Amélioration du passage à l'échelle de LoRa par une utilisation récursive des démodulateurs

Alexandre Guitton (LIMOS, France) et Megumi Kaneko (NII, Japon)

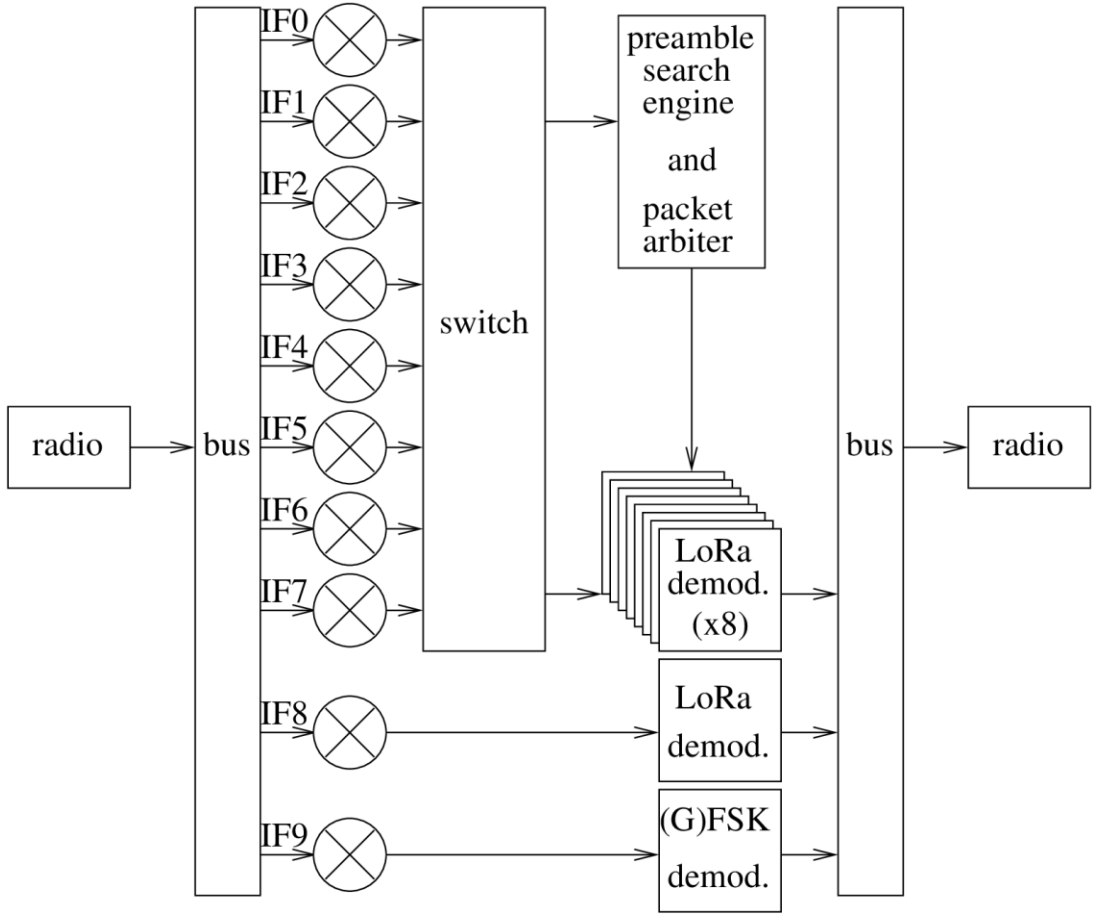
Papier accepté à Globecom 2020

Passage à l'échelle dans LoRa

- LoRa est une technologie de communication sans fil longue distance à basse consommation énergétique
  - La grande portée (quelques kilomètres en environnement urbain) permet de considérer de nombreuses applications innovantes
  - La grande portée conduit à étudier des déploiements avec beaucoup de nœuds
  - Mais le débit dans LoRa est très limité
- LoRa utilise un paramètre nommé *spreading factor* (SF)
  - Plus le SF augmente, plus la portée augmente, mais plus le débit baisse

Démodulateurs

- La modulation LoRa permet à plusieurs trames d'être démodulées en parallèle…
  - … si les trames sont envoyées sur des canaux différents
  - … ou si les trames sont envoyées sur des SF différents
- Les passerelles LoRa utilisent généralement le composant radio SX1301
  - Le SX1301 possède 8 démodulateurs LoRa configurables
  - Cela lui permet de démoduler jusqu'à 8 trames en parallèle



