Ecole Polytechnique de Louvain | UCLouvain

LINGI1341 - Reseaux informatique

Project: Truncated Reliable Transport Protocol

Auteurs :

CLAES Alexandre	alexandre.a.claes@student.uclouvain.be	(6057-20-00)
POLLET Yann	yann.pollet@student.uclouvain.be	(6081-20-00)

Professeur :

BONAVENTURE Olivier

Année Académique 2020-2021

1 Implementation

1.1 Window

Le sender et le receiver ont tous les deux une array de taille 32 représentant la window et permettant de stocker des paquets. Le fonctionnement de ces arrays est simple, nous déterminons la place du paquet dans l'array grâce au au numéro de séquence modulo 32 (le modulo 32 a l'avantage de faciliter l'utilisation d'une liste circulaire, contrairement à un modulo 31). Le début de la window coulisse en fonction des paquets traités (il peut arriver que la window commence à l'indice 5 et termine à l'indice 4).

1.2 Receiver

Le receiver, lors de son initialisation, attend la connexion d'un sender. Lorsque ce dernier est connecté, le receiver va simplement renvoyer des paquets ACK ou NACK en fonction des paquets DATA reçus. Le receiver stock les paquets en attente d'etre affiché et les supprime dès qu'ils ont été affichés dans la partie standard. Lorsqu'il reçoit un paquet non tronqué, le receiver renvoit un paquet ACK avec le next-seqnum qu'il a besoin et mets dans le champ timestamp, le timestamp qu'il a reçu afin de laisser la possibilité au sender de calculer le round-trip-time de son paquet. Ajoutons que lorsque le receiver reçoit le paquet contenant l'EOF, il renvoie un paquet ACK avant de terminer son processus.

Concernant les cas d'erreurs possible, le receiver aura plusieurs réactions déterminées. Si le paquet reçu est hors de la séquence de la window, le receiver l'ignorera et renvoiera juste un paquet ACK contenant le next-seqnum requis. Si le paquet reçu est tronqué, il renvoie un NACK avec comme next-seqnum le seqnum reçu dans le paquet.

1.3 Sender

Lors de son initialisation, le sender crée un socket le liant vers le receiver. Après cela, le sender va continuellement écouter entre son entrée (d'où vient les données qu'il doit envoyer) et ce socket. Le sender va également stocker les paquets qu'il a envoyé au receiver tant qu'il n'a pas reçu de confirmation du receiver de la bonne reception de celui-ci afin de pouvoir les renvoyer en cas de timeout. Dès que le sender a envoyé le paquet contenant le EOF, il attends le packet ACK correspondant avant de terminer son processus.

Le sender gère les erreurs de différentes manières.
Tout d'abord, concentrons-nous sur le cas où le sender atteint un time-out concernant un de ses paquets. Le timer est initialisé à 5 secondes et est ensuite recalculé en fonction du round trip time du premier paquet. Pour pouvoir vérifier régulièrement les timers, nous utilisons un poll avec un timeout afin de sortir de l'appel bloquant si aucun file descriptor n'a été appelé. Ensuite, quand le sender reçoit un packet NACK, il vérifie le seqnum de ce paquet et le renvoie directement au receiver.

2 Tests

Expliquer la stratégie de test

3 Performances

Faire les graphes et expliquer les endroits où ça peut prendre du temps

4 Conclusion

Nous tenons à signaler que nous avons beaucoup travaillé en pair programming (donc simultanément sur les memes fichiers). C'est pour cela que les commits ont été fait par la meme personne sur les gitlogs.