1 Introduction

Pour ce projet, il nous a été demandé de conçevoir un programme en language C permettant d'envoyer un fichier vers hôte distant connecté en réseau local. La difficulté du programme résidait dans le fait qu'il devait permettre de déctecter les erreurs lors du transfert ainsi que les informations non-reçues et renvoyer les parties défectueuses afin que le fichier reçu soit exactement le même que celui envoyé à la base. Tout ceci, alors qu'il nous est imposé de travailler sur un protocole de transmission qui ne le fait pas nativement. Il a donc fallu ces obstacles lors de la transmission manuellement.

2 Choix d'implémentation

Le protocle utilisé a dû être l'UDP, l'algorithme de transmission était aussi fixé au selective repeat et la structure d'un packet étant donnée, les grands points de l'implémentation étaient fixés dès le début.

Par contre, le programme étant clairement divisé en 2 parties (envoyeur et reçeveur), nous avons décidé de nous le diviser en 2 parties distinctes tel quel. Ce qui fait que nous avons deux implémentations assez différentes.

2.1 Sender

2.2 Receiver

Le receiver étant un peu plus simple à réaliser à première vue, nous avons juste préféré que le receiver utilise la structure packet qui est stockée dans un fichier externe ainsi, lorsque la structure d'un packet doit changer au cours du développement du programme, comme il a été le cas, nous n'avons jamais eu de problèmes à cause de ça. C'est le seul packet externe utilisé par le receiver car même pour préparer le packet d'un accusé, il n'est pas intéressant de créer un nouveau fichier externe.

Nous aurions pû, après réception du premier packet, créer une connection avec l'hôte local. Cela aurait permis de passer en TCP mais nous avons décidé dès le début de travailler constamment en UDP, ce qui fait que nous n'avons aucune connection avec l'autre hôte. Celà permet entre outre, de n'avoir qu'une boucle dans notre programme et que le programme soit plus malléable, nous ne dépendons effectivement jamais du protocole de plus haut niveau.

Le troisième choix d'implémentation a été plus délicat à prendre. En effet, il est obligatoire d'avoir un buffer tampon entre la réception d'un packet et l'écriture dans le fichier car autrement, les packets auraient été écrits dans la mauvais ordre dans le fichier. Nous avons décidé, par facilité, de créer un tableau de 256 lignes sur 512 colonnes comme buffer. Ceci est assez imposant comme taille et il aurait été possible de s'arranger en faisant un buffer tampon plus petit mais nous n'y avons pas porté trop d'importance car celà permettait d'y voir plus clair pour le déboguage du programme et celà reste ridiculement petit

pour les ordinateurs de nos jours mais c'est celà reste sans contester une des dernières parties possibles à optimiser dans notre programme.

3 Conclusion

Pour conclure, malgré que le projet ait a première vue semblé facile, nous nous sommes vite rendu compte qu'il n'était quand même pas à sous-estimé. Nous avons eu beaucoup de bugs, autant d'un côté que de l'autre du programme dont certains parfois très perturbant et demandant de creuser pas mal au sein de la mémoire de l'ordinateur. Mais au final, toutes les fonctionnalités requises ont pû être correctement implémentées et fonctionnelles car nous y avons pas mal de temps et chacun de nous deux connaissant par coeur son côté du programme.