PRACTICA Multicast

OBJETIVO:

El objetivo de esta práctica es:

Comprender la comunicación ente procesos mediante multidifusión de mensajes IP. La operación de ***multidifusión*** (***multicast*** en inglés), consiste en enviar un único mensaje desde un proceso a cada uno de los miembros de un grupo de procesos. La pertenencia a un grupo es transparente al emisor, es decir, el emisor no conoce el número de miembros del grupo ni sus direcciones IP.

Python proporciona una interfaz de datagramas para multidifusión IP a través de la librería *Socket* con la capacidad adicional de ser capaz de pertenecer a grupos multidifusión.

La configuración básica para unirse a un grupo ***multicast*** para recibir y enviar mensajes se puede conseguir con la siguiente parametrización:

```python
import socket
import struct

MCAST_GRP = '224.1.1.1'
MCAST_PORT = 5007

# socket.AF_INET: Internet Protocol v4 addresses
# socket.SOCK_DGRAM: UDP datagrams
# socket.IPPROTO_UDP: set UPD protocol
sock = socket.socket(socket.AF_INET, socket.SOCK_DGRAM, socket.IPPROTO_UDP)

# socket.SOL_SOCKET: Set property at socket level
# socket.SO_REUSEADDR: Reuse address
sock.setsockopt(socket.SOL_SOCKET, socket.SO_REUSEADDR, 1)

# use '' to listen and write on the socket
# use MCAST_GRP to listen only on the socket
sock.bind(('', MCAST_PORT))
sock.sendto(str.encode("Hello this is user 1"), (MCAST_GRP, MCAST_PORT))

# Tell the operating system to add the socket to the multicast group on all interfaces.
mreq = struct.pack("4sl", socket.inet_aton(MCAST_GRP), socket.INADDR_ANY)
sock.setsockopt(socket.IPPROTO_IP, socket.IP_ADD_MEMBERSHIP, mreq)

# Setting TTL
ttl = struct.pack('b', 1)
sock.setsockopt(socket.IPPROTO_IP, socket.IP_MULTICAST_TTL, ttl)
```

Es posible que para equipos Macintosh sea necesario utilizar esta parametrización en caso de que con el de la captura de pantalla de un error de IP duplicada:

```
sock.setsockopt(socket.SOL_SOCKET, socket.SO_REUSEPORT, 1)
```

Notar que en este caso no es necesario conectarse a la IP de un servidor fijo, sino que se utilizan las direcciones reservadas para este tipo de comunicación.

https://en.wikipedia.org/wiki/Multicast_address

En caso de que se quiera ejecutar el código en dos equipos diferentes conectados vía red, hay que indicar al socket sobre qué interfaz se quiere realizar el Multicast. Esto se consigue con la siguiente parametrización:

```
import socket
import struct
import time
import threading

MCAST_GRP = '224.1.1.1'
MCAST_IF_IP = '192.168.1.132'
MCAST_PORT = 5007

sock = socket.socket(socket.AF_INET, socket.SOCK_DGRAM, socket.IPPROTO_UDP)
sock.setsockopt(socket.SOL_SOCKET, socket.SO_REUSEADDR, 1)

sock.bind(('', MCAST_PORT))
ide="user1"
sock.sendto(str.encode(ide+": Entering user"), (MCAST_GRP, MCAST_PORT))

mreq = struct.pack("4s4s", socket.inet_aton(MCAST_GRP), socket.inet_aton(MCAST_IF_IP))
sock.setsockopt(socket.IPPROTO_IP, socket.IP_ADD_MEMBERSHIP, mreq)
```

En el que la variable MCAST_IF_IP identifica la IP del interfaz de red que se va a usar para establecer la comunicación en Multicast.

PRACTICA Multicast 1:

Probar y analizar el funcionamiento de la aplicación Multidifusión. Esta se implementa de una manera muy simple, las acciones que debería de hacer un participante en un grupo de multidifusión son:

- Primero se unirá al grupo,
- A continuación, enviará un mensaje para saludar al grupo
- Después se quedará a la espera de recibir mensajes de otros participantes, hasta que el mensaje recibido sea “Adiós”. (en ese momento todos los participantes saldrán del grupo de multidifusión en bloque)

PRACTICA Multicast 2:

Se realizará una aplicación de chat utilizando la multidifusión IP. En este caso cada miembro del grupo, podrá enviar sucesivamente tantos mensajes como desee al resto de participantes, hasta que este decida salir del grupo con el mensaje “*Quit*”. (El resto de participantes seguirán permaneciendo en el grupo de difusión, y solo saldrán de este cuando ellos mismo decidan dejarlo mediante el mensaje “*Quit*”).

Los participantes recibirán junto con el mensaje, un identificador o nombre del emisor del mensaje. (Por tanto, a cada participante deberá tener un identificador o nombre dentro del grupo)

A continuación, se proporcionan unos ejemplos de posibles salidas para dos usuarios conectados al chat.

User1 console:

```
user2: Entering user
Message <quit>: Hola, soy user1
user1: Hola, soy user1
Message <quit>: Alguien por ahi?
user1: Alguien por ahi?
user2: Entering user
user2: Hola? Soy user2
Message <quit>: Hola user 2
user1: Hola user 2
Message <quit>: Bienvenido
user1: Bienvenido
user2: Gracias!
user2: Ahora me voy
user2: quit
Message <quit>: vaya... otra vez solo
user1: vaya... otra vez solo
Message <quit>: quit
user1: quitStop writer

Stop reader
```

User2 console:

```
Message <quit>: Hola? Soy user2
user2: Hola? Soy user2
user1: Hola user 2
user1: Bienvenido
Message <quit>: Gracias!
user2: Gracias!
Message <quit>: Ahora me voy
user2: Ahora me voy
Message <quit>: quit
user2: quitStop writer

Stop reader
```

PRACTICA ENTREGABLE:

Vamos a implementar un módulo de comunicaciones y anuncios globales en el entorno de un servicio de juegos online.

El módulo debe permitir comunicar a todos los jugadores los siguientes eventos:

- Anunciar el inicio del juego
- Enviar actualizaciones de las puntuaciones en forma de ranking
- Avisar del abandono del juego de algún jugador
- Felicitar públicamente al ganador
- Anunciar el fin del juego

ENTREGA:

Se deberá entregar un documento donde se recojan y analicen los siguientes puntos:

a) Describir brevemente las partes del código e instrucciones para ejecutarlo
b) Capturas de pantalla del juego
c) Código de la práctica planteada