UNIVERSIDAD DE LAS FUERZAS ARMADAS

"ESPE"

CIENCIAS DE LA COMPUTACIÓN

APLICACIÓN DE SISTEMAS OPERATIVOS

Integrantes:

- Roberson Mauricio Constante Negrete
- Anthony Esteven Quishpe Guaytarilla
- Nayeli Michelle Tipantiza Cumbal

NRC: 4647 **Carrera:** Ingeniería en Tecnologías de la Información

Fecha: 20/06/2022 **Ing.** Andrea López

Actividad 2.2

1. **Consultar cómo se realiza el paso de mensajes en una arquitectura cliente-servidor usando lenguajes de programación como: C, C + +, Python.**
   a) **Consultar cómo se implementan las máquinas servidor cliente. ¿Se recomienda usar Ubuntu o Windows en este caso?**

La arquitectura cliente/servidor ahora se distribuye en aplicaciones cliente/servidor que sirven a miles de clientes. Estas aplicaciones suelen ejecutarse en muchos servidores y constan de cientos de componentes software. Ejecutan las funciones sustanciales de una empresa. Con Internet, pueden hacerse solicitudes a los servidores desde cualquiera de los millones de computadoras personales conectadas a esa red en el mundo.

Consiste básicamente en un cliente que realiza peticiones a otro programa (el servidor) que le da respuesta. La interacción cliente-servidor es el soporte de la mayor parte de la comunicación por redes.

Los clientes y los servidores pueden estar conectados a una red local o una red amplia. Se tiene la libertad de obtener la información que requiera en un momento dado proveniente de una o varias fuentes locales o distantes y de procesarla como según le convenga. Los distintos servidores también pueden intercambiar información dentro de esta arquitectura.

Paso de mensajes en una arquitectura cliente-servidor:

Para establecer una conexión cliente-servidor es necesario conocer qué son los sockets y cuál es su función.

“Una interfaz de programación de socket proporciona las rutinas necesarias para la comunicación entre procesos entre aplicaciones, ya sea en el sistema local o distribuidas en un entorno de red distribuido basado en TCP/IP . Una vez que se establece una conexión de igual a igual, se utiliza un descriptor de socket para identificar de manera única la conexión. El descriptor de socket en sí mismo es un valor numérico específico de la tarea.” (IBM)

Descripción general de la API de socket

Para establecer una conexión cliente servidor se debe definir el lenguaje de programación que se utilizará en el desarrollo, la forma de creación puede variar entre cada lenguaje pero posee los mismos fundamentos. En esta ocasión se utiliza el python por su versatilidad y el conocimiento previo que existe en dicho lenguaje de programación.

El módulo de socket de Python proporciona una interfaz para la API de sockets de Berkeley . Este es el módulo que usaré en este tutorial.

Las funciones y métodos principales de la API de socket en este módulo son:

- socket()
- .bind()
- .listen()
- .accept()
- .connect()
- .connect_ex()
- .send()
- .recv()
- .close()

Python proporciona una API conveniente y consistente que se asigna directamente a las llamadas del sistema, sus contrapartes de C.

Como parte de su biblioteca estándar, Python también tiene clases que facilitan el uso de estas funciones de socket de bajo nivel. Aunque no está cubierto en este tutorial, puede consultar el módulo socket server , un marco para servidores de red. También hay muchos módulos disponibles que implementan protocolos de Internet de alto nivel como HTTP y SMTP.

