Unidad II: Remote Procedure Call (RPC)

Profesor:

M. en C. Hermes Francisco Montes Casiano

Desarrollo de Sistemas Distribuidos

Instituto Politécnico Nacional

Contenido

1. Introducción
2. ¿Cómo trabaja RPC?
3. Desarrollo de aplicaciones con RPC
4. Llamadas a procedimientos Remotos

Introducción

¿Qué es RPC?

RPC es una técnica para la construcción de aplicaciones distribuidas basadas en la arquitectura cliente servidor.

- RPC se basa e la forma convencional de invocar un procedimiento de forma local, permitiendo que el procedimiento a invocar exista en otro espacio de direcciones.
- Ambos procesos se pueden encontrar en el mismo host o en host distintos siempre y cuando exista una red de computadoras que los interconecte.
- Con RPC los desarrolladores de aplicaciones se evitan lo tedioso de trabajar con el detalle de las interfaces de red.

Independencia de transporte

La independencia del medio de transporte aísla a RPC de los elementos físicos y lógicos de las tecnologías de telecomunicación, permitiendo a la aplicación utilizar una amplia variedad de medios de transporte.

¿Cómo trabaja RPC?

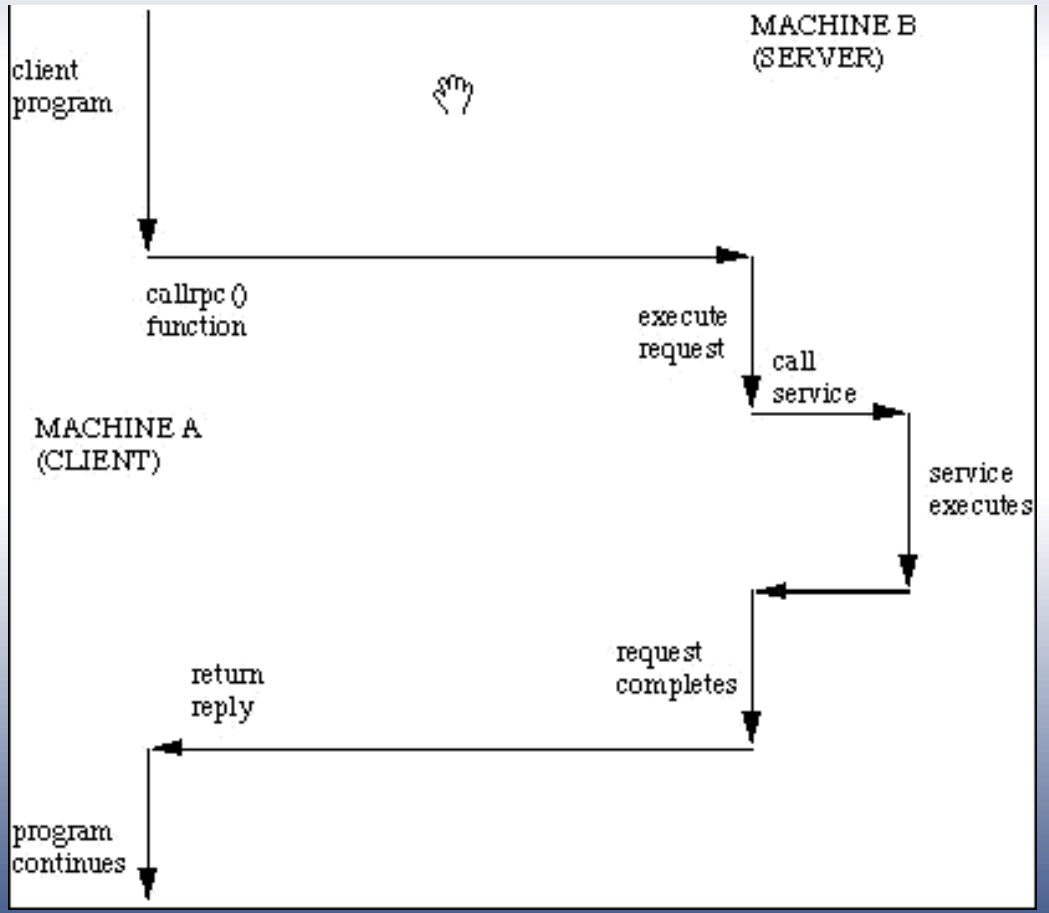
- RPC funciona de forma similar a la invocación de una función.
- Cuando se hace una llamada RPC, los argumentos de llamada se pasan al procedimiento remoto y quien invoca espera la respuesta.
  1. Un proceso cliente invoca un procedimiento remoto y se pone en espera de la respuesta.
  2. El hilo principal del proceso cliente es bloqueado hasta que la respuesta es recibida o se ha excedido el tiempo de espera.
  3. Cuando la petición se recibe en el proceso servidor, el servidor invoca a la rutina que atiende la petición con base en el servicio solicitado.
  4. El proceso servidor envía la respuesta al cliente.
  5. Después de que se completa la llamada RPC, el proceso cliente continua su ejecución.

