Simulation de réseaux sans fils avec ns2

Projet d'algorithmique des réseaux

Jérémie Dumas

Janvier 2010

ENS de Lyon

Introduction

1 Premier exemple sans fil

Q.1.1 $n0$ et $n1$ reçoivent tout deux leur premier paquet vers $t = 76.4s$ (quand le nœud $n1$ se rapproche), et leur dernier paquet est reçu vers $t = 116.3s$ (quand le nœud $n1$ s'éloigne).

Q.1.2 Sur l'exemple décrit dans `exo1.tcl`, on obtient les statistiques suivantes :

```
Nb de paquets TCP dropés            : 15
Nb de paquets TCP envoyés/non reçus : 14
Nb de paquets TCP envoyés           : 3244
Nb de paquets TCP reçus             : 3230
Pourcentage de paquets perdus       : 0.431566
```

Conclusion : presque autant de paquets TCP "*dropped*" que de paquets perdus (i.e. envoyés mais non reçus), pour un taux de l'ordre de $0,43\%$. Il y a toutefois un paquet "*dropped*" qui a été reçu, cela peut correspondre à un paquet reçu avec une puissance trop faible, qui aura donc été jeté par le récepteur.

Remarque. Notez qu'on n'a considéré que les paquets TCP, et non les ack, dans les statistiques ci-dessus.

2 Portée de communication

Q.2.1 Contrairement au modèle de propagation *Free space* qui ne considère qu'un chemin direct (ligne droite) pour propager les ondes radios, le modèle *Two-ray ground* tente également de considérer la réflexion des rayons sur le sol (d'où le nom de *Two-ray*). La longueur de ce deuxième chemin ne dépend bien sûr que de la hauteur des antennes de communications. Des problèmes d'interférences rendent cependant le modèle moins réaliste sur des faibles distances. En revance il présente un réel intérêt dès lors que les distances séparants les antennes de communications deviennent plus importantes.

Une autre limitation du modèle *Two-ray ground* (mais aussi du modèle *Free space*), est qu'il se contente de prévoir la puissance moyenne d'un rayon en fonction de la distance à la source. Alors qu'en pratique, la distribution des puissances à une distance d donnée est rarement uniforme, et causée par de nombreux facteurs dus à l'hétérogénéité du milieux dans lequel on se place. C'est notamment ce défaut que tente de corriger le dernier modèle de propagation implanté dans `ns2`, le *Shadowing model*.

Q.2.2 Sachant qu'ici les paquets échangés ont une taille "utile" de $m = 1500$ octets, on peut cacluler le débit utile de la communication de la manière suivante : on compte le nombre de paquets *received* (i.e. effectivement transmis), noté r, sur une durée δ entre le premier message envoyé et le dernier message reçu. Il suffit alors de calculer le débit utile donné par $r \times m/\delta$, et de le convertir en Mb/s par exemple.

Q.2.3 En répétant la mesure décrite précédemment sur différentes distances d, on obtient le graphique suivant :

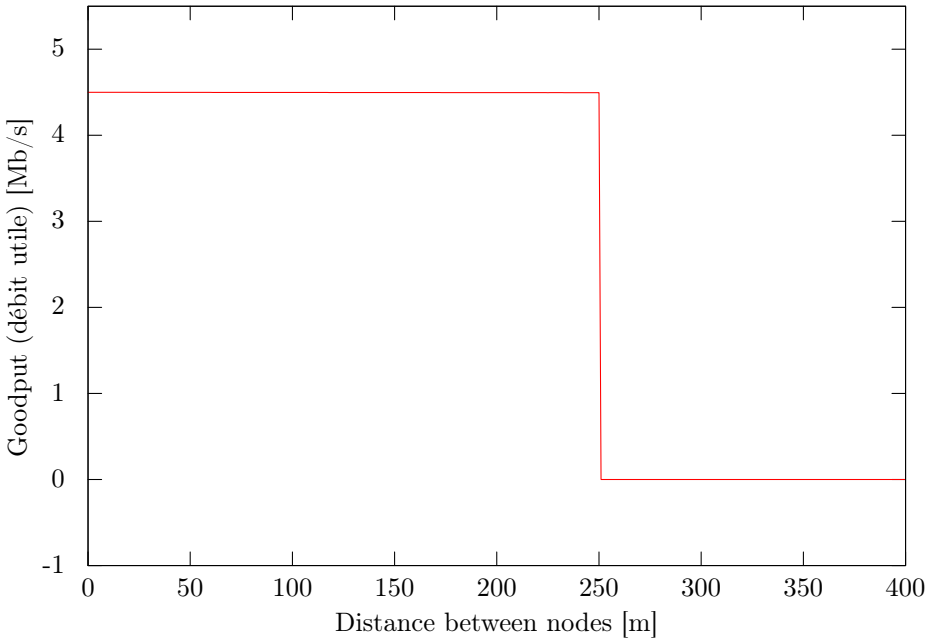

FIGURE 1: Corrélation entre le débit utile et la distance entre deux nœuds, dans le modèle de propagation *Two-ray ground*

On a donc une nette coupure à une distance de 250m, qui est donc la portée de communication recherchée. Un tel ordre de grandeur paraît en tout cas raisonnable, et cohérent avec ce que l'on peut rencontrer dans la vie réelle.

