Université de Ngaoundéré
Faculté des Sciences
Département de Mathématiques et Informatique

The University of Ngaoundere
Faculty of Science
Department of Mathematics and Computer Sciences

MÉMOIRE DE MASTER II

Systèmes et Logiciels en Environnements Distribués **(SLED)**

Une approche de multithreading efficace dans les réseaux de capteurs sans fil

WOHWE SAMBO Damien
09A062FS

Sous l'encadrement de:

Dr YENKE Blaise Omer (CC)

28 novembre 2015

Plan de présentation

Plan de présentation

Un monde de capteurs

- Besoins d'observer et de contrôler des phénomènes physiques ;
- Les progrès de le microfabrication ⇒ **les capteurs** ;
- (Température, pression, luminosité ...) ⇒ **CAPTEUR** ⇒ Signal électrique ;
  - Détecteur de fumée/incendie ;
  - Détecteur de proximité ;
  - Accéléromètre ;
  - Détecteur de mouvement.
- Petite taille ⇒ ressources limitées ;
- Harmonie entre ses éléments ⇒ bon fonctionnement du capteur.

Un monde de capteurs

- Besoins d'observer et de contrôler des phénomènes physiques ;
- Les progrès de le microfabrication ⇒ **les capteurs** ;
- (Température, pression, luminosité ...) ⇒ **CAPTEUR** ⇒ Signal électrique ;
  - Détecteur de fumée/incendie ;
  - Détecteur de proximité ;
  - Accéléromètre ;
  - Détecteur de mouvement.
- Petite taille ⇒ ressources limitées ;
- Harmonie entre ses éléments ⇒ bon fonctionnement du capteur.

Un monde de capteurs

- Besoins d'observer et de contrôler des phénomènes physiques ;
- Les progrès de le microfabrication ⇒ **les capteurs** ;
- (Température, pression, luminosité ...) ⇒ **CAPTEUR** ⇒ Signal électrique ;
  - Détecteur de fumée/incendie ;
  - Détecteur de proximité ;
  - Accéléromètre ;
  - Détecteur de mouvement.
- Petite taille ⇒ ressources limitées ;
- Harmonie entre ses éléments ⇒ bon fonctionnement du capteur.

Un monde de capteurs

- Besoins d'observer et de contrôler des phénomènes physiques ;
- Les progrès de le microfabrication ⇒ **les capteurs** ;
- (Température, pression, luminosité ...) ⇒ **CAPTEUR** ⇒ Signal électrique ;
  - Détecteur de fumée/incendie ;
  - Détecteur de proximité ;
  - Accéléromètre ;
  - Détecteur de mouvement.
- Petite taille ⇒ ressources limitées ;
- Harmonie entre ses éléments ⇒ bon fonctionnement du capteur.

Un monde de capteurs

- Besoins d'observer et de contrôler des phénomènes physiques ;
- Les progrès de le microfabrication ⇒ **les capteurs** ;
- (Température, pression, luminosité ...) ⇒ **CAPTEUR** ⇒ Signal électrique ;
  - Détecteur de fumée/incendie ;
  - Détecteur de proximité ;
  - Accéléromètre ;
  - Détecteur de mouvement.
- Petite taille ⇒ ressources limitées ;
- Harmonie entre ses éléments ⇒ bon fonctionnement du capteur.

Architecture d'un noeud

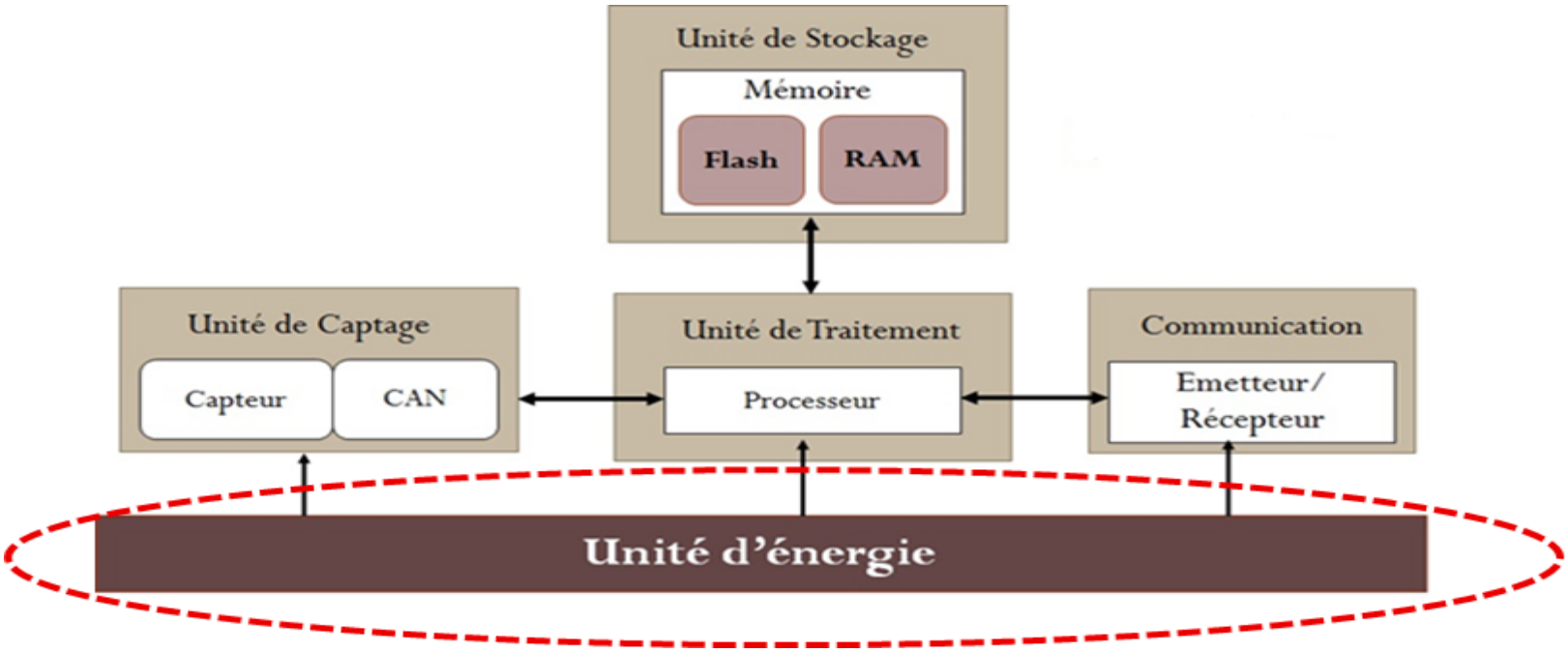


Figure 1: Architecture d'un capteur

- 04 principales unités + 01 source d'énergie (Batterie).
- L'énergie limitée ⇒ durée de vie du capteur ;
- Interconnexion de capteurs couvre une grande zone d'étude : **Réseaux de Capteurs Sans Fil (RCSF)**.

Exemple de RCSF

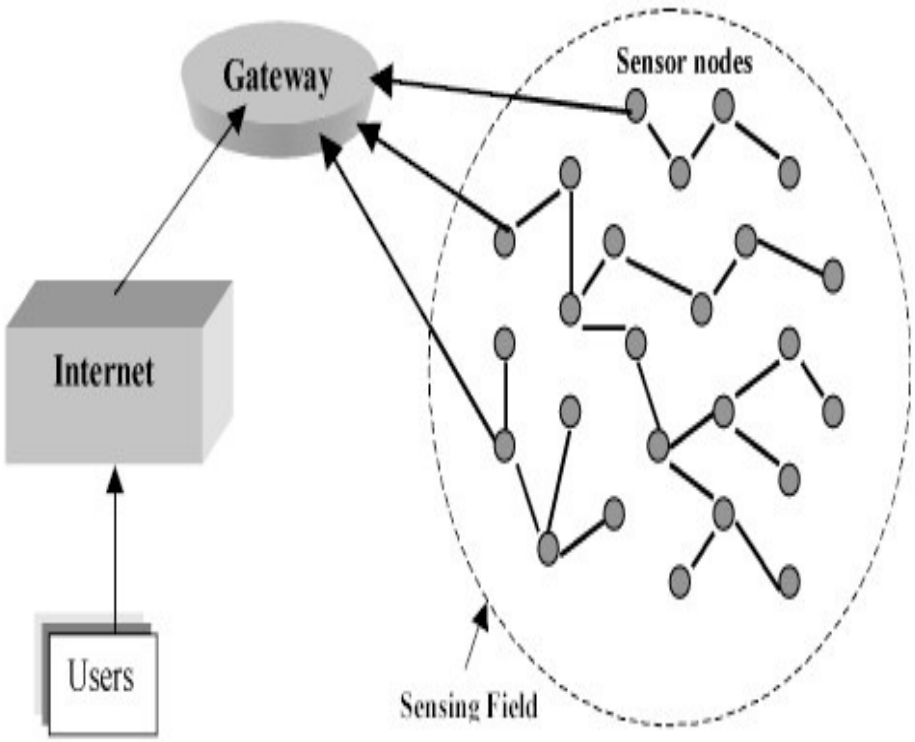


Figure 2: Réseau de capteurs sans fil

- Communications ⇒ Technologie sans fil ;
- Transfert des données par des noeuds intermédiaires (multisauts) ;
- Données transitent vers le noeud puits (gateway) ;
- L'utilisateur et le puits sont reliés à travers Internet ou LAN.

