INSTITUTO POLITÉCNICO NACIONAL
ESCUELA SUPERIOR DE CÓMPUTO

ALUMNOS:
ARISTA BOJORGES VICTOR USIEL
HERNANDEZ RADILLA JOSÉ ANGEL
VALLEJO SERRANO EHECATZIN

MATERIA
aplicaciones para comunicaciones en red

PRACTICA
BROADCAST

En python tenemos dos módulos que nos permiten tratar con hilos. El módulo thread es el de más bajo nivel y tiene algunas funciones que nos permiten trabajar con hilos. El módulo threading es de más alto nivel y tiene la clase Thread, que representa un hilo. Vamos a ver algunos ejemplos con ambos módulos

Arrancando un hilo

Para arrancar un hilo, tenemos la función *start_new_thread()* del módulo *thread*. A esta función únicamente debemos pasarle la función *python* que queremos que se ejecute en un hilo y la *tupla* que hará de parámetros de dicha función. Dicho de otra forma, si definimos la función

```
def funcion (inicio, fin):
```

y queremos arrancarla como un hilo, pasándole, por ejemplo, un 3 y un 11 como parámetros de inicio y fin, haremos la llamada a la función *start_new_thread()* de la siguiente forma

```
import thread
...
thread.start_new_thread(funcion, (3,11))
```

y eso es todo, la función comenzará a ejecutarse en un hilo separado y hará lo que codifiquemos en ella. Como ejemplo, haremos una función que cuente desde inicio a fin, de uno en uno, esperando 0.1 segundos entre cuenta y cuenta. El *main* lanzará el hilo y se pondrá a su vez a contar. Debemos ver en pantalla como van contando en paralelo el *main* y el hilo. El código de la función puede ser este

Espera de unos hilos por otros

Si necesitamos que unos hilos esperen por otros, *python* en su módulo *thread* nos ofrece la función *thread.allocate_lock()* que nos devolverá un objeto de tipo *thread.LockType*. Los hilos deben pedir a este objeto permiso para hacer su tarea y liberarlo cuando quieran dar paso a otros hilos. Para ello, llaman a los método *acquire()* y *release()*

Un hilo simple

Un módulo de más alto nivel que nos permite hacer hilos en *python* es *threading*. Este módulo, entre otras cosas, tiene una clase *Thread*, que es la que representa el hilo. Para hacer un hilo, debemos heredar de ella y definir el método *run()*. Lo que pongamos en ese método se ejecutará en un hilo aparte. Para arrancar el hilo, debemos instanciar la clase hija de *Thread* que hayamos hecho y llamar a su método *start()*. Los pasos básicos con estos

