# TD1 maîtrisez les outils réseaux (1h)

## Exercice 1 : installation

Pour réaliser ce TD1 , il faut installer plusieurs outils qui vous seront utiles dans le reste de la séquence. Il vous faut aussi 1 machine sous windows et une machine sous Debian

-Installer sur windows : Hercules et Wireshark

-Connectez-vous en ssh sur votre debian et installer sur Debian tcpdump

## Exercice 2 : Utilisation des outils de filtrage

-Ouvrez une nouvelle fenêtre terminale de votre Debian. Puis démarrer l’écoute du port 9012 avec tcpdump.

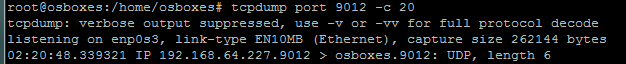
-Ouvrez whireshark et préparez un filtre UDP sur le port 9012

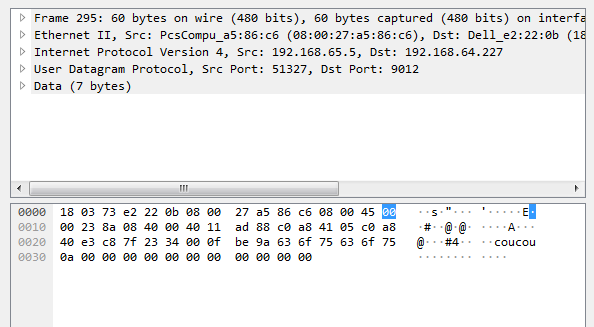
## Exercice 3 : Analyse de Communication UDP

-Démarrer hercules et paramétrée le en serveur UDP sur le port d’écoute 9012

-Envoyer un paquet UDP avec le message coucou vers hercules depuis votre machine Debian avec la commande echo voir (partie outil réseaux)

-Retrouvez votre trame « coucou » dans whireshark et tcpdump puis faites en une capture d’écran





Capture tcpDump :

Capture wireshark :

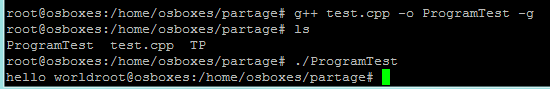
# TD2 Premier Programme C++ (Envoie message UDP) (2h)

## Exercice 1 : installation

Pour programmer un c++ sous debian il faut suivre un petit tuto d’initiation au c++ sous debian,

-faire la première partie du tuto dans les ressources SN qui se nomme : Compiler un projet c++ sous debian ( ne faire que la première partie : Compiler un fichier source cp++ sous debian ) uniquement.

-Lancer votre programme et afficher la capture d’écran



Capture d’écran du lancement de mon programme.

## Exercice 2 : Mon premier programme UDP

Dans cet exercice on va créer son premier programme c++ qui enverra un message à notre serveur hercules UDP en écoute sur le port 9012.

-reprendre le cours partie : **(**Création d’un socket**)**

-en ssh sur votre debian créer un fichier sendUDP.cpp avec nano.

-Ecrire le code pour envoyer le message "coucou je suis un client" sur ip et le port de votre serveur UDP. Hercule

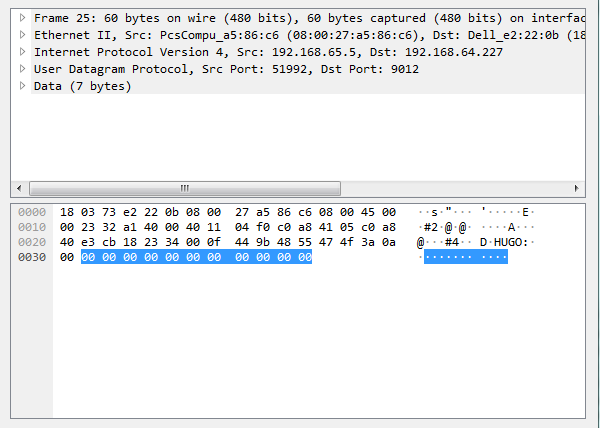
-compiler et lancer votre programme avec gcc voir exercice 1

-Afficher la capture d’écran d’hercules qui reçoit votre message

Capture d’écran Hercules



-Afficher la capture d’écran de votre message dans wireshark windows.

