TP 1 Introduction à la programmation réseau

Table des matières

Objectifs

- Analyser l'état des connexions de processus;
- Travail à rendre : pour chaque question/étape prenez une capture d’écran *complète (les captures d'écran partielles ne seront comptabilisés)*.

TP 1 Réseau M1

Afin de réaliser le TP, vous devez télécharger et installer python 2.7.9 (*https://www.python.org/ftp/python/2.7.9/python-2.7.9.msi*)

Les codes présentés dans ce support sont téléchargeables à partir du lien suivant *https://drive.google.com/file/d/1bUdNvVk3ap2pjOK2zyy8R61g_nbVF8hM/view?usp=sharing*

1. Interopérabilité

Question 1

- Lancez votre invité de commande (*CMD*) et exécutez la commende suivante pour afficher les connexions établies entre les applications dans votre machine et d'autres de processus dans des machines distantes:

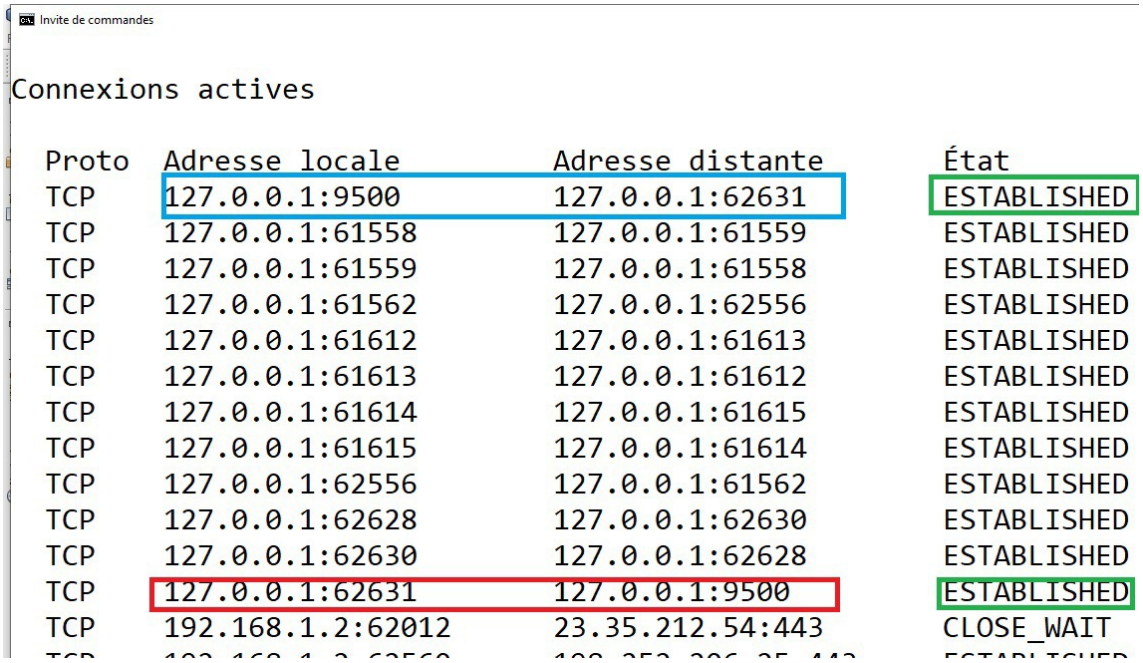
```
netstat -n
```

Remarque : Une connexion peut être établie entre deux applications qui fonctionnent dans la même machine.

```
Proto  Adresse locale         Adresse distante       État
TCP    127.0.0.1:61558        127.0.0.1:61559        ESTABLISHED
TCP    127.0.0.1:61559        127.0.0.1:61558        ESTABLISHED
TCP    127.0.0.1:61562        127.0.0.1:62556        ESTABLISHED
TCP    127.0.0.1:61605        127.0.0.1:61562        TIME_WAIT
TCP    127.0.0.1:61612        127.0.0.1:61613        ESTABLISHED
TCP    127.0.0.1:61613        127.0.0.1:61612        ESTABLISHED
TCP    127.0.0.1:61614        127.0.0.1:61615        ESTABLISHED
TCP    127.0.0.1:61615        127.0.0.1:61614        ESTABLISHED
TCP    127.0.0.1:62556        127.0.0.1:61562        ESTABLISHED
TCP    127.0.0.1:62562        127.0.0.1:5357         TIME_WAIT
TCP    192.168.1.2:62012      23.35.212.54:443       CLOSE_WAIT
TCP    192.168.1.2:62557      40.69.216.129:443      ESTABLISHED
TCP    192.168.1.2:62558      93.186.135.58:80       TIME_WAIT
TCP    192.168.1.2:62560      198.252.206.25:443     ESTABLISHED
TCP    192.168.1.2:62564      52.202.3.127:443       ESTABLISHED
TCP    192.168.1.2:62566      52.169.123.48:443      TIME_WAIT
TCP    192.168.1.2:62567      23.50.163.76:80        ESTABLISHED
```

