Université de Mons
Faculté des Sciences
Département d'Informatique
Réseaux et Télécommunications

Conception et évaluation d'une architecture hybride de réseaux de capteurs reposant sur les technologies radio LoRa et IEEE 802.15.4

Directeur : Mr Bruno QUOITIN

Rapporteurs : Mr Prénom NOM
Mr Prénom NOM

Mémoire réalisé par
Arnaud PALGEN

en vue de l'obtention du grade de
Master en Sciences Informatiques

UMONS
Université de Mons

Faculté
des Sciences

Année académique 2020-2021

Remerciements

Nous remercions ...

Table des matières

Table des figures

Chapitre 1

Etat de l'art

1.1 IEEE 802.15.4e

802.15.4 est un protocole définis par IEEE en 2003. Il est destiné aux communications à débit faibles réalsisées par des dispositifs ayant une alimentation en énergie limitée. Ce protocole qui est un standart pour les réseaux PANs (Personak Area Networks) couvre la couche physique et MAC du modèle OSI. En 2012, IEEE 802.15.4e a été défini pour palier à certains problèmes de IEEE 802.15.4.

IEEE 802.15.4

Types de noeuds et topologie

Cette norme défini deux types de noeuds : les Full Function Devices (FFD) qui peuvent être des coordinateurs de PAN, de simple coordinateurs ou de simple noeuds et les Reduced Function Device (RFD) qui utilisent une implémentation réduite du protocole et ne peuvent être que de simples noeuds.

Ces noeuds peuvent former des réseaux suivantes plusieurs topologies comme la topologie en étoile pour laquelle plusieurs RFD sont connectés à un FFD qui joue le rôle de coordinateur ou encore la topologie peer-to-peer pour laquelle les FFD sont connectés les uns aux autres.

Modes d'accès

802.15.4 défini deux modes d'accès : Beacon Enabled mode et Non-Beacon Enabled mode. Dans le premier, le réseau est synchronisé par des messages de contrôles (beacons) et une structure appelée Superframe.

Comme l'illustre la figure 1.1, Cette Superframe est divisée en deux périodes : la période active et la période inactive. La période active est elle-même divisée en deux périodes : contention acces period (CAP) et contention free period (CFP). Dans la première, l'accès au canal se fait par l'algorithme CSMA-CA (Carrier Sense Multiple Access with Collision Avoidance).

Dans la deuxième, l'accès au canal se fait par TDMA (Time Division Multiple Access). C'est à dire que les 7 slots de cette période sont attribués par le coordinateur aux noeuds ayant émis une requête durant la CAP pour l'utilisation d'un slot.

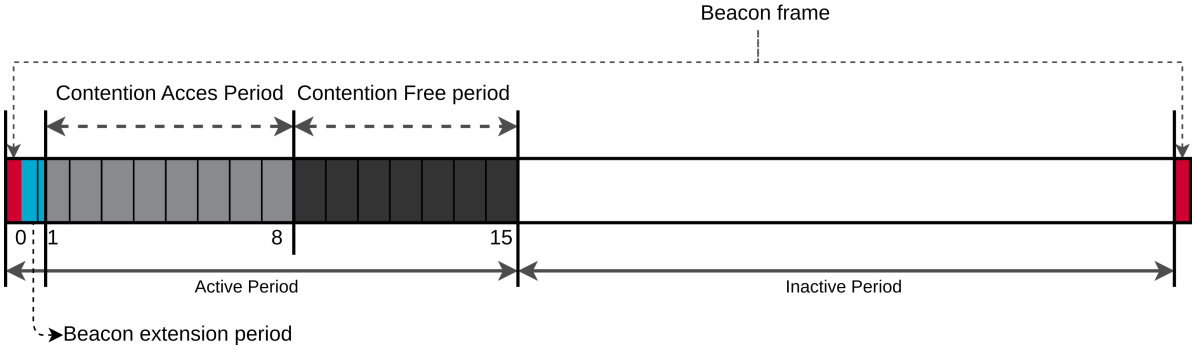


FIGURE 1.1 – 802.15.4 structure de la Superframe.

Dans le mode Non-Beacon Enabled, il n'y a pas de synchronisation. l'accès au canal se fait par l'algorithme Unslotted CSMA-CA.

IEEE 802.15.4e

802.15.4 a un certains nombres de limitations. [?] met les limitations suivantes en évidence :

- Aucune garantie sur le délai maximal pour qu'un paquet puisse atteindre sa destination ne peut être fourni avec l'alorithme CSMA-CA.
- La fiabilité des communications est limitée par l'utilisation de l'algorithme slotted CSMA-CA qui offre un taux de transmission faible.
- Aucune protection contre les interférences due à l'utilisation d'un seul canal et à l'absence de mécanismes de sauts de fréquence (frequency hopping)

Ces limitations ont menées à la création de 802.14.4.e en 2012 qui redifinis les protocoles MAC du standard. Ainsi, 5 modes de fonctionnement de la couche MAC sont introduis :

1. Time Slotted Channel Hopping (TSCH)

2. Deterministic and Synchronous Multi-channel Extension (DSME)
3. Low Latency Deterministic Network (LLDN)
4. Asynchronous multi-channel adaptation (AMCA)
5. Radio Frequency Identification Blink (BLINK)

D'après [**?**], le standart ne définit que brièvement AMCA et BLINK. LLDN est destiné aux réseaux à un seul saut et utilisant un seul canal. Il n'est donc pas pertinent pour ce projet. DSME utilise le concept de multi-superframe semblable aux superframe de IEEE 802.15.4 mise à part la CFP qui divie chacun des 7 slots en plusieurs fréquences.

