TRAVAIL 1

PRÉSENTÉ À

M BOUCIF AMAR BENSABER

COMME EXIGENCE PARTIELLE

DU COURS

RÉSEAUX D'ORDINATEURS 1

PAR

TOMA ALLARY,

ETIENNE DELISLE,

BENOIT MATTEAU,

GABRIEL TREMBLAY,

ET SIMON VENNE

26 AVRIL 2021

Contents

Description du logiciel 3

Instructions spéciales 3

Description des fichiers. 3

S_lec : 3

S_ecr : 3

R_ecr : 4

L_ecr : 4

L_lec : 4

Description des classes 4

Les classes de paquet 4

La classe Constantes 4

La classe ConnexionTransport 5

La classe transport 5

La classe EntitéeReseau 5

Description du logiciel

Notre logiciel se veut une simulation de connexion et de transfert de données vers un ordinateur distant. Le système local et divisé en une couche transport et une couche réseau. La liaison de données et l'ordinateur distant sont simulés complètement par la couche réseau. Ces composantes seront expliquées plus en détail ultérieurement. Chacune de ces couches communique avec la couche qui lui est supérieure et la couche qui lui est inférieure. La couche transport lis une liste d'instruction dans un fichier, les passe ensuite à des threads de la couche réseau qui fabrique des paquets et les envois au distant qui est en fait simulé par réseau. Distant prends les paquets, inscrit la donnée reçue dans un fichier et répond adéquatement à la couche réseau, en plus de garder un log des transactions. Réseau décide ensuite de conserver ou fermer les connexions avec le distant au besoin, et indique à transport lorsqu'une connexion est fermée. Transport note aussi les ouvertures et fermetures de connexion dans un fichier.

Instructions spéciales

Il y a peu d'instructions spéciales pour exécuter le programme, il suffit de le lancer et d'attendre quelques secondes, tout se fait tout seul. On peut ensuite constater ce qui s'est passé à l'aide des différents fichiers.
Il est normal que le programme prenne une dizaine de secondes à s'effectuer, car il doit parfois attendre qu'un thread ne finisse.
Les fichiers textes son situés sous : INF1009_TPSESSION\INF1009_TPSESSION\bin\Debug\netcoreapp3.1\

Description des fichiers.

Les différents fichiers textes que nous utilisons sont les suivants :

S_lec :

Contient les différentes instructions que nous souhaitons envoyer au distant.
Les trames sont formées de deux parties; premièrement, un chiffre qui représentera le programme qui envoi la données, suivi de la donnée à transmettre. Cette donnée peut prendre trois formes : « DebutDesDonnees », pour signifier la demande de création d'une connexion, « FinDesDonnees », pour signifier la demande de fermeture d'une connexion, ou encore une chaîne de caractères quelconque qui représente la donnée à transmettre.

S_ecr :

Contient les logs de la couche transport. Les trames sont encore une fois séparées en deux, commençant par le numéro du programme, suivi du log en tant que tel.

Les deux opérations que note transport sont création d'une connexion, et terminaison ou encore échec d'une connexion.

R_ecr :

Ce fichier est propre à notre travail et sert de pipe pour transférer l'information entre les couches transport et réseau. Les données dans ce fichier sont sous la forme d'une suite de nombres de 0 à 255 séparés par des virgules. Ces nombres sont les bytes des paquets reçus par réseau qui doivent être communiqués à transport.

L_ecr :

Ce fichier correspond à ce que reçoit l'application sur la machine distante, Il n'inscrit que le contenu des paquets de data reçus, précédés du numéro de la connexion qui les a envoyés.

L_lec :

Ce fichier correspond au fichier log de la liaison de données. Le distant note toutes les transactions dans ce fichier. Il commence encore une fois par le numéro de la connexion, suivi de la transaction en question. Les différentes transactions possibles sont :

- Connexion établie
- Demande de déconnexion
- .....simulation de non reponse.....
- Paquet de données invalide reçu
- Paquet de données reçu suivi du numéro du paquet et du numéro du prochain paquet

Description des classes

Les classes de paquet

Tous les types de paquet sont représentés par une classe distincte qui hérite de la classe générale Paquet.

