Comparison of IPC (Inter-Process Communication) and RPC (Remote Procedure Call) in Linux

1. Overview

Feature	IPC (Inter-Process Communication)	RPC (Remote Procedure Call)
Definition	Mechanism for communication between processes on the same machine.	Allows a program to invoke a procedure on a remote machine as if it were local.
Communication Scope	Local (within the same machine)	Distributed (across different machines)
Performance	High speed (shared memory, pipes, etc.)	Slower due to network overhead
Complexity	Lower, as it involves local OS mechanisms	Higher, due to network issues like latency, failures, and serialization
Security	Processes run under the same OS security model	Needs authentication, encryption, and authorization
Reliability	More reliable, less chance of data loss	Network failures can cause communication issues
Use Cases	Shared memory, message passing, file sharing, signaling	Distributed computing, microservices, remote APIs

---

2. IPC (Inter-Process Communication)

IPC allows processes running on the same system to exchange data.

✅ Common IPC Mechanisms in Linux

IPC Mechanism	Description	Pros	Cons
Pipes (Named & Anonymous)	Stream-based communication between related processes.	Simple, efficient for parent-child processes.	Only works between related processes.
Message Queues	Message passing through queues managed by the OS.	Persistent, allows structured messages.	More overhead than shared memory.
Shared Memory	Memory-mapped region accessible by multiple processes.	Fastest IPC method, avoids copying.	Requires synchronization (e.g., semaphores).
Semaphores	Used for process synchronization and mutual exclusion.	Prevents race conditions.	Can lead to deadlocks if not managed correctly.
Sockets (Local UNIX Sockets)	Communication via socket files on the filesystem.	Can be used by unrelated processes.	More overhead than shared memory.

✅ **Example Use Case:**

- **Multithreading applications** use shared memory and semaphores.
- **Client-server models within the same machine** use UNIX domain sockets.

---

3. RPC (Remote Procedure Call)

RPC enables communication between processes on **different machines** by allowing one program to execute a function on a remote machine.

✅ **Common RPC Mechanisms in Linux**

RPC Mechanism	Description	Pros	Cons
gRPC (Google RPC)	High-performance RPC using HTTP/2 and Protobuf.	Fast, language-agnostic, streaming support.	Requires Protobuf serialization.
XML-RPC	Uses XML for encoding messages over HTTP.	Simple and human-readable.	Slower due to XML parsing overhead.
JSON-RPC	Uses JSON over HTTP or TCP.	Lightweight, widely supported.	Less efficient than Protobuf.
CORBA (Common Object Request Broker Architecture)	Object-oriented RPC system.	Language-neutral, supports complex data.	Heavy and outdated compared to gRPC.

✅ **Example Use Case:**

- **Microservices** communicate over gRPC.
- **Distributed databases** use RPC to sync data between nodes.