1.1.1 TSCH (Time Slotted Channel Hopping)

Ce mode de fonctionnement de la couche MAC, comme son nom l'indique, supporte à la fois les sauts en fréquence et des communications divisées en temps. Ces mécanismes réduisent efficacement les effets des interférences et les collisions ce qui améliore la fiabilité du réseau.

Slotframe

Dans ce mode, le concept de superframe de 802.15.4 est remplacé par le concept de slotframe. Une slotframe est un intervalle de temps qui divisé en timeslots. Chaque timeslot permet à un noeud d'envoyer une trame et d'éventuellement recevoir son acquitement (ack). Chaque timeslot possède un identifiant appellé *Absolute Slot Number* (ASN) et un identifiant au sein de la slotframe appelé *Time Slot Number* (TSN).

TSCH permet l'utilisation de timeslots partagés et dédiés. Dans les timeslots partagés plusieurs noeuds peuvent communiquer. Dans ce cas, CSMA/CA est utilisé. Pour les timeslots dédiés, seul deux noeuds peuvent communiquer. La figure 1.2 illustre une slotframe composée de trois timeslots. Dans cet exemple, on considère 3 noeuds : N, T et U. Chaque timeslot permet une communication entre deux noeud. Par exemple le timeslot ayant comme TSN 0, permet à N de transmettre vers T.

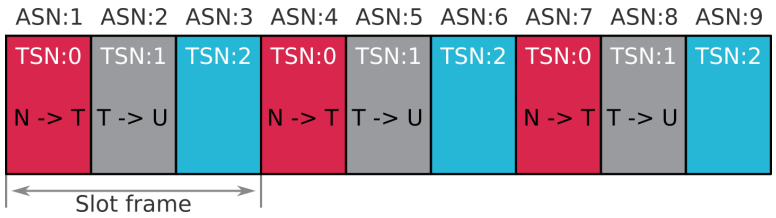


FIGURE 1.2 – Slotframe.

Channel Hopping

TSCH peut utiliser 16 canaux différents numérotés de 0 à 15. Un lien entre deux noeuds dans TSCH est alors défini par la paire (timeslot, canal). Ainsi, chaque slot frame est divisiée par le nombre de canaux utilisé dans le réseau (Fig. 1.3). figure 1.2 Soit f la fréquence utilisée pour la communication entre deux noeuds :

$$f = F[(ASN + channelOffset)\%N_{channels}]$$

où $N_{channels}$ est le nombre de canaux utilisés pour le réseau. F peut être vue comme une table qui contient une séquence de canaux.

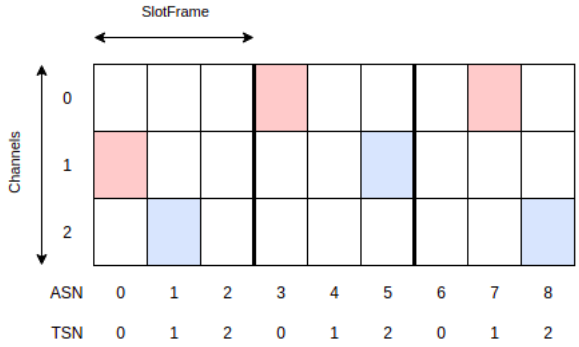


FIGURE 1.3 – Time Slotted Channel Hopping.

1.2 RPL

Routing Protocol for Low-Power and Lossy Networks [**?**] est un protocole de routage IPv6 destiné aux réseaux dont les noeuds sont contraints en énergie et dont les liens entre ces noeuds sont soumis à des pertes importantes de paquets (Low-power and Lossy Networks (LLNs)). Ce protocole à vecteur de distance est un protocole proactif, c'est à dire que les routes sont établies avant qu'elles ne soient nécessaires.

RPL sépare le traitement et la transmission des paquets de l'optimisation de l'objectif de routage. Cela permet de l'adapter à un large éventail d'applications des LLNs.

Topologie

La topologie utilisée par RPL est le DODAG (Destination Oriented Dag). Un DODAG est un graphe dirigé acyclique (DAG) ayant une seule racine (Fig. 1.4). De part cette architecture RPL est adapté aux applcations de collecte de données ce qui est le principale objectif de ce projet. En effet, la route d'un noeud à la racine est facilement établie car tous les noeuds du chemins envoient les données à leur parent.

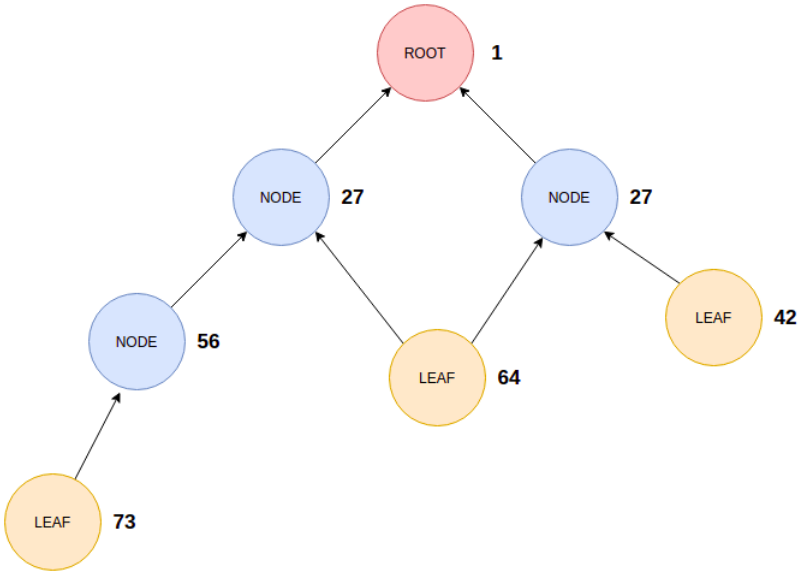


FIGURE 1.4 – DODAG.

