Projet "IoT Interconnection" - Simulation 6TiSCH

[redacted] - Édouard LUMET - 3SN IBDIOT 2020

Prise en main

Tout d'abord, nous extrapolons les paramètres présents dans le fichier de configuration (*config.json*) :

- exec_numNotes : nombre de nœuds dans la topologie
- sf_class : utilisation ou non d'une fonction d'ordonnancement
- tsch_slotDuration : la durée d'un slot TSCH (802.15.4), qui permet, à longueur de slotframe fixe, de modifier la durée de la slotframe
- tsch_slotframeLength : la longueur d'une slotframe TSCH, permettant, à durée de slot fixe, de modifier le nombre de slots par slotframe
- tsch_probBcast_ebProb : la probabilité d'envoi d'EB, voir *Observations, Effet des EBs sur le nombre de nœuds dans le réseau*
- conn_class : la topologie RPL
- rpl_of : Fonction Objectif de RPL

Nous avons aussi fait quelques modifications dans le fichier *plot.py* :

- changement de la police et de l'orientation des légendes, en ajoutant le code suivant dans la fonction *plot_box*

```
plt.xticks(fontsize=8, rotation=45, ha='right')
plt.tight_layout()
```

- changement de la résolution des images dans la fonction *savefig* :

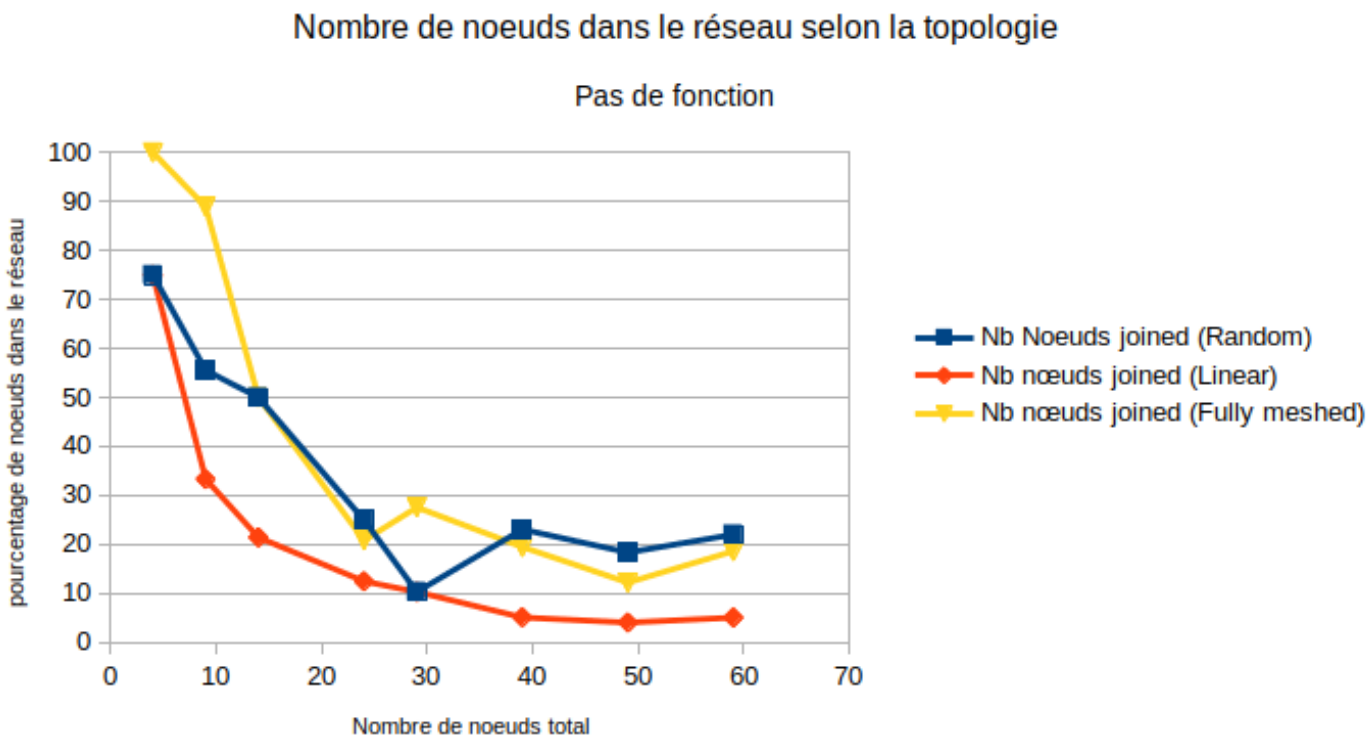
```
plt.savefig(os.path.join(output_folder, output_name + "." + output_format),
      bbox_inches     = 'tight',
      pad_inches      = 0,
      format          = output_format,
      dpi             = 150 //new resolution
  )
```

