

T.P. n°3 : Etude des protocoles UDP et TCP

*Dans ce TP, vous allez étudier les protocoles TCP/IP de niveau 4, à savoir UDP et TCP.
Vous utiliserez le programme **socklab** qui propose une interface interactive de manipulation des sockets.
Ainsi vous pourrez créer des sockets et effectuer des échanges de données.*

Déroulement du TP

On utilisera pour ce TP les 2 machines virtuelles **vm-fedora-1**, **vm-fedora-2**, connectées par leurs interfaces **eth0** en NAT

- Les captures et les analyses des paquets générés par les manipulations se feront sur la machine réelle (Windows Seven) à l'aide de l'analyseur de trames **Wireshark** (cf. documentation) sur l'interface VMware Network Adapter VMnet8
- Vous pouvez sauvegarder vos captures au format «Wireshark», les exporter en texte ou faire des captures d'écran (PicPick)
- Un outil intéressant de Wireshark pour visualiser les échanges se trouve dans le menu Statistics / **Flow Graph**

Vous devrez rédiger un compte-rendu, qui devra contenir les commandes tapées, les captures de paquets à l'aide de croquis temporel faisant apparaître les paquets et les champs intéressants des entêtes, des explications et des commentaires.

I – Installation de socklab

Socklab est un logiciel qui permet de manipuler les sockets sans les programmer. Son but est de comprendre leur fonctionnement.

Pour l'installer vous disposez de son code source en C dans le répertoire « socklab source ». Ce code dépend d'une librairie qu'il faut installer au préalable « readline-devel ». Procéder à l'installation de socklab en suivant les étapes suivantes :

- Transfert le répertoire « socklab source » dans votre machine virtuelle
- A l'aide de la commande `tar -zxvf`, extraire les sources de votre logiciel
- En se plaçant dans le répertoire des sources de socklab, installez readline-devel à l'aide de yum
- Installer socklab à l'aide de la commande `make`
- Vérifier que socklab fonctionne en le lançant
- Une fois terminer, colonner votre machine virtuelle

I – Le protocole UDP

On rappelle le format d'un segment UDP (en-tête + données)

1		32
16		
Port Source	Port Destination	
Longueur Segment	Checksum UDP	
Données		

- Lancez **socklab** sur vm-fedora-1 et vm-fedora-2 : - **# socklab**

- Sur chacune des 2 machines, créez un socket UDP : - **socklab> socket udp**

Ces deux sockets seront utilisés pour échanger des messages entre les deux stations.

Notez bien les identifiants qui sont retournés. Ils identifient de façon unique le socket sur chaque machine.

- Sur la troisième machine vm-fedora-3 lancez l'analyseur de trames Wireshark (menu Applications / Internet)

1) Vous allez maintenant affecter une adresse IP et un numéro de port à aux sockets de chaque machine :

- **socklab> bind Id-socket @IP-locale port**

- a) Utilisez l'adresse IP sur son interface **eth0** et un port attribué automatiquement pour la machine vm-fedora-1.
Précisez comment vous trouvez l'adresse IP et comment attribuer un port automatiquement. Notez bien ce port.
- b) Utilisez l'adresse IP * et le port 10000 pour la machine vm-fedora-2.
A quoi correspond et que signifie le paramètre * ? (voir la documentation de socklab)

2) Sur vm-fedora-1, vous allez émettre un paquet de données vers vm-fedora-2, avec la syntaxe suivante:

- **socklab> sendto Id-socket @IP-distante port message**

- a) Déterminez pour chacun des paramètres, s'il faut indiquer la valeur locale (expéditeur) ou distante (récepteur).
A quoi sert l'identificateur de socket ? Connait-on le numéro du socket distant ?
- b) Tapez la commande et validez la chaîne de caractères qui doit être transmise vers la machine distante.

3) Sur la deuxième machine, demandez à recevoir un paquet de données sur le socket précédemment créé, en précisant le nombre maximum d'octets à lire:

- **socklab> recvfrom *Id-socket nb-octets***

a) Analysez le (ou les) paquets engendrés par l'émission du message précédent avec Wireshark :

- Déterminez à quel instant à lieu l'échange de trames sur le réseau.
- Retrouvez les en-tête UDP et précisez le rôle et les valeurs de chaque champ.
- Quelles sont les informations passées à IP par UDP à l'émission d'un paquet (données + paramètres de service) ?

b) Recommencez les manipulations précédentes (« mission et réception de données), en permutant le rôle des deux machines (un socket peut servir à la fois pour émettre et recevoir des données).