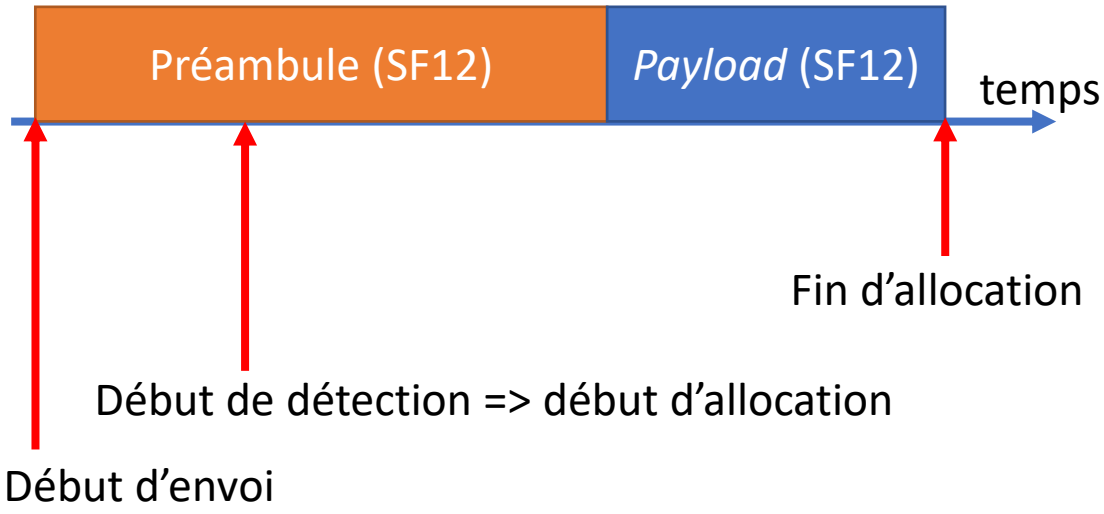
Problématique

- Problème : le nombre limité de démodulateurs contraint les performances d'une passerelle, et donc du réseau
- Problème : la manière dont les démodulateurs sont alloués aux trames a un impact sur les performances d'une passerelle
- Malheureusement, peu de travaux considèrent la limite physique du nombre de démodulateurs [1,2,3]
- Objectif : proposer un algorithme d'allocation des démodulateurs, permettant d'augmenter le débit et l'équité

[1] R.B.Sorensen, J.J.Nielsen, P.Popovski. « Analysis of LoRaWAN uplink with multiple demodulating paths and capture effect », ICC, 2019.
[2] J.C.Liando, A.Gamage, A.W.Tengourtius, M.Li. « Known and unknown facts of LoRa: Experiences from a large-scale measurement study », ACM Transactions on Sensor Networks, 2019.
[3] L.Amichi, M.Kaneko, E.H.Fukuda, N.El Rachkidy, A.Guitton. « Joint allocation strategies of power and spreading factors with imperfect orthogonality in LoRa networks », IEEE Transactions on Communications, 2020.

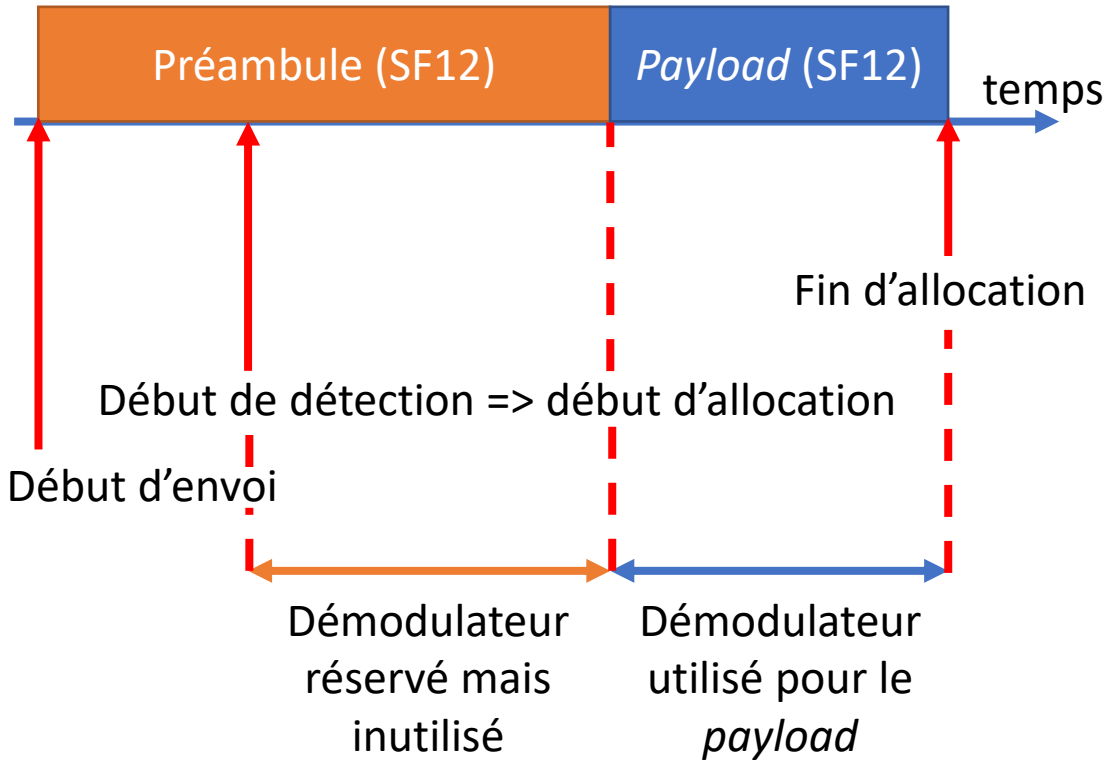
Approche FIFO (1/2)

