Escuela Superior de Computo
Aplicaciones para Comunicaciones en Red

Sockets Orientados a conexion bloquenate

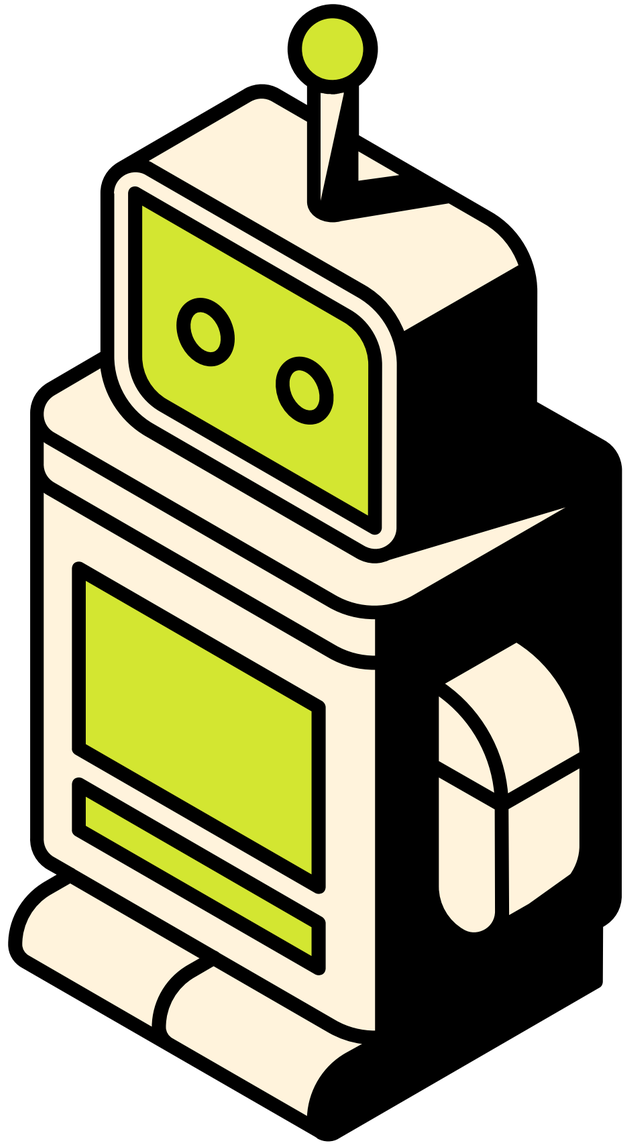
Fecha: Marzo 26 del 2023

Equipo 1.4:

- Morales Torres Alejandro
- Ramírez Contreras Angel Humberto
- Reyes Vivar Fernando

En esta exposicion:

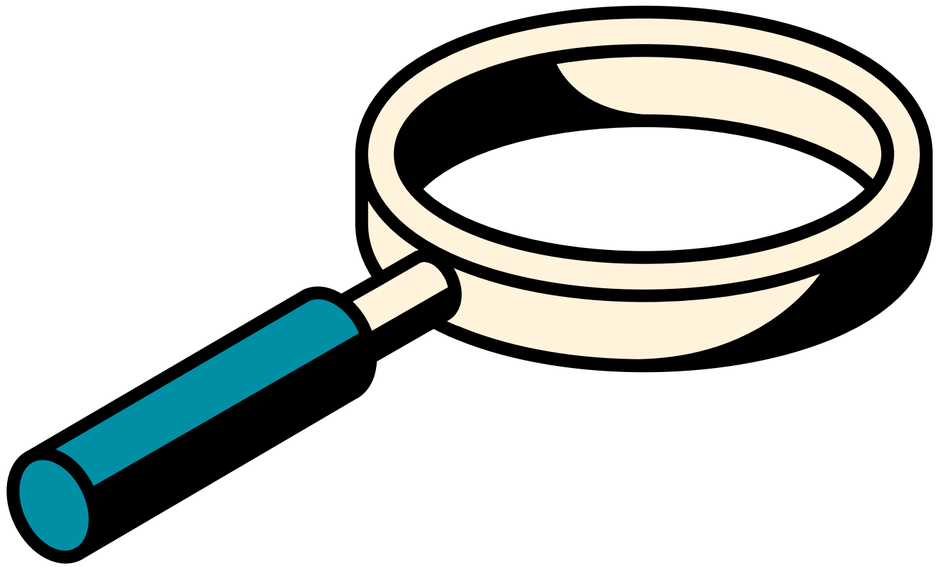
1. Introducción
2. Desarrollo
    a. ¿Qué es un socket?I
    b. Inicializando un socket para cualquier SO
    c. Funciones para sockets
    d. Opciones Client-Side en Java
3. Conclusiones



