TP n°3 *Programmation socket*

Exercice 1. Serveur pour le service date/heure par UDP

Un socket est une interface de programmation qui permettent d'utiliser les protocoles de la couche transport.

Pour utiliser le protocole UDP par exemple, il suffit d'utiliser le constructeur `socket` avec en paramètre le type `SOCK_DGRAM`. Il est nécessaire d'associer un socket à un port, cela est possible avec la méthode `bind`

Le socket possède plusieurs méthodes pour envoyer (`sendto`) et recevoir des données (`recv, recvfrom`)

Examinez et taper le code suivant (pas de copier/coller). Allez sur la documentation `https://docs.python.org/3/library/socket.html` pour comprendre comment les données sont reçues et envoyées.

```
#!/usr/bin/python3

import socket
import datetime

BUFSIZE = 1024
def server (port ):
    sock = socket.socket (type= socket.SOCK_DGRAM )
    sock.bind(("0.0.0.0", port ))
    while True:
        data , addr = sock.recvfrom ( BUFSIZE )
        data = datetime.datetime.now()
        data = "%s\n" % data
        sock. sendto (data.encode(), addr)

server(5555)
```

Exercice 2. Vous allez pouvoir tester ce serveur, en vous y connectant grâce à l'utilitaire en ligne de commande `netcat` :

```
netcat -4 -u localhost 5555
```

Rien ne se passe ! En effet, il faut provoquer l'envoi d'un message au serveur pour que celui-ci vous réponde. Si vous entrez une ligne de texte, netcat l'encapsulera dans un message et enverra

celui-ci au serveur. Il vous suffit donc d'appuyer sur la touche Entrée. Si vous appuyez plusieurs fois, chaque fois une message sera envoyé et le serveur vous enverra une réponse.

Pour arrêter `netcat`, utilisez `Ctrl-C`.

Exercice 3. Client pour le service date/heure

Le code suivant permet de développer un client qui permet de communiquer avec le serveur précédent

```
#!/usr/bin/python3
import socket
BUFSIZE = 1024
def client (host , port ):
    sock = socket.socket(type=socket.SOCK_DGRAM )
    addr = (host , port)
    sock.sendto(b"", addr)
    data = sock.recv( BUFSIZE )
    print (data.decode (), end="")

client("localhost", 5555)
```

3.1 Testez ce client en l'invocant à la ligne de commande.

Exercice 4. Le module sys (voir `https://docs.python.org/3/library/sys.html`) exporte la variable argv qui contient une liste de chaînes : ce sont les éléments de la ligne de commande. Servez- vous en pour permettre à l'utilisateur de donner sur la ligne de commande un argument indiquant le nom ou l'IP de la machine où contacter le serveur.

4.1 Testez cette modification en interrogeant le serveur de votre voisin.

Exercice 5.

Pour l'instant le serveur ne sait fournir que la date. Vous allez en créer un nouveau qui saura répondre à différentes requêtes. Pour cela, vous allez dupliquer les codes que vous avez :

```
cp date-server.py ext-server.py
cp date-client.py ext-client.py
```

et vous modifierez ces nouveaux fichiers. Quelle autre information pourrait-on vouloir publier ? Par exemple l'identifiant de la personne qui fait tourner le serveur. Voici comment obtenir cette info :

```
import os
os.environ["USER"]
```

Voici comment le client envoie un message au serveur :

```
sock.sendto(b"", addr)
```

Notez que le message est vide. Pour permettre au client de faire différentes requêtes, il suffirait que le message contienne un mot indiquant l'information voulue :

```
sock.sendto(b"date", addr)
```

ou bien :

```
sock.sendto (b"user", addr)
```

De son côté le serveur doit examiner le mot reçu dans le message du client, et doit renvoyer l'information correspondante. Il faudrait que l'utilisateur puisse invoquer le client comme ceci :

```
./ext-client.py localhost date
./ext-client.py localhost user
./ext-client.py IP-DU-VOISIN date
./ext-client.py IP-DU-VOISIN user
```

où IP-DU-VOISIN serait remplacé par l'adresse IP de votre voisin, pour interroger son serveur.

Exercice 6. Broadcasting server

Il est également possible de broadcaster des messages de sorte que tous les clients d'un réseau local puissent les recevoir.

```
#!/usr/bin/python3

import socket
import datetime
import time
import os

def server(port):
    sock = socket.socket(type=socket.SOCK_DGRAM)
    sock.setsockopt(socket.SOL_SOCKET, socket.SO_BROADCAST,1)
    addr = ("255.255.255.255",port)
    while True:
```

```
        data = str(datetime.datetime.now())
        data = os.environ["USER"] + " "+ data
        sock.sendto(data.encode(),addr)
        time.sleep(1.0)


server(6666)
```

```
#! /usr/bin/python3

import socket

def client(port):
    sock = socket.socket(type=socket.SOCK_DGRAM)
    addr = ("255.255.255.255", port)
    sock.bind(addr)
    while True:
        data, saddr = sock.recvfrom(128)
        print(saddr, data.decode())


client(6666)
```