Tema 4.
Llamada a Procedimientos Remotos

Sistemas Distribuidos
Grado de Ingeniería Informática.

ETSI
ESCUELA TÉCNICA SUPERIOR DE INGENIERÍA

DEPARTAMENTO DE TECNOLOGÍAS DE LA INFORMACIÓN

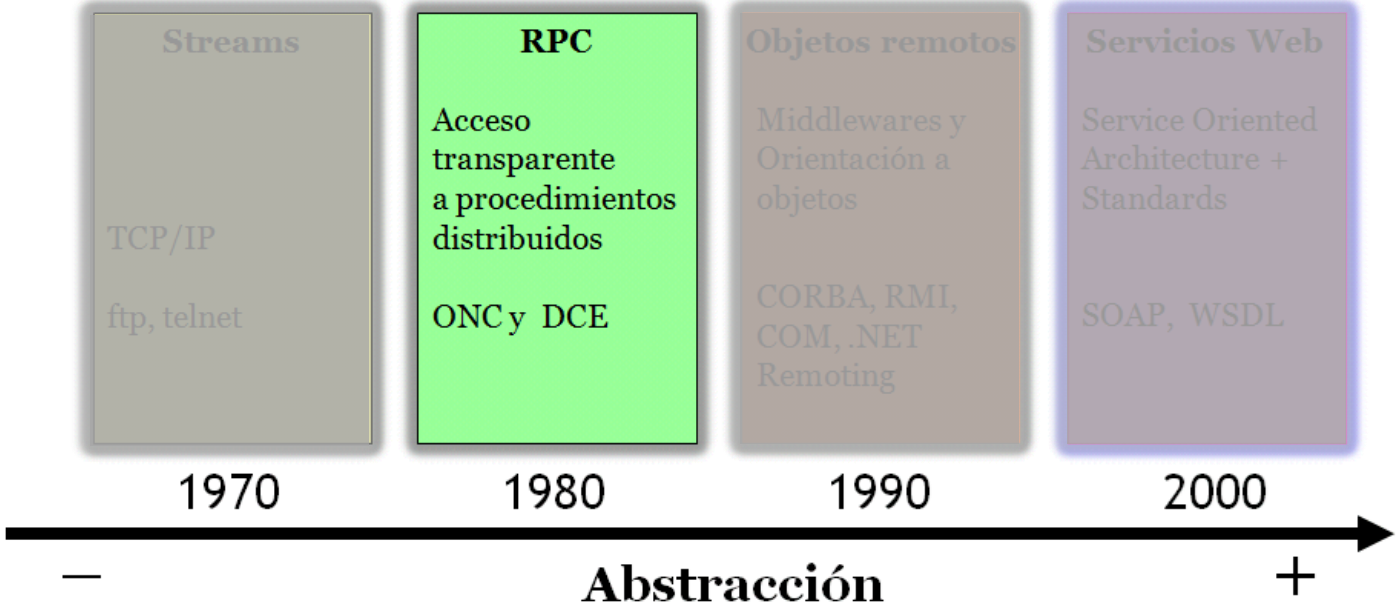
Universidad de Huelva

Contenido

1. Introducción .
2. Funcionamiento general del RPC.
3. Diferencia con las llamadas locales
4. Comportamiento ante fallos.
5. Semántica de las llamadas RPC
6. Implementación de los entornos RPC
7. Componentes típicos de los entornos RPC.
8. Ejemplo

Introducción

- La siguiente evolución del mecanismo de comunicación entre procesos (IPC) es la llamada a procesos remotos RPC (*Remote Procedure Call*) desarrollada sobre los años 80