- Il est difficile de savoir quel est l'algorithme implémenté dans le *packet arbiter* du SX1301 : nous supposerons qu'il s'agit d'un algorithme FIFO
  - = dès qu'un nouveau préambule est détecté, un démodulateur libre est alloué



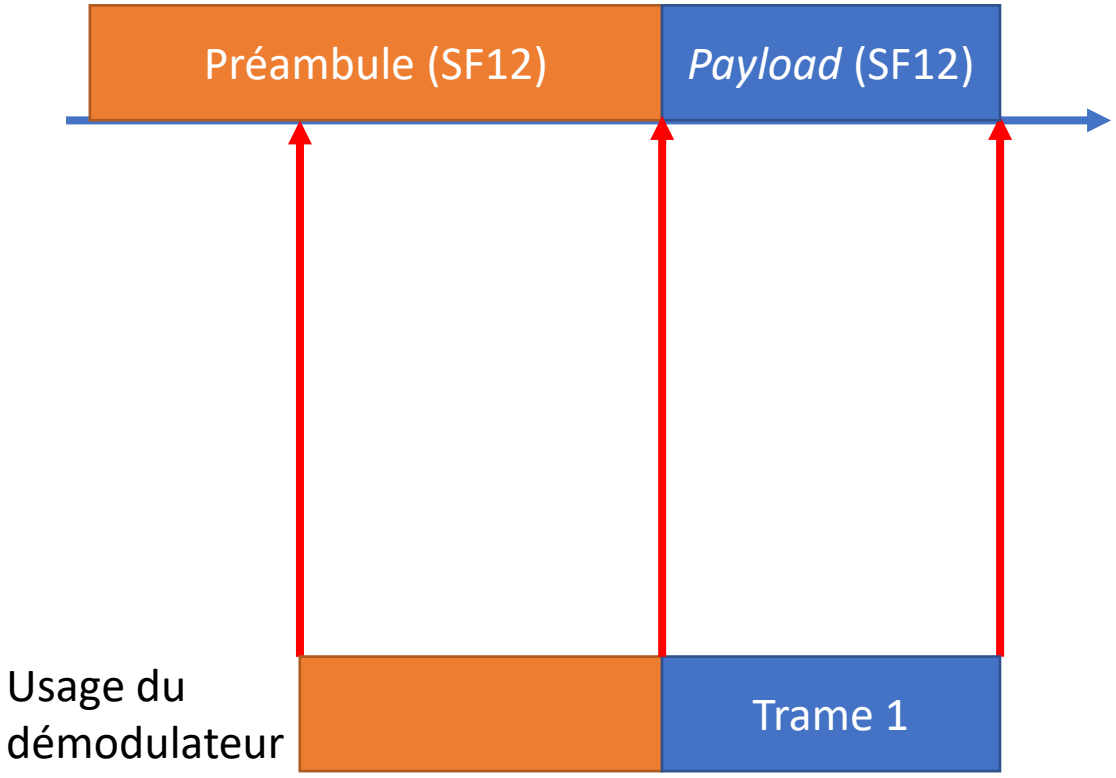
Approche FIFO (2/2)

- Mais, le démodulateur est réservé pendant la fin du préambule, sans être réellement utilisé