Forma de funcionamiento de API socket de python

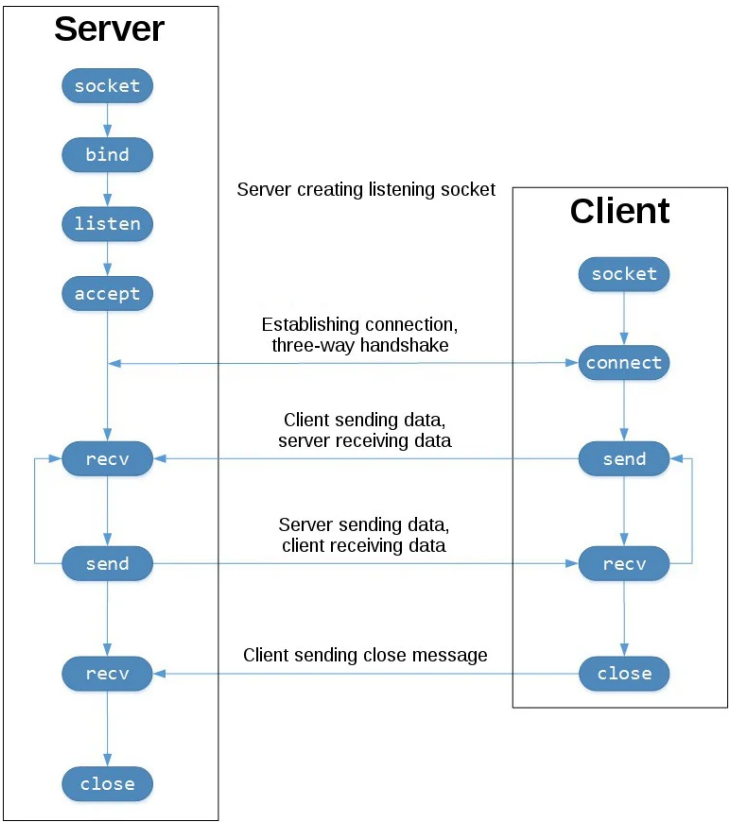

Figura 1. comunicación Cliente- Servidor API socket

Cliente y servidor de eco

Ahora que obtuvo una descripción general de la API de socket y cómo se comunican el cliente y el servidor, está listo para crear su primer cliente y servidor. Comenzará con una implementación simple. El servidor simplemente devolverá al cliente todo lo que reciba.

Servidor de eco

Aquí está el servidor:

```python
# echo-server.py

import socket

HOST = "127.0.0.1"  # Standard loopback interface address (localhost)
PORT = 65432  # Port to listen on (non-privileged ports are > 1023)

with socket.socket(socket.AF_INET, socket.SOCK_STREAM) as s:
    s.bind((HOST, PORT))
    s.listen()
    conn, addr = s.accept()
    with conn:
        print(f"Connected by {addr}")
        while True:
            data = conn.recv(1024)
            if not data:
                break
            conn.sendall(data)
```

Figura 2. Script Servidor

Cliente de eco

Ahora veamos al cliente:

```python
# echo-client.py

import socket

HOST = "127.0.0.1"  # The server's hostname or IP address
PORT = 65432  # The port used by the server

with socket.socket(socket.AF_INET, socket.SOCK_STREAM) as s:
    s.connect((HOST, PORT))
    s.sendall(b"Hello, world")
    data = s.recv(1024)

print(f"Received {data!r}")
```

Figura 3. Script Cliente

Ejecutar el cliente y servidor Echo

Ejecución en el servidor

```shell
$ python echo-server.py
```

Figura 4. Ejecución Script servidor

Ejecución en el cliente

```
Shell
$ python echo-client.py
Received b'Hello, world'
```

En la ventana del servidor, deberías notar algo como esto:

```
Shell
$ python echo-server.py
Connected by ('127.0.0.1', 64623)
```

Figura 5. Ejecución Script Cliente

b) ¿Qué herramientas de gestión pueden ayudar a la implementación del ejercicio?

Para la implementación de este ejercicio nos puede ayudar la herramienta **mRemoteNG**, ya que el programa es compatible con los sistemas operativos Windows, Linux y MacOS y permite realizar tareas de administración remotas en ordenadores y servidores.

La herramienta **OpenSSH** también podría ayudar, debido a que es un programa servidor/cliente SSH más utilizado para servidores y varios dispositivos. El programa es completamente gratuito y de código abierto

Otra herramienta que nos sería útil es **PuTTy**, el cual nos permite conectar con otra máquina, de modo remoto o mediante serial, comúnmente es usado para gestionar un servidor o una máquina remota con Linux mediante SSH.

Bibliografía:

De Luz, S. (13 de mayo de 2021). redeszone. Obtenido de Conece mRemoNG: https://www.redeszone.net/tutoriales/servidores/mremoteng-terminal-avanzado-windows/

De Luz, S. (12 de agosto de 2021). redeszone. Obtenido de Servidor OpenSSH: https://www.redeszone.net/tutoriales/servidores/servidor-openssh-linux-configuracion-maxima-seguridad/

mejorsoftware. (2022). Obtenido de SmarTTY: https://mejorsoftware.info/tools/smartty