Pourquoi RPL est un échec et comment faire autrement ?

Henry-Joseph Audéoud, Martin Heusse

Laboratoire d'Informatique de Grenoble, UMR 5217

Introduction — Routage

- Une problématique connue : réseau à sauts multiples
- Des solutions disponibles
  - Réseaux « classiques » (RIP, OSPF, EIGRP, BGP…)
  - MANets (AODV, OLSR…)
- … mais pas adaptées aux exigences des capteurs…

Introduction — Spécificités des capteurs

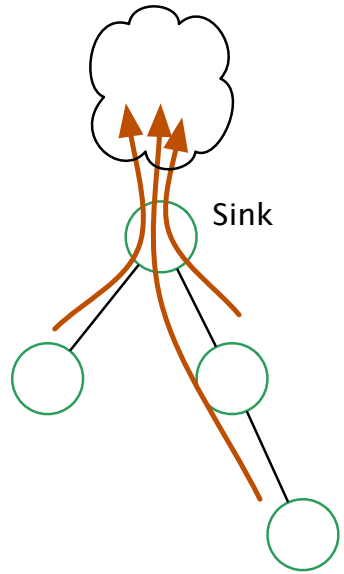
- **Producteurs** de données
  - Les nœuds doivent rester **accessibles**
  - Peu de trafic pair à pair
- **Énergie** limitée (ou **puissance** limitée)
  - **Émission** & **réception** d'un paquet coûteuse
    (la radio pèse lourd dans le budget énergétique)
  - **Diffusion** encore plus
    (avec ContikiMAC, 9× plus qu'un *unicast* !)

RPL — Routing Protocol for Low-Power and Lossy Networks

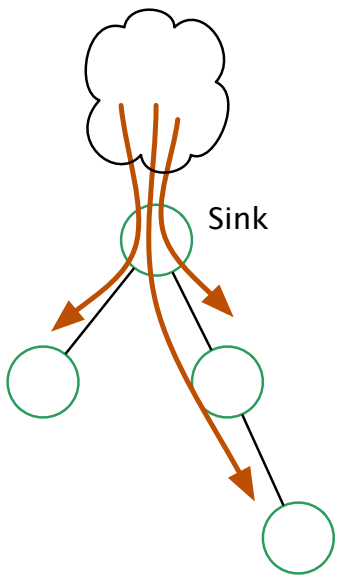
- Standard : IETF RFC 6550
- Ce n'est pas la première critique
  - ✘ Standard complexe (159 pages, 4 RFC additionnelles)
  - ✘ Implémentation lourde (mécanismes complexes, empreinte mémoire importante)
  - ✘ Coexistence de plusieurs instances
  - ✘ Volume (taille et fréquence) des messages de routage
  - ✘ ...

RPL — Collecte

- Structure en **DODAG**
  - Ensemble de routes par défaut
  - ✔ **Collecte** des données efficace
  - Bellman-Ford distribué
  - Vecteur de distance & numéro de séquence
- **Réparation** du DODAG
  - Locale : *poisoning* puis reconnexion locale
    - ✔ Comptage à l'infini limité
    - ✘ Et après ?...
  - Globale : reconstruction complète
    - Initiée par le puits (numéro de séquence)
    - ✘ C'est les autres nœuds qui en ont besoin !