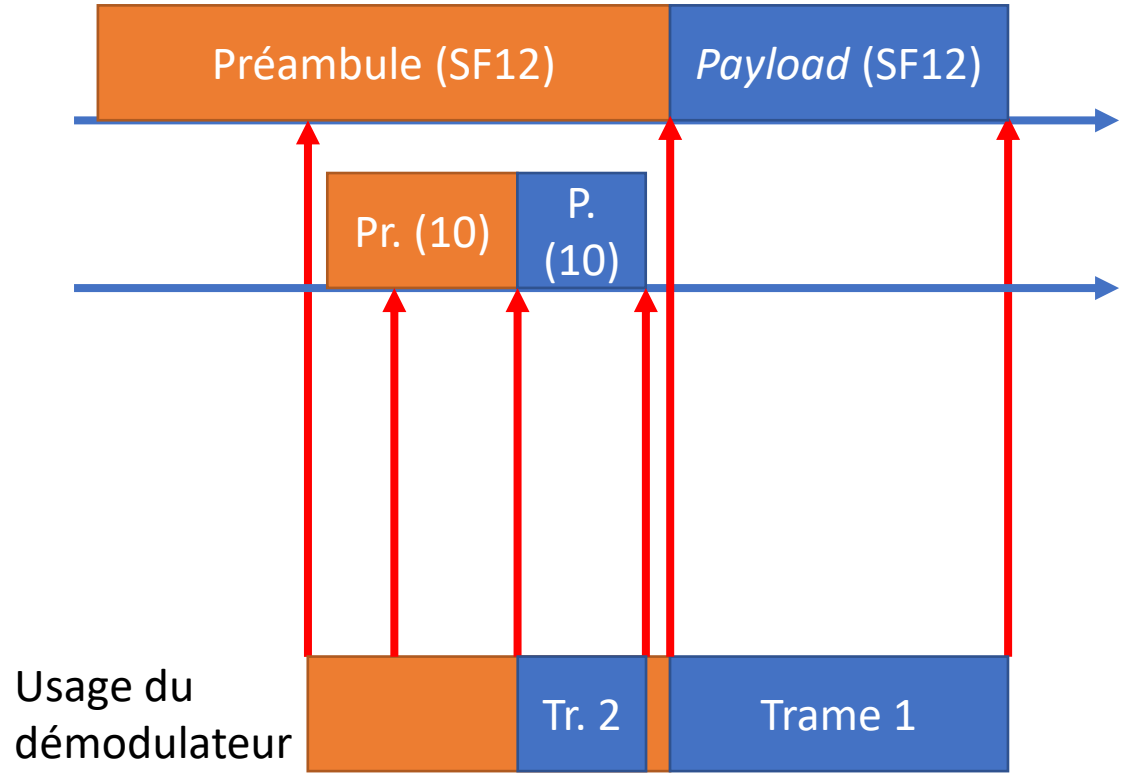
Idée 1 : FIFO-RR1 (1/3)

- Pendant le temps où le démodulateur est réservé entre la détection du préambule et la fin du préambule, on peut démoduler des trames (avec un SF plus petit)…



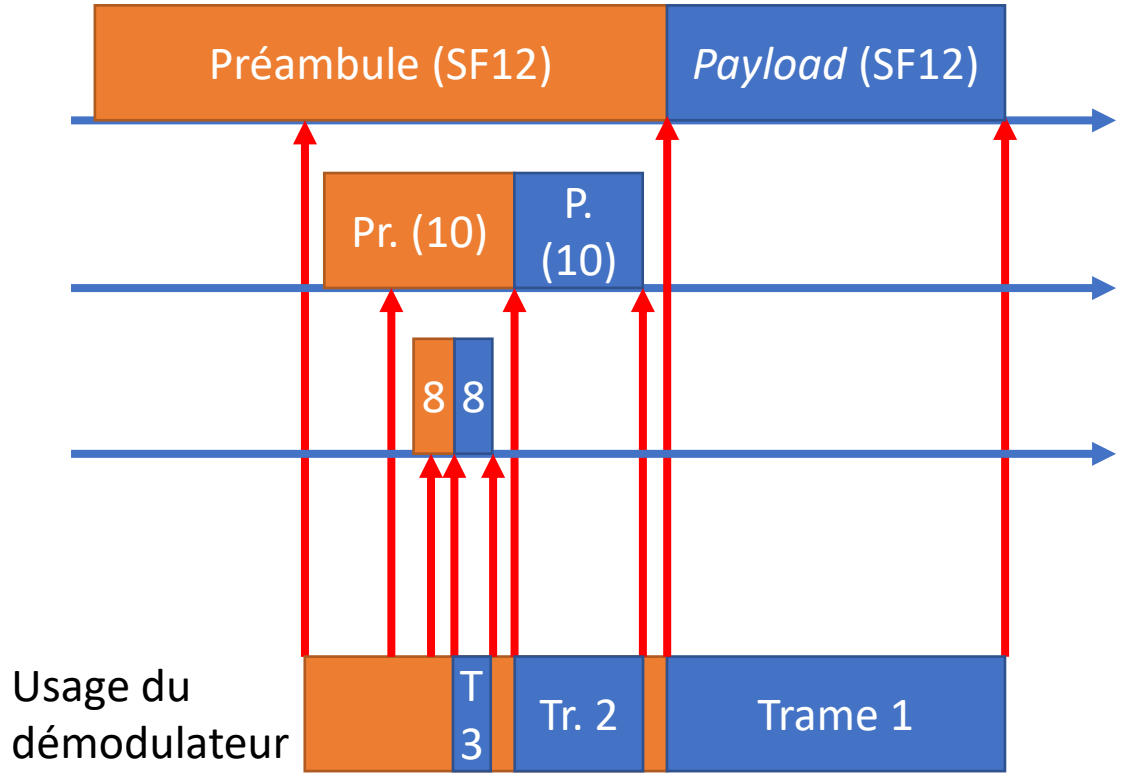
Idée 1 : FIFO-RR1 (2/3)

- Pendant le temps où le démodulateur est réservé entre la détection du préambule et la fin du préambule, on peut démoduler des trames (avec un SF plus petit)…



