

République Algérienne Démocratique et Populaire
Université Abou Bakr Belkaid–Tlemcen
Faculté des Sciences
Département d'Informatique

Mémoire de fin d'études

Pour l'obtention du diplôme de Master en Informatique

Option: Réseaux et Systèmes Distribués (R.S.D)

Thème

MISE EN PLACE D'UNE APPLICATION D'AGREGATION DES DONNEES DANS UN RESEAU DE CAPTEURS SANS FIL SOUS LA PLATFORME CONTIKI

Réalisé par :

- FEKIH Kawther
- GORINE Asmàa

Présenté le 22 juin 2015 devant le jury composé de MM.

- Mr. BENAMAR Abdelkrim (Président)
- Mme. LABRAOUI Nabila (Encadreur)
- Mr. BEKARA Chakib (Examineur)
- Mr. BELHOUCINE Amine (Examineur)

Année universitaire: 2014-2015

*Soyons reconnaissants aux personnes qui nous
Donnent du bonheur ; elles sont les charmants jardiniers
Par qui nos âmes sont fleuries.
Marcel Proust (1871-1922).*

Merçi !

Nous remercions dieux le plus puissant qui nous a donné la force et la volonté pour réaliser ce travail.

Ah !les remerciements !ça permet à chacun de remémorer les moments importants ainsi que les personnes qui nous a accompagnés et encouragés.

Un grand merci ira à Madame LABRAOUI Nabila d'avoir accepter d'être notre encadreur, pour leur encadrement efficace dans la conduite de ce travail, pour la confiance qui nous a donné, ses précieux conseils, son soutien et ses encouragements permanents dans nos moments de doute.

Un second grand merci ira aux membres de jury d'avoir accepter de juger notre modeste travail.

Nos remerciements vont également à nos parents qui nous ont encouragés, qui nous ont appris à travailler honnêtement et qui ont toujours supporté moralement et financièrement pendant toutes nos longues années d'études. Nous concluons, en remercions vivement tous les personnes qui nous a soutenu et nous a aidé durant la préparation de ce mémoire de près ou de loin.

Dédicaces

Je dédie ce modeste travail :

A mes chères parents, mon très cher père pour sa patience et tous ses efforts et à ma mère pour m'avoir encouragé à reprendre les études.

A ma sœur Meriem.

A mes frères : Farouk et Tarek.

A toute ma famille : GORINE et ZENASNI.

A mon encadreur Madame LABRAOUI Nabila pour son excellence encadrement et ses conseils judicieux.

A mon binôme Kawther avec qui j'ai partagé de belles années d'études.

A tous mes professeurs ainsi que tous les étudiants de la promotion RSD2015.

A mes amies : Samira, Hanane, Badia, Amina, Karima...

Une dédicace spéciale pour Houcine, pour ses encouragements et ses conseils.

Et toute personne que je connais et qui sont chers.

GORINE ASMÀA.

Dédicaces

Nul ne peut atteindre l'aube sans passer par le chemin de la nuit

— Khalil Gibran—

Je dédie ce modeste travail :

À mon papa chéri, école de mon enfance, qui a été mon ombre durant toutes ces années d'études, et qui a veillé tout au long de ma vie à m'encourager, à m'aider et à me protéger.

À celle qui m'a donné la vie, le symbole de tendresse, qui s'est sacrifiée pour mon bonheur et ma réussite, à ma petite maman chérie que j'aime énormément. Elle seule mérite bien que je la couvre de bisous et de diamants.

À ma sœur, je dis juste merci Yasmine, merci pour tout.

À mes deux experts Lotfi & Selma, qui ont toujours su comment me rendre le sourire, merci mes poussins.

À Oussama, merci pour ton amour, ton amitié sans faille, ton encouragement sans limite et pour m'avoir fait rire souvent. Ce travail t'est dédié.

À Asmaà, mon binôme, ma copine et ma sœur, merci d'avoir partagé ma joie et ma peine durant ces cinq longues années.

À mon grand-père qui nous a quittés en silence, j'aurai aimé partager cette joie avec toi, tu me manques énormément, reposes en paix.

À ma grand-mère, et à toute la famille FEKIH.

À mes copines : Mariem, Ikram, Sarah. Merci pour les moments de folie que nous avons partagé ensemble.

Et enfin un dédicace spécial pour mes professeurs et pour la promotion RSD 2015.

FEKIH Kawther

Table des matières

INTRODUCTION GENERALE	4
<u>CHAPITRE 1: LES RESEAUX DE CAPTEURS SANS FIL</u>	
1. INTRODUCTION.....	5
2. LES CAPTEURS	5
2.1. DEFINITION D'UN CAPTEUR	5
2.2. LES COMPOSANTS D'UN CAPTEUR SANS FIL	5
3. PRESENTATION DES RESEAUX DE CAPTEURS.....	6
4. LES APPLICATIONS.....	7
5. LES LIMITATIONS D'UN RESEAU DE CAPTEUR SANS FIL	8
6. AGREGATION DE DONNEES DANS UN RCSF	9
7. CONCLUSION.....	11
<u>CHAPITRE 2: LES SYSTEMES D'EXPLOITATION POUR LES RCSF</u>	
1. INTRODUCTION	13
2. LES DIFFERENTS SYSTEMES D'EXPLOITATION POUR LES RCSF	13
2.1. SYSTEMES MULTITACHES	13
2.1.1 <i>Avantages et inconvénients</i>	14
2.2. LES SYSTEMES BASES SUR LES EVENEMENTS	14
2.2.1. AVANTAGES ET INCONVENIENTS.....	14
2.3. LES SYSTEMES HYBRIDES	15
2.3.1. LE SYSTEME D'EXPLOITATION CONTIKI	15
2.3.1.1. <i>L'architecture hybride du système Contiki</i>	15
2.3.1.2. <i>Les caractéristiques</i>	18
2.3.1.3. <i>Les avantages et inconvénients du système Contiki</i>	21
2.3.1.4. <i>Un simulateur réseau pour Contiki : Cooja</i>	22
3. COMPARAISON	23
4. CONCLUSION	25
<u>CHAPITRE 3: SIMULATION ET REALISATION DE L'APPLICATION DES DONNEES AGREGES</u>	
1. INTRODUCTION.....	27
2. ENVIRONNEMENT DE TRAVAIL ET OUTILS DE DEVELOPPEMENT	27
• OUTILS MATERIELS	27
• OUTILS LOGICIELS.....	28
3. LES ETAPES DE DEVELOPPEMENT DE L'APPLICATION	29
• INSTALLATION LOGICIELLE	29
• INSTALLATION MATERIELLE.....	29

4. LES PARTIES DU CODE	29
4.1. COMPREHENSION DU CODE.....	29
4.2. LA PARTIE SENDER.....	32
4.3. LA PARTIE CLUSTER-HEAD	32
4.4. LA PARTIE STATION DE BASE	32
5. LA SIMULATION SOUS COOJA	32
6. REALISATION DE L'APPLICATION SUR UNE PLATEFORME MATERIEL TMOTE SKY	36
8. CONCLUSION.....	41
CONCLUSION GENERALE	43
REFERENCES BIBLIOGRAPHIQUES.....	46
ANNEXE : INSTALLATION DE CONTIKI	49

Introduction générale

Introduction générale

En 2003, selon le magazine technology review du MIT, le réseau de capteurs sans fil est l'une des dix nouvelles technologies qui bouleverseront le monde et notre manière de vivre et de travailler, car il offre des solutions économiquement intéressantes et facilement déployables à la surveillance à distance et au traitement des données dans les environnements hostiles.

Un réseau de capteur sans fil (RCSF) est un ensemble de dispositifs très petits à faible coût nommés nœuds capteurs, variant de quelques dizaines d'éléments à plusieurs milliers. Dans ces réseaux, Chaque nœud est capable de surveiller son environnement et de réagir en cas de besoin en envoyant l'information collectée à un ou plusieurs points de collecte à l'aide d'une connexion sans fil.

La majorité des travaux sur les réseaux de capteurs vise à réduire la consommation énergétique. En effet les réseaux de capteurs sont destinés, le plus souvent, à relever des informations dans des environnements difficilement accessibles sans aucune intervention humaine, il est donc difficilement envisageable de trouver une autre source d'énergie par conséquent, leur durée de vie égale à la durée de vie des batteries. Les nœuds qui n'ont plus de batteries peuvent créer de sérieux problèmes de perte d'informations et de partitionnement du réseau. Par conséquent, il est important de diminuer la consommation d'énergie de chaque nœud et de mieux gérer la première source d'énergie. On outre, la consommation énergétique est donc la contrainte clef dans les réseaux de capteurs puisque l'énergie est considérée comme une ressource précieuse.

Une des méthodes très utilisée pour minimiser la consommation est « l'agrégation des données ». Cette technique permet de remplacer les lectures individuelles des capteurs en une seule lecture via l'utilisation d'une fonction d'agrégat telle que la moyenne ou la somme. Un nœud désigné comme chef de groupe nommé cluster-head va calculer l'agrégat et envoyer un seul message à la station de base au lieu de plusieurs. Ce qui permet d'économiser considérablement la consommation d'énergie et ainsi allonger la durée de vie du réseau de capteurs. Dans notre travail, nous avons simulé une application de surveillance utilisant l'agrégation des données grâce au

Introduction générale

simulateur Cooja et nous avons également fait un testbed, i.e réalisé l'application sur des capteurs de type Telosb dans un environnement réel.

Ce manuscrit s'articule autour des chapitres suivants :

nous présentons dans le premier chapitre les réseaux de capteurs sans fil et leurs domaines d'application ainsi que l'agrégation de donnée qui est une solution pour l'utilisation efficace de ressources et d'énergie et l'évitement de la congestion dans le réseau. Le deuxième chapitre est une présentation des outils matériels et logiciels nécessaires à la réalisation de notre application. Le troisième chapitre nous présentons notre application.

Enfin, nous terminons par une conclusion générale en donnant quelques perspectives.

Chapitre1

Les réseaux de capteurs sans fil

1. Introduction

L'évolution dans le domaine des communications sans fil et l'informatique mobile gagne de plus en plus de popularité et les composants mobiles deviennent de plus en plus fréquents (PDA, LAPTOPS, HANDSETS...). Ceci a permis l'apparition d'un nouveau type de réseaux sans fils appelé réseaux de capteur sans fils (RCSF en Français ou WSN en anglais) qui sont devenu de plus en plus populaires du fait de leur facilité de déploiement. Ces réseaux jouent un rôle primordial au sein des réseaux informatiques. Ils offrent des solutions ouverts pour fournir la mobilité ainsi que des services essentiels là ou l'installation d'infrastructure n'est pas possible.

Dans ce chapitre, nous présentons les réseaux de capteurs sans fils et leurs domaines d'applications.

2. Les capteurs

2.1. Définition d'un capteur

Un capteur est un dispositif ayant pour tâche de transformer une grandeur physique en une grandeur quantifiable, généralement électrique, qui peut être interprétée en une donnée binaire exploitable et compréhensible par un système d'information [1,2] On trouve des capteurs de position, d'accélération, de mouvement, de luminosité, de température...

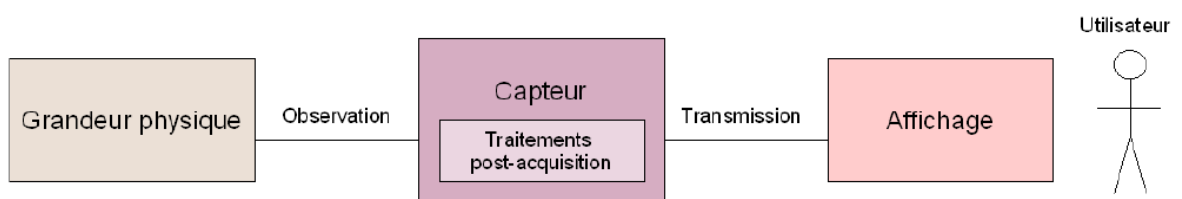


Figure I. 1: Schématisation de la chaîne de collecte de données [3].

2.2. Les composants d'un capteur sans fil

Un nœud capteur est composé de quatre unités principales, qui sont présentées dans la figure 1.2. [4]

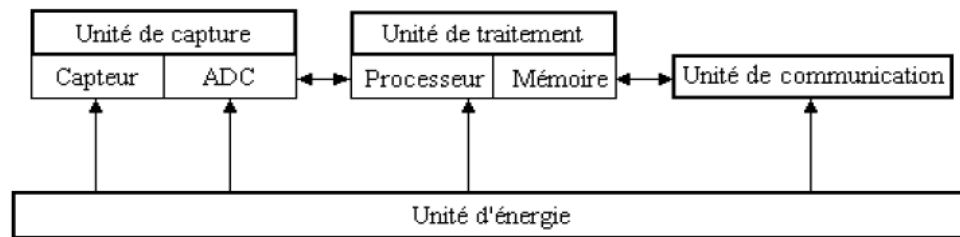


Figure 1. 2.: les composants d'un nœud capteur [4].

- **Unité de capture** (*Sensing unit*) : elle est composée de deux sous unités, un dispositif de capture physique qui prélève l'information de l'environnement local et un convertisseur analogique/numérique appelé ADC (*Analog to Digital Converters*).
- **Unité de traitement de données** (*Processing unit*): les données captées sont communiquées au processeur où elles sont stockées dans la mémoire.
- **Unité de transmission de données** (*Transceiver unit*) : elle est composée d'un émetteur/récepteur (module radio) permettant la communication entre les différents nœuds du réseau.
- **Unité d'énergie** (*Power unit*) : C'est un élément primordial de l'architecture du capteur, le capteur doit disposer de sa propre source d'énergie, c'est elle qui fournit en énergie toutes les autres unités. Cette unité se trouve sous la forme de batteries ou piles alimentant le capteur.

3. Présentation des réseaux de capteurs

Un réseau de capteurs sans fil est un réseau ad hoc avec un grand nombre de nœuds qui sont des micro-capteurs capables de récolter et de transmettre des données environnementales d'une manière autonome. La position de ces nœuds n'est pas obligatoirement prédéterminée. Ils peuvent être aléatoirement dispersés dans une zone géographique, appelée « champ de captage » correspondant au terrain d'intérêt pour le phénomène capté (ex : lâché de capteurs sur un volcan pour étudier les phénomènes volcanologiques et leurs évolutions, etc....) [5]

Les nœuds capteurs utilisent une communication sans fil pour acheminer les données captées avec un routage multi-sauts vers n nœuds collecteur appelé nœud puits (ou sink) qui va transmettre via internet ou satellite, ces informations à l'utilisateur du réseau (voir figure 1.1). Ainsi L'utilisateur peut adresser des requêtes aux autres nœuds du

réseau, précisant le type de données requises et récolter les données environnementales captées par le biais du nœud puits [6].

