TD 5 – Le protocole TCP et autres protocoles réseaux

Ce TD a pour objectif de vous faire découvrir ce qui se passe au niveau du réseau lorsque vous réalisez des échanges entre différentes applications. Sera étudié le fonctionnement du protocole TCP, mais aussi des autres protocoles présents sur un réseau, ainsi que les interactions qui existent entre-eux.

I - Exercice d'introduction à TCP

Remise en ordre

La commande ipconfig exécutée sur une machine donne le résultat suivant :

```
Adresse physique . . . . . .  00-60-08-61-04-7b
Adresse IP. . . . . . . . . . :  110.10.111.2
Masque de sous-réseau . . .  255.255.0.0
Passerelle par défaut . . . . .  110.10.1.0
Serveurs DNS . . . . . .  110.10.1.1
```

1. Quelle est le contenu « minimal » de la table de routage de la machine 110.10.111.2

Sur cette même machine vous installez et activez un analyseur de trames afin d'observer les échanges réalisées. Vous lancez ensuite un navigateur et tapez dans la zone adresse : http://www.monsite.fr
Suite à une erreur de manipulation vous récupérez l'ensemble des trames échangées, mais dans le désordre.

Protocole	Source	Destination	Options
TCP	110.10.111.2 : 1234	163.34.45.1 : 80	Flag=fin,ack Seq=615 Ack=145 Win=64240 Len=0
TCP	110.10.111.2 : 1234	163.34.45.1 : 80	Flag=syn Seq=0 Ack=0 Win=64240 Len=0
DNS	110.10.1.1 : 53	110.10.111.2 : 1234	
HTTP	163.34.45.1 : 80	110.10.111.2 : 1234	HTTP/1.1 304 Not Modified
ARP	000102aff5e2	00600861047b	
HTTP	110.10.111.2 : 1234	163.34.45.1 : 80	Get /http://…
TCP	163.34.45.1 : 80	110.10.111.2 : 1234	Flag=ack Seq=1 Ack=615 Win=6754 Len=0
ARP	00600861047b	ffffffffffff	
TCP	110.10.111.2 : 1234	163.34.45.1 : 80	Flag=ack Seq=615 Ack=145 Win=64240 Len=0
DNS	110.10.111.2 : 1234	110.10.1.1 : 53	
TCP	110.10.111.2 : 1234	163.34.45.1 : 80	Flag=ack Seq=1 Ack=1 Win=64240 Len=0
TCP	163.34.45.1 : 80	110.10.111.2 : 1234	Flag=ack, syn Seq=0 Ack=1 Win=5840 Len=0

2. Remettez dans l'ordre chronologique les trames ci-dessus, en justifiant vos choix
3. Expliquez le rôle des options qui figurent dans les lignes TCP
4. Quelle est la longueur des données transmises dans les lignes HTTP . Justifiez vos réponses.

II - Analyse de trames TCP et autres protocoles avec Wireshark (Version Linux)

1 - Présentation

Wireshark est un analyseur de trafic réseau, ou "sniffer" qui est installé sur les machines linux. Il utilise une interface graphique basée sur GTK+, il est basé sur la bibliothèque libpcap, qui fournit des outils pour capturer les paquets réseau.

2 - Utilisation de Wireshark

Pour lancer ce logiciel, il suffit d'utiliser un terminal et d'y taper la commande : **sudo wireshark.**

Pour plus d'informations sur l'utilisation même du produit, tapez **man** Wireshark à l'invite de commande d'un shell.

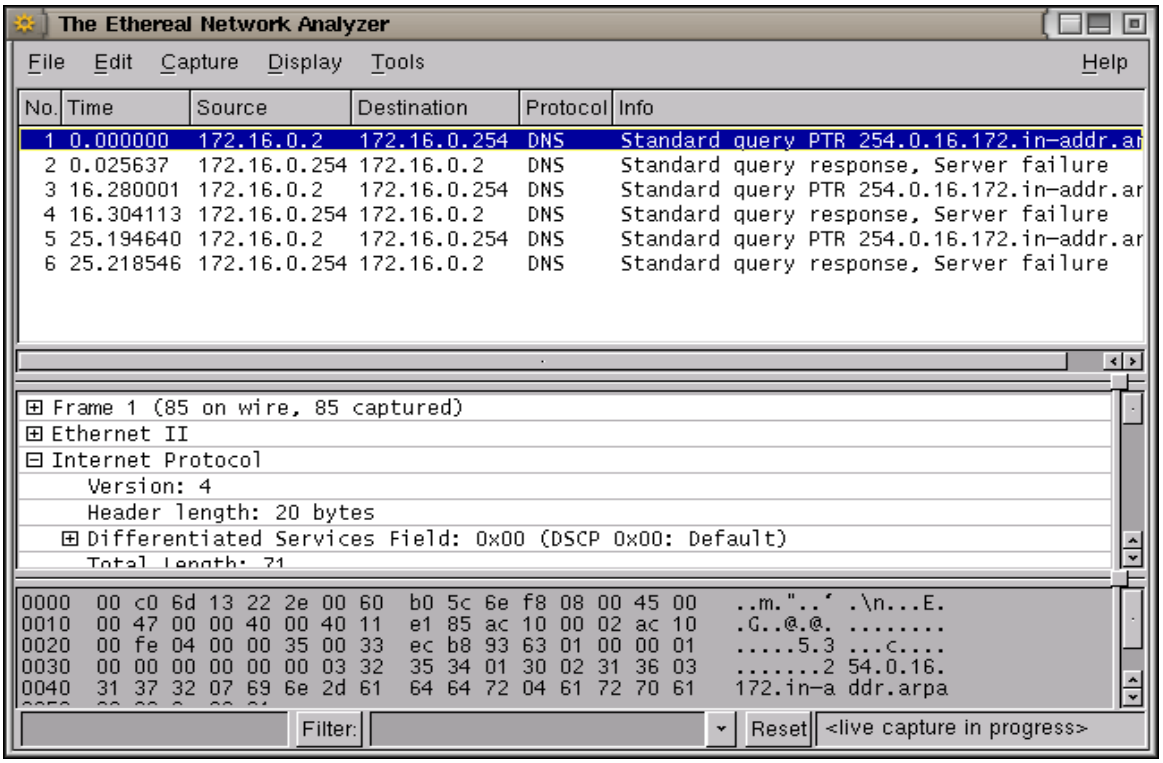
Le principe est simple, vous lancez une session de capture à l'aide du menu **Capture**. Cette session peut être interactive ou pas. En d'autres termes, les paquets capturés peuvent être affichés au fur et à mesure ou à la fin de la capture.

