

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 : M^r Bruno QUOITIN

Mémoire réalisé par
Arnaud PALGEN

Rapporteurs : M^r Prénom NOM
M^r Prénom NOM

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



Année académique 2020-2021

Remerciements

Nous remercions ...

Table des matières

| | | |
|-----|--|---|
| 0.1 | Plateformes de développement | 2 |
| 0.2 | RTOS | 6 |
| 0.3 | RPL | 8 |

Table des figures

| | | |
|---|--|----|
| 1 | Zolertia RE-Mote révision B [14]. | 2 |
| 2 | RN2483 [10]. | 3 |
| 3 | Schéma-bloc du RN2483 [8]. | 4 |
| 4 | Raspberry Pi 3B+ [9]. | 5 |
| 5 | DODAG. | 8 |
| 6 | DODAG Information Object. | 11 |
| 7 | DODAG Information Object. | 12 |
| 8 | Propagation des DIOs et DAOs. | 13 |
| 9 | Chemin d'un paquet en fonction du MOP. | 14 |

0.1 Plateformes de développement

Zolertia RE-Mote

Pour ce mémoire, la plateforme Zolertia RE-Mote revision B (Fig. 1) est utilisée.

Cette plateforme, basée sur un system on chip (SoC) CC2538 ARM Cortex-M3, a été conçue par des universités et des industriels dans le but de permettre aux chercheurs et makers de développer des applications IoT et des objets connectés.

Le Zolertia RE-Mote a été choisi car elle est équipée de deux radios compatibles IEEE 802.15.4, permet une consommation électrique faible et possède de nombreux pins de connexion qui peuvent être utilisés pour y connecter des capteurs, actuateurs, radios, etc.

Le prix du constructeur pour cette plateforme est de 93,95€ [14].



FIGURE 1 – Zolertia RE-Mote révision B [14].

La table 1 reprend les principales spécifications du Zolertia RE-Mote rev.b et sa table ... la consommation électrique.

| Element | Spécification |
|--------------------|--|
| Radio | Deux radios IEEE 802.15.4 à 2.4 GHz et 863-950 MHz |
| CPU | ARM [®] Cortex [®] -M3 jusqu'à 32 MHz |
| RAM | 32 KB (16 KB pour tous les Power Modes) |
| Flash programmable | 512KB |
| I/O | RGB led, bouton user et reset, USB 2.0 à 12Mbps, Real-Time Clock |

TABLE 1 – Spécifications du Zolertia RE-Mote rev.b [13].
reprend la consommation de courant du RN2483

RN2483

Le RN2483 (Fig. 2) est un modem LoRa compatible LoRaWANTM basse énergie. La communication avec ce modem se fait par des commandes ASCII envoyée via une interface UART. Il prend en charge les modulations FSK, GFSK et LoRa. Il possède également 14 GPIOs pour le contrôle et le status, partagés avec 14 inputs analogiques. Ses fréquences opérationnelles sont situées dans les bandes de fréquences 433 MHz et 868 MHz. D'après la datasheet, sa portée maximale est de 15km en agglomération et 5km en zone urbaine. Comme l'illustre la figure. 2, pour ce mémoire, le RN2483 a été monté sur une carte d'interface réalisée par B.Quoitin qui comporte deux leds, une petite antenne ainsi que les connecteurs permettant d'utiliser des câbles de prototypages.

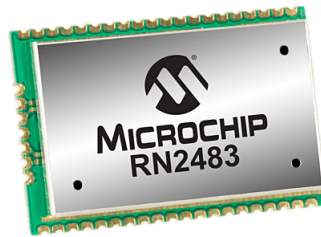


FIGURE 2 – RN2483 [10].

La figure 3 reprend le schéma-bloc du RN2483. Il contient notamment l'interface UART, les antennes 433 MHz et 868Mhz ainsi que les GPIOs et la stack LoRaWan.