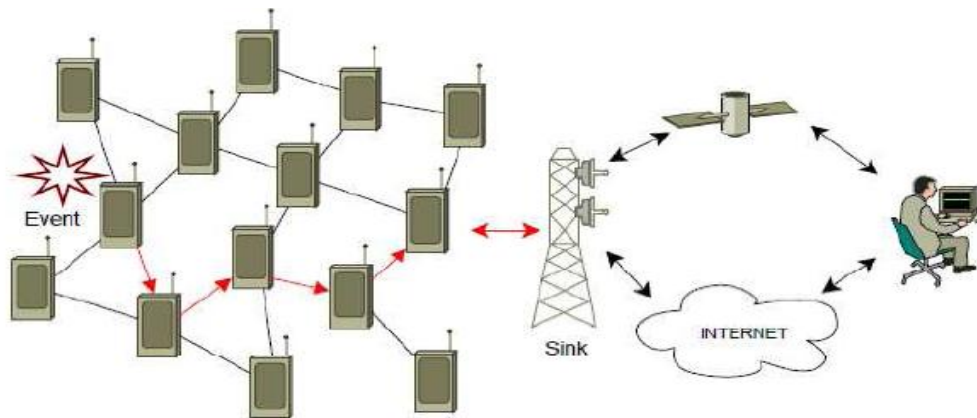


Figure 1. 3: Architecture d'un réseau de capteur [6].

4. Les applications

Le concept de réseaux de capteurs sans fil est basé sur une simple équation [7] :

« **Capteurs + Processeur + Radio = Une centaine d'applications potentielles** »

Les RCSF peuvent avoir beaucoup d'applications. Nous citons parmi elles :

- **Les applications militaires** : Impulser de nouvelles stratégies de communication ou encoir servir à détecter des dispositifs nucléaires et les dépister, à surveiller les activités d'ennemies, ou à analyser un endroit stratégique et difficile d'accès avant un déploiement de troupes.
- **Les applications environnementales** : Des thermo-capteurs dispersés à partir d'un avion sur un forêt peuvent signaler un éventuel début d'incendie : ce qui permettre une meilleur efficacité pour la lutte contre les feux de forêt, surveiller des catastrophes naturelles, détecter des pollutions et suivre des écosystèmes.
- **Les applications médicales** : La surveillance permanente des patients et la possibilité de collecter des informations physiologiques de meilleures qualités facilitant le diagnostique de maladies grâce à des micros-capteurs qui pourront être avalés ou implantés sous la peau (surveillance de la glycémie, détection de cancers...).
- **Les applications domestiques** : En plaçant sur le plafond ou dans le mur des capteurs qui permettent d'automatiser plusieurs opérations domestiques tels-

que : la lumière, la climatisation et le chauffage en fonction de la localisation des personnes.

- **Les applications agricoles :** Les capteurs collectent des données sur les cultures et la qualité du sol et les transmettent à une station centrale présente dans la ferme.
- **Les applications industrielles :** Les RCSF peuvent donner des indicateurs sur l'état d'une machine ou sur le fonctionnement de l'ensemble d'une chaîne de production.

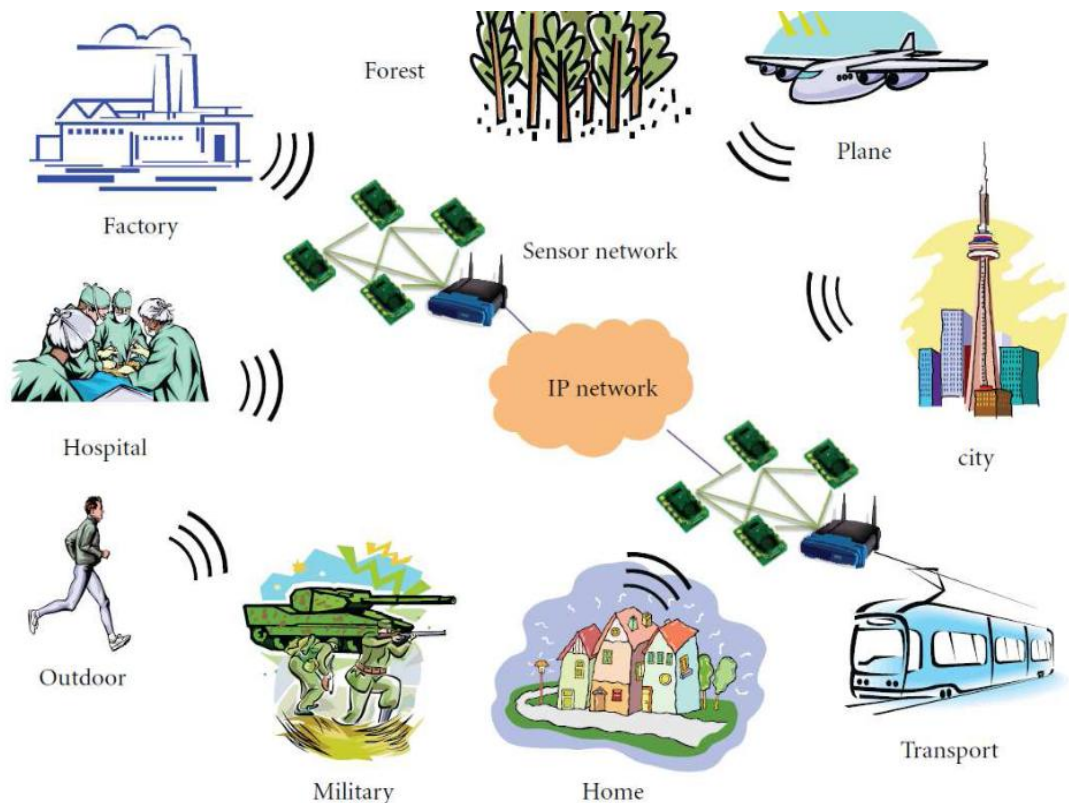


Figure I. 4: Quelques domaines d'application des RCSF [8].

5. Les limitations d'un réseau de capteur sans fil

Au niveau de la communication :

- La bande passante : Limitée et partagée par tous les nœuds du réseau WSN.
- Porté radio limitée à quelque dizaines de mètres et faible débit.
- Collision entre les centaines voire les milliers des nœuds.
- Perte de données à cause de la transmission radio : Le renvoi des paquets est limité à cause du manque en énergies

Au niveau du matériel :

- Puissance de calcul limité fonctionnant dans la majorité des cas avec des registres 8 ou 16 bits.
- Mémoire limitée (2 à 250Kode RAM et 1 à 3250 de mémoire FLACH).
- Energie limité : un facteur critique. Toute solution implémentée (sécurité, routage ...) doit minimiser sa consommation d'énergie [9].

6. Agrégation de données dans un RCSF

Dans un capteur, la problématique principale concerne la consommation d'énergie : en effet, ces derniers doivent restés opérationnels le plus longtemps possibles, dans des conditions parfois difficiles (ex : lâchés par avion sur les parois d'un volcan). Comme il n'est pas possible de recharger leur énergie ni changer les piles par exemple (on ne sait pas toujours où se trouvent les capteurs), il est nécessaire d'économiser au maximum l'énergie consommée par ces derniers [5].

On estime que la transmission des données d'un capteur représente environ 70% de sa consommation d'énergie.

De plus, les réseaux de capteurs étant assez denses en général, cela signifie que des nœuds assez proches en terme de distance peuvent capter les mêmes données (température, pression, humidité équivalentes par exemple) et donc il apparaît nécessaire d'introduire une approche intéressante est d'agrèger les données basée sur le principe que la station de base n'a pas besoin de toutes les données collectées par chaque capteur en raison de leur redondance dans le réseau mais seulement d'un agrégat de données effectué au niveau d'un nœud appelé agrégateur en utilisant des simples fonctions d'agrégat telles que :la moyenne, la médiane, la somme, le max ou le min...qui permettent à partir d'une série de n messages reçus par un « chef de zone » (capteur chef d'une zone) de ne renvoyer vers le puits qu'un seul message résumant l'information contenue dans ces n messages (voir figure) et qui aide à préserver l'énergie en évitant la duplication de l'information et donc d'augmenter la durée de vie du réseau [5,6].

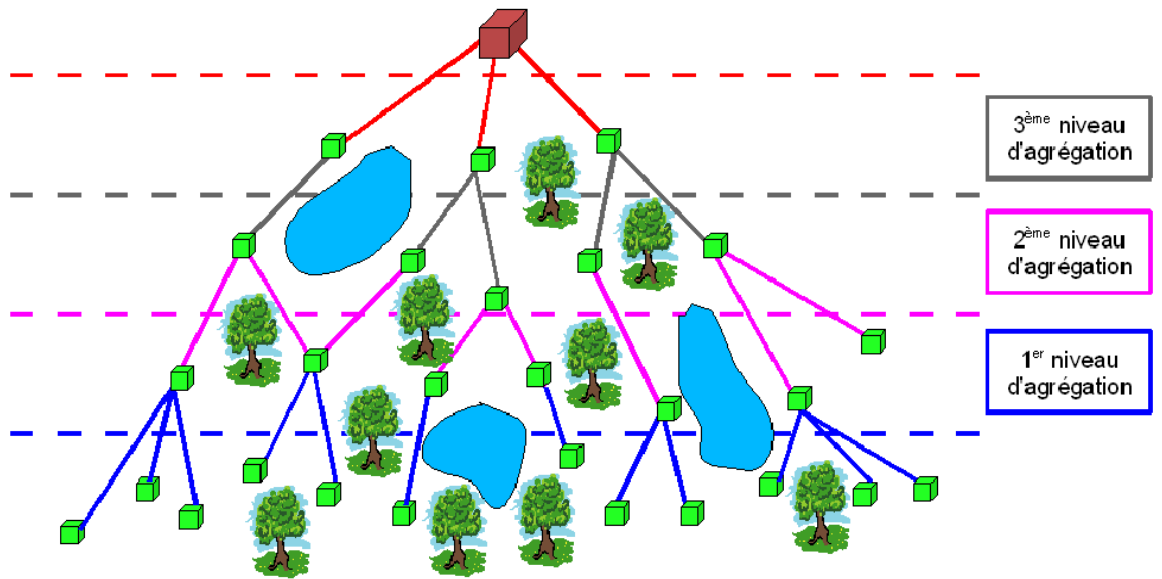


Figure I. 5: Arbre d'agrégation de données [3].

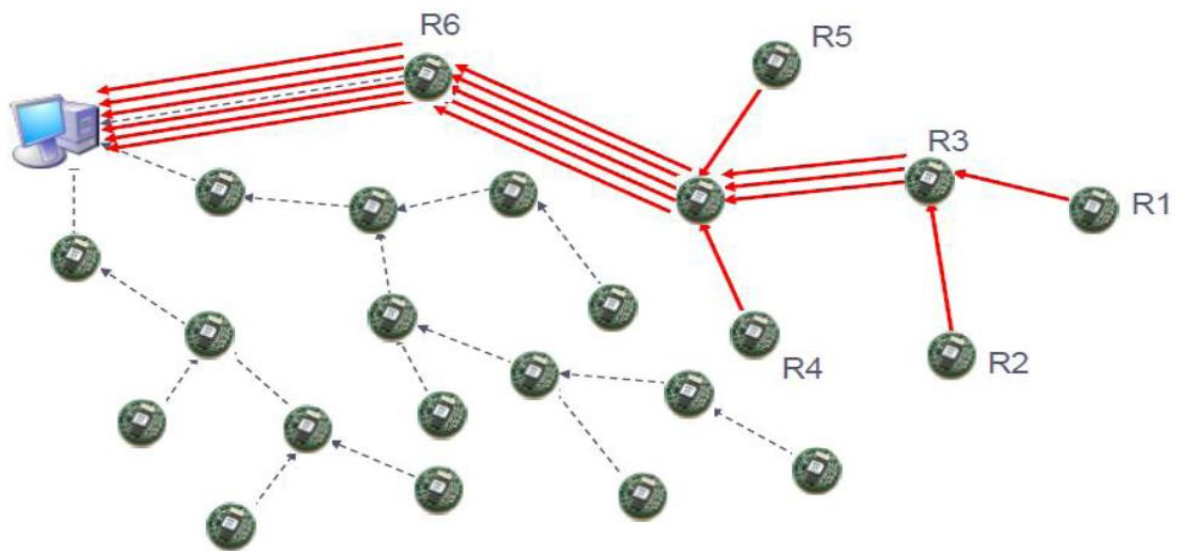


Figure I. 6: Exemple sans agrégation [5].

Au total, 18 messages sont envoyés sur le réseau de capteurs. En utilisant le mécanisme d'agrégation de données, on obtient un total de 7 messages envoyés sur le réseau :

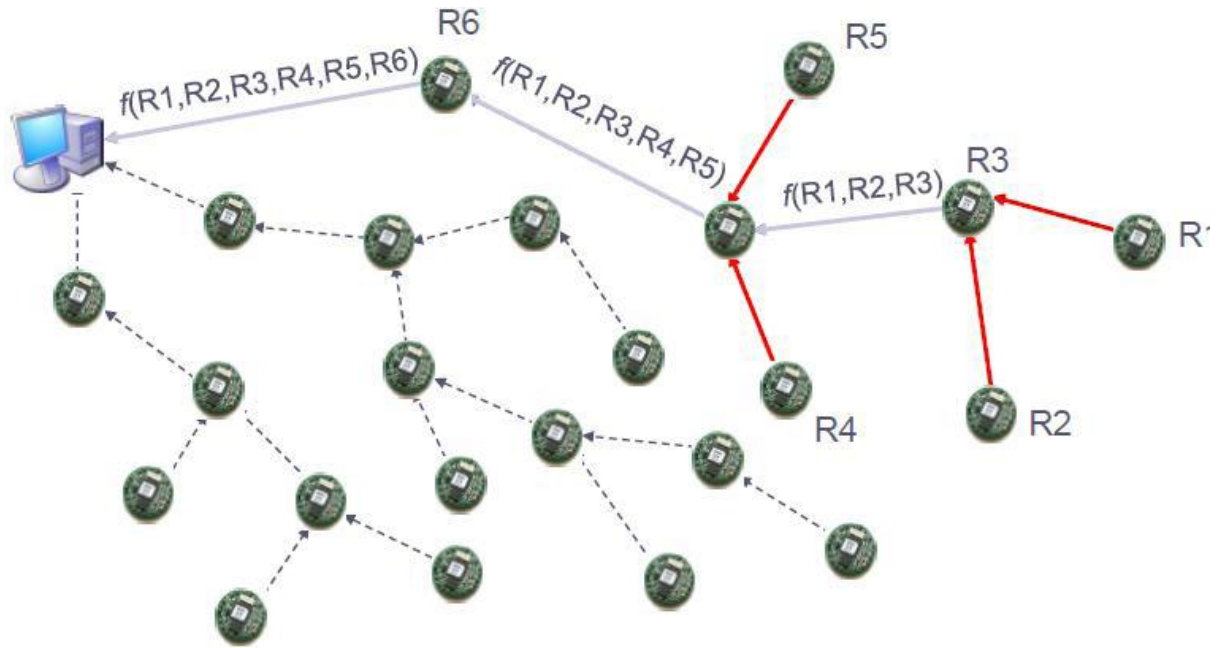


Figure I. 7: exemple avec agrégation de données [5].

7. Conclusion

Le domaine des réseaux de capteurs sans fil représente une des technologies du futur car ces réseaux possèdent des caractéristiques qui les différencient des autres types de réseaux sans fil et ils peuvent être utilisés dans de nombreux domaines : santé, agriculture, industrie...

Nous avons présenté dans ce chapitre les WSN. Cependant la mise en place d'une application de réseau de capteurs doit prendre en considération les contraintes qui caractérisent les nœuds. En effet, les WSN se caractérisent par une capacité énergétique limitée ce qui minimise la durée de vie du réseau donc nous avons mis le point sur l'agrégation de données qui permet l'économie de l'énergie en réduisant au maximum le nombre de communications.

