TP2 : Programmation Réseau Année : 2022/2023

1. Tester les deux programmes Serveur/Client
2. Ecrire les programmes Serveur et Client qui s’envoient des messages alternativement .

# **CLIENT 1 tcp**

import socket

UDP\_IP\_ADDRESS = "127.0.0.1"

UDP\_PORT\_NO = 1234

Message = "Hello, Server".encode()

clientSock = socket.socket(socket.AF\_INET, socket.SOCK\_DGRAM)

while True:

msg = str(input("Enter your message: "))

if msg.lower() == "bye" :

break

msg = msg.encode()

clientSock.sendto(msg, (UDP\_IP\_ADDRESS, UDP\_PORT\_NO))

# **SERVEUR1 UDP**

# Again we import the necessary socket python module

import socket

# Here we define the UDP IP address as well as the port number that we have

# already defined in the client python script.

UDP\_IP\_ADDRESS = "127.0.0.1"

UDP\_PORT\_NO = 1234

# declare our serverSocket upon which

# we will be listening for UDP messages

serverSock = socket.socket(socket.AF\_INET, socket.SOCK\_DGRAM)

# One difference is that we will have to bind our declared IP address

# and port number to our newly declared serverSock

serverSock.bind((UDP\_IP\_ADDRESS, UDP\_PORT\_NO))

while True:

data, addr = serverSock.recvfrom(1024)

print ("Message: " + data.decode())

# **CLIENT 1 tcp**

import socket

msgFromClient = "Hello UDP Server"

bytesToSend = str.encode(msgFromClient)

serverAddressPort = ("127.0.0.1", 20001)

bufferSize = 1024

# Create a UDP socket at client side

UDPClientSocket = socket.socket(family=socket.AF\_INET, type=socket.SOCK\_DGRAM)

# Send to server using created UDP socket

UDPClientSocket.sendto(bytesToSend, serverAddressPort)

msgFromServer = UDPClientSocket.recvfrom(bufferSize)

msg = "Message from Server {}".format(msgFromServer[0])

print(msg)

# **SERVEUR UDP**

import socket

localIP = "127.0.0.1"

localPort = 20001

bufferSize = 1024

msgFromServer = "Hello UDP Client"

bytesToSend = str.encode(msgFromServer)

# Create a datagram socket

UDPServerSocket = socket.socket(family=socket.AF\_INET, type=socket.SOCK\_DGRAM)

# Bind to address and ip

UDPServerSocket.bind((localIP, localPort))

print("UDP server up and listening")

# Listen for incoming datagrams

while(True):

bytesAddressPair = UDPServerSocket.recvfrom(bufferSize)

message = bytesAddressPair[0]

address = bytesAddressPair[1]

clientMsg = "Message from Client:{}".format(message)

clientIP = "Client IP Address:{}".format(address)

print(clientMsg)

print(clientIP)

# Sending a reply to client

UDPServerSocket.sendto(bytesToSend, address)

Ecrire un scanner de ports en Python.

#import pyfiglet

import sys

import socket

from datetime import datetime

#ascii\_banner = pyfiglet.figlet\_format("PORT SCANNER")

#print(ascii\_banner)

# Defining a target

if len(sys.argv) == 2:

# translate hostname to IPv4

target = socket.gethostbyname(sys.argv[1])

else:

print("Invalid amount of Argument")

# Add Banner

print("-" \* 50)

print("Scanning Target: " + target)

print("Scanning started at:" + str(datetime.now()))

print("-" \* 50)

try:

# will scan ports between 1 to 65,535

for port in range(1,65535):

s = socket.socket(socket.AF\_INET, socket.SOCK\_STREAM)

socket.setdefaulttimeout(1)

# returns an error indicator

result = s.connect\_ex((target,port))

if result ==0:

print("Port {} is open".format(port))

s.close()

except KeyboardInterrupt:

print("\n Exiting Program !!!!")

sys.exit()

except socket.gaierror:

print("\n Hostname Could Not Be Resolved !!!!")

sys.exit()

except socket.error:

print("\ Server not responding !!!!")

sys.exit()