radio impacte faiblement la consommation

énergétique globale de l'objet, car encore une fois, la couche d'adaptation 6LoWPAN gérera cette perte en effectuant une transmission directe à saut unique.

Malgré l'efficacité énergétique qui caractérise la pile IPv6 de Contiki, la problématique des interférences environnantes et la difficulté de modélisation de ces dernières se pose toujours. Ainsi, nous avons déjà remarqué sur l'expérience IPv6 en environnement réel, une dispersion des valeurs d'écart type assez marquée ainsi que plusieurs valeurs hautes, malgré le fait que chaque expérience ait été rejouée 50 fois. De plus, le fait d'utiliser la couche d'adaptation 6LoWPAN n'influe guère sur l'impact des interférences lors de la transmission de plusieurs paquets de données. Nous avons d'ailleurs fait l'expérience de bloquer le mécanisme de fragmentation de cette couche, induisant ainsi l'obligation d'envoi de multiple paquets IP lors de la demande d'une grande charge utile. Le résultat est comme attendu, quasi-identique avec les résultats de l'expérimentation en environnement réel sur IPv4, caractérisé ainsi par une haute consommation énergétique ainsi qu'une variabilité (écart-type) en inconstante augmentation.

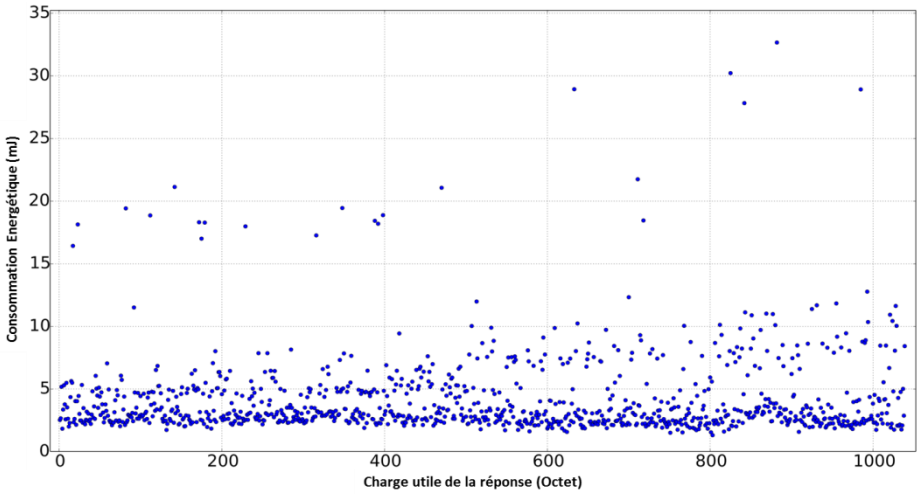

Figure 7 Écart-type de la consommation énergétique de l'objet du 2ème saut en environnement réel sur IPv6

4 Travaux futurs : une approche de profilage énergétique matérielle temps réel

Dans nos travaux en cours, nous prenons en compte les problématiques et constatations tirées de nos expérimentations présentées dans ce document, afin de concevoir une nouvelle approche de mesure énergétique, ou plus spécifiquement, de profilage énergétique. Cette approche de mesure en cours de développement est une approche matérielle temps réel. Plus précisément, nous nous basons sur une plate-forme de mesure *matérielle* pour mesurer le courant réel débité par l'objet alors qu'il est en cours d'exécution réel (non simulé), d'où le terme, *temps réel*. Notre objectif en choisissant une telle approche, est de capturer lors de la mesure, les impacts réels de l'environnement de déploiement sur l'objet, ainsi que d'observer comment ce dernier réagit d'un point de vu énergétique. Intégrée dans un cycle de développement logiciel pour systèmes embarqués, cette approche permettra au développeur du logiciel d'analyser précisément et finement le comportement énergétique conjoint de l'objet et du logiciel embarqué, puis de mettre en œuvre des modifications de ces derniers en conséquence.

La mesure énergétique en elle-même n'apporte qu'une vision globale de la consommation énergétique de l'objet. Nous voulons aller plus loin en proposant via notre approche un mécanisme de *profilage énergétique* du logiciel embarqué. Ainsi, la mesure énergétique globale produite peut être décomposée, afin d'obtenir plus de précision sur la consommation du logiciel, qui est au final, le responsable dans la consommation énergétique de l'objet matériel. Dans nos travaux, afin d'obtenir une analyse à grain fin de la consommation, nous fixons le grain de mesure à un niveau fonctionnel. Spécifiquement, un développeur pourra obtenir de notre approche, la consommation énergétique de toutes (ou une majorité) les fonctions composant le code source de son logiciel embarqué. Ainsi, il pourra tracer la consommation énergétique à travers les fonctions exécutées et détecter les points chauds de consommation dans son code source.