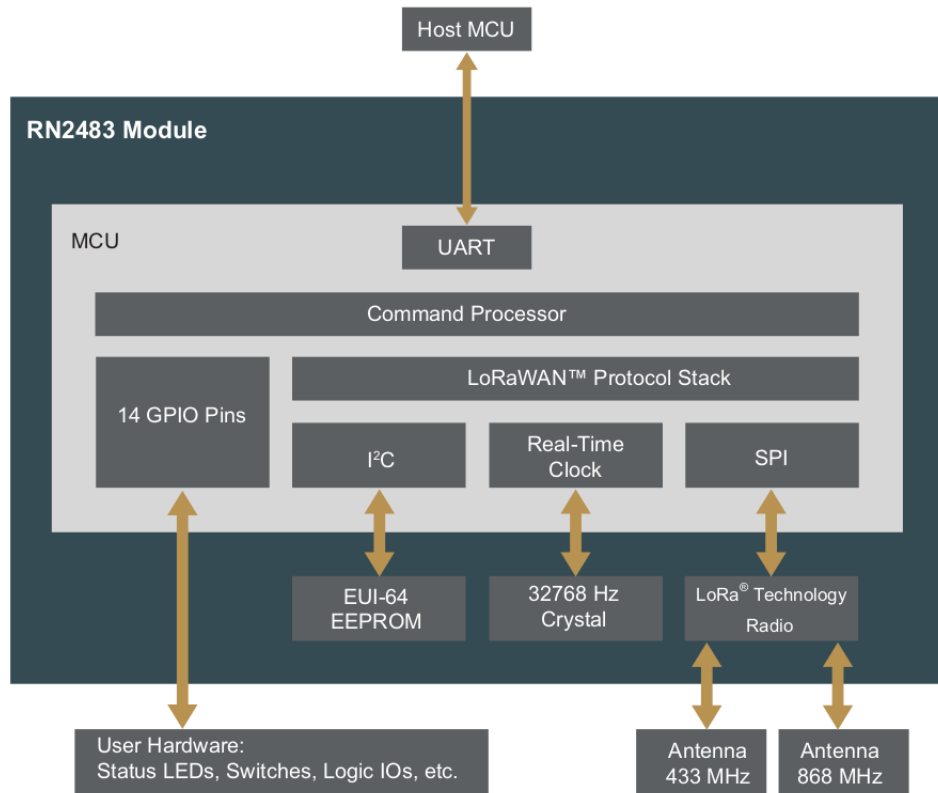


FIGURE 3 – Schéma-bloc du RN2483 [8].

La table 2 reprend la consommation électrique du RN2483 en fonction de son mode de fonctionnement.

| Mode | Courant (mA) | | |
|----------|--------------|------------|------------|
| | VDD = 2.1V | VDD = 3.3V | VDD = 3.6V |
| Idle | 1.7 | 2.8 | 3.1 |
| Transmit | 28.6 | 38.9 | 44.5 |
| Sleep | 0.0015 | 0.0016 | 0.0016 |
| Receive | 12.96 | 14.22 | 14.69 |

TABLE 2 – Consommation de courant (à 25 °C) [8].

Raspberry Pi

Le Raspberri Pi est un ordinateur monocarte. Le modèle utilisé pour ce projet est un Raspberry Pi 3 modèle B+ (Fig. 4). La table 3 reprend les

principales caractéristiques de ce modèle.

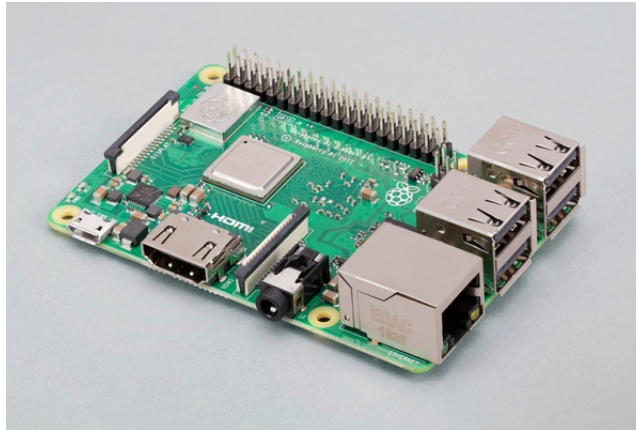


FIGURE 4 – Raspberry Pi 3B+ [9].

| Element | Spécification |
|--------------|--|
| CPU | Broadcom BCM2837B0, Cortex-A53 64-bit SoC à 1.4GHz |
| Mémoire | 1GB LPDDR2 SDRAM |
| Connectivité | <ul style="list-style-type: none">• IEEE 802.11.b/g/n/ac, Bluetooth 4.2, BLE• Gigabit Ethernet over USB 2.0• 4 × USB 2.0 ports |
| Alimentation | 5V/2.5A DC |

TABLE 3 – Spécifications du Raspberry Pi 3B+ [9].

0.2 RTOS

Un RTOS (Real Time Operating System) est un système 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 algorithmes 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 contrôlé par UART.