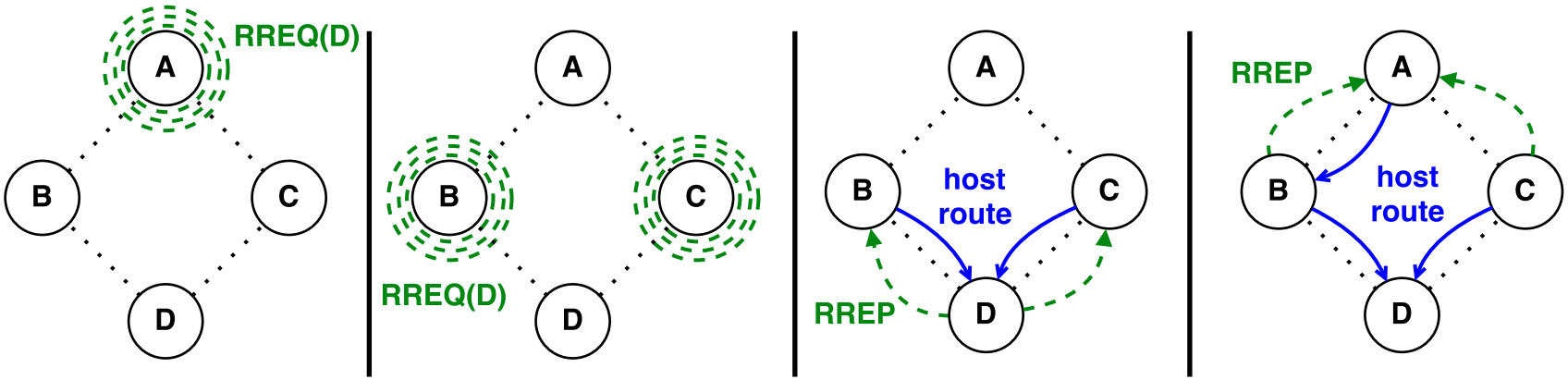
RPL — Distribution & routage



- **Distribution** du trafic
  - **Routes d'hôtes**, descendantes
  - Construction initiée par les hôtes
  - ✘ C'est le puits qui en a besoin !
- Suppression des **boucles de routage**
  - ✘ Information embarqué **dans chaque paquet de donnée** (option IPv6 *Hop-by-Hop*)
  - ✘ Peut nécessiter un tunnel IPv6-sur-IPv6 !
    - Au mieux, 48 octets de surcharge
    - Petits MTUs (802.15.4 : 127o ; BLE : 254o…)

LOADng — Routage réactif

- AODV simplifié
- Protocole réactif :
  - ✔ Création des routes **à la demande**



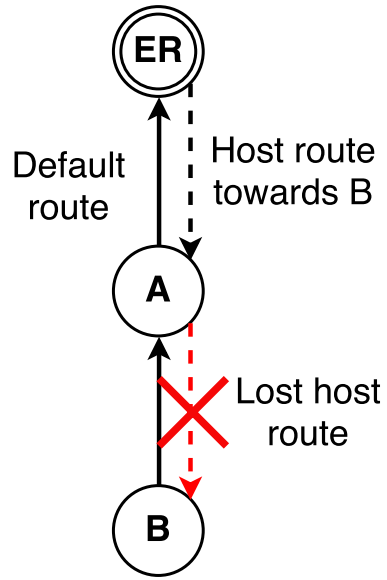
- L'essentiel du trafic est sortant
  - ✘ Pas prévu pour la **collecte** de données

✘ N'importe quel nœud peut provoquer une inondation.

LRP — The Lightweight Routing Protocol

- LRP = LOADng + DODAG
  - ✔ **Collecte** aisée
  - ✔ **Recherche** d'un hôte possible
  - ✔ **Silencieux** : peu de trafic dû au routage