Event

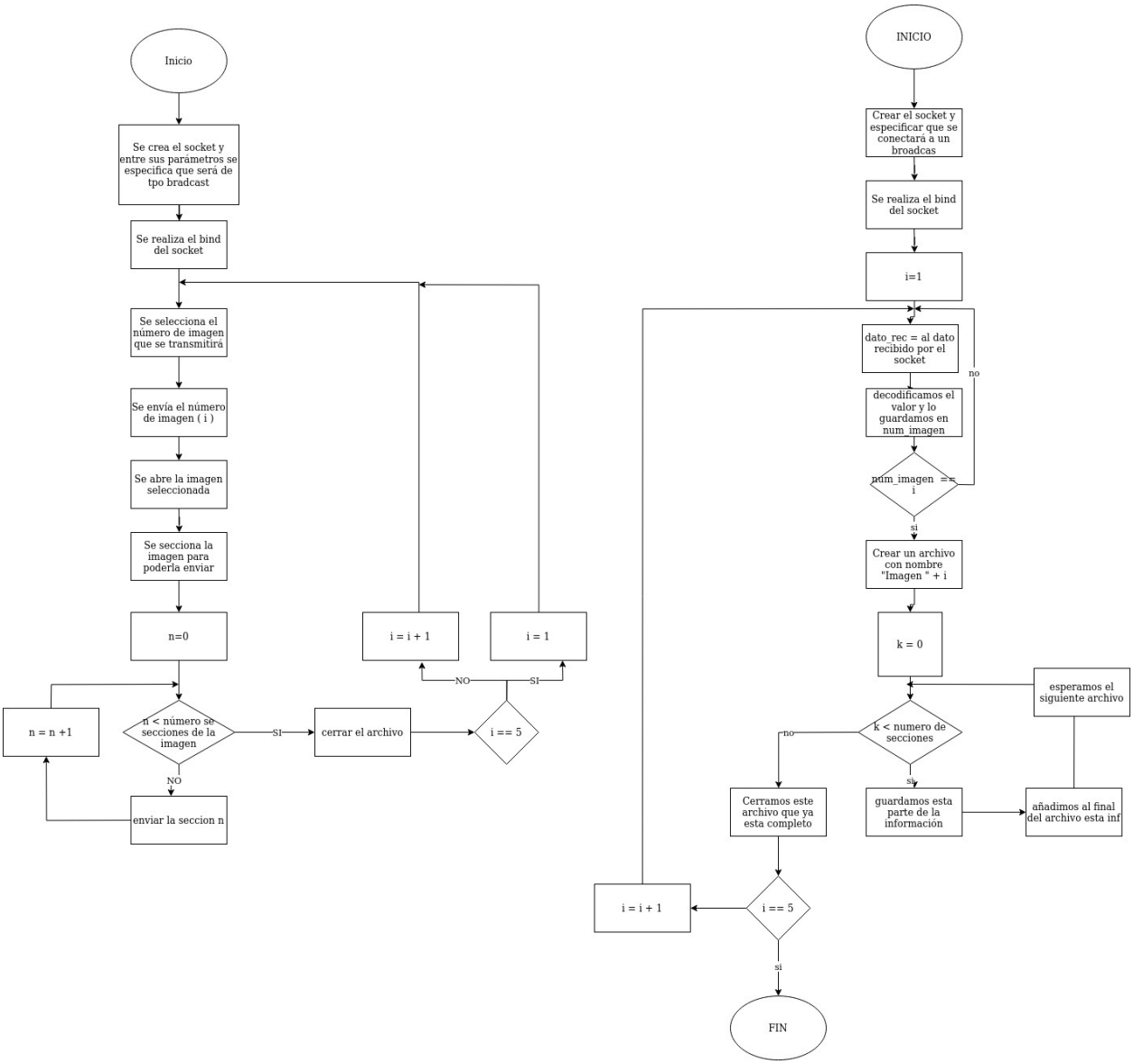
La forma más fácil de hacer que un hilo espere a que otro hilo le avise es por medio de *Event*. El *Event* tiene un *flag* interno que indica si un hilo puede continuar o no. Un hilo llama al método *Event.wait()* y se queda bloqueado en espera hasta que el *flag* interno de *Event* se ponga a *True*.

Otro hilo llame a *Event.set()* para poner el flag a *True* o bien a *Event.clear()* para ponerlo a *False*.

DESCRIPCIÓN DE LA PROBLEMÁTICA

se tendran dos partes, por el lado del servidor realizará el envió continuo de las tramas de 5 imágenes en bucles por su socket sin importar cuantos estén conectados.