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

Contiki OS

Le développement public de Contiki a débuté en octobre 2012¹. Dans un premier repository : **contiki** [2]. Un nouveau développement a démarré en mai 2017 sous le nom de **contiki-ng** [3]. C'est donc ce dernier qui est utilisé pour la comparaison est qui est dénommé par la suite "Contiki".

Cet OS 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 [4] qui abstrait la gestion de la programmation événementiel par des protothread dont l'utilisation est similaire aux threads. L'ordonnanceur de cette librairie est coopératif, c'est à dire qu'il ne va jamais forcer un changement de contexte d'un processus à un autre. 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 nœuds utilisant Contiki.

FreeRTOS

D'après le site officiel de FreeRTOS [5], ce RTOS est développé depuis 15 ans. TODO compatible avec zolertia ? **TODO: fff**

1. Date de création du repository Github.

RIOT OS

Le développement public de RIOT a débuté en décembre 2012¹.

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

La pile réseau de RIOT comporte les protocoles notamment 6LoWPAN, IPv6, RPL, LoRaWan, 802.15.4. TODO openwsn

Zephyr

TODO pas supporté tiny os plus de commit depuis 1 an. stack réseau ? supporté ?

Le table 4 résume la comparaison de ces RTOS.

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

| RTOS | 802.15.4 | ord. TSCH | LoRa | IPv6 | routage IP | comp. |
|-------------------|----------|-------------------|---------------|------|------------|-------|
| Contiki OS | ✓ | 6Tisch, Orchestra | Projet KRATOS | ✓ | RPL | ✓ |
| FreeRTOS | × | × | ✓ | ✓ | × | × |
| RIOT OS | ✓ | × | ✓ | ✓ | RPL | ✓ |
| Zephyr | ✓ | × | ✓ | ✓ | Thread | × |

TABLE 4 – Comparatif de différents RTOS.

TODO supprimer colonne LoRa mais citer kratos ?

0.3 RPL

Routing Protocol for Low-Power and Lossy Networks [1] 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. 5). De part cette architecture RPL est adapté aux applications 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.

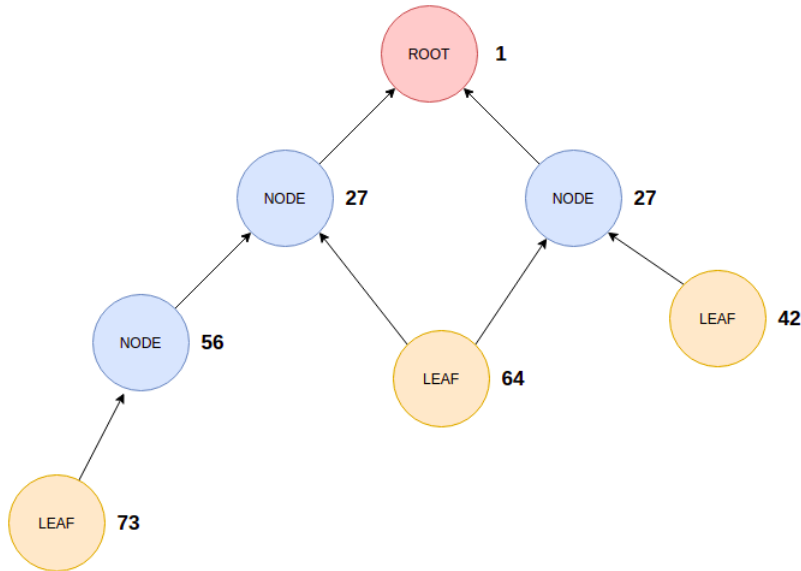


FIGURE 5 – DODAG.

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 strictement dans le sens descendant et diminue strictement dans le sens montant. Ainsi, pour un noeud n , $rang(n) > rang(parent(n))$. La figure 5 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 décrit brièvement deux fonctions objectif implémentées dans Contiki : OF0 et MRHOF.