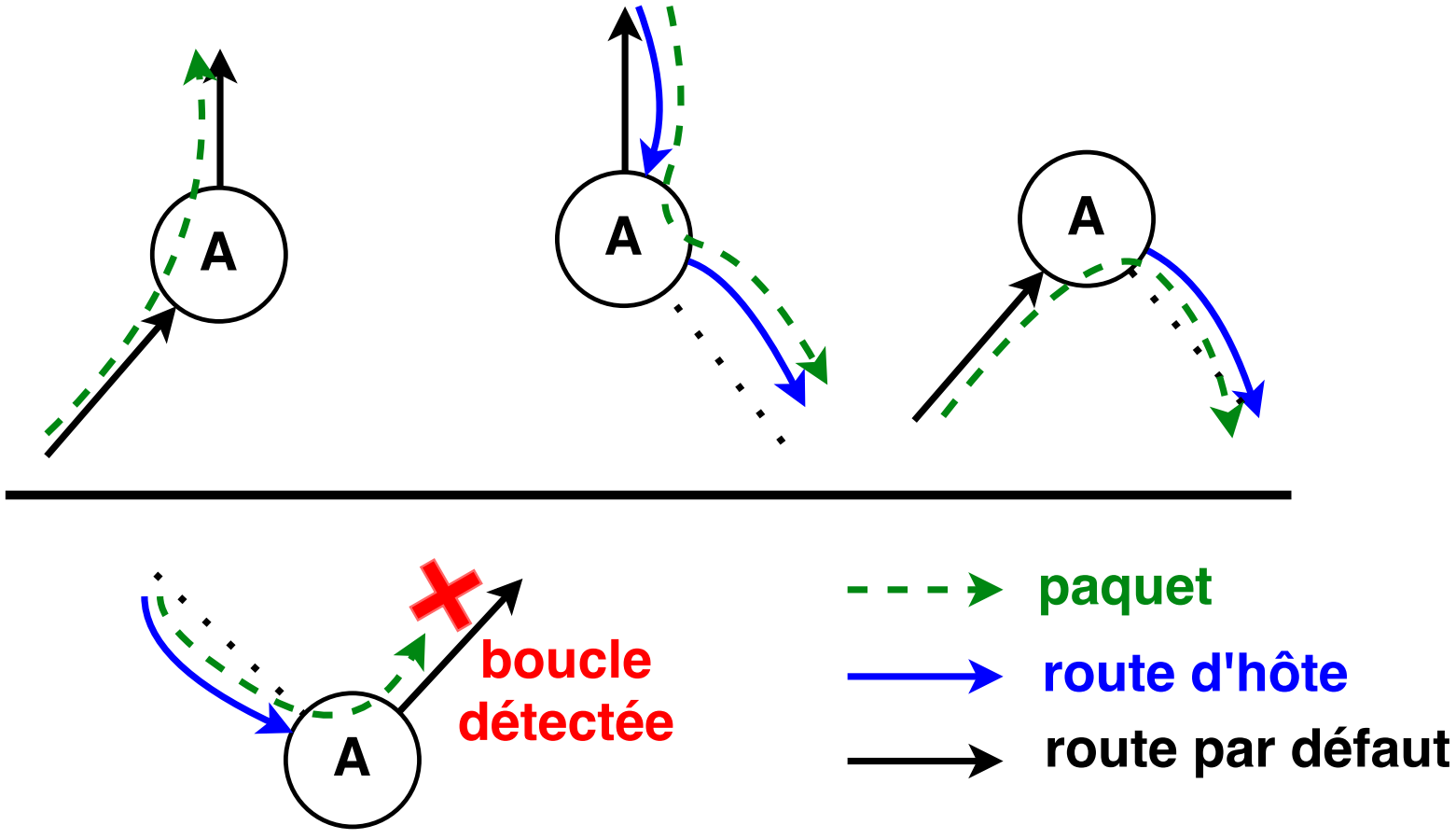
- Coexistence de routes par défaut et d'hôtes
  - ✘ Attention aux boucles de routage !
- DODAG
  - ✘ Comment réparer ?



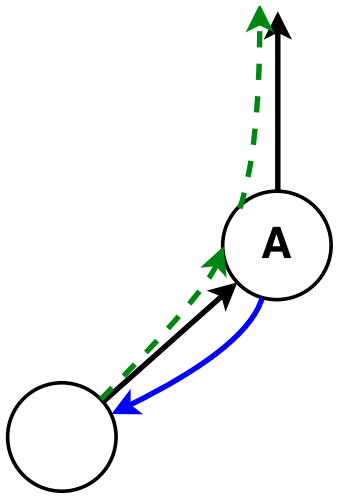
LRP — Boucles de routage

- Suppression des boucles de routage
  - Vérification **au moment du routage**
  - ✔ Pas d'ajout d'information dans les paquets IP.
  - Fonctionne grâce à une relation d'ordre sur les routes
    - Routes plus précises (longueur de préfixe)
    - Routes plus récentes (numéro de séquence)
    - Plus proche de la destination (métrique)

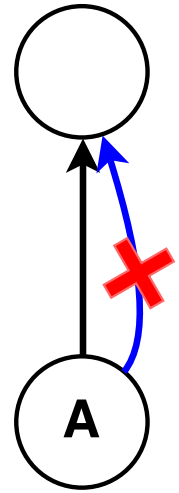
LRP — Boucles de routage (cont.)



