Rapport NS2 -partie2-

EL Khadir Hamza

I- Approche du Sujet

Le sujet consiste, d'après ce que j'ai compris, à recréer un scénario de transfert de trafic au sein d'un réseau géant (Un Long Fat Network). Les liens étant paramétrés sur les fichiers topo.top, et le trafic dans traff.traf.

L'approche que j'ai considéré pour le traitement de ce sujet était d'écrire un script python qui génère du code TCL. Après la fin de la simulation en utilisant ns2 et Gnuplot. J'arrivais à avoir des fichiers de traces et des graphes de ces traces, que j'ai réparti dans les dossiers queues/ outs/ et sinks. (en utilisant un script bash).

Une approche que j'ai suivi consistait à attacher un agent tcp-src source sur chaque nœud qui envoyait du trafic et un agent tcp-sink sink sur chaque nœud qui reçoit ce trafic, ainsi je peux suivre le flux entre telle source et telle destination. Ainsi j'avais des fichier de traces de la forme : flux-i-j.tr représentant par exemple fenêtre de congestion du flux entre le nœud i vers le nœud j.

J'ai pensé à subdiviser chaque volume de trafic en totalité mais cela prenait beaucoup trop de temps et puisqu'on avait le droit de travailler avec le modèle ON/OFF, je l'ai fait. En effet, sur chaque nœud envoyant du trafic j'ai placé un agent pareto-i-j avec une moyenne du temps burst (ON) 500ms et du temps idle (OFF) 500ms, dans les périodes ON j'envois du trafic udp, ainsi j'ai placé des agents Null à la réception.

Chaque volume était donc diviser en 2 grandes partions 20 %+-(10%) de volume tcp, et 80 % +-(10%) de volume pareto.

La partition tcp étant elle même divisée en toutes petites partions de taille tirée aléatoirement suivant différentes loi de distribution. Ainsi, je prenait les paramètres des distribution de sorte à avoir dans l'ordre des milliers de partition par volume tcp. Puis pour chaque partition tirée, un temps de départ aléatoire est pris selon différentes distribution lui aussi.

Le scénario se déroulait donc de la sorte :
$ns at t ''$tcp_src-i-j send v''
où v = volume tiré aléatoirement
t = temps de départ tiré aléatoirement entre 0 et 300s

1-Choix des paramètres

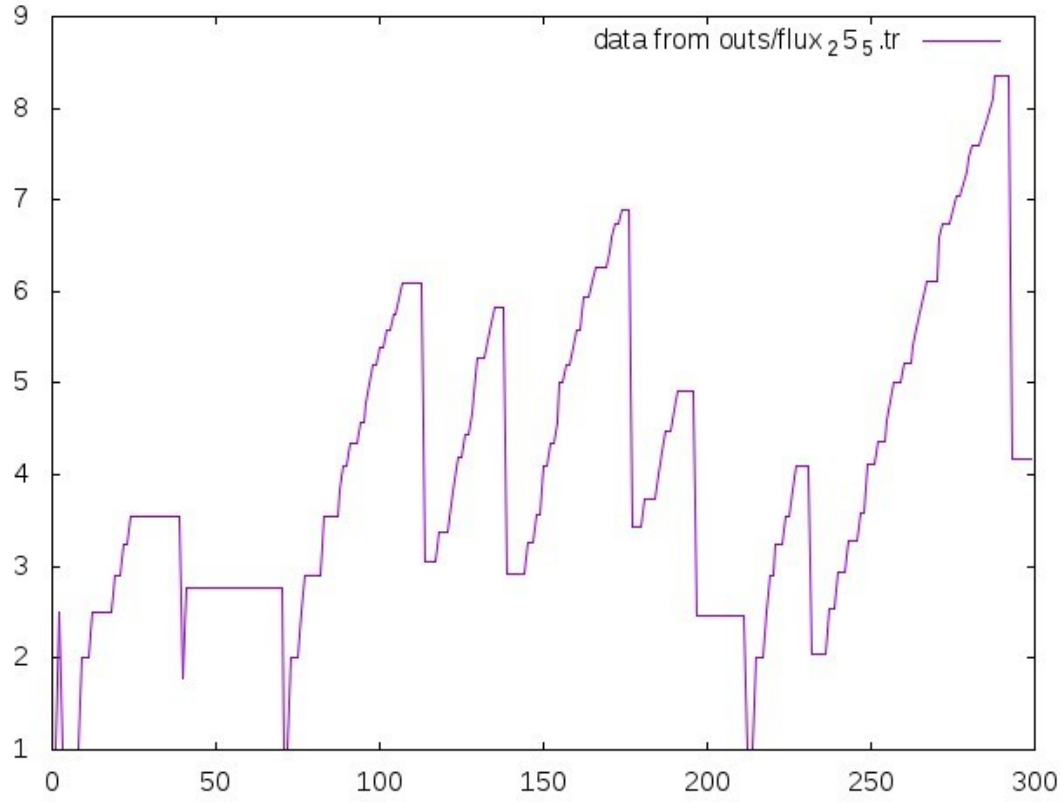
J’avais essayé de choisir des paramètres qui se rapprochaient le plus de la réalité, et qui me permettaient de ressortir les résultats demandé.