- 4) Effectuez les manipulations, analysez les résultats et commentez les variantes suivantes :
- Demandez la réception avant l'émission des données.
 - Envoyez plusieurs paquets avant de demander leur réception.
 - Croisez l'émission des paquets : sur chacune des stations, demandez d'envoyer un paquet, puis demandez de lire le paquet envoyée par la station distante.
 - Notez ce qu'il se passe quand vous demandez à recevoir plus ou moins d'octets que la station distante en envoi.
 - Envoyez depuis `vm-fedora-1` un paquet UDP vers une machine inexistante. Vers un port non attribué de `vm-fedora-2`.
 - Envoyez 6 paquets de 5000 octets vers une machine de façon à "saturer" le récepteur (remplir son buffer de réception), puis essayez de les réceptionner (Sous socklab, tapez `#5000` à la place du message à envoyer).
- Vous pouvez connaître la taille du buffer de réception à l'aide de la commande options de Socklab.

+

II – Le protocole TCP : Etablissement d'une connexion

On rappelle le format d'un segment TCP (en-tête + données)

Effectuez les manipulations suivantes sur les machines `vm-fedora-1` et `vm-fedora-2` en capturant les trames avec Wireshark sur `vm-fedora-3`

- 1) a) Sur `vm-fedora-1`, créez un socket TCP : **- socklab> socket tcp**

Tapez la commande pour l'associer à n'importe qu'elle adresse IP locale et au port 2000 :

- socklab> bind Id-socket * 2000

Mettez ce socket à l'écoute de requêtes de connexions :

- socklab> listen Id-socket 5 (taille file d'attente : 5)

Mettez le enfin en attente d'acceptation d'une première connexion :

- socklab> accept Id-socket

- b) Sur `vm-fedora-2`, créez un socket TCP : **- socklab> socket tcp**

Tapez la commande pour l'associer à l'adresse IP de son interface `eth1` et à un port choisi dynamiquement :

- socklab> bind Id-socket @IP-locale 0 (précisez l'adresse IP et notez le n° de port affecté)

Tapez la commande pour effectuer une requête de connexion auprès du socket précédent sur `vm-fedora-1`

- socklab> connect Id-socket @IP-distante port (précisez les paramètres utilisés)

- c) Pourquoi le premier socket créé est-il dit « passif », et le second « actif » ?
Quels sont les rôles respectifs de ces deux sockets dans l'établissement de la connexion ?
- d) Analysez les paquets générés lors de l'établissement de la connexion et décomposez les étapes de cette connexion : enchainement dans le temps des demandes de services à TCP et des messages échangés.
- e) 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 ?
Remarque : les numéros de séquence donnés par Wireshark sont « relatifs », voir les vrais dans l'hexadécimal

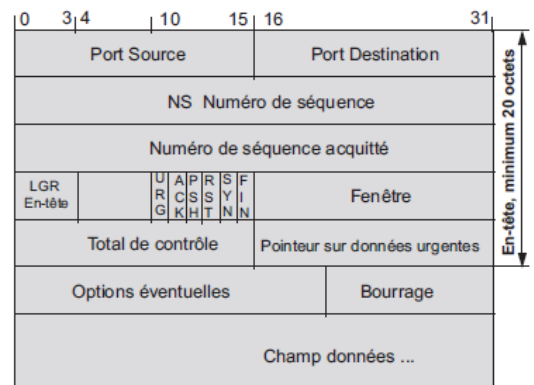
- 2) a) Regardez et commentez la liste des sockets en cours sur `vm-fedora-1` et `vm-fedora-2` : **- socklab> status**

Créez un autre socket « client » sur `vm-fedora-2` et ouvrez une autre connexion de la machine `vm-fedora-2` vers le même port destinataire de `vm-fedora-1`

Regardez de nouveau la liste des sockets en cours sur `vm-fedora-1` et `vm-fedora-2` : **- socklab> status**

Comment TCP associe-t-il les messages reçus aux différentes connexions en cours ?

- b) Créez un autre socket « client » sur `vm-fedora-2` et essayez de faire sur une demande de connexion (connect) vers un port de `vm-fedora-1` non attribué. Expliquez ce qui se passe.
- c) Créez un nouveau socket « serveur » sur `vm-fedora-1` et un nouveau socket « client » sur `vm-fedora-2` en faisant le **accept** sur `vm-fedora-1` après le **connect** sur `vm-fedora-2`. Commentez.