Fonctions objectif

Une fonction objectif (OF) défini comment plusieurs métriques sont utilisées pour calculer le rang d'un noeud. Le rang d'un noeud détermine sa position dans le DODAG par rapport aux autres noeuds. Le rang augmente stictement dans le sens descendant et diminue strictement dans le sens montant. Ainsi, pour un noeud n, $rang(n) > rang(parent(n))$. La figure 1.4 illustre un DODAG avec des valeurs de rang fictives attribuées aux noeuds. Sur cet exemple, la racine du DODAG a comme rang, la valeur par défaut ROOT_RANK définie dans le RFC.

Cette section décris brièvement deux fonctions objectif implémentées dans Contiki : OF0 et MRHOF.

- **OF0**

  Objective Function Zero [**?**] est une fonction dont l'objectif est de choisir un parent qui permettra à un noeud d'avoir la racine du DODAG le plus proche possible. Le rang d'un noeud $R(N)$ est calculé comme suit :

  $$ranck\_increase = (Rf * Sp + Sr) * MinHopRankIncrease$$

  $$R(N) = R(P) + ranck\_increase$$

  où
  — Rf est le $rank\_factor$ et $Sr \leq stretch\_of\_rank$ avec $rank\_factor$ et $stretch\_of\_rank$ deux paramètres de la fonction
  — Sp est le $step\_of\_rank$ qui est une valeur basée sur les propriétés du lien
  — $MinHopRankIncrease$ est une constante
  — $R(p)$ est le rang d'un parent p
  $R(N)$ est calculé pour chaque parent potentiel. Le parent potentiel qui implique le plus petit $R(N)$ sera choisi.

- **MRHOF**

  Minimum Rank with Hysteresis Objective Function [**?**] est une fonction dont l'objectif est de sélectionner les routes qui minimisent une métrique additive en utilisant l'hystérésis pour réduire les changements de parents en réponse a de petites variations de la métrique. Pour cela, un noeud va calculer le coût des chemins avec chaque candidat parent. Ce coût sera la somme de deux éléments :

— La valeur de la métrique utlisée contenue dans les DIOs d'un candidat parent
— La valeur de la métrique for le lien entre ce noeud et le candidat parent

Si le coût d'un chemin est plus grand qu'une constante MAX_LINK_METRIC, le noeud va exclure le candidat parent utilisant ce lien.
Un noeud va choisir comme parent le candidat parent pour lequel le coût du chemin est le plus petit. Le changement de parent ne s'effectue pas si ce nouveau coût est plus petit que le coût actuel moins PARENT_SWITCH_THRESHOLD. Ceci est l'hystérésis de MRHOF.

Messages RPL

DIO

Les DIOs (DODAG Information Object) annoncent des informations sur le DODAG qui permettent aux noeuds de découvrir une instant RPL, de sélectionner un parent ou encore de maintenir le DODAG. Un DIO, illustré à la figure 1.5, inclu notemment les champs suivants :

- InstanceID : Identifie l'instance RPL du DODAG
- Rank : Le rang du noeud qui émet le DIO
- MOP (Mode of Operation) : Le mode d'opération de l'instance RPL (c.f. section 1.2 Modes de fonctionnements).
  0. Pas de routes descendantes
  1. Non-storing mode
  2. Storing mode sans multicast
  3. Storing mode avec multicast
- DODAGID : Adresse IPv6 définie par la racine qui identifie le DODAG

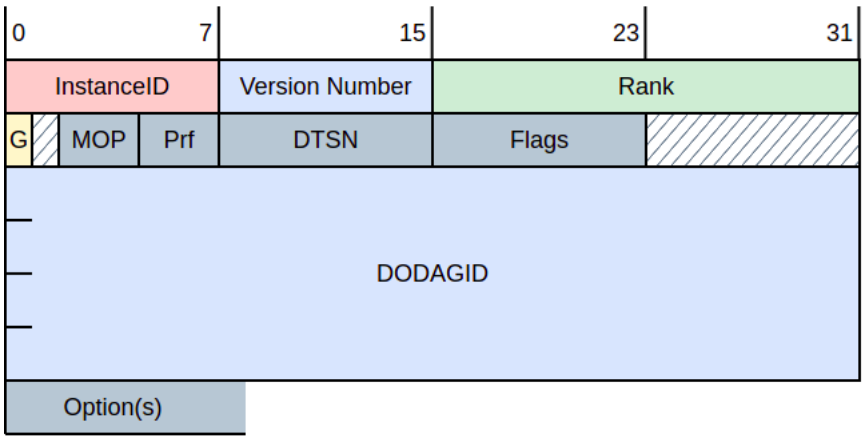


FIGURE 1.5 – DODAG Information Object.

Les DIOs sont transmis en utilisant l'algorithme Trickle définis dans [**?**].

DIS

Les DODAG Information Solocitation (DIS) sont utilisés pour solicité un DIO d'un noeud RPL.