Pour faire, j’ai opté pour des implémentations Tahoe, Reno et Vegas de TCP vu que c’est parmis les versions tcp les plus utilisées. Ainsi que Cubic, qui me semblais être plus efficace sur les LFN. En un premier lieu j’ai essayé de créer des agents ftp sur les agents tcp, vu que les ports 20 et 21 sont les ports les plus utilisés en général. Mais cela ne changeait pas beaucoup les résultats obtenus avec juste des ‘tcp send’.

Les temps de départs des paquets suivaient d’abord une loi uniforme, cela permettait d’étaler les instants d’envoi uniformément sur les 300 secondes. Mais cela impliquait aussi moins de congestion sur quelques flux voir aucune congestion dans certains. Après, j’ai choisi de mettre une loi exponentielle. Ainsi je pouvais bien simuler les période où il y avait peut de trafic dans le réseau et les moment ou il y en avait plus. Mais avec une loi exponentielle les périodes warm-up était souvent en fin de simulation, et le warm-down était donc au début de la simulation. Or pour centrer la période du warm-up sur l’intervalle [0,300s] et créer plus de congestion j’ai opté pour une distribution gaussienne des instants de départs.

J’avais essayé dans un premier temps de faire une distribution uniforme pour produire dans les 1500 « petites » partitions par volume tcp. Cela produisait des partitions ayant ± la même taille . Alors que dans des réseaux réalistes le nombre de grande data est nettement plus petit que celui des data de taille moyenne rou petite. Pour faire, j’ai choisi une distribution zipf de shape 1,1.

Loi Uniforme



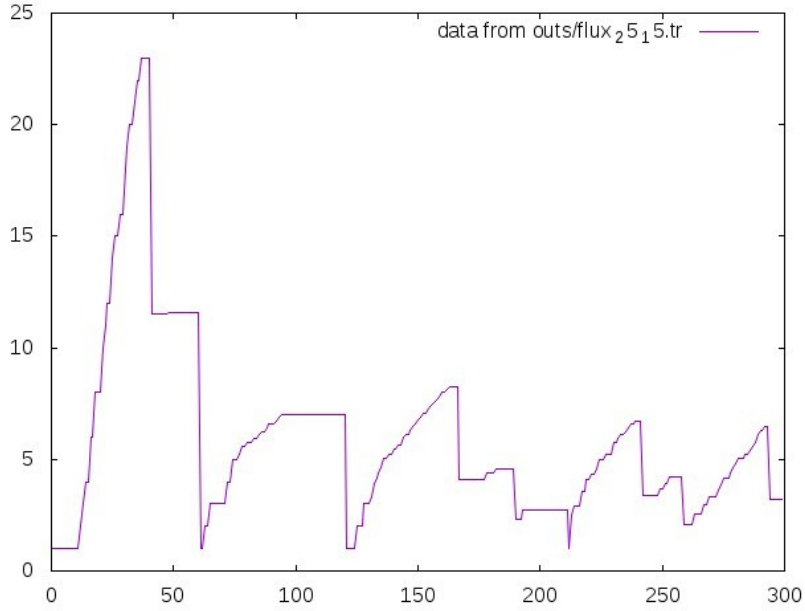
Conditions de simulation :
Distribution des temps de départs : Loi uniforme
Distribution des subdivisions : Loi uniforme