Néanmoins, plusieurs problématiques peuvent naître de la nature de notre approche de mesure : 1) la plate-forme de mesure matérielle ; 2) la lourdeur de la procédure pour le développeur ; 3) La visualisation des résultats.
Concernant le premier point, nous utilisons en l'état actuel de nos travaux, un instrument de mesure haute précision mais qui reste onéreux (*Agilent N6705A*). De ce fait, afin d'avoir une méthode utilisable par le plus grand monde, nous concevons une nouvelle plate-forme de mesure bas prix, basée sur un microcontrôleur ARM Cortex M4, qui pourra offrir des performances et une précision suffisantes pour les objectifs fixés.

Afin de pouvoir profiler énergétiquement le code source, nous instrumentons ce dernier afin d'avoir des informations logiques sur l'exécution du logiciel embarqué, en addition de la mesure énergétique. Ainsi, nous insérons à chaque début et fin de fonction, 3 instructions nous permettant de récupérer l'adresse de la fonction courante, l'adresse de la fonction appelante ainsi qu'une estampille temporelle générée, par exemple, par le compteur du processeur. Les informations d'adressage nous permettent de reconstituer l'arbre d'exécution dynamique du logiciel. Les estampilles temporelles corrélées à la trace énergétique sous forme de transposition temporelle nous permettent de déterminer le début et fin de chaque fonction sur la trace de puissance électrique, et donc de définir la consommation énergétique de chacune de ces fonctions en reformant de plus, via l'arbre d'exécution dynamique, la causalité énergétique existante entre les fonctions.

Cette tâche pouvant être lourde, nous avons décidé de totalement l'automatiser afin de la rendre transparente du point de vue du développeur, via l'utilisation de l'infrastructure de compilation Clang/LLVM. De ce fait, nous utilisons le *frontend* de compilation Clang afin d'instrumenter automatiquement le code source, en insérant les tags précédemment décrits dans toutes les fonctions pouvant être appelées depuis le point d'entrée du logiciel ou alors un sous ensemble de celles-ci défini par le développeur.

Enfin, dans le but de guider et aider le développeur à analyser les résultats brut du profilage énergétique, nous offrons la possibilité à ce dernier de visualiser graphiquement les consommations énergétiques de chacune des fonctions, ainsi que l'arbre d'appel dynamique via l'utilisation du logiciel KCacheGrind. Ce dernier est initialement utilisé conjointement avec CallGrind pour l'affichage des résultats d'un profilage des cycles processeur d'une application. Néanmoins, quelques modifications suffisent pour le rendre viable à un affichage d'un profil énergétique.

5 Conclusion

Dans ce papier, nous avons présenté le profil et le comportement énergétique global d'une application web embarquée adaptée à l'idée d'interopérabilité poussée par l'internet des objets. Nous avons expérimentalement évalué cette dernière selon deux configurations réseaux et deux environnements de déploiement. Grâce à cela, nous avons montré que dans bien des cas de la vie réelle, la simulation ne peut représenter un substitut de qualité à une expérimentation en environnement réel. Pour ne citer que quelques raisons, nous avons identifié les interférences radios présentes en environnement réel tel que le WiFi en tant que perturbations ayant un impact significatif sur la qualité du lien radio de communication 802.15.4 entre les objets. Cette détérioration du lien implique un accroissement non négligeable mais surtout variable de la consommation énergétique de chaque objet du réseau. Spécifiquement, cela prend une ampleur plus importante lors de l'utilisation d'un protocole nécessitant une garantie de livraison des données car une retransmission des fragments perdus devient alors nécessaire.

La simulation ne représente pas réellement l'entité à blâmer, car cette dernière repose sur un modèle qui doit être à même de retranscrire les impacts d'un environnement de déploiement réel. Néanmoins, ces derniers étant nombreux et variables dans le temps, la création d'un modèle de simulation précis devient ardue et lourde à réaliser, ainsi qu'à répéter pour les nombreuses plate-formes embarquées hétérogènes existantes. Afin de passer outre cette problématique, nous travaillons sur la mise en place d'une méthode de profilage énergétique matérielle. Celle-ci nous permet d'obtenir des chiffres de consommation précis et réels tout en capturant les impacts de l'environnement de déploiement. Nous pensons ainsi pouvoir aider les développeurs d'applications embarquées dans le cadre de l'IdO, à mieux concevoir leurs logiciels embarqués dans le but de garantir une durée de vie maximale à l'objet connecté.

Références bibliographiques

[1] R. Lim, F. Ferrari, M. Zimmerling, C. Walser, P. Sommer, and J. Beutel, "FlockLab: A Testbed for Distributed, Synchronized Tracing and Profiling of Wireless Embedded Systems," in *Proceedings of the 12th International Conference on Information Processing in Sensor Networks*, New York, NY, USA, 2013, pp. 153–166.