✦ Les RCSFs permettent d'observer plusieurs applications variées.

Exemple de RCSF

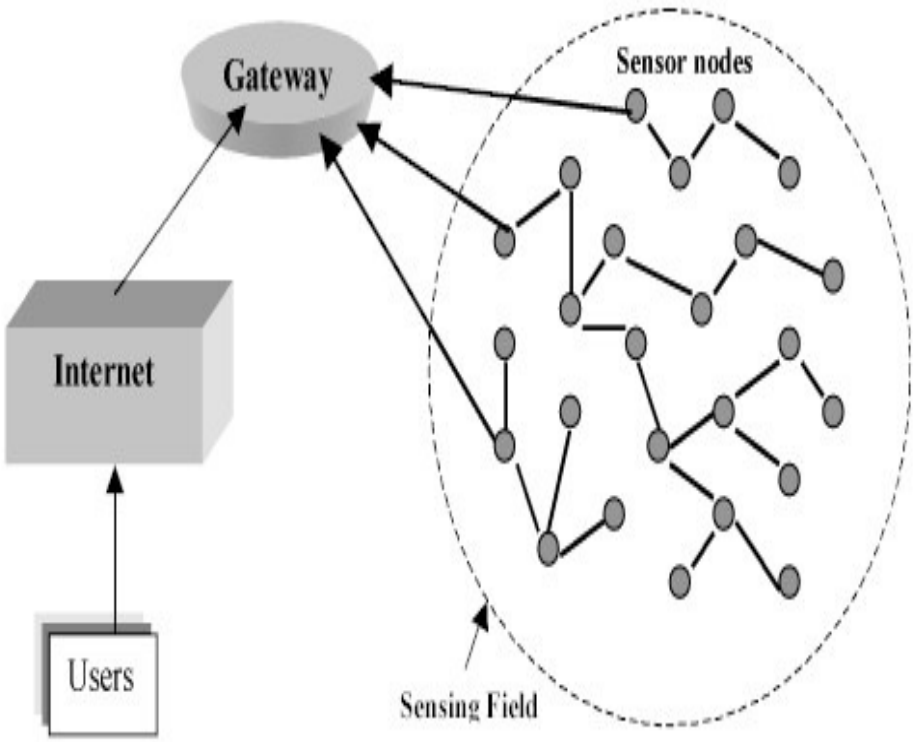


Figure 2: Réseau de capteurs sans fil

- Communications ⇒ Technologie sans fil ;
- Transfert des données par des noeuds intermédiaires (multisauts) ;
- Données transitent vers le noeud puits (gateway) ;
- L'utilisateur et le puits sont reliés à travers Internet ou LAN.

✦ Les RCSFs permettent d'observer plusieurs applications variées.

Applications des RCSFs

(L. Hill, 2003)

DETECTION + CPU + RADIO = MILLIERS D'APPLICATIONS POTENTIELLES

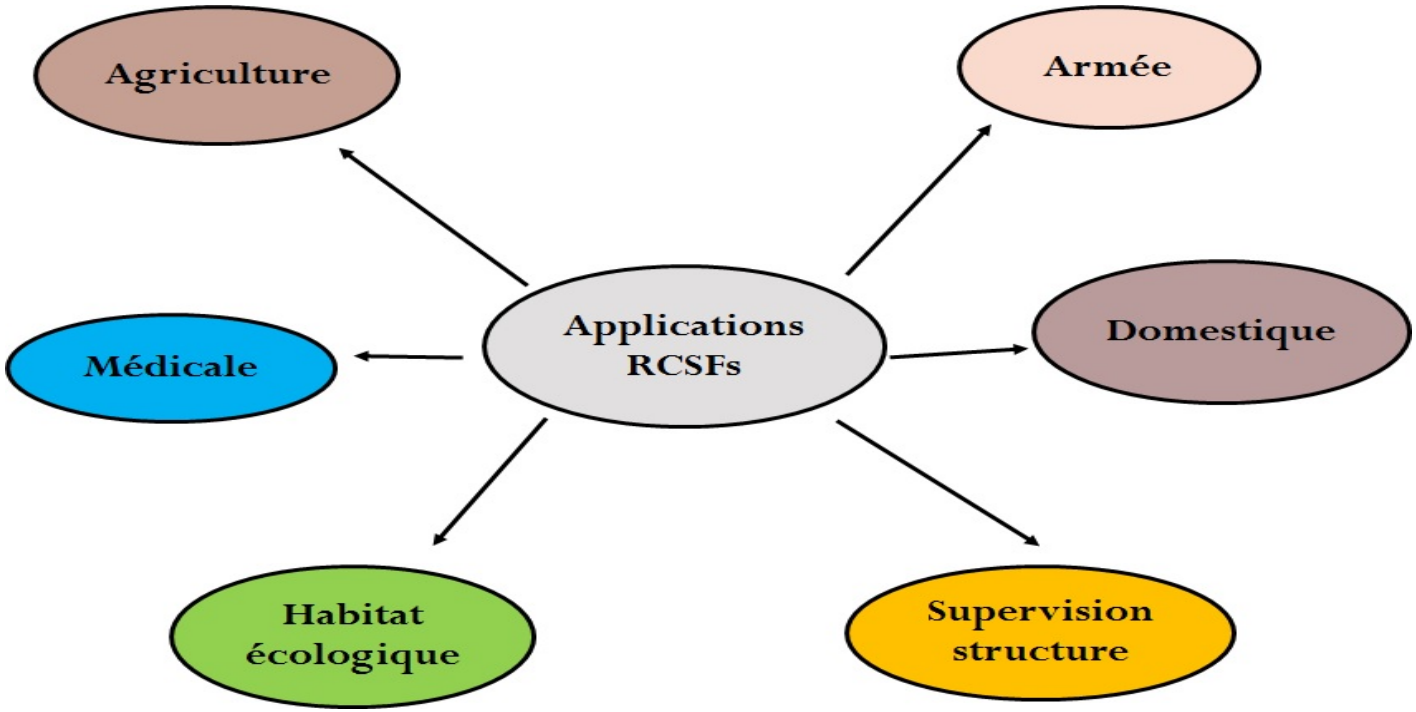


Figure 3: Quelques applications pour RCSF

Certains RCSFs utilisent un traitement multitâche dans leur application.

Applications des RCSFs

(L. Hill, 2003)

DETECTION + CPU + RADIO = MILLIERS D'APPLICATIONS POTENTIELLES

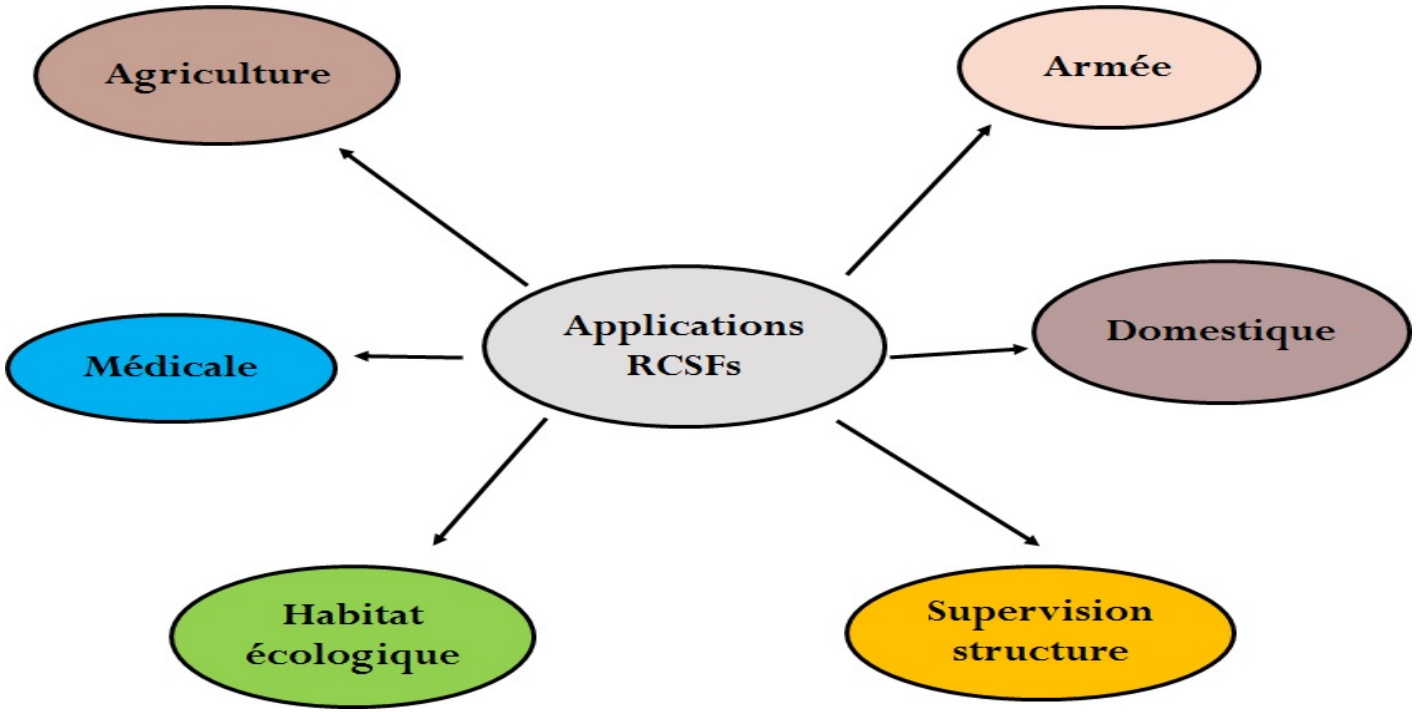


Figure 3: Quelques applications pour RCSF

Certains RCSFs utilisent un traitement multitâche dans leur application.

