Compte rendu TP1 Mode Connecté

Nom : Ibrahim Lahlou | IDSCC4

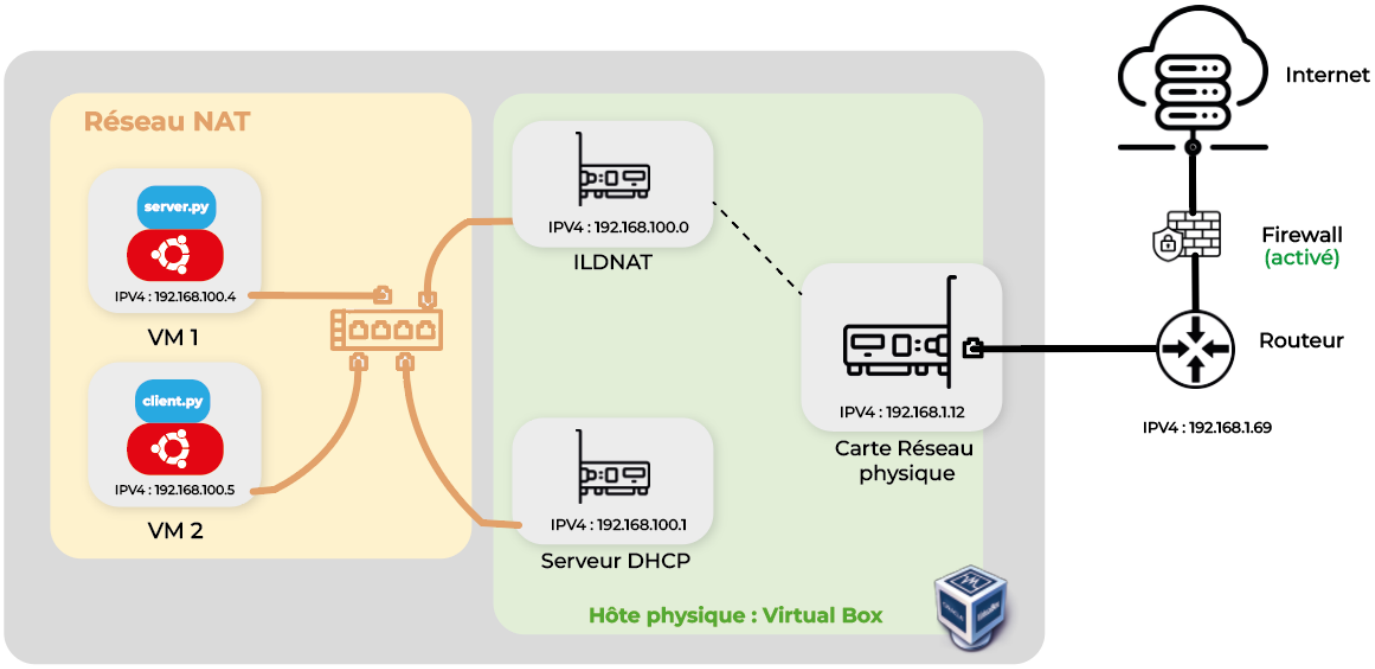
0. Programmation de Socket

Création d'un programme communicant des sockets avec python entre les deux machine VM1 et VM2 des instances d'Ubuntu server

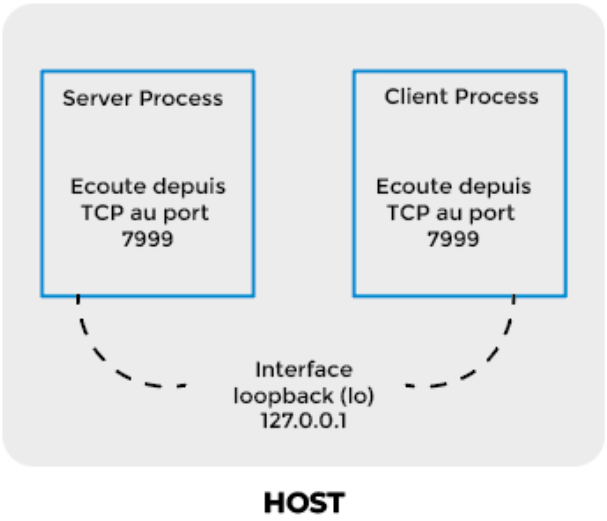
.

