Mecanismos de Comunicación entre procesos Sockets TCP

Roberto Romero, Juan Malo

Ciencias de la Computación
Universidad Politécnica Salesiana

***Resumen*—La comunicación entre procesos es un componente esencial en el desarrollo de sistemas distribuidos y aplicaciones modernas. Este informe se centra en los Sockets como un mecanismo clave para facilitar la interacción entre procesos, ya sea en una misma máquina o a través de redes. Se analiza cómo los Sockets permiten el intercambio de datos mediante protocolos como TCP y UDP, destacando su rol en arquitecturas cliente-servidor. Además, se presentan ejemplos prácticos de su implementación, junto con un análisis de sus ventajas, limitaciones y casos de uso. Este estudio resalta la relevancia de los Sockets en la sincronización y colaboración de procesos en entornos concurrentes y distribuidos.**

***Abstract*--Inter-process communication (IPC) is a critical component in the development of distributed systems and modern applications. This report focuses on Sockets as a key mechanism to enable process interaction, whether on the same machine or across networks. It examines how Sockets facilitate data exchange through protocols like TCP and UDP, emphasizing their role in client-server architectures. Practical implementation examples are presented alongside an analysis of their advantages, limitations, and use cases. This study highlights the importance of Sockets in synchronizing and coordinating processes in concurrent and distributed environments.**

I. Introducción

Un componente esencial de los sistemas operativos y las aplicaciones distribuidas es la comunicación entre procesos (IPC, como se conoce en inglés), que permite el intercambio de datos y la coordinación entre procesos independientes. Unos mecanismos de comunicación eficaces y fiables son esenciales en los entornos actuales, donde las aplicaciones suelen diseñarse para funcionar en sistemas distribuidos o en varios núcleos. Los Sockets, introducidos como parte de la interfaz BSD Unix, han evolucionado hasta convertirse en una herramienta clave para implementar la comunicación entre procesos, tanto en sistemas locales como en redes distribuidas. Este mecanismo permite que los procesos se comuniquen mediante protocolos estándar como TCP, que garantiza una conexión fiable, o UDP, que ofrece una comunicación más rápida y liviana. Gracias a su flexibilidad y soporte nativo en la mayoría de los sistemas operativos, los Sockets son ampliamente utilizados en aplicaciones como mensajería, transferencia de datos y servicios cliente-servidor.

Universidad Politécnica Salesiana

II. Marco teorico

Los Sockets son mecanismos de comunicación mecanismos de comunicación entre procesos que permiten que un proceso hable entre procesos que permiten que un proceso hable
(emita o reciba información con otro proceso incluso estando en distintas máquinas.
Generalmente el flujo de datos se hace de forma fiable y ordenada, no obstante, dependemos de si los procesos están utilizando el protocolo TCP o el protocolo UDP.

- El protocolo TCP es un protocolo de la capa de transporte que es orientado a conexión, esto significa que antes de intercambiar los datos reales hay un paso previo para establecer una comunicación. Este protocolo también garantiza que toda la transmisión de los datos se hace sin errores, el propio TCP se encarga de reenviar los datos nuevamente en caso de que el receptor no los reciba a tiempo o los reciba dañados, además, garantiza también el orden, por lo que nos aseguramos de que los procesos van a recibir todos los datos en orden desde su origen.
- En el caso del protocolo UDP, no es orientado a conexión, no hay un paso previo en la comunicación sino que se envían los datos directamente. Este protocolo no garantiza que la transmisión se realice sin errores, aunque hará todo lo posible para que sí lo haga, además, tampoco garantiza el orden de los datagramas que el origen envíe al destino. La parte positiva de UDP es que tiene una cabecera muy pequeña y es muy rápido, ya que no hay una fase de establecimiento de la conexión.

III. Componentes

Sockets TCP
Los Sockets basados en el protocolo TCP (Transmission Control Protocol) son orientados a conexión, lo que significa que establecen una conexión antes de transmitir datos. Sus principales componentes son:
Dirección IP y Puerto: identifican de manera única un socket en una red.

- **Dirección IP:** Es la identificación del dispositivo (IPv4 o IPv6).
- **Puerto:** Es el número asociado al servicio o aplicación (por ejemplo, 80 para HTTP).

Establecimiento de Conexión: Handshake de Tres Pasos (Three-Way Handshake), este proceso garantiza que ambas partes estén listas para la comunicación.

- **SYN:** El cliente envía una solicitud para iniciar la conexión.
- **SYN-ACK:** El servidor confirma la solicitud.
- **ACK:** El cliente confirma la respuesta del servidor.

Flujo de Datos Orientado a Conexión: Los datos se envían como un flujo continuo, divididos en segmentos. TCP garantiza:

- Entrega fiable.
- Orden correcto de los datos.

Control de Congestión y Retransmisión: TCP implementa mecanismos para evitar la congestión de la red y retransmite los paquetes perdidos.

Cierre de Conexión: Utiliza un proceso de cuatro pasos para cerrar la conexión de manera segura: FIN, ACK, FIN, ACK.

Sockets UDP

Los Sockets basados en el protocolo UDP (User Datagram Protocol) son sin conexión, lo que significa que no requieren un establecimiento previo de conexión. Sus componentes son:

Dirección IP y Puerto: Similar a TCP, se utilizan para identificar el socket en la red.

- UDP utiliza la misma combinación de IP y puerto, pero sin la necesidad de un handshake.

Datagramas: los datos se envían en bloques independientes llamados datagramas, que contienen:

- **Encabezado (Header):** Incluye información como el puerto de origen, puerto de destino, longitud y suma de verificación (checksum).
- **Cuerpo (Payload):** Los datos que se transmiten.

Simplicidad y Velocidad: UDP no garantiza:

- La entrega de los datos.
- El orden de llegada.
- La retransmisión en caso de pérdida.

Esto lo hace más rápido y eficiente para aplicaciones que pueden tolerar la pérdida de datos (como streaming o juegos en línea).

Multiplexación y Demultiplexación: Los sockets UDP permiten que múltiples aplicaciones compartan la misma dirección IP mediante el uso de números de puerto para diferenciar los datagramas.

IV. Comparaciòn sockets TCP y UDP

Componente	TCP	UDP
Conexión	Orientado a conexión	Sin conexión
Garantía de entrega	Sí, mediante retransmisión	No
Orden de los datos	Garantizado	No garantizado
Velocidad	Más lento por el control adicional	Más rápido por su simplicidad
Aplicaciones típicas	HTTP, FTP, SSH	DNS, VoIP, streaming

V. Ventajas

- Flexibilidad para múltiples protocolos de transporte.
- Compatibilidad en diversos sistemas operativos.
- Soporte para comunicación local y remota.

VI. Desventajas

- Requiere manejo explícito de errores y conexiones.
- Complejidad en aplicaciones concurrentes.
- Mayor sobrecarga comparado con mecanismos específicos para IPC en sistemas locales (como pipes o memoria compartida).

VII. Conclusiones

Los Sockets son una herramienta esencial para la comunicación entre procesos, especialmente en sistemas distribuidos. Su flexibilidad y capacidad para manejar múltiples protocolos los hacen indispensables en aplicaciones modernas. Sin embargo, su uso eficiente requiere experiencia en la gestión de conexiones y manejo de errores. Este informe destaca su importancia y propone que futuros trabajos analicen técnicas para simplificar su implementación en entornos altamente concurrentes.