Introducción

- ✓ Hasta ahora no se puede decir que la comunicación por socket otorgue a la propiedad de transparencia a los Sistemas Distribuidos.
- ✓ Surge la necesidad de ocultar o abstraer los detalles relativos a la comunicación entre aplicaciones. Para ello se necesitaría abstraer:
  - ➢ La *gestión* de los diálogos petición-respuesta.
  - ➢ El *aplanamiento* y *formateo* de datos (enteros, reales, cadenas, estructuras,...) [*marshaling, serializacion*]
    - *Aplanar: organizar datos complejos en un mensaje*
    - *Desaplanar: extrar datos complejos de un mensaje aplanado*
    - *Gestionar: representación de la información (orden de bytes, tipos complejos, alineado en* memoria), diferencias de hardware y S.O.
  - ➢ La *gestión* de la interfaz de comunicación (crear y configurar sockets, conectar, escribir, leer, etc.)

Introducción

- ✓ *Aproximación*: llamada a procedimientos remotos (RPC) (*Remote Procedure Call)*
  - ➢ Generar automáticamente el código usado en esas tareas comunes.
  - ➢ Ofrecer el entorno y los componentes de apoyo necesarios para dar soporte a esa infraestructura.
  - ➢ Procedimiento llamante y procedimiento llamado se ejecutan en máquinas distintas.
  - ➢ Ofrecer la ilusión de que la llamada remota parezca idéntica a una llamada local (transparencia).

Introducción

- ✓ *Objetivo*: Proporcionar un middelware que simplifique el desarrollo de aplicaciones distribuidas
  - ➢ Evitar que programador tenga que interactuar directamente con el interfaz de Sockets. *Abstraer* los detalles relativos a la red.
  - ➢ Servidor ofrece procedimientos que el cliente llama como si fueran procedimientos locales.

    - Se busca ofrecer un entorno de programación lo más similar posible a un entorno no distribuido.
    - El sistema RPC oculta los detalles de implementación de esas llamadas remotas. Éstas se implemente mediante un diálogo petición-respuesta.

El funcionamiento general de un sistema RPC

- ✓Proceso llamador (*cliente*):
  - ➢Proceso realiza la llamada a una función.
  - ➢Llamada empaqueta id. de función y argumentos en mensaje
  - ➢Envía mensaje a otro proceso.
  - ➢Queda a la espera del resultado.
  - ➢Al recibirlo, lo desempaqueta y retorna el valor

- ✓Proceso llamado (*servidor*):
  - ➢Recibe mensaje con id. de función y argumentos.
  - ➢Se invoca función en el servidor.
  - ➢Resultado de la función se empaqueta en mensaje.
  - ➢Se transmite mensaje de respuesta al cliente.

Diferencias con llamadas locales (LPC)

- ✓ *El manejo de errores*: con RPC pueden existir fallos en *servidor* en el *cliente* o en la *red*
- ✓ *Acceso a variables globales y efectos laterales en el cliente no son posible*. El procedimiento remoto (servidor) no tiene acceso al espacio de direcciones del cliente. Imposibilidad de usar punteros.
- ✓ *Los parámetros para la llamada remota no pueden pasarse por referencia*, sólo por valor. Generalmente se usan mecanismos de copia y restauración para ”simular” el paso por valor.
- ✓ *Rendimiento de llamadas RPC mucho menor que en llamadas locales*. Hay una mayor sobrecarga en llamadas RPC (transferencia por red, aplanamiento de datos, etc)
- ✓ En alguno entornos se *limita el intercambio* de estructuras complejas, en otros se usan métodos de *aplanado/desaplanado*.

Comportamiento ante fallos

- ✓ Aspecto clave que determina la equivalencia semántica entre llamadas remotas y llamadas locales.
- ✓ Llamadas locales ofrecen una semántica "*exactamente una vez*" (ejecución fiable).
  - ➢ El entorno de ejecución de las llamadas locales garantiza que el procedimiento llamado se ejecuta exactamente una vez.
  - ➢ Llamada termina devolviendo un valor de retorno si tuvo éxito o una indicación del error (excepción, código de error) en caso de fallo Llamador se queda en espera indefinidamente hasta que finalice llamada.
  - ➢ En RPC no es posible espera indefinida (posibilidad de fallos).

