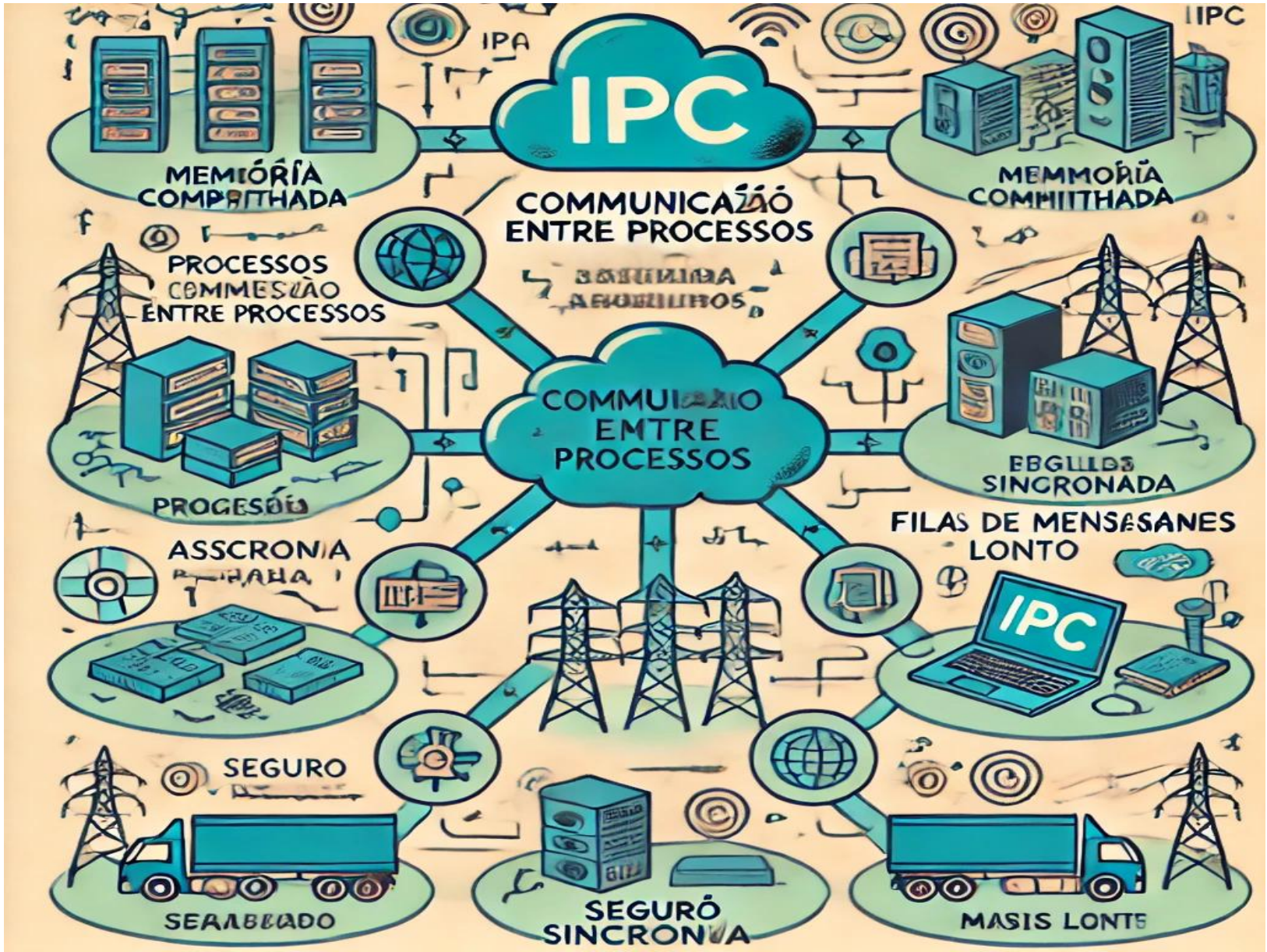
IPC

PIPES


IPC
IPA
IPC
MEMÓRIA
COMPARTILHADA
MEMÓRIA
COMPARTILHADA
COMUNICAÇÃO
ENTRE PROCESSOS
PROCESSOS
COMUNICAÇÃO
ENTRE PROCESSOS
COMUNICAÇÃO
ENTRE
PROCESSOS
SINCRONIZADA
FILAS DE MENSAGENS
IPC
ASSÍNCRONA
SEGURO
SEGURO
SINCRONIA
MAIS LENTO

