

# Sistema distribuido

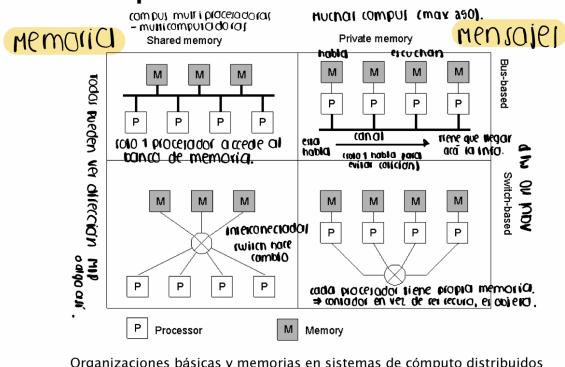
## Sistemas de información

B2B Business 2 Business  
cliente - servidor

- Sistemas Distribuidos sistemas
- formado por conjunto de programas que se ejecutan en Procesos ≠
- Contribuyen a tarea en común
- $\Rightarrow$  o ≠ máquinas conocidas a través de red de comunicación (ilusión de sistema único y homogénea)

conectar memorias con CPU.

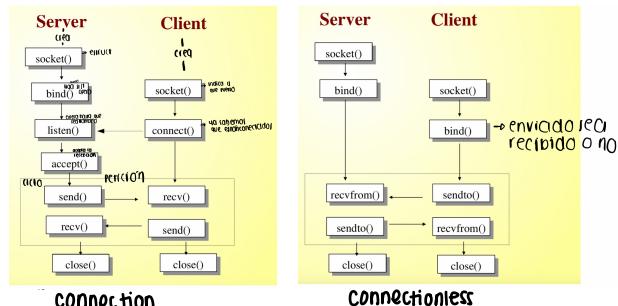
## Conceptos de hardware\*



Organizaciones básicas y memorias en sistemas de cómputo distribuidos

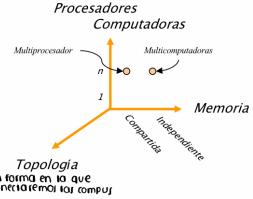
### Byte Ordering

- Network byte order: High stored in lowest
- Host Byte order: Low stored in lowest Little Endian



- int CLOSE () closes connection corresponding to the sockets descriptor and frees socket descriptor
- sendto() & recvfrom() Prevents sends & receives

## Hardware



### Aplicaciones distribuidas:

- Multicomputadoras
- Sin memoria compartida
- SO de red o basado en middleware

## Software



### Aplicaciones paralelas:

- Multiprocesador
- Memoria compartida
- SO distribuido

un programa puede planear su ejecución paralela

## Software

- DOS fuertemente acoplado: esconde y administra recursos
- NOS débilmente acoplado: ofrece servicios locales
- Middleware sobre NOS: transparencia en la distribución
- help application process to communicate with each other using unix

## Sockets

- interface between application process + transport layer
- file descriptor
- types:
  - internet → IP address (4 bytes)
  - unix → Port number (2 bytes)
  - X25
- Stream S: connection oriented, rely on TCP (2 way)
- Datagram S: connection unreliable, rely on UDP

### Socket Structure

- int socket(): returns socket descriptor for use in later calls
- struct sockaddr: Hold socket address info for many types of sockets
- struct sockaddr\_in: Parallel struct, easy to reference elements on socket address

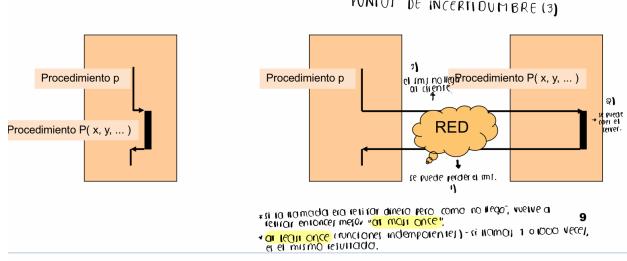
## Bind

- used to associate a socket with a port to match incoming packets to a process
- int bind () → always before listen
- int listen () → waits for incoming connections
- int connect () → connects to remote host (not always bind)
- int accept () → gets pending connection on the port you are listening
- info stored in address returns new socket file descriptor
- int send () → communicating over stream
- int received () → return # bytes actually sent
- int received () → sockets or connected diagrams
- return # bytes actually read
- 0 → closed connection

## RPC

- herramienta de alto nivel para implementación del modelo cliente - servidor
- Al menos 2 mensajes intercambiados
  - consulta: llamada al procedimiento (parámetros de llamada)
  - respuesta: parámetros de resultados

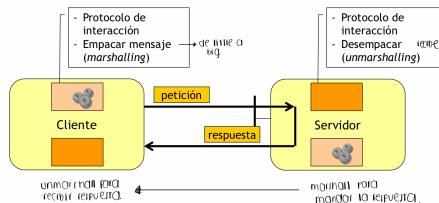
### Principio:



- Ventajas
  - Forma y efecto = a llamada de proceso local
  - Aplicaciones no modificadas al pasar por RPC
  - Validación local antes de pasar a la dist
  - Simplificación de uso

- Problemas
  - Semántica compleja si falla
  - Procesos son independientes
  - red- fuente de incertidumbre
  - Restrición sobre parámetros
  - Difícil transferencia de estructuras complejas

## Primitivas de bajo nivel



### Talón Cliente Stub

- Recibe llamada en modo local
- Transforma local en distante enviando mensajes
- Recibe resultados
  - después de ejecución
- Regresa parámetros de resultados como regreso de los procedimientos

### Talón Servidor Skeleton

- Recibe llamada en forma de mensaje
- Ejecución sobre sitio servidor por el procedimiento servidor
- Retransmite resultados del mensaje

