1.6 Sockets no orientados a conexión bloqueantes

- Rodriguez Sanchez Daniel
- Garcia Hernandez Manuel Hoscani
- Tapia Delgadillo Diego Alejandro

Introduccion

Nuestro tema es: Sockets no orientados a conexión bloqueante

A lo largo de esta exposición explicaremos:

- ¿Qué es un socket?
- Tipos de socket
- ¿Cómo crear un socket?
- Socket bloqueante
- Socket no bloqueante
- Ejemplo de socket no bloqueante
- Conclusiones

Desarrollo

¿Qué es un socket?

Un socket es el punto final de un enlace de comunicación bidireccional entre dos programas en una red, compuesto por una dirección IP y un número de puerto para identificar lugares específicos en la red. Permiten a dispositivos y aplicaciones diferentes conectarse y compartir datos [1].

Tipos de sockets

Sockets de flujo

Los sockets de flujo (TCP) para comunicaciones seguras y ordenadas

Sockets de datagramas

Los sockets de datagramas (UDP) para transmisiones más rápidas pero sin garantía de entrega

Sockets de dominio

Los sockets de dominio UNIX para comunicación entre procesos en el mismo sistema operativo

```python
import socket

# Definir el host y el puerto en el que el servidor escuchará
HOST = '127.0.0.1'  # Dirección IP del servidor
PORT = 65432        # Puerto en el que el servidor estará escuchando

# Crear un socket TCP/IP
with socket.socket(socket.AF_INET, socket.SOCK_STREAM) as s:
    # Enlazar el socket al host y puerto especificado
    s.bind((HOST, PORT))
    # Escuchar conexiones entrantes
    s.listen()
    print('Servidor escuchando en', (HOST, PORT))
    # Aceptar la conexión entrante
    conn, addr = s.accept()
    with conn:
        print('Conexión establecida desde', addr)
        while True:
            # Recibir datos del cliente
            data = conn.recv(1024)
            if not data:
                break
            print('Mensaje recibido:', data.decode())
            # Devolver los datos recibidos al cliente
            conn.sendall(data)
```

¿Cómo se crea un socket?

```python
import socket

# Definir el host y el puerto del servidor al que se conectará el
cliente
HOST = '127.0.0.1'  # Dirección IP del servidor
PORT = 65432        # Puerto en el que el servidor está escuchando

# Crear un socket TCP/IP
with socket.socket(socket.AF_INET, socket.SOCK_STREAM) as s:
    # Conectar al servidor
    s.connect((HOST, PORT))
    # Enviar datos al servidor
    s.sendall(b'Hola, servidor!')
    # Recibir respuesta del servidor
    data = s.recv(1024)

print('Respuesta del servidor:', data.decode())
```

¿Cómo se crea un socket?

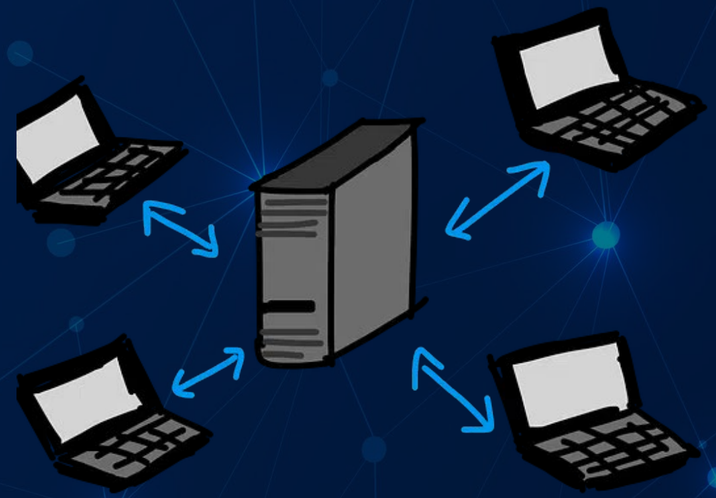
Socket bloqueante

En estos socket las operaciones de lectura y escritura bloquean la ejecución del programa hasta que la operación se complete. Si no hay datos disponibles para leer, la llamada a la función de lectura bloqueará el programa hasta que lleguen los datos. Esto puede hacer que el rendimiento del programa se vea afectado si necesita esperar por operaciones de entrada/salida. [1]

Socket no bloqueante

- Las operaciones de lectura y escritura no bloquean el programa
- El programa puede continuar ejecutándose mientras espera la disponibilidad de datos .
- Si no hay datos disponibles para leer la llamada a la función de lectura retornará inmediatamente con un valor indicando que no hay datos disponibles .