DAO

Les DAOs (Destination Advertisement Object) sont utilisés pour établir les routes descendantes. Ils sont donc envoyés vers le haut du DODAG. Le format du DAO est illustré à la figure 1.6. Un DAO est composé des champs suivants :

- InstanceID : identifie l'instance du DODAG
- K : indique que le destinataire doit répondre avec un DAO-ACK
- D : indique que le DODAGID est présent
- DAOSequence : Incrémenté à chaque DAO d'un noeud et répété dans le DAO-ACK
- DODAGID (optionnel)

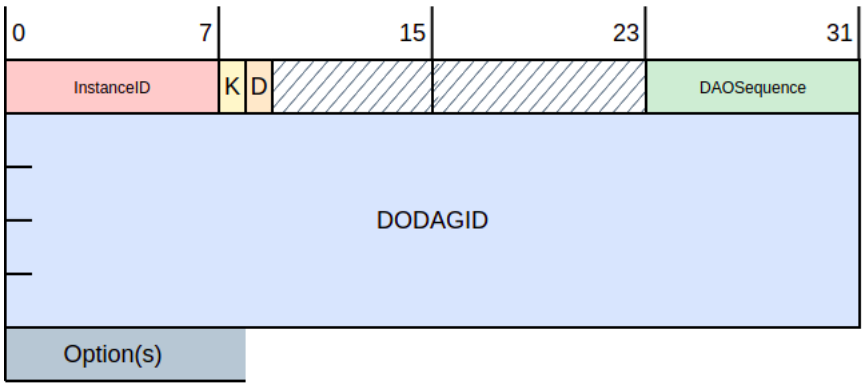


FIGURE 1.6 – DODAG Information Object.

DAO-ACK

Le format d'un DAO-ACK est similaire à celui d'un DAO. Il n'est donc pas utile de le décrire.

Construction du réseau

La construction d'un DODAG est réalisée via l'échange de messages DIOs pour les routes montantes et de DAOs pour les routes descendantes. La figure 1.7 illustre l'échange de ces messages.

Routes Montantes

Les routes montantes sont construites et maintenues avec les DIOs. L'algorithme de construction de DODAG est le suivant :

1. Les noeuds étant configurés comme racine d'un DODAG diffusent des DIOs en mutlicast à tous les noeuds RPL
2. Les noeuds voulant rejoindre un DODAG écoutent ces DIOs et utilisent leurs informations pour rejoindre le DODAG (i.e. sélectionner un parent) ou maintenir le DODAG existant en accord avec la fonction objectif
3. Les noeuds rajoutes des entrée dans leur table de routage pour les destinations spécifiées dans le DIO via leurs parents.

Si une adresse de destination n'appartient pas au DODAG, la racine du DODAG peu transférer les paquets à l'exterieur de ce réseau, ou, si elle n'en est pas capable, les ignorer.

Routes Descendantes

Les routes descendantes sont construites et maintenues avec les DAOS. L'échange des DAOs différe selon le mode de fonctionnement (MOP) du réseau. Ces différences sont détaillées en section 1.2 Modes de fonctionnements.

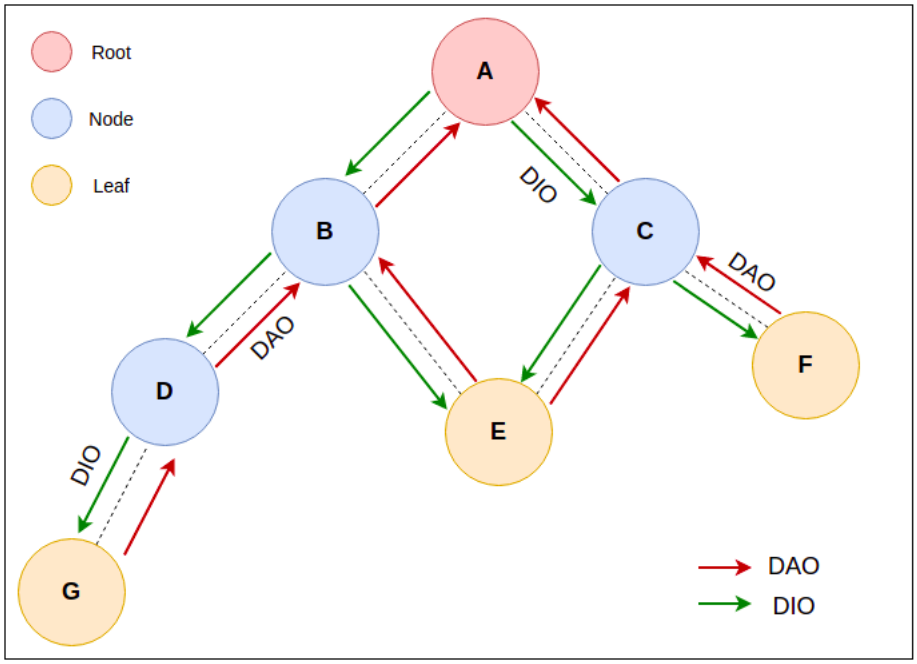


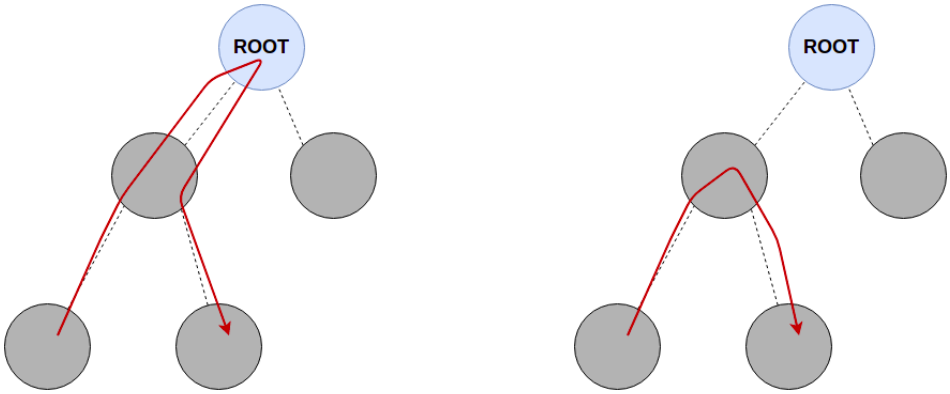
FIGURE 1.7 – Propagation des DIOs et DAOs.