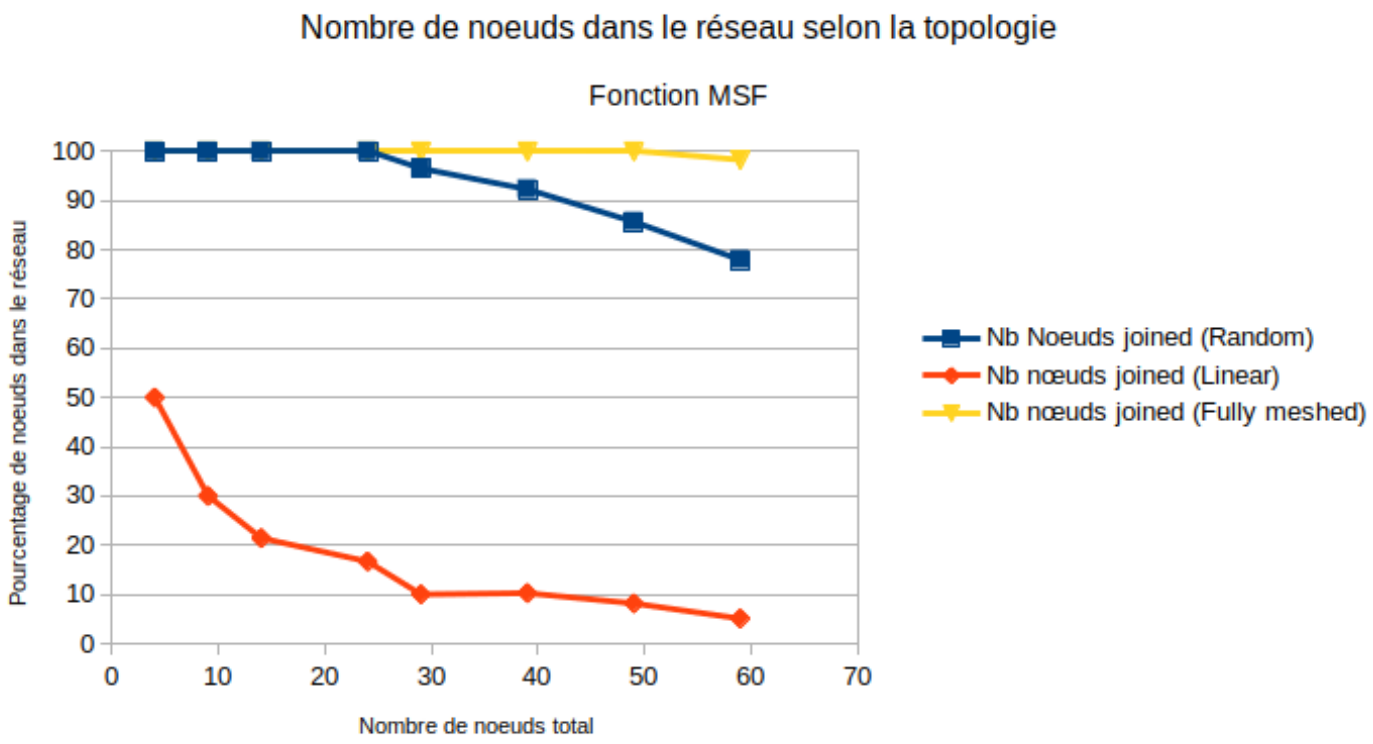
Par la suite, nous avons modifié les différents paramètres notés ci-dessus selon des scénarios imaginés.

Pour les observations futures, nous considérons qu'un nœud a rejoint le réseau lorsqu'il est **synchronisé avec le réseau** (envoi d'EBs et de DIOs), qu'il a **au moins un parent dans la topologie RPL** et deux cellules particulières négociées (auto TxCell et TxCell pour le parent sélectionné).

Observations

Effets de la topologie et de la fonction d'ordonnancement sur le nombre de nœuds dans le réseau





Nous pouvons de suite observer une forte différence entre le pourcentage de nœuds ayant réussi à rejoindre le réseau avec une fonction d'ordonnancement (ici, MSF).

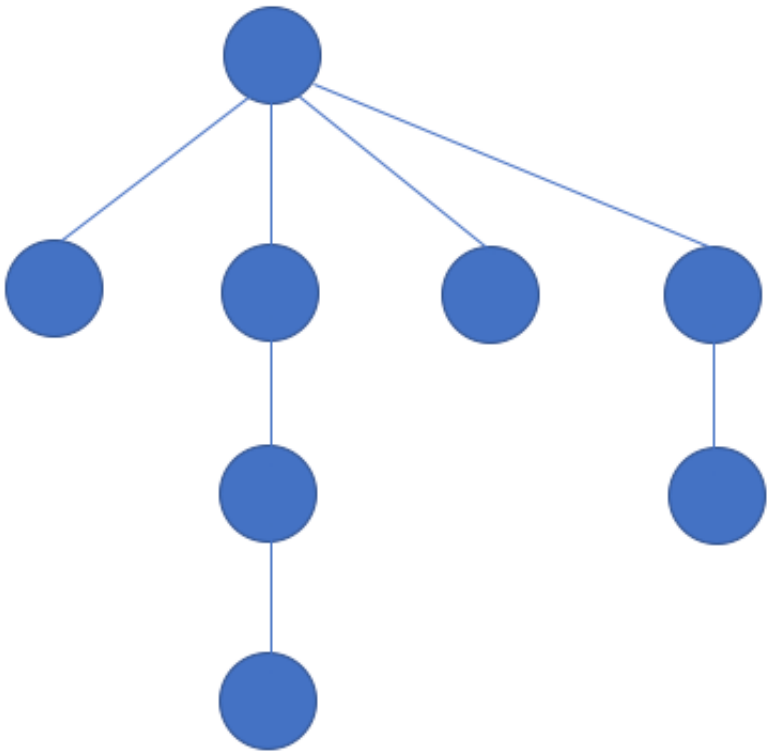
En effet, le processus permettant de rejoindre le réseau sans ordonnancement est basé sur ALOHA, qui comme nous le savons est très moyen en performances même sur un petit réseau, et s'effondre totalement lorsque l'échelle du réseau augmente. On voit ici qu'un protocole adapté aux spécificités du réseau fonctionne mieux.

