COMUNICACIÓN RPC

GÓMEZ EDGAR, Universidad Central de Venezuela, Venezuela

La comunicación mediante Llamada a Procedimiento Remoto (RPC) es un paradigma en sistemas distribuidos que permite a un programa ejecutar procedimientos en espacios de direcciones diferentes, típicamente en máquinas remotas, de manera transparente. Este documento analiza los principios de RPC , enfocándose en sus dos variantes principales: síncrona y asíncrona. Se examinan los componentes esenciales, mecanismos de implementación, ventajas y limitaciones de cada enfoque.

CCS Concepts: • **Organización de sistemas informáticos → Sistemas Distribuidos**; *Procedimientos en Programación*; *Tareas síncronas*; Tareas asíncronas.

Additional Key Words and Phrases: Organización de sistemas informáticos, síncorina, asíncornia

ACM Reference Format:
GÓMEZ EDGAR. 2025. COMUNICACIÓN RPC. *Proc. ACM Hum.-Comput. Interact.* 1, 1, Article 1 (December 2025), 3 pages. https://doi.org/10.1145/nnnnnnn.nnnnnnn

1. Introducción

La comunicación RPC (Remote Procedure Call) permite la ejecución de procedimientos en equipos remotos (diferentes al que hace el llamdao del procedimiento) como si se tratase de la llamada de un procedimiento local, el cliente nunca se entera que esta tratando de acceder a un recurso remoto, de esta manera el proceso es transparente para el usuario final. Se oculta la complejidad de la comunicacion generalmente a traves de la red permitiendo al sistema trabajar como si fuera un monolito (centralizado).

2. Comunicación RPC

El RPC como un mecanismo de comunicación entre procesos que permite a un programa (cliente) solicitar la ejecución de un procedimiento en otro programa (servidor) ubicado potencialmente en una máquina diferente. El modelo RPC consta de varios componentes esenciales:

1. **Cliente:** Aplicación que realiza la llamada al procedimiento remoto.
2. **Cliente stub:** Representación local del procedimiento remoto que realiza el empaquetado de parámetros (marshalling) y la comunicación con el servidor.
3. **Servidor stub:** Componente que recibe las solicitudes, desempaqueta los parámetros (unmarshalling) y llama al procedimiento real.
4. **Servidor:** Aplicación que contiene la implementación del procedimiento remoto.

El proceso de RPC sigue un flujo estandarizado: (1) el cliente llama al stub local, (2) el stub cliente empaqueta los parámetros (marshalling) y los envía al servidor, (3) el stub servidor desempaqueta los parámetros y llama al procedimiento real, (4) el resultado sigue el camino inverso. Esta arquitectura proporciona transparencia de acceso a recursos remotos.

3. RPC síncrono

En RPC síncrono, el cliente que realiza la llamada se bloquea hasta que recibe una respuesta del servidor. Este modelo replica más fielmente la semántica de las llamadas a procedimiento locales, donde el llamante espera necesariamente el resultado antes de continuar. Donde el flujo de control es lineas, el cliente espera

Author's Contact Information: GÓMEZ EDGAR, Universidad Central de Venezuela, Caracas, Venezuela.

2025. ACM 2573-0142/2025/12-ART1
https://doi.org/10.1145/nnnnnnn.nnnnnnn

automáticamente la terminación del proceso remoto, similar al paradigma de programación secuencial, un ejemplo se muestra en la Fig 1.