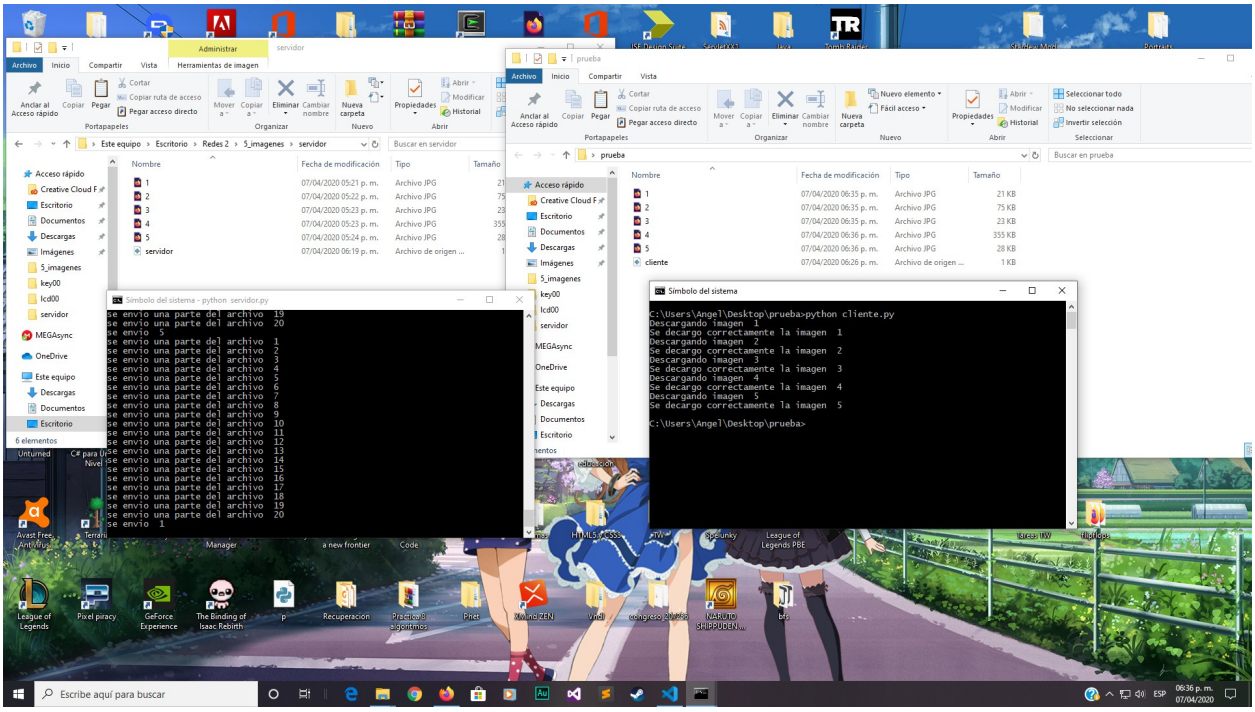
Por el lado del usuario este tendrá que recibir las 5 imágenes y finalizar con la recepcion de información.