Idée 1 : FIFO-RR1 (3/3)

- Pendant le temps où le démodulateur est réservé entre la détection du préambule et la fin du préambule, on peut démoduler des trames (avec un SF plus petit)…
  - … récursivement



Algorithmes pour FIFO-RR1

Algorithm 1: Demodulator allocation upon the preamble detection, for FIFO-RR1.

input: a new preamble is detected on SF s at $t_{current}$
$t_{max}(s) \leftarrow$ remaining time on air for the longest frame at SF s
if *there is* d *such that* $state[d] =$ `IDLE` **or** $(state[d] =$ `BOOKED` *and* $next[d] > t_{current} + t_{max})$ **then**
  $t_{preamble} \leftarrow$remaining preamble duration for SF s
  $next[d] \leftarrow t_{current} + t_{preamble}$
  push $next[d]$ on $timeStack[d]$
  push the frame parameters on $frameStack[d]$
  $state[d] \leftarrow$ `BOOKED`
  $demodulator[d] \leftarrow$ parameters of the frame
  start timer for $t_{preamble}$
else
  frame is rejected
end

Algorithm 2: Demodulator reuse after the payload.

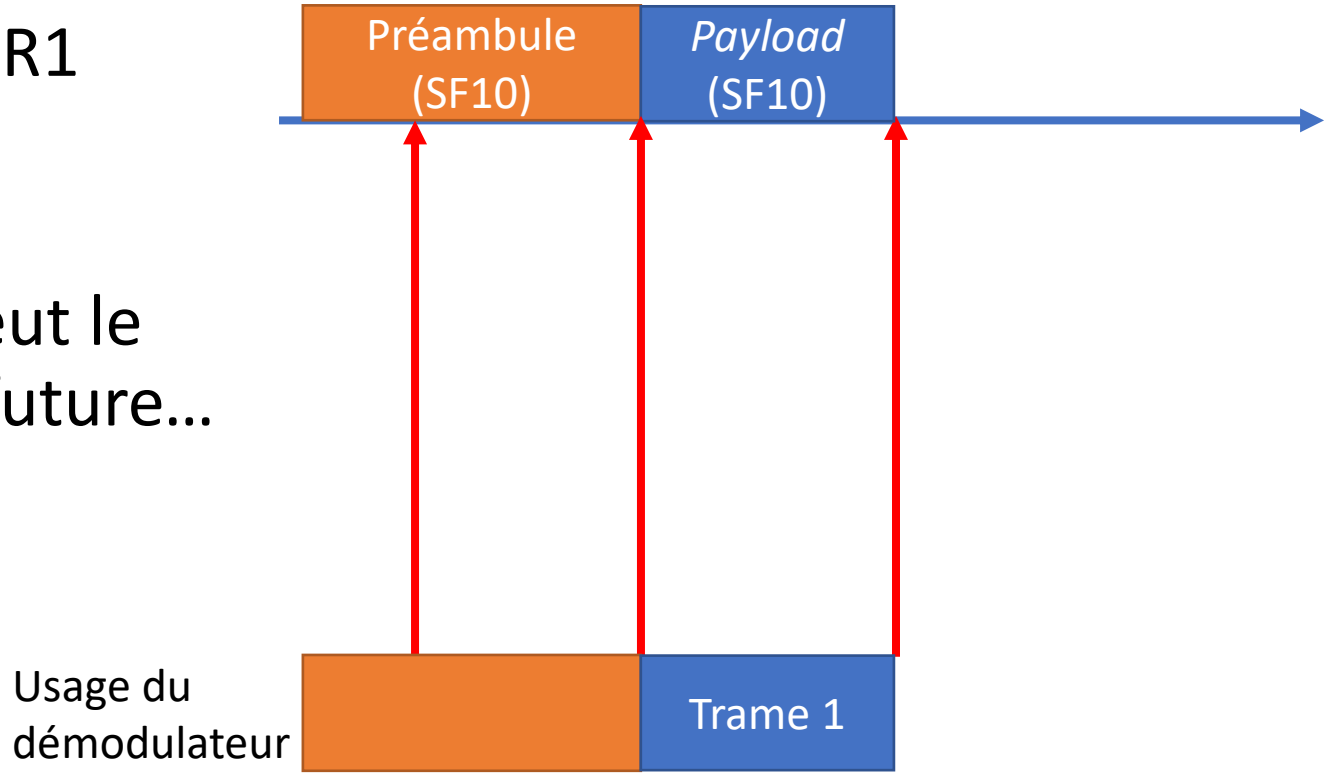
input: A demodulator d finishes demodulating a frame
pop $timeStack[d]$
pop $frameStack[d]$
if $timeStack[d]$ *is empty* **then**
  $state[d] \leftarrow$ `IDLE`
  $demodulator[d] \leftarrow \emptyset$
else
  $state[d] \leftarrow$ `BOOKED`
  $next[d] \leftarrow$ top of $timeStack[d]$
  $demodulator[d] \leftarrow$ top of $frameStack[d]$
  start timer in $next[d] - t_{current}$
end