III – Le protocole TCP : Etude du séquençement et du contrôle d'erreur

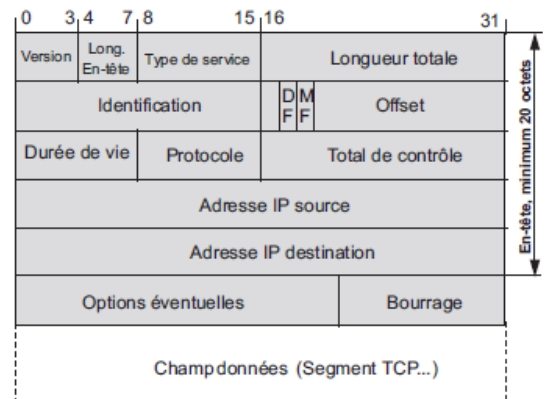
- 1) Etablissez une connexion par sockets TCP entre les deux machines, grâce aux commandes vues précédemment.
 - a) Echangez des données entre les deux machines grâce aux deux commandes **read** et **write** (cf manuel soklab) (à la place d'un message normal, vous pouvez utiliser la notation #nnn pour envoyer un message de nnn octets). Essayez en particulier avec une taille de donnée importante (5000 octets par exemple).
 - b) Analysez les paquets engendrés par le transport des données.
 - c) Expliquez le rôle des champs « Numéro de séquence » et « Numéro de séquence acquitté » dans l'entête des paquets TCP. Donnez les valeurs que vous avez observées dans plusieurs trames consécutives.
- 2) Envoyez un message vers une machine que vous aurez au préalable « débranchée » juste avant de le recevoir. (on peut par exemple fermer sa fenêtre VMware Player ce qui « suspend » la machine temporairement)
 - a) Observez ce qui se passe en capturant les paquets de la machine émettrice.
 - b) Expliquez les mécanismes utilisés dans ce genre de cas (perte d'un message), comparez avec le protocole UDP.

IV – Etude de la fragmentation des paquets IP

Vous venez d'étudier les principaux protocoles de la famille TCP/IP. Dans cette manipulation, nous allons étudier 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 à 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 réassemblé lors de sa réception (toujours au niveau IP). Il arrive parfois que la machine à laquelle sont envoyés les datagrammes, ne possède pas un mécanisme de réassemblage au niveau IP. Dans ce cas, il faut interdire la fragmentation des datagrammes qui lui sont destinés. Le flag DF (Don't Fragment) est utilisé à cet effet. Quand il est positionné à 1, la fragmentation est interdite. Il se peut alors qu'il soit impossible d'acheminer un paquet.



Structure du datagramme IP

Le but des manipulations suivantes est d'analyser le mécanisme de fragmentation et de réassemblage.

- 1) Lancez Wireshark sur la machine vm-fedora-3.

Effectuez à l'aide de sockets UDP l'émission d'un paquet UDP de 4600 octets de vm-fedora-1 à destination d'un port de vm-fedora-2 précédemment créé (cf manipulations du I).

