Université de Mons
Faculté Des Sciences
Département d'Informatique

Projet réseau - Selective-repeat

Professeur :
Bruno Quoitin
Assistants :
Jeremy Dubrulle

Auteurs :
Théo Godin 210170
François Vion 210809
Groupe :
23

Année académique 2021-2022

Table des matières

1 Construction et exécution

La compilation du projet se fait en exécutant la commande :

```
$ javac reso/examples/selectiverepeat/Demo.java
```

Et l'exécution :

```
$ java reso.examples.selectiverepeat.Demo x
```

(où x est le pourcentage de paquets perdus).
Une fois l'exécution terminée les fichiers log et csv sont générés à la racine du projet.

2 Applications

Nous avons décidé de suivre l'exemple ping-pong et de créer deux classes application, une qui envoie les données et une autre qui les reçcoit. L'app sender recoit les données à envoyer en paramètre de son constructeur et passe ensuite ces données à la méthode sendMessage du protocol. Tandis que l'app receiver, au lancement de l'application, ne fait qu'écouter.

3 Selective-repeat

3.1 Envoi d'un message

Un message peut être envoyé par une application via la méthode sendMessage de notre protocol. Cette méthode va séparer ce message en plus petits messages de 8 caractères qui vont pouvoir ensuite être envoyés en pipelining.

3.2 Pipelining

La méthode pipeliningSend s'occupe d'envoyer des segments tant que la window n'a pas atteint sa taille maximale.

3.3 Envoi d'un segment

Afin d'envoyer un segment, la méthode send envoie ce segment à la destination, l'ajoute à l'arrayList représentant la window et finalement crée et démarre un timer afin de gérer les timeout pour ce segment.
Du côté du receveur, la méthode sendAck permet d'envoyer un ack de même sequence number que le segment reçu.

3.4 Timeout et RTO

Si un timer arrive à sa fin, le segment associé est ré-envoyé et un nouveau timer est créé. La durée du timer est calculée de la même manière que vue au cours. Si l'on a un timeout, la durée de celui ci est doublé et si le receveur reçoit un ACK, les valeurs pour R, SRTT, devRTT et RTO (durée du timer) sont recalculées avec les formules indiquées dans le cours.

3.5 Réception d'un ack

4 Contrôle de congestion

Afin de gérer le contrôle de congestion, la taille de la fenêtre est modifié au cours du processus afin d'avoir une taille optimale tout au long de l'envoi. Cela se fait lorsque l'envoyeur reçoit un ACK ou lors d'un timeout. Lors de la réception d'un ACK, on double la taille de la fenêtre si celle-ci est inférieure au sstresh (fixé à 100 au départ). Si elle est inférieure, on l'incrémente de 1. Lors d'un timeout, la taille de fenêtre est fixée à 1 et le sstresh est fixé à la moitié de la dernière valeur de fenêtre. Dans le cas où la fenêtre avait une taille de 1, le sstresh est fixé à 1.

5 Bugs rencontrés

Nous avons remarqués grâce aux logs que le receveur recevait des paquets ackés. Après avoir testé chaque méthode envoyant des messages, la source du bug n'a toujours pas été découverte car aucune d'entre elles n'envoie de message acké. Ce bug a donc été contourné en ajoutant au protocol l'attribut actor permettant de savoir si l'application courrante est l'envoyeur ou le receveur, ce qui est testé lors de la réception d'un message.