Master 1
Sciences et Ingénierie du Réseaux, de l'Internet et des Système

Évaluation de performance

Rapport

Travaux Pratique
NS2 - Simulations avancées & Développement

Constantin DIVRIOTIS

Tables des Matières

1. Réalisation et pertes

Manuel d'utilisation :

- `make` : lancement de la simulation
- `make analyze` : analyse des données et création de `loss_results.txt` et de `analyze.txt`
- `make clean` : nettoyer le répertoire

Avec les différentes informations obtenues, il est apparu que :

- le fichier traff.traf contenait des données à envoyer en MégaBit
- et le fichier topo.top contenait la capacité de la bande passante des liens en GigaBit et un délai en m/s.

Lors de mes différentes simulations, il m'est apparu **impossible** de réussir à faire tourner la simulation, car le débit à envoyer était bien trop élevé.

Exemple : Entre le noeud 1 et le noeud 6, il est demandé d'échanger 16683403 MégaBit en 900 secondes (15 minutes). Notre débit est alors de :

Débit_durant_On = 2D = 2 * (Data / Durée) = 2 *(16683403/900) = 2 * 18537.1144444 = 36000 MégaBit = 36 GigaBit

Or, nous avons un lien de capacité de 10 GigaBit. Il est totalement impossible d'y arriver alors. Il est à noter que même si l'on divise nos paramètres par un facteur 1000, le ratio donnée/débit reste le même.
J'ai décidé alors de modifier la valeur des différents paramètres, après de nombreux essais :

- les données envoyées sont des octets
- et la capacité de la bande passante des liens passent en MégaBit.

En effet, il a été nécessaire de baisser les données d'un facteur 100 000 (1000 * 1000) afin d'obtenir des résultats cohérents. Il était nécessaire de diminuer également la capacité de la bande passante des liens d'un facteur 1000 pour garder un ratio donnée/débit cohérent avec le sujet. En résumé, les données ont été baissés d'un facteur 100 000 et la capacité de la bande passante d'un facteur 1000. Ces changements ont été effectués car, la configuration est moins lourde que la configuration avec des données en KiloBit et la capacité de la bande passante en GigaBit et permet d'avoir des simulations rapides.

Le modèle ON/OFF est la distribution utilisée afin de générer un trafic pseudo-réaliste. On génère alors un flux par ligne accompagné des flux TCP. Le programme est entièrement commenté afin d'expliquer chaque ligne.
Il y a une période de démarrage et une période d'arrêt (5% de la durée simulée au début et à la fin) permettant de simuler les démarrages et les fins d'un trafic.
Ensuite, les volumes du modèle ON/OFF et des flux TCP témoins sont calculés selon la durée de période ON que l'on a défini.

Dans notre simulation, nous avons placé une durée ON de 0.90. On choisit alors d'avoir au minimum 1 flux témoin TCP.

Le débit est, comme dit précédemment, calculé selon la formule :

Débit_durant_On = 2D = 2 * (Data / Durée)

Concernant les flux témoins, le volume et le temps sont bornés de façon à ce qu'il ne dépasse pas les données à générer. Le volume est généré aléatoirement grâce à une loi Zipf dont l'argument 'a' peut être modifié. L'argument est placé à 1.10. Après plusieurs tests, si l'argument est trop grand (exemple avec 1.4), le volume généré est trop gros et cela provoque automatiquement des pertes énormes. Plus l'argument s'approche de 1, plus il y aura des flux témoins TCP générés.
J'ai également pu constater que la génération des flux témoins par Zipf ou par une fonction random renvoie un nombre plus ou moins similaire , c'est-à-dire entre 900 (minimum) et 1500 (maximum) flux témoins. Les fonctions random utilisés ont été laissées en commentaires.

Les dates d'envois des flux témoins TCP sont également crées de manière aléatoire grâce à fonction random. La date d'envoi est choisie aléatoirement entre la date du début de la simulation et la date de la fin de la simulation. Cette valeur est bornée si la date dépasse la fin de la simulation : des paquets s'envoyaient encore après la fin de ma simulation, alors après avoir borné la date d'envoi, les paquets sont forcément envoyées dans l'intervalle.

En allongeant la durée de la simulation, les pertes diminuent, car le débit est moins congestionné. En effet, un test permet de l'identifier :

- une simulation sur 45 minutes (3200 secondes) perd 350 environ paquets,
- alors qu'une simulation sur 15 minutes (900 secondes c'est-à-dire la simulation de base) perd 800 paquets.

En terme de réalisme, les résultats paraissent cohérents. Le débit pour une simulation de 900 secondes paraît assez conséquent par rapport aux données en octets et réaliste : 3400 octets/s. Le taux de perte affiché paraît également réaliste : `taux_de_perte = 0.5%`.

Grâce à l'analyse des résultats, on constate que les pires liens sont (le plus souvent) :

- le lien entre le noeud 5 et le noeud 22,
- le lien entre le noeud 18 et le noeud 22 et
- le lien entre le noeud 4 et le noeud 15.

Les différents résultats des pires flux et des pires liens sont dans les fichiers (utilisation de `make analyze`) :

- `loss_results.txt` : les 3 pires liens avec la totalité des pertes de chacun des liens qui ont eu des pertes,

- `analyze.txt` : les 3 pires flux avec la totalité des pertes de chacun des flux qui ont eu des pertes et des résultats généraux de la simulation.

2. Stresser la simulation

Pour stresser la simulation et intensifier la congestion, il suffit de réduire la durée de la simulation. Nous avons précédemment que si la durée de la simulation s'allonge, le débit diminue, donc il y a moins de congestion et cela mène à moins de pertes.

