Réseau informatique et musique

Mageshwaran SEKAR

IMA4

Encadrant : Xavier REDON

Table des matières

Remerciements

Je tiens tout d'abord à remercier, toute l'équipe pédagogique de Polytech'Lille et les responsables de la formation Informatique, Microélectronique et Automatique de m'avoir enseigné les bases pour réaliser ce projet dans de bonnes conditions. Je tiens à remercier aussi M. Xavier REDON pour encadrer ce projet tout au long de sa réalisation.

Introduction

Dans le cadre de la formation en 4ème année à Polytech Lille, nous devons réaliser un projet d'une durée minimale de 40 heures. Celui qui m'a été attribué est un projet de *réseau informatique et musique*. Ce projet doit permettre un administrateur de système de connaître l'état de son réseau informatique à tout moment. De ce fait, il faut mettre en œuvre un programme qui traduira la présence de différents types de paquets par des sons (ou mélopée). Ce projet se base principalement en langage C.

1 Présentation du projet

1.1 Description

Dans un réseau informatique, il est important, pour un administrateur, de connaître l'état de son réseau à un moment donné. En revanche, ce travail est très prenant et pénible à faire car il devrait étudier les paquets un par un et filtrer que les informations utiles. C'est pour ça qu'il est intéressant de filtrer les paquets utiles et avertir l'administrateur (par l'intermédiaire du son) de l'arrivée de tels types de paquets dans le réseau.

Dans le cadre de ce projet, le matériel qui sera utilisé est un PC sous Linux. Au niveau de logiciel, l'outil *LSF*, les APIs de *libsndfile* et *libasound* (bibliothèque d'ALSA) seront utilisés pour faire la programmation en C.

1.2 Objectif

Le projet est pour le but de récupérer tous les paquets qui traversent le réseau et de l'analyser ses paquets en produisant un son correspondant à ces paquets. Celui-ci facilitera un administrateur car il ne devait plus étudier tous les paquets en détails.

1.3 Cahier des charges

- Récupérer les paquets avec LSF
- Analyser des paquets sur différentes couches OSI (liaison de données, réseau, transport, application)
- Ecrire un programme (en utilisant la bibliothèque de son) pour pouvoir produire de son par rapport au type de paquets
- Appariement des sons correspondants aux paquets étudiés en fusionnant LSF avec le programme créé précédemment

2 Réalisation du projet

2.1 Analyse

Dans un réseau local, il y aura plusieurs types de paquets qui le traversent. Il est important d'analyser ce type de paquets afin de savoir l'état actuel de ce réseau. Le but principal de ce projet est de capturer les paquets qui traversent le réseau et les étudier. Pour cela, l'outil LSF (Linux Socket Filter) est utilisé. Tout d'abord, il fallait étudier les types de paquets qui peuvent circuler dans le réseau local et les catégoriser d'après le modèle OSI (Open Systems Interconnection).

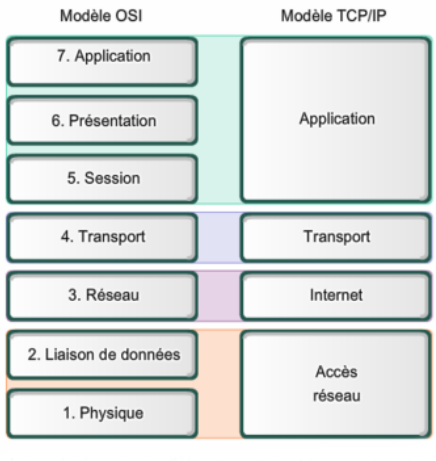

Figure 1: Modèle OSI

Les paquets qui seront analysés dans ce projet sont :

Couche	**Protocole**
Liaison de données	Broadcast
Réseau	ICMP - ping IP - flux local IP - flux distant
Application	Flux de grande volume : NFS, SSH/SFTP Flux de faible volume : DNS, HTTP

Il est envisagé de produire un son court pour tous les flux de faible volume tels que broadcast, ICMP, DNS et HTTP. Et, un son un peu long est réservé pour les flux de grands volumes tels que NFS et SSH/SFTP.

2.2 Etude de paquets avec LSF

2.2.1 Outil LSF

LSF, qui est basé sur *BSF* (Berkeley Socket Filter) est un outil qui permet de filtrer les paquets qui arrivent sur la carte réseau d'un ordinateur. Puisque cet outil est attaché au noyau de système, il nous permet d'écouter tous les paquets en mode brute. C'est-à-dire on peut même récupérer les paquets qui ne nous sont pas destiné.