La figure représente l'évolution de la fenêtre de congestion **cwnd** du flux **25→15** en fonction du **temps.** On remarque bien le comportement de tcp Reno sur le graphe. La taille de la cwnd se divise par deux à chaque perte puis début de la phase de la Congestion Avoidance. J'avais choisi le graphe de ce flux **25→15** car il était parmi ceux où il y avait le plus de congestion.

<u>Remarque</u>

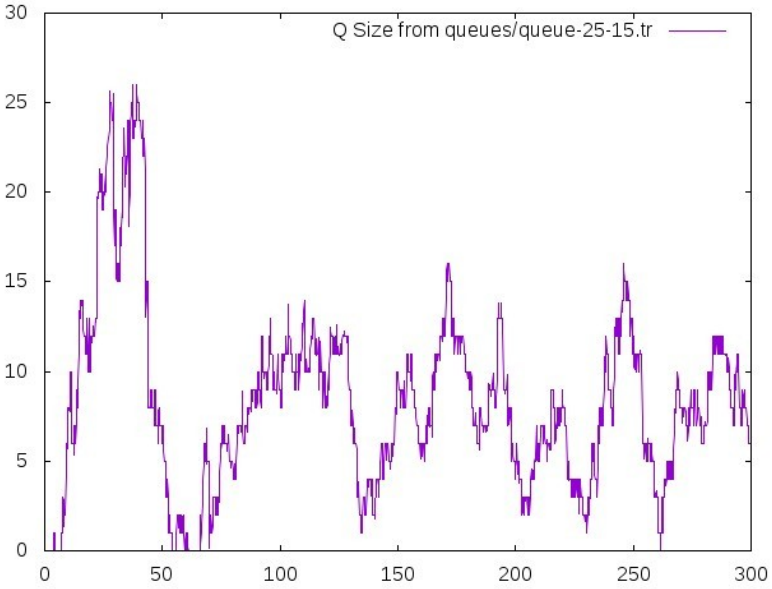
Parfois j'avais remarqué que la Congestion Avoidance ne commençait pas instantanément après que la taille de la **cwnd** est mise à moitié. Ceci peut être dû au fait que les temps de départ des partitions ne sont pas continus, ainsi il y a des périodes de temps ou il n'y a pas de transfert de données, tout a été envoyé et le temps de départ du la prochaine partition n'est pas encore arrivé, ainsi la taille de **cwnd** reste constante pendant cette interval (ex : alentours de 50s, 200s)

Départ Uniforme / Partitions Zipf



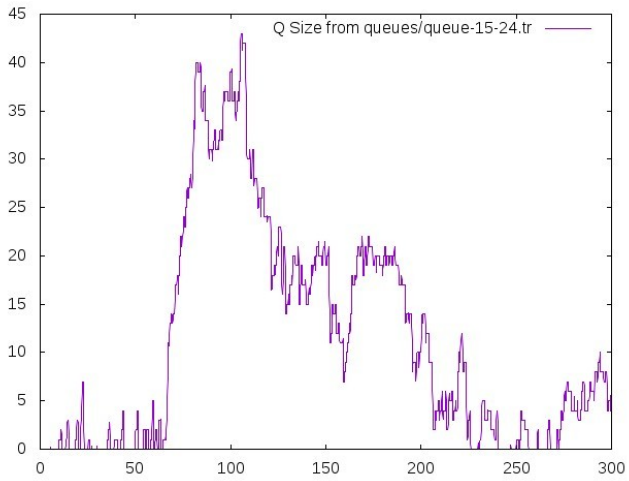
Conditions de simulation :
Distribution des temps de départs : Loi uniforme
Distribution des subdivisions : Loi zipf