Comportamiento ante fallos

- ✓En llamadas remotas las posibles fuentes de fallos son múltiples.
  - ➢*Fallos en los procedimientos llamados*.
    - La ejecución del proceso llamado se detiene por errores del hardware o del sistema operativo que lo ejecuta (ej.: caída del sistema).
    - También por errores internos del propio procedimiento (divisiones por cero, índices de arrays fuera de rango, etc.).
  - ➢*Fallos en la comunicación*.
    - *Pérdida de la conexión*: la red deja de enviar paquetes (caída de la red, pérdida de un enlace, etc.).
    - *Corrupción* del *contenido* de alguno de los mensajes enviados.
    - *Pérdida de paquetes*: algún mensaje/s no llega a su destino.
    - *Recepción fuera de orden*: paquetes retrasados recibidos de forma desordenada.

Comportamiento ante fallos

- ✓ En general el *proceso cliente no tiene capacidad* para distinguir los diferentes tipos de errores.
  - ➢ Cliente sólo percibe que una o más de sus peticiones no reciben respuesta, pero no llega a saber por qué:
    - La petición no llegó al proceso remoto.
    - El servidor está caído o no ha llegado a procesar la petición.
    - La petición llegó y el servidor la procesó, pero la respuesta no llegó al cliente.
  - ➢ Dependiendo de los mecanismos definidos y de la semántica se puede volver a pedir la ejecución del procedimiento remoto o no.

- ✓ La forma de gestionar los fallos por parte del entorno RPC determina la semántica efectiva de las llamadas remotas.

Semántica de las llamadas RPC

- ✓ Dependiendo del modelo de gestión de fallos por parte del entorno RPC se pueden soportar distintas aproximaciones a la semántica *"exactamente una vez"* de las llamadas locales (LPC).

- ✓ No es posible garantizar la semántica *"exactamente una vez" debido* a la posibilidad de fallos de comunicación.

- ✓ Hay tres tipos de semántica en las llamadas RPC (de menor a mayor complejidad):
  - semántica *tal vez*.
  - semántica *al menos una vez*.
  - semántica *como máximo una vez*.

Semántica de las llamadas RPC

Semántica *tal vez*: *El p*rocedimiento remoto puede ejecutarse una vez o ninguna.

- ✓ Cliente puede recibir una respuesta o ninguna
- ✓ Funcionamiento:
  - Cliente envía petición y queda a la espera un tiempo.
  - Si no llega respuesta dentro del tiempo de espera, continúa su ejecución.
  - Cliente no tiene realimentación en caso de fallo (no sabe que pasó)
- ✓ Sólo admisible en aplicaciones donde se tolere la pérdida de peticiones y la recepción de respuestas con retaso (fuera de orden)

Semántica de las llamadas RPC

Semántica *al menos una vez:* Procedimiento remoto se ejecuta una o más veces.

- ✓ Cliente puede recibir una o más respuestas
- ✓ Funcionamiento:
  - Cliente envía petición y queda a la espera un tiempo.
  - Si no llega respuesta o ACK dentro del tiempo de espera, repite la petición.
  - Servidor no filtra peticiones duplicadas, el procedimiento remoto puede ejecutarse repetidas veces.
  - Cliente puede recibir varias respuestas.
- ✓ Sólo es aplicable cuando se usan exclusivamente operaciones idempotentes (repetibles), es decir, *se puede ejecutar varias veces resultando el* mismo efecto que si se hubiera ejecutado sólo una.
- ✓ Admisible en aplicaciones donde se tolere que se puedan repetir invocaciones sin afectar a su funcionamiento.

Semántica de las llamadas RPC

Semántica *como máximo una vez*: El procedimiento remoto se ejecuta exactamente una vez o no llega a ejecutarse ninguna.