En revanche, nous avons été surpris par la différence de performance selon les différentes topologies proposées par le simulateur, à savoir Linéaire, Aléatoire et Maillée.

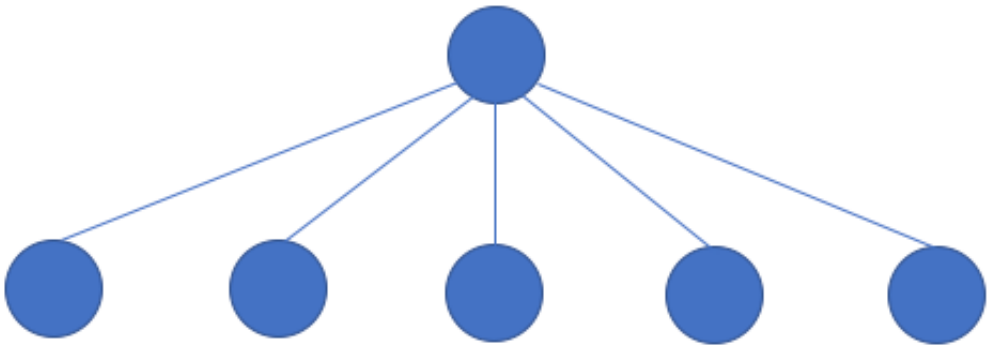
Rappel des topologies :