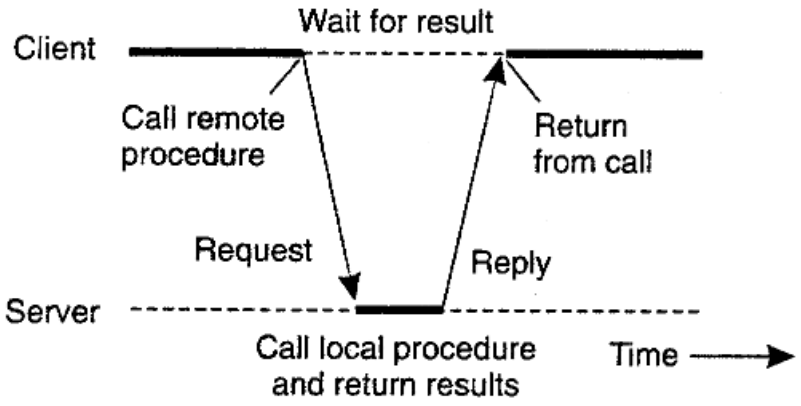

Fig. 1. RPC síncrono. (Adaptado de [1])

Algunas de las deventajas o limitaciones del modelo sincrono: El bloqueo del cliente durante la llamada puede llevar a una utilización ineficiente de recursos, especialmente cuando las latencias de red son altas. Además, los fallos de red o del servidor pueden resultar en bloqueos indefinidos, requiriendo mecanismos adicionales como timeouts y manejo de excepciones.

4. RPC asíncrono

El RPC asíncrono, permite que el cliente continúe su ejecución inmediatamente después de realizar la llamada, sin esperar la respuesta del servidor. Como se muestra en la Fig 2.

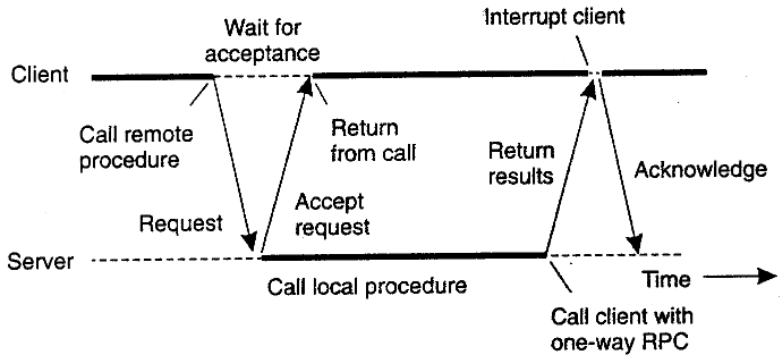

Fig. 2. RPC asíncrono. (Adaptado de [1])

Las ventajas del RPC asíncrono incluyen mejor utilización de recursos, mayor concurrencia y mejor capacidad de respuesta de las aplicaciones cliente, este modelo introduce complejidad en la programación, especialmente en el manejo de la sincronización y el orden de las respuestas.

5. Comparación

Supongamos que un cliente realiza una llamada RPC asíncrona a un servidor y espera a que este devuelva un resultado mediante otra llamada RPC asíncrona. ¿Es este enfoque lo mismo que permitir que el cliente ejecute una llamada RPC normal? ¿Qué sucedería si sustituyéramos las llamadas RPC asíncronas por síncronas?

No, no es lo mismo, una llamada de RPC asíncrona retorna un verificación de que se ha aceptado la ejecución del procedimiento remoto al cliente, no devuelve el resultado del procedimiento como en una llamada síncrona, el resultado del procidiento en caso que lo tenga se devolvera en el futuro interrumpiendo al cliente. Si cambiamos los RCP sincronos por asíncornos volvemos al bloqueo del cliente esperando por la respuesta del servidor.

6. Conclusión

La comunicación RPC es esencial en sistemas distribuidos, ofreciendo dos enfoques principales: síncrono y asíncrono. El síncrono bloquea al cliente hasta recibir respuesta, siendo simple pero ineficiente con alta latencia. El asíncrono permite al cliente continuar trabajando mientras espera, mejorando la concurrencia aunque con mayor complejidad.

El paradigma RPC añade una capa de transparencia a los sistemas distribuidos y es excelente para usarlo en la configuración cliente-servidor.

Referencias

[1] Tanenbaum, A. S. and Van Steen, M. *Distributed Systems: Principles and Paradigms (2nd Edition).* Pearson Prentice Hall, Upper Saddle River, NJ 07458, 2007.