3 Stations cachées

Q.3.1 Sans le mécanisme RTS/CTS, on trouve les débits utiles suivants (calculés grâce aux paquets TCP reçus)

```
Débit utile du nœud _0_ : 2.59385 Mb/s
Débit utile du nœud _2_ : 2.56627 Mb/s
```

Q.3.2 En activant le mécanisme RTS/CTS, on obtient les nouveaux débits utiles :

```
Débit utile du nœud _0_ : 1.85613 Mb/s
Débit utile du nœud _2_ : 1.8016 Mb/s
```

Cela parait un peu étrange a priori. En effet, RTS/CTS a été conçu pour éviter justement ce problème des stations cachés, et permettre aux stations émettrices d'éviter des collisions (et donc d'avoir un meilleur débit). Peut-être le phénomène s'explique-t-il par le

mécanisme d'évitemment des congestions propre à TCP, qui permettrait dans le premier cas aux deux stations émettrices de transmettre plus efficacement leurs données.

4 Trois paires

Q.4.1 On obtient les débits suivants sur le scénario des trois paires :

```
Débit du nœud _0_ : 5.92506 Mb/s
Débit du nœud _1_ : 0.13182 Mb/s
Débit du nœud _2_ : 5.92658 Mb/s
```

On constate donc que les deux nœuds qui commencent en premier émettent au débit qui leur est indiqué dans le script, alors que le nœud $n1$, qui commencent après, n'a pas le loisir d'occuper toute sa bande passante. Ce phénomène peut s'expliquer de plusieurs façons. Par exemple quand le nœud $n1$ veut commencer à communiquer, le canal est déjà occupé (il entend $n0$ et $n2$ émettre) : il doit donc tirer un backoff et attendre encore. Comme $n1$ est soumis entends les communications des deux nœuds voisins $n0$ et $n2$, il sera plus souvent amené à augmenter son backoff, et donc à émettre encore moins souvent. Au final c'est un genre de cercle vicieux, qui fait que $n1$ ne peut quasiment pas émettre.

5 Stations cachées asymétriques

Q.5.1 Cf. la réponse à la Q.2.1 : modèle *Free space* = on ne considère que la ligne droite pour propager les ondes radios.

Q.5.2 DSDV (Destination-Sequenced Distance Vector routing) est un algorithme de routage à vecteur de distance, i.e. basé sur l'algorithme de Bellman-Ford pour le calcul des plus courts chemins. Les mises à jours sont diffusées assez régulièrement dans le réseau, mais peuvent être également déclenchées en cas de changement de topologie par exemple (ce qui va vraisemblablement forcer une mise à jour de la table de routage).

Q.5.3 On trouve les débits utiles suivants :

```
Débit utile du nœud _0_ : 0.0578207 Mb/s
Débit utile du nœud _2_ : 0.400056 Mb/s
```

Sachant que le débit utile est calculé en fonction des paquets qui sont effectivement *reçus* par le destinataire, on comprend que le nœud $n0$ ait un débit utile beaucoup plus faible (alors qu'il aurait un débit matériel plus important). En effet, $n0$ ne se rend pas compte qu'il gêne $n2$ dans ses communications, et continue donc d'envoyer ses paquets comme si de rien n'était. Il y a donc assez peu de paquets de $n0$ qui passent correctement (i.e. sans collision, cf. question suivante).

Q.5.4 Eu égard aux paquets perdus, on obtient les statistiques suivantes :

```
##### Nœud  _0_ #####
Nb de paquets TCP dropés            : 1712
Nb de paquets TCP envoyés/non reçus : 1713
Nb de paquets TCP envoyés           : 2000
Nb de paquets TCP reçus             : 287
Pourcentage de paquets perdus       : 85.65
##### Nœud  _2_ #####
Nb de paquets TCP dropés            :
Nb de paquets TCP envoyés/non reçus : 0
Nb de paquets TCP envoyés           : 1900
Nb de paquets TCP reçus             : 1900
Pourcentage de paquets perdus       : 0
```

Cela corrobore ce qui a été dit plus haut. En effet, le nœud $n0$ a beau avoir envoyé un peu plus de paquet (il a commencé avant), beaucoup sont dropés car ils ont des erreurs, dues à des collisions/interférences avec les paquets envoyés par $n2$.

6 Multi-saut

7 Conclusion

L'utilisation de `ns2` permet de mettre en place rapidement des scénarios aussi bien simples que complexes, pour des réseaux filaires ou sans fils. Le logiciel permet notamment de simuler des scènes qu'il serait beaucoup plus coûteux de mettre en place dans la réalité. En ce qui concerne les réseaux ad hoc qui plus est, les aléas d'une expérimentation en condition réelle ne rendent pas toujours plus pertinents les résultats obtenus par rapport à une "simulation" (quoique ce point soit discutable).

En tout cas, malgré cette simplicité apparante, le format de trace un peu plus complexe et l'absence d'outils plus haut niveau pour traiter les données produites rendent assez difficile une utilisation vraiment généralisé du logiciel. Reste à espérer que `ns3` améliore les choses de ce côté là ...