Modes de fonctionnements

Le mode de fonctionnement d'une instance RPL, est défini administrativement et est annoncé par la racine. Les modes suivants sont disponibles :

- **Pas de routes descendantes**
  RPL ne maintient pas de routes descendantes. Dans ce MOP, les DAOs ne sont pas émis par les noeuds d'un DODAG, et les noeuds ignorent les DAOs.
- **Non-storing mode**
  Dans ce mode, les DAOs sont envoyés en unicast à la racine du DODAG. Les noeuds ne stockent pas de routes descendantes. Donc les routes sont établies par source routing. C'est à dire que les paquets remontent jusqu'à la racine du DODAG qui place dans les paquets tous les sauts de la route pour ensuite redescendre (Fig. 1.8a).
- **Storing mode**
  Le storing mode peut être utilisé avec ou sans multicast. Dans ce

mode, les DAOs sont envoyées en unicast par les noeuds à leur parent(s) Les paquets remontent jusqu'à un ancêtre commun avec la destination avant de redescendre vers celle-ci (Fig. 1.8b). Les noeuds stockent les routes de leur sous DODAG. Ainsi, chanque saut dans un chemin examine sa table de routage pour choisir le saut suivant.



(a) Non-Storing. (b) Storing.

FIGURE 1.8 – Chemin d'un paquet en fonction du MOP.

Un noeud voulant rejoindre un DODAG doit être compatible avec le MOP du DODAG. Si ce n'est pas le cas, il doit rejoindre le DODAG comme feuille.

Discussion

D'après [**?**], après avoir été étudiés par le groupe de travail IETF roll (Routing Over Low power and Lossy networks (roll), il s'est avéré que les protocoles tel que OSPF, AODV ou OLSR ne satisfont pas toutes les exigences des LLNs. C'est pour cette raison que ce groupe de travail a introduit RPL.

De plus, de part sa topologie et son fonctionnement, RPL est optimisé pour la collecte de données ce qui est l'utilisation principale pour ce mémoire.

1.3 RTOS

Un RTOS (Real Time Operating System) est un sytème d'exploitation temps réels principalement destiné aux systèmes embarqués.

Etant donné que ce projet utilise le protocole 802.15.4 ainsi que TSCH, le RTOS choisis doit les prendre en charge ainsi qu'un ou plusieurs algorithme

d'ordonnancement de TSCH.

Il est également préférable, que le RTOS choisis, supporte déjà la plateforme utilisée. Une implémentation de la LoRa n'est pas nécessaire car les communications Lora sont réalisées via le RN2483 qui est controlé par UART.

Pour effectuer ce choix, les RTOS suivants ont été comparés : Contiki OS, FreeRTOS et RIOT OS.

Contiki OS

Le développement publique de Contiki a débuté en octobre 2012 [1]. Dans un premier repository : **contiki** [**?**]. Un nouveau développement a démarré en mai 2017 sous le nom de **contiki-ng** [**?**]. C'est donc ce dernier qui est utilisé pour la comparaison est qui est dénommé par la suite "Contiki".

Ce RTOS open-source et multi-plateforme implémente toute une série de protocoles de communications basse énergie tels que IEEE 802.15.4, 6TiSCH, IPv6/6LoWPAN et RPL. En plus de 6TiSCH, un ordonnanceur TSCH, Orchestra, est implémenté.

L'implémentation des processus est basée sur la librairie Protothreads [**?**] qui abstrait la gestion de la programmation évènementiel par des protothread dont l'utilsations est similaire aux threads. L'ordonnenceur de cette libraire est cooperatif, c'est à dire qu'il ne va jamais forcer un changement de contexte(ensemble des ressources nécessaires à l'éxécution d'un processus) d'un processus à un autre (contrairement à un ordonnenceur préemptif). Le changement de contexte ne s'effectue que quand un processus rend volontairement le contrôle à l'ordonnanceur.

Contiki est également compatible avec le Zolertia RE-Mote. Il est accompagné de Cooja, un simulateur réseau qui permet de simuler les communications entre plusieurs noeuds utilisant Contiki.

FreeRTOS

D'après le site officiel de FreeRTOS [**?**], ce RTOS est développé depuis 15 ans. Le repository Github a néanmoins été créé en septembre 2019.

Ce RTOS également open-source et multi-plateforme offre des librairies divisées en trois catégories :

- **FreeRTOS**+ qui contient notemment la librairie TPC/IP et les protocoles applicatifs MQTT et HTTP,

1. Date de création du repository Github.

- **AWS IoT Libraries** qui fourni des librairies permettant la connectivité avec Amazon Web Services (AWS)
- **FreeRTOS Labs** qui contient des librairies complètement fonctionelles mais qui sont encore en cours d'amélioration comme le support d'IPv6, LoRaWan ou le support de plusieurs interfaces réseau

L'ordonnanceur de tâche de FreeRTOS peut être configurer comme coopératif ou préemptif via le flag *configUSE_PREEMPTION* du fichier de configuration.

FreeRTOS n'a pour le moment pas été porté pour le Zolertia RE-Mote ou le CC2538.

RIOT OS

Le développement publique de RIOT a débuté en décembre 2012[1].

La pile réseau de RIOT comporte notemment les protocoles 6LoWPAN, IPv6, RPL, LoRaWan, 802.15.4. RIOT OS intègre la pile réseau OpenWSN [**?**] mais cette intégration est pour le moment expérimentale.

OpenWSN est une implèmentation open-source d'une pile réseau destinée à l'IoT. Cette pile réseau comprend les protocoles IEEE802.15.4e, 6LoWPAN, RPL, UDP et CoAP. L'objectif de cette implémentation est qu'elle puisse être utilisée sur une variété de RTOS et de plateformes.