LRP — Boucles de routage (cont.)



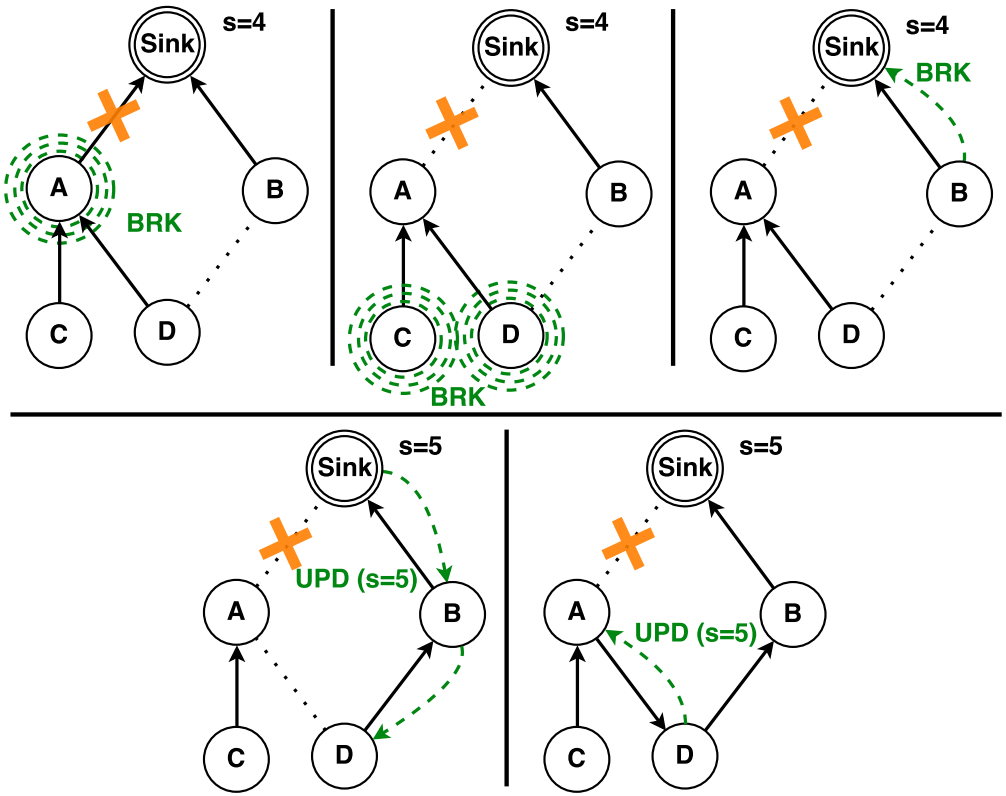
Identifier les prédécesseurs



Pas d'autre route en parallèle de la route par défaut

LRP — Réparation locale

- Locale : renversement des liens
  - ✘ Arbre pas forcément optimal
  - ✔ Jamais de boucles



LRP — Réparation locale (cont.)

- Globale : reconstruction complète du DODAG
  - Nouveau numéro de séquence
  - Initié par le puits
  - Déclenché après un certain nombre de réparation locale

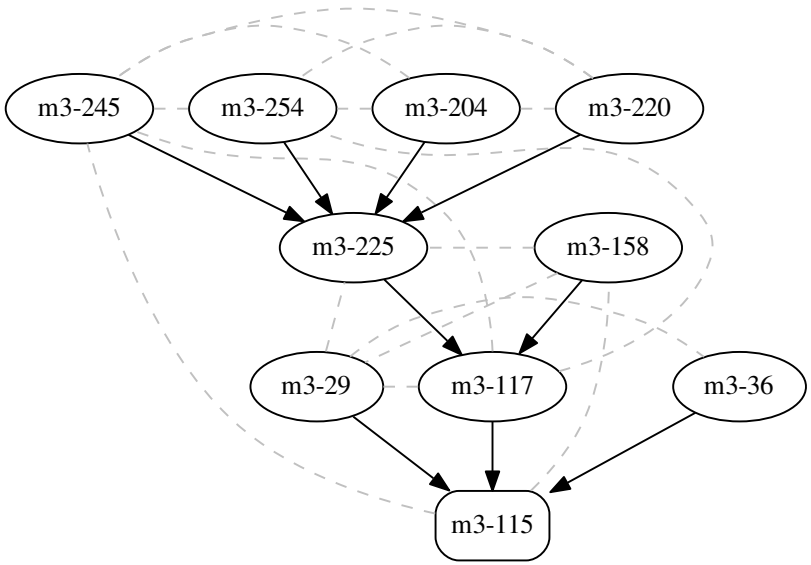
RPL, LRP — quelles différences ?