- ✓ Cliente recibe una respuesta o una indicación de que no se ha ejecutado el procedimiento remoto.
- ✓ Funcionamiento:
  - Cliente envía petición y queda a la espera un tiempo.
  - Si no llega respuesta o ACK dentro del tiempo de espera, repite la petición.
  - Servidor filtra las peticiones duplicadas y guarda historial con las respuestas enviadas (servidor con memoria). El procedimiento remoto sólo se ejecuta una vez.
  - Cliente sólo recibe una respuesta si la petición llegó y se ejecutó el procedimiento, si no recibe informe del error.

Implementación de entornos RPC

- ✓ Para ofrecer un mecanismo de llamada a procedimientos remotos sintácticamente equivalente al de llamada local el entorno RPC debe proporcionar y dar soporte a una infraestructura que ofrezca transparencia en la invocación remota.

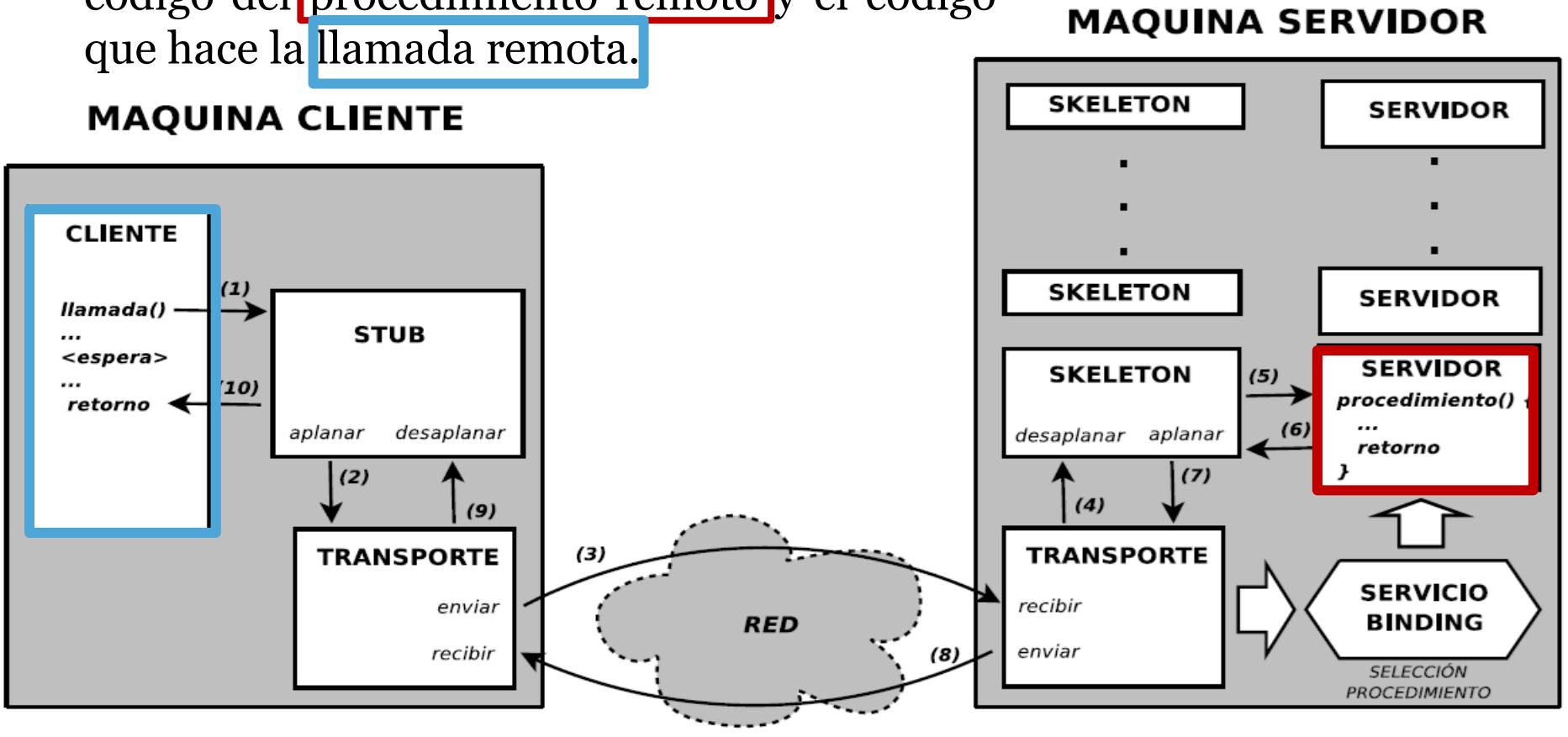
Objetivo: es deseable que el programador del sistema distribuido no perciba la diferencia entre llamada local y llamada remota.