Idée 1 : Synthèse

- FIFO-RR1 permet d'utiliser les démodulateurs pendant le temps libre entre la détection du préambule et le début du *payload*
  - Donc FIFO-RR1 permet de démoduler plus de trames avec des petits SF (qui sont acceptées pendant l'inactivité liée au préambule des trames avec des grands SF)
  - FIFO-RR1 augmente le débit… mais réduit l'équité entre les SF
- Remarques :
  - Implémentation simple (deux piles et des parcours de tableaux)
  - Utilisation limitée de la mémoire (au plus 90 bits par démodulateur)

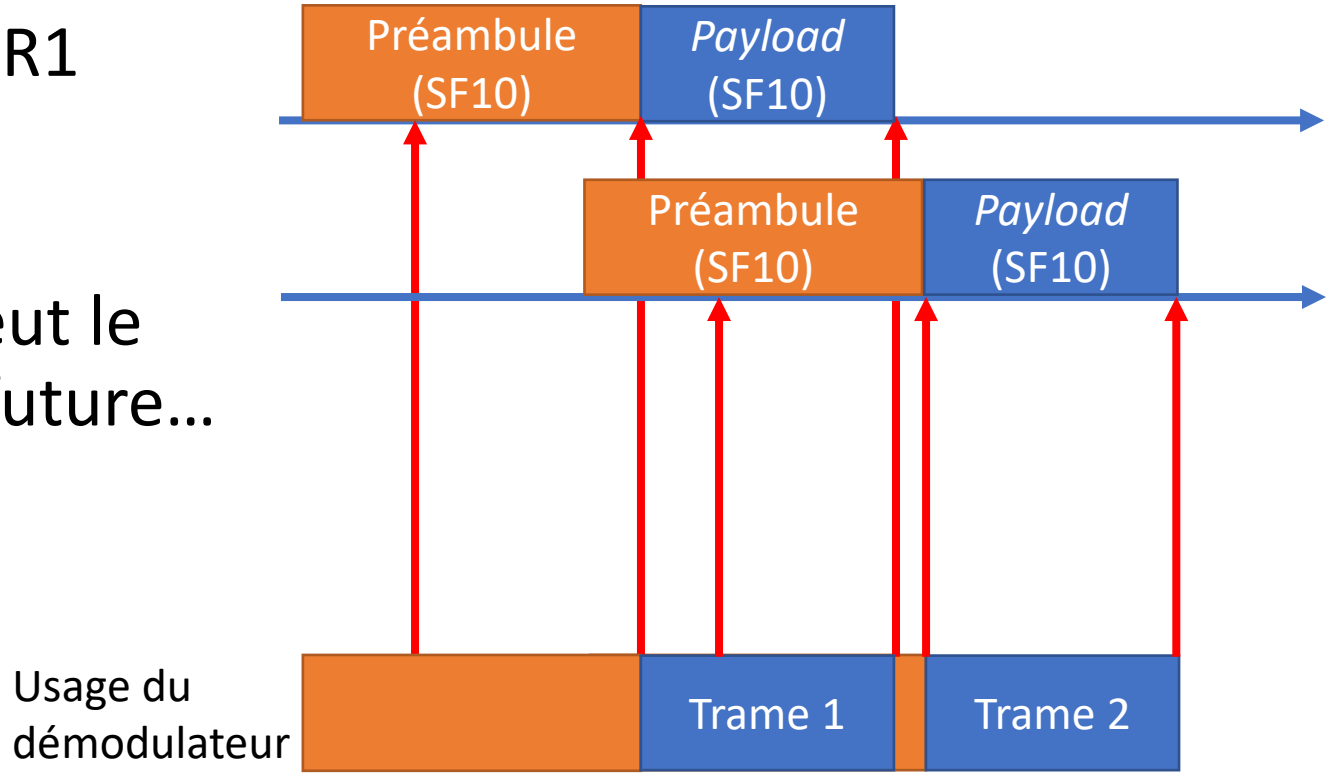
Idée 2 : FIFO-RR2 (1/2)

- En complément de FIFO-RR1
- Pendant le temps où le démodulateur est utilisé pendant le *payload*, on peut le réserver pour une trame future…
  - … récursivement aussi



Idée 2 : FIFO-RR2 (2/2)

- En complément de FIFO-RR1
- Pendant le temps où le démodulateur est utilisé pendant le *payload*, on peut le réserver pour une trame future…
  - … récursivement aussi



Algorithme pour FIFO-RR2

Algorithm 3: Demodulator allocation after the preamble detection, for FIFO-RR2.

