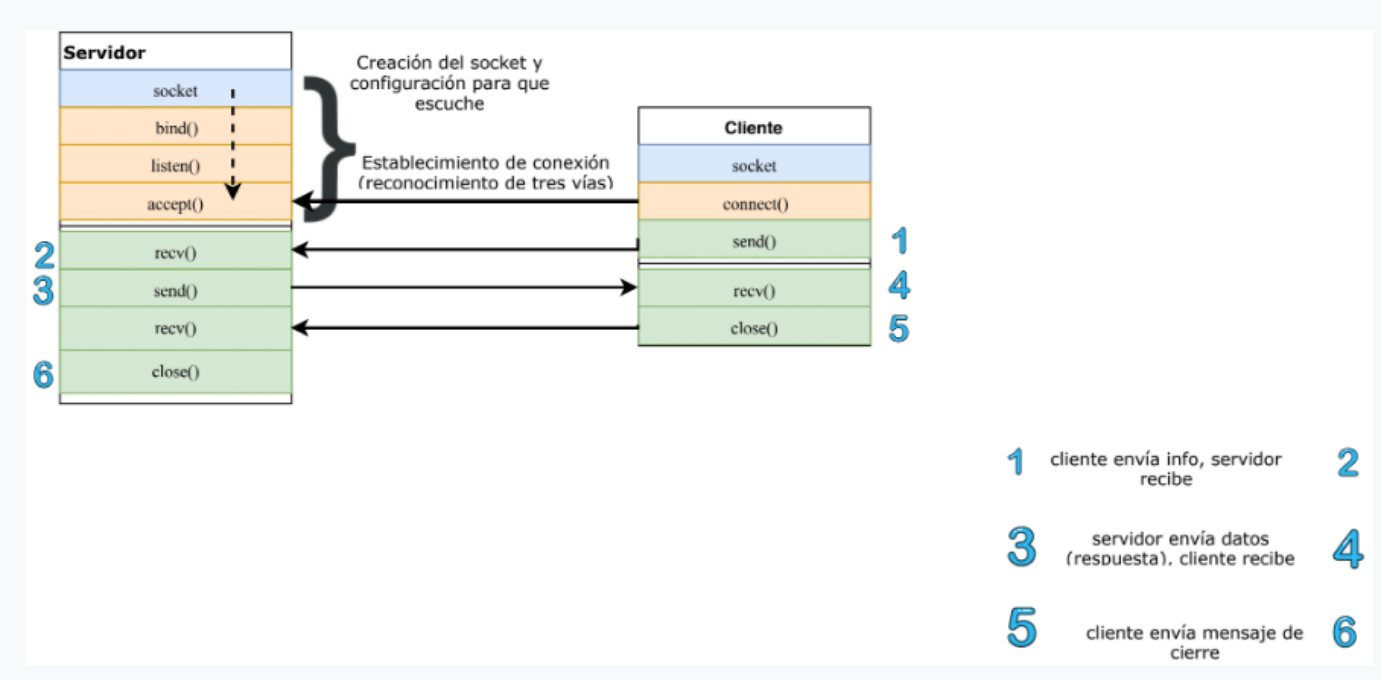
# Cliente y servidor TCP

## Introducción

En el laboratorio anterior creamos un cliente TCP que se conectaba a un servidor remoto. Sin embargo, en ese modelo no teníamos control sobre el funcionamiento/comportamiento del servidor. En este laboratorio vamos a crear un cliente y servidor TCP (desde el mismo dispositivo), aunque bien podría hacerse con dos dispositivos/equipos diferentes, si tiene acceso a ambos. La imagen acontinuación resume la interacción ente cliente y servidor:




## Servidor TCP

Primero vamos a crear el servidor TCP porque algunos detalles necesitan definirse y ser conocidos por el cliente. A continuación se detalla el código:

```
#importamos el modulo socket
import socket
#Definimos las variables a utilizar
destino = '127.0.0.1'
puerto_destino = 10000
bufer = 1024
```

```
# Creamos el socket de tipo TCP

with socket.socket(socket.AF_INET, socket.SOCK_STREAM) as
nuevo_socket:
    #Vinculamos el socket con los parametros que definimos en las
variables
    nuevo_socket.bind((destino, puerto_destino))

        # Establecemos que el servidor esta listo para "escuchar"
    nuevo_socket.listen(5)

        # Utilizamos el metodo accept() para recibir las conexiones
entrantes

    conexion, addr = nuevo_socket.accept()
    with conexion:
        print('[x] La conexion se ha establecido correctamente')
        while True:

            # Lo que recibimos en el bufer (bytes) lo convertimos a
string
            datos = conexion.recv(bufer)

            # Visualizamos los datos que hemos recibido

            if not datos:
                break
            else:
                print('[x] Datos recibidos:
{}'.format(datos.decode('utf-8')))
            conexion.send(datos)
```

Veamos algunos aspectos en la construcción del codigo. Como podran haber notado, el servidor tambien definimos el bufer como en el cliente porque al trabajar en una interaccion cliente-servidor se envia y recibe datos de ambos. El siguiente paso es crear el socket especificando la familia y tipo de socket. Luego debemos vincular la instancia de socket que recien creamos con el socket TCP (direccion IP y puerto) para que este puede "escuchar", como se le conoce. Lo que sigue es aceptar las conexiones entrantes (de los cliente), esto con el metodo *accept()*. Una vez completado esto significa que se ha realizado el reconocimiento de tres vías y puede iniciar el intercambio de informacion. Para ello

utilizamos sentencias de control de flujo *(while)* para controlar si recibimos datos o no y ejecute acciones de acuerdo al caso. Si efectivamente recibimos datos entonces los imprimimos en la pantalla y deberiamos ver lo que envia el cliente *(siguiente seccion)*.