	RPL	LRP
Type de protocole	À vecteur de distance	
Collecte de données	Arbre (DODAG) de collecte	
Réparation de l'arbre	*poisoning*	Renversement des liens
Routes d'hôtes	proactif	proactif *et*/*ou* réactif
Validation des trajets	Option *Hop-by-Hop* IPv6	Vérification au moment du routage
Diffusion de messages	Trickle	Libre
Empreinte mémoire	Élevée (7,6ko = 10% NVRAM, 476o = 3,3% RAM)	Restreinte (1,9ko = 2,6% NVRAM 304o = 2,2% RAM)
Interopérabilité	Beaucoup de variantes incompatibles	À la carte

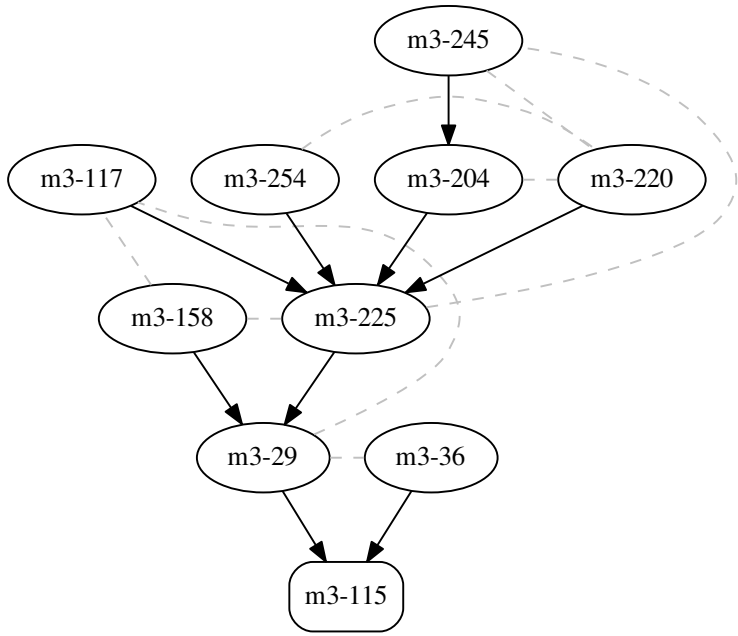
Expériences

- Plateforme FIT IoT-lab de Lille
- ContikiOS
- Nœuds : *M3 open nodes*
- Simulations
  - 30 minutes
  - 9 nœuds + 1 puits
  - 2 messages par nœud et par minute
- Réduction de la puissance d'émission
  - ✔ Réseau à sauts multiples
  - ✘ Beaucoup de pertes de paquets

