TP NS : Etude du protocole TCP

RSA : Réseaux et Systèmes Avancés

Deuxième année

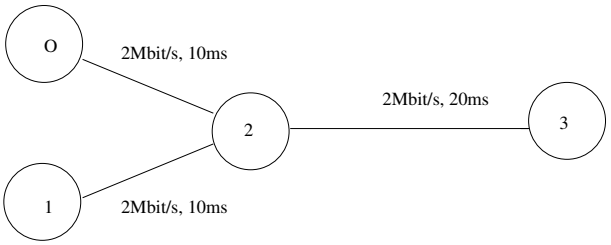
1 Introduction

NS (Network Simulator) est un outil de simulation de réseaux orienté objet développé à Berkeley. ns est souvent utilisé pour étudier le comportement des flux selon un protocole de transport ou de couche réseau. On l'utilisera dans ce cours pour une fin pédagogique. ns est écrit en C++ mais la définition des configurations se fait en Otcl (Object-Tcl).

2 Premier exemple ns

Le but de cette premiére section est de simuler un réseau simple afin de se familiariser avec ns.

- Récupérez le contenu du répertoire /home/depot/chrismen/. Lisez le contenu du fichier ns-simple.tcl pour voir à quoi correspond un script ns (cf. figure 2).
- Pour lancer la simulation, faites dans un terminal : ns ns-simple.tcl. Cette commande exécute la simulation et lance NAM qui visualise le déroulement de la simulation. Vérifiez aussi que le simulateur a bien généré les fichiers simple.tr et simple.nam.
- Cliquez sur les différents liens pour voir leurs caractéristiques. Vérifiez également la position des nœuds ainsi que les liens en fonction du script tcl.
- Mettez en marche l'animation, surveillez le compteur de temps pour vérifier que la source commence l'émission au bon moment, jouez sur le pas (step) pour diminuer ou augmenter la vitesse d'animation, essayez un peu tout. Vérifiez par exemple les caractéristiques des différents flux qui circulent.
- Repérez dans le script, la ligne qui détermine la capacité du lien qui relie les nœuds 2 et 3 de la topologie. Changez sa capacité de 2Mb à 1.7Mb par exemple et examinez dans nam l'impact de cette diminution ainsi que le comportement des 2 flux concurrents.



3 Le protocole TCP

TCP (Transmission Control Protocol ou le protocole de contrôle de transmission) est le protocole de transport le plus utilisé sur Internet. Les applications comme HTTP et FTP l'utilisent. Le but de ce TP est de comprendre le comportement de TCP avec certaines de ses variantes à l'aide de ns. Pour cette section on utilisera le contenu du répertoire TCP.

3.1 Rappel

Une des premières implémentations de TCP proposée par Van Jacobson en 1988 est connue sous le nom de *TCP Tahoe*. Plusieurs autres variantes ont été proposées par la suite comme TCP Reno, New Reno, Vegas et SACK. Dans ce TP, on se limitera à Tahoe et Reno.

TCP Tahoe

TCP Tahoe mets en oeuvre 3 algorithmes, le *slow start*, le *congestion avoidance* (évitement de congestion), et le *fast retransmit* (retransmission rapide).

Initialement l'émetteur est dans la phase slow start la fenêtre de congestion (cwnd) est initialement mise à 1. A chaque réception d'un acquittement du récepteur, l'émetteur augmente sa fenêtre de 1. Ceci augmente exponentiellement la fenêtre de congestion.

Le protocole introduit un seuil de fenêtre une fois atteint, l'émetteur entre dans la phase d'évitement de congestion où la fenêtre de congestion est incrémentée d'un paquet par RTT (round trip time : temps aller/retour entre l'émetteur et le récepteur).

Si 3 ACKs dupliqués contenant le même numéro de séquence sont reçus, le paquet est supposé perdu et il est retransmis sans attendre l'expiration d'un temporisateur. La fenêtre de congestion est réduite à 1 et une nouvelle phase de slow start est entamée.

TCP Reno

La différence principale avec Tahoe est que la fenêtre de congestion est réduite à sa moitié au lieu de 1 lors de la réception d'un triple ACK (Fast Recovery).

L'expiration d'un temporisateur indique une congestion sévère et la fenêtre de congestion est réduite à 1.

TCP New Reno

La différence principale entre Reno et New Reno se réside dans le cas où plusieurs paquets consécutifs sont perdus. Contrairement à Reno qui quitte le mode de Fast Recovery à la réception d'un acquittement partiel, TCP New-Reno ne quitte le mode de Fast Recovery que si l'ACK reÁu acquitte tous les segments envoyés. A la réception d'un ACK partiel, New-Reno retransmet immédiatement le paquet suivant le dernier paquet acquitté dans cet ACK, diminue la taille de la fenêtre du nombre de paquets acquittés par cet ACK partiel et retransmet un paquet si c'est permis par la taille de cwnd.

Un ACK partiel est un ACK qui acquitte au moins le segment perdu (celui qui nous a fait entrer en Fast Recovery) mais pas tous les segments envoyés.

TCP SACK

Dans cette variante est rajoutée la notion d'acquittements sélectifs. Le récepteur peut signaler des plages de paquets qui ont été reçus.

