Projet : Implémentation d'un protocole de type Go-Back-N avec contrôle de la congestion

Maazouz Mehdi, Zielinski Pierre

May 15, 2017

Annee Academique 2016-2017

Contents

1 Introduction

Dans le cadre du cours de Reseau, nous avons eu comme objectif d'implémenter un protocole de type Go-Back-N avec un contrôle de la congestion de type Reno au sein d'un simulateur fourni par les professeurs. Le tout était à effectuer en Java.

2 Lancement

Lorsque vous voulez lancer le projet, vous pouvez définir plusieurs variables qui influeront sur le programme.
Tout d'abord, en première position, vous pouvez définir le nombre de paquets que vous voulez envoyer en première variable.

En deuxième position vous pouvez aussi définir la position de depart du treshold.

De plus, en troisième position, si vous voulez tester plus aisément la perte de paquets, vous pouvez définir le pourcentage de perte des paquets sur le lien. (il doit être compris en 0 et 100).

Enfin, dernière position, vous pouvez aussi choisir le pourcentage de perte de ACK(il doit être compris en 0 et 100).

Pour plus de facilité lors de l'Implémentation, à partir du moment où vous souhaitez changer une de ces variables, veuillez indiquer 0 pour toutes les variables précédentes (exemple si vous voulez juste changer le pourcentage de perte de paquets vous devez lancer le programme suivi de 0,0,20).

3 Utilisation du SenderProtocol et du GoBackNProtocol dans d'autre simulation

Le SenderProtocol et le GoBackNProtocol peuvent être utilisés dans d'autres simulations. Ils doivent cependant être lancé ensemble car ils fonctionnent ensemble. Une fois, le SenderProtocol et le GoBackNProtocol créé, et rajouté dans un ip listener, vous pouvez lancer la communication entre les deux en lançant la méthode "launch" du SenderProtocol. Après la méthode launch lancée vous pouvez rajouter des messages de deux façons au SenderProtocol. Soit vous pouvez envoyer une ArrayList de messages (sous forme d'entier de 32 bits) avec la méthode "addMessageTosend", soit rajouter directement une ArrayList de "PayloadMessage" mais ceux-ci doivent alors déjà avoir leur bon numéro de séquence ou alors ils vont créer des gestions de timeout à l'infini. Une fois tout les messages envoyer à l'instance de SenderProtocol vous devez lui signaler que vous avez fini en laçant la méthode "end" qui signalera que vous ne voulez plus envoyer de messages et qu'il peut s'arrêter une fois qu'il a envoyer tout les messages.

4 Lecture des données d'une simulation

Durant une simulation deux sortes de données vont être créée. Il y aura les données transférées sur le fichier "log.txt" et les données affichées dans le terminal. Les données du fichier texte permettront si elles sont utilisée dans "gnuplot" de voir l'évolution de la taille de la fenêtre d'envoi par rapport au temps et ainsi d'observer visuellement le slow start , l'additive increase, les 3 ACK dupliqués et les time out. L'affichage du terminal montrera des informations plus précise comme la valeur actuel du timer, la taille de la fenêtre d'envoi et la position du curseur se trouvant à l'intérieur, les messages envoyés , les acks envoyés, ... Ces informations sont plus précise et permettent de mieux observer l'évolution du programme. De plus la gestion de 3ACK dupliqué ou de time out y est signalé par des "/!\".

5 Problèmes rencontrés

Tout au long du développement, plusieurs problèmes ont été rencontrés.
Afin de déterminer le temps de RTT d'un paquet, nous avions décidé d'utiliser un paquet test composé d'un numéro de séquence -1. Cependant, due à notre implémentation, le numéro de séquence est transformé en binaire. Ce qui a causé quelques problèmes étant donné que l'entier était négatif. Nous avons donc décidé que le numéro de séquence 0 serait notre paquet test.

Suite à une discussion avec les professeurs, nous avons choisis d'exclure la probabilité de perte sur le paquet test. Ce qui permet de recevoir un ACK(0) sans aucune ambiguïté.

6 Conclusion

Plusieurs objectifs étaient à atteindre lors de la réalisation de ce projet. Tout d'abord, nous avons dû implémenter le protocole de type Go-Back-N, nous avons également implémenter le "pipelining" permettant l'envois de plusieurs paquets. De plus , nous avons également implémenté la congestion comme demandé par les professeurs.