Multitâche dans les RCSFs

★ Multitâche dans les RCSFs $\Longleftrightarrow$ Multithreading.

Multithreading

Facilite le traitement des tâches en les exécutant simultanément par des threads.

Exemple

L'unité de traitement peut interagir à la fois avec les unités de communication et de captage.

Contrainte mémoire

Dans les environnements à ressources limitées, le multithreading utilise plus de mémoire par conséquent une plus grande consommation énergétique.

Multitâche dans les RCSFs

★ Multitâche dans les RCSFs $\Longleftrightarrow$ Multithreading.

Multithreading

Facilite le traitement des tâches en les exécutant simultanément par des threads.

Exemple

L'unité de traitement peut interagir à la fois avec les unités de communication et de captage.

Contrainte mémoire

Dans les environnements à ressources limitées, le multithreading utilise plus de mémoire par conséquent une plus grande consommation énergétique.

Multitâche dans les RCSFs

★ Multitâche dans les RCSFs $\Longleftrightarrow$ Multithreading.

Multithreading

Facilite le traitement des tâches en les exécutant simultanément par des threads.

Exemple

L'unité de traitement peut interagir à la fois avec les unités de communication et de captage.

Contrainte mémoire

Dans les environnements à ressources limitées, le multithreading utilise plus de mémoire par conséquent une plus grande consommation énergétique.

Multitâche dans les RCSFs

★ Multitâche dans les RCSFs $\Longleftrightarrow$ Multithreading.

Multithreading

Facilite le traitement des tâches en les exécutant simultanément par des threads.

Exemple

L'unité de traitement peut interagir à la fois avec les unités de communication et de captage.

Contrainte mémoire

Dans les environnements à ressources limitées, le multithreading utilise plus de mémoire par conséquent une plus grande consommation énergétique.

Multithreading et durée de vie du RCSF

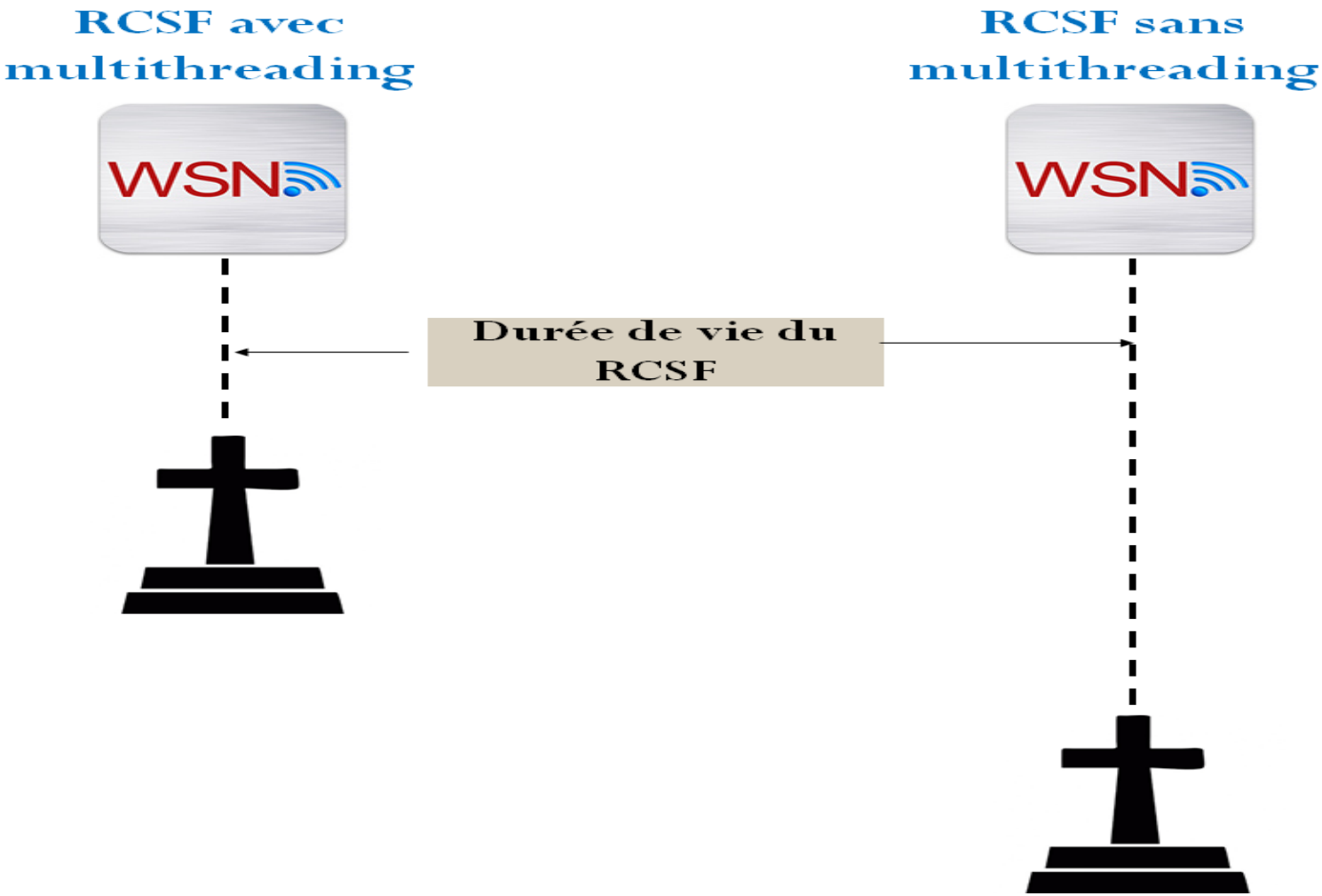


Figure 4: Durée de vie d'un RSCF et multithreading

Problème et objectifs

Problématique

Est-ce que le problème de l'utilisation du multithreading dans les environnements de RCSF peut être résolu ?.

Objectif principal

Proposer une nouvelle vision du multithreading faible consommation dans les RCSFs

Objectifs spécifiques

- Proposer un module de multithreading adapté aux RCSFs ;
- Proposer une bonne performance de traitement ;
- Maintenir une basse consommation énergétique.

Problème et objectifs

Problématique

Est-ce que le problème de l'utilisation du multithreading dans les environnements de RCSF peut être résolu ?.

Objectif principal

Proposer une nouvelle vision du multithreading faible consommation dans les RCSFs

Objectifs spécifiques

- Proposer un module de multithreading adapté aux RCSFs ;
- Proposer une bonne performance de traitement ;
- Maintenir une basse consommation énergétique.