Idea clave: uso de "*representantes*", tanto del cliente como del servidor

- ➢ Representante del servidor en la máquina cliente (*stub*): realiza el papel de servidor en la máquina cliente.
- ➢ Representante del cliente en la máquina servidor (*skeleton*): realiza el papel de cliente en la máquina servidor.
- ➢ Proporcionan transparencia en la llamada remota.
- ➢ Generados automáticamente en base a la interfaz definida para el procedimiento remoto.

Implementación de entornos RPC

- ✓ El programador sólo debe programar el código del procedimiento remoto y el código que hace la llamada remota.



Componentes típicos de los entornos RPC

Stubs (representante del servidor → recibe la llamada del cliente).

- ✓ Proporciona transparencia en el lado del cliente.
- ✓ Posee un interfaz idéntico al del procedimiento remoto (misma declaración). Cada procedimiento remoto que desee llamar el cliente debe tener su propio stub.
- ✓ El cliente realiza una llamada local al procedimiento del stub como si fuera el servidor real.
- ✓ Tareas realizadas por el stub
  - Localiza al servidor que implemente el procedimiento remoto
  - Empaqueta los parámetros de entrada (*aplanado*, *marshalling*) en un formato común para cliente y servidor.
  - Envía el mensaje resultante al servidor.
  - Espera la recepción del mensaje de respuesta.
  - Extrae resultados (*desaplanado*, *unmarshalling*) y los devuelve al cliente que hizo la llamada.

Componentes típicos de los entornos RPC

Skeleton (representante del cliente → realiza la llamada al servidor)

- ✓ Proporciona transparencia en el lado del servidor.
- ✓ Conoce el interfaz ofrecido por el procedimiento remoto. Cada procedimiento remoto que ofrece el servidor debe tener su propio skeleton.
- ✓ Realiza llamadas locales al servidor como si fuera el cliente real responsable de la invocación "*real*" al procedimiento remoto.
- ✓ Tareas realizadas por el skeleton.
  - Ejecuta bucle de espera de mensajes.
  - Recibe petición y desempaqueta el mensaje (desaplanado).
  - Determina qué método concreto invocar.
  - Invoca el procedimiento con los argumentos recibidos y recupera el valor devuelto.
  - Empaqueta el valor devuelto (aplanado, marshalling).
  - Envía mensaje al stub del cliente.

Componentes típicos de los entornos RPC

Servicio de binding

- ✓ Responsable de la transparencia de localización.
- ✓ Servicio auxiliar que complementa a stub y skeleton.
- ✓ Gestiona la asociación entre el nombre del procedimiento Remoto con su localización en la máquina servidor (dirección, puertos, skeleton, etc.).
- ✓ Realiza la búsqueda del skeleton de la implementación concreta del procedimiento remoto llamado por un cliente.
- ✓ Selecciona skeleton+servidor que atenderá la llamada remota.
- ✓ Ejemplos: portmapper en Sun-RPC