- **1. Compartilhamento de Dados: Permite que processos troquem informações, como em bancos de dados, sistemas de arquivos distribuídos e aplicações colaborativas.**
- **2. Modularização de Sistemas: Facilita o desenvolvimento de sistemas grandes e complexos dividindo-os em múltiplos processos menores e independentes, tornando o código mais organizado e escalável.**
- **3. Execução Concorrente: Permite que múltiplos processos executem simultaneamente, otimizando o uso de recursos do sistema e melhorando a performance de aplicações.**
- **4. Comunicação entre Aplicações Distribuídas: Usado em microserviços, sistemas em nuvem e arquiteturas cliente-servidor para comunicação entre diferentes máquinas, permitindo a construção de sistemas escaláveis.**
- **5. Sincronização de Processos: Processos que dependem uns dos outros precisam se coordenar. Por exemplo, semáforos e mutex ajudam a evitar condições de corrida e deadlocks.**
- **6. Balanceamento de Carga e Distribuição de Tarefas: Permite que sistemas distribuam tarefas entre múltiplos processos ou servidores, melhorando o desempenho e evitando sobrecarga em um único ponto.**
- **7. Comunicação entre Componentes do SO: Sistemas operacionais usam IPC para interação entre seus componentes, como drivers de dispositivos, chamadas de sistema e gerenciamento de processos.**

O que são Pipes?

- Mecanismo de comunicação entre processos (*Inter-Process Communication - IPC*).
- Permite a transferência de dados de um processo para outro.
- Baseia-se no conceito de fluxo de dados unidirecional ou bidirecional.

Principais tipos:

- **Pipes anônimos** – Comunicação entre processos relacionados (pai-filho).
- **Pipes nomeados (FIFO)** – Permitem comunicação entre processos não relacionados.

Pipes Anônimos

Características:

- Criados com a chamada `pipe()` no Linux/Unix.
- Unidirecionais (leitura ou escrita).
- Apenas entre processos relacionados.
- Maior flexibilidade na comunicação entre processos.
- Persistem na memória e são removidos ao fim de execução.

Pipes Nomeados (FIFO)

Diferenças dos pipes anônimos:

- Criados com `mkfifo()`.
- Permitem comunicação entre processos independentes.
- Possuem um nome no sistema de arquivos.
- O dado precisa ser consumido em ordem FIFO (*First In, First Out*).
- Persiste na memória secundária mesmo após o fim da execução dos processos.

Comparação

Característica	Pipes Anônimos	Pipes Nomeados (FIFO)
Comunicação	Pai-filho	Qualquer processo
Direcionalidade	Unidirecional	Unidirecional
Persistência	Temporário	Permanente
Criação	`pipe()`	`mkfifo()`

Resumo:

- **Pipes anônimos** são mais simples, mas limitados.
- **Pipes nomeados** oferecem mais flexibilidade e podem ser usados entre processos sem parentesco.
- Ambos são fundamentais para comunicação eficiente entre processos.

O que são Sockets

- Tipo de **IPC** (Inter-Process Communication).
- Permite comunicação entre **processos locais e remotos**.
- Funciona como uma **ligação telefônica**, mas usando **IP e porta**.

Introdução aos Sockets

- **Dispositivo de comunicação bidirecional**.
- Possível **enviar e receber mensagens**.
- Dois principais protocolos: **TCP e UDP**.

TCP

- **TCP**
- **Protocolo orientado à conexão**.
- **Garante a entrega e a ordem correta** dos dados.
- **Indicado para**:
  - Transferência de arquivos.
  - Páginas web.
  - Imagens estáticas.

UDP

- **Protocolo sem conexão**.
- **Sem verificação de erros** ou garantia de entrega.
- **Indicado para**:
  - Streaming de vídeo.
  - Jogos online.
  - Transmissões em tempo real.

O que é RMI?

- **RMI (Remote Method Invocation)** permite que um programa em Java chame métodos em objetos remotos.
- Baseado na arquitetura cliente-servidor.
- Utiliza **serialização de objetos** para enviar e receber dados.

Componentes principais:

- **Servidor** – Define e disponibiliza métodos remotos.
- **Cliente** – Chama métodos remotos no servidor.
- **Registry** – Serviço que registra e localiza objetos remotos.