Plan de présentation

Fonctionnalités d'un S.E. dans les RCSFs

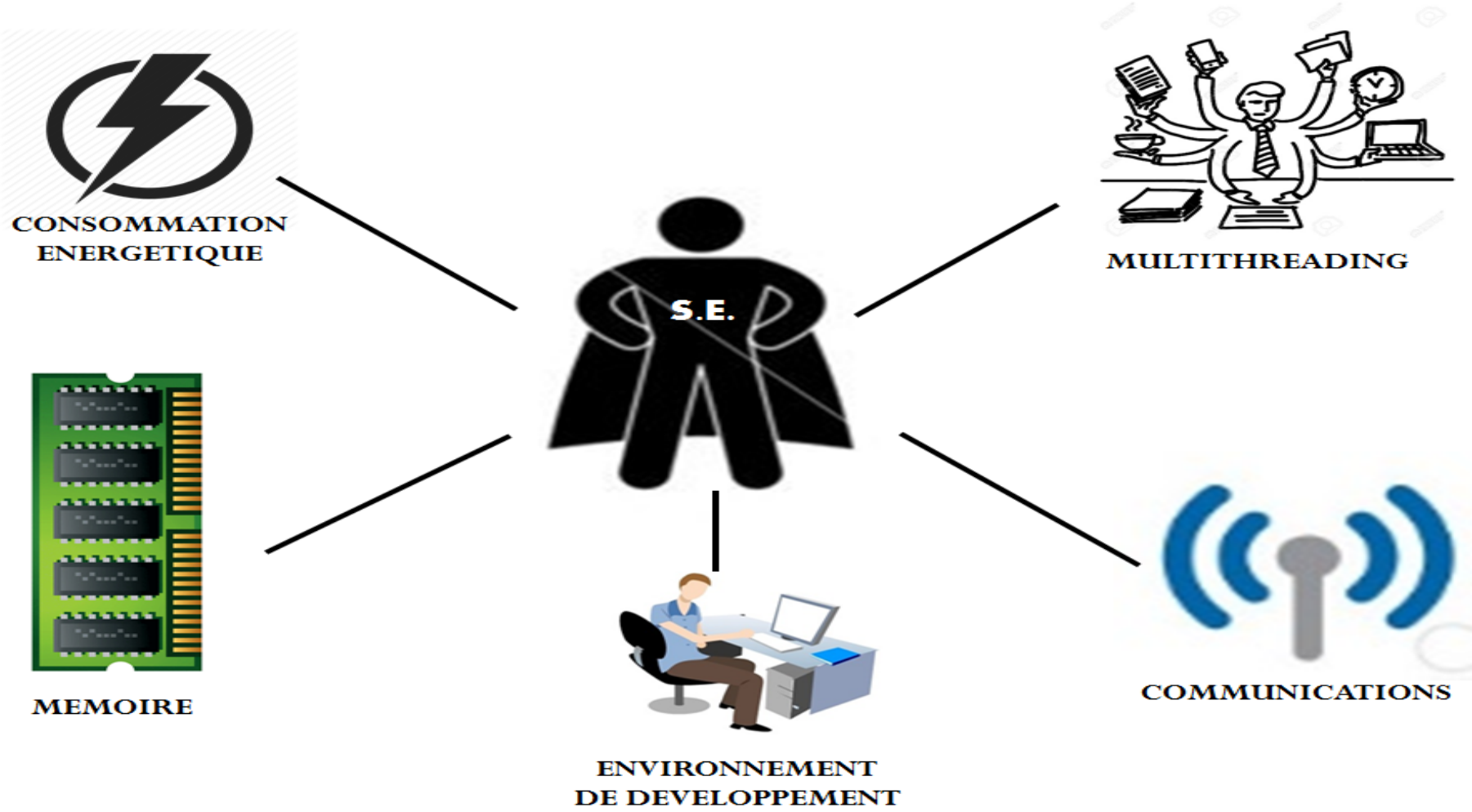


Figure 5: Fonctionnalités S.E. dans les RCSFs

Différents S.E. pour RCSF

☞ Plusieurs S.E. pour RCSF sont recensés dans la littérature :

(L. Saraswat & S. Yadav, 2011)

- Mantis OS
- TinyOS
- Contiki OS
- SOS
- RETOS
- LiteOS
- ...

(T. Reusing, 2012)

- ✳ TinyOS et Contiki ont les plus faibles consommations ;
- ✳ Contiki plus flexible.

Différents S.E. pour RCSF

☞ Plusieurs S.E. pour RCSF sont recensés dans la littérature :

(L. Saraswat & S. Yadav, 2011)

- Mantis OS
- TinyOS
- Contiki OS
- SOS
- RETOS
- LiteOS

...

(T. Reusing, 2012)

- ✳ TinyOS et Contiki ont les plus faibles consommations ;
- ✳ Contiki plus flexible.

Différents S.E. pour RCSF

☞ Plusieurs S.E. pour RCSF sont recensés dans la littérature :

(L. Saraswat & S. Yadav, 2011)

- Mantis OS
- TinyOS
- Contiki OS
- SOS
- RETOS
- LiteOS

...

(T. Reusing, 2012)

✳ TinyOS et Contiki ont les plus faibles consommations ;

✳ Contiki plus flexible.

Contiki OS

(A. Dunkels et al., 2004)

Contiki : Un système d'exploitation multitâche et léger pour RCSF, adaptable et portable sur plusieurs plateformes.

Caractéristiques Contiki

- ✓ Fonctionnement hybride EVENT-BASED/THREAD-BASED ;
- ✓ 02 types de connectivités : la couche faible consommation Rime et la couche orientée Internet uIP ;
- ✓ Le multitâche est géré en option par le module de multithreading préemptif qui utilise à contre partie plus de ressource mémoire ;
- ✓ Implémentation des [illegible] se situant entre un événement et un thread pour une faible consommation.

Contiki OS

(A. Dunkels et al., 2004)

Contiki : Un système d'exploitation multitâche et léger pour RCSF, adaptable et portable sur plusieurs plateformes.

Caractéristiques Contiki

- ✓ Fonctionnement hybride EVENT-BASED/THREAD-BASED ;
- ✓ 02 types de connectivités : la couche faible consommation Rime et la couche orientée Internet uIP ;
- ✓ Le multitâche est gérée en option par le module de multithreading préemptif qui utilise à contre partie plus de ressource mémoire ;
- ✓ Implementation des primitives de synchronisation et communication entre un événement et un thread pour une faible consommation

Contiki OS

(A. Dunkels et al., 2004)

Contiki : Un système d'exploitation multitâche et léger pour RCSF, adaptable et portable sur plusieurs plateformes.

Caractéristiques Contiki

- ✓ Fonctionnement hybride EVENT-BASED/THREAD-BASED ;
- ✓ 02 types de connectivités : la couche faible consommation **Rime** et la couche orientée Internet **uIP** ;
- ✓ Le multitâche est géré en option par le module de multithreading préemptif qui utilise à contre partie plus de ressource mémoire ;
- ✓ Implémentation des processus légers (**protothreads**) se situant entre un événement et un thread pour une faible consommation ;

Contiki OS

(A. Dunkels et al., 2004)

Contiki : Un système d'exploitation multitâche et léger pour RCSF, adaptable et portable sur plusieurs plateformes.

Caractéristiques Contiki

- ✓ Fonctionnement hybride EVENT-BASED/THREAD-BASED ;
- ✓ 02 types de connectivités : la couche faible consommation **Rime** et la couche orientée Internet **uIP** ;
- ✓ Le multitâche est géré en option par le module de multithreading préemptif qui utilise à contre partie plus de ressource mémoire ;
- ✓ Implémentation des processus légers (**protothreads**) se situant entre un événement et un thread pour une faible consommation ;

Contiki OS

(A. Dunkels et al., 2004)

Contiki : Un système d'exploitation multitâche et léger pour RCSF, adaptable et portable sur plusieurs plateformes.

Caractéristiques Contiki

- ✓ Fonctionnement hybride EVENT-BASED/THREAD-BASED ;
- ✓ 02 types de connectivités : la couche faible consommation **Rime** et la couche orientée Internet **uIP** ;
- ✓ Le multitâche est géré en option par le module de multithreading préemptif qui utilise à contre partie plus de ressource mémoire ;
- ✓ Implémentation des processus légers (**protothreads**) se situant entre un événement et un thread pour une faible consommation ;

Contiki OS

(A. Dunkels et al., 2004)

Contiki : Un système d'exploitation multitâche et léger pour RCSF, adaptable et portable sur plusieurs plateformes.

Caractéristiques Contiki

- ✓ Fonctionnement hybride EVENT-BASED/THREAD-BASED ;
- ✓ 02 types de connectivités : la couche faible consommation **Rime** et la couche orientée Internet **uIP** ;
- ✓ Le multitâche est géré en option par le module de multithreading préemptif qui utilise à contre partie plus de ressource mémoire ;
- ✓ Implémentation des processus légers (**protothreads**) se situant entre un événement et un thread pour une faible consommation ;

Rime VS uIP

uIP

- Orientée Internet ;
- Applications variées (Http, ftp ...).

Rime

- Située au dessus de la couche MAC ;
- Fonctionne à l'aide de Radio Fréquence.

(P. Licudis et al, 2011)

- Rime moins complexe que uIP ;
- Rime faible consommation par rapport à uIP ;
- Certaines applications uIP utilisent le couche Rime.

Rime VS uIP

uIP

- Orientée Internet ;
- Applications variées (Http, ftp ...).

Rime

- Située au dessus de la couche MAC ;
- Fonctionne à l'aide de Radio Fréquence.

(P. Licudis et al, 2011)

- Rime moins complexe que uIP ;
- Rime faible consommation par rapport à uIP ;
- Certaines applications uIP utilisent le couche Rime.