[2] H. Ritter, J. Schiller, T. Voigt, A. Dunkels, and J. Alonso, "Experimental evaluation of lifetime bounds for wireless sensor networks," in *Wireless Sensor Networks, 2005. Proceeedings of the Second European Workshop on*, 2005, pp. 25–32.

[3] P. Dutta, M. Feldmeier, J. Paradiso, and D. Culler, "Energy Metering for Free: Augmenting Switching Regulators for Real-Time Monitoring," in *Information Processing in Sensor Networks, 2008. IPSN '08. International Conference on*, 2008, pp. 283–294.

[4] R. Fonseca, P. Dutta, P. Levis, and I. Stoica, "Quanto: Tracking Energy in Networked Embedded Systems," in *Proceedings of the 8th USENIX Conference on Operating Systems Design and Implementation*, Berkeley, CA, USA, 2008, pp. 323–338.

[5] A. Dunkels, "Powertrace: Network-level power profiling for low-power wireless networks." Technical Report T2011:05, Swedish Institute of Computer Science, 2011.

[6] Q. Li, M. Martins, O. Gnawali, and R. Fonseca, "On the Effectiveness of Energy Metering on Every Node," in *2013 IEEE International Conference on Distributed Computing in Sensor Systems (DCOSS)*, 2013, pp. 231–240.

[7] V. Shnayder, M. Hempstead, B. Chen, G. W. Allen, and M. Welsh, "Simulating the power consumption of large-scale sensor network applications," in *In Sensys*, 2004, pp. 188–200.

[8] N. Fournel, A. Fraboulet, and P. Feautrier, "eSimu: a Fast and Accurate Energy Consumption Simulator for Real Embedded System," in *World of Wireless, Mobile and Multimedia Networks, 2007. WoWMoM 2007. IEEE International Symposium on a*, 2007, pp. 1–6.

[9] A. Dunkels, *The ContikiMAC Radio Duty Cycling Protocol*. 2011.

[10] J. Polastre, J. Hill, and D. Culler, "Versatile Low Power Media Access for Wireless Sensor Networks," in *Proceedings of the 2Nd International Conference on Embedded Networked Sensor Systems*, New York, NY, USA, 2004, pp. 95–107.

[11] M. Buettner, G. V. Yee, E. Anderson, and R. Han, "X-MAC: A Short Preamble MAC Protocol for Duty-cycled Wireless Sensor Networks," in *Proceedings of the 4th International Conference on Embedded Networked Sensor Systems*, New York, NY, USA, 2006, pp. 307–320.
[12] D. Moss and P. Levis, "BoX-MACs: Exploiting Physical and Link Layer Boundaries in LowPower Networking," 2008.
[13] J. W. Hui and D. E. Culler, "IP is Dead, Long Live IP for Wireless Sensor Networks," in *Proceedings of the 6th ACM Conference on Embedded Network Sensor Systems*, New York, NY, USA, 2008, pp. 15–28.
[14] C. Westphal, "Layered IP Header Compression for IP-enabled Sensor Networks," in *Communications, 2006. ICC '06. IEEE International Conference on*, 2006, vol. 8, pp. 3542–3547.
[15] A. Dunkels, "Full TCP/IP for 8 Bit Architectures," in *Proceedings of the First ACM/Usenix International Conference on Mobile Systems, Applications and Services (MobiSys 2003)*, San Francisco, 2003.
[16] G. Mulligan, "The 6LoWPAN Architecture," in *Proceedings of the 4th Workshop on Embedded Networked Sensors*, New York, NY, USA, 2007, pp. 78–82.
[17] A. Dunkels, "SICSlowpan - Internet-connectivity for Low-power Radio Systems," SICS, 2008.
[18] J. Granjal, E. Monteiro, and J. S. Silva, "Network-layer security for the Internet of Things using TinyOS and BLIP," *Int. J. Commun. Syst.*, vol. 27, no. 10, pp. 1938–1963, 2014.
[19] A. Dunkels, T. Voigt, and J. Alonso, "Making TCP/IP Viable for Wireless Sensor Networks," in *Proceedings of the First European Workshop on Wireless Sensor Networks (EWSN 2004), work-in-progress session*, Berlin, Germany, 2004.
[20] R. Braden, "Requirements for internet hosts – communication layers - RFC 1122," RFC, 1989.
[21] E. Altman and T. Jiménez, "Novel Delayed ACK Techniques for Improving TCP Performance in Multihop Wireless Networks," *Pers. Wirel. Commun.*, vol. 2775, pp. 237–250, 2003.