L'ordonnanceur de RIOT est tickless, c'est à dire qu'il n'utilise pas un timer déclanché périodiquement pour effectuer les changements de contexte. L'ordonnanceur est préemptif et basé sur des priorités. Les changements de contexte sont initiés lors d'interruptions, volontairement ou quand une opération bloquante a lieue.

D'après le site officiel, ce RTOS supporte 229 carte de développement et 64 CPU dont le Zolertia RE-Mote.

Conclusion

Le table 1.1 résume la comparaison de ces RTOS avec les critères les plus importants pour ce mémoire.

RTOS	802.15.4	TSCH	ord. TSCH	IPV6	Routage IP	comp. RE-Mote
Contiki OS	√	√	6Tisch, Orchestra	√√	RPL	√
FreeRTOS	×	×	×	√	×	×
RIOT OS	√	×	×	√√	RPL	√

TABLE 1.1 – Comparatif de différents RTOS.

Le RTOS choisi est Contiki OS. Il est choisi pour sa maturié, la prise en charge du Zolertia RE-Mote et sa pile réseau complète et stable.

Chapitre 2

Architecture

2.1 Topologie

Cette section décrit et motive la topologie et les types de noeuds du réseau hybride IEEE 802.15.4-LoRa de capteurs. Cette topologie a été choisie pour correspondre au mieux au cas d'utilisation décrit dans l'introduction.**TODO: REF INTRO**

Types de noeuds

Le protocole élaboré pour ce réseau hybride défini 3 types de noeuds :
- La **racine LoRa** est la passerelle entre le réseau et un réseau IP externe. Elle possède donc une interface LoRa et par exemple une interface Wi-Fi.
- La **racine RPL**, comme son l'indique, est la racine d'un réseau RPL. Elle possède également une interface LoRa pour communiquer avec la racine LoRa.
- Les **noeuds RPL** sont des noeuds du réseau RPL.

Topologie

Comme l'illustre la figure 2.1 la topologie choisie est une topologie mixte.

On a d'abord un ensemble de réseaux RPL possèdant chacun un préfixe IP attribué par la racine LoRa (par exemple $0x02$ ou $0x04$ sur la figure 2.1), dans lesquels les communications sont réalisées par des liens IEEE 802.15.4. De part l'utilisation de RPL, chaque réseaux forme un DODAG.

Ensuite, chaque réseau RPL est connecté à la racine LoRa via sa racine. Les liens entre les racines RPL et la racine LoRa forment une topologie en

étoile.

Ce choix de topologie offre plusieurs avantages : Le fait d'avoir un ensemble de réseaux RPL implique qu'il n'y a pas qu'une seule racine RPL et donc pas qu'un seul point de défaillance ; L'administration du réseau est également simplifiée. En effet, un réseau RPL pourraît, par exemple, correspondre à une parcelle de terrain.

L'inconvéninent de cette topologie est que la racine LoRa constitue un seul point de défaillance. Une solution plus robuste qui a été envisagée, est un réseau maillé pour les communications LoRa. Cependant, une telle architecture est difficile à mettre en oeuvre car le protocole MAC a établir pour les liens LoRa est plus complexe et nécessite un protocole de routage.

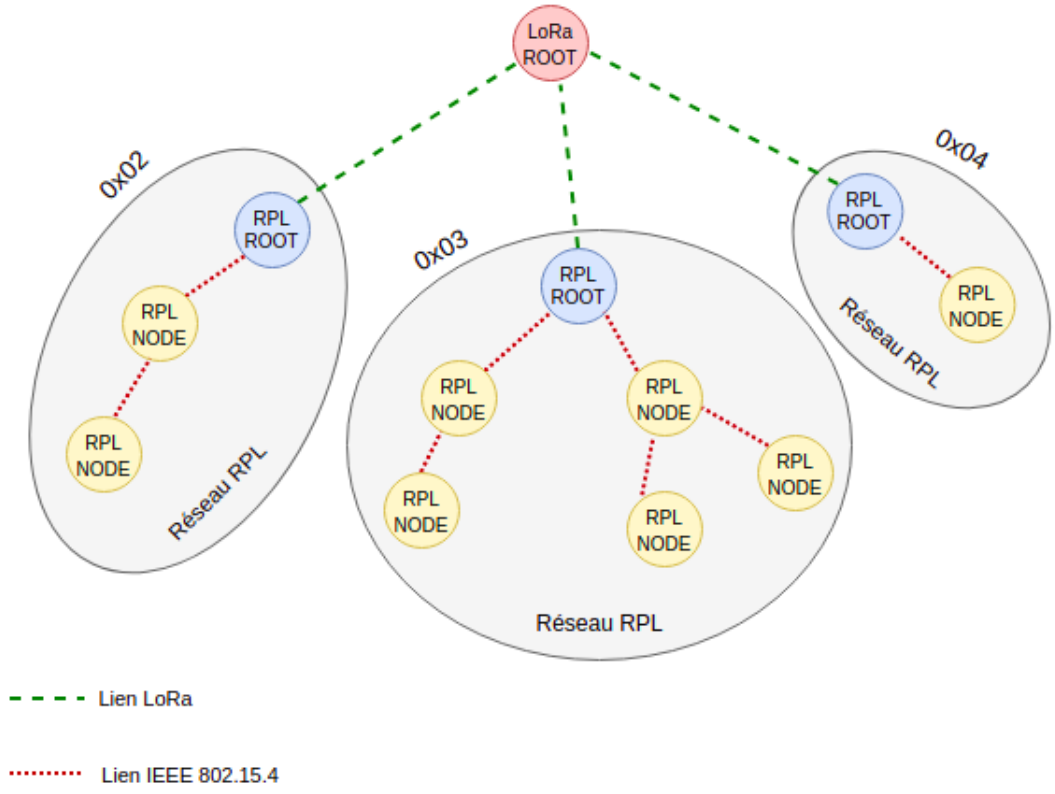


FIGURE 2.1 – Topologie du réseau hybride.