3.2 Fenêtre de congestion

1. Lisez le fichier tcp_sim.tcl. Il permet de simuler une topologie simple avec un émetteur qui émet du flux ftp (en utilisant le protocole de transport TCP, ici il s'agit de Tahoe) à un récepteur via un routeur. Lancer ns et ensuite nam et vérifier les caractéristiques des liens et des paquets TCP. Devinez o se trouve le goulot d'étranglement. Examinez aussi l'évolution de la fenêtre de congestion à partir de l'animation.
2. La fonction *record* est utilisée pour aller voir périodiquement la taille de la fenêtre de congestion au niveau de l'émetteur (**cwnd_**). *record* est appelé au début de la simulation et ensuite périodiquement tous les *interval* secondes. Les résultats sont stockés dans un fichier qui s'appelle *trace_file.tr* avec le temps en première colonne et la taille de la fenêtre de congestion en 2ème colonne. Lisez le fichier trace_file.tr pour vérifier son contenu.
3. Il est plus intéressant de représenter le contenu de trace_file.tr par une courbe. Le fichier plot_window.cmd contient les instructions gnuplot nécessaires pour tracer la taille de la fenêtre de congestion au cours du temps. Créer un sous répertoire "courbes" et taper gnuplot < plot_window.cmd. Lisez bien le contenu de plot_window.cmd. Visualiser courbes/cwnd.eps en utilisant display.
4. Examinez la courbe obtenue et distinguez les 2 phases du slow start et d'évitement de congestion.

3.3 Acquittements

Pour comprendre comment la fenêtre de congestion est mise-à-jour dans la simulation précédente, nous avons besoin d'examiner les émissions de paquets et les acquittements correspondants. Dans le fichier tcp_sim.tcl, arrêtez le flux ftp à 1.2 et appelez la procédure finish à 1.3. Pour arriver à nos fins, nous allons exploiter le fichier de trace out.tr généré par ns. Un fichier .tr contient quelques informations sur le déroulement du transport des paquets dans le réseau. Les lignes de fichier de trace ont le format suivant :

```
<evt> <temps> <from> <to> <ptype> <size>
<flags> <id_> <src node.port> <seqnum> <pkt_id>
```

où *evt* est 'r' (recv), 'd' (drop) '+' (enqueue) '-' (dequeue). On peut retirer des statistiques de ce fichier avec des scripts shell en utilisant awk, grep, sed, etc.

1. Lisez le fichier script1.sh et devinez ce que chaque ligne fait.
2. Lisez également plot_ack.cmd et devinez la courbe tracée.
3. Lancer le script ./script1.sh (vérifier les droits d'exécution s'il le faut) et visualiser la courbe tracée.
4. Essayez d'interpréter les courbes **ack.eps** et de faire le lien avec la courbe **cwnd.eps** du paragraphe précédent.

3.4 TCP Tahoe

Copiez le fichier tcp_sim.tcl dans tahoe.tcl. Toutes les modifications sont à faire sur ce dernier.

1. Pour provoquer des pertes, la file (buffer) au niveau du routeur sera limité à 3. décommentez la ligne correspondante. Faîtes de telle sorte que le flux ftp s'arrête à 3.0 et que la procédure finish soit appelée à 3.1. Lancer ensuite ns et puis nam et examinez ce qui se passe.
2. Le script script2.sh est similaire à script1.sh pour les simulations de Tahoe. Lancez-le et visualisez la simulation à l'aide de nam. Visualisez ensuite les courbes courbes/ack_tahoe.eps et courbes/cwnd_tahoe.eps.
3. A l'aide de la courbe cwnd_tahoe.eps, observez et essayez d'interpréter l'évolution de la taille de la fenêtre de congestion.
4. Observez maintenant les courbes de ack_tahoe.eps, essayez de faire le lien avec la courbe cwnd.
5. Vous pouvez vérifiez dans le fichier de trace les numéro de séquence des paquets éliminés ('d') au niveau du routeur. Utilisez la commande **grep**, par exemple : **grep d out.tr**. Revenez à vos courbes pour mieux les interpréter.
6. Quel évènement a provoqué le passage au slow-start ?
7. Dans les simulations précédentes, le seuil pour quitter le slow start était fixé à 16 (variable window_). Changez cette valeur en 8 et puis 4 et examinez l'impact de ce changement.

3.5 TCP Reno

1. Copiez le fichier tahoe.tcl dans reno.tcl en rétablissant la valeur 16 de seuil. Pour remplacer Tahoe, il suffit de remplacer **set tcp [new Agent/TCP]** par **set tcp [new Agent/TCP/Reno]**. Pour générer les courbes, écrivez le script correspondant. Essayez d'analyser les courbes comme vous l'avez fait pour Tahoe en montrant à chaque fois les différences.
2. Il est intéressant de voir l'évolution à long terme de la fenêtre de congestion. Essayer de faire durer la simulation plus longtemps (par exemple 20 à 50s si besoin). Essayez de changer les paramètres de la simulation comme Áa vous semble en observant à chaque fois l'impact. Faites-le à la fois pour la version Tahoe et Reno en essayant de distinguer les 2 variantes.

3.6 Pour aller plus loin ...

Pour les plus curieux parmi vous, je vous propose d'expérimenter d'autres variantes de TCP qui sont par exemple NewReno, Vegas et Sack. Il suffit de remplacer Reno successivement par Newreno, Vegas et Sack. Bonne chance !