• OF0

Objective Function Zero [11] 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 :

$$rank_increase = (Rf * Sp + Sr) * MinHopRankIncrease$$

$$R(N) = R(P) + rank_increase$$

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 [6] 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 à 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 utilisée 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 6, 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 0.3 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

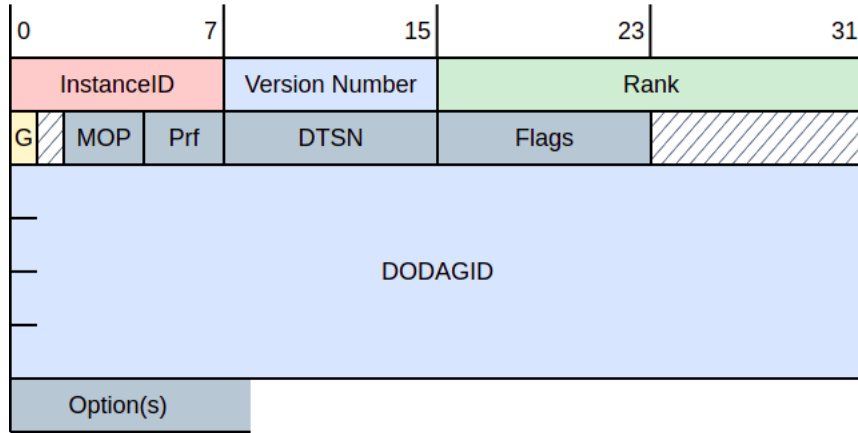


FIGURE 6 – DODAG Information Object.

Les DIOs sont transmis en utilisant l'algorithme Trickle définis dans [7].

DIS

Les DODAG Information Solocitation (DIS) sont utilisés pour solliciter 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 7. 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)

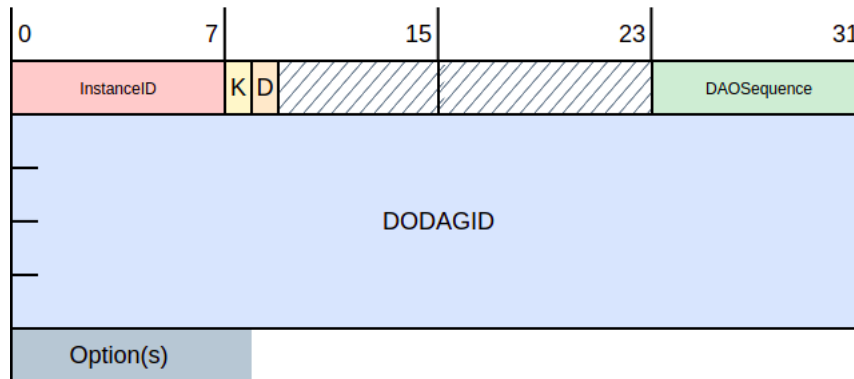


FIGURE 7 – DODAG Information Object.

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 8 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'extérieur 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 diffère selon le mode de fonctionnement (MOP) du réseau. Ces différences sont détaillées en section 0.3 Modes de fonctionnements.

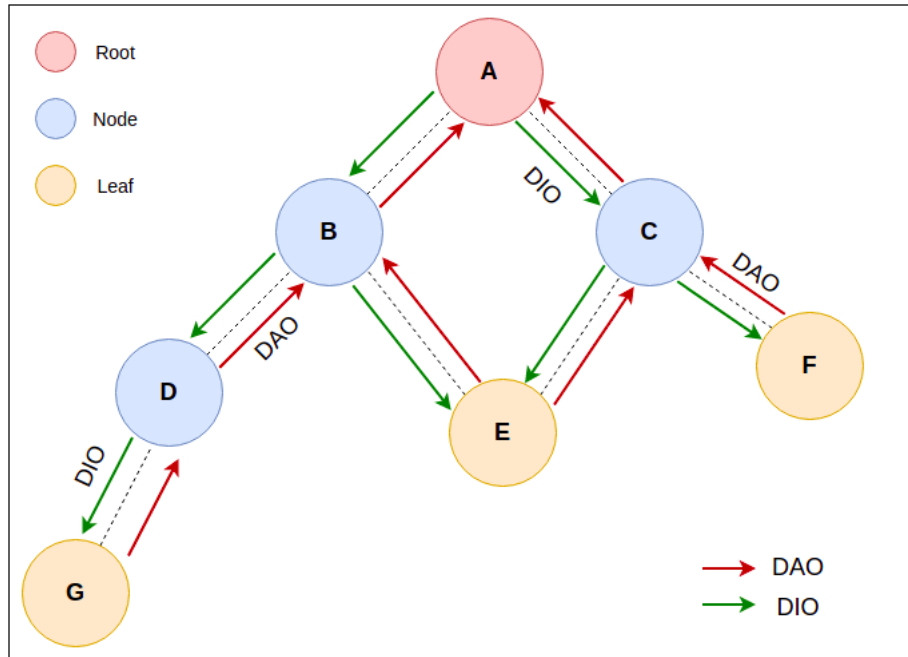


FIGURE 8 – Propagation des DIOs et DAOs.