- Linéaire

- Aléatoire

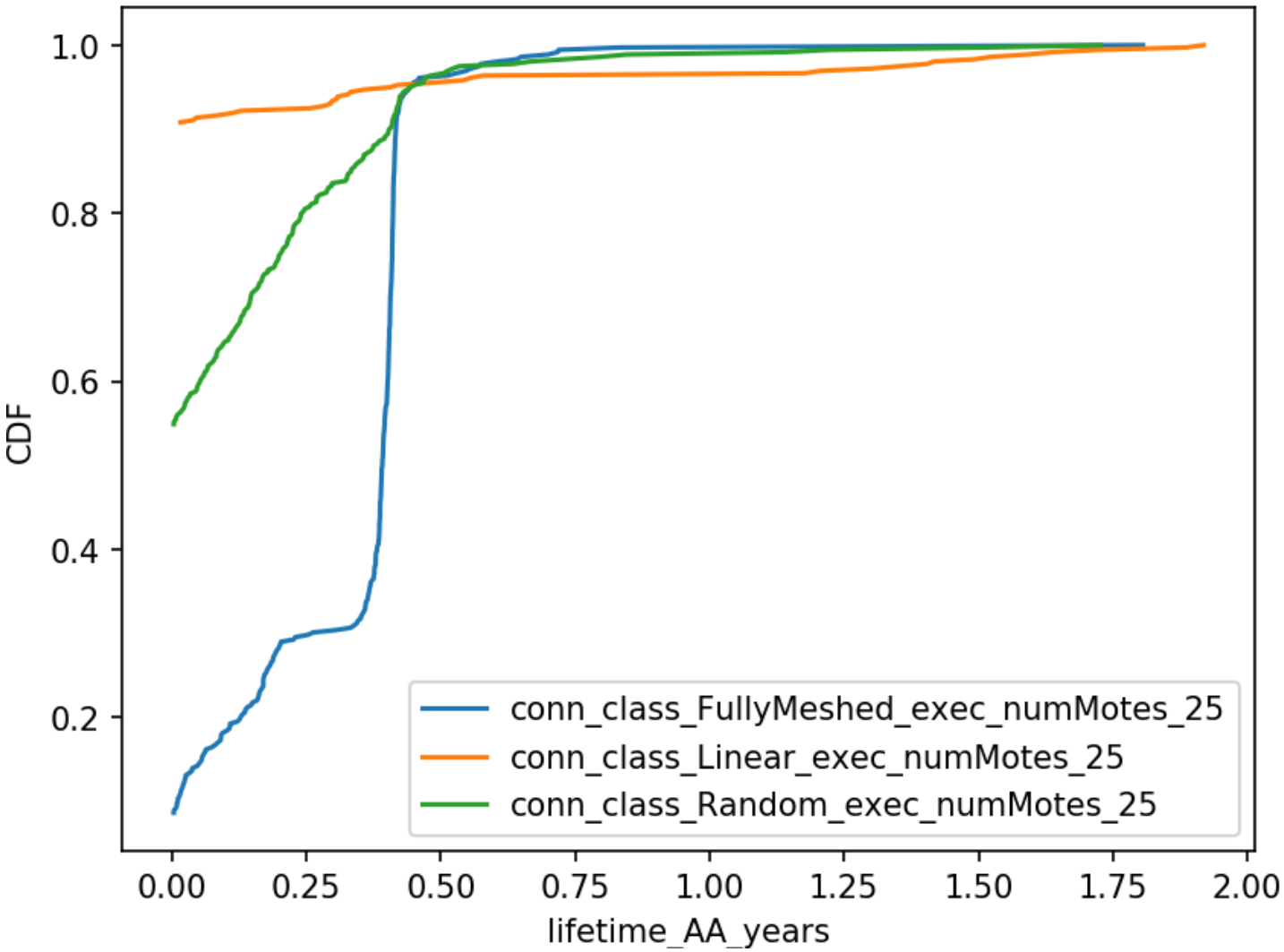


- Maillée



La surprise ne réside pas dans la topologie Maillée, qui est faite pour maximiser la connexion à un réseau, et donc fonctionne efficacement (un seul parent pour chaque nœud dans RPL). De même, en Aléatoire, les nœuds parviennent la plupart du temps à connecter la racine en 1 ou 2 sauts sur un nombre de nœuds limités (ce qui sera de moins en moins vrai à mesure que le nombre de nœuds augmente). En revanche, en topologie Linéaire, pour joindre la racine, un nœud doit passer par un nombre important de sauts, qui augmente avec la taille du réseau. C'est très lent et consomme plus de ressources sur chaque nœud traversé afin de gérer le relayage. Ce qui explique les performances très basses en topologie linéaire observée en *figure 1* et *figure 2* .

Cette différence de performances se remarque aussi sur la durée de vie des nœuds, et incidemment de la durée de vie du réseau :



Nous pouvons observer en *figure 6* les durées de vies des nœuds selon chaque topologie. Dans le cas Linéaire, plus de 90% des nœuds ne vivent pas plus de 140 jours (0,4 année). En Aléatoire, ce pourcentage tombe à 80, ce qui rend le réseau un peu plus résilient. Mais en topologie Maillée, la durée de vie des nœuds est bien meilleure : seulement 30% des nœuds cessent d'émettre après 140 jours. Ce qui confirme l'efficacité de cette dernière topologie.

Cependant, dans un cadre réel, les conditions sont rarement idéales, et la topologie qui a le plus de chance de voir le jour est l'Aléatoire. En effet, sur par exemple un réseau de capteur déployé en extérieur, des obstacles, différences de hauteur et différence de terrain peuvent changer la portée de communication, ne permettant pas à chaque nœud une communication directe avec la racine.

La topologie Linéaire n'aura elle d'utilité que si la topologie physique du réseau est très linéaire aussi, par exemple pour la surveillance de lignes à haute-tension, ou de pipelines.

Effet des EBs sur le nombre de nœuds dans le réseau

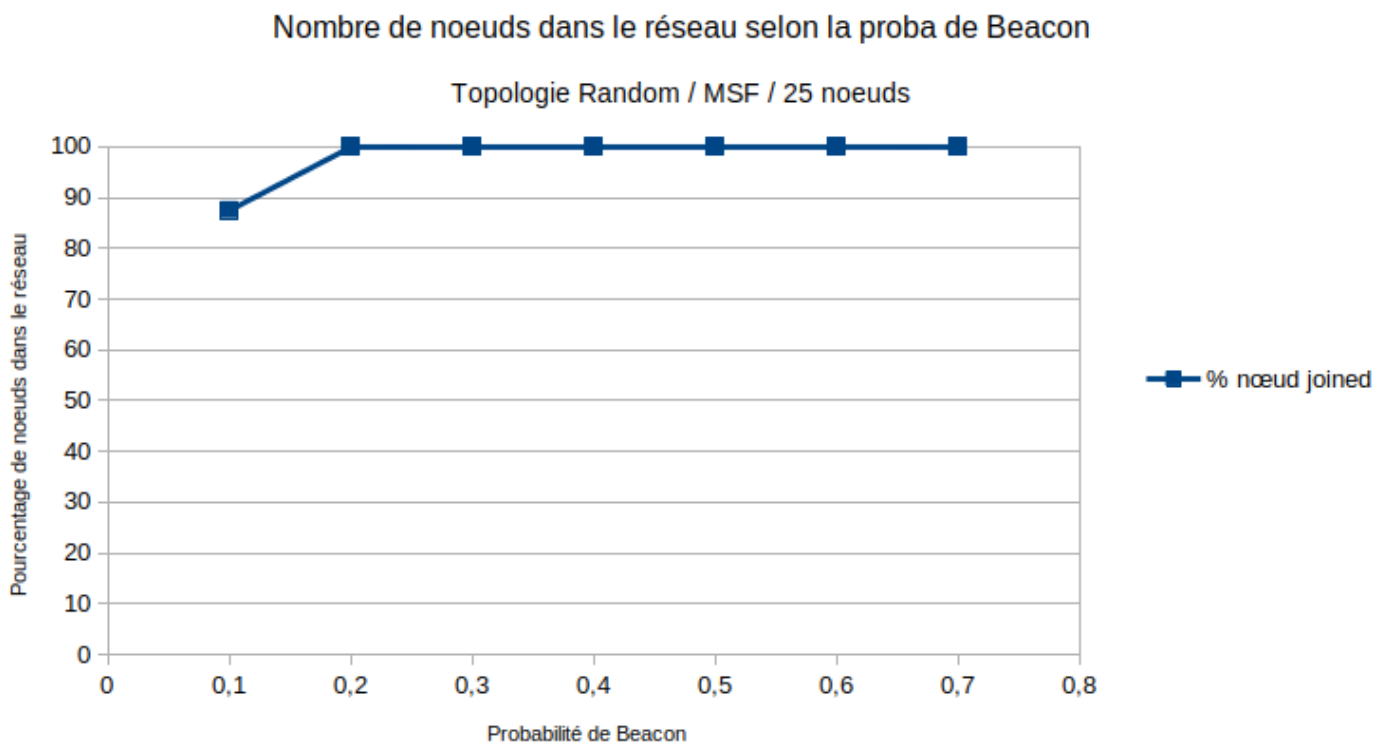
Après avoir analysé le code du fichier *SimEngine/Mote/tsch.py* visible ci-dessous, on peut penser que le paramètre *tsch_probBcast_ebProb* influe sur la fréquence d'envoi des EBs (la probabilité d'envoi).

```
def _decided_to_send_eb(self):
    # short-hand
    prob = float(self.settings.tsch_probBcast_ebProb)
    n    = 1 + len(self.neighbor_table)

    # following the Bayesian broadcasting algorithm
    return (
        (random.random() < (old_div(prob, n)))
        and
        self.iAmSendingEBs
    )
```