Fluxo de comunicação:

- O servidor registra o objeto remoto no **RMI Registry**.
- O cliente busca a referência do objeto remoto.
- O cliente chama o método remoto, que é executado no servidor.
- O resultado é enviado de volta ao client

Transparência: O cliente chama o método como se fosse local, mas ele executa remotamente.

Prós e Contras do RMI

Característica	Vantagens	Desvantagens
Simplicidade	Fácil de usar em Java	Depende do Java
Serialização	Objetos são passados diretamente	Pode ser lenta para grandes dados
Comunicação	Transparente para o usuário	Necessário configurar rede/firewall
Integração	Usa **Java Security Manager** para segurança	Não compatível com outras linguagens

Resumo:

- **RMI é útil para comunicação distribuída em Java**.
- **Fácil de usar** para quem já trabalha com Java.
- **Alternativas mais modernas** incluem WebSockets, REST APIs e gRPC.

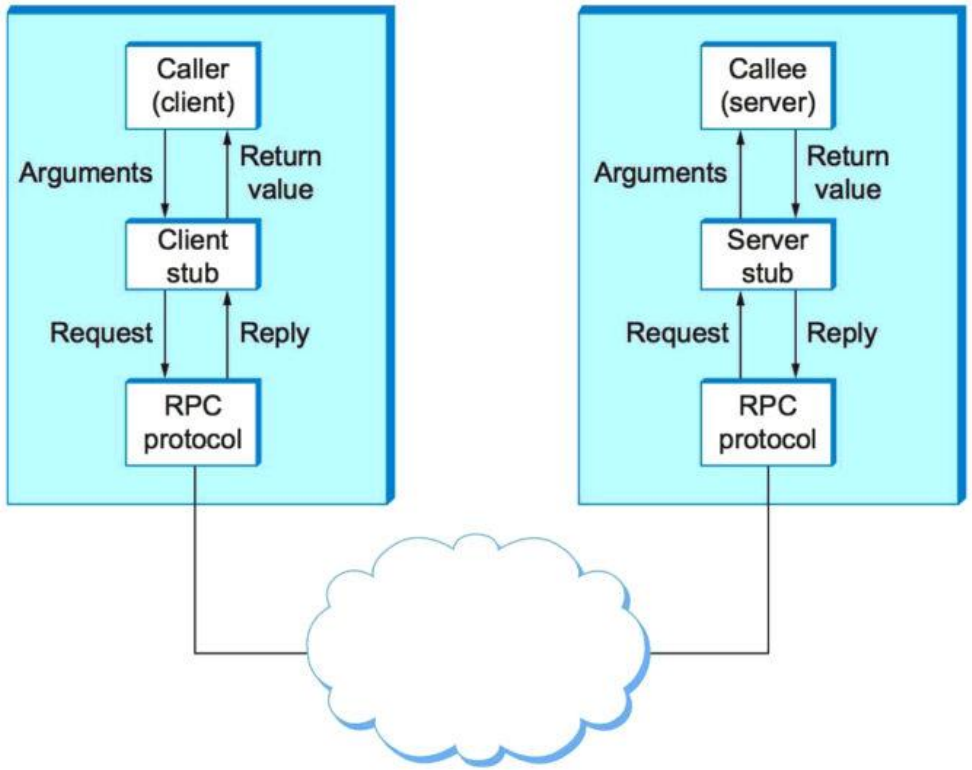
O QUE É O RPC?

REMOTE PROCEDURE CALL ou chamada de procedimento remoto, é um tipo de IPC que permite que um sistema chame uma função ou procedimento em outro sistema como se estivesse no local, mesmo que estejam em locais remotos. Abordagem bastante útil em **sistemas distribuídos**, já que diferentes partes do sistema precisam trocar informações de forma rápida e eficaz.

- **Como funciona o RPC?**

Funciona da seguinte forma: vamos imaginar dois computadores conectados por uma rede. Um computador faz uma solicitação de uma função(procedimento) o outro computador responde, executa essa função e devolve o resultado. Com isso temos dois componentes principais: O **Cliente RPC e o Servidor RPC.** O cliente RPC é a máquina que faz a solicitação e o servidor RPC processa essa solicitação e envia a resposta. Quando o cliente faz a chamada, o **RPC protocol** traduz essa solicitação para um formato que o servidor entenda, assim o servidor executa o procedimento solicitado e envia a resposta de volta para o cliente.

- **Arquitetura RPC**



Remote Procedure Call (RPC)