La simulation de base est de 15 minutes c'est-à-dire de 900 secondes.

Si la simulation passe à 5 minutes c'est-à-dire 300 secondes, les congestions s'intensifient et des pertes se produisent plus fréquemment, car le débit devra augmenter :

- on a un débit d'environ 3400 octets/s pour une simulation de 900 secondes et
- on a un débit d'environ 9500 octets/s pour une simulation de 300 secondes.

Il y a un facteur 3 entre les deux simulations, les résultats sont cohérents et réalistes.
La simulation de 5 minutes doit alors présenter un taux de perte supérieure :

- taux_de_perte = 0.6% pour la simulation de 900 secondes,
- taux_de_perte = 1.4% pour la simulation de 300 secondes.

Pour plus de précision aux niveaux des pertes, on a :

- on perd 800 paquets sur la simulation de 900 secondes et
- on perd 10 000 paquets sur la simulation de 300 secondes.

Les pires liens et les pires flux sont différents également (voir les fichiers générés `lost_results.txt` et `analyze.txt` après avoir effectué les commandes `make` et `make analyze`).

3. Améliorations apportés

Dans cette partie, seules deux simulations sont retenues :

- la simulation de base de 900 secondes,
- la simulation réaliste de 300 secondes.

Les critères d'évaluation sont le taux de perte et le débit utile proposé. Ces critères sont les premiers paramètres à regarder pour comparer différents réseaux. En effet, si on constate qu'un réseau propose un très bon débit associé à un taux de perte proche de zéro, on décidera très probablement de choisir ce type de réseau. La capacité de la bande passante des liens est un facteur très important également, mais, dans notre cas, la capacité de la bande passante des liens est posé et on ne peut pas la modifier.

Vegas :

Parmi toutes les améliorations à apporter, j'ai décidé dès le départ de partir des résultats de la première partie. J'ai placé TCP Vegas comme agent sur les flux témoins TCP, car lors de la première partie du projet NS2, j'ai constaté que Vegas apportait énormément de sécurité et de débit. En effet, il y avait très peu de pertes et un débit conséquent sur toutes les simulations TCL effectuées avec TCP Vegas.

Si on reprend la simulation de base d'une durée de 900 secondes, on obtient des résultats similaires en matière de perte :

- taux_de_perte = 0.5 %
- 800 paquets perdus seulement.

On constate que les performances observées dans la première partie ne sont pas en cohérence avec les résultats obtenus. On notera que le débit utile est également assez similaire : 3200 octets/secondes.

De même pour la simulation réaliste de 300 secondes, on obtient un taux de perte similaire (et même presque plus élevé) que la simulation de base :

- taux_de_perte = 1.5 %
- et environ 15 000 paquets perdus.

Le débit utile est le même avec un débit d'environ 9500 octets/s. On constate que les attentes de TCP Vegas ne sont pas la hauteur des résultats obtenues dans la première partie. On peut émettre l'hypothèse que TCP Vegas rencontre des problèmes avec des simulations trop grosses.

New Reno :

Lors de la première partie du projet NS2, j'ai constaté que TCP New Reno était le deuxième TCP le plus efficace après TCP Vegas.

Sur la simulation de base (900 secondes), on obtient :

- `taux_de_perte = 0.7 %`,
- un débit utile de 3200 octets/seconde,
- et 500 paquets perdus.

Sur la simulation de 300 secondes, on obtient :

- `taux_de_perte = 1.5 %`,
- un débit utile de 9000 octets/seconde
- et 15 000 paquets perdus.

On constate que TCP New Reno n'est pas non plus une alternative à TCP Tahoe (TCP de base sur les scripts TCL). En effet, on constate que les taux de perte obtenues et le nombre de paquet perdus ne sont pas réellement meilleurs que les résultats proposés par TCP Tahoe. On ajoutera que les débits utiles proposés par TCP New Reno sont similaires à TCP Tahoe.

Sack :

Avec SACK (RFC2018) appliqué aux agents TCP et TCPSink, je m'attendais à des meilleurs résultats sur les pertes.

Or, dans la simulation de base (900 secondes), on obtient un taux de perte moins bon :

- `taux_de_perte = 0.7 %`,
- et 600 paquets perdus.

On peut noter que l'on perd moins de paquets, mais ce sont des paquets plus conséquents, car le taux de perte est plus élevé que le simulation de base avec le TCP de base c'est-à-dire TCP Tahoe. Le débit utile est, lui, similaire à TCP Tahoe : 3500 octets/secondes.

Pour la simulation à 300 secondes, on obtient un taux de perte légèrement en-dessous de la simulation avec TCP Tahoe et Vegas :

- `taux_de_perte = 1.3 %`,
- et environ 15 000 paquets perdus.

On constate que si on pose les acquittements sélectifs (SACK) sur l'émetteur et le récepteur, il n'y pas réellement d'impact sur la simulation. De même, le débit utile est identique à TCP Tahoe : 9500 octets/seconde.
J'ai alors retiré les acquittements sélectifs du récepteur et on obtient des résultats assez similaires aux résultats précédents. La simulation de base avec SACK présente un taux de perte plus élevé qu'avec TCP Tahoe.

En conclusion, on constate qu'il n'y pas réellement de très grosses différences sur les taux de perte et les débits sur des gros trafics à tel point que TCP Tahoe apparaît comme un TCP performant. J'avais beaucoup d'espérance sur TCP Vegas, car TCP Vegas apparaissait comme un TCP avec des très bonnes performances. Cette deuxième partie est assez contradictoire avec la première partie.

Remarque : Les résultats et les chiffres sont une approximation de la valeur obtenue et ont été **vérifiées au minimum 2 fois.**