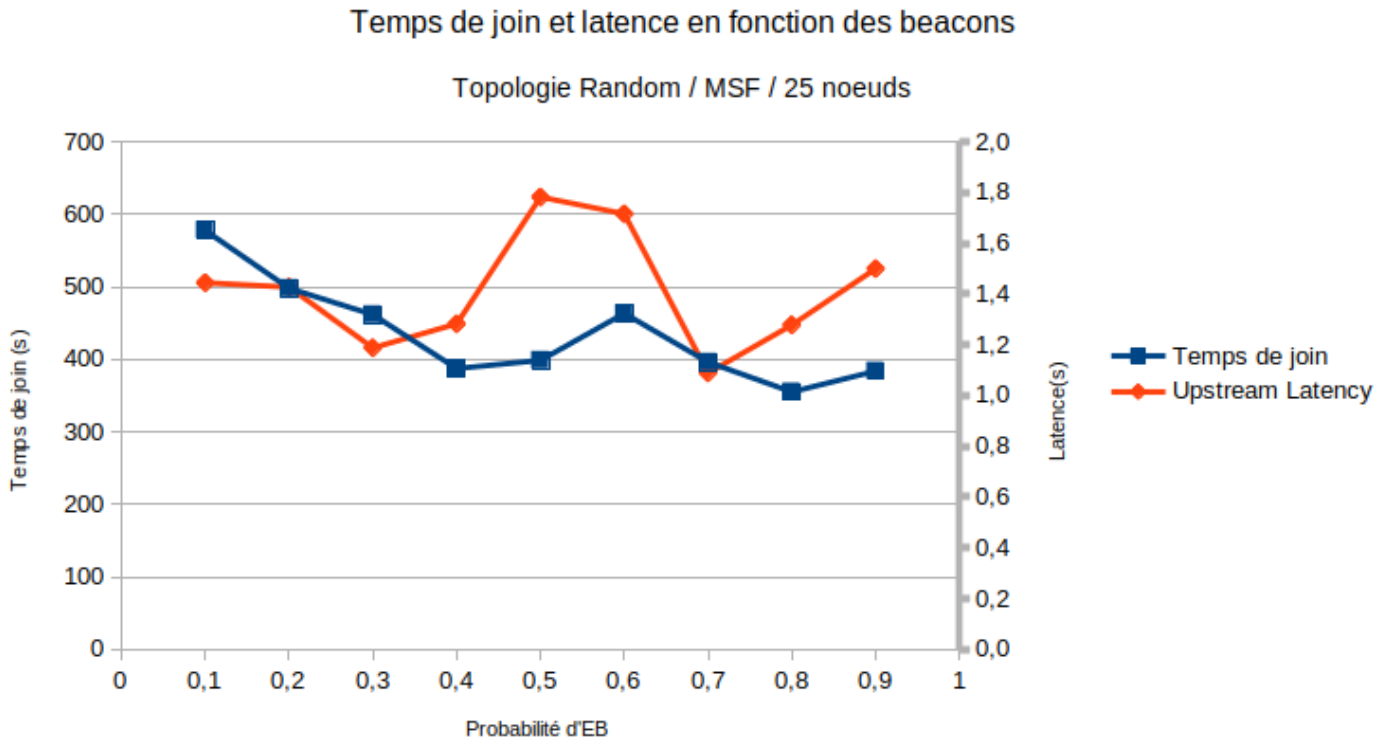
Par la suite, nous avons généralement considéré une topologie aléatoire en partant du principe énoncé dans la section précédente. Nous avons également fait le choix de la MSF pour notre configuration type car ayant déjà étudié l'impact de l'absence de SF. De plus, la présence d'une SF est recommandée et courante.

Ici, on observe sommairement et sans surprise qu'avec une faible probabilité d'envoi d'EBs les nœuds rejoignent moins le réseau. Ce paramètre simule alors différentes conditions plus ou moins dégradées on l'on ne pourrait recevoir correctement les EBs.