Expériences — DODAG et routes d'hôtes

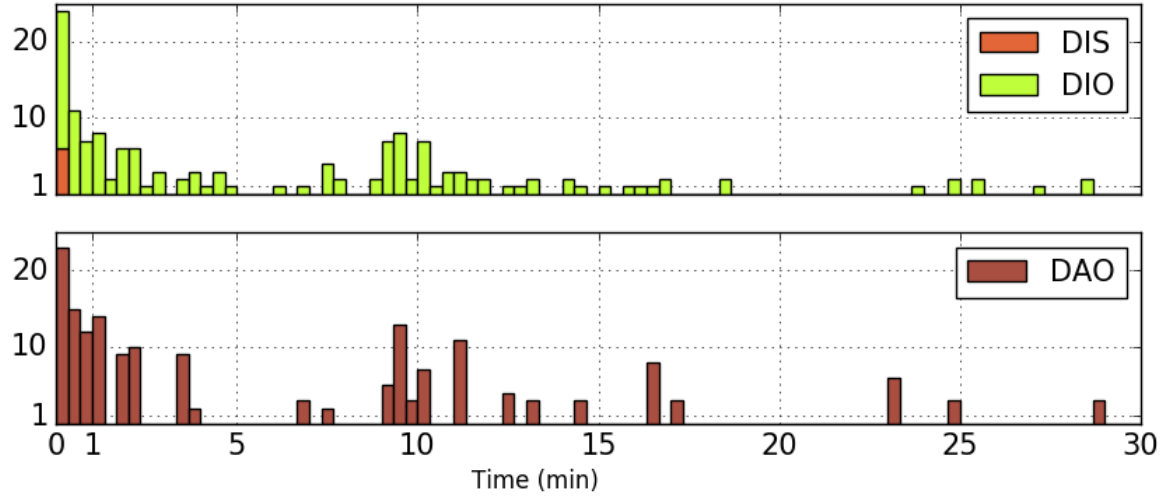


RPL

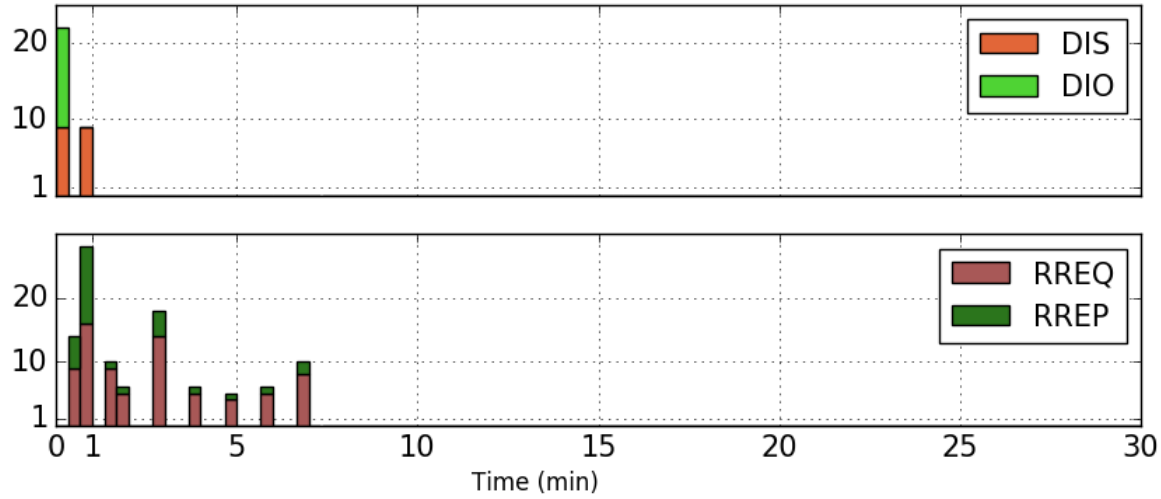


LRP

Expériences — DODAG et routes d'hôtes (cont.)