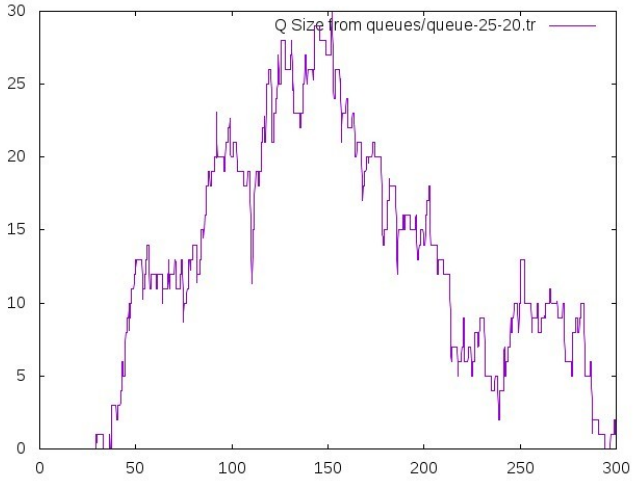
La figure au dessus représente toujours l'évolution de la fenêtre de congestion **cwnd** du flux **25→15** en fonction du **temps**. Mais cette fois-ci en changeant la loi de distribution des subdivisions. On remarque bien que la fenêtre de congestion s'incrémente au début vu qu'il y a un grand envoi de données au début de la simulation puis la taille de la **cwnd** se stabilise sur l'envoi des autres données de taille moins grande.

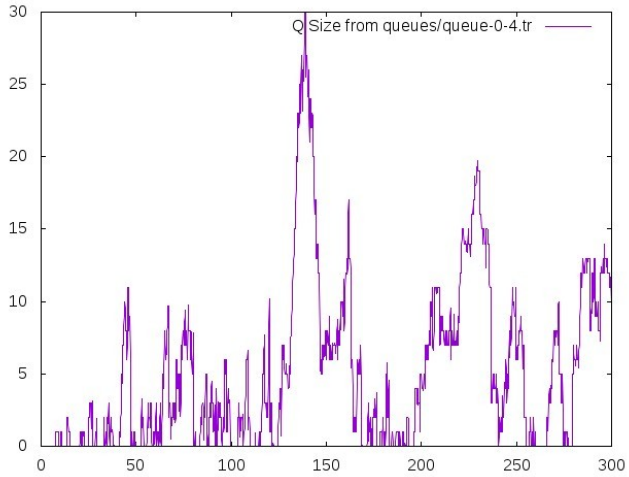


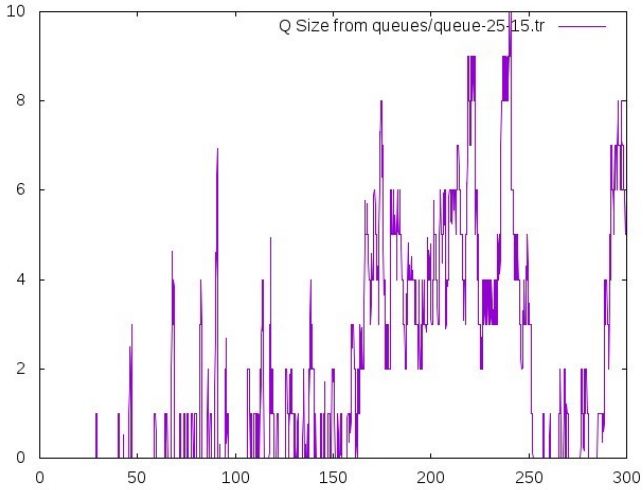
La taille de la file d'attente représente bien les périodes où il y a un grand nombre de paquets qui est envoyé, or au début de la simulation un grand nombre de paquets est envoyé, ce qui inplique un pique dans la taille de la file plus grand que les autres piques où un nombre de paquets moins important de paquets est envoyé.

Départ Gaussien / Partitions Uniforme









Conditions de simulation :
Distribution des temps de départs : Loi Normale, Moyenne = 150 (Temps de Simu/2) , Variance =75
Distribution des subdivisions : Loi Uniforme

Selon les figures des tailles Queue size dans les différents liens. On remarque bien les tailles maximales de ces files sont atteintes au alentours de 150s. On Distingue bien les périodes warm-up, où les file d'attente prennent une valeur maximale, se situent dans cet intervalle, et les périodes warm-down, où la taille des files est moins grande, se situent au début et à la fin de la simulation. En effet, le fait que les temps de les temps de départs suivent une loi gaussienne de moyenne 150, permet d'avoir plus d'envoi de paquets au alentours de cette valeurs par les différentes sources, donc le nombre de paquets « vivants » dans le réseau est plu grand dans cet intervalle, les files sont plus remplies et on a plus de perte de paquets.