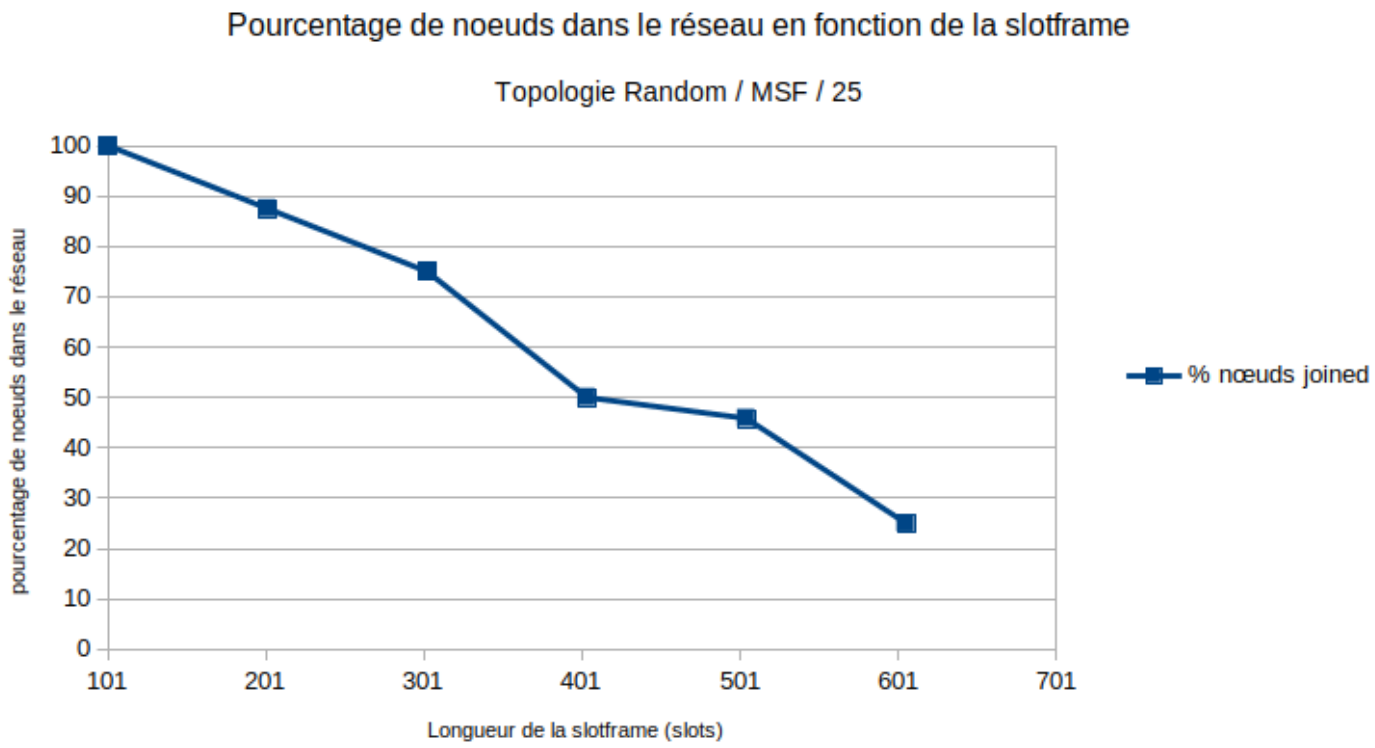
On voit ici que le temps mis par les nœuds pour rejoindre le réseau décroît lorsque les envois d'EBs augmentent. On observe néanmoins une latence sur les liens montants des nœuds variable en fonction du nombre d'EBs envoyés. On peut supposer que lorsque chaque nœud envoie peu d'EBs après avoir rejoint le réseau, la latence est faible étant donné le fait que l'émission est plus fréquente (on ne peut

simultanément émettre et recevoir). En revanche, on peut supposer du même fait que la latence augmente lorsque chaque nœud émet plus régulièrement, sorte de congestion.



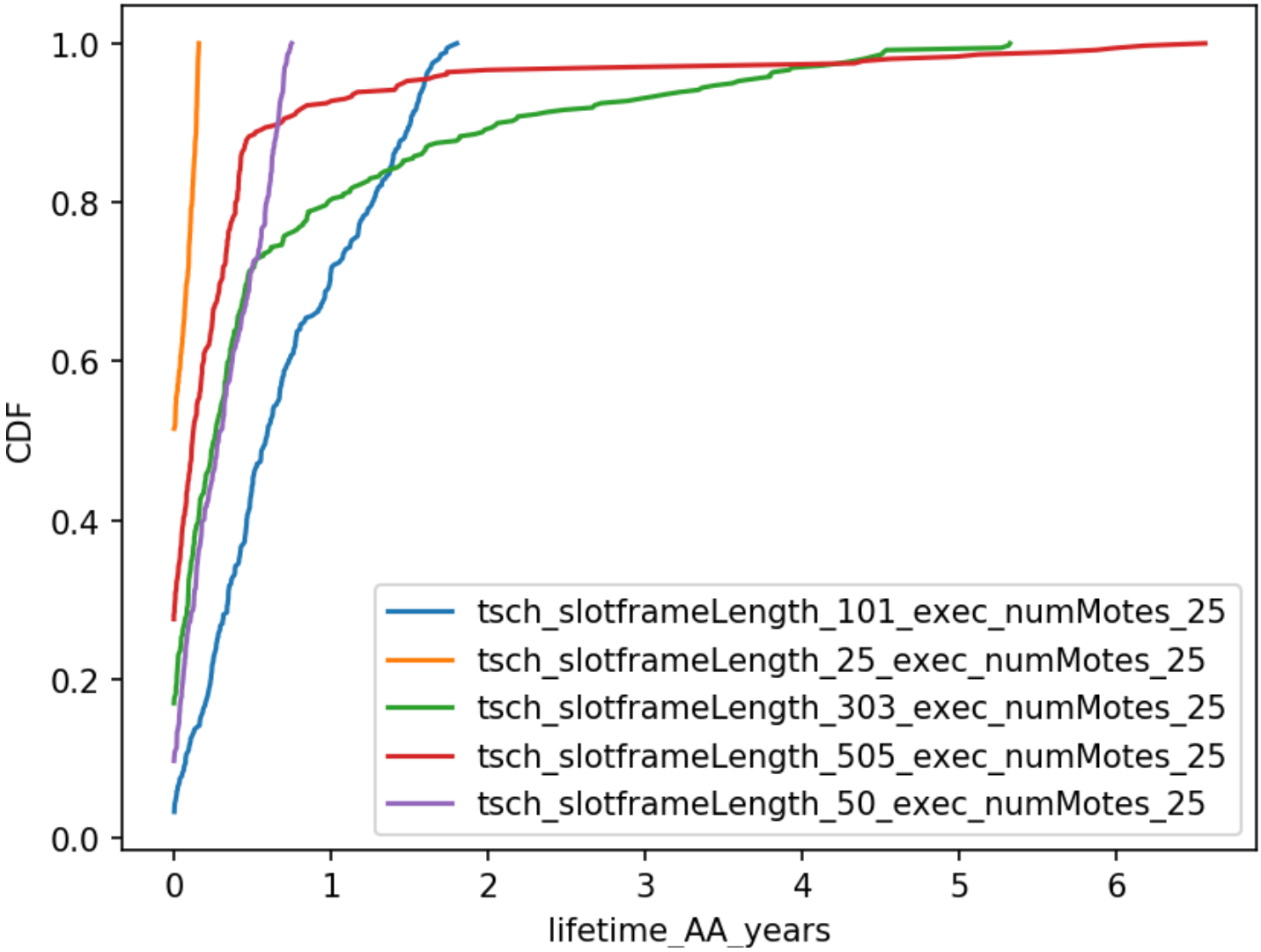
Effet du nombre de slots par slotframe sur le nombre de nœuds dans le réseau