Protothreads VS Threads

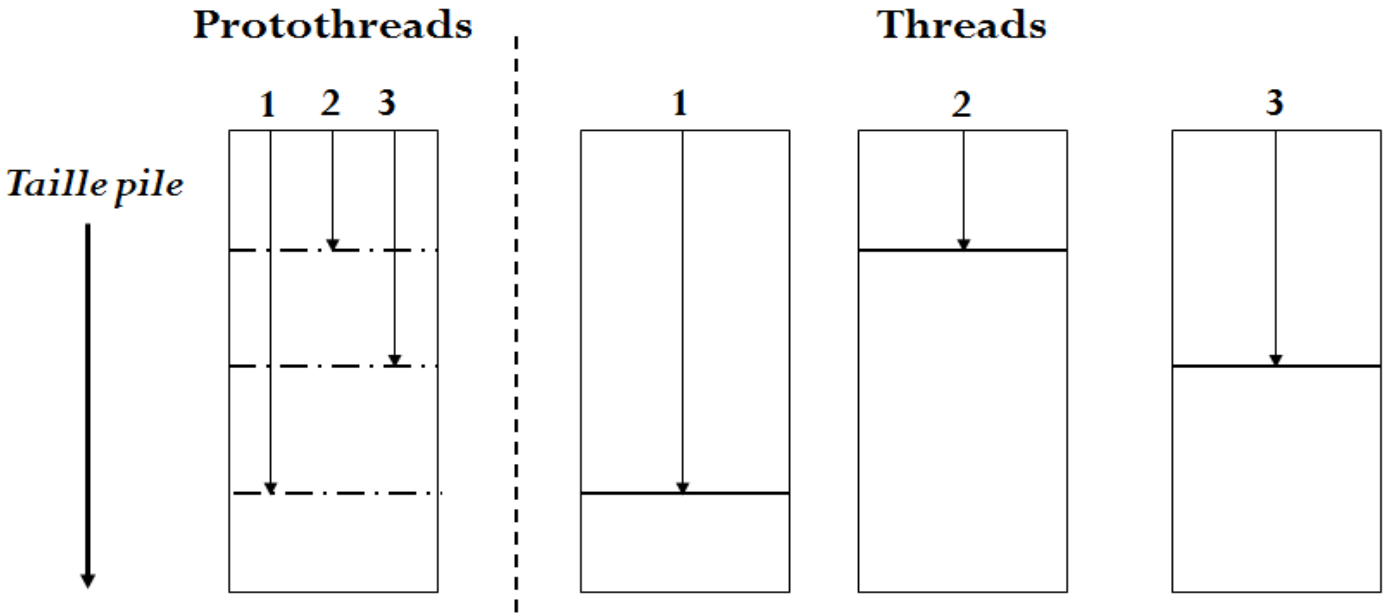


Figure 6: Protothreads et les threads

(A. Dunkels et al, 2006)

- Les protothreads partagent une même pile mémoire.
- Chaque thread s'exécute sur son espace mémoire.

Plan de présentation

Environnement de travail

Outil matériel

- Ordinateur : Hp envy 15
- CPU : core i7 4710HQ @ 2.50 GHz
- Mémoires : 8Gb RAM, 1To Disque dur

Outils logiciels

- Ubuntu 14.04 i386 LTS ;
- Emulateur COOJA intégré dans Contiki-2.7.

Application test

RCSF avec de fortes communications en multisauts à travers un noeud central.

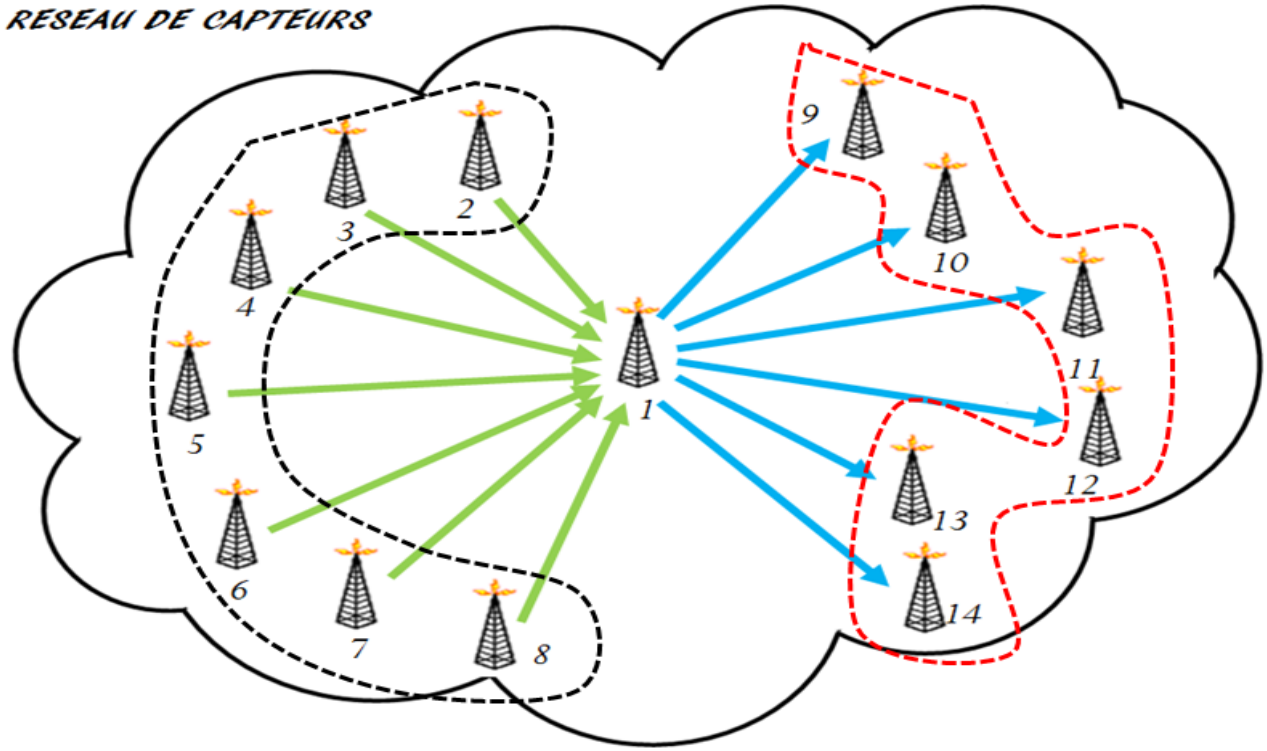


Figure 7: RCSF expérimental

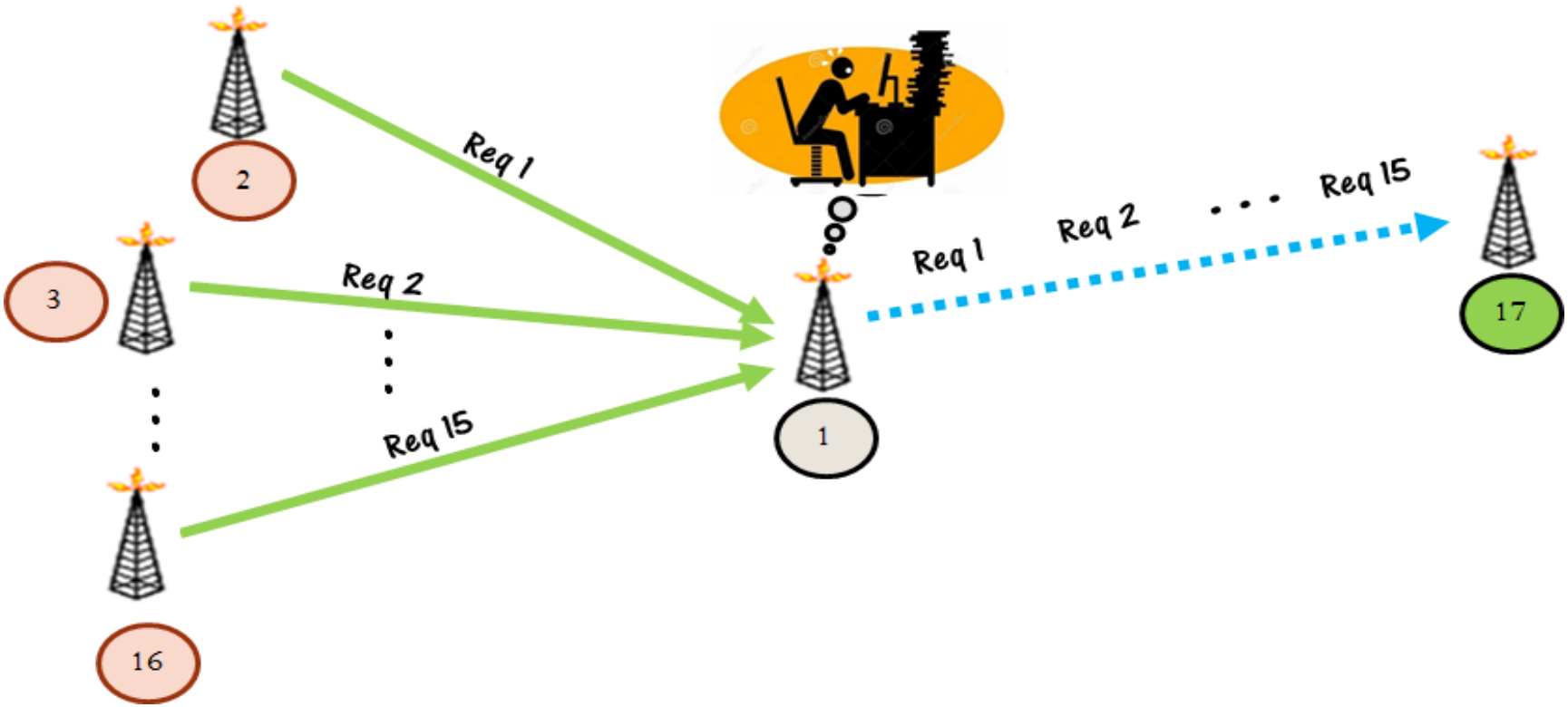
Méthodologie

Méthodologie

- ☆ Les *senders* envoient les paquets au noeud *receiver* à travers *inter* ;
- ☆ Le nombre de *senders* varie de 1 à 15 ;
- ☆ Observation sur 30 cycles (2 secondes entre 2 cycles) ;
- ☆ Une dizaine de test réalisée ;
- ☆ Analyse : rapidité de traitement et consommation ;