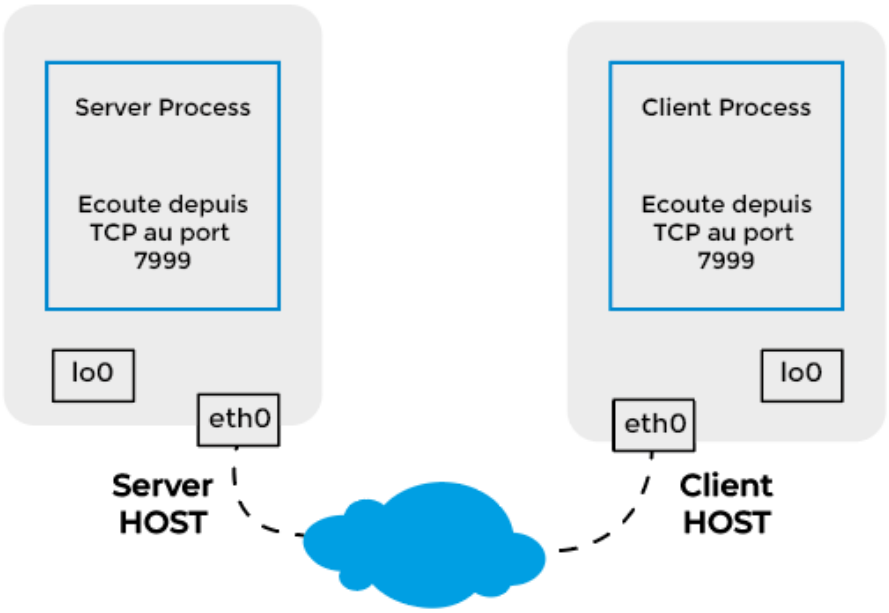
1 . Architecture réseau de cette manipulation

Réseau NAT



Les modèles d'architecture de communication

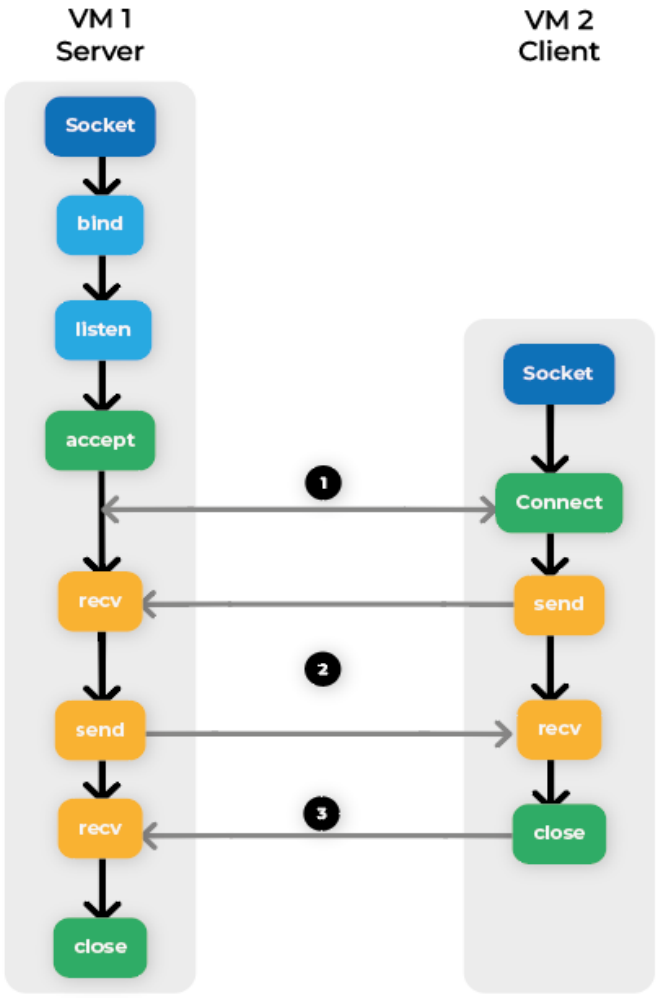




Dans ce tp on utilisera l'interface réseau ethernet "eth0" pour communication entre sockets .

2. Architecture Client-Server

Dans le module socket de python



1. Etablissement de connection (3-vay handshake)liason à trois voies
2. Transfère de donnée entre client et server
3. Fermeture de la communication entre client et server

