TP N°3 – Etude des protocoles TCP/UDP

Dans ce TP, vous allez étudier les protocoles TCP/IP de niveau 4, à savoir UDP et TCP. Vous utiliserez le programme **socklab** situé dans le répertoire **/2005.socklab3/socklab/** et dont le but est de proposer une interface interactive de manipulation des sockets. Ainsi vous pourrez créer des sockets et effectuer des échanges de données.

Pour l'ensemble des manipulations de ce TP, vous travaillerez par binôme (1 machine pour chaque étudiant).

Il sera demandé un compte rendu individuel qui sera noté et qui contiendra les commandes tapées, leurs résultats (traces) (lorsque c'est possible), des explications et des commentaires.

Rappel: Une Socket est un objet local au système, qui permet à un processus d'envoyer des données sur le réseau, en général par l'intermédiaire d'un protocole de couche 4 (TCP ou UDP), mais il est possible d'utiliser directement IP (cas par exemple de la commande ping). Au niveau du système une Socket est donc identifiée par un descripteur (comme les fichiers ouverts, les pipes ...) qui a une portée locale uniquement. Ce descripteur est utilisé par un processus pour préciser quelle Socket il veut utiliser pour envoyer/recevoir.

1) Le protocole TCP

1.1 Etablissement d'une connexion

Sur une station lancez le logiciel *wireshark* pour analyser les paquets générés.

1. Sur deux machines, lancez **socklab** en précisant que vous désirez uniquement travailler avec le protocole TCP, comme suit :

   #. /socklab tcp

2. Sur la première machine PC1, créez une Socket "passive" et notez le numéro de port de la socket ainsi créée:

   #socklab-tcp> passive **(**Equivalent de l'appel aux fonctions C: "Socket" et "listen").

3. Mettez-la en mode d'attente d'une demande de connexion à accepter:

   #socklab-tcp> accept **(**Equivalent de l'appel de la fonction C: "accept")

4. Sur la deuxième machine PC2, créez une Socket, et connectez-la sur la machine et le port de la Socket passive précédemment créée. Ceci se fait grâce à la commande:

   #socklab-tcp> connect @IP-machine numero-de-port (appel des fcts "socket" et" connect").

a) Quelle est la station que l'on peut appeler le client et celle que l'on peut appeler le serveur ? Expliquez pourquoi la première Socket créée est dite "passive", et la seconde "active".

b) Analysez les paquets générés lors de l'établissement d'une connexion de PC1 vers PC2. Décomposez les étapes de cette connexion: enchaînement dans le temps des demandes de services à TCP et des messages échangés.

c) Quelles sont les informations échangées durant ces étapes et à quoi servent elles ? Précisez les valeurs et le sens des champs **ACK**, **SYN**, **FIN**, **ACK-number** et **SEQ-number**. Quel est le rôle du flag SYN?

d)

- Sur PC1, créez une nouvelle Socket « passive ». Notez le statut de la connexion avec: **#status**.
- Sur PC2, créez une nouvelle Socket, et connectez-la sur PC1.
- Après exécution de la commande "accept" sur PC1, consultez à nouveau le statut de la connexion. Expliquez ce qu'il se passe au moment du retour de "accept".

e) Etablissez maintenant plusieurs connexions du même client vers la même socket serveur.

- vous devez pour cela relancer accept sur le serveur.
- Analysez le résultat de la commande "status" sur le serveur, que constatez vous ?

f) Déterminez comment procède TCP pour différencier les connexions arrivant sur un même port.

1.2 Libération d'une connexion : analyse de la fermeture d'une connexion

Après ouverture d'une connexion entre deux machines (par exemple entre PC1 et PC2), fermez la connexion de PC1 vers PC2, puis de PC2 vers PC1 (commande **socklab-tcp> close**).

a) Analysez avec Wireshark les paquets générés lors de la fermeture dans chaque sens et décomposez les étapes de cette fermeture. Quel est le rôle du flag FIN ?