2.2.2 Analyse de paquets

Pour analyser les paquets qui arrivent sur le réseau, il suffit de regarder quelques premiers octets de ces paquets afin d'identifier son type. Par exemple, pour déterminer un paquet broadcast, j'ai étudié les 6 premiers octets et les comparer à ***FF:FF:FF:FF:FF:FF***.

2.3 Partie Audio

Les deux bibliothèques qui seront utilisées pour la suite sont *libasound* (ALSA) et libsndfile. La bibliothèque *libasound* est utilisé pour ouvrir une poignée vers la périphérique de son et y écrire des données audio. Avant de pouvoir écrire, il nous faut des données audio. Pour cela, il faut utiliser la bibliothèque *libsndfile* afin de lire des fichiers (tel que *.wav*) pour récupérer les données audio.

2.3.1 Bibliothèque libsndfile

libsndfile est une bibliothèque qui permet de lire et écrire un fichier contenant des échantillons audio. Il permet de lire (et écrire) plusieurs type de fichier audio tels que *wav*, *FLAC*, etc. Il permet aussi de récupérer les informations d'un fichier audio comme le nombre de canaux, le format de fichier, le taux d'échantillons etc. Pour exploiter les fichiers de données, il fallait d'abord l'ouvrir avec la fonction sf_open comme ci-dessous :

SNDFILE sf_open(const char *path, int mode, SF_INFO *sfinfo)*

Le troisième paramètre de cette fonction récupère les caractéristiques d'un fichier audio. Ensuite, il a fallu lire les données audio. Pour cela, j'ai testé plusieurs fonctions qui lisaient les données en tant qu'un entier, un flottant et une double. Après plusieurs test, je me suis rendu compte que les fonctions qui convertit les données en flottant ou double produisent un son assez bruité (c'est-à-dire qu'il y a de perte d'informations).

*sf_count_t sf_read_short(SNDFILE *file, short *ptr, sf_count_t items);*
*sf_count_t sf_read_int(SNDFILE *file, int *ptr, sf_count_t items);*
*sf_count_t sf_read_float(SNDFILE *file, float *ptr, sf_count_t items);*
*sf_count_t sf_read_double(SNDFILE *file, double *ptr, sf_count_t items);*

De ce fait, j'ai choisi la deuxième fonction qui lit les données en tant que des entiers. A la fin de la lecture, il suffit de libérer les ressources de pointeur de SNDFILE avec :

*int sf_close (SNDFILE *sndfile);*

2.3.2 Bibliothèque libasound

libasound est une bibliothèque appartenant à ALSA (Advanced Linux Sound Architecture), qui permet d'envoyer les données audio à la périphérique concernant pour produire du son sur un haut-parleur (ou autre support).

Puisque la lecture de fichier audio a déjà été faite, il est suffisant d'ouvrir une poignée vers le périphérique de son, configurer le paramétrage de la poignée et d'envoyer les données vers le périphérique du son.

Par contre, le fonctionnement de ce programme était très basique car il était capable de configurer une seule poignée vers la périphérique. Afin de pouvoir gérer plusieurs poignées et lui envoyer des données en même temps, il a fallu utiliser les *threads* (processus léger).

En outre, le réglage de son a été aussi ajouté dans les threads. Cette fonction va modifier le niveau de son à chaque fois qu'un nouveau thread est lancé.

2.4 Les threads

Un thread est une partie de code qui est capable de s'exécuter en parallèle à d'autres traitements. Il sert à faire des choses en tâche de fond et aussi exécuter plusieurs instances d'un même code pour accélérer le traitement.

Dans ce projet, j'ai implémenté les threads pour pouvoir ouvrir une poignée vers la périphérique de son dès qu'un paquet arrive sur la carte réseau. De ce fait, multiple de poignées peuvent être ouverte s'il y a plusieurs paquet qui sont en train d'être traités. De plus, ces threads seront lancés en mode détachée afin de ne pas attendre la fin de leurs traitements.

2.5 Interface graphique avec GTK+

J'ai décidé d'ajouter une interface graphique à ce projet afin qu'un utilisateur puisse choisir les types de paquets à analyser. Pour réaliser celui-là, j'ai utilisé GTK+-3.0.