II- Versions TCP

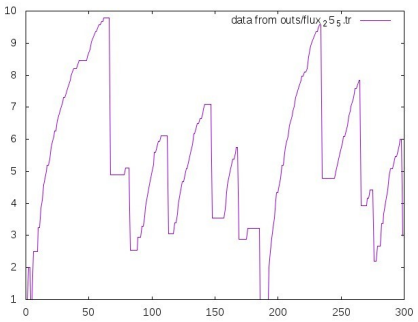
Comportement avec différentes versions TCP

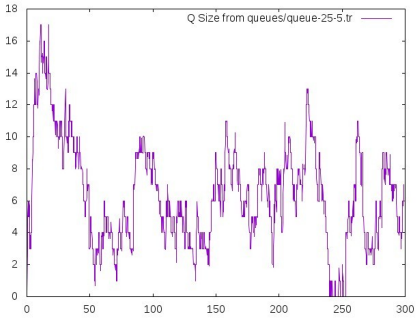
Conditions des simulation :

Distribution des temps de départs : Loi Normale, Moyenne = 150 (Temps de Simu/2) , Variance =75

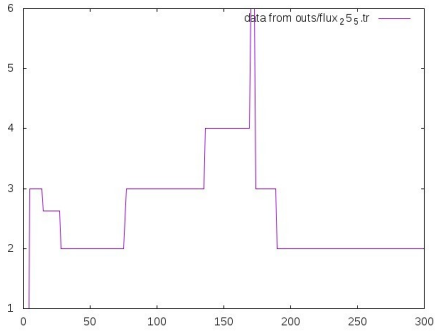
Distribution des subdivisions : Loi Uniforme

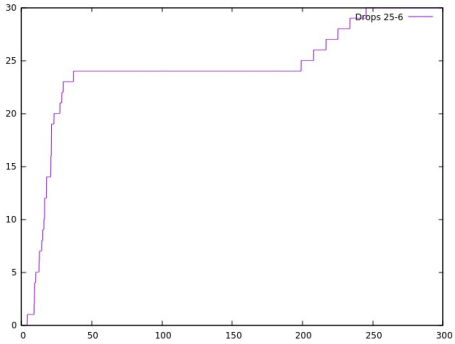
TCP NewReno



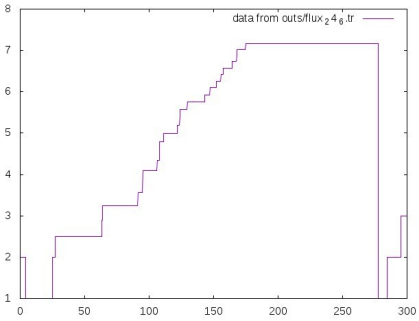


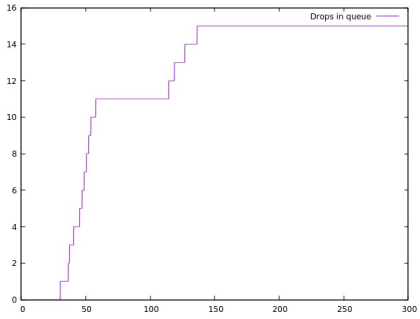
TCP Vegas





TCP Cubic





Synthèse

Le comportement de la **cwnd** dépend bien-sur de la version implémentée de tcp. Or j'ai essayé de testé l'implémentation cubic qui semblait avoir le moins de congestion sur les flux, et le moins de perte de paquets sur l'ensemble des liens. On voit bien selon les figures, la croissance de la fenêtre de congestion selon le max-probing à partir des alentours de l'instant t= 50s. Le nombre de paquets perdu en effet est aussi inférieur au autre versions tcp.