RPL



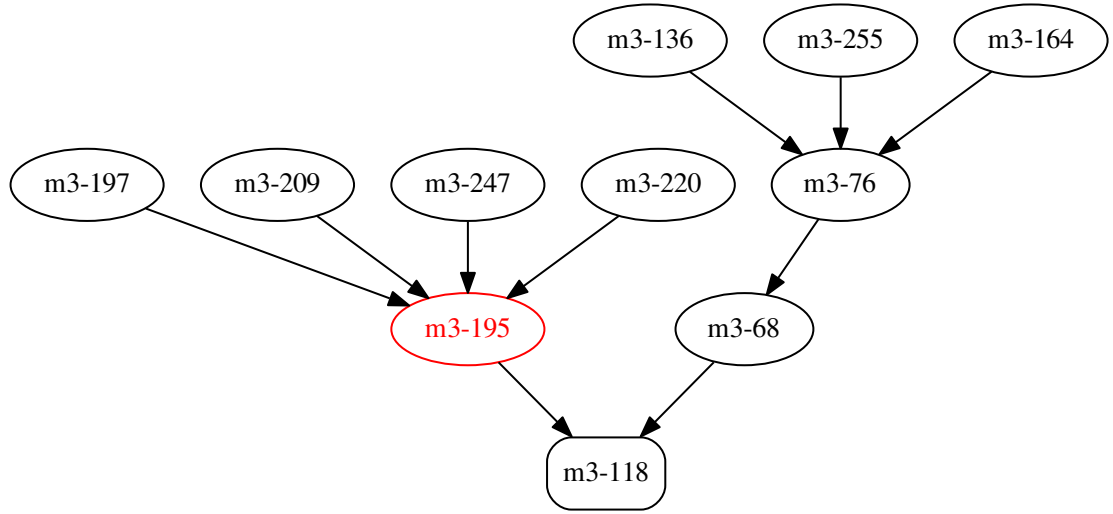
LRP

Expériences — DODAG et routes d'hôtes (cont.)

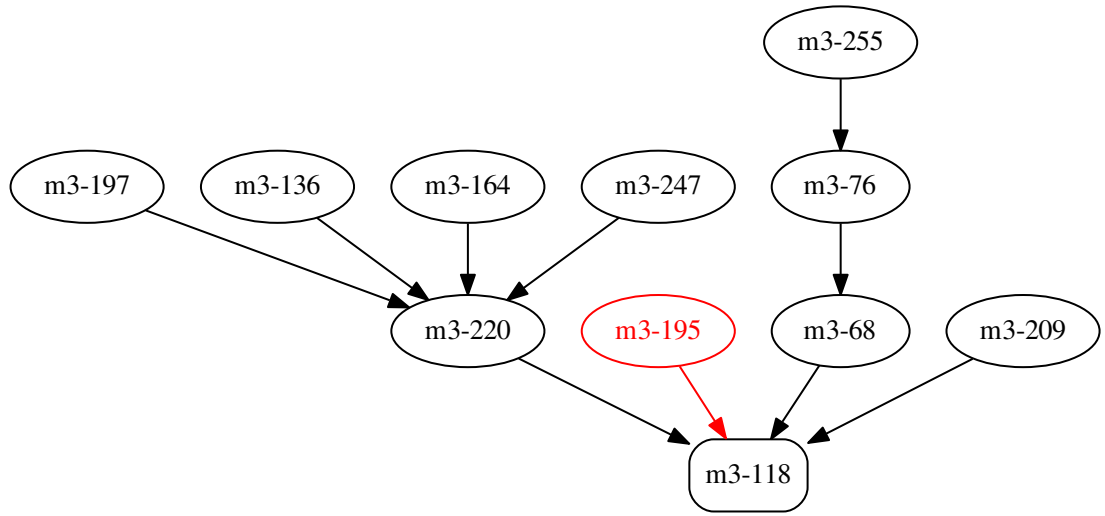
- RPL
  - 146 broadcast
    - 146 DIS & DIO
  - 171 unicast (DAO)
  - 431/495 messages (13% de pertes)
- LRP
  - 106 broadcast
    - 31 DIS & DIO
    - 75 RREQ
  - 33 unicast (RREP)
  - 480/524 messages (9% de pertes)