Au mode Connecté , on spécifie un socket de flux. pour une communication fiable

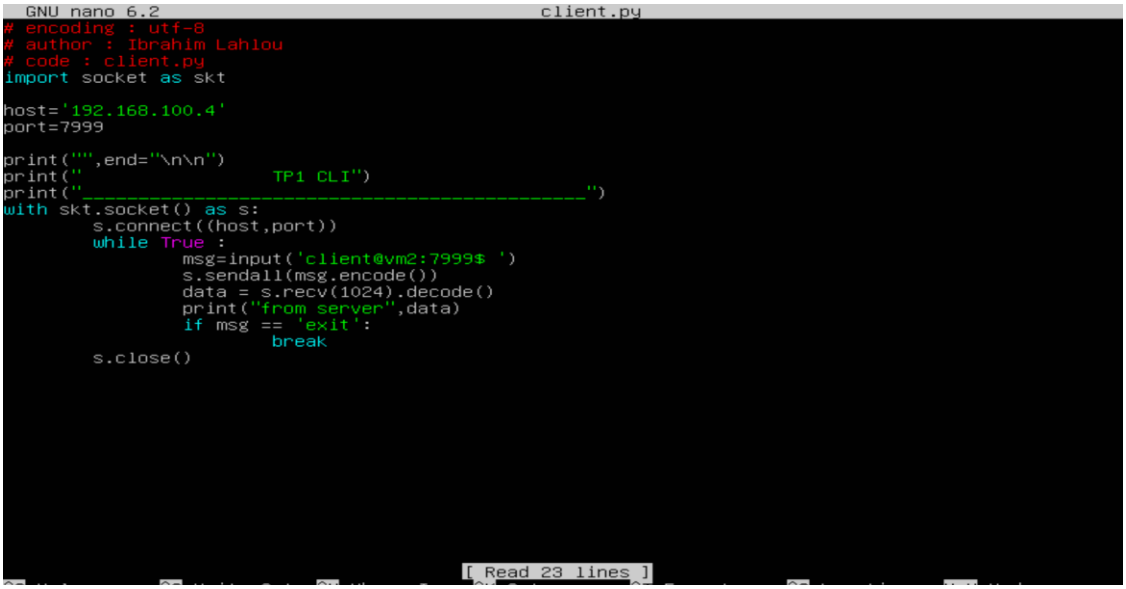
3. Programmation

Le module socket de Python fournit une interface à l'API des sockets de Berkeley.

Python fournit une API pratique et cohérente qui mappe directement aux appels système, leurs homologues C.

Voir la documentation de python-socket :
Programmation de sockets en Python (Guide) - Real Python

3.1. Programmation coté client



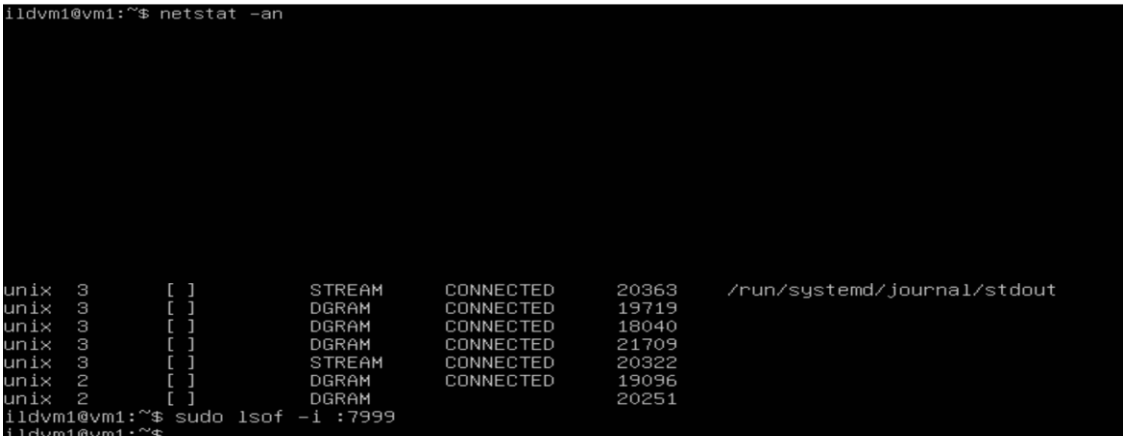
3.2 Programmation coté serveur

SOCK_STREAM : Spécifie un socket de flux. Doit être fourni lorsqu'un client ou un serveur basé sur une connexion de streaming est nécessaire et sera utilisé pour une communication fiable , utilisant le protocole TCP

4. Éxecution du Programme

0. Verification du port

netstat -an
sudo lsof -i :7999



Le port n'est pas occupé , on peut donc l'utilisez pour la communication client-server

sinon on pourra utilisez la commande kill

sudo kill 12345

1. Exécution du server

Server :

le server attends le client pour qu'il se connecte ...

2. Exécution du client

Client :

le client se connecte au server

Server :

le server accepte la connexion du client

3. Transfère de donné depuis le client
   Client :

```
                TP1 CLI
_____________________________________________
client@vm2:7999$ Hello vm1
_
```

   Envoie de donné , le client attends le retour du server

Server :

```
                  TP1 Server
_____________________________________________________
Listening ...
Connected by : ('192.168.100.5', 36222)
from client 'Hello vm1'
server@vm1:7999$ _
```

Réception de donné , le server peut répondre

4. Transfère de donné depuis le server

Server :

```
                  TP1 Server
_____________________________________________________
Listening ...
Connected by : ('192.168.100.5', 36222)
from client 'Hello vm1'
server@vm1:7999$ Hello ! ,What's up VM2
_
```

Envoie de donné , le server attends le retour du server

Client :

```
                TP1 CLI
_____________________________________________
client@vm2:7999$ Hello vm1
from server Hello ! ,What's up VM2
client@vm2:7999$ _
```

Réception de donné , le client peut répondre

5. Extinction de la session

Client :

```
                TP1 CLI
_____________________________________________
client@vm2:7999$ Hello vm1
from server Hello ! ,What's up VM2
client@vm2:7999$ exit
from server Hello vm1
ildvm2@vm2:~$ _
```

Le client mets fin à la communication