I2L - Linux en réseau - TD1

Éric Leblond

30 novembre 2009

Introduction

Le but de ce TD est d'expérimenter quelques unes des fonctionnalités réseaux de GNU/Linux.

1 Mise en place d'une interface virtuelle ATM

1.1 Paramétrage du système

1.1.1 Installation des outils

Nous installons le paquet `atm-tools` qui contient la majeure partie des utilitaires nécessaires au TD.

1.1.2 Paramétrage du noyau

La mise en place d'une interface virtuelle ATMTCP va permettre d'établir une liaison ATM entre les deux machines. La figure 1 illustre le mécanisme.

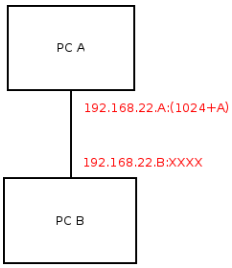

FIG. 1 – Mise en place du lien ATMTCP

Sur le poste A :

```
# modprobe atmtcp
# atmtcp virtual listen \$((1024+A))
```

Question 1 *Vérifier ensuite grâce à* `netstat` *que* `atmtcp` *est à l'écoute. La commande à utiliser devra indiquer le processus à l'origine de la connexion.*

Sur le poste B :

```
# modprobe atmtcp
# atmtcp virtual connect 192.168.22.A
```

On ajustera la commande pour permettre l'établissement correct de la connectivité.

1.2 Description

Question 2 *Réaliser un schéma représentant le trajet dans le noyau et sur le réseau d'un paquet émis sur l'interface.*

1.3 Test de l'interface

1.3.1 Utilisation des outils AAL

Nous utiliserons `aread` et `awrite` pour valider la connectivité ATM des interfaces virtuelles.

1.3.2 Écoute sur un VC

La syntaxe de `aread` est la suivante :

```
aread [-c] [itf.]vpi.vci
```

Écoutons sur le VC de notre choix sur le poste A :

```
aread -c $ITF.0.42
```

Question 3 *Spécifier le numéro de l'interface* [1].

1.3.3 Écriture sur le VC

La syntaxe de `awrite` est la suivante :

```
awrite [itf.]vpi.vci data
```

La commande `awrite $ITF.0.42 toto` lancée sur le poste B permet d'écrire sur le VC et il ne reste plus qu'à vérifier que les données ont été reçues sur le poste A.

Question 4 *Spécifier le numéro de l'interface.*

2 Découverte d'informations sur un réseau ethernet

2.1 Écoute du réseau et collecte d'information

Question 5 *Utiliser tcpdump pour étudier le trafic global sur le réseau et trouver le réseau caché.*

Question 6 *Sauver dans un fichier un enregistrement de trafic par tcpdump.*

Question 7 *Travailler sur le fichier pour limiter l'affichage au requête arp "who-has".*

Question 8 *Ecrire un script permettant de récupérer l'IP des machines actives d'un réseau donné.*

Question 9 *Déterminer le fabriquant de la carte réseau de la machine du réseau caché. On pourra utiliser "arpwatch" et le dump réseau précédemment réalisé.*

Question 10 *Pourquoi ping IP caché ne répond pas ?*

2.2 Intrusion

Question 11 *Prendre une adresse dans le réseau caché en créant un alias sur l'interface eth0.*

Question 12 *Scanner la machine avec nmap et donner la liste des services disponible en TCP et UDP. On veillera à scanner l'ensemble des ports TCP.*

Question 13 *Déterminer le système d'exploitation probable grâce à "nmap".*

2.3 Utilisation du routeur

Question 14 *Procéder à une modification routage pour utiliser la machine comme routeur par défaut.*

Question 15 *Tester le routage des paquets à travers la machine.*

Question 16 *Déterminer le prestataire de service internet utilisé par la machine.*

[1] On pourra se référer à `cat /proc/net/atm/devices` pour trouver le numéro de l'interface

3 Développement d'un logiciel de scan de port

On pourra utiliser l'exemple de code suivant comme base de travail pour le développement du client :

```
#include <stdio.h>
#include <stdlib.h>
#include <unistd.h>
#include <sys/socket.h>
#include <sys/types.h>
#include <sys/time.h>
#include <netinet/in.h>
#include <netinet/tcp.h>
#include <string.h>
#include <arpa/inet.h>
#include <errno.h>

int main (int argc, char *argv[])
{
  int sock_fd;
  int ret;
  int ch;
  int n=1;
  char connectaddrstr[128]="0.0.0.0";
  struct sockaddr_in bind_addr;
  struct sockaddr_in connect_addr;
  int port=80;

  /* assign connect address */
  memset(&connect_addr,0,sizeof bind_addr);
  connect_addr.sin_family= AF_INET;
  connect_addr.sin_port=htons(port);
  connect_addr.sin_addr.s_addr=inet_addr(connectaddrstr);

  sock_fd=socket (AF_INET,SOCK_STREAM,0);
  if (sock_fd == 0){
          fprintf(stderr,"Can not open socket\n");
          return EXIT_FAILURE;
  }

  /* connect */
  if ( connect(sock_fd,(struct sockaddr*)&connect_addr,sizeof(connect_addr)) == -1){
          if (errno != ECONNREFUSED){
                  fprintf(stderr,"connect failed\n");
                  return EXIT_FAILURE;
          }
  } else {
          /* close socket */
          shutdown(sock_fd,SHUT_RDWR);
  }
  close(sock_fd);
  return EXIT_SUCCESS;
}
```

3.1 Obtention d'un programme fonctionnel

Question 17 *Paramétrer la sortie pour afficher un message clair en cas d'échec de la connexion. On différenciera le cas du refus de connexion et le cas du délai dépassé.*

Question 18 *Rendre le programme "utilisable" grâce à getopt. En ajoutant la possibilité de spécifier l'adresse destination.*

Question 19 *Ajouter une boucle permettant d'itérer sur tous les ports.*

Question 20 *Tester le code obtenu et comparer les performances à celle de "nmap".*

Question 21 *Pourquoi les performances sont-elles moins bonnes ?*

3.2 Paramétrage de l'IP source

Question 22 *Utiliser la fonction bind pour spécifier l'adresse source des paquets*[2].

[2] L'appel doit se faire avant `connect`