Rapport du projet de réseau : Implémentation d'un protocole de type Go-Back-N avec contrôle de la congestion

Houyaux Sacha, Marco Salemi

9 mai 2016

1 Introduction

Pour ce second projet de réseau, nous avons dû réaliser, à l'aide du bq-simulator, un protocole de type Go-Back-N avec du contrôle de congestion. Nous allons expliquer dans ce rapport les différentes stratégies adoptées lors de ce projet, afin d'améliorer la compréhension de celui-ci.

2 Classes implémentées

Notre classe principale est "Demo.java" c'est cette classe qu'il faudra exécuter afin de lancer notre application.

Nous avons également deux autres classes qui représentent la partie applicative du projet, ces classes sont "AppSender.java" et "AppReceiver.java".

Nous avons aussi implémenté notre propre lien non-fiable qui perd des paquets avec une certaine probabilité (80 % des paquets passent), elle s'appellent "UnreliableLink.java". Elle étend de la classe "Link.java" implémentée dans le bq-simulator.

Nous avons une classe "Scenario.java" qui permet de créer des scénarios, le fonctionnement de la classe est expliquée en commentaire du code.

On a également deux classes pour les messages, une classe pour les messages "Go-Back-N", et une autre pour les acquittements.

Enfin, nous avons deux classes pour l'implémentation du protocole, car l'implémentation pour le receveur et l'envoyeur est différente, donc nous avons choisi de le diviser en deux classes pour des soucis d'implémentation.

N.B. : Vous pouvez changé de scénario en changeant "scenar" dans "Demo.java"

3 Implémentation de Go-Back-N

Comme expliqué précédemment, nous avons choisi d'implémenté le protocole en deux parties. Une partie pour l'envoyeur et une partie pour le receveur.

En ce qui concerne l'envoyeur, selon le scénario, l'application enverra un certain nombre de message à un intervalle de temps donné par le scénario. Ces messages seront mis dans un buffer (de taille 1000), afin de délivré au fur et à mesure les messages avec le protocole Go-Back-N.

Le protocole au niveau de l'envoyeur fonctionne comme suit : quand un paquet va être envoyé, on regarde si le paquet est utilisable, c'est-à-dire qu'il est dans la "sliding window" et qu'il n'a pas déjà été envoyé, si c'est le cas, on l'envoie et on augmente le la variable contenant le numéro de séquence du prochain paquet à envoyé.

Quand le protocole reçoit un paquet, on augmente le numéro de séquence du prochain paquet à envoyer, si le ce numéro est égale au numéro de séquence du prochain paquet à envoyé, alors on lance on arrête le timer, sinon on le lance. Et on envoie le paquet avec le numéro de séquence du prochain paquet à envoyé.

Dans le cas d'un timeout, on renvoie simplement tout les paquets dans la "sliding window".

Dans la partie du receveur, il regarde si le numéro de séquence du message correspond à celui qu'il attend, si c'est le cas il envoie un acquittement avec le numéro de séquence attendu et augmente ce numéro d'une unité, sinon il renvoie un acquittement mais il n'augmente pas son numéro de séquence.

4 Contrôle de congestion

Au niveau de contrôle de congestion, nous avons quatre cas possibles.

Le premier cas est si la taille de la fenêtre de congestion est plus petite qu'un certain seuil (ici ce seuil est de 16), alors l'envoyeur est en slow-start et donc augmente la taille de sa fenêtre de une unité, une fois ce seuil dépassé, on passe en additive-increase.

L'additive increase consiste à augmenter de une unité la fenêtre de congestion tant que il n'y a pas de signe de congestion.

Si on a trois acquittements dupliqués, on diminue par deux la fenêtre de congestion.

Enfin, en cas de timeout, on revient à la taille 1 + slow-start.

5 Compilation et exécution

Vu que nous avons utilisé la libraire JFreechart afin de pouvoir tracer le graphique demandé dans l'énoncé, l'exécution et la compilation se font avec les commandes suivantes.

Compilation : javac -classpath jfreechart-1.0.19/lib/jfreechart-1.0.19.jar :jfreechart-1.0.19/lib/jcommon-1.0.23.jar *.java

Exécution : java -cp . :jfreechart-1.0.19/lib/jfreechart-1.0.19.jar :jfreechart-1.0.19/lib/jcommon-1.0.23.jar Demo

Vous pouvez télécharger la librairie à l'adresse suivante :
http ://www.jfree.org/jfreechart/download.html

6 Conclusion

En conclusion, ce projet nous a permis de mieux comprendre le fonctionnement du protocole Go-Back-N et de la congestion, malgré tout, nous avons rencontré plusieurs difficultés comme la compréhension de certaines fonctionnalités du bq-simulator, mais une fois celles-ci comprises l'utilisation du simulateur fût facile.