Dans le chapitre qui suit, nous présentons le système d'exploitation Contiki ainsi que le simulateur Cooja.

Chapitre2
Les systèmes d'exploitation
pour les RCSF

1. Introduction

Les systèmes d'exploitation représentent un socle sur lequel s'appuient les programmes d'application. Ils servent de lien ou d'interface entre l'architecture matérielle et la partie logicielle. Dans les réseaux de capteurs sans fil (RCSF), les systèmes d'exploitation possèdent différentes fonctionnalités qui conservent ce rôle.

Le thème central de ce chapitre est le système d'exploitation dédié aux RCSF Contiki. Dans un premier temps, les différents systèmes d'exploitation seront présentés. Puis, dans un second temps, le système d'exploitation contiki sera détaillé. Enfin, une synthèse avec les notions essentielles à retenir conclura ce chapitre.

2. Les différents systèmes d'exploitation pour les RCSF

2.1. Systèmes multitâches

Dans le domaine des systèmes d'exploitation temps réel, une distinction existe entre les systèmes pilotés par le temps et ceux pilotés par les événements [10]. Les premiers effectuent leurs traitements à des instants déterminés à partir d'un référentiel temporel interne. Les seconds systèmes n'accomplissent leur tâche qu'à la suite de stimuli provenant de leur environnement. Cette classification se retrouve partiellement dans les systèmes dédiés aux RCSF [11].

Certains travaux de recherche ont adapté, au domaine des RCSF, l'architecture des systèmes d'exploitation multitâches classique. Le système d'exploitation UNIX est un exemple de système multitâche mais ne peut être utilisé dans les RCSF pour deux raisons majeures. La première est qu'il ne s'agit pas d'un système temps réel qui est une propriété importante pour les RCSF. L'autre raison est son empreinte mémoire qui est trop importante pour un capteur sans fil.

Dans les systèmes multitâches nous avons le système d'exploitation MANTIS, conçu au sein de l'université du Colorado à Boulder, qui se base sur le partage de temps. Les ressources logicielles et matérielles du système sont ainsi affectées alternativement et pour un temps fixé entre les différents processus légers en cours d'exécution.

2.1.1 Avantages et inconvénients

Parmi les avantages des systèmes multitâches, c'est un système qui a une logique de programmation classique (UNIX, ...) robuste, tolérant aux fautes d'une ou plusieurs tâches.

Le principal inconvénient des systèmes multitâches reste l'espace mémoire alloué à chaque processus ou tâche [12] et la durée des changements de contexte [13]. Ainsi la taille du système dépend énormément des traitements à réaliser et donc de l'application supportée.

2.2. Les systèmes basés sur les événements

De part leur mode de fonctionnement, les systèmes d'exploitation pilotés par les événements sont adaptés aux RCSF [11]. L'architecture de ce type de système est organisée autour des traitements à effectuer et de la gestion des ressources disponibles. Le système d'exploitation TinyOS, développé à l'Université de Californie de Berkeley, fait partie de cette famille de systèmes d'exploitation et permet d'en illustrer certaines caractéristiques communes [14]. En outre, TinyOS représente un cadre pour le développement de systèmes d'exploitation dédiés aux RCSF.

2.2.1. Avantages et inconvénients

Les systèmes basés sur les événements ont plusieurs inconvénients, ils sont mal ou des fois pas adaptés aux applications temps-réel et nativement multitâches, en plus ils ont une intégration complexe des traitements à longue durée d'exécution, mais le principal inconvénient reste la programmation d'une application demande au préalable la définition d'une machine d'états [14]. Cette dernière peut s'avérer plus ou moins complexe selon la durée des traitements considérés [13].

Par contre, ces systèmes apportent plusieurs avantages :

- Utilisation de la mémoire.
- Pas de compétition pour accéder aux ressources critiques.
- Modularité : construction d'une application par combinaison d'événements.

Les avantages et inconvénients des systèmes basés sur les événements et multitâches, ont conduit à la conception de systèmes d'exploitation hybrides.

2.3. Les systèmes hybrides

Les systèmes hybrides ont combiné les fonctionnalités des systèmes multitâches et des systèmes basés sur les événements afin de bénéficier des avantages de chacun tout en minimisant leurs inconvénients. L'un des premiers systèmes hybrides dédié aux RCSF est le système Contiki qui sera étudié dans le second paragraphe.

2.3.1. Le système d'exploitation Contiki

Contiki est un système open source, léger, flexible et générique qui s'appuie sur un modèle de fonctionnement hybride [15].

Ce système a été développé par un groupe de développeurs de l'industrie et du monde universitaire par ADAM Dunkels de l'institut suédois d'informatique en 2002. Destiné à être embarqué dans des capteurs miniatures ne disposant généralement que de ressources limitées, Contiki a présenté l'idée d'utiliser la communication IP dans des réseaux de capteurs basse consommation. En plus il supporte les protocoles IPV6 et 6LOWPAN cela s'avère particulièrement utile dans la mesure où les nœuds communiquent en IPV6 et utilisent le standard 802.15.4 définie par l'IEEE.

Contiki contient deux piles de communications : uIP et Rime [16]

uIP est une petite pile de TCP/IP RFC-CONFORME qui permet a Contiki de communiquer sur internet.

Rime est une pile de communication légère conçue pour des radios basse puissance. Il fournit une vaste gamme de communications primitives.

La programmation d'application dans ce système d'exploitation se fait dans le langage C et pour permettre la concurrence, contiki utilise des protothreads qui sont en fait des threads légers sans pile spécialement conçus pour les environnements avec peu de mémoire comme les réseaux de capteurs.

2.3.1.1. L'architecture hybride du système Contiki

Pour économiser de la mémoire, l'approche basée sur les événements a été, dans un premier temps, privilégiée. Les incertitudes sur la taille de la pile d'exécution et le nombre de processus à prévoir sont ainsi évitées. Cependant, les opérations longues telles que la cryptographie de données s'accordent mal avec l'utilisation des événements par la monopolisation du système pour une durée conséquente. Pour cette raison, dans un second temps, le système Contiki s'est vu ajouter un composant qui lui permet de fonctionner comme un système multitâche. Ce composant est une bibliothèque de

fonctions optionnelle appelée explicitement par le programme qui en a besoin. Cette bibliothèque permet la gestion des processus et de leur pile d'exécution respective [3].

Le cœur du système Contiki est composé de différents éléments :



Figure II. 1 : Architecture du système Contiki [15].

• **le noyau :**

Le noyau gère des événements asynchrones et synchrones. Les premiers sont placés dans une file d'attente après leur appel. Les traitements associés à ce type d'événement sont donc déclenchés après un certain délai. A l'inverse, la réponse à un événement synchrone est quasi immédiate [3].

Dans le système Contiki, des fonctions utilisées par les traitements associés à différents événements sont regroupés dans un même service. Cette notion est très proche de celle d'une bibliothèque partagée. Les bibliothèques comme les services sont modifiables et remplaçables dynamiquement en cours d'exécution. Ils offrent donc des possibilités de reconfiguration très intéressantes et utiles si l'on considère le cadre d'utilisation des RCSF [3].

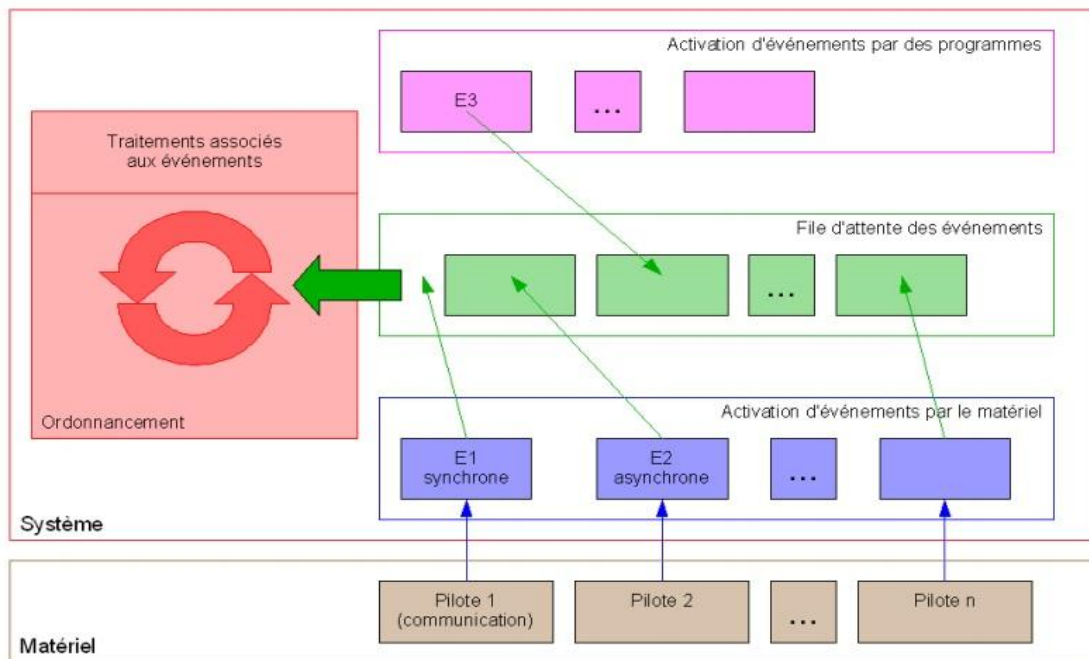


Figure II. 2 : Gestion des événements par le système Contiki [3].

- **le chargeur de programme :**

L'ajout de programmes configure le noyau pour une application donnée. Le système ne contient pas de niveau d'abstraction pour l'économie d'énergie mais offre des informations comme la taille de la file d'attente des événements pour que l'application puisse réaliser cette opération [3].

- **le module de gestion des bibliothèques systèmes :**

En ce qui concerne les bibliothèques, selon leur emplacement, leur reconfiguration est plus ou moins envisageable. Celles qui font partie du cœur du système, le plus souvent intégrées au module de gestion des bibliothèques, sont considérées comme statiques. Les bibliothèques associées aux programmes de l'application et les services ont vocation à être remplacés dynamiquement [3].

- **la pile de communication avec les pilotes pour le matériel :**

Le service de communication établit le lien entre l'application et le matériel à l'aide des pilotes associés. Le noyau du système intervient dans la gestion des événements synchrones activés durant les phases de réception et d'émission d'un paquet ou d'un message. De manière générale, la communication entre services est obtenue par publication d'événements [3].

Jusqu'à présent, seul le fonctionnement basé sur les événements a été abordé. Logiquement, par rapport au système TinyOS, certains principes sont en commun,

d'autres différents. Comme nous l'avons indiqué précédemment, le mode multitâche est assuré par une bibliothèque spécifique [3].

Sous le système Contiki, pour simplifier la transformation d'une application en un ensemble d'événements, un nouveau principe dénommé « protoprocessus » (« protothreads ») a été développé et est également disponible grâce à une bibliothèque [13].

Les « protoprocessus » sont semblables dans leur fonctionnement aux événements à la différence près, qu'à la manière des processus, ils peuvent être bloqués. Cependant, ce blocage est localisé à des emplacements précis symbolisés par la présence de la primitive `PT_WAIT_UNTIL` (`cond1`). Le déblocage se produit quand la condition « `cond1` » est vérifiée. Comme pour les événements, les « protoprocessus » partagent la même pile d'exécution [3].

Contiki ne permet pas l'exécution d'applications en temps réel. Bien qu'il soit possible de charger la bibliothèque permettant d'exécuter des threads en parallèle, le multi-threading est chargé par-dessus l'ordonnanceur événementiel. Une tâche lancée par l'ordonnanceur en mode événementiel ne peut pas être interrompue par les tâches exécutées en mode multi-thread.

Comme nous avons pu le constater tout au long de cette présentation dédiée au système Contiki, celui-ci offre une architecture hybride novatrice mais qui s'appuie surtout sur un système basé sur les événements. On passe d'un mode de fonctionnement à l'autre sans pouvoir réellement les combiner sauf si l'on utilise les « protoprocessus ».

2.3.1.2. Les caractéristiques

Un système d'exploitation pour capteur en réseau a différentes caractéristiques :

- **Empreinte mémoire**

L'espace mémoire utilisé par le système d'exploitation et par l'application doit être suffisamment faible pour être contenu dans la mémoire du capteur. Une configuration typique de Contiki (le noyau et le chargeur de programmes) consomme 2 kilooctets de RAM et 40 kilooctets de ROM [16].

- **Consommation électrique :**

L'énergie électrique, souvent apportée par une batterie de piles, peut être problématique à renouveler. Si des systèmes de captage, comme des éléments photovoltaïques, éoliens, ou autres peuvent être utilisés dans certains cas, les recherches

scientifiques explorent les possibilités de réduire la consommation des capteurs. L'élément le plus consommateur est le module radio [16].

La réduction de temps de transmission et de réception radio est primordiale. Pour cela, le module radio est activé lorsque nécessaire, et arrêté ou mis en veille le reste du temps. Mais lorsque le module radio est arrêté, le capteur ne reçoit pas les messages qui lui sont destinés. Un réveil périodique risque d'être inutile, et donc de consommer de l'énergie de façon inefficace. Pour gérer cette problématique, Contiki propose par défaut ContikiMAC, un mécanisme conçu pour rester en communication avec le réseau efficacement, tout en permettant la mise hors tension du module radio 99 % du temps. D'autres techniques permettent de limiter la consommation telle que le compactage des données à transférer, le pré-calcul (afin de ne transmettre que les données réellement utiles), mais aussi une optimisation du routage. Dans certains cas, il peut être utile de stocker des informations dans une base de données locale au capteur, en effet, si le capteur doit envoyer en ensemble de mesures semblables à des résultats déjà envoyés à un autre moment, il peut être préférable d'envoyer une référence à ces données déjà envoyées (Parcourir 100 enregistrements dans une base coûte moins d'énergie que de transmettre un paquet radio) [16].

- **Communications :**

Contiki implémente deux mécanismes de communication [16]:

- 1. La couche de protocole Rime :**

Rime est une légère couche de communication qui réduit la complexité d'uIP. Elle fournit à la couche applicative un jeu d'instructions de communication, permettant les différentes connexions avec les capteurs voisins. Les protocoles de la pile Rime sont disposés dans un mode en couches, où les protocoles les plus complexes sont implémentés en utilisant les protocoles moins complexes. Toutes les communications en Rimes ont identifiées par une chaîne de 16 bits.