Afin d'analyser facilement le résultat de la commande, redirigez le résultat de netstat vers un fichier de sortie (par exemple : *netstat -n>FichierSortie.txt*)

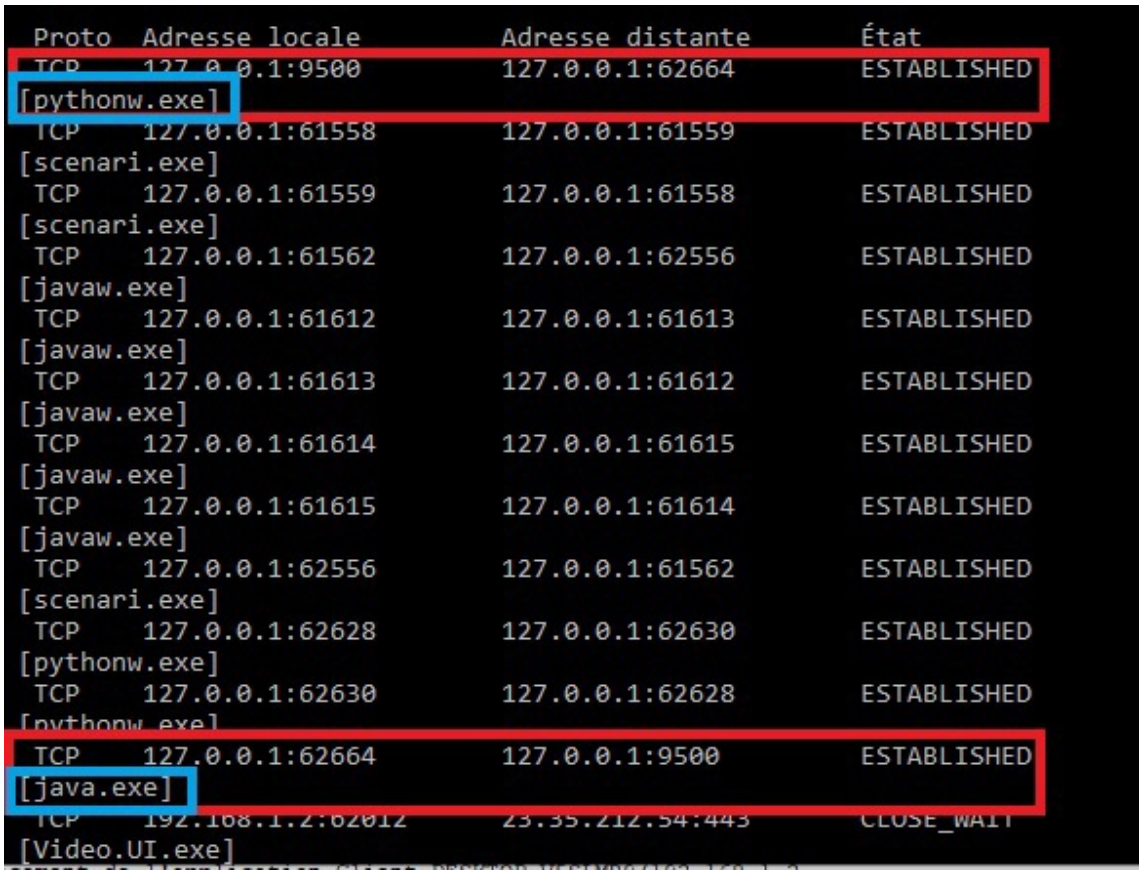
- Exécutez le programme dans *Listing 2* (Serveur.py) avant de lancer le programme de *Listing 1* (Client .py).
- Exécutez la commende *netstat -n* à nouveau et vérifiez s'il y a un nouveau port ouvert correspondant au port de votre application Serveur (9500).