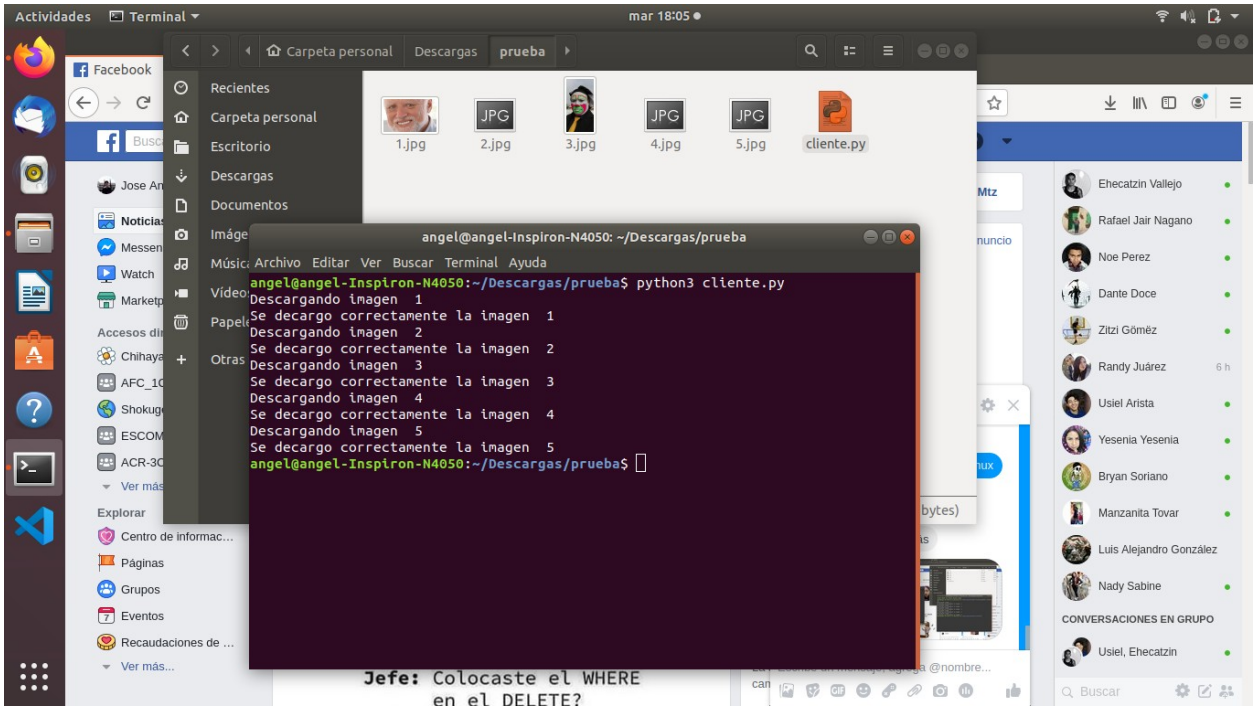


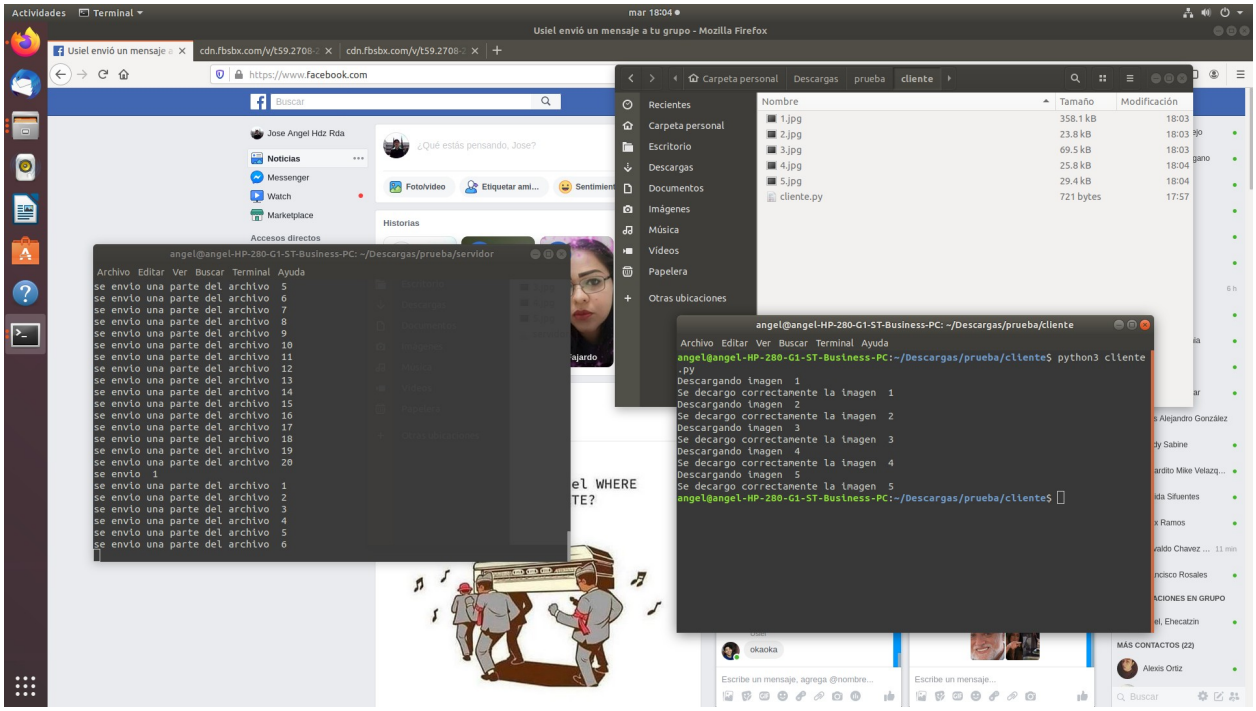
Descripción de diagrama de servidor:
iniciamos creadno el socket y en las especificaciones le ponemos que será de tipo broadcast.
Continuamos realizando el correspondiente bind.
Aquí es donde iniciará el bucle infinito, este inicia seleccionando la imagen en la que vamos, al ser la primera asuem el valor de 1
se el envia a los usuaros la imagen que va a empezar a transmitir
Se abrira el archivo donde se encuentra la imagen seleccionada, como no se puede enviar en una sola trama por le tamaño se divide o secciona en otras de menor tamaño
empezamos un bucle donde una a una las secciones son enviadas el por el método sendto a todos los usuarios que se encuentren escuchando.
Cuando se termina de enviar cerramos el archivo y reasignamos el valor de la imagen para seguie enviando, ya sea la siguiente o la del principio en caso de que se haya enviado la última.