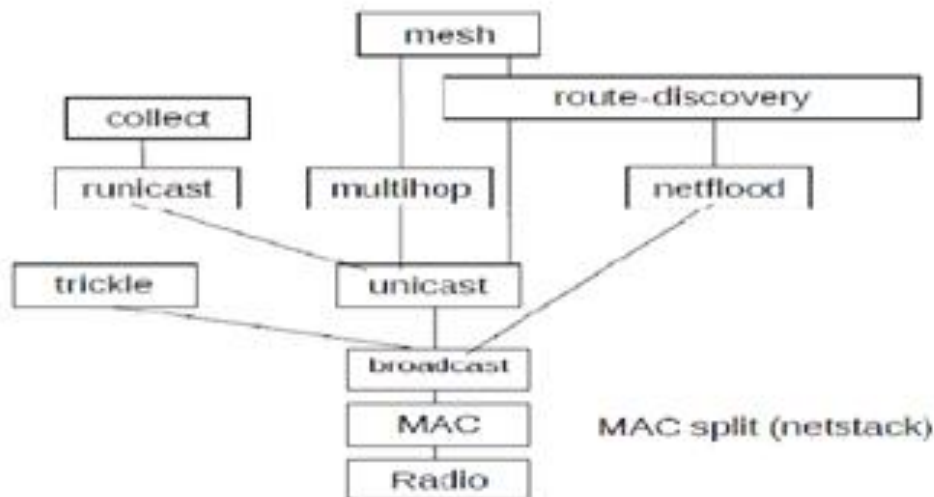


Figure II. 3: Carte générale de la pile rime [17].

2. La couche uIP :

Elle supporte les protocoles IP, TCP, UDP et ICMP. Contiki implémente uIPv4 et uIPv6 et ce dernier est la première implémentation d'IPv6 pour capteurs miniatures. Pour les communications radio via le protocole IEEE 802.15.4, Contiki implémente 6LoWPAN. Lors de communications radio suivant la norme IEEE 802.15.4 communément utilisée par les capteurs, la taille d'un paquet est limitée à 127 octets, ce qui est insuffisant pour transmettre un paquet IPv6 dont la taille maximum est de 1280 octets. L'IETF a créé une couche d'adaptation 6LoWPAN qui se situe entre les couches liaison et réseau du modèle OSI. 6LoWPAN permet la compression de l'entête du paquet IPv6 ainsi que la fragmentation et le réassemblage des datagrammes. Pour le routage d'IPv6 à travers le réseau de capteur, Contiki intègre ContikiRPL dans la couche uIPv6, une implémentation du protocole de routage RPL.

- **Portabilité :**

La portabilité consiste à adapter le système d'exploitation aux capteurs, selon les éléments électroniques les constituant. Contiki est complètement écrit en langage C, ce langage de programmation est le langage le plus répandu pour la programmation des systèmes. Le portage de Contiki est effectué pour les plateformes suivantes: exp5438, z1, wismote, avr-raven, avr-rcb, avr-zigbit, iris, micaz, redbee-dev, redbee-econotag, mb851, mbxxx, sky, jcreate, sentilla-usb, msb430, esb, avr-atmega128rfa, cc2530dk, sensinode ainsi que sur apple2enh, atari, c128, c64 [16].

- **Interopérabilité :**

L'interopérabilité d'un capteur est le fait de pouvoir communiquer avec des capteurs gérés par un système d'exploitation différent. Adams Dunkels de l'équipe scientifique suédoise présente dès 2003 uIP et lwIP permettant d'implémenter le protocole IP sur les systèmes limités en ressources tel que les capteurs. Jusque là, les capteurs utilisaient des protocoles de communication propriétaires ou alors des adaptations d'IP permettant le fonctionnement des applications mais sans offrir toutes les fonctionnalités du protocole IP. Dès la présentation de Contiki en 2004, uIP et lwIP étaient disponibles. De ce fait, les applications exécutées sur Contiki pouvaient dialoguer vers n'importe quel matériel supportant le protocole IP. L'arrivée d'IPv6 et uIPv6 sur Contiki apporte une nouvelle interopérabilité vers les matériels supportant ce protocole. Le support de 6LoWPAN permet à Contiki de communiquer avec les matériels via un réseau sans-fil suivant la norme 802.15.4. Contiki est réputé pour être un système d'exploitation robuste et mature, fournissant IPv4 et IPv6 pour les réseaux de capteurs sans fil. Selon une étude publiée en 2011, comprenant des tests d'interopérabilité entre des capteurs sous Contiki et d'autres sous TinyOS, l'interopérabilité est bien au rendez-vous, mais des efforts sont à faire pour mesurer et améliorer les performances des couches réseaux [16].

2.3.1.3. Les avantages et inconvénients du système Contiki

L'architecture hybride du noyau Contiki autorise deux modes de fonctionnement soit multitâche, soit basé sur les événements. A ce titre, elle permet à ce système d'offrir plusieurs solutions pour répondre au, plus près, aux contraintes de l'application supportée, un mode pouvant être plus performant que l'autre. Cela constitue le principal avantage de ce système d'exploitation [3].

Cependant, le noyau Contiki reste, nativement, un système d'exploitation basé sur les événements. Pour obtenir le mode multitâche, une bibliothèque doit être installée. Les fonctions associées à cette bibliothèque n'accèdent pas directement à l'ensemble des ressources du capteur sans fil. Elles doivent, dans certains cas, faire appel à la partie du noyau dédié à la gestion des événements. Cette structure à deux niveaux a pour conséquence une dégradation des performances du système quand le mode multitâche est activé [3].

2.3.1.4. Un simulateur réseau pour Contiki : Cooja

COOJA est l'acronyme de Contiki OS Java Simulator.

Pour développer les programmes au sein de Contiki, le système met à disposition un simulateur réseau appelé Cooja. Le logiciel permet d'émuler des nœuds et de charger un programme compilé. Ceci est particulièrement utile pour tester les programmes avant de les mettre dans la mémoire flash des nœuds réels, puisque le logiciel simule les conditions d'exécution et de mémoire de la plateforme TI MSP430. Les données collectées provenant du sink via sa sortie standard peuvent être enregistrées dans des fichiers ou lues par des logiciels qui peuvent par la suite traiter et présenter les données à l'utilisateur. On peut par exemple citer le logiciel collect-view intégré dans Contiki qui permet de visualiser les valeurs des capteurs et des données de supervision du réseau. En revanche, Cooja a un intérêt limité si l'on veut tester des algorithmes de géolocalisation basés sur le RSSI. En effet, le logiciel utilise un modèle linéaire pour simuler ces valeurs en fonction de la distance, alors qu'en réalité, le modèle est logarithmique. De plus, l'environnement simulé ne reflète pas la réalité en raison de la non prise en compte des perturbations engendrées par les murs, sols...

Figure II. 4: une image sur Cooja.

- **Fenêtre de simulation**

Dans une simulation nous avons plusieurs fenêtres :

- A. La fenêtre Timeline : en bas de l'écran, nous affiche tous les événements de communication dans la simulation dans le temps, très pratique pour comprendre ce qui se passe dans le réseau.
- B. La fenêtre Network : en haut à gauche de l'écran, nous montre tous les nœuds dans le réseau simulé.
- C. La fenêtre Timeline : en bas de l'écran, nous affiche tous les événements de communication dans la simulation dans le temps, très pratique pour comprendre ce qui se passe dans le réseau.
- D. La fenêtre Mote Output, sur le côté droit de l'écran, nous montre toutes les impressions port série de tous les nœuds.
- E. La fenêtre Notes en haut à droite est l'endroit où nous pouvons mettre des notes pour notre simulation.
- F. La fenêtre Simulation control : est où nous pouvons lancer, mettre en pause et charger de notre simulation.

3. Comparaison

Le document « Power Consumption Analysis of Operating Systems for Wireless Sensor Networks » de 2010 détaille que TinyOS et Contiki, en rajoutant Mantis sont les plus importants et les plus utilisés dans le domaine des WSN, et au vu du nombre de publications scientifiques concernant ces système d'exploitation dans les bases IEEE Xplore, ACM Digital Library et Science Direct. Les pourcentages sont de 81 % pour TinyOS, 9 % pour Contiki, 8 % pour Mantis et de 2 % pour les autres [16].

	TinyOS	Contiki	Mantis
Statique/Dynamique	Statique	Dynamique	Dynamique
Événementiel/Thread	Event&Thread (TinyThread, TOSThreads)	Event&Threads (Protothreads)	Thread&Event (TinyMOS)
Monolithique/Modulaire	Monolithique	Modulaire	Modulaire
Réseau	Message Actif	uIP, uIPv6, Rime	"comm"
Langage de programmation	nesC	C	C
Système de fichier	Un seul niveau	Coffee	Non
Reconfiguration	Oui	Oui	Non

Tableau II. 1 : Caractéristiques des systèmes d'exploitation pour WSN[16].

Une expérimentation réalisée par un centre de recherche Brésilien, compare TinyOS et Contiki lorsqu'ils sont implémentés sur un capteur de type CrossbowTelosB : Il en ressort que Contiki et avec son système dynamique et modulaire est contrairement à TinyOS basé sur un système statique et monolithique, plus flexible et donc plus adapté pour un changement d'environnement dynamique ou dans le cas d'une reprogrammation au travers du réseau [16].

Le système d'exploitation de Contiki, contrairement à TinyOS écrit en NesC est codé en langage C ce qui le rend très portable. L'article de Laraja[17] précise que la programmation en C implique une courbe d'apprentissage beaucoup moins raide que celle nécessaire en NesC pour lequel les développeurs doivent s'approprier un nouveau paradigme dans les concepts tel que les modules, les composants, les interfaces ou configurations. La programmation complexe en NesC permet d'obtenir un code léger. Ce point est également souligné par Philip Levis dans son article [19] retraçant le développement de TinyOS sur la dernière décennie. Il y précise qu'aujourd'hui, même si

TinyOS est plus léger et efficace, la majorité des recherches autour des réseaux de capteur sans fil se fait avec Contiki qui est plus facile d'apprentissage [16].

L'implémentation du mécanisme de multithreads est également différente puisque TinyOS la gère uniquement grâce à sa librairie TinyThreads, et contiki obtient ce mode d'exécution soit par librairie, soit par protothreads, qui réduit l'utilisation de la mémoire [3].

Les tâches de communication fonctionnent mieux sur TinyOS, probablement en raison d'une utilisation plus efficace de la pile de communication qui est la plus légère car elle est basée sur le principe de messages pondérés, Celle de Contiki contient 2 couches de communication, Rime et uIP qui lui permettent de communiquer avec les protocoles de l'internet, y compris en IPv6. Contiki implémente uIPv6, la plus petite couche IPv6 au monde, utilisable dans le domaine des capteurs sans fils [18].

L'empreinte mémoire du noyau de Contiki est aussi plus importante que celle de TinyOS qui ne propose qu'un ordonnancement FIFO, contrairement à Contiki qui permet en plus la gestion de priorités

Les tâches sont exécutées plus rapidement avec Contiki, mais TinyOs est moins consommateur. Pour l'exécution d'algorithme de sécurité, les résultats des 2 OS sont similaires. Les résultats ont montré que les deux OS peuvent être optimisés pour réduire la consommation d'énergie lorsqu'ils sont paramétrés en conséquence par les

développeurs. TinyOS et Contiki ont été validés sur de multiples plates-formes matérielles. Mais, l'article précise que généralement TinyOS peut fonctionner avec des conditions de ressource inférieure liées au fait que le noyau Contiki est plus complexe. (Voir la figure)

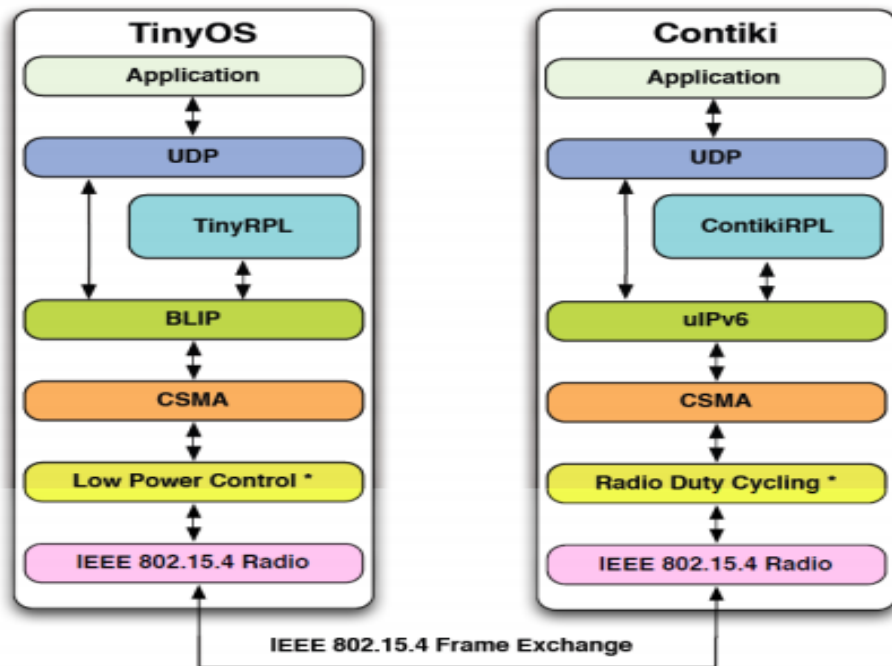


Figure II. 5: Contiki VS TinyOS [20].

4. Conclusion

Dans ce chapitre, nous avons présenté les différents systèmes d'exploitation pour les réseaux de capteurs sans fil en se basant sur le système d'exploitation Contiki, ses propriétés, caractéristiques, son architecture ainsi que le simulateur Cooja.

Dans le chapitre suivant, nous montrons comment nous exploitons ces outils pour développer une application de surveillance.

Chapitre3

Simulation et Réalisation d'une Application des Données Agrégées

1. Introduction

Les capteurs sont généralement déployés dans des zones hostiles là où l'accès humain est difficile voire impossible.

Puisque les batteries sont généralement irremplaçables et non rechargeables la durée de vie de ces capteurs est limitée, de ce fait on introduit une approche intéressante qui sert à maximiser la durée de vie du réseau « l'agrégation de donnée ».

Dans ce chapitre, nous détaillons le fonctionnement de l'application que nous avons développée et les éléments matériels et logiciels utilisés dans la réalisation. Finalement, nous présentons les différentes parties du code en détail.

2. Environnement de travail et outils de développement

- **Outils matériels**

De point de vue matériel, nous avons utilisé pour le développement de notre application des capteurs de type TloseB (voir la figure 2.1), des piles de type pour alimenter les capteurs, un PC portable de type TOSHIBA. La figure 2.2 montre l'architecture de notre application.



Figure III. 1: Un capteur de type TloseB.



Figure III. 2: l'architecture du système.

- **Outils logiciels**

Dans cette partie, nous présentons les outils logiciels nécessaires utilisés dans le développement de l'application. Pour cela, nous avons utilisé Contiki 2.7 qui met à disposition un simulateur réseau appelé Cooja qui est un émulateur qui permet d'exécuter des programmes sur contiki sans avoir besoin de matériels, et le langage de programmation Java.

La programmation des parties d'application dans ce système d'exploitation se fait dans le langage C.

Pour développer une application pour les réseaux de capteurs sans fil, il est nécessaire d'avoir un environnement dédié : un système d'exploitation léger tels que TinyOS ou ContikiOS et non pas des systèmes d'exploitation traditionnels comme Windows ou Linux car ces derniers nécessitent un grand espace mémoire pour leur mise en place. Pour cela on a utilisé le système d'exploitation Contiki. La programmation d'application dans ce système d'exploitation se fait dans le langage C qui est un langage de programmation impératif, généraliste, conçu pour la programmation système. D'autre part pour réaliser l'interface graphique nous avons choisi Java qui est un langage de programmation informatique orienté objet et permet la création des interfaces graphiques grâce à sa librairie Swing, et aussi parce que c'est le seul langage dont le code est portable.

