Rapport de projet

Go-back-N & Congestion

Réseau I

Membres du groupe:
SAHLI Yacine
HUYLENBROECK Florent

Année Académique 2017-2018
Bachelier en Sciences Informatiques

Faculté des Sciences, Université de Mons

1 Implémentation

1.1 L'application

L'application de base est implémentée dans le package *reso.examples.gobackn* comme demandé dans les consignes.

1.2 Génération d'évènements aléatoires

Problèmes simulés aléatoirement (voir "Paramètres modifiables" pour les probabilités de ceux-ci) et comportement de l'application lorsqu'ils surviennent:

- **Ajout d'un court délai à certains ACK.**
  Le thread est endormi pendant un nombre paramétrable de millisecondes avant l'envoi de l'ACK.
- **Ajout d'un délai supérieur à la valeur du timeout à certains ACK.**
  Le thread est endormi pendant un nombre non-paramétrable de millisecondes (correspondant au TIMEOUT_ DELAY dans *reso.examples.gobackn.GoBackNSenderApp.java* avant l'envoi de l'ACK).
- **Non-envoi de certains ACK.**
  L'envoi de l'ACK correspondant est annulé.
- **Perte de certains paquets.**
  L'application agit comme si elle n'avait rien reçu.
- **Certains paquets seront traités comme étant corrompus.**
  L'application renvoie l'ACK correspondant au dernier paquet vérifié.

1.3 Go-back-N

L'envoyeur envoie un paquet toutes les 100ms tant qu'il reste de la place dans la fenêtre et n'attend pas les ACK. La taille initiale de la fenêtre est de 1 (paramètrable). Le destinataire reçoit et traite les paquets un par un. Le numéro de séquence du paquet attendu est gardé dans une variable et un ACK avec le numéro du paquet est envoyé si ce paquet ainsi que tous les précédents ont bien été reçus.

1.4 Contrôle de congestion

- **Additive increase :** une fois le seuil du slow start atteint (slowStartThreshold), l'additive increase le remplace. À chaque fois qu'une fenêtre entière a été "acknowledgée" la taille de la fenêtre est incrémentée de 1.
- **Slow Start :** La taille de la fenêtre est incrémentée de 1 à chaque ACK reçu. Une fois le seuil (slowStartThreshold) atteint, la stratégie utilisée devient Additive increase. Le seuil initial est paramétrable (voir "Paramètres modifiables").
- **Packet Timeout :** On considère qu'un paquet est perdu après 1 seconde sans avoir reçu son ACK (paramètrable). En cas d'ACK non reçu ou reçu après 1 seconde à partir de l'envoi du paquet, un timeout est lancé et s'applique la procédure suivante:
  - Le slowStartThreshold est maintenant égal à la moitié de la taille de la fenêtre.
  - La taille de la fenêtre est réinitialisée à 1.
  - Le slow start est activé.

- **Triple ACK dupliqué :** Cette situation arrive en cas de paquet corrompu ou de paquet n'arrivant jamais au receveur. Celui-ci, attendant un certain paquet, renverra jusqu'à 3 fois le même ACK pour signaler que le paquet suivant n'a toujours pas été reçu. Quand l'envoyeur reçoit 3 ACK dupliqués, la taille de la fenêtre et le slowStartThreshold sont divisés par 2, le slow start est également désactivé et on passe en additive increase.

2 Utilisation de l'application

2.1 Lancement de l'application

Pour démarrer l'application, lancer le fichier Main.java du package *reso.examples.gobackn* .
Pour cela, se placer dans le dossier src/ et lancer la commande suivante :

javac reso/examples/gobackn/Main.java && java reso.examples.gobackn.Main

2.2 Paramètres modifiables

- *reso.examples.gobackn.GoBackNReceiverApp.java*
  - NUMBER_ OF_ EVENT et les variables commençant par PROB_ permettent de modifier les probabilités que des évènements inattendus se produisent, selon la formule :

$$\frac{\text{PROB\_ EVENT}}{\text{NUMBER\_ OF\_ EVENT}}$$

    Exemple : Si l'on souhaite que 5% des ACK ne soient pas envoyés,
    PROB_ ACK_ NOT_ SENT=5; et NUMBER_ OF_ EVENT=100;

    La variable NUMBER_ OF_ EVENT sert principalement à augmenter/réduire toutes les probabilités en une seule fois.
  - SMALL_ DELAY_ RANGE_ MIN et SMALL_ DELAY_ RANGE_ MAX correspondent à l'intervalle dans lequel seront choisis les petits délais aléatoires avant d'envoyer certains ACK.

- *reso.examples.gobackn.GoBackNSenderApp.java*
  - TIMEOUT_ DELAY modifie le délai avant un timeout.
  - PACKET_ SENT modifie le nombre de paquets à envoyer.
  - slowStartThreshold modifie le seuil initial.
- *reso.examples.gobackn.Main.java*
  - LINK_ SIZE modifie la taille du lien entre les interfaces.
  - DEBIT modifie le débit.

2.3 Comportement de l'application

Une fois démarrée, une série de logs s'afficheront dans la console, détaillant étape par étape l'échange de paquets au sein de l'application.
Un fichier *log.txt* sera créé au même niveau que *reso.examples.gobackn.Main.java* contenant les données nécessaires au plotting des résultats. (*Nous assumons que l'utilisateur de l'application voulant mettre les résultats en graphique sait se servir d'un logiciel de plotting*).