Ainsi, l'objectif de cette topologie est de transmettre les paquets IP à destination de la racine LoRa.

2.2 Format d'adresse

Cette section décris les possibilités envisagées pour le choix du format des adresses utilisées pour les communications LoRa et motive le choix d'un format. Dans chaque hypothèse, un réseau RPL possède un préfixe IPv6.

Adresse IPv6

La première solution envisagée est d'utiliser les adresses IPv6 des noeuds. Une racine RPL peut ainsi vérifier si un paquet est destiné à son réseau et la racine LoRa à elle même, via le préfixe de l'adresse de destintation. Cette solution est la plus simple à mettre en oeuvre car elle ne nécéssite pas de table de routage ou de processus de conversion d'adresse. Néanmoins la taille des adresses IP est de 128 bits.

Préfixe et *node-id*

Le *node-id* est un identifiant sur 16 bits de noeud utilisé dans Contiki. Son calcul, basé sur les deux derniers octets de l'adresse MAC du noeud (*link-layer address*) est le suivant :

$$node\_id = link\_addr[s-1] + (link\_addr[size-2] << 8)$$

où s est la taille de la *link-layer address*.

Le format d'adresse utilisée serait alors un préfixe sur 8 bits et le *node-id* sur 16 bits ce qui fait un total de 24 bits.

L'avantage de cette solution est que la taille des adresses est nettement réduite. Néanmoins, un processus de conversion est nécessaire ainsi qu'une table de conversion d'un *node-id* vers une adresse MAC pour chaque noeud RPL. En effet, le calcul du *node-id* ne permet de reconstituer l'adresse MAC de 64 bits à partir de celui-ci.

Préfixe et adresse MAC

Pour résoudre le problème de la conversion *node-id*↔ adresse MAC, cette solution consiste à remplacer le *node-id* par l'adresse MAC d'un noeud.

Cette solution permet une conversion d'adresse simple et la taille des adresses est de 72 bits (64 bits de l'adresse MAC et 8 bits du préfixe).

Solution retenue

La solution retenue est l'utilisation du préfixe de 8 bits et du *node-id* car c'est la solution qui offre les adresses les plus courtes. Pour éviter l'utilisation d'une table de conversion *node-id*↔ adresse MAC, une meilleure solution que d'utiliser les adresses MAC des noeuds et de modifier les 6 premiers octets de cette dernière. Cela permet de reconstituer une adresse IPv6 simplement, mais cela réduit le nombre d'adresses possible Malgré tout, le nombre d'adresse possible reste largement suffisant pour ce projet.

L'adresse de la racine LoRa est une adresse configurée administrativement.

2.3 Trames LoRaMAC

Le protocole MAC mis en place pour les communications LoRa utilise un seul format de trames illustré à la figure 2.2.

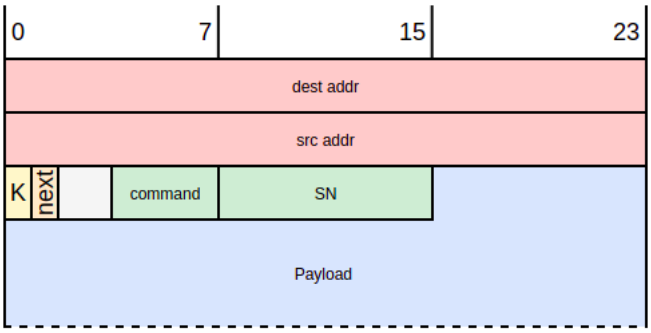


FIGURE 2.2 – Format de trame LoRaMAC.

Les trames LoRaMAC sont composées des champs suivants :

- **dest_addr** : l'adresse de destination (24 bits)
- **src_addr** : l'adresse source (24 bits)
- **K** : un flag qui indique si la trame nécessite un acquittement (1 bit)
- **next** : un flag qui indique si une autre trame suit (1 bit)
- **reserved** : 2 bits non utilisés
- **command** : la commande MAC (4 bits).
  Les commandes disponibles ainsi que leurs valeurs sont reprises dans la table 2.1
- **SN** : le numéro de séquence de la trame (8 bits)
- **Payload** : la payload ayant un MTU de 247 octets

Valeur	Commande
0	JOIN
1	JOIN_RESPONSE
2	DATA
3	ACK
3	QUERY

TABLE 2.1 – Commandes LoRaMAC.

2.4 LoRaMAC

Cette section décris le protocole MAC mis en place pour les communications LoRa. Ce protocole est donc utilisé pour les communications entre la racine LoRa et les racines RPL. Ces dernières étant contraintes en énergie, leur radio LoRa n'est pas tout le temps allumée. Le protocole prend donc en compte cette contrainte.

2.4.1 Etats d'une Racine RPL

Les racines RPL utilisent 3 états illustrés dans le diagramme d'état de la figure 2.3. A l'initiation du réseau, une racine RPL se trouve dans l'état *alone*. Elle demande ensuite son préfixe à la racine LoRa, et une fois celui-ci reçu, la racine RPL se trouve dans l'état *ready*. Quand une racine RPL envoie une trame qui nécéssite un acquittement, tant que celui-ci n'est pas reçu elle se trouvera dans l'état *wait_response*. Une fois l'acquittement reçu, elle retournera dans l'état *ready*.

Les trames qui doivent être envoyées le seront en fonction de l'état dans lequel se trouve une racine RPL. Dans l'état *ready*, les trames peuvent être envoyées, mais ce n'est pas le cas dans les états *alone* et *wait_response*. En effet dans le premier la racine RPL ne peut pas envoyer de données car elle n'a pas encore rejoint le réseau et dans le second, elle attend une réponse.