Capture d’écran wireshar

# TD3 Programmation Serveur UDP C++

## Exercice 1 : Création d’un serveur d’écoute

-Pour programmer un c++ sous debian il faut suivre un petit tuto d’initiation au c++ sous debian si cela n’est pas déjà fait

-reprendre le cours partie : **(**Programmer un programme en écoute (serveur)**)**

-Créer et compiler un program qui ouvre une socket et bind sur le port 9013

(Attention à mettre un getline ou sleep ) pour ne pas que votre programme se ferme trop vite. N’oublier pas le close(sock) en fin de programme

-Lancer votre programme Afficher la liste des ports ouverts sur votre machine debian Capture d’écran Netstat debian

S:\partage\TP\netstat.PNG

-Depuis hercules envoyer un message à votre serveur debian

|  |
| --- |
| Capture d’écran de TCPDump sous debian pour voir votre message arrivé  S:\partage\TP\Capture tcpdump TD3.PNG |

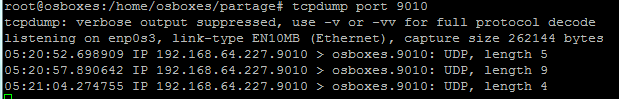
## Exercice 2 : Création d’un serveur d’écoute et lecture de message

-reprendre le programme de l’exercice 1 et le cours **(**Réception de message UDP.**)**

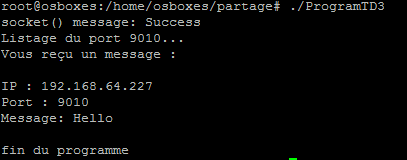
-Ajoutez à votre programme une boucle qui lit la socket uniquement ci celle-ci reçoit un message. Fermer le programme si elle reçoit le message quit (c’est votre premier protocole de couche 7 propriétaire. ;)

-Envoyer depuis hercules les messages suivants : Hello , aurevoir , quit

-Afficher la capture d’écran de TCPDump pour voir vos messages Capture TCPdump



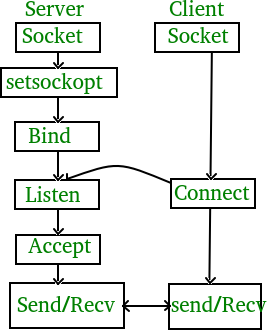
1. Afficher la capture d’écran de votre console pour voir vos messages Capture Console



# TD4 Programmation Serveur TCP C++

Reprenez les mécanismes des 3 premiers TD mais adaptez les en réalisant le même travail mais avec un protocole TCP. Il faudra faire quelque recherche sur google.

Bien relire le cours sur la couhe 4 Transport pour comprendre le mécanisme connecté TCP

* 1. Réalisez un petit client TCP qui envoie des messages à hercules Pour vous aider voici la démarche

-Il faut créer la **socket** : int socket(AF\_INET, SOCK\_STREAM, IPPROTO\_TCP);

-Il faut se **connect** à un serveur : intconnect(sock, const struct sockaddr \*addr, socklen\_t addrlen);

-il faut **send** un message: send(sock , hello , strlen(hello) , 0 );

* 1. Réalisez un petit serveur TCP qui reçoit les messages d’hercules.

-Il faut créer la **socket** : int socket(AF\_INET, SOCK\_STREAM, IPPROTO\_TCP);

- Il faut des options: **setsockopt**(server\_fd, SOL\_SOCKET, SO\_REUSEADDR | SO\_REUSEPORT, &opt, sizeof(opt))

-il faut être lié (bind) **bind**(sock, (struct sockaddr \*)&address, sizeof(address)

-Il faut en écoute **listen**(sock, 3)

-il faut accept : **accept**(sock, (struct sockaddr \*) &address, (socklen\_t\*)&addrlen)

-Puis il faut read : valread = **read**( new\_socket , buffe

# TP1 Programmation Client/Serveur TCP/IP C++

## Exercice 1 : Création d’un serveur de traitement de l’information

Réaliser un programme qui reçois en UDP un message de la forme

Chiffre :Chiffre :Chiffre etc…. Il doit répondre au client en calculant la somme des chiffres envoyés. (vous pouvez si vous le souhaitez changer ce que fais le server selon vos souhaits mais le serveurs doit décoder un message faire un traitement et répondre. )

Screen shot de des échanges wireshark

## Exercice 2 : Création d’un serveur TCP multi Client

Réaliser un programme qui reçoit en TCP des clients. une fois connecté les clients peuvent envoyer et recevoir des réponse automatique (comme un robot). ils peuvent se déconnecter quand ils envois un code de votre choix. Le serveur doit être codé avec des threads car il doit y avoir plusieurs clients qui peuvent se connecter

Screen shot de des échanges wireshark d’un test d’echange de message