Objetivo

RPC fue creado específicamente para dar soporte a aplicaciones en red.

¿Cómo trabaja RPC?

- RPC funciona de forma similar a la invocación de una función.
- Cuando se hace una llamada RPC, los argumentos de llamada se pasan al procedimiento remoto y quien invoca espera la respuesta.



Mecanismo RPC

- Un procedimiento remoto es identificado de forma única por los siguientes componentes:

  Número de programa: Identifica un grupo de procedimientos remotos relacionados, cada uno de los cuales tiene un número de procedimiento único.

  Versión: Cada versión consiste de una colección de procedimientos que están disponibles para ser invocados de forma remota. El número de versión habilita que múltiples versiones de un protocolo RPC se encuentren disponibles de forma simultánea.

  Número de procedimiento: Cada procedimiento tiene un número de procedimiento que lo identifica de forma única.

Desarrollo de aplicaciones con RPC (Ejemplo)

- Considere una búsqueda cliente/servidor en una base de datos con información de *personas* en una máquina remota.
- Se asume que no se puede conectar al equipo remoto a través de NFS.
- Se podría utilizar una terminal remota para ejecutar los comandos de la consulta.
  1. La ejecución podría se un poco lenta.
  2. Pero para poner hacerlo es necesario contar con una cuenta de acceso.
- RPC es la alternativa
  1. Establecer un servidor en el equipo remoto que puede responder a las consultas.
  2. Recuperar información llamando a una consulta que será más rápido que el método anterior.

Desarrollo de aplicaciones con RPC (Ejemplo)

- Se podría utilizar una terminal remota para ejecutar los comandos de la consulta.
  1. La ejecución podría se un poco lenta.
  2. Pero para poner hacerlo es necesario contar con una cuenta de acceso.
- RPC es la alternativa
  1. Establecer un servidor en el equipo remoto que puede responder a las consultas.
  2. Recuperar información llamando a una consulta que será más rápido que el método anterior.

Desarrollo

Para desarrollar una aplicación RPC es necesario realizar los siguientes tres pasos:

1. Especificar el protocolo para la comunicación cliente-servidor.
2. Desarrollar el programa cliente.
3. Desarrollar el programa de servidor.

Definir el protocolo

- La forma más fácil de definir el protocolo a utilizar es con un compilador como *rpcgen*.
- Para el protocolo es necesario definir el nombre de los procedimientos, los tipos de datos y los valores de retorno.
- El *compilador de protocolo* lee la definición y genera automáticamente los stub para el cliente y el servidor.
- *rpcgen* utiliza su propio lenguaje (RPCL), que es muy similar a las directivas del preprocesador.
-

Definir el protocolo

- El *compilador de protocolo* lee la definición y genera automáticamente los stub para el cliente y el servidor.
- *rpcgen* utiliza su propio lenguaje (RPCL), que es muy similar a las directivas del preprocesador.
-

Compilación

$ rpcgen rpcprog.x

Lo anterior generará:

1. rpcprog_clnt.c – stub cliente.
2. rpcprog_svc.c – stub servidor.
3. rpcprog_xdr.c – si es necesario XDR.
4. rpcprog.h – archivo de definición de XDR.

Definir el protocolo

Compilación

$ rpcgen rpcprog.x

Lo anterior generará:

1. rpcprog_clnt.c – stub cliente.
2. rpcprog_svc.c – stub servidor.
3. rpcprog_xdr.c – si es necesario XDR.
4. rpcprog.h – archivo de definición de XDR.

XDR

La representación externa de datos (XDR) es un resumen de los datos necesarios para realizar la comunicación independiente. El cliente y el servidor no necesitan ser máquinas del mismo tipo.

Lenguaje de Definición de Interfaz

- El lenguaje **SUN XDR** se diseño originalmente para especificar representaciones externas de datos.
- XDR se extendió para convertirse en un lenguaje de definición de interfaces.
- Puede utilizarse para la definición de una interfaz de comunicación y también para la definición de los tipos que las soportan.

XDR

1. SUN RPC no permite especificar nombres de interfaces, en su lugar se proporciona un número de programa y una versión.
2. Una definición de procedimiento especifica un signatura de un procedimiento y un número de procedimiento.
3. Sólo se permite un parámetro de entrada.
4. Los parámetros de salida de devuelven como un sólo resultado.
5. La signatura de un procedimiento consta de un tipo de resultado, el nombre de un procedimiento y el tipo de parámetro de entrada.