## Principios de Implementación

- La transmisión de parámetros entre el cliente y servidor:
  - Modos: valor o referencia
  - Estructuras complejas
  - Representación
    - marshalling
    - unmarshalling
  - Tratar la heterogeneidad

Los errores se tratan:

- hipótesis sobre los errores, modos de detección
- Semántica de la llamada en caso de error
- Construcción de talones generación automática

- Administración de la ejecución: inicialización del servidor

- Portmapper
  - puerto fijo (111)
  - servidor conocido
  - servicio local al servidor

### Análisis de Código

- Determinar el conjunto de funciones que el servidor maneja
- Determinar el conjunto de variables relacionadas con dichas funciones

Objetos a designar

## Designación y ligado

- El proceso llamado, el sitio del servidor:
  - Designación independiente de la localización
  - Potencialidad de reconfiguración
  - errores
  - distribución de carga

### Momentos del ligado:

- Precoz / Estático: localización del servidor conocida desde la Composición
- Tardío / Dinámico: localización del servidor en tiempo de ejecución
  - Designación simbólica de servicios (no asociada a un sitio de ejecución)
  - Potencialidad de implementación selección

- Ligado a la Primera llamada
  - Consulta del servidor de nombres sólo en la primera llamada. No te mueves, siempre mismo servidor

- Ligado a cada llamada

- Consulta del servidor de nombres a cada llamada

- Ninguna llamada al servidor de nombres (Llamada resueta en tiempo de compilación)

- solo incluye los prototipos de las funciones en la parte servidor

- Código para el 'Cliente'
  - se debe separar del código así como de las variables asociadas

## Sun RPC: Interfaz y generación

- Hay una herramienta para la generación de los talones para la aplicación distribuida:
- El comando rpcgen
- Un lenguaje de especificación

### Archivo de especificación

- Declaraciones de las constantes que son usadas por ambos: el cliente y el servidor
- Declaraciones de tipos de datos que serán usados en las llamadas remotas
- Declaraciones del Programa remoto: Los procedimientos provistos en el Programa La información para el control de versiones

### En código:

- mensaje.h
  - Declaraciones de funciones, constantes y tipos que serán usadas
- mensaje-clnt.c
  - Código no genérico
  - Talon cliente → stub
- mensaje-SVC.C
  - Talon servidor → skeleton

### Ideas para la adaptación del código:

- Adaptación del cliente para iniciar/cerrar la conexión
- Escrutura de las rutinas de adaptación de llamadas y de parámetros

### Adaptadores

- El código debe declarar una función: que reciba la llamada (formato original) → lo adapte a la forma del llamado remoto
- Usar el prototipo de función generado por "rpcgen"

### Gestión de parámetros

- Los espacios de direccionamiento del que llama y del llamado ≠: No hay paso por referencia. Los apuntadores pierden significado → Dificultad para pasar estructuras complejas
- Los parámetros se transmiten sobre la red: Representación serializada de las estructuras
- Las máquinas cliente-servidor pueden ser heterogéneas → Conversión de formatos protocolo: Copy, Restore

## Problemas de Representación

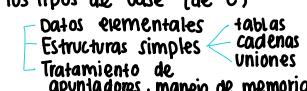
- sentido de los octetos de una cadena de caracteres
- representación de números derecha
- representación de los reales izquierda
- representación de las estructuras complejas flotantes simples y dobles

## Representación de Parámetros

- Solución ASN.1 → Abstract Syntax Notation Normalizada
  - Sintaxis abstracta para representar la estructura
  - Codificación independiente, SO's de los datos
  - Herramientas de generación disponibles para máquinas específicas
- Otras soluciones
  - Representación externa común SUN XDR usada por NFS
  - Conversión Explícita a partir de una representación local al servidor → al cliente
  - Negociación entre el cliente - servidor

## Conversion de parámetros: XDR → External Data Representation

- Conjunto de rutinas disponibles para los tipos de base (de C)



- Rutinas de conversión XDR

Los parámetros son apuntadores hacia las zonas que contienen los valores en formato interno o en formato de red  
La misma rutina para los dos sentidos el parámetro indica el sentido

## Especificación de Interfaces

- Uso de un lenguaje Especificación común al cliente y al servidor (a veces adaptado a lenguajes múltiples)

Definición de los tipos y la naturaleza de los parámetros

IN, OUT, IN-OUT, por valor, x referencia automáticamente:

- El talon cliente → proxy-stub
- El talon servidor → esqueleto-skeleton

Uso de programas empaquetado → desempaquetado  
Formato de transmisión sobre la red  
Biblioteca de programas de conversión

Corriente: realizada por los generadores de talones  
Avanzado: directamente por el programador para el tratamiento de estructuras de datos complejos

## Generación de Talones

- La estructura de los talones está bien definida y puede ser creada automáticamente
- información necesaria:

Dependiente del ambiente local: procedimientos de conversión y protocolo de comunicación

Dependiente de la aplicación: formato de parámetros y resultados

- Realización: la información dependiente de la aplicación se define con un lenguaje de descripción de interfaces IDL

Interfaz: contrato entre cliente y servidor

Definición abstracta común: Independencia de un lenguaje particular (adaptado a lenguajes múltiples)  
Independencia de representación de tipos  
Independencia de la máquina

Contenido mínimo: Identificación de procedimientos → nombre versión  
Definición de tipos de parámetros, resultados y excepciones  
Definición de modo de paso (IN, OUT, IN-OUT)

## Extensiones posibles

- Procedimientos de conversión de tipos
- Propiedades no-funcionales complejas QoS → (seguridad, velocidad)