Rapport de projet - FTP pair à pair

Boncorps Robin & Daniel Gwendal

28 mars 2013

Chapitre 1

Introduction

Dans le cadre de nos TP de réseaux, nous avons du développer une application client/serveur. Le but de ce projet est de comprendre les principes de fonctionnement des sockets, bases de toutes applications réseaux réparties. C'est dans cette optique que nous avons choisi de développer une application d'échange de fichiers en pair à pair. Nous allons donc, dans le présent rapport, présenter les différents protocoles et moyens utilisés pour mener à bien ce projet. Nous nous intereserons, par la suite, à la modélisation de l'application pour finir par présenter les différentes performances de l'application.

Chapitre 2

Analyse

Dans cette partie, nous allons présenter succintement les différentes bases sur lesquelles s'appuie l'application.

2.1 FTP : File Transfer Protocol

FTP ou File Transfert Protocol est un protocol de communication destiné à l'échange de fichiers sur un réseau TCP/IP. FTP obéit à un modéle client-serveur, un client interrogant un serveur distant qui fonctionne sur la machine distante.

2.2 Pair à pair

2.3 Application

Chapitre 3

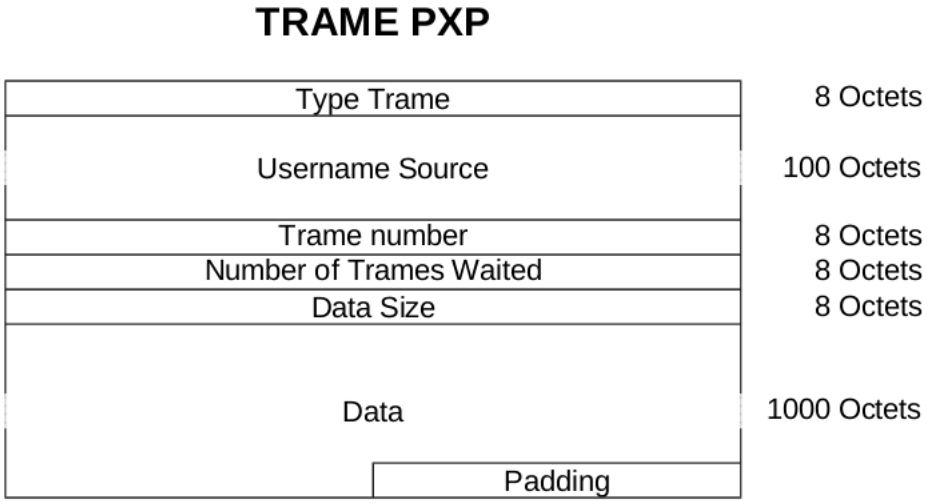
Modélisation

3.1 Architecture

3.2 PXP : Personal eXchange Protocol

L'application permet d'échanger des fichiers grâce à son propre protocole (de la meme famille que ftp). Ce protocole, que nous nommerons par la suite PXP pour Personal eXchange Protocol, permet le découpage/reconstruction d'un fichier entre 2 machines distante.

3.2.1 La trame PXP



Signification des champs :

- Type Trame : Le type de trame envoyé (nous détaillerons par la suite les différents types)
- Username Source : le nom de l'utilisateur qui envoi la trame
- Trame number : le numéro de la trame envoyée
- Number of Trame Waited : le nombre de trames attendues pour avoir tout le fichier
- Data Size : la taille effective des données comprises dans la partie Data
- Data : les données (une partie du fichier)
- Padding : 0 ajoutés pour remplir le champ Data

Type Trame

Chaque action effectuée sur une machine distante a un type particulier
- CON_SERV : demande de connexion au server
- ACK_CON : le serveur accepte votre connexion
- DEM_AMI : demande d'ami pour echange pair à pair
- DEM_CON_AMI : demande d'établissement d'une connexion avec autre client distant
- CHECK_CON : vérification que le client distant est connecté
- CMD_CON : initialisation du mode commande
- CMD_HOME : le path home du client distant est envoyé
- CMD : envoi d'une commande
- LS_RET : retour d'une commande LS
- CD_RET : retour d'une commande CD
- MAJ_PATH : mise à jour du path courant
- CMD_END : fin du mode commande
- DEM_FIC : demande d'envoi d'un fichier
- ENV_FIC : envoi du fichier
- ACK : acquittement de reception du fichier
- FIN_CON_AMI : fin de connection avec le client distant
- FIN_CON_SERV : fin de connexion avec le serveur
- ERROR : Trame d'erreur sur le client distant

3.2.2 Transfert de données

Lors de l'envoi d'un fichier, si sa taille dépasse la taille maximum des datas d'une trame (1000 Octets ici), le fichier doit être découpé en plusieurs partie. Ceci se fait tout simplement en parcourant le fichier par bloc de 1000 octets et en envoyant directement une trame contenant ces données. Il faut toute fois noter qu'il faut tout d'abord connaître la taille du fichier pour déterminer le nombre de trames qui vont être envoyées. On determine donc le nombre de trame a envoyer ainsi que la taille des données qui seront contenues dans la dernière trame (qui peut ne pas être complete)

La reconstruction du fichier par ecriture direct des données contenues

dans les trames reçues dans le fichier destination. Lors de la reception de la première trame du fichier, le nombre de trame attendu est extrait et permet de savoir combien de trame sont encore à recevoir. Toutefois, la reception des trames du fichier s'arretera si l'ordre n'est pas respecté. Ceci n'est pas sensé arriver car TCP gère déjà l'ordre d'arrivé des trames avec son propre système.

Chapitre 4

Résultats

4.1 Vitesses de transfert

4.2 Format de trames

Nous avons opté pour une taille de trame de 1132 octets (entête comprise) pour ne pas dépasser la taille maximale des données d'une segment de TCP qui est 1500 octets. Ceci permet de ne pas avoir a redécouper les données envoyées, ce qui ralentirait la vitesse de transfert.

Chapitre 5

Conclusion

5.1 Apports

5.2 Difficultés rencontrées

Nous avons principalement rencontré des difficultés lors du découpage des fichiers ainsi qu'au niveau de la procedure de reception de trame. La première difficulté était de savoir comment découper un fichier et de le reconstruire après. La deuxième difficulté était que, lors de la reception de toutes les trames contenant le fichier, le système s'arrêtait. Ce problème était du à l'utilisation de la fonction read dans un socket pour récupérer son contenu. En effet, cette fonction n'attend pas que le buffer de reception soit rempli, ce qui entrainait une perte d'information. La solution apporté a été d'utiliser la fonction recv qui permet, grâce à un flag, d'attendre que le buffer soit rempli avant de recevoir une autre trame.