Expériences — Réparation locale

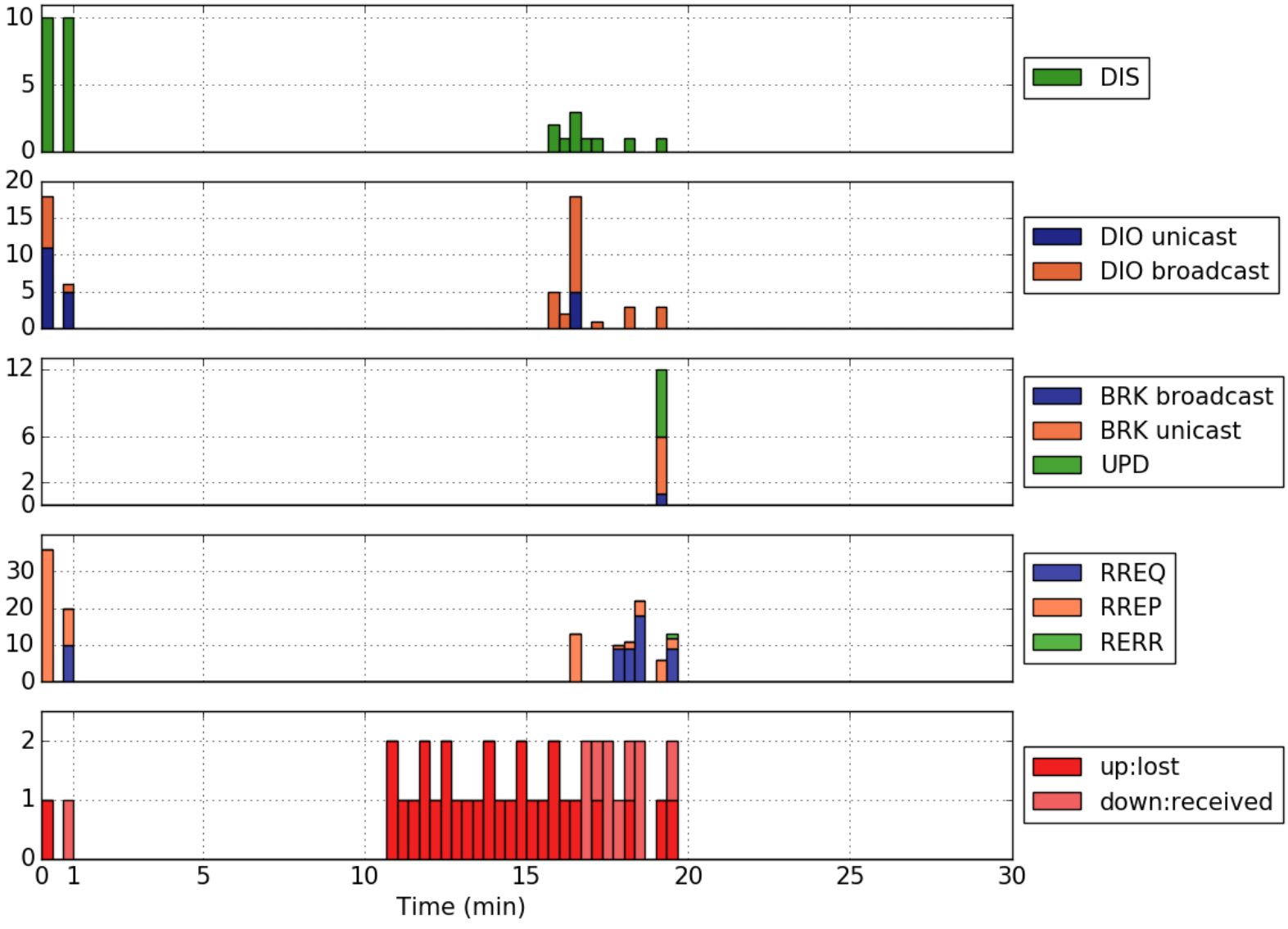
Avant :



Après :



Expériences — Réparation locale (cont.)



Conclusion

Moins coûteux et plus performant que RPL,
c'est possible !

- ✔ Réparation du DODAG à l'initiative des hôtes
  - et en plus, sans boucles de routage
- ✔ Recherche d'un hôte possible
  - Peu de surcharge dans la plupart des cas
- ✔ Pas de boucle de routage
  - Et ce, sans surcharger le paquet routé !
- ✔ Fonctionnement silencieux

Perspectives

- LRP est extensible :
  - Optimiser la diffusion (MPR, BFT, Trickle...)
  - Utiliser *expanding ring search* lors de la réparation locale
  - Protection contre les liens asymétriques
  - Autoriser les routes de longueurs de préfixe variées

Sources de LRP sur contiki disponibles à :

`https://github.com/drakkar-lig/contiki/tree/lrp`

Merci pour votre écoute !