Pour lancer une session de capture, il faut accéder au menu **Capture** puis cliquer sur l'option **Interface...**.

Apparaît la boîte de dialogue qui permet de spécifier sur quelle interface vous souhaitez « écouter » les paquets.

Il est préférable d'opter pour l'option « any » qui écoute tous les interfaces en même temps.

Durant la capture, une boîte de dialogue récapitule les paquets qui sont conservés. En même temps les paquets apparaissent dans la fenêtre principale. L'appui du bouton Stop de l'interface permet d'arrêter la capture. Les paquets deviennent disponibles dans la fenêtre principale s'ils n'étaient pas déjà disponibles



3 - Analyse des échanges

1 – Lancer simplement la capture de paquets et attendez quelques instants. Couper la capture et observer tous les paquets capturés :

- Quels protocoles étaient actifs dans le réseau ?

2 – Lancer la capture de paquets, puis lancer un navigateur et allez à l'adresse : http://ffctlr.free.fr

- Combien de requêtes http ont-elles été faites ? Pourquoi ?

- Pour toutes ces requêtes, combien de connexions du protocole TCP ont-elles été créées ?
- Identifier les paquets relatifs à ces connexions.
- Identifier les différentes phases d'un échange TCP. Quelles sont les adresses IP et les ports utilisés ?

4 - Analyse du protocole TCP

Dans l'analyse des entêtes des messages TCP on peut observer une zone nommée « Flags ». A l'aide du filtre de Wireshark afficher une session TCP complète (connexion, échange, déconnexion). Pour cela TCP utilise à chaque session un nouveau port, il suffit donc de mettre un filtre sur un port local donné et demander l'affichage.

- Relevez dans chaque message les valeurs des indicateurs « flags » utilisés. Quel est le rôle de chaque indicateur ?
- Analysez le paramètre « len » (longueur des données), pourquoi certains messages ont une longueur = 0 et d'autres non ?
- Analysez le paramètre « sequence number », comment évoluent les valeurs au cours de cet échange ? A quoi sert cette donnée ?

5 - Manipulation amusante :

- Réinitialiser la capture des paquets et aller à l'adresse : http://ffctlr.free.fr/formulaire.html
- Remplir les champs du formulaire, mot de passe compris et le « soumettre »
- Arrêter la capture, repérez les paquets relatifs à la connexion et essayez d'identifier votre login (nom + mot de passe).

III – Anatomie d'une application client-serveur

Le programme s1.c présente un mini service sous linux.

```
#include <stdio.h>
#include <stdlib.h>
#include <errno.h>
#include <strings.h>
#include <netinet/in.h>
#include <sys/socket.h>
#include <sys/types.h>

#define PORT 12345
int sock, socket2, lg;
char mess[80];
struct sockaddr_in local;
struct sockaddr_in distant;

creer_socket()
{
 // Création des champs adresses
   bzero(&local, sizeof(local));              // mise a zero de la zone adresse
   local.sin_family = AF_INET;                        // famille d adresse internet
   local.sin_port = htons(PORT);              // numero de port
   local.sin_addr.s_addr = INADDR_ANY;        // types d'adresses prises en charge
   bzero(&(local.sin_zero),8);                // fin de remplissage
```

```
  lg = sizeof(struct sockaddr_in);
  // Création socket
  if((sock=socket(AF_INET, SOCK_STREAM,0)) == -1){perror("socket"); exit(1);}
  // Nommage
  if(bind(sock, (struct sockaddr *)&local, sizeof(struct sockaddr)) == -1)
    {perror("bind");exit(1);}
  // Mise en service
  if(listen(sock, 5) == -1){perror("listen");exit(1);}

}

main()
{
  creer_socket();

     while(1)
    {  // Attente connexion
            if((socket2=accept(sock, (struct sockaddr *)&distant, &lg)) == -1)
      {perror("accept");exit(1);}
            // lecture flux
            read(socket2,mess,80);
            printf ("mess : %s \n",mess);
            // fermeture
      close(socket2);
    }

}
```

Test en local:

Pour executer ce programme vous devez le compiler en tapant : gcc -o s1 s1.c

- Ouvrez un autre terminal et lancez : **telnet 127.0.0.1 12345** Tapez un texte, que se passe-t-il ?
- Lancer plusieurs fois la commande telnet et observez les échanges avec wireshark et commentez …