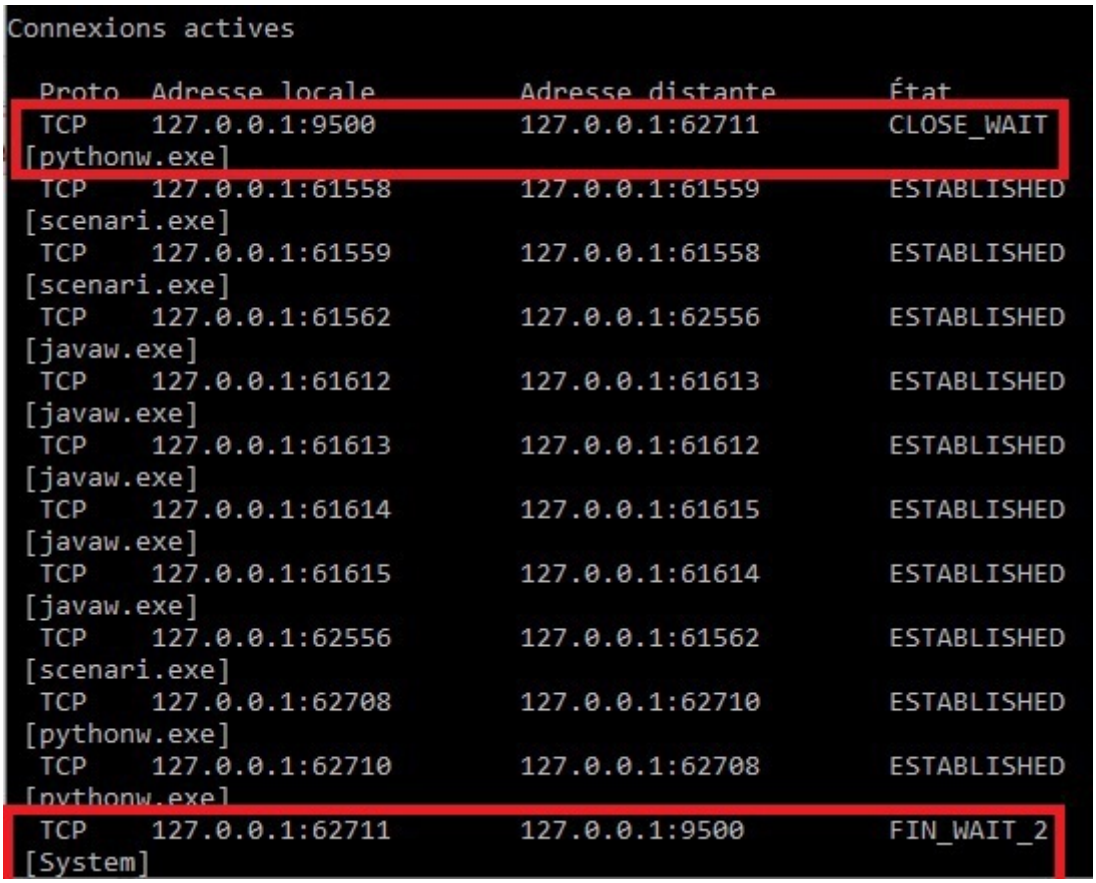
- *Quel est l'état de la connexion avec le serveur (voir la dernière colonne de l'affichage netstat)?*
- *Transmettez un message du client pour le recevoir sur le serveur (écrivez un message dans la console Client et appuyez sur Enter).*

Question 2

- Lancez votre CMD en *mode administrateur* et exécutez la commende *netstat -n -b* pour voir le nom des processus correspondants à chaque connexion.



- Transmettez un message *"Fin"* de votre client active pour clôturer la connexion et ensuite lancez *netstat -nb* pour voir l'état de connexion.



- Quel est l'état de la connexion avec le serveur *après la clôture de la connexion*?

Question 3

- Lancez deux instances du client avec une seule instance du serveur. Transmettez ensuite des messages a partir des deux clients.
- Que remarquez-vous (est ce que les messages des deux clients sont reçus)?

2. Multi-threading

Le programme précédant peut servir un seul client à la fois. Un serveur web par exemple doit servir plusieurs interlocuteurs simultanément. Ceci est réalisable grâce à la programmation *multi-processus* (*multi-threading*). Le programme python dans Listing 3 est un simple serveur multithread qui lance un processus chaque fois qu'un nouvel client est connecté.