Le paramètre suivant sur lequel nous avons décidé d'expérimenter est le nombre de slots dans une slotframe, donc au niveau *tsch*. Nous sommes donc partis de la valeur par défaut, 101, suggérée dans la RFC :



Nous pouvons observer qu'augmenter le nombre de slots "empêche" la majorité des nœuds d'entrer dans le réseau. Nous ne sommes pas parvenus à expliquer pourquoi.

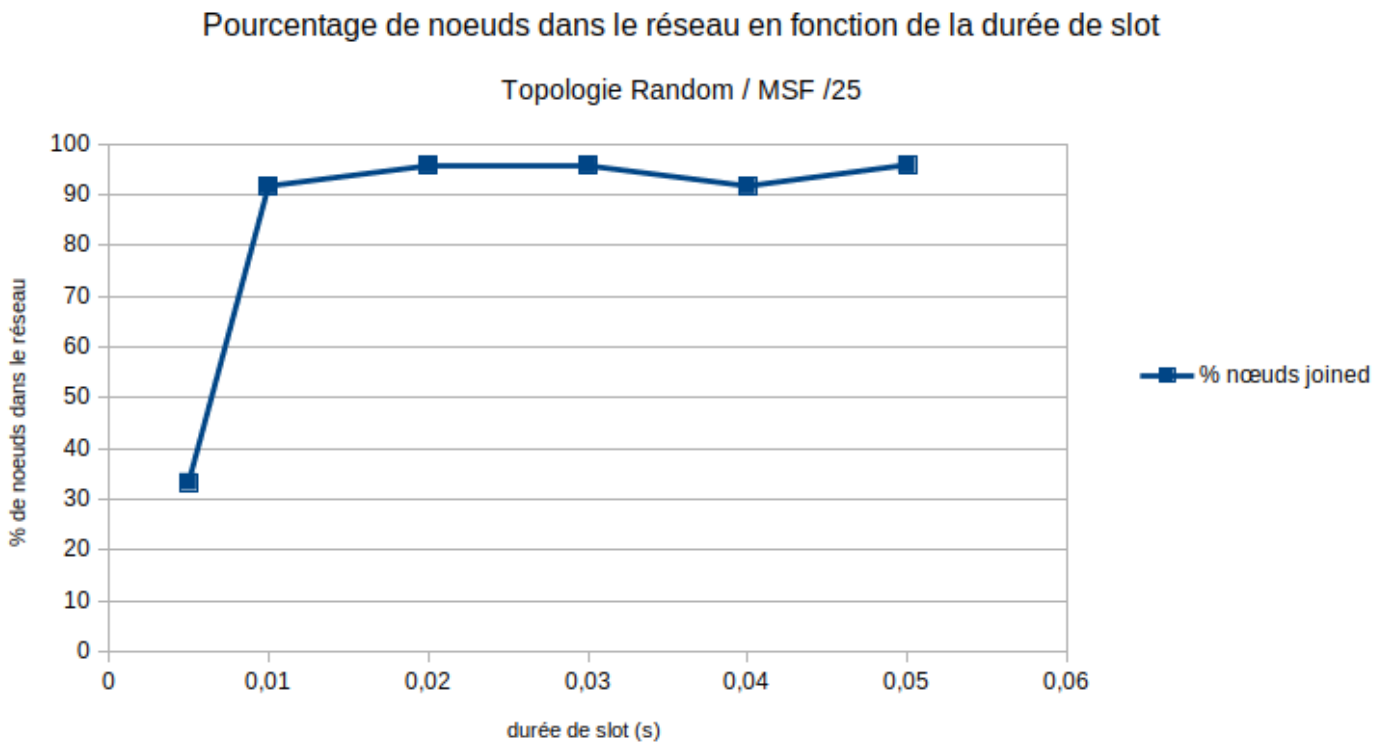
En revanche, nous pouvons nous poser la question de l'efficacité avec une slotframe plus petite que celle recommandée, notamment du point de vue de la durée de vie :



