Rapport
Travaux Pratiques Réseaux (Ethernet)

Quentin Tonneau - Adrien Lardenois

Table des matières

1 Introduction

Après avoir mis en réseau un parc de pc (cf rapport tp n° 1), nous nous intéressons maintenant à la communication entre ces derniers. En s'appuyant sur le cours de réseaux Ethernet et nos connaissances en langage C dans l'environnement Linux, nous allons concevoir une suite d'applications destinées à interagir avec les PC voisins. Une bibliothèque de création et d'envoi de trame, ainsi qu'un sniffeur de réseau (affiche l'ensemble des trames circulant au voisinage de notre matériel) nous sont fournis afin de franchir les couches non étudiées en classe (4,5,6 et 7).

1.1 Écouter le réseau

Avant de communiquer sur un réseau, il nous faut un certain nombre d'informations sur le matériel avec lequel on souhaite "entrer en discussion", c'est à dire :

- Le nombre de personnes présentes sur le réseau
- Leurs adresses
- Le protocole de communication
- La nature des messages (émetteur, destinataire, type de message)

Pour cela, nous concevons un programme qui filtre les trames circulant sur le réseau, en ne conservant que les trames de type 9000 dont nous sommes le destinataire, puis affiche le message en question,tout en dressant une liste de toutes les machines (adresses MAC [1]) présentes sur le réseau. On pourrait associer ce programme à un module de conversation type IRC [2], qui n'affiche que les messages personnels ou à destination de l'ensemble des utilisateurs.

1.2 Envoyer un message

Après avoir pris connaissance du materiel voisin, nous écrivons une fonction d'envois de trame simple, ainsi qu'un algorithme d'envois en boucle d'un "bonjour" sur le réseau (broadcast). Le type choisi pour ces échange est toujours 9000, afin de pouvoir être facilement reçu par les autres groupes du projet. Nous construisons la trame nécéssaire, et l'envoyons à l'aide des bibliothèques fournies, et du programme write_eth_frame.

1.3 Accusé la réception d'un message

L'unification des deux parties précédantes nous permet de programmer un répondeur automatique, qui en reçevant une trame de type 9000 sous la forme "Bonjour de XX" répond immédiatement à son emmeteur "Bonjour de XX : bien reçu par Quentin et Adrien", accusant ainsi la réception du message. Nous ré-implémentons l'analyse de la première partie, ainsi que nos connaissances en matière de traitement de la langue pour détecter les messages qui nous destinés, et nous utilisons la fonction d'envois de trame crée en deuxième partie pour répondre aux messages.

2 Structure du projet

2.1 Structure des trames

Avant de nous lancer dans l'écriture des algorithmes de ce projet, nous avons dù analyser les bibliothèques fournies, et définir la structure de notre projet. La structure d'une trame est imposée, sous la forme :

```
struct eth_frame {
char adr_dest[6];
char adr_send[6];
char type[2];
char data[1500 - 6 - 6 - 2];
```

1. Media access control
2. Internet Relay Tchat

};

On remarque donc que notre trame sera définie par deux adresses (destinataire et emetteur) MAC au format hexadécimal (6 octets → 12 caracteres), d'un type (même encodage), et de données, n'excedant pas $1500 - 6 - 6 - 2 = 1486$ caractères. Une structure étant allouée de manière contigüe en mémoire, l'accès à l'adresse de l'emetteur pourra se faire sous la forme trame→adr_send ou bien (char*)trame[6]
.

2.2 Arborescence du projet

Afin de faciliter la compréhension du projet, l'archive fournie est organisée de la sorte :
- Un fichier de compilation (script sh) se trouve à la racine
- Les fichiers sources (.c et .h) utiles au projet sont dans un répertoire **src**
- Une fois compilés, les 5 programmes se trouves dans le repertoire **bin**

Liste et description des fichiers :
- **compilation.sh** : Fichier de Compilation (voir partie suivante)
- src
  + **envoyer_trame .c/.h** : Fichier source de l'envois de trame (bonjour)
  + **eth_lib .c/.h** : Bibliothèque d'utilisation du réseau fournie
  + **fonctions .c/.h** : Bibliothèque de fonctions communes au trois principaux programmes
  + **inet_str.h** : Structures des trames, nécéssaire à la compilation de eth_lib
  + **recevoir_trame .c/.h** : Programme d'affichage des trames circulant sur le réseau
  + **reponse_auto .c/.h** : Programme de réponse automatique aux messages recu (accusé de réception)
  + **sniff.c** : Programme de sniffage du réseau (fourni)
  + **write_eth_frame.c** : Programme permettant l'envois de la trame donnée en premier paramètre (fourni)
- bin
  + **envois** : programme d'envois des bonjours
  + **recevoir** : programme d'affichage des trames
  + **reponse** : programme d'accusé de reception
  + **sniff** : programme utilisé pour l'execution de recevoir et reponse
  + **write_eth_frame** : programme appelé par envois et reponse