2.5.1 Outil Glade

Glade est un logiciel qui permet de concevoir des interfaces graphiques pour GTK+. Cet outil contient les éléments tels que fenêtres, boutons, etc. pour construire une interface graphique complète. Il génère un fichier de type XML.

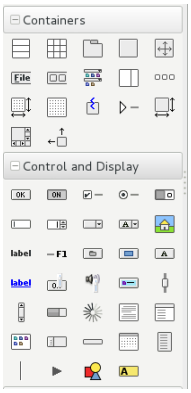

Figure 2: Différents éléments dans Glade

2.5.2 Gestion de l'interface graphique en C

En C, il a fallu d'abord charger le fichier XML créé par Glade et construire l'interface graphique. Puis, j'ai ajouté des évènements pour chaque bouton. Ces évènements permettent de choisir d'analyser les paquets concernés. J'ai aussi ajouté une statistique de nombre de paquets analysés.

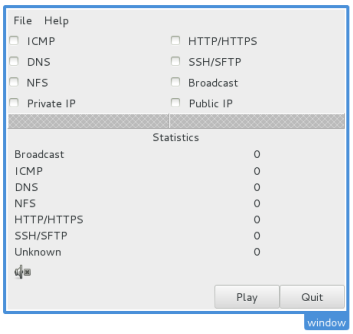

Figure 3: Fenêtre d'affichage

2.5.3 Gestion de threads dans GTK+

L'ajout des fonctions LSF et les fonctions de son (de libsndfile et libasound) nécessite un thread de GTK (qui s'appelle GThread). En effet, Gthread est un sous-ensemble de thread POSIX (pthread).

Il a fallu lancer l'appel à la fonction LSF dans GThread. La fonction de LSF quant à elle, va appeler la fonction analyse de paquet des fonctions audio dans un thread POSIX (dès la réception d'un paquet sur la carte réseau) comme montre la figure ci-dessous.

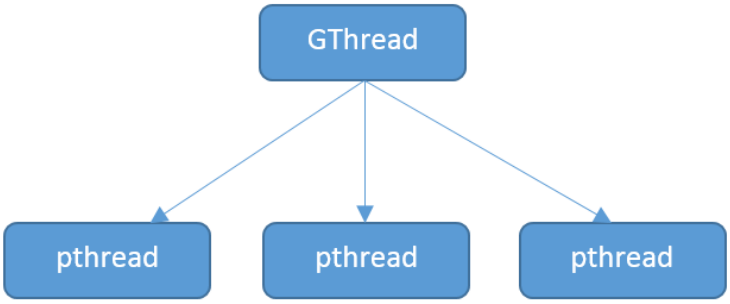

Figure 4: Lancement de threads

2.6 Résumé des étapes

2.6.1 Sans GTK+

Les étapes réalisés pour appariement des sons au paquet correspondant (sans interface graphique) :

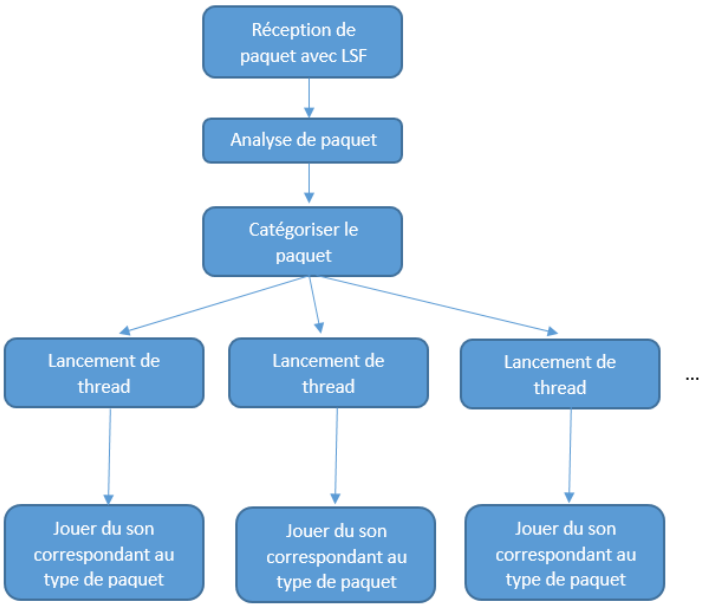

Figure 5: Etapes générales sans GTK+

2.6.2 Avec GTK+

Les étapes réalisés pour appariement des sons au paquet correspondant (avec interface graphique) :