1.3 Etude du séquencement et du contrôle d'erreur

Vous allez étudier maintenant un échange de données. Pour cela, après connexion, vous effectuez

- depuis PC2, « write » pour expédier un message :
  socklab-tcp> write <Id de socket> <message>
- depuis PC1, « read » pour le recevoir :
  socklab-tcp> read <Id de socket> <nb d'octets>

a) Analysez les paquets engendrés par le transport des données. Expliquez le rôle des champs **SEQUENCE NUMBER** et **ACK NUMBER** dans l'entête des paquets TCP. Donnez les valeurs que vous avez observées dans plusieurs trames consécutives.

b) Expliquez comment s'effectue la mise à jour des ces 2 champs. Faites attention à distinguer le cas où l'on reçoit un paquet et le cas ou l'on doit émettre un paquet.

2) Le protocole UDP :

1. Sur deux machines différentes (PC1 et PC2), lancez **socklab** en précisant que vous désirez uniquement travailler avec le protocole UDP:
   - **# ./socklab udp**

2. Sur chacune des deux machines, créez une Socket UDP. Ces deux sockets seront utilisées pour échanger des messages entre les deux stations:
   - **socklab-udp> sock (Equivalent de l'appel à la fonction C "socket")**

3. Notez bien l'identifiant et le numéro de port qui est retourné. Ils identifient de façon unique la Socket sur chaque machine.

4. Sur une troisième machine lancez l'analyseur de trames Wireshark.

a) Sur PC1, vous demanderez à émettre un paquet de données vers PC2, avec la syntaxe suivante:

- **socklab-udp> sendto Id-de-Socket @IP-machine numéro-de-port**

- Déterminez pour chacun des paramètres de la commande précédente, s'il faut indiquer la valeur locale (expéditeur) ou distante (récepteur).

- Ensuite, tapez la commande et validez la chaîne de caractères qui doit être transmise vers la machine distante.

- Sur la deuxième machine, demandez à recevoir un paquet de données sur la Socket précédemment créée, en précisant le nombre maximum d'octets à lire:
    - **socklab-udp> recvfrom Id-de-Socket nb-octets**

b) Analysez le (ou les) paquets engendrés par l'émission du message précédent. Déterminez à quel instant à lieu l'échange de trames sur le réseau. Précisez le rôle de chaque champs de l'entête UDP. Quelle observation pouvez vous faire au niveau des numéros de ports précisés dans les trames ?

3) Etude de la fragmentation des paquets IP (partie subsidiaire)

Vous venez d'étudier les principaux protocoles de la famille TCP/IP. Dans cette dernière manipulation, nous allons revenir sur un aspect particulier du protocole IP: la **fragmentation**.

La fragmentation est un mécanisme utilisé au niveau du protocole IP lorsque le paquet traité est trop "gros" pour être directement envoyé sur le réseau (un réseau Ethernet ne peut pas traiter des paquets de données de taille supérieure a 1500 octets). Dans ce cas le paquet doit être fragmenté en deux ou plusieurs paquets de taille plus petite.

Quand un paquet a été fragmenté lors de son émission, il doit évidemment être ré-assemblé lors de sa réception (toujours au niveau IP). Le flag DF (Don't Fragment) est utilisé pour autoriser la fragmentation. Quand il est positionné à 1, la fragmentation est interdite.

Sur une station, demandez l'émission d'un paquet UDP de 4600 octets à destination d'un port que vous aurez précédemment crée (voir manipulation précédente).

a) Analysez les paquets engendrés sur le réseau. En particulier, étudiez les entêtes IP et expliquez les rôles des champs TOTAL LENGTH, IDENTIFICATION, FRAGMENT OFFSET, et du flag MF ("More Fragment"). Répéter si nécessaire la manipulation avec d'autres tailles de paquets.

b) Que représente la valeur du champ FRAGMENT OFFSET ? Pour quel paquet le flag MF est-il à 0 ?

c) En TCP, Le flag MF est il utilisé ? Pourquoi ?

d) Donnez vos conclusions sur la fragmentation d'un échange UDP et d'un échange TCP.