3. Les étapes de développement de l'application

- **Installation logicielle**

L'installation du système d'exploitation Contiki a été la phase la plus simple. Pour éviter d'installer tout l'environnement de développement, nous avons utilisé une machine virtuelle nommée instant Contiki2.7 et pour faire tourner cette machine nous avons utilisé un lecteur des machines virtuelles « VMplayer ». Nous avons également installé Netbeans 8.0.2 pour développer l'interface Java.

- **Installation matérielle**

Une fois l'installation logicielle est terminée, il faut installer le matériel: une station de base reliée à l'ordinateur, 3 capteurs TloseB (Sender) et un cluster-Head. Les capteurs communiquent entre eux via des liaisons sans fil.

4. Les parties du code

Dans cette section nous détaillons l'implémentation et les différentes parties de notre application. Notre application est composée de trois parties : une partie embarquée sur la station de base, une partie embarquée sur un Cluster-Head et une autre partie embarquée sur des nœuds capteurs (Senders).

Nous nous sommes basées sur l'exemple unicast qui existe au sein de la plateforme Contiki et qui utilise la pile rime pour que les nœuds puissent communiquer entre eux.

4.1. Compréhension du code

- Ces cinq premières lignes incluent les en-têtes (header) C du Contiki. Ces lignes sont nécessaires pour inclure les bibliothèques et les définitions du Contiki.

```
#include "contiki.h"
#include "net/rime.h"
#include "dev/button-sensor.h"
#include "dev/leds.h"
#include <stdio.h>
```

- Cette première ligne déclare le processus Contiki. Ce processus possède une variable nommée (example_unicast_process) et une chaîne de caractère nommée

(Exemple unicast). La chaîne de caractères est utilisée pour le débogage et comme une commande pour le Shell de Contiki.

```
PROCESS (example_unicast_process, "Example unicast");
```

- Cette ligne informe le système Contiki que le programme `example_unicast_process` doit se lancer automatiquement lors du démarrage du système.

```
AUTOSTART_PROCESSES (&example_unicast_process);
```

- Cette fonction analyse un paquet entrant et affiche le message et l'adresse de l'expéditeur. `recv_uc` est automatiquement appelée quand un paquet est reçu.

```
static void recv_uc (struct unicast_conn *c, const rimeaddr_t *from)
{
    printf("unicast message received from %d.%d\n",
           from->u8[0], from->u8[1]);
}
```

- Et la fonction `unicast_callbacks` soulignent `recv_uc` et fonctions envoyés (dans cet exemple, juste `recv_uc`).

```
static const struct unicast_callbacks unicast_callbacks = {recv_uc};
```

- Ceci est identique à la variable de `recv_uc` ().

```
static struct unicast_conn uc;
```

- Cette ligne définit le processus `example_unicast_process`. Les arguments `ev` et `data` dans la macro « `PROCESS_THREAD ()` » sont des variables du nombre d'évènement et de données respectivement que le processus peut recevoir.

```
PROCESS_THREAD(example_unicast_process, ev, data)
```

- Indique une action lorsqu'un processus se termine.

```
PROCESS_EXITHANDLER (unicast_close(&uc);
```

- Cette macro définit le début d'un processus

```
PROCESS_BEGIN();
```

- &uc: Un pointeur vers une struct unicast_conn
- 146: Le canal sur lequel la connexion fonctionnera
- Unicast_callbacks: une structure unicast_callbacks avec des pointeurs de fonction vers des fonctions qui seront appelées quand un paquet a été reçu

```
unicast_open (&uc, 146, &unicast_callbacks);
```

- La boucle while est utilisée pour créer une boucle infinie dans laquelle la réponse de gestion des événements proprement dite a lieu

```
While (1) {
```

- etimer est un pointeur de la minuterie de l'événement

```
static struct etimer et;
```

```
rimeaddr_t addr;
```

- Cette fonction est utilisé pour lorsque la minuterie de l'événement expire, le PROCESS_EVENT_TIMER d'événement sera affiché sur le processus qui a appelé la fonction de etimer_set ().

```
etimer_set (&et, CLOCK_SECOND);
```

```
PROCESS_WAIT_EVENT_UNTIL (etimer_expired(&et);
```

- Copiez les données externes dans le packetbuf.

```
packetbuf_copyfrom("Hello", 5);
```

```
addr.u8[0] = 1;
```

```
addr.u8[1] = 0;
```

- Comparer l'adresse Rime 1.0 avec l'adresse du nœud actuel.

```
if(!rimeaddr_cmp(&addr, &rimeaddr_node_addr)) {
```

- Envoyer le paquet en unicast en utilisant un pointeur vers la connexion et vers l'adresse de destinataire

```
unicast_send(&uc, &addr);
```

```
    printf(" message sent\n");  
  }  
}  
PROCESS_END();  
}
```

4.2. La partie Sender

C'est la partie de l'application qui est embarquée sur des nœuds capteurs et qui contient les différentes fonctionnalités permettant la capture de paramètre de l'environnement (température) et leur envoie au nœud cluster-Head.

- Pour calculer la température, nous avons utilisé la formule suivante :

```
(((sht11_sensor.value(SHT11_SENSOR_TEMP) / 10) - 396) / 10));
```

4.3. La partie cluster-head

Cette partie de l'application est embarquée sur un nœud appelé cluster-head qui permet de capter les informations envoyées depuis les senders, après il applique une fonction d'agrégat (dans notre cas c'est la moyenne)

4.4. La partie station de base

C'est une autre partie de l'application embarquée sur un nœud capteur relié à un PC. Elle reçoit la moyenne des températures envoyée de la part du cluster -head.

5. La simulation sous Cooja

Pour la simulation de réseau de capteurs sans fil il faut utiliser l'outil de simulation de Contiki Cooja.

Dans cette partie il faut fixer l'architecture matérielle :

- Organisation du réseau.
- Nombre de nœuds.
- Répartitions géographique.

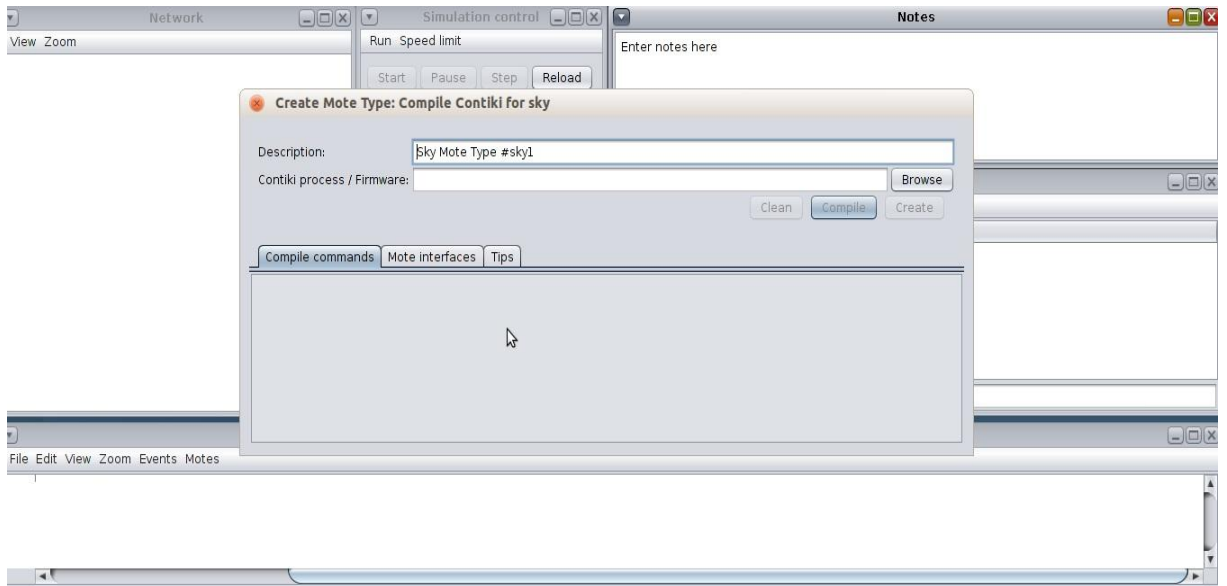


Figure III. 3: choix de type de nœuds avant de faire la compilation de code.

Dans ce projet nous avons cinq capteurs.



Figure III. 4: visionnaire de capteur sur cooja.

La répartition géographique de nœuds est bien paramétrée sur Cooja puisque il donne une interface pour organiser les nœuds sur le plan réel à trois dimensions X, Y, Z.

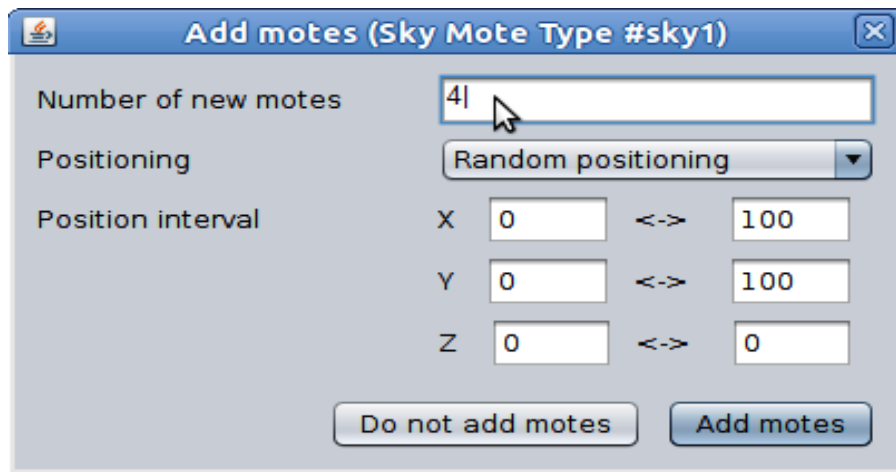


Figure III. 5: introduire le nombre de nœuds nécessaires pour la simulation.



Figure III. 6: les positions des nœuds sur le plan géographique.

Enfin nous avons un réseau simulé sur l'outil Cooja de Contiki il nous permet de voir tous les paramètres d'un capteur ainsi les communications entre les nœuds, la figure suivante montre l'interface graphique de notre réseau simulé.

❖ Démarrer la simulation :

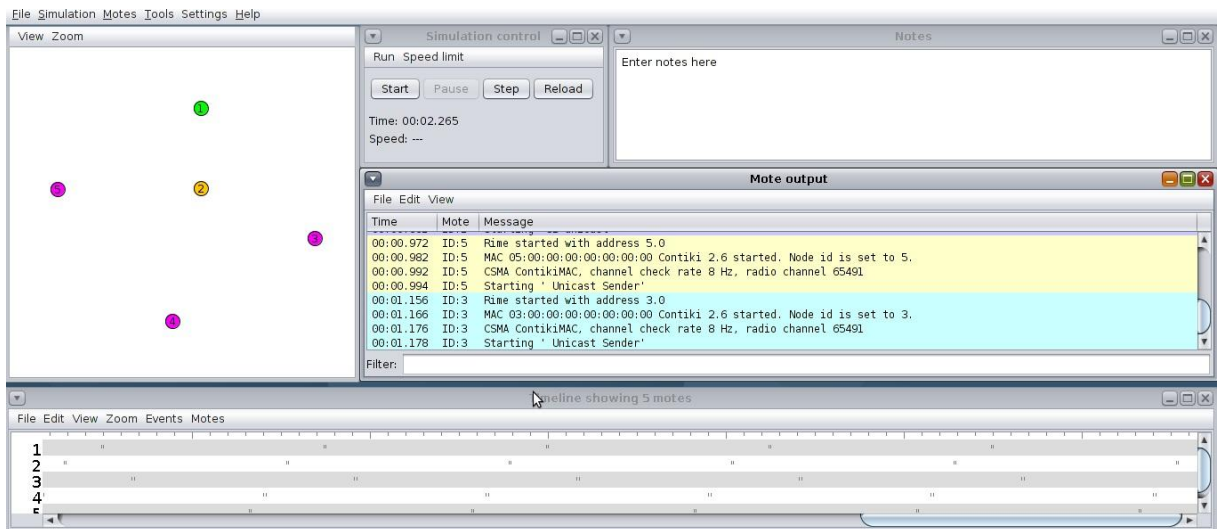


Figure III. 7: interface graphique sur le simulateur cooja.

- La première ligne : nous montre l'adresse Rime du nœud.
- La deuxième ligne : nous donne l'adresse MAC du nœud ainsi que son identifiant
- La troisième ligne : Nous montre le mécanisme par défaut ContikiMac , La couche MAC est responsable pour éviter les collisions au moyen de la radio et de retransmettre les paquets si il y avait une collision. Contiki fournit deux couches MAC: un mécanisme CSMA (Carrier Sense Multiple Access) et un mécanisme NullMAC mais dans cette simulation nous avons le mécanisme CSMA et un taux de contrôle de canal de 8 Hz, ainsi qu'un canal radio 65491.
- La quatrième ligne : désigne le nom du processus qui a démarré.

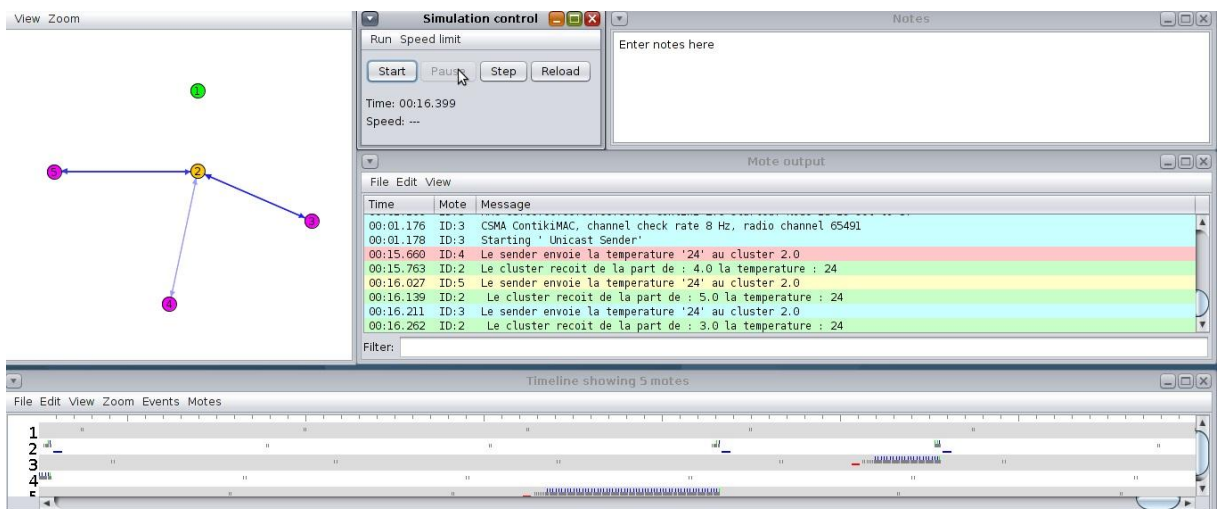


Figure III. 8: Communication entre les senders et cluster-head.