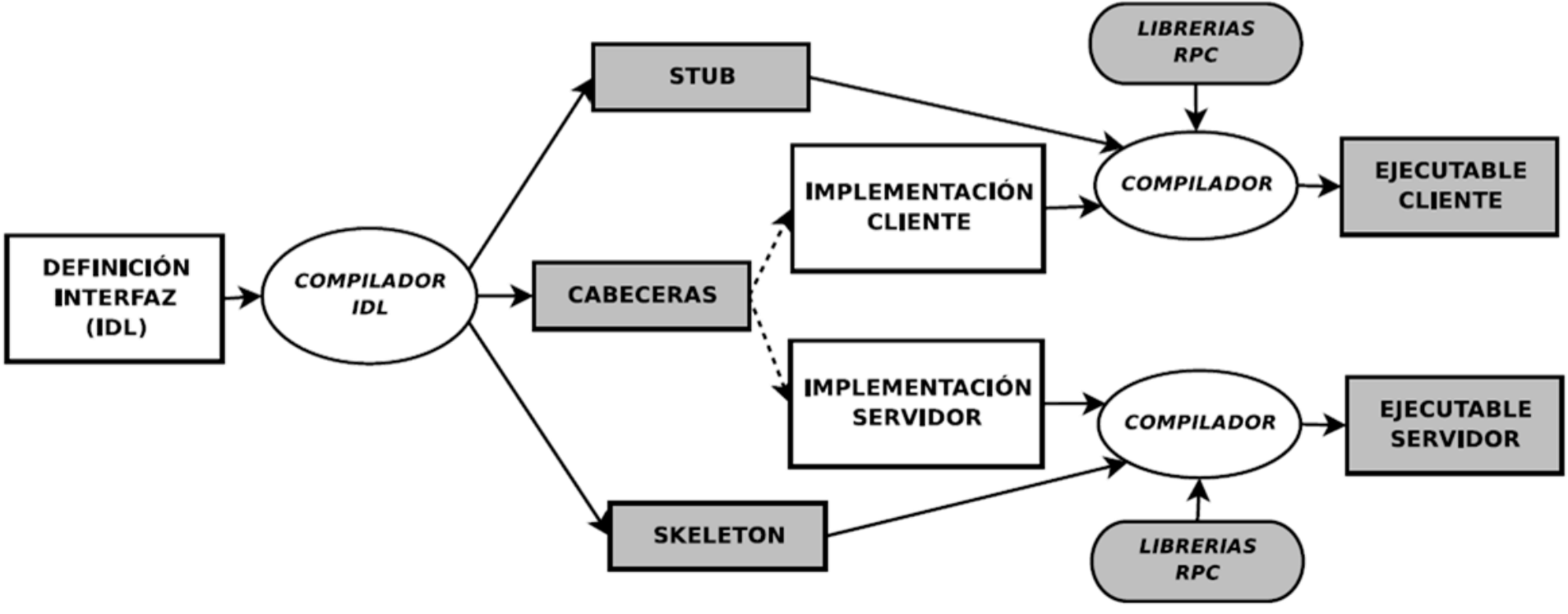
Componentes típicos de los entornos RPC

Compilador de interfaces

- ✓ A partir de la descripción del interfaz del procedimiento remoto genera de forma automática el código del stub y del skeleton.
- ✓ Dependiendo del entorno RPC puede generar otro código adicional necesario como el esqueleto del servidor y del cliente.
- ✓ El Interfaz del procedimientos remoto especifica:
  - ➢ El interfaz ofrecido por el procedimiento (args. de entrada + valor devuelto).
  - ➢ Cómo será el aplanado/desaplanado.
  - ➢ Opcionalmente aporta información que se usará para localizar el procedimiento remoto (número de versión).

Componentes típicos de los entornos RPC

Compilador de interfaces



Ejemplo: Calculadora remota

Esquema de generación de stub y skeleton a partir de la definición XDR del interfaz remoto