input: a new preamble is detected on SF s at $t_{current}$
execute Algorithm 1
if *the frame was rejected by Algorithm 1* **then**
  $t_{preamble} \leftarrow$ remaining duration for the new preamble
  $t_{newPayload} \leftarrow t_{current} + t_{preamble}$
  /* compute the time when each busy demodulator finishes its frame */
  forall *demodulator* d *such that* $state[d] =$ BUSY **do**
    $payloadDuration \leftarrow$payload duration of frame on top of $frameStack[d]$
    $t_{end}[d] \leftarrow next[d] + payloadDuration$
  end
  if *there is* d *such that* $state[d] =$ BUSY *and* $t_{end}[d] \leq t_{newPayload}$ *and size of* $frameStack[d]$ *equals to 1* **then**
    /* insert the new time in the stack */
    $previousTopTime \leftarrow$ pop $timeStack[d]$
    push $t_{newPayload}$ on $timeStack[d]$
    push $previousTopTime$ on $timeStack[d]$
    /* insert the new frame in the stack */
    $previousTopFrame \leftarrow$ pop $frameStack[d]$
    push the new parameters on $frameStack[d]$
    push $previousTopFrame$ on $frameStack[d]$
  else
    frame is rejected
  end
end

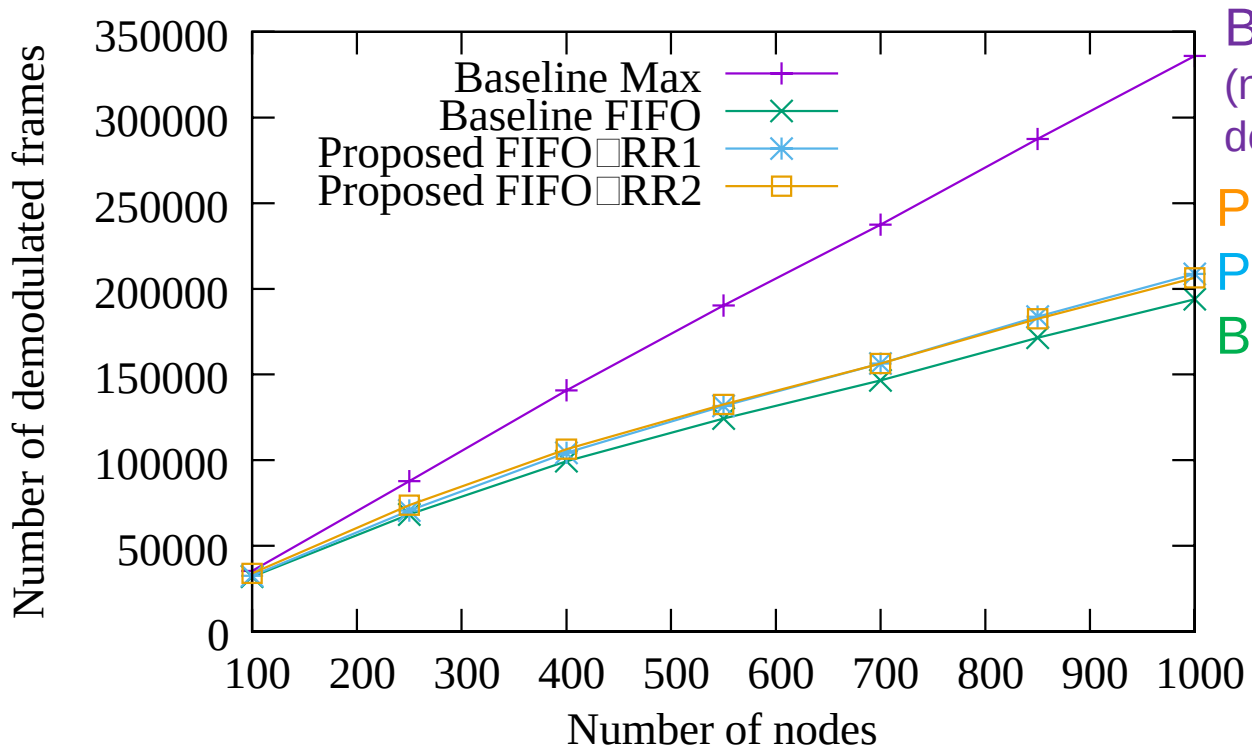
Idée 2 : Synthèse

- FIFO-RR2 permet de réserver des trames futures pendant que le démodulateur est occupé
  - Notamment, FIFO-RR2 réserve des trames éloignées dans le temps, donc avec un grand SF
  - FIFO-RR2 augmente le débit et l'équité entre les SF
- Remarques :
  - Implémentation simple (deux piles et des parcours de tableaux)
  - Utilisation limitée de la mémoire (idem à FIFO-RR1)

Paramètres de simulation

- Génération (en Java) de trames avec la méthode d'accès LoRaWAN, avec un canal, un taux d'activité de 1%, et pendant 10000 secondes
  - Le SF est fixé en fonction de la distance du nœud à la passerelle
  - Les nœuds sont uniformément répartis
  - Hypothèse : le préambule est détecté après 4 symboles (sur 8)
  - 100 répétitions
- Métriques :
  - Nombre de trames démodulées (ce qui avantage les petits SF)
  - Equité (calculée avec l'index de Jain, en fonction du nombre de trames démodulées par SF)