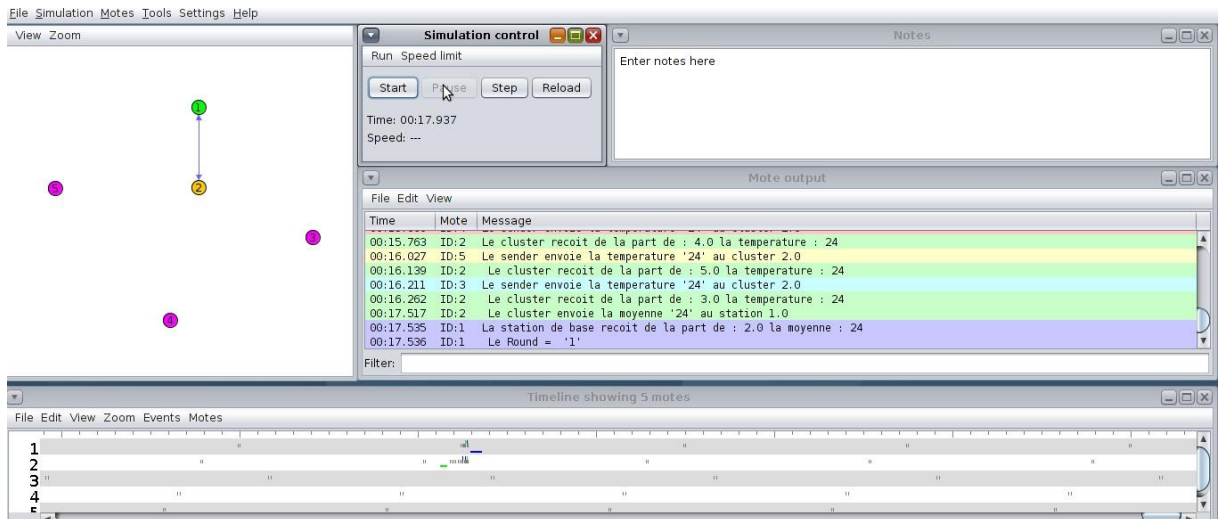


Figure III. 9: Communication entre le cluster-head et la station de base.

6. Réalisation de l'application sur une plateforme matériel Tmote Sky

Nous avons présenté dans la section suivante la simulation de notre application sous le simulateur Cooja. Néanmoins, on peut affirmer que la simulation n'est pas une solution « clé en main », car il y a une grande différence entre la simulation et la réalisation sur des capteurs réels. En environnement réel, les capteurs peuvent tomber en panne, être soumis à des collisions ou même des craches dus aux caractéristiques intrinsèques du médium radio.

Pour ces raisons, nous avons estimé qu'il serait encore plus intéressant, si on exécute l'application sur une plateforme matérielle réelle.

Dans ce qui suit, nous présentons les étapes suivies :

- Insérer le capteur Tmote Sky (TelosB) sur le port USB du PC.
- Il faut configurer le VMware Player ou le VirtualBox pour accéder au capteur. Cette procédure peut être effectuée dans le menu de la machine virtuelle. Dans le cas de VMware Player, il faut aller dans (Virtual Machine -> Removable Devices-> future devices mote sky).
- Une fois cette procédure est terminée, le capteur Tmote Sky devient accessible par Instant Cintiki. Vous pouvez tester cette partie par la commande : `make sky-motelist`

- Une fois que le capteur est visible via Instant Contiki, allez dans le dossier où se trouve votre programme et chargez le programme sur le capteur avec la commande suivante :

```
make nom-programme.upload TARGET=sky
```

Une fois le code compilé, vous pouvez vous connecter sur le nœud avec la commande suivante :

```
make login TARGET=sky
```

- qui affiche :

```
../../tools/sky/serialedump-linux -b115200 /dev/ttyUSB0
```

```
connecting to /dev/ttyUSB0 (115200) [OK]
```

- cette ligne nous montre que le serialdump-linux qui se trouve dans le dossier tools/sky permet de récupérer les données sur l'ordinateur dont le débit est 115200, et que le dispositif est relié au fichier /dev/ttyUSB0 .
- Le capteur connecté à l'ordinateur par l'intermédiaire du port USB envoie des données à l'ordinateur avec la commande printf ().
- Dans notre cas nous avons trois résultats :
 - Le Sender
 - Le cluster-head
 - La station de base

Pour chacun des résultats, nous présentons dans ce qui suit les captures d'écran de l'exécution de notre application.

❖ Le sender :

```
user@instant-contiki: ~/Aggregation-Project/TestBedS
File Edit View Search Terminal Help
user@instant-contiki:~$ cd Aggregation-Project/
user@instant-contiki:~/Aggregation-Project$ cd TestBedS
user@instant-contiki:~/Aggregation-Project/TestBedS$ sudo chmod 666 /dev/ttyUSB0
[sudo] password for user:
user@instant-contiki:~/Aggregation-Project/TestBedS$ make login TARGET=sky
/home/user/contiki/tools/sky/serialedump-linux -b115200 /dev/ttyUSB0
connecting to /dev/ttyUSB0 (115200) [OK]
Le sender envoie la temperature '31' au cluster 194.108
Le sender envoie la temperature '32' au cluster 194.108
Le sender envoie la temperature '34' au cluster 194.108
Le sender envoie la temperature '33' au cluster 194.108
Le sender envoie la temperature '32' au cluster 194.108
Le sender envoie la temperature '32' au cluster 194.108
Le sender envoie la temperature '33' au cluster 194.108
Le sender envoie la temperature '33' au cluster 194.108
```

Figure III. 10: Les valeurs des températures capturées par les Senders.

❖ Le Cluster-Head :

```
user@instant-contiki: ~/Aggregation-Project/TestBedC
File Edit View Search Terminal Help
user@instant-contiki:~/Aggregation-Project/TestBedC$ make login TARGET=sky
/home/user/contiki/tools/sky/serialedump-linux -b115200 /dev/ttyUSB0
connecting to /dev/ttyUSB0 (115200) [OK]
Rime started with address 194.108
MAC c2:6c:00:00:00:00:00:00 Contiki 2.6 started. Node id is not set.
CSMA ContikiMAC, channel check rate 8 Hz, radio channel 65491
Starting 'unicast_receiver'
Le cluster recoit de la part de : 255.112 la temperature : 29
Le nombre de capteurs reçus = 1
Le cluster recoit de la part de : 95.43 la temperature : 30
Le nombre de capteurs reçus = 2
Le cluster envoie la moyenne '29' au station 77.124
```

Figure III. 11: la réception des températures envoyées par les Senders et l'envoi de la moyenne à la station de base.

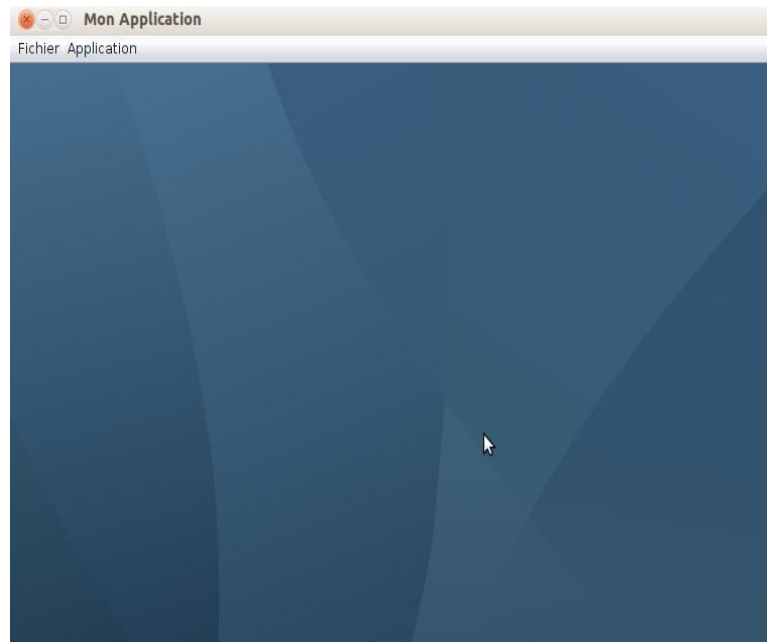


Figure III. 13: L'interface JAVA qui représente notre application.

La moyenne envoyée par le Cluster-Head et le round sont affichés sur l'interface de l'application.

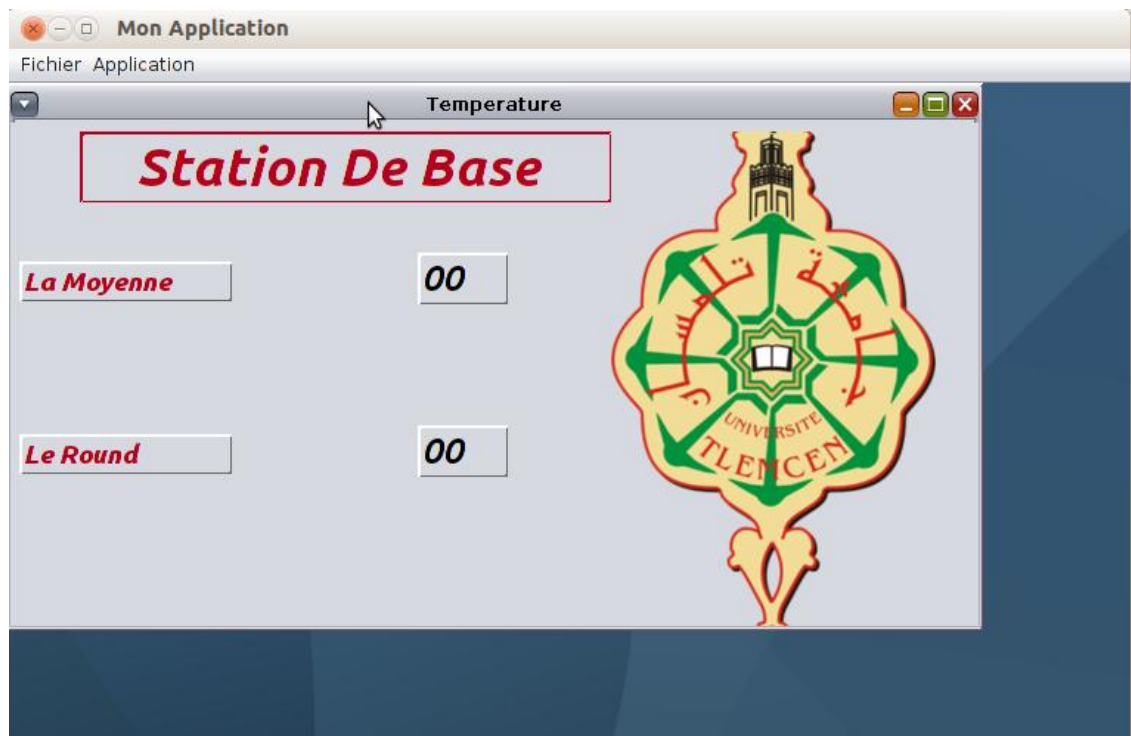


Figure III. 14: L'interface de la station de base avant la réception de la moyenne.

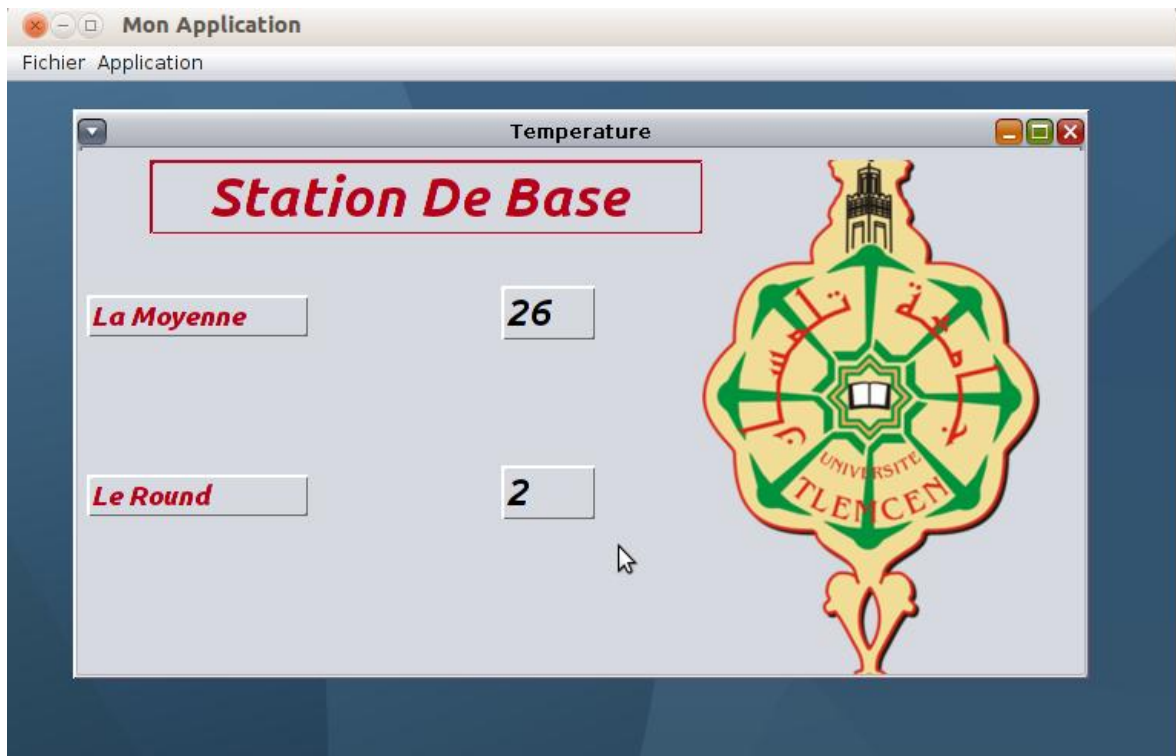


Figure 3.15 : L'interface de la station de base après la réception de la moyenne.

8. Conclusion

Dans ce chapitre nous avons montré la démarche de notre application qui permet de faire l'agrégation de donnée au sein d'un réseau de capteurs sans fil afin de prolonger la durée de vie de ce dernier.

Le développement de cette application nécessite des outils logiciels bien particuliers tels qu'un système d'exploitation simple « Contiki » qui est programmé en C et le simulateur Cooja pour faire la simulation.

Conclusion générale

Conclusion générale

Les réseaux de capteurs constituent un axe de recherche très fertile ainsi qu'un outil qui peut être appliqué dans plusieurs domaines différents.

Cependant, il reste encore de nombreux problèmes à résoudre dans ce domaine pour un fonctionnement efficace de ces réseaux et afin de pouvoir les utiliser dans les conditions réelles. On ne peut pas aborder toutes ces problématiques en même temps, ce mémoire est centré sur celle de la consommation d'énergie dans un capteur sans fil qui est une nécessité absolue à laquelle des solutions adéquates doivent être proposées et qui est causée par les données redondantes et du fait que les RCSF sont déployés dans des environnements complexes et distribués et que leur batteries à faible puissance ne peuvent pas être rechargées, sachant que la communication entre les nœuds constitue l'opération la plus consommatrice d'énergie, donc on aura un gaspillage d'énergie qui est une ressource critique dans les réseaux de capteurs sans fil, de ce fait, toute limitation de la communication entre les nœuds sera sûrement bénéfique.