2.3 Compilation et mode d'emplois

2.3.1 Compilation

Pour compiler l'ensemble du projet, il suffit d'executer le fichier **compilation.sh**

3 Réception des trames

La reception de trames se fait par redirection de la sortie du sniffer vers notre programme défini par **recevoir_trame.c**, au moyen d'un pipeline. Les informations reçues sont traitées une à une par la fonction *lire_trame()*. On vérifie alors si l'adresse de destination et l'adresse source nous sont déjà connus puis, si l'un et/ou l'autre est "nouveau", on l'ajoute à notre liste des matériels connectés.

Nous avons choisi de représenter la liste des adresses rencontrées par une liste chainée. Cela a pour avantage de permettre l'ajout d'un maillon sans connaitre la taille actuelle de la liste, au contraire d'un tableau qu'il aurai fallu redimensionner au fur et à mesure.

Dans un second temps, on vérifie si la trame appartient au type défini spécialement pour le TP (9000). Si c'est le cas, on affiche à l'écran les informations suivantes :
- Numéro de la trame (au moyen d'un compteur)
- Adresse source
- Contenu de la trame

Enfin, chaque fois que Y trames ont été affichées[3], on affiche un récapitulatif des adresses qui circulent sur le réseau. Suivant le choix de Y (50 dans nos exemples) et en fonction du traffic sur le reseau, on beneficie alors d'un affichage clair des trames qui circulent avec un bilan regulier des utilisateurs presents.

3.1 la fonction lire_trame

La fonction *lire_trame* est définie dans **fonctions.c**. Elle retire les 3 premiers caractères de la sortie du sniffer qui ne nous intéressent pas, puis récupère la longueur de la trame. Apres quoi elle capte toute la trame grâce à la fonction *get_buf* et la retourne en sortie.

3.2 la fonction recherche

La fonction *recherche* parcoure la liste chainée à partir du premier maillon et s'arrête si en passant au suivant elle arrive à NULL. Il est possible de faire le test à la fin car l'initialisation de la liste garanti l'existence d'au moins un maillon different de NULL.

Pour chaque maillon, on compare le champ adresse à celle que l'on cherche : s'il y a correspondance on sort directement en renvoyant 1, sinon on passe au suivant. Si la recherche a eu lieu jusqu'au bout sans succes, on retourne 0.

3.3 la fonction ajout

La fonction d'ajout ajoute un élément en tete de liste. On crée simplement un nouvel élément que l'on défini comme l'adresse à ajouter et dont le suivant est l'ancienne tete de liste.

3.4 la fonction affiche

La fonction *affiche* parcoure la liste de la même maniere que *recherche* . Pour chaque étape de la liste, elle affiche l'adresse et separe les adresses par : : :

4 Envois des trames (type 9000)

4.1 Fonction d'envois

Maintenant que nous recevons les trames de type 9000, nous devons à notre tour, pour :
- Nous identifier sur le réseau
- Échanger des informations
- Accuser la réception d'informations externes

Pour celà, une fonction nommée "envois_trame" dans le fichier fonctions.c construit la trame en fonctions des paramètres :

```
//Envois une trame sur le reseau, de type 9000
void envois_trame(char *source, char *dest, char *message)
```

où source et dest sont les adresses MAC des machines concernées au format XX :XX :XX :XX :XX :XX. Pour celà, nous avons créé une procédure convaddr qui converti les adresses en ajoutant le caractère « : ». Ensuite, nous créeons la trame à l'aide de la fonction

```
int make_ping_request(char* eth_addr_dest, char* eth_addr_send,
 char* type_frame, char* data, struct eth_frame* frame);
```

dont la valeur de retour nous indique la taille de la trame générée. Nous convertissons ensuite cette trame au format hexadécimal par la fonction char_to_charhexa et nous envoyons la trame en passant cette dernière en paramètre du programme write_eth_frame fourni dans le projet.

3. Y étant défini au début du programme

4.2 Programme de présentation

Afin de nous présenté une fois connecté sur le réseau, nous avons créé le programme *recevoir_trame.c* dont les paramètres suivants sont à définir avant la compilation :

```
#define NOMBRE_ENVOIS 10   //Nombre de message a envoyer
#define TEMPS_ATTENTE 1    //Temps d'attente entre deux envois (secondes)
#define ADRESSE "e0:cb:4e:2f:fc:98" //Adresse par defaut
#define MESSAGE "Bonjour de Quentin et Adrien" //Message
```

Ce programme envois un certain nombre de fois un message d'identification sur le réseau, à l'adresse de broadcast *ff :ff :ff :ff :ff :ff*.

5 Synthèse et automatisation

6 Bilan