Introducción

Los sockets juegan un papel crucial como punto final para enviar o recibir datos, y es por ello que hablaremos sobre los sockets orientados a conexión bloqueante y su importancia en las aplicaciones para comunicaciones en red.

Trataremos sobre lo fundamental de los sockets orientados a conexiones no bloqueantes; comenzando con una descripción detallada sobre lo que son y cómo funcionan, hasta cómo se pueden usar en la programación de red y, además, proporcionaremos ejemplos de código en lenguaje C



DES ARROLLO

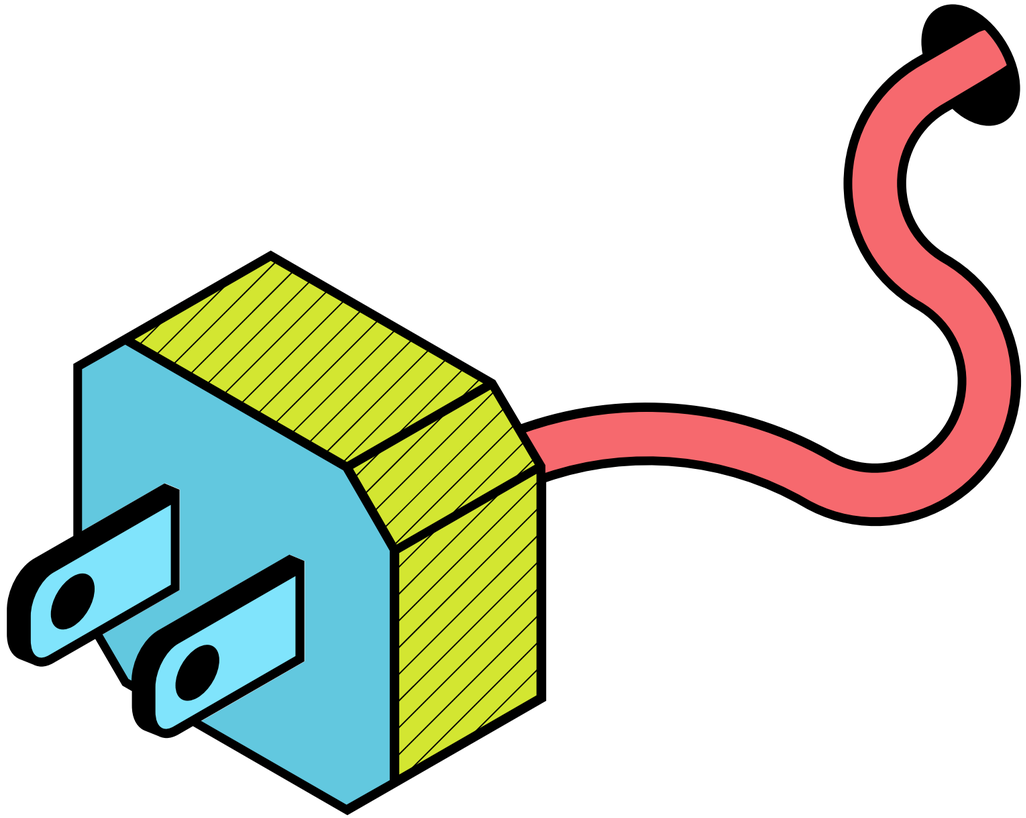
¿Qué son los sockets?

Las aplicaciones en una red consisten en pares de procesos que se comunican mediante el intercambio de mensajes, los cuales transitan a través de la red. Esta comunicación se realiza a través de una interfaz de software conocida como socket

¿QUÉ SON LOS SOCKETS?

