LINGI1341 : Réseaux informatiques
Project 1 : Protocole de transfert fiable

Damiano-Joseph Siciliano
Noma : 3039-10-00
Benjamen Daubry
Noma : 3079-10-00

1. Sender

Les variables globales clé qui communiqueront entre les thread sont les suivantes :

pkt_t * window[32] : Il s'agit du buffer dans lequel seront ajoutés/suprimmés des paquets au fur et à mesure
struct timeval timeBuf [32] : il contient le temps exact

int nProd : le nombre de paquet produit mais qui n'ont pas été encore envoyé
int nSend : le nombre de paquet qui POURRAIT être encore envoyé (initialisé donc à 16)
int newPKT : représente le numéro de paquet envoyé le plus recement
int oldPKT : représente le numéro de paquet envoyé le plus anciennement

Il y a 4 threads permettant d'obtenir un protocole type selective repeat :

a) Producer

Il va simplement rajouter des paquets dans l'ordre (numéro de séquence) en revenant au début du buffer si nécéssaire. Il ne s'arrête que lorsque le contenu du buffer sur lequel il compte rajouter des paquets est déjà rempli ou simplement losque le fichier a été lu entierement. A chaque rajout de papquet il incrémente nProd de 1 (protection Mutex)

b) Sender

Tout comme le producer, il va se déplacer en bouble dans le buffer a partir du début et enverra des paquets lorsqu'il saura que le producer à mis des paquets (nProd > 0) et que la limite de paquets à envoyé n'est pas atteinte (nSend > 16)

A chaque envoie, il décrémente nProd et nSend de 1 et incrémente newPKT de 1 et associe un timer à cette envoie dans timeBuf (protection Mutex).

c) Listen

Il s'agit du thread recevant les ACK.

Si il recoit un NACK, il va simplement renvoyé le paquet et remettre un nouveau timer

Si il recoit un ACK, on verifie si il est dans la fenetre d'envoie. Si tel est le cas, on sait qu'il n'y a plus besoin du timer qui va donc s'arreter de fonctionner (via un mutex), permettant ainsi de supprimer ce paquet et son timer associé en étant sur qu'entre temps il n'y a pas une demande de renvoie (simple perte de performance dans le cas échéant).

A noter que si l'ACK est destiné à celui le plus à gauche de la fenêtre, nous allons la decaller de 1 + un bonus selon le nombre d'ACK recuit à la suite de ce dernier. Ce chiffre sera incrémenté dans oldPKT et nSend (en effet on peux envoyé d'autre paquet).

d) CheckTime

Il va constament regarder le timer associé au plus ancien paquet en le coparant au temps actuel, si il depasse les 3 secondes. On entre dans la section critique vu précédement pour renvoyé le paquet et mettre à jour le timer.

Problème actuel : il connait le plus ancien paquet par rapport au positionement de la fenetre, il il recoit un NACK (14), puis un NACK(12), il se peut qu'il regarde le 12 en premier.

Solution : afin de prioriser les timers les plus anciens, nous allons creer une liste FIFO accompagner d'un nouveau MUTEX qui bloquera le check si l'on rajoute/enleve/deplace les elements de cette liste.

Exemple d'un envoie parfait sans NACK : 1->2->3 , 2->3->4

Exemple d'un envoie avec NACK 3 puis NACK 5 : 1->2->3->4->5 , 1->2->4->3->5

2. Receiver

Les variables clé qui communiquent entre les threads sont les suivantes :

pkt_t * window[32] : Comme pour le sender, il s'agit du buffer dans lequel seront ajoutés/suprimmés des paquets au fur et à mesure
int nProd : le nombre de paquet recu mais qui n'ont pas été écrit
int oldN : le numero de sequence pour la position la plus a gauche de la fenetre de reception (init à 0)
int newN : le numero de sequence pour la position la plus a droite de la fenetre de reception (init à 15)

a) Writer

Il va simplement parcourir en boucle le buffer (puisque ce dernier sera remplis dans l'ordre), mais il restera en pause tant que nProd ne sera pas superieur à 0 (inutile de chercher si c'est NULL). A chaque écriture on décrémente donc de 1 nProd et on fait avancer oldN et newN de 1 (protéger par un MUTEX).

b) Listener

Si il recoit un mauvais paquet, il renvoie simplement un NACK.
Si c'est un bon paquet, si son numero de sequence est equivalent à oldN, on decalle simplement la fenetre de 1 + du nombre de numéro de sequence bien recu a la suite de cela (incrémétation oldN et newN), et decrémente nProd de ce meme chiffre (protection MUTEX).

Cependant si le le paquet est dans la fenetre de reception (entre oldN et newN), on va simplement le placer dans le buffer mais ne rien décrémenter et incrémenter car ce paquet n'est pas encore pret a être écrit.

Temps du Timer ?

Fixé à 3 secondes par défaut, nous prendrons le RTT moyen suite à de nombreux auquel au rajoutera un surplus de temps suffisamanet grand pour assurer a tout pris un bon envoie avant un reset de timer

Partie Critique de l'implémentation

Il s'agit du code appliqué entre tout les mutex cités précédemment qui toucheront au variable globale.

Stratégie de test

Nous n'avons malheureusement pas une implémentation fonctionnel des différentes fonctions qu'utilse les thread. Les tests n'ont donc pas pu etre fait.