# Arquitecturas Distribuidas Simples

## Cliente Servidor

- Roles:
  - **Servidor:** pasivo, provee servicios.
  - **Cliente:** activo, envía pedidos.
- **Centralización** en toma de decisiones.
- Servidor tiene ubicación conocida.
- Modelos de *callback* posibles: long polling, push notifications.

## Peer to Peer

- **Red de nodos pares** entre sí.
- Objetivos de **colaboración**.
  - **Protocolo acordado** entre partes.
  - Lógica distribuida requiere **coherencia entre nodos**.
- Tracker (esquema mixto C-S) como servicio de nombres.

## RPC

- **Ejecución remota** de procedimientos.
- Modelo C-S: servidor ejecuta el procedimiento y devuelve resultado.
- Comunicación transparente.
- **Portabilidad** mediante **interfaces**.

### IDL

- Diferentes lenguajes se invocan entre sí.
- **Interfaz s/ input y output.**
- **Definición de tipos de mensajes a enviar** como parte de IDL.

#### Tolerancia a Fallos

- Puede o no ser ejecutado.
- **Garantizar delivery** con estrategias:
  - **REQ-REP** con timeout.
  - Filtrado de duplicados.
  - **Re-transmisión / Re-ejecución** de operación.

#### Implementación

`Cliente <-> Stub <-> Communications Module (Client-Side) <-> Communications Module (Server-Side) <-> Stub <-> Server`

- **Cliente.** Conectado a stub p/ realizar llamadas al servidor.
- **Servidor.** Conectado a stub p/ recibir parámetros.
  - Posee lógica particular del remote procedure.
- **Stubs.**
  - Administra el marshalling de la información.
  - Envía info de las **calls** al módulo de comunicación y al C-S.
- **Módulo de comunicación.** Abstrae al stub de la comunicación con el server.

## Distributed Objects

- Servidores proveen **objetos**.

- **Middleware** p/ ocultar complejidad.
  - Referencias a objetos.
  - Invocaciones de acciones.
  - Errores.
  - Recolección de basura.

**CORBA**

- Estandar definido.
- Protocolo y serialización.
- Transporte.
- Seguridad.
- Discovery de Objetos.

**RMI**

- Lo mismo, optimizado en Java.
- **Registry:** directorio de servicios.