Descripción del User:
De misma manera se crea un socket que por el cual podrá recibir la información, también se le tiene que especificar que sera desde un broadcast. Hacemos su respectivo bind.
Nos encontramos en esper de que el servido diga que enviará la primera imagen. Eseramos hasta que lo haga.
Creamos un archivo en blanco con el nombre y le agregamos el identificador de la imagen en la que vamos
entramos en un bucle en donde iremos leyendo una a una los datos que nos envia el servidor y cada una la iremos agregando al final del archivo que creamos anteriormente.
Al terminar cerramos y guardamos el archivo.
Cambiaremos a la siguiente imagen, y repetiremos el proceso hasta que tengamos las 5 imágenes, después procederemos a terminar el programa.

EVIDENCIA DE FUNCIONAMIENTO DEL PROGRAMA




Actividades
Terminal
mar 18:05
Carpeta personal
Descargas
prueba
Recientes
Carpeta personal
Escritorio
Descargas
Documentos
1.jpg
2.jpg
3.jpg
4.jpg
5.jpg
cliente.py
angel@angel-Inspiron-N4050: ~/Descargas/prueba
Archivo Editar Ver Buscar Terminal Ayuda
angel@angel-Inspiron-N4050:~/Descargas/prueba$ python3 cliente.py
Descargando imagen 1
Se decargo correctamente la imagen 1
Descargando imagen 2
Se decargo correctamente la imagen 2
Descargando imagen 3
Se decargo correctamente la imagen 3
Descargando imagen 4
Se decargo correctamente la imagen 4
Descargando imagen 5
Se decargo correctamente la imagen 5
angel@angel-Inspiron-N4050:~/Descargas/prueba$
Facebook
Ehecatzin Vallejo
Rafael Jair Nagano
Noe Perez
Dante Doce
Zitzi Gömëz
Randy Juárez
Usiel Arista
Yesenia Yesenia
Bryan Soriano
Manzanita Tovar
Luis Alejandro González
Nady Sabine
CONVERSACIONES EN GRUPO
Usiel, Ehecatzin
Jefe: Colocaste el WHERE
en el DELETE?


Actividades
Terminal
mar 18:04
Usiel envió un mensaje a tu grupo - Mozilla Firefox
https://www.facebook.com
Carpeta personal
Descargas
prueba
cliente
Nombre
Tamaño
Modificación
1.jpg 358.1 kB 18:03
2.jpg 23.8 kB 18:03
3.jpg 69.5 kB 18:03
4.jpg 25.8 kB 18:04
5.jpg 29.4 kB 18:04
cliente.py 721 bytes 17:57
angel@angel-HP-280-G1-ST-Business-PC: ~/Descargas/prueba/servidor
se envio una parte del archivo 5
se envio una parte del archivo 6
se envio una parte del archivo 7
se envio una parte del archivo 8
se envio una parte del archivo 9
se envio una parte del archivo 10
se envio una parte del archivo 11
se envio una parte del archivo 12
se envio una parte del archivo 13
se envio una parte del archivo 14
se envio una parte del archivo 15
se envio una parte del archivo 16
se envio una parte del archivo 17
se envio una parte del archivo 18
se envio una parte del archivo 19
se envio una parte del archivo 20
se envio 1
se envio una parte del archivo 1
se envio una parte del archivo 2
se envio una parte del archivo 3
se envio una parte del archivo 4
se envio una parte del archivo 5
se envio una parte del archivo 6
angel@angel-HP-280-G1-ST-Business-PC: ~/Descargas/prueba/cliente
angel@angel-HP-280-G1-ST-Business-PC:~/Descargas/prueba/cliente$ python3 cliente.py
Descargando imagen 1
Se decargo correctamente la imagen 1
Descargando imagen 2
Se decargo correctamente la imagen 2
Descargando imagen 3
Se decargo correctamente la imagen 3
Descargando imagen 4
Se decargo correctamente la imagen 4
Descargando imagen 5
Se decargo correctamente la imagen 5
angel@angel-HP-280-G1-ST-Business-PC:~/Descargas/prueba/cliente$

CONCLUSIONES:
Victor Usiel Arista Bojorges

El uso de broadcast y multicast en el lado de la aplicación no se ve mucha diferencia con un envío normal de información como lo hemos realizado con anterioridad, solamente se tiene que configurar en los parámetros se se van a enviar de esta manera, de misma instancia se hara para los usuarios que quieran acceder.
Cuando se enviá un archivo de tamaño grande es necesario seccionarlo debido a que las tramas tienen un cierto límite de información que pueden enviar, por lo consecuente, al enviar paquetes de información es importante planear bien como los van a identificar los usuarios para que puedan volver a unirlos de forma correcta.