Puisque le problème d'énergie joue un rôle important pour prolonger la durée de vie du réseau, une solution qui remédie à ce problème c'est l'agrégation de donnée qui sert à donner une information sur un phénomène physique qui a lieu dans un champ de captage, les valeurs individuelles récupérées par chaque nœud de manière individuelle, n'ont pas de grande utilité. Dans l'agrégation de données, un nœud capteur nommé Cluster-head est désigné pour récupérer les valeurs des autres nœuds dans son cluster, et calculer une fonction d'agrégat qui remplace les valeurs individuelles en une valeur globale. De ce fait, au lieu d'envoyer n messages, le cluster-head va envoyer un seul message à la station de base. Cette technique est très utile et permet de minimiser l'énergie des capteurs.

Ce travail nous a permis de se familiariser avec les réseaux de capteurs sans fil, il nous a aussi fait découvrir une nouvelle plateforme de programmation adéquate qui est Contiki. En outre, dans ce projet on a rencontré certaines difficultés dans la réalisation réelle sur des capteurs. Ce qui nous laisse dire que la simulation ou l'émulation d'une application n'est pas une solution « clé en main », car dans la réalité, les capteurs ont beaucoup de contraintes à savoir les collisions, les pertes de messages ainsi que les craches.

En guise de perspective, il serait intéressant d'évaluer une application d'agrégation de données en termes d'énergie par rapport à une approche traditionnelle afin de bien cerner le bénéfice. Une autre perspective serait la sécurisation de l'agrégation des données, puisque la compromission du cluster-head peut mettre en péril la fonction d'agrégat, et pour des applications critiques, cela peut-être néfaste.

Références bibliographiques

Références bibliographiques

- [1] S. Maarouf, S. Ouadah, «Implémentation et évaluation des schémas de routage sur une plateforme réelle de réseau de capteur sans fil», Mémoire de Master, Département d'Informatique, Université de Tlemcen, Juin 2014.
- [2] F. Khedim , « Détection des attaques par réplication dans un réseau de capteurs sans fil », Mémoire de Master, Département d'Informatique, Université de Tlemcen, Juin 2013.
- [3] G. De Sousa, « Etude en vue de la réalisation de logiciels bas niveau dédiés aux réseaux de capteurs sans fil : microsystème de fichiers», Thèse de Doctorat, Département d'Informatique, Université Blaise Pascal-Clermont II, Octobre 2008.
- [4] I. Akyildiz, W. Su, E. Cayirci, Y. Sankarasubramaniam. "A survey on sensor Networks", *IEEE Communications Magazine*, vol. 40, no. 8, pp. 102-114, Georgia Institute of Technology, Atlanta, USA. Août 2002.
- [5] N. Labraoui, « La sécurité dans les réseaux sans fil ad hoc», Thèse de Doctorat, Département d'Informatique, Université de Tlemcen, 2012.
- [6] M. Abdallah, « Réseaux de capteurs : localisation, couverture et fusion de données», Thèse de Doctorat, Université de Franche-Comté, Novembre 2008.
- [7] Y. Younes, « Minimisation d'énergie dans un réseau de capteurs», Mémoire de Master, Département d'Informatique, Université Mouloud Mammeri de Tizi-Ouzou, Septembre 2012.
- [8] Boucif Amar Bensaber, « Introduction à la sécurité des réseaux de capteurs sans fil», Département de Mathématique Informatique, Université de Québec à Trois Rivières, Juin 2013.
- [9] X. Yong, « Agrégation de données dans les réseaux de capteurs sans fil», Projet SR04, Université de Technologie Compiègne, Automne 2010.
- [10] C. Bonnet et I. Demeure « *Introduction aux systèmes temps réel* » Hermes Science Publications, 1999.
- [11] I. Stojmenovic « *Handbook of Sensor Networks: Algorithms and Architectures*» Wiley Series on Parallel and Distributed computing, November 2005.

- [12] P. Levis, S. Madden, D. Gay, J. Polastre, R. Szewczyk, A. Woo, E. Brewer, D. Culler « *The Emergence of Networking Abstractions and Techniques in TinyOS* » 1st USENIX/ACM Symposium on Networked Systems Design and Implementation, (NSDI 2004), San Francisco, California, USA, March 29-31, 2004.
- [13] A. Dunkels, O. Schmidt, T. Voigt, M. Ali « *Protothreads: Simplifying Event-Driven Programming of Memory-Constrained Embedded Systems* » 4th International Conference on Embedded Networked Sensor Systems (SenSys'06), Boulder, Colorado, USA, November 2006.
- [14] J. Hill, R. Szewczyk, A. Woo, S. Hollar, D. Culler, K. Pister « *System Architecture Directions for Networked Sensors* » 9th International Conference on Architectural Support for Programming Languages and Operating Systems, Cambridge, Massachusetts, USA, November 2000.
- [15] A. Dunkels, B. Grönwall, T. Voigt « *Contiki – a Lightweight and Flexible Operating System for Tiny Networked Sensors* » 1st IEEE Workshop on Embedded Networked Sensors (IEEE EmNetS-I), Tampa, Florida, USA, November 2004.
- [16] A. Badaoui, « Acquisition de données à distance dans les réseaux de capteurs sans fil », Mémoire de master, Département d'Informatique, Université de Tlemcen, juin 2013.
- [17] R. Lajara, J. Peligri-Sebastia, J. P. Solano, « Power Consumption Analysis of Operating Systems for Wireless Sensor Networks », SENSORS, 2010, p. 5809-5826.
- [18] S. Hallab, A. Jraidi, « Développement s'un système de surveillance de l'environnement à base d'un réseau de capteurs sans fil », Mémoire de licence, Université de Monastir, Institut Supérieur d'Informatique de Mahdia, Juin 2013.
- [19] P. LEVIS, « *Demo: Experiences from a decade of TinyOS development integrated development environment* », OSDI'12 Proceedings of the 10th USENIX conference on operating Systems Design and Implementation, 2012
- [20] A. Rachedi, « Réseaux sans infrastructure », Université Paris-Est Marne-la-Vallée (UPEM), 2014/2015.

Annexe

Annexe : installation Contiki

Contiki est un système d'exploitation pour les réseaux de capteurs sans fil. Cooja est un émulateur réseau basé sur Contiki qui permet d'exécuter des programmes sur Contiki sans avoir besoin du matériel.

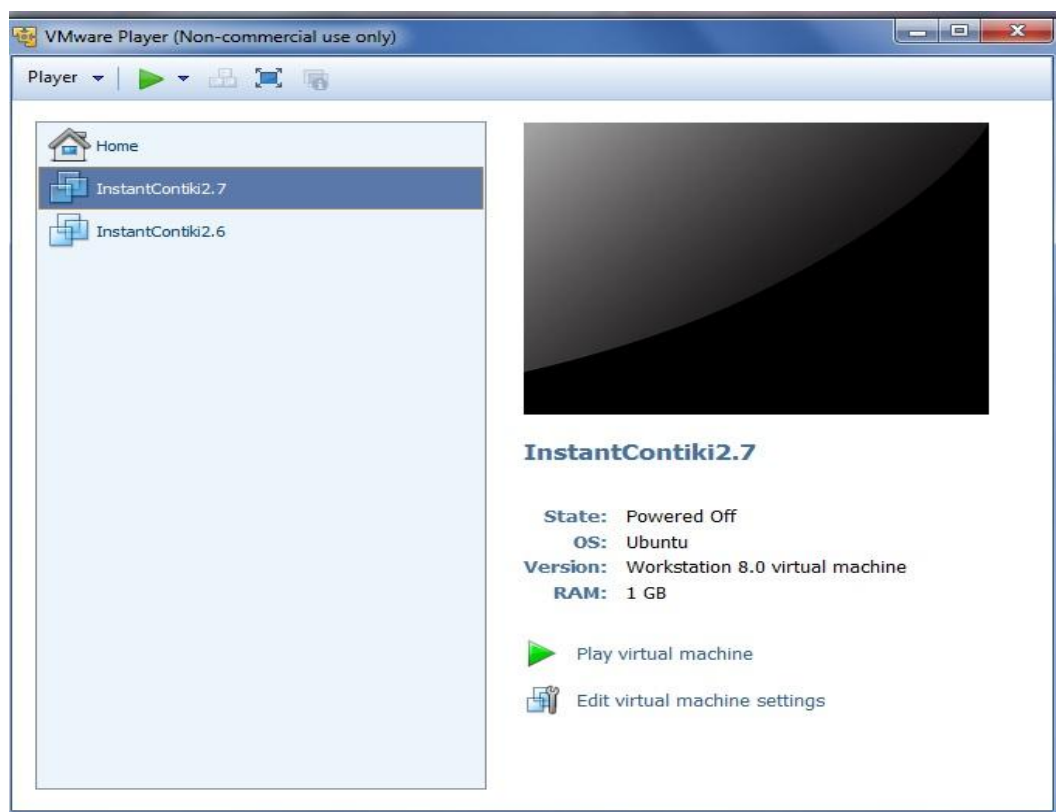
Pour éviter d'installer tout l'environnement de développement, nous allons utiliser une machine virtuelle (VM) nommée « Instant Contiki ».

Etapes à suivre pour mettre en place la VM :

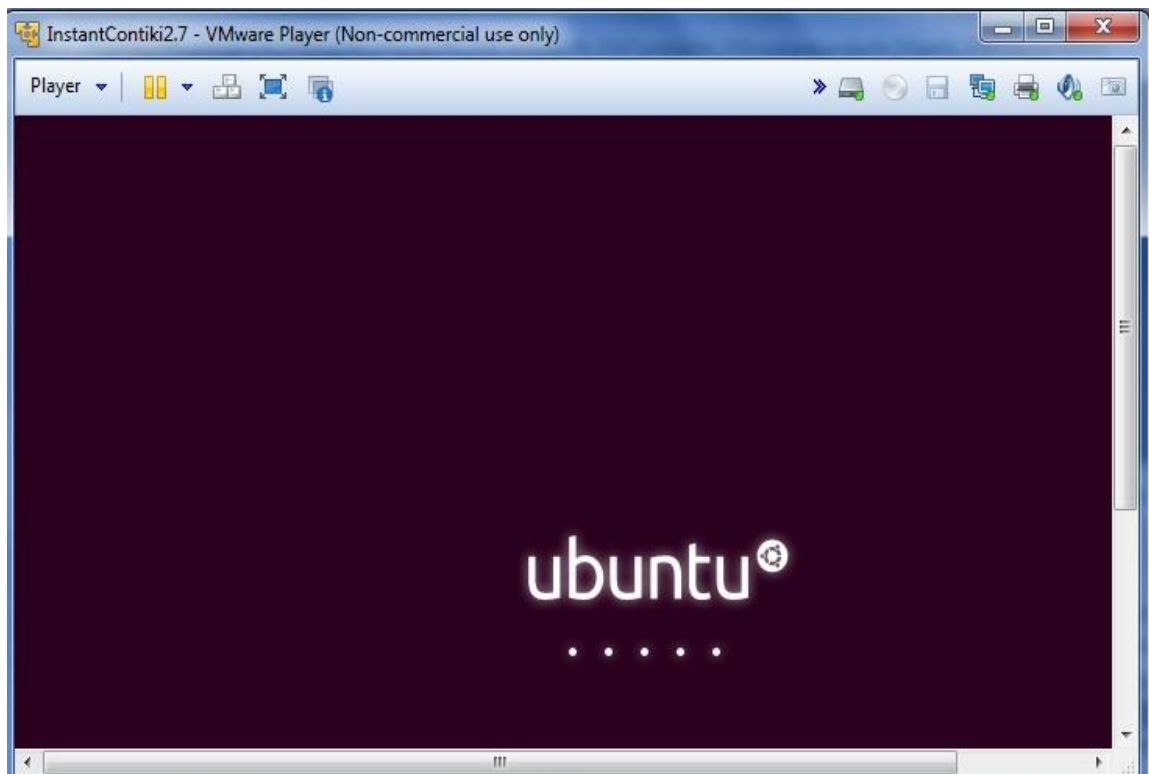
1- Télécharger la machine virtuelle nommée : « Instant Contiki 2.7 ». Le site : <http://sourceforge.net/projects/contiki/files/Instant%20Contiki/> ou <http://www.contiki-os.org/>

Il s'agit d'un fichier de grande taille, un peu plus de 1 Gigaoctet. Une fois télécharger, décompressez le fichier et placez le répertoire décompressé sur le bureau

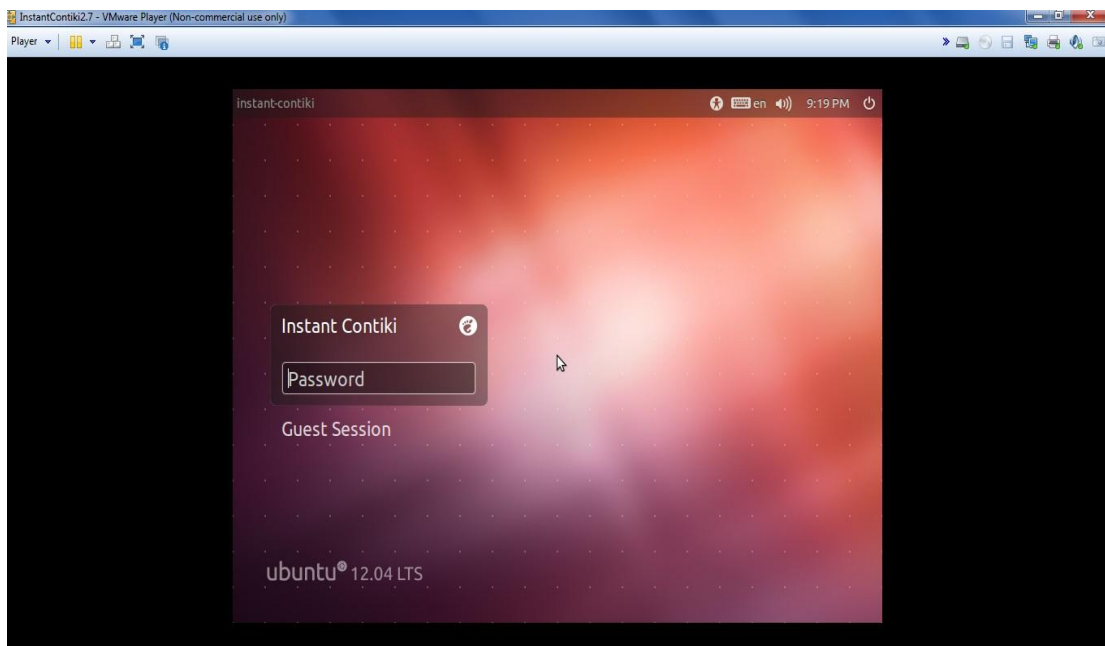
2- Pour faire tourner cette machine virtuelle, il faut télécharger un lecteur des machines virtuelles comme VirtualBox ou VMPlayer. Si vous utilisez VirtualBox, il faut vérifier que l'option « PAE/NX » est activée (Settings → System → Processor)



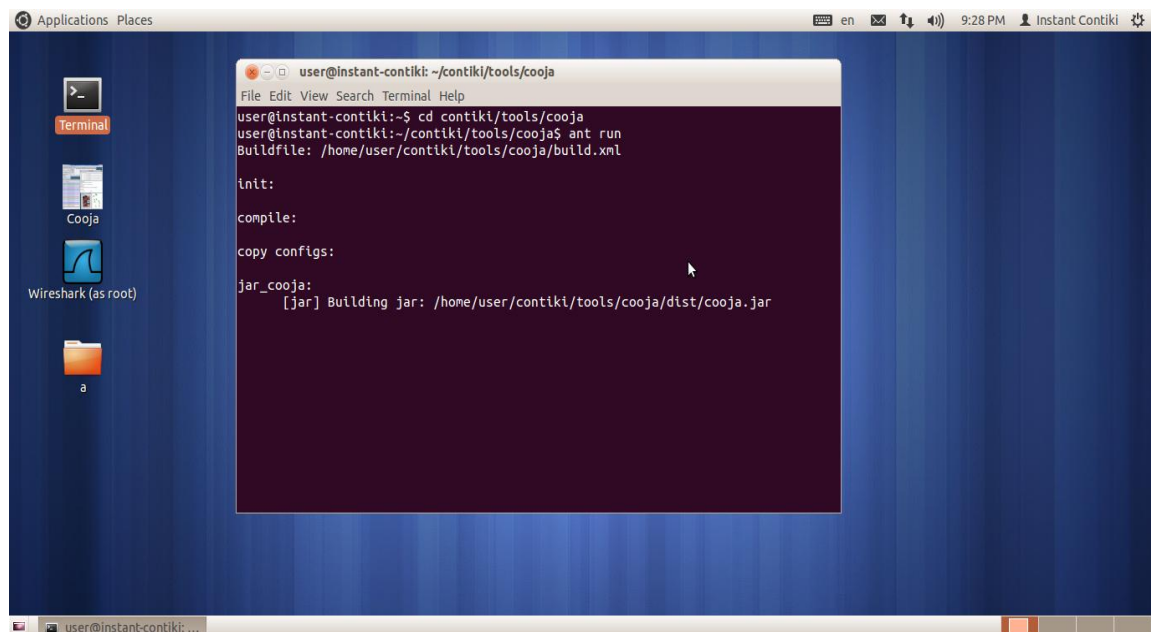
3- Une fois que vous avez installé le lecteur des machines virtuelles avec la machine Instant Contiki, vous allez avoir besoin du login et mot de passe.