La classe Constantes

La classe constantes contient toutes les constantes nécessaires au bon fonctionnement du programme.

La classe ConnexionTransport

Cette classe est un objet qui représente le transfert de données entre la couche transport et la couche réseau. L'objet est constitué 3 bytes, contenant l'adresse source de la requête, l'adresse de destination, et le numéro de connexion.
L'objet contient aussi une queue de bytes qui contient l'information transférée d'une couche à l'autre.

La classe transport

La classe transport est une des deux classes principales de notre application.
Elle est lancée tout de suite au début du programme et commence par créer les fichiers text nécessaire au bon fonctionnement du programme.
Transport gère une table de connexionTransport sous forme de dictionnaire.
Transport va ensuite lire le fichier s_lec ligne par ligne depuis la méthode lectureFichier. Ces lignes sont ensuite passées à la méthode processLigne qui vérifie si une entrée dans la table des connexions existe déjà pour l'appelant, la crée si ça n'est pas le cas, ou ajoute la requête dans la queue si l'entrée existe déjà. C'est aussi cette méthode qui attribue les adresses de destinataires et de source.
Une fois ce processus terminé, Transport crée un thread par entrée dans la table des connexions. Chaque thread contient une instance de la couche réseau auquel on envoi le connexionTransport correspondant, et un thread qui lira le fichier R_ecr pour recevoir les retours de réseau. Transport attend finalement la fermeture des threads lancés.
Le thread chargé de lire le fichier R_ecr utilise la méthode lireReception qui accède au fichier à l'aide du sémaphore Ers_TO_ET_File lis toutes les lignes qui s'y trouvent à chaque passage et écrit les données reçues dans S_ecr.

La classe EntitéeReseau

La classe EntitéRéseau est la deuxième classe particulièrement importante de notre projet. Elle s'occupe de simuler la couche réseau et la liaison de données.
Premièrement, cette classe reçoit une queue de requête qui viennent de transport.
Pour chaque requête dans la queue, Réseau vérifie s'il s'agit d'une demande de connexion, d'une demande de déconnexion, ou de données brutes venant du demandeur.
La méthose procesTask s'occupe ensuite de créer les paquets en fonction de la nature de la demande et l'état des connexion.
EntitéRéseau refuse systématiquement toute connexion dont l'adresse source est un multiple de 27.
Si une chaine de données brute contient plus de 128 bytes, c'est aussi cette méthode qui la sépare en paquets de 128 bytes (plus les deux bytes

d’identification). Lors d’une connexion ou d’une déconnexion, c’est aussi cette méthode qui revoit les paquets de confirmation vers la couche transport, via la méthode returnData.

Les paquets destinés à la couche de liaison de données sont dont ensuite passés à la méthode liaisonDonnées qui fait office de distant.
LiaisonDonnées passe les paquets de demande de connexion à la méthode establishConnexion qui refuse systématiquement les paquets dont l’adresse de source est un multiple de 19, mais ne retourne rien dans ce cas-là.
La méthode refuse aussi les paquets dont l’adresse source est un multiple de 13, mais retourne cette fois un paquet d’indication de déconnexion.
La connexion est établie dans tous les autres cas.
Les paquets de données seront, quant à eux, envoyés à la méthose receiveData.
Cette méthode ne retourne rien si l’adresse source est un multiple de 15.
De plus, si le numéro Ps du paquet reçu est équivalent à un nombre aléatoire contenu entre 0 et 7, la méthode retourne un acquittement négatif pour simuler un paquet endommagé.
Dans les autres cas, la méthode écrit le contenu des paquets de données reçus dans L_ecr à l’aide de la méthode writeData, reconstruit les trames de plus de 128 bytes, et retourne un byte contenant le numéro du paquet reçu et le numéro du prochain paquet attendu.