Question 1

- Lancez le programme client (*Listing 1 du 1er exercice*) et le programme serveur multi-thread (*Listing 3*) ;
- Lancez un nouvel client sans fermer le 1er client et transmettez des messages depuis le 1er et 2em client (vérifiez que les messages des deux clients sont reçus par le serveur multithread python)
- Exécutez la commande *netstat -n*;
- *Que remarquez-vous ?*

Proto	Adresse locale	Adresse distante	Etat
TCP	127.0.0.1:9500	127.0.0.1:62778	ESTABLISHED
TCP	127.0.0.1:9500	127.0.0.1:62779	ESTABLISHED
TCP	127.0.0.1:61558	127.0.0.1:61559	ESTABLISHED
TCP	127.0.0.1:61559	127.0.0.1:61558	ESTABLISHED
TCP	127.0.0.1:61562	127.0.0.1:62556	ESTABLISHED
TCP	127.0.0.1:61612	127.0.0.1:61613	ESTABLISHED
TCP	127.0.0.1:61613	127.0.0.1:61612	ESTABLISHED
TCP	127.0.0.1:61614	127.0.0.1:61615	ESTABLISHED
TCP	127.0.0.1:61615	127.0.0.1:61614	ESTABLISHED
TCP	127.0.0.1:62556	127.0.0.1:61562	ESTABLISHED
TCP	127.0.0.1:62768	127.0.0.1:62770	ESTABLISHED
TCP	127.0.0.1:62770	127.0.0.1:62768	ESTABLISHED
TCP	127.0.0.1:62778	127.0.0.1:9500	ESTABLISHED
TCP	127.0.0.1:62779	127.0.0.1:9500	ESTABLISHED

Question 2

```
1 import socket
2
3 ConnexionAUnServeur = socket.socket() # Creation de socket
4 host = "127.0.0.1" # adresse de la machine distante
5 port = 9500 # numero de port
6 print "Console Client"
7 ConnexionAUnServeur.connect((host, port)) # connexion avec le serveur ( envoie
  flag SYN)
8 Message_A_Transmettre=""
9 while Message_A_Transmettre!="Fin":
10     Message_A_Transmettre=raw_input() # lecture depuis le clavier
11     ConnexionAUnServeur.send(Message_A_Transmettre) # transmettre le message
  vers le serveur
12 ConnexionAUnServeur.close() # cloture de la connexion avec le serveur (envoie
  flag FIN)
13 print "Fin du programme"
```

Listing 1 Programme client

```
1 import socket
2
3 SocketServeur = socket.socket()
4 port = 9500
5 # creation d'un socket en ecout sur l'interface 127.0.0.1, port=9500
6 SocketServeur.bind(("127.0.0.1", port))
7 SocketServeur.listen(1) # nombre de maximale de connexion qui peuvent etre mis
  en file d'attente
8
9 print "Lancement serveur" # instruction d'affichage
10 while True: # boucle infinie
11    # accept bloque le programme en attendant une connexion d'un client (la
  reception du SYN)
12    ConnexionAUnClient, addrclient = SocketServeur.accept()
13    # une fois que la connexion est recue, accept renvoie l'adresse du client et
  un objet socket
14    # permetant l'envoie et la reception sur cette connexion
15    print "Connexion de la machine = ", addrclient
16    MessageRec=""
17    while(MessageRec.strip().lower()!="fin"):# tanque le message recu est
  different de "fin"
18       MessageRec=ConnexionAUnClient.recv(1024) # recv permet de recevoir une
  sequence d'octets
19       print MessageRec
20
21    print "Deconnexion de :",addrclient
22    #ConnexionAUnClient.close() # coloture la connexion (envoie FIN ACK au client)
23
24
```

Listing 2 Programme serveur python

```
import socket
import thread

def Traiter_Connexion(connexion_avec_client,adresse_client):
   MessageRec=""

   print "Connexion de la machine = ", adresse_client
   try:
      while(MessageRec.strip().lower()!="fin"):
          MessageRec=connexion_avec_client.recv(1024)
          MessageRec=MessageRec
          print "Client" ,adresse_client," a dit :",MessageRec
   except:
      print "Deconnexion"
   print "Deconnexion de :",adresse_client
   try:
      connexion_avec_client.close()
   except:
      pass

SocketServeur = socket.socket()
host = socket.gethostname()
port = 9500
SocketServeur.bind(("127.0.0.1", port))

SocketServeur.listen(5)

print "Lancement serveur"
while True:
   ConnexionAUnClient, addrclient = SocketServeur.accept()
   thread.start_new_thread(Traiter_Connexion,(ConnexionAUnClient,addrclient))
```

Listing 3 Programme serveur python multi-thread