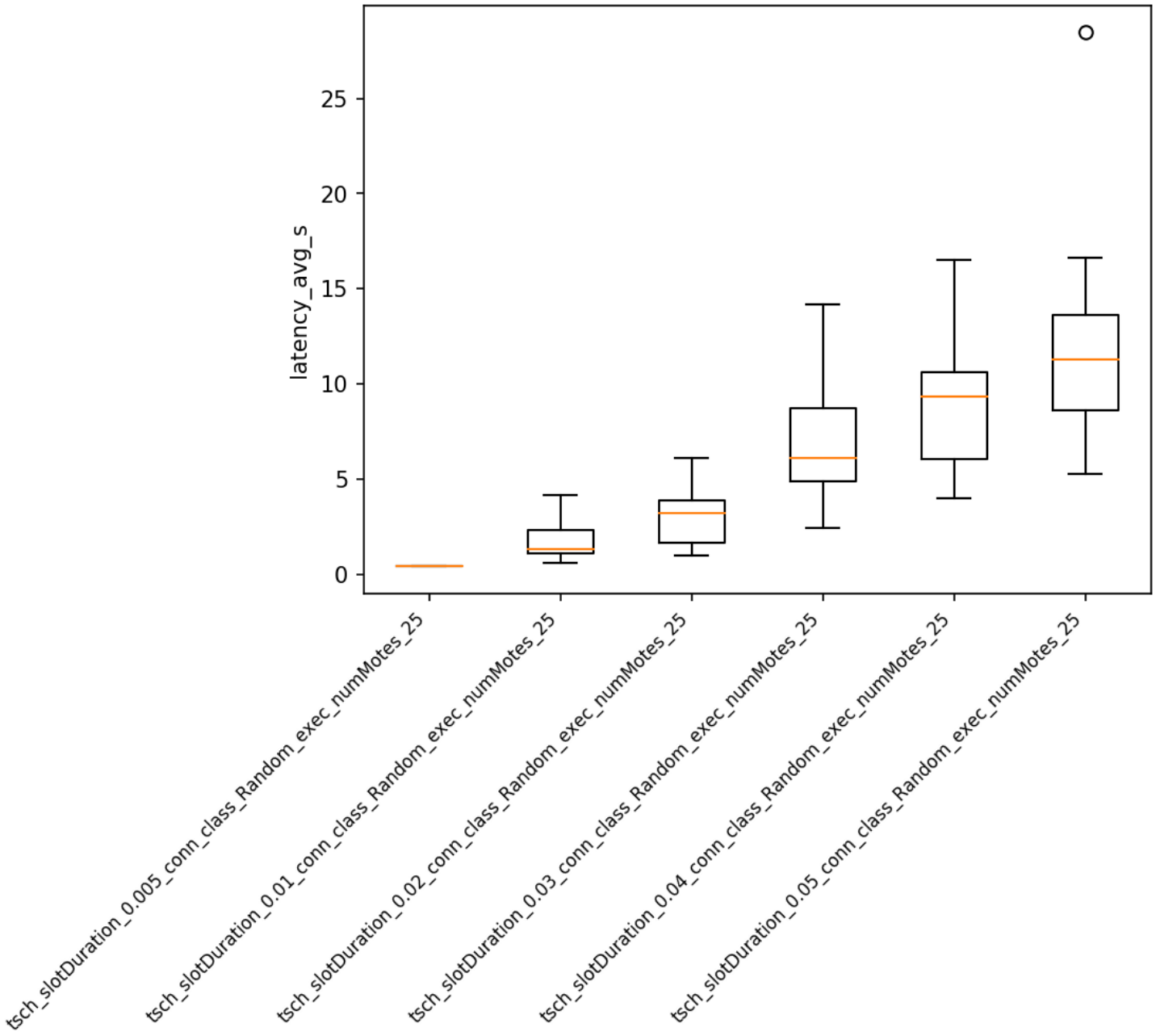
Nous pouvons voir ici que les nœuds avec la meilleure chance de survie sont ceux avec une slotframe de 101 slots. Baisser comme augmenter la longueur de cette dernière a une influence assez importante sur la batterie des nœuds, rendant donc le réseau moins efficace sur la durée. Il semble logique de garder la valeur recommandée de 101, aucun gain n'étant observable en changeant cette valeur.

Effet de la durée d'un slot sur le nombre de nœuds

Ayant fait des tests avec la longueur de la slotframe, nous avons ensuite décidé de nous pencher sur la durée d'un slot à l'intérieur d'une slotframe, et de son effet :



Nous pouvons observer que, sauf pour un slot très court, la grande majorité des nœuds parvient à entrer dans le réseau. Il semblerait au premier abord que la durée d'un slot n'a donc que peu d'influence sur le fonctionnement global. Nous avons donc décidé d'étudier ensuite la latence moyenne de communication entre un nœud et la racine :



Nous pouvons observer ici une nette augmentation de la latence moyenne, plus la durée d'un slot augmente. Ce qui est logique, toutes les communications ont lieu au travers des slots, donc un slot plus long augmentera la durée de tout échange. Il paraît donc adapté de garder la durée de slot suggérée, à savoir 0.01s, pour conserver à la fois un nombre important de nœuds dans le réseau, et une latence de communication basse.