Dans ce cas vous utilisez le mot de passe : user.



4- Lancer l'émulateur Cooja : Pour lancer l'émulateur "Cooja", il suffit d'aller dans le répertoire "contiki/tools/cooja", ensuite d'exécuter la commande : « ant run »



```
user@instant-contiki:~/contiki/tools/cooja
File Edit View Search Terminal Help
user@instant-contiki:~$ cd contiki/tools/cooja
user@instant-contiki:~/contiki/tools/cooja$ ant run
Buildfile: /home/user/contiki/tools/cooja/build.xml

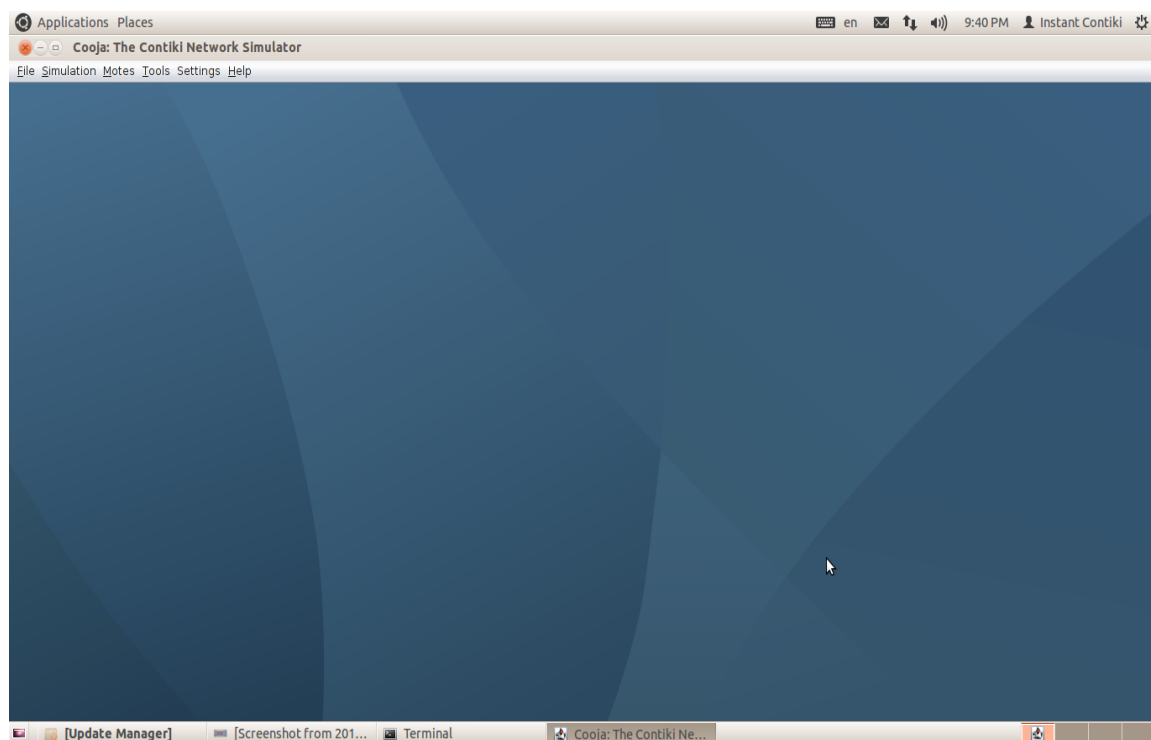
init:

compile:

copy configs:

jar_cooja:
[jar] Building jar: /home/user/contiki/tools/cooja/dist/cooja.jar
```

Lorsque Cooja est compilé, nous obtiendrons une fenêtre bleue vide.



Exécuter l'exemple "Hello, World"

Cet exemple est basé sur un réseau formé de deux nœuds qui affichent uniquement le message « Hello, Word ». Pour pouvoir utiliser ce programme avec l'émulateur Cooja, il faut suivre les étapes ci-dessous :

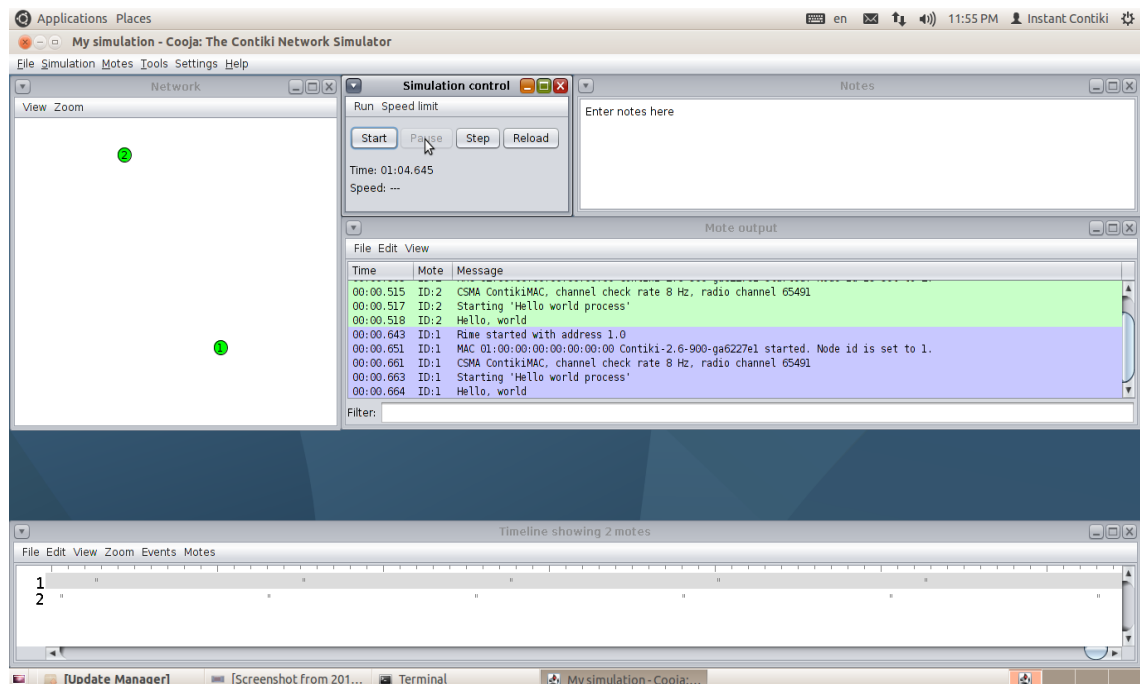
- Créer une nouvelle simulation : File → New simulation
- Compiler le programme « hello-world » et créer 2 Motes :

Motes → Add motes → Create new mote type → Sky mote. Dans le champ « contiki process».

sélectionner le programme à compiler « contiki/examples/ hello-world /hello-world.c».

Ensuite, créer 2 motes à l'aide du bouton Create.

- Lancer la simulation avec le bouton « Start »



- Explication du code :

Hello Word

```

PROCESS (hello_world_process, "Hello world");
AUTOSTART_PROCESSES (&hello_world_process);
PROCESS_THREAD (hello_world_process, ev, data) {
    PROCESS_BEGIN ();
    printf ("Hello, world!\n");
    While(1) {
        PROCESS_WAIT_EVENT ();
    }
    PROCESS_END ();
}

```

Déclarer le processus
 Rendre le processus démarre lorsque le module est chargé
 Définir le code de procédure
 Doit toujours venir en premier
 Code d'initialisation
 Boucle pour toujours
 Attendre que quelque chose se produise
 Doit toujours venir en dernier

Liste des figures

Chapitre 1 : Les réseaux de capteurs sans fil

Figure I. 1: Schématisation de la chaîne de collecte de données [3].	5
Figure I. 2.: les composants d'un nœud capteur [4].	6
Figure I. 3: Architecture d'un réseau de capteur [6].	7
Figure I. 4: Quelques domaines d'application des RSCF [8].	8
Figure I. 5: Arbre d'agrégation de données [3].	10
Figure I. 6: Exemple sans agrégation [5].	10
Figure I. 7: exemple avec agrégation de données [5].	11

Chapitre 2 : Les systèmes d'exploitation pour les réseaux de capteurs sans fil

Figure II. 1 : Architecture du système Contiki [15].	16
Figure II. 2 : Gestion des événements par le système Contiki [3].	17
Figure II. 3: Carte générale de la pile rime [17].	20
Figure II. 4: une image sur Cooja.	22
Figure II. 5: Contiki VS TinyOS [20].	25

Chapitre 3 : L'implémentation

Figure III. 1: Un capteur de type TloseB.	27
Figure III. 2: l'architecture du système.	28
Figure III. 3: choix de type de nœuds avant de faire la compilation de code.	33
Figure III. 4: visionnaire de capteur sur cooja.	33
Figure III. 5: introduire le nombre de nœuds nécessaires pour la simulation.	34
Figure III. 6: les positions des nœuds sur le plan géographique.	34
Figure III. 7: interface graphique sur le simulateur cooja.	35
Figure III. 8: Communication entre les senders et cluster-head.	35
Figure III. 9: Communication entre le cluster-head et la station de base.	36
Figure III. 10: Les valeurs des températures capturées par les Senders.	38
Figure III. 11: la réception des températures envoyées par les Senders et l'envoi de la moyenne à la station de base.	38
Figure III. 12: L'affichage des moyennes reçues par la station de base dans le terminal.	39
Figure III. 13: L'interface JAVA qui représente notre application.	40
Figure III. 14: L'interface de la station de base avant la réception de la moyenne.	40

Liste des tableaux

Chapitre 2 : Les systèmes d'exploitation pour les réseaux de capteurs sans fil

Tableau II. 1 : Caractéristiques des systèmes d'exploitation pour WSN[16]..... 23

Chapitre 3 : Simulation et Réalisation d'une Application des Données Agrégées

Tableau III. 1: Les jeux de couleurs pour l'envoi et la réception de données. 39

Liste des acronymes

A

ACM Association for Computing Machinery
ADC Analog to Digital Converters

C

Cooja Contiki Os Java simulateur
CSMA Carrier Sense Multiple Access

F

FIFO First In First Out

H

HZ Hertz

I

ICMP Internet Control Message Protocol
IEEE Institute of Electrical and Electronics Engineers
IETF Internet Engineering Task Force
IP Internet Protocol
IPV4 Internet Protocol Version4
IPV6 Internet Protocol Version6

K

KO Kilo Octet

M

MAC Media Access Control
MANTIS Multimodal System for Networks of In-situ wireless Sensors
MSP430 Mixed-Signal microcontroller

O

OS Operating System
OSI Open System Interconnections

P

PC Personnel Computer
PDA Personnel Digital Assistant

R

RAM Random Access Memory
RCSF Réseau de Capteurs Sans Fil
RFC Request For Comments
RPL Routing Protocol for Low Power

RSSI	Received Signal Strength Indication
T	
TCMP	Transmission Control Protocol
TI	Texas Instruments
U	
UDP	User Datagram Protocol
UIP	micro IP
W	
WSN	Wireless Sensor Network
6LOWPAN	IPV6 Low Power Wireless Personnel Area Network

Résumé

De nombreux travaux de recherche actuels s'intéressent aux réseaux de capteurs sans fil (RCSF) et à leurs différentes problématiques. L'une d'entre elle est la consommation d'énergie. Dans un capteur et comme il n'est pas possible de recharger leur énergie ni changer les piles, il est nécessaire d'économiser au maximum l'énergie consommée par ces derniers.

L'objectif de ce mémoire est de se familiariser avec le domaine des RCSF et d'implémenter une solution à cette problématique dans une application de surveillance, une approche intéressante a été conçue pour cela «l'agrégation de donnée» qui sert à réduire le nombre de communication, par conséquent, prolonger la durée de vie de réseau en réduisant la consommation d'énergie au niveau des capteurs.

Mots-clefs : RCSF, capteur, agrégation de donnée, énergie, Contiki.

Abstract

Many current research works are interested in wireless sensor networks (RCSF) and in their various problems. One of them is energy consumption in a sensor. Since it is not possible to recharge or to change batteries, it is necessary to minimize the energy consumed by the latter.

The objective of this paper is to become familiar with the field of WSN and to implement a solution for this problem in an application of surveillance, for that an interesting approach was designed «The aggregation of data» Which serves to reduce the number of communication, Consequently, Extend the life expectancy of the network By reducing the energy consumption at the level of the sensors.

Keywords : Wireless sensor networks, Sensor, aggregation of data, energy, Contiki.

ملخص

هناك عدة أبحاث تهتم بشبكات الاستشعار اللاسلكي وبمختلف مشاكلها، من بين هذه المشاكل استهلاك الطاقة في المستشعر، بما أن شحنه أو تغيير بطاريته يعد أمراً مستحيلاً يجب تخفيض الطاقة المستهلكة من طرف هذا الأخير إلى حد أقصى. الهدف من هذه المذكرة هو التعرف على مجال شبكات الاستشعار اللاسلكي واقتراح حل لهذه المشكلة في تطبيق المراقبة و هو " تجميع البيانات" الذي يهدف إلى تخفيض عدد الرسائل مما ينتج عنه زيادة في عمر الشبكة بتخفيض استهلاك الطاقة داخل المستشعر.

الكلمات المفتاحية: شبكة الاستشعار اللاسلكي، مستشعر، تجميع البيانات، الطاقة، نظام التشغيل،