Réaliser sur 2 modes de traitement : Normal et Thread

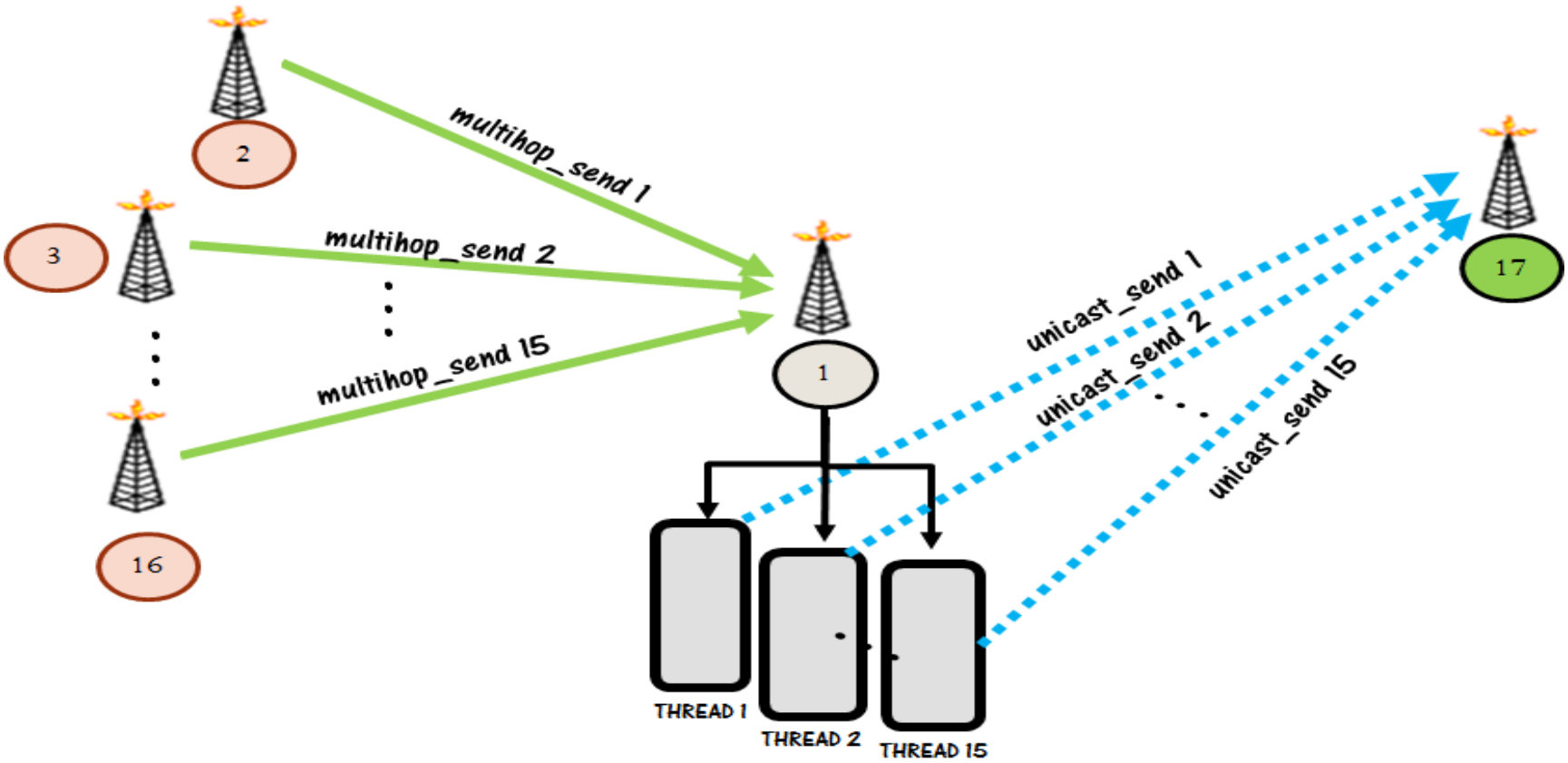
Traitement normal



Module Normal

- ☆ Les requêtes reçues sont traitées séquentiellement ;
- ☆ Transfert en FIFO.

Traitement avec threads



Module Thread

- ☆ Un thread créé après réception d'un paquet.
- ☆ Le thread est responsable de l'envoi à destination du *receiver*.

Résultats

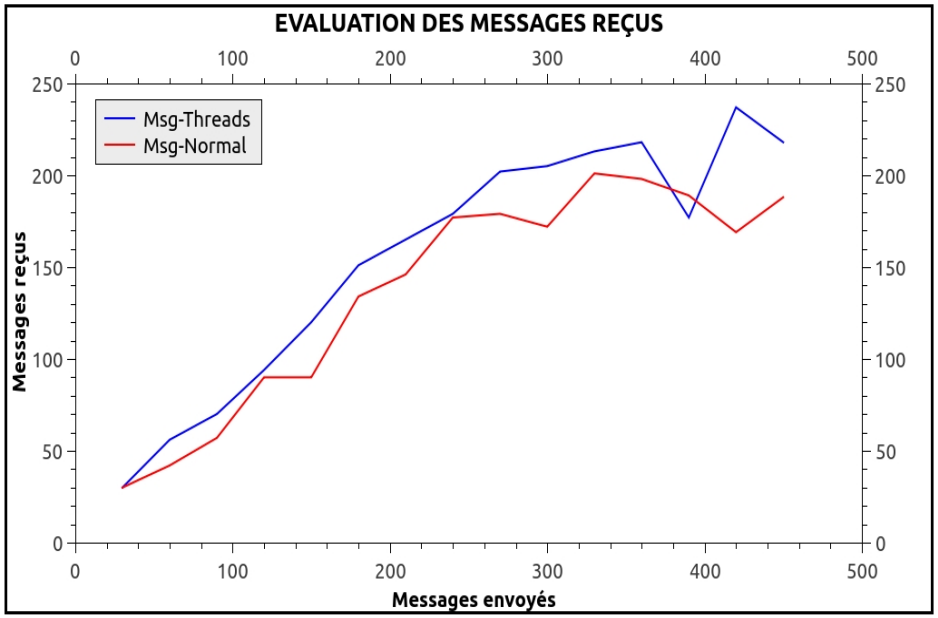


Figure 8: Comparaison taux réception de requête (Normal-Thread)

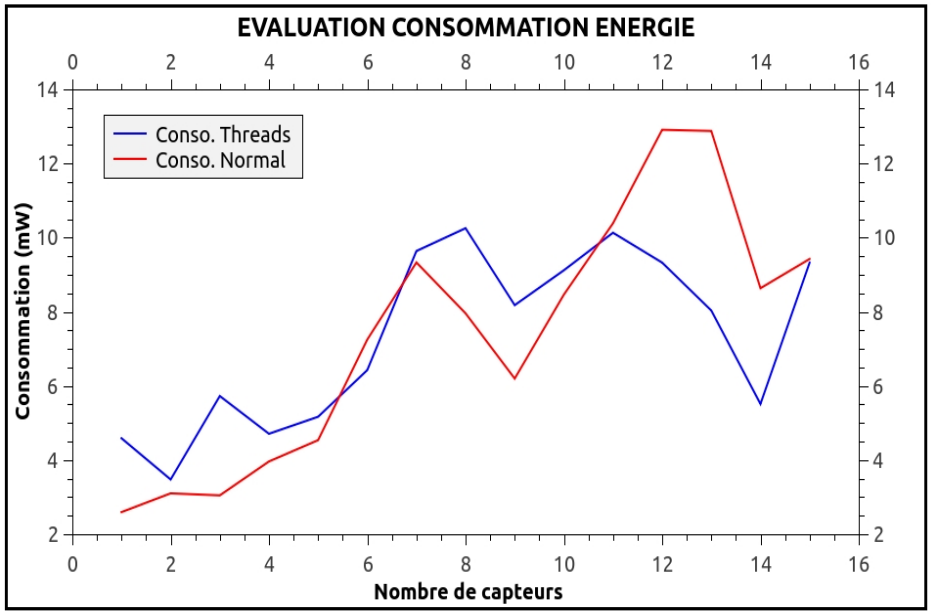


Figure 9: Consommations (Normal-Thread)

Résultats

- Taux de réception Thread de **72.33%** contre **66.96%** pour Normal ;
- Les consommations énergétiques sensiblement identiques.