Los sockets son del tipo bloqueantes por defecto y no son más que un endpoint de enlace de comunicación de dos vías entre sistemas que se ejecutan a través de red que nos permite:

1. Conectarse remotamente a una maquina
2. Enviar datos
3. .Recibir datos
4. Cerrar o terminar una conexión
5. Enlazar un puerto
6. Escuchar datos entrantes
7. Aceptar conexiones de maquina remotas en el puerto enlazado



DES ARROLLO

Como inicializar un socket

```c
#if defined(_WIN32)
    #ifndef _WIN32_WINNT
        #define _WIN32_WINNT 0x0600
    #endif
    #include <winsock2.h>
    #include <ws2tcpip.h>
    #pragma comment(lib, "ws2_32.lib")                                  // esta linea no es necesatia
#else
    #include <sys/types.h>
    #include <sys/socket.h>
    #include <netinet/in.h>
    #include <arpa/inet.h>
    #include <netdb.h>
    #include <unistd.h>
    #include <errno.h>
#endif

#include <stdio.h>

int main() {
    #if defined(_WIN32)
        WSADATA d;
        if (WSAStartup(MAKEWORD(2, 2), &d)) {
            fprintf(stderr, "Error al inicializar.\n");
            return 1;
         }
    #endif

    printf("Listo para usar sockets!\n");

    #if defined(_WIN32)
        WSACleanup();
     #endif
    return 0;
    }
```

Inicializando un socket en C

DES ARROLLO

Funciones para sockets

FUNCIONES BÁSICAS

- **socket ( )** crea e inicializa un socket.
- **bind ( )** asocia un socket con una IP particular a un numero de puerto
- **listen ( )** hace que el servidor escuche llamadas para nuevas conexiones
- **connect( )** se usa para establecer una dirección remota y un numero puerto a un cliente TCP.
- send ( ) y recv ( ) se usan para enviar y recibir datos con un socket.
- **sendto ( )** y **recvfrom ( )** son funciones usadas para enviar y recibir datos desde sockets sin una direccion remota vinculada.
- **close ( )** o **closesocket ( )** se usan para cerrar o terminar la conexión de un socket.
- **shutdown ( )** se usa para terminar la conexión TCP de solo un lado y es útil para cerrar las conexiones de forma ordenada
- **getnameinfo ( )** nos proporciona una forma de trabajar con los nombres de host y direcciones
- **setsocketopt ( )** se usa para cambiar alguna de las opciones de los sockets

DES ARROLLO

Opciones Client-Side en Java

TCP_NODELAY

```
public void setTcpNoDelay(boolean on) throws SocketException
public boolean getTcpNoDelay() throws SocketException
```

SO_TIMEOUT

```
public void setSoTimeout(int milliseconds) throws SocketException
public int getSoTimeout() throws SocketException
```

SO_LINGER

```
public void setSoLinger(boolean on, int seconds) throws SocketException
public int getSoLinger() throws SocketException
```

SO_REUSEADDR

```
public void setReuseAddress(boolean on) throws SocketException
public boolean getReuseAddress() throws SocketException
```

SO_SNDBUF

```
public void setReceiveBufferSize(int size)
    throws SocketException, IllegalArgumentException
public int getReceiveBufferSize() throws SocketException
public void setSendBufferSize(int size)
    throws SocketException, IllegalArgumentException
public int getSendBufferSize() throws SocketException
```

OOBINLINE

```
public void setOOBInline(boolean on) throws SocketException
public boolean getOOBInline() throws SocketException
```

```
if (!s.getOOBInline()) s.setOOBInline(true);
```

IP_TOS

```
public int getTrafficClass() throws SocketException
public void setTrafficClass(int trafficClass) throws SocketException
```

SO_KEEPALIVE

```
public void setKeepAlive(boolean on) throws SocketException
public boolean getKeepAlive() throws SocketException
```

```
if (s.getKeepAlive()) s.setKeepAlive(false);
```

CONCLUSIONES

Los sockets orientados a la comunicación no bloqueante son esenciales para desarrollar aplicaciones de red eficientes y receptivas. Al permitir que una aplicación realice otras tareas mientras espera la disponibilidad de datos en la red, mejoran considerablemente la escalabilidad y el rendimiento.

Esta característica es particularmente útil en entornos de alta concurrencia o en aplicaciones que necesitan gestionar múltiples conexiones simultáneamente, como servidores web, juegos en línea y sistemas de chat en tiempo real.

Gracias