Evaluation de Performance – Master 1
TP 1 : Simulations en NS-2 & Nam

Utilisation de Network Simulator - 2

NS-2 se trouve sur *lucien*. Pour y accéder, tapez :

```
ssh -X lucien
```

depuis un terminal du réseau interne (l'option `-X` vous permets d'ouvrir des éditeurs de texte en modalité graphique). Repondez 'yes' à la première question, ensuite insérez le même mot de passe que vous avez utilisé pour accéder à la machine.

Vous êtes prêts ! Vous pouvez taper `ns` pour rentrer dans la modalité intéractive du simulateur (ce qu'on n'utilisera pas dans ce tp). Pour sortir de la modalité intéractive, tapez `exit`.

Pour écrire un programme en *NS-2* vous pouvez utiliser un éditeur de texte à votre choix (*vim*, *emacs*, *joe*, ...). *NS-2* est basé sur le langage *TCL*. Vous devrez donc donner une extension `.tcl` à vos fichiers. Quand vous avez écrit un programme `simu.tcl`, vous pouvez faire partir la simulation en tapant `ns simu.tcl`. Si votre simulation génères des "traces", vous pouvez les regarder à l'aide d'un éditeur de texte. Si votre simulation génères un fichier `nom.nam`, vous pouvez l'ouvrir en tapant `nam nom.nam`.

Exercice 1 : Programmer une simulation qui ne fait rien et s'arrête après cinq secondes.

Exercice 2 : Modifier le programme pour que tous les évènements soient enrégistrés dans un fichier `trace.tr`. Faire partir la simulation et ouvrir le fichier qui est créé. Qu'est-ce qu'on peut observer ?

Exercice 3 : Modifier le programme pour enregistrer une autre trace, à simuler avec `nam`. Redemmarrer la simulation du programme (fichier `.tcl`) pour que cette dernière trace soit créée. Ensuite, la simuler avec `nam`.

Exercice 4 : Modifier le programme pour que `nam` soit appellé à l'intérieur de la première simulation (utiliser `exec COMMANDE`).

Exercice 5 : Rajouter un canal unidirectionel entre deux noeuds $n0$ et $n1$, où la largeur de bande est de $1Mb$, le delai de propagation est de $10ms$ et la queue associée au canal, de type *DropTail*, est très grande (par exemple 100000). Comment a changé la visualisation `nam` ?

Exercice 6 : Pour introduire des évènements, il faut associer les noeuds à des agents, puis faire communiquer les agents. Une façon de faire cela est la suivante :

1. associer le noeud $n0$ à un agent UDP, $udp0$ (un objet `Agent/UDP`) ;
2. associer $udp0$ à un générateur de traffic constant (un objet `Application/Traffic/CBR`, où `CBR` est l'acronyme de 'Constant Bit Rate') ;

3. définir la dimension des paquets de ce générateur (500) et l'intervalle entre envois (0.005) ;
4. faire démarrer le générateur à un moment donné (par exemple, après $0.5s$ du debut de la simulation) et le faire terminer quelques temps plus tard (on va dire, après $4.5s$), possiblement avant la fin programmée de la simulation, mais pas forcement.

Comment est changée la simulation en `nam` maintenant ? Ouvrir le fichier `trace.tr` et essayer de comprendre le sense de chaque colonne. Modifier tous les paramètres avec plusieurs combinaisons et voir les effets sur la simulation (et sur la trace). En particulier, mettre le delai du canal à 0 (ça représente quel paramètre vu en td et en cours ?). Mettre la dimension des pacquets à une valeur supérieur à la largeur de bande, qu'est-ce qui se passe ?

Exercice 7 : Écrire un programme qui, sans utiliser un générateur de traffic, simule une file d'attente de type $M/M/1$, où le premier paramètre représente l'intervalle moyen entre arrivées (qui suit une distribution de Poisson) et le troisième représente le fait que notre système a un seul serveur, alors que le deuxième paramètre represente ici la dimension du paquet[1].

Pour cet exercice, au lieu d'utiliser un générateur de traffic, définir une méthode `sendpacket` qui, à des intervalles données par une variable aléatoire exponentielle (vous pourriez trouver utile la méthode `now` de la classe `Simulator`), envoi des paquets de dimension donnée par une autre variable aléatoire exponentielle. Les variable aléatoires exponentielles sont modélisées en *NS-2* avec des objets de type `RandomVariable/Exponential`. Ces objets ont notamment une méthode pour definir la valeur moyenne (`set avg_`) et une pour obtenir la valeur actuelle (`value`). Définir l'intervalle moyenne comme l'inverse d'une certaine constante `lambda`.

[1] En cours, on avait suivi la notation où le deuxième paramètre indique le temps de traitement. En fait, dans le scénario actuel, le temps de traitement ne depends que de la dimension du paquet, pourquoi ?