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. 9a).

- **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. 9b). Les noeuds stockent les routes de leur sous DODAG. Ainsi, chaque saut dans un chemin examine sa table de routage pour choisir le saut suivant.

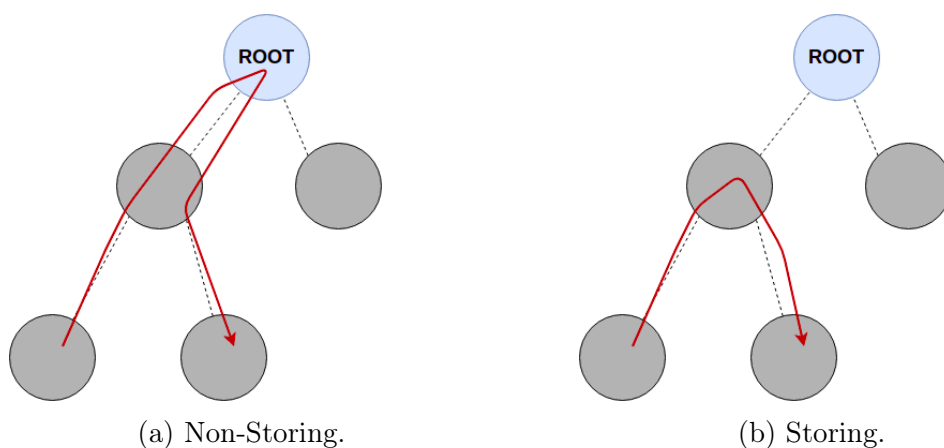


FIGURE 9 – Chemin d'un paquet en fonction du MOP.

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[12], 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.

Bibliographie

- [1] Roger Alexander, Anders Brandt, JP Vasseur, Jonathan Huii, Kris Pister, Pascal Thubert, P Levis, Rene Struik, Richard Kelsey, and Tim Winter. RPL : IPv6 Routing Protocol for Low-Power and Lossy Networks. RFC 6550, March 2012.
- [2] The Contiki Operating System. <https://github.com/contiki-os/contiki>. [Accès en ligne le 15 juin 2021].
- [3] Contiki-NG : The OS for Next Generation IoT Devices. <https://github.com/contiki-ng/contiki-ng>. [Accès en ligne le 15 juin 2021].
- [4] Adam Dunkels, Oliver Schmidt, Thiemo Voigt, and Muneeb Ali. Protothreads : Simplifying event-driven programming of memory-constrained embedded systems. In *Proceedings of the 4th International Conference on Embedded Networked Sensor Systems*, SenSys '06, page 29–42, New York, NY, USA, 2006. Association for Computing Machinery.
- [5] FreeRTOS Real-time operating system for microcontrollers. <https://freertos.org/index.html>. [Accès en ligne le 15 juin 2021].
- [6] Omprakash Gnawali and P Levis. The Minimum Rank with Hysteresis Objective Function. RFC 6719, September 2012.
- [7] P Levis, Thomas H. Clausen, Omprakash Gnawali, Jonathan Huii, and JeongGil Ko. The Trickle Algorithm. RFC 6206, March 2011.
- [8] Microchip. *Low-Power Long Range LoRa (®) Technology Transceiver Module*, 6 2020. Rev. E.
- [9] Raspberry Pi 3 Model B+. <https://www.raspberrypi.org/products/raspberry-pi-3-model-b-plus/>. [Accès en ligne le 15 juin 2021].
- [10] RN2483. <https://www.microchip.com/wwwproducts/en/RN2483>. [Accès en ligne le 14 juin 2021].
- [11] Pascal Thubert. Objective Function Zero for the Routing Protocol for Low-Power and Lossy Networks (RPL). RFC 6552, March 2012.
- [12] Joydeep Tripathi, Jose Oliveira, and J.-P Vasseur. A performance evaluation study of rpl : Routing protocol for low power and lossy networks. pages 1–6, 03 2010.

- [13] Zolertia. *Zolertia RE-Mote Revision B Internet of Things hardware development platform, for 2.4-GHz and 863-950MHz IEEE 802.15.4, 6LoWPAN and ZigBee® Applications*, 9 2016. v.1.0.0.
- [14] Zolertia RE-Mote. <https://zolertia.io/product/re-mote/>. [Accès en ligne le 14 juin 2021].