Résultats en terme de débit



Baseline MAX
(nombre infini de démodulateurs)

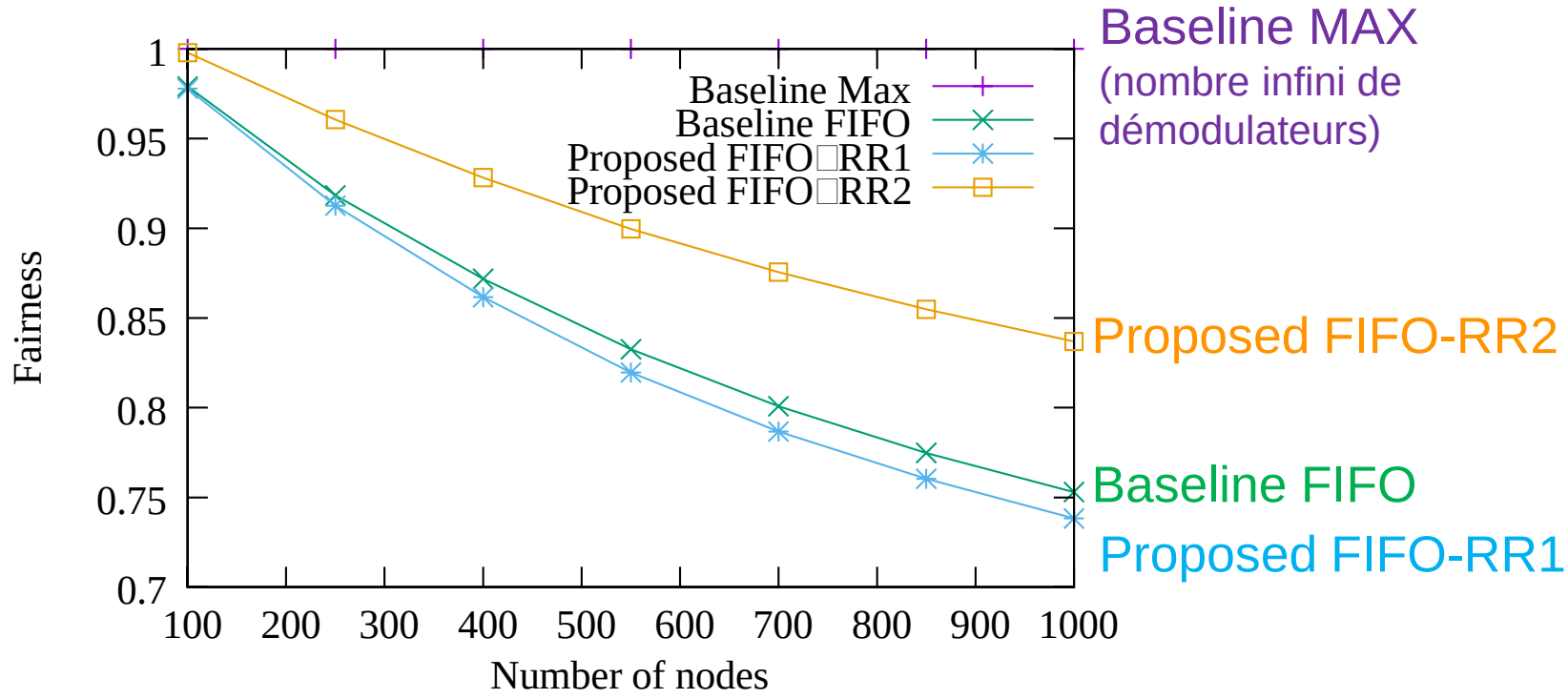
Proposed FIFO-RR2

Proposed FIFO-RR1

Baseline FIFO

- FIFO-RR1 et FIFO-RR2 décodent environ 6,5% de trames de plus que FIFO, pour 1000 nœuds

Résultats en terme d'équité



Baseline MAX
(nombre infini de démodulateurs)

Proposed FIFO-RR2

Baseline FIFO

Proposed FIFO-RR1

- FIFO-RR1 favorise les petits SF, donc est moins équitable que FIFO
- FIFO-RR2 est plus équitable que FIFO, d'environ 11% pour 1000 nœuds

Conclusion

- Le nombre limité de démodulateurs dans les passerelles contraint le trafic LoRa
- FIFO-RR2 améliore le débit et l'équité, avec un surcoût matériel faible (en terme de difficulté d'implémentation et d'occupation mémoire)
  - En terme de débit, FIFO-RR2 à 8 démodulateurs obtient des performances similaires à FIFO avec 12 démodulateurs
  - En terme d'équité, FIFO-RR2 à 8 démodulateurs obtient des performances similaires à FIFO avec 16 démodulateurs

Merci !