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.

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) présentes sur le réseau. On pourrait associer ce programme à un module de conversation type IRC [1], qui n'affiche que les messages personnels ou à destination de l'ensemble des utilisateurs.

1.2 Envoyer un message

Après avoir pris connaissance des appareils connectés à notre pc, nous pouvons maintenant écrire un programme d'envois de message.

1.3 Accusé la réception d'un message

blabla

2 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 [2], 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.

1. Internet Relay Tchat
2. Y étant défini au début du programme

2.1 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.

2.2 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.

2.3 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.

2.4 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 : : :

3 Envois des trames (type 9000)

4 Synthèse et automatisation

5 Bilan