Plan de présentation

Principe de fonctionnement

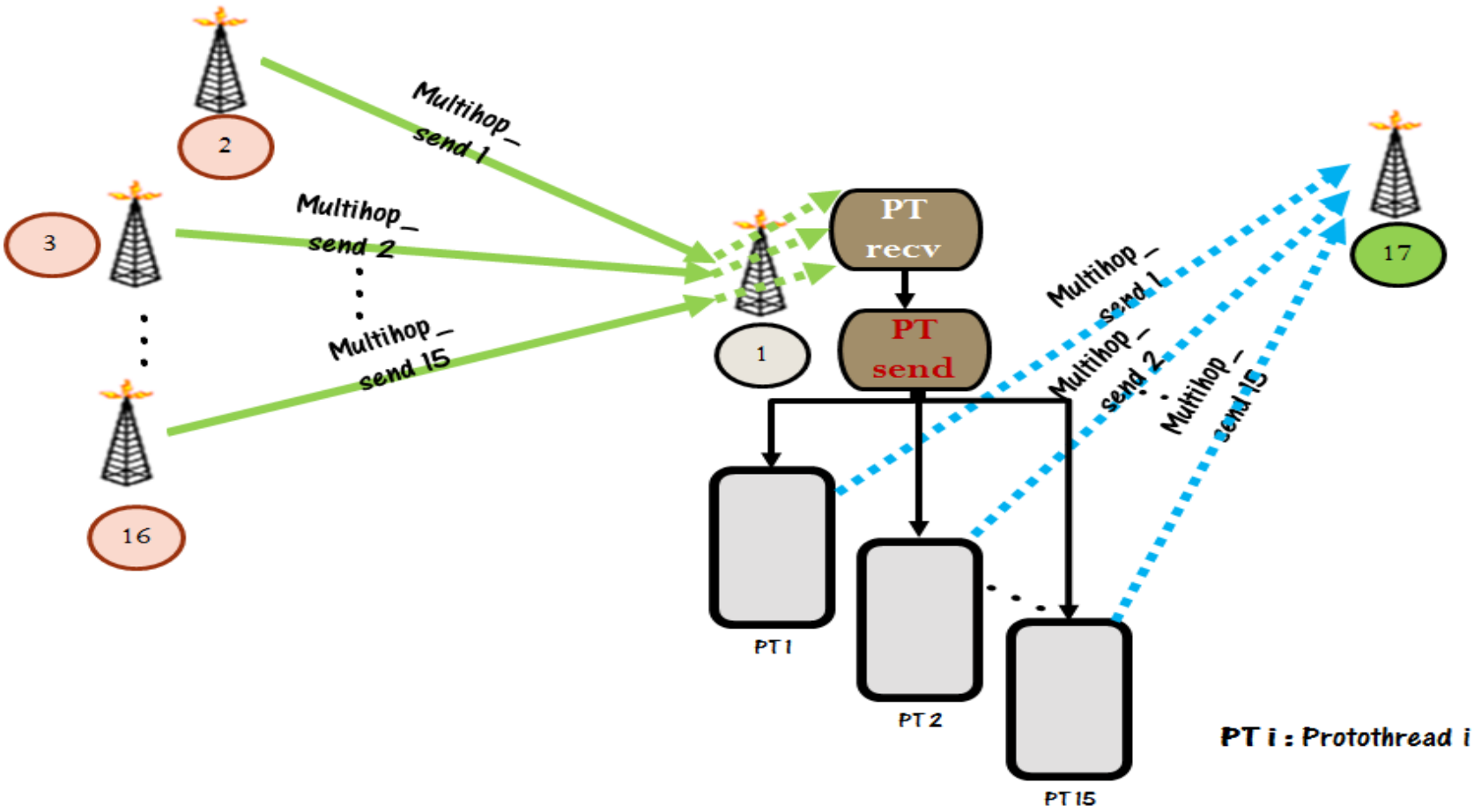


Figure 10: Principe fonctionnement Prototype

Fonctionnement de l'architecture

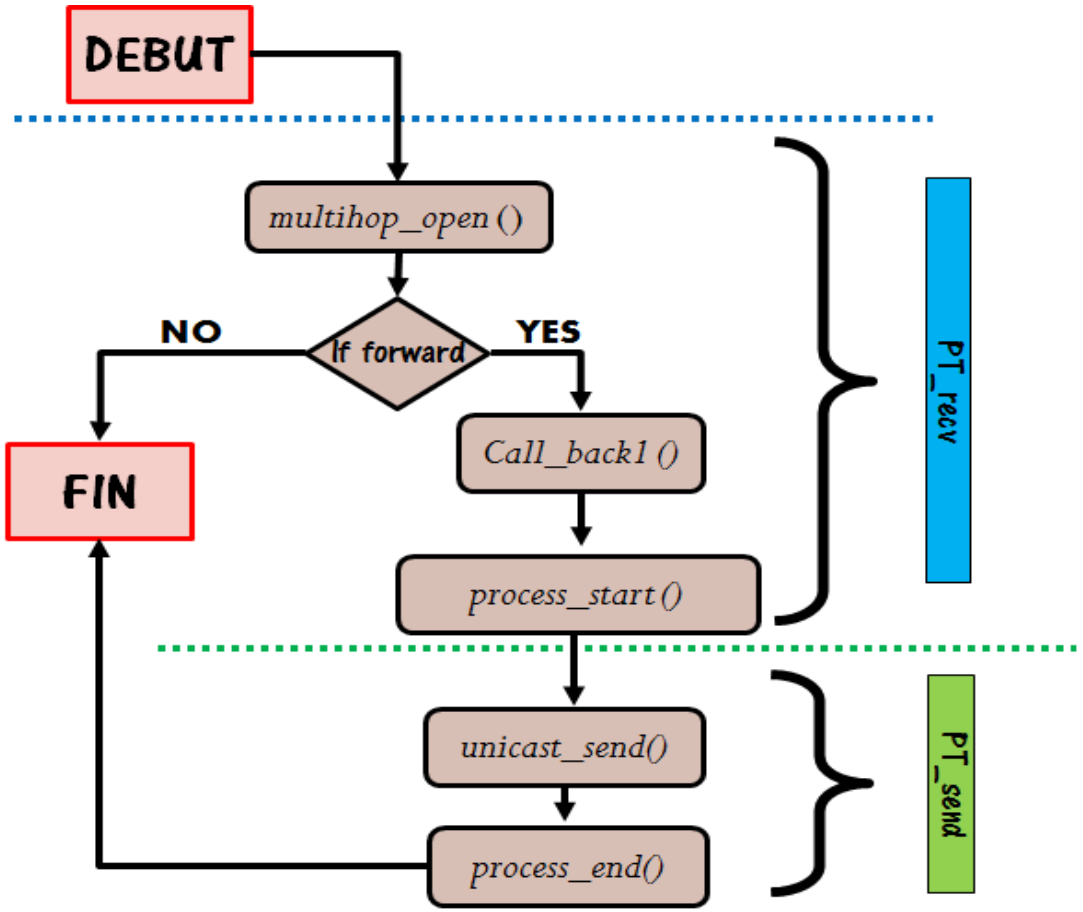


Figure 11: Fonctionnement interne

Normal vs Prototype

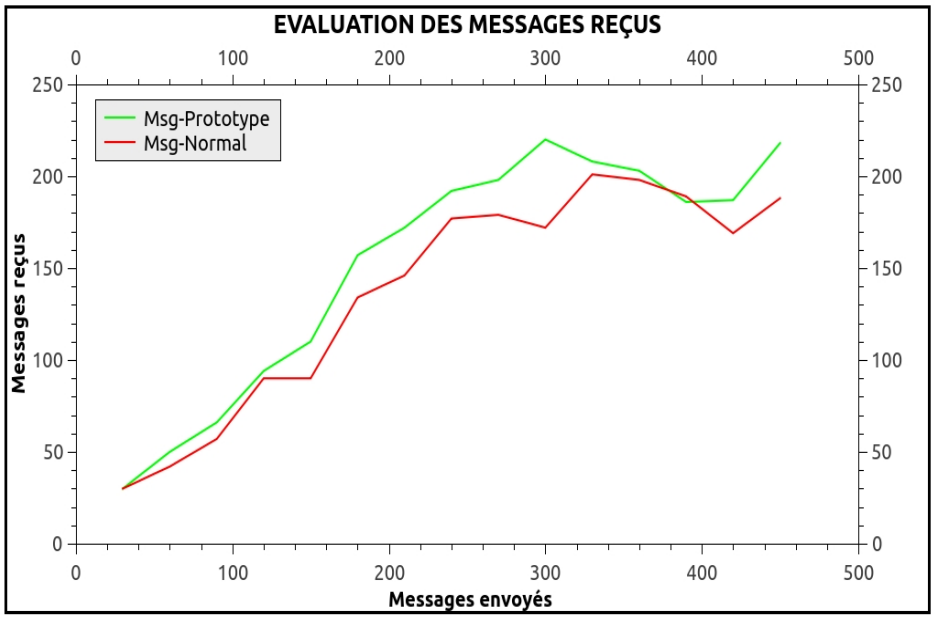


Figure 12: Comparaison taux réception de requête (Normal-Thread)