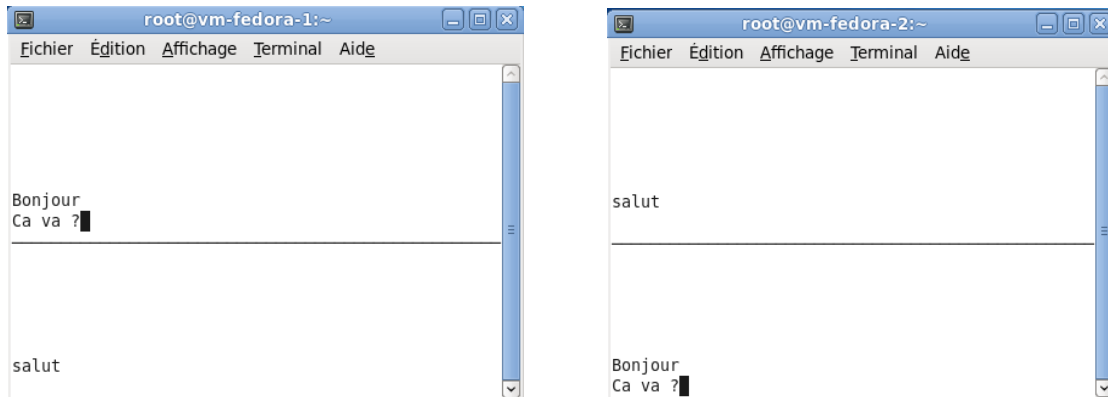
 - a) Analysez les paquets engendrés sur le réseau. En particulier, étudiez les entêtes IP et expliquez les rôles des champs « Longueur totale », « Identification », « Offset », et du flag MF ("More Fragment"). Répéter si nécessaire la manipulation avec d'autres tailles de paquets.
 - b) Que représente la valeur du champ « Offset » ? Pour quel paquet le flag MF est-il à 0 ?
- 2) Refaites maintenant ces manipulation en utilisant TCP et notez les différences.
 - a) Le flag MF est-il utilisé ? Pourquoi ?
 - b) Donnez vos conclusions sur la fragmentation d'un échanges UDP et d'un échange TCP. Justifiez les différences observées.
 - c) Vérifier les conclusions concernant les champs « Numéro de séquence » et « Numéro de séquence acquitté » de l'entête de TCP étudiés précédemment dans le cas des 4600 octets de données.

V – Observation de la commande *talk*

Ouvrez des terminaux sur les machines *vm-fedora-1* et *vm-fedora-2* et capturez le trafic sur *eth1* depuis *vm-fedora-3*

Tapez la commande suivante sur *vm-fedora-1* pour demander une session de « *talk* » : `- # talk root@vm-fedora-2`
Un message vous invitant à répondre s'affiche sur le terminal de *vm-fedora-2*. Tapez-y : `- # talk root@vm-fedora-1`

1) Faites des échanges de messages entre les deux machines et capturez les trames avec Wireshark sur la troisième



2) Analysez les paquets capturés et expliquez en le justifiant ce qu'il se passe au niveau du réseau :
protocole transport utilisé, ouverture de connexion, échange des caractères, fermeture de connexion, ...

VI – Observation de la commande *rlogin*

Ouvrez un terminal sur la machine *vm-fedora-1* et capturez le trafic sur *eth1* depuis *vm-fedora-3*

Connectez-vous en *rlogin* depuis *vm-fedora-1* :
à la machine *vm-fedora-2* sous le compte *etudiant*
(mot de passe : *etudiut*)

```
[root@vm-fedora-1 ~]# rlogin -l etudiant vm-fedora-2
Password: etudiut
Last login: Thu Sep  9 19:57:06 from vm-fedora-1
[etudiant@vm-fedora-2 ~]$
```

- 1) Procédez à une ou deux commandes Unix à travers ce login.
- 2) Enumérez les messages échangés lors de cette manipulation et expliquez ce qui se passe au niveau du réseau.
protocole transport utilisé, ouverture de connexion, échange des caractères, fermeture de connexion, ...
- 3) Comment les commandes sont-elles envoyées à destination ?
- 4) Pourquoi les caractères des commandes sont-ils renvoyés à la source ?

VII – Etude du mécanisme TCP de retransmission Go-Back-N

Lisez les pages 44 à 48 du document PDF « TCP et UDP » dans le dossier « Documentation \ Docs TP1 »

- 1) a) Pourquoi dit-on que le Go-Back-N est un mécanisme de retransmission globale sur TCP ?
b) Dans quelle(s) situation(s) est-il approprié ?
- 2) Ouvrez l'applet Java suivante : http://media.pearsoncmg.com/aw/aw_kurose_network_2/applets/go-back-n/go-back-n.html
 - a) Essayez d'envoyer plus de 5 segments consécutifs ? Que se passe-t-il ?
 - b) Envoyez 5 segments consécutifs, cliquez sur « Stop Animation » avant l'arrivée destination du 1^{er} segment, supprimez le 3^{ème} segment (clic droit dessus puis « kill Packet »), relancez avec « Resume » et observez (patientez une minute)
Décrivez et expliquez ce qui s'est passé.
 - c) Envoyez 5 segments consécutifs, cliquez sur « Stop Animation » après le départ du dernier acquittement retourné, supprimez les acquittements 2 et 4, relancez et observez. Décrivez et expliquez ce qui s'est passé.
 - d) Même question en supprimant d'abord le 4^{ème} segment envoyé et ensuite le 2nd acquittement retourné.