Esto permite que el programa realice otras tareas mientras espera datos, mejorando el rendimiento y la eficiencia en ciertas situaciones .



```python
import socket
import select  # Agrega esta línea

# Definir el host y el puerto en el que el servidor escuchará
HOST = '127.0.0.1'
PORT = 65433

# Crear un socket TCP/IP
with socket.socket(socket.AF_INET, socket.SOCK_STREAM) as s:
    # Configurar el socket como no bloqueante
    s.setblocking(False)
    # Enlazar el socket al host y puerto especificado
    s.bind((HOST, PORT))
    # Escuchar conexiones entrantes
    s.listen()
    print('Servidor escuchando en', (HOST, PORT))

    # Crear lista de sockets para gestionar conexiones
    sockets_list = [s]

    while True:
        # Esperar a que haya conexiones listas para leer, escribir o
errores
        readable, _, _ = select.select(sockets_list, [], [])
```

Ejemplo de socket no bloqueante: Servidor

```
        for sock in readable:
            # Si el socket es el socket de escucha, aceptar una nueva
conexión
            if sock == s:
                conn, addr = s.accept()
                print('Conexión establecida desde', addr)
                conn.setblocking(False)
                sockets_list.append(conn)
            # Si el socket es de una conexión existente, recibir datos
            else:
                data = sock.recv(1024)
                if data:
                    print('Mensaje recibido:', data.decode())
                    # Devolver los datos recibidos al cliente
                    sock.sendall(data)
                else:
                    # Si no hay datos, cerrar la conexión y eliminar el
socket de la lista
                    sock.close()
                    sockets_list.remove(sock)
```

Ejemplo de socket no bloqueante:

```python
import socket

# Definir el host y el puerto del servidor al que se conectará el
cliente
HOST = '127.0.0.1'
PORT = 65433

# Crear un socket TCP/IP
with socket.socket(socket.AF_INET, socket.SOCK_STREAM) as s:
    # Configurar el socket como no bloqueante
    s.setblocking(False)
    # Conectar al servidor
    try:
        s.connect((HOST, PORT))
    except BlockingIOError:
        pass

    # Enviar datos al servidor
    message = b'Hola, servidor!'
    while True:
        try:
            s.sendall(message)
            break
        except OSError:
            pass

    # Esperar a recibir la respuesta del servidor
    data = b""
    while True:
        try:
            part = s.recv(1024)
            if part:
                data += part
            else:
                break
        except OSError:
            pass

print('Respuesta del servidor:', data.decode())
```

Ejemplo de socket no bloqueante: Cliente

Conclusión

Los sockets orientados y no orientados a conexión bloqueante se centra en destacar la importancia y eficiencia que los sockets no bloqueantes ofrecen en el desarrollo de aplicaciones de red. A través de la presentación, se detalla cómo los sockets permiten la comunicación bidireccional entre programas en una red, distinguiendo entre los tipos de sockets (TCP, UDP, UNIX) y su adaptación a diferentes necesidades de comunicación .

Referencias

- Introducción a la programación en red: Sockets y Windows Sockects (1°Edición). (1996). Mguel Angel Quintana Suárez. https://accedacris.ulpgc.es/bitstream/10553/601/1/4631.pdf pp. 44-46, 52, 120-123.
- Ramuglia, G. (2023, Septiembre 5). Python Socket Programming Guide (With Examples). IOFLOOD. https://ioflood.com/blog/python-socket/
- Python Software Foundation. (2024). Cómo trabajar con sockets. Documentación oficial de Python. https://docs.python.org/es/3/howto/sockets.html


¡Gracias!