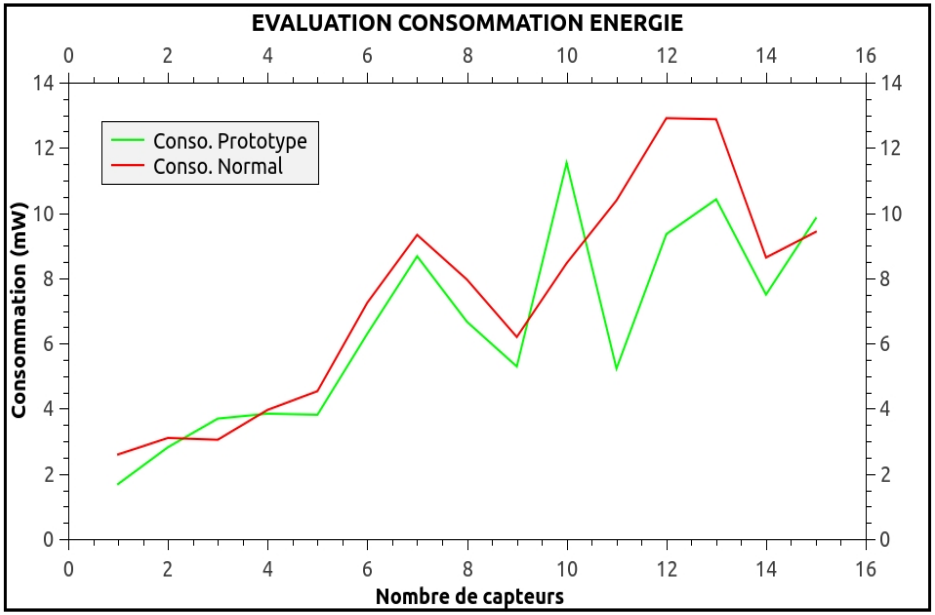


Figure 13: Consommations (Normal-Prototype)

Résultats

- Prototype reçoit **3.99%** de messages en plus ;
- Normal consomme **0.78234 mW** en moyenne de plus que notre architecture.

Thread vs Prototype

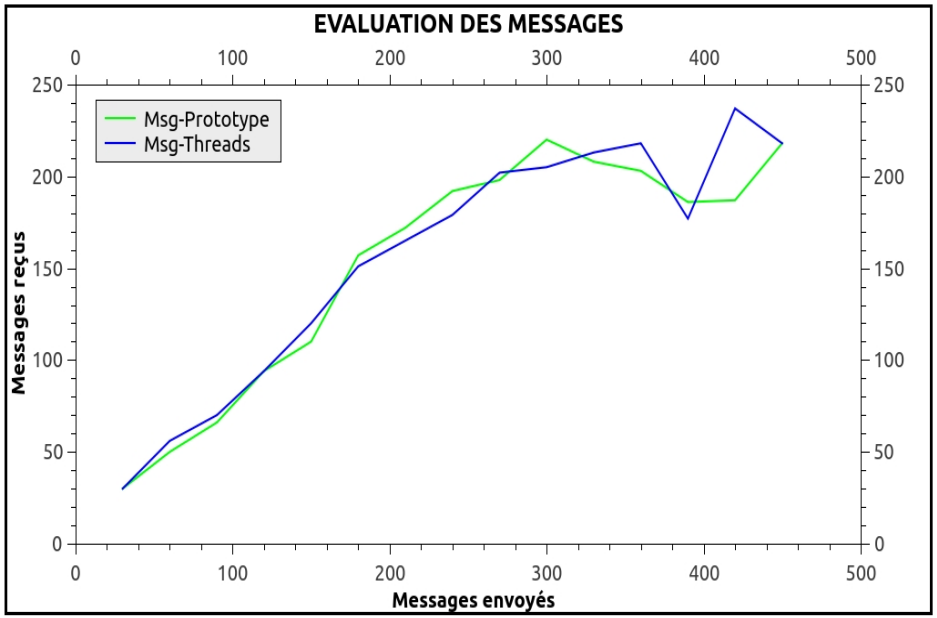


Figure 14: Comparaison taux réception de requête (Thread-Prototype)

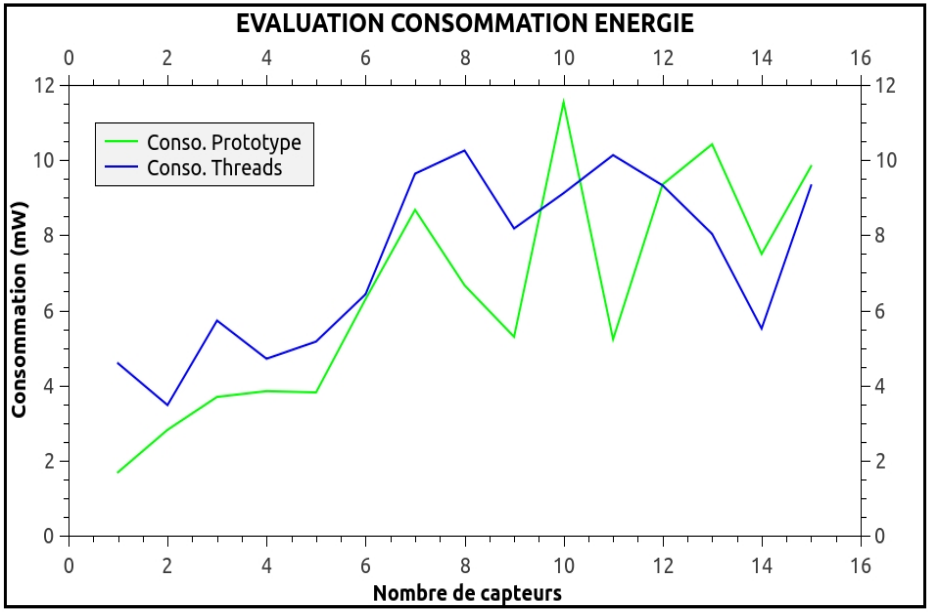


Figure 15: Consommations (Thread-Prototype)

Résultats

- Taux de réception sont sensiblement identiques ;
- Prototype consomme **0.86 mW** en moyenne de moins que Thread.

Plan de présentation

Conclusion

- ☆ Plusieurs contraintes dans les RCSFs à cause des ressources limitées ;
- ☆ Multithreading pouvant réduire la durée de vie d'un RCSF ;
- ☆ Protothreads implémentés par Contiki ;
- ☆ Définition d'un nouveau paradigme du multithreading dans les RCSFs ;
- ☆ Bonne capacité de traitement et meilleure consommation que les deux modules utilisés.

Perspectives

- ✳ Améliorations possibles ;
- ✳ Déploiement du module sur d'autres S.E. ;
- ✳ Vérification des résultats en environnement réel.

Conclusion

- ☆ Plusieurs contraintes dans les RCSFs à cause des ressources limitées ;
- ☆ Multithreading pouvant réduire la durée de vie d'un RCSF ;
- ☆ Protothreads implémentés par Contiki ;
- ☆ Définition d'un nouveau paradigme du multithreading dans les RCSFs ;
- ☆ Bonne capacité de traitement et meilleure consommation que les deux modules utilisés.

Perspectives

- ✳ Améliorations possibles ;
- ✳ Déploiement du module sur d'autres S.E. ;
- ✳ Vérification des résultats en environnement réel.

Références bibliographiques

A. Dunkels, B. Grönvall, and T. Voigt.

Contiki-a lightweight and flexible Operating System for tiny networked.
Technical report, Swedish Institute of Computer Science (SICS), 2004.

A. Dunkels, O. Schmidt, T. Voigt, and M. Ali.

Protothreads : simplify event-driven programming of memory-constrained embedded systems.
Technical report, Swedish Institute of Computer Science (SICS), 2006.

J. L. Hill.

Wireless Sensor Networks.
PhD thesis, University of California-Berkeley(USA), 2003.

J. P. Leal Licudis, J. C. Abdala, G. G. Riva, and J. M. Finochietto.

Flexible Prototyping for Ad Hoc Wireless Sensor Network Protocols.
In *40th Jornadas Argentinas de Informatica (JAIIO),Cordoba (Argentina)*, September 2011.

T. Reusing.

Comparison of operating systems tinyos and contiki.
In *Network Architectures and Services SN SS2012*, volume 02, August 2012.

L. Saraswat and P. S. Yadav.

A comparative analysis of Wireless Sensor Network Operating Systems.
In *5th National conference, INDIACom-2011*, March 2011.

MERCI POUR VOTRE ATTENTION