## Cliente TCP

Ahora vamos a visualizar el código que se utilizará como cliente TCP. Este será bastante parecido al que trabajamos en la guía anterior con ligeras diferencias tal como se muestra a continuacion:

```
#importamos el modulo socket
import socket

# El cliente debe tener y conocer las especificaciones del servidor

destino = '127.0.0.1'
puerto_destino = 10000
bufer = 1024

datos_enviar = 'Esta es mi primera aplicacion TCP'

# Creamos el socket de tipo TCP
with socket.socket(socket.AF_INET, socket.SOCK_STREAM) as
nuevo_socket:

    nuevo_socket.connect((destino, puerto_destino))

    # Convertimos los datos a enviar de string a bytes

    nuevo_socket.send(datos_enviar.encode('utf-8'))

    data = nuevo_socket.recv(bufer)
```

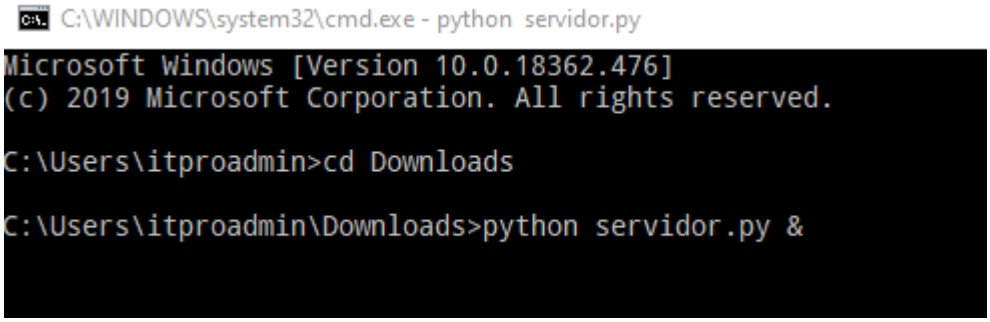
Como pueden observar, la diferencia es que ahora enviamos una cadena de caracteres en lugar de solo conectarnos al servidor y consultar informacion. El envio de la cadena lo hacemos utilizando el metodo *send()*.

## Correr aplicaciones

Cuando hemos editado el codigo podemos proceder a probar su funcionamiento. Debemos correr el servidor en primer lugar (como es de esperarse) y luego el cliente. Ambos scripts debemos correrlos desde la consola de comandos de la siguiente manera:

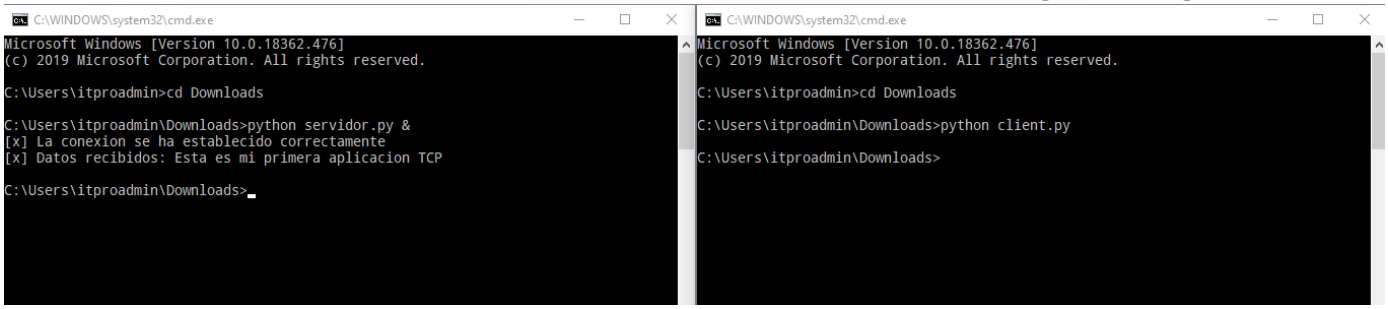
```
python servidor.py &
```

En el caso del servidor, agregamos el `&` para decirle que se ejecute y espere a recibir a una conexión. Así, cuando este se ejecuta podemos observar el curso en la consola:




Cuando el servidor esté corriendo entonces podemos ejecutar el cliente: `python cliente.py`

Finalmente podemos ver el funcionamiento correcto porque la cadena enviada por el cliente se puede visualizar en el servidor, tal como se muestra en la siguiente figura:




Uno de los puntos a destacar es que luego de esta de la primera ejecución de la aplicación si queremos volver a ver su funcionamiento entonces debemos correr los scipts nuevamente.