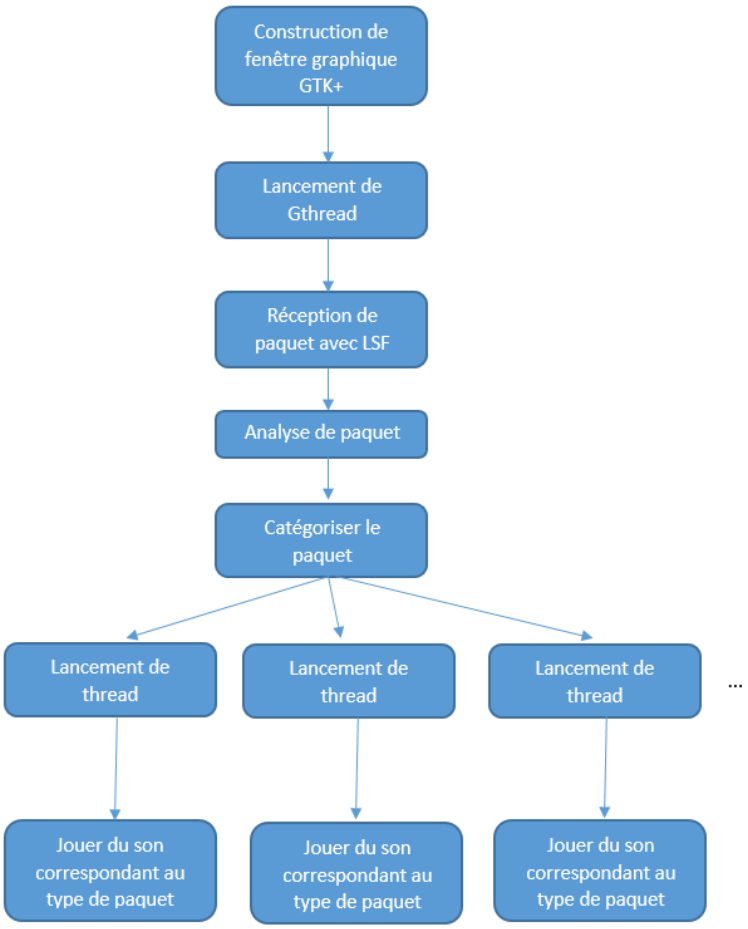

Figure 6: Etapes générales avec GTK+

3 Bilan

3.1 Difficultés rencontrés

Même en ayant beaucoup travaillé en langage C tout au long de formation en IMA, le projet n'était pas tout à fait facile à s'y mettre due à de nouveaux bibliothèques à exploiter surtout la bibliothèque *libasound*. J'ai appris des fonctions de cette bibliothèque en regardant les exemples sur le site officiel et en commençant à programmer. Une des problématiques des fonctions qui se trouvent sur le site officiel est le manque d'explication. Du coup, il était assez difficile pour exploiter ces fonctions pour ce projet.

3.2 Amélioration

Même si le projet est en état fonctionnel, il faut améliorer certains fonctionnements.

3.2.1 Analyse de protocole

Le programme, qui a été réalisé, prend en compte que certains protocoles tels que SSH, NFS, etc. (*cf.* tableau à la page 5). Ce sera mieux d'analyser beaucoup plus de paquets de différents types.

3.2.2 Réglage de son

Pour ce moment, la fonction de réglage de son ne fonctionne pas. Cette fonction diminue (ou augmente) le niveau de son en globale (celui de périphérique *Master*). Il faudra qu'il soit capable de modifier le niveau sonore si et seulement si un nouveau paquet arrive sur la carte réseau.

Conclusion

Pendant le déroulement de ce projet, j'ai pu mettre en pratique les connaissances acquises durant la formation en IMA surtout en système et réseaux. J'ai aussi acquis de nouvelles connaissances surtout en travaillant avec les bibliothèques de son. De plus, j'ai pu exploiter l'utilisation d'une interface graphique GTK+ lors de ce projet. Même s'il prenait un peu plus de temps à apprendre, c'était une bonne expérience. Et, à la fin du projet, l'objectif fixé a été atteint même s'il existe encore des améliorations à faire.