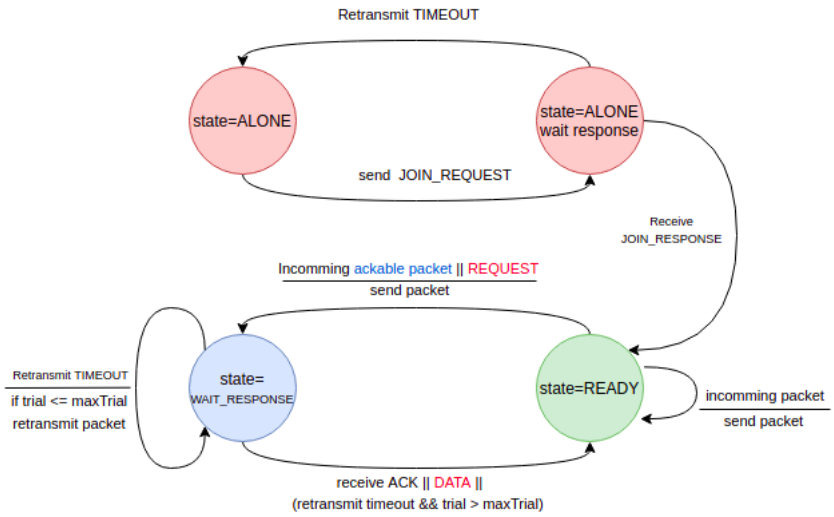


FIGURE 2.3 – Diagramme d'état des racines RPL.

2.4.2 Construction du réseau

A l'initialisation du réseau, une racine RPL doit envoyer une trame avec la commande JOIN. Une fois reçue par la racine LoRa, celle-ci va répondre avec une trame JOIN_RESPONSE qui fait office d'acquittement de la trame JOIN et dont la payload est le préfixe IP que la racine RPL doit diffuser dans son réseau.

Si le trame JOIN ou la trame JOIN_RESPONSE ne sont pas reçues, la racine RPL retransmettra la trame JOIN à l'expiration d'un timer *retransmit_timer* déclanché à l'envoi de la trame JOIN. Cette retransmission se réalise tant que la réponse n'a pas été reçue.

Une fois le préfixe reçu, la racine RPL peut initialiser le réseau RPL et y diffuser le préfixe.

2.4.3 Communications montantes

Lorsq'un paquet est destiné à la racine LoRa, il va remonter jusqu'à la racine RPL. Ensuite celle-ci va l'envoyer à la racine LoRa. Cette dernière a toujours sa radio allumée car elle n'est pas contrainte en énergie.

Si la trame LoRaMAC envoyée à la racine LoRa nécéssite un acquittement, la racine RPL n'enverra pas d'autres trames tant que l'acquittement n'a pas été reçu (état *wait_response*). A l'envoi de cette trame, le timer *retransmit_timer* est déclanché. S'il expire avant que l'acquittement ne soit reçu, la trame sera retransmise pour un maximum de 3 fois.

Si la trame LoRaMAC envoyée à la racine LoRa ne nécéssite pas d'acquittement, la récéption ne cette trame n'est pas garantie et la racine RPL peut directement envoyé d'autres trames.

2.4.4 Communications Descendantes

La racine LoRa possède un buffer pour chaque réseau RPL dans lesquel elle accumule les paquets destinés à ce réseau. Pour chaque racine RPL, lors de l'expiration, à interval régulier, du timer *query_ timer*, une trame avec la commande QUERY est transmise à la racine LoRa.

La réponse à cette trame doit être un acquittement, si la racine LoRa n'a pas de paquets pour cette racine RPL, ou une trame avec la commande DATA.

Si la racine LoRa possède plusieurs paquets à destination du même réseau RPL, la valeur du flag *next* doit être à 1 pour toutes les trames envoyées expecté la dernière.

Comme pour les communications montantes, les retransmissions sont initiées par les racines RPL. Le timer *retransmit_ timer* est activé lorsque la QUERY est envoyée et lorsqu'une trame DATA a le flag *next* à 1.

Si une trame nécéssite un acquittement et que l'acquittement n'est pas reçu, la racine LoRa n'envoie pas le paquet suivant. Dans ce cas deux situations sont possibles :

- La trame avait le flag *next* à 1 :
  La racine RPL s'attend à une autre trame. L'acquittement va donc être retransmis à l'expiration du *retransmit_ timer*. Il y a maximum 3 retransmissions.
- La trame avait le flag *next* à 0 ou l'acquittement a déja été retransmis trois fois :
  La racine LoRa retransmettra le paquet concerné lors de la réception d'une QUERY.

2.4.5 Gestion des numéros de séquence

Les racines RPL et la racine LoRa possèdent deux compteurs : *expected_ sn* et *next_ sn*.

La valeur du compteur *expected_ sn* est le numéro de séquence attendu de la prochaine trame reçue. Ce compteur permet de savoir si une trame a été perdue. Sa valeur est le numéro de séquence de la dernière trame reçue incrémenté de 1.

La valeur du compteur *next_ sn* est le numéro de séquence de la prochaine trame. Il est incrémenté de 1 lorsqu'une trame a été envoyée.

2.4.6 Evitement de collision

L'envoi des trames est réalisé avec l'algorithme CSMA/CA pour les racines RPL. Avant l'envoi, ces noeuds écoute le canal. Si le canal est occupé, le noeud va resonder le canal après un temps aléatoire. Ce processus se répète 3 fois. Après ces trois fois, si le canal est toujours occupé, la trame ne sera pas envoyée. A chaque sondage du canal, l'interval dans lequel le temps aléatoire est choisi est agrandit. Si le canal n'est pas occupé, la trame est envoyée.