Definiendo el código del cliente y el servidor

- Deben comunicarse a través de los procedimientos y los tipos de datos especificados en el Protocolo.
- El lado del servicio tendrá que registrar los procedimientos que pueden ser llamados por el cliente y recibir y devolver los datos necesarios para su procesamiento.
- El proceso cliente invoca un procedimiento remoto enviando todos los parámetros requeridos y recibirá el valor de retorno.

Resumen de la interfaz de rutinas

- RPC tiene múltiples niveles de aplicación para sus servicios, para proveer diferentes grados de control a las aplicaciones.
- Las interfaces simplificados se utilizan para hacer llamadas a procedimiento remoto a rutinas en otras máquinas y especificar sólo el tipo de transporte a utilizar.

Nivel simplificado

rpc_gen: Registra un procedimiento como un programa RPC en todos los transportes del tipo especificado.

rpc_call: Llamadas remotas al procedimiento especificado en el host remoto especificado.

rpc_broadcast: Emite un mensaje de llamada *broadcast* a través de todos los medios de transporte del tipo especificado.

- La interfáz está divida de la siguiente manera:
  1. top level,
  2. intermediate level,
  3. expert level,
  4. bottom level.

Estas interfaces proporcionan al desarrollador mayor control sobre los parámetros de comunicación tales como el medio de transporte a utilizar, tiempos de espera, retransmisión de peticiones, etc.

Rutinas de nivel superior *(top level)*

- En el nivel superior, la interfaz sigue siendo simple.
- El programa tiene que crear un identificador de cliente antes de realizar una llamada o crear un identificador de servidor antes de recibir llamadas.
- Si desea que la aplicación se ejecute en todos los transportes debe utilizar ésta interfaz.

Función	**Descripción**
clnt_create	Creación de un cliente genérico.
clnt_create_timed	Permite al programador especificar el tiempo máximo permitido para cada tipo de transporte durante el intento de creación.
svc_create	Crea el servidor que manejará todos los tipos de transporte especificado.
clnt_call	El cliente invoca un procedimiento mediante una petición al servidor.

Rutinas de nivel medio

- La interfaz de nivel intermedio de RPC le permite controlar los detalles.
- Los programas escritos en este nivel son más complicados, pero son más eficientes.
- El nivel intermedio permite especificar el transporte a utilizar.

Función	**Descripción**
clnt_tp_create	Crea un cliente para el transporte especificado.
clnt_tp_create_timed	Permite al programador especificar el máximo tiempo permitido.
svc_tp_create	Crea un servidor para atender peticiones sobre un tipo especifico de transporte.
clnt_call	El cliente invoca un procedimiento mediante una petición al servidor.

Rutinas de nivel experto

- El nivel experto tiene un mayor número de funciones con las que se puede especificar los parámetros de transporte.

Función	**Descripción**
clnt_tli_create	Crea un cliente para un transporte específico.
svc_tli_create	Create un servidor para un transporte específico.
rpcb_set	Invoca *rcpbind* para asignar una correspondencia entre un RPC y una dirección de red..
rpcb_unset	Elimina el vínculo establecido por *rpcb_set*.

Rutinas de nivel experto

- El nivel experto tiene un mayor número de funciones con las que se puede especificar los parámetros de transporte.

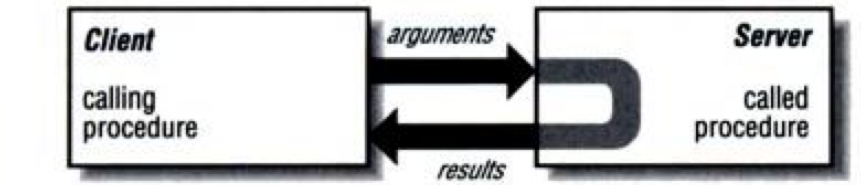
rpcb_getaddr	Invoca *rpcbind* para obtener la dirección de transporte de un servicio RPC.
svc_reg	Asocia el par número de programa y vesión con una especifica rutina de atención.
svc_unreg	Elimina la asociación establecida por *svc_unreg*.
clnt_call	El cliente invoca un procedimiento mediante una petición al servidor.

Interfaz de desarrollo RPC: interfáz simple

- Es el nivel más fácil de usar, ya que no requiere el uso de cualquier otra rutina de RPC.
- Limita el control de los mecanismos de comunicación subyacentes.
- El desarrollo del programa en este nivel puede ser rápida, y es apoyado directamente por el compilador *rpcgen*.
- Para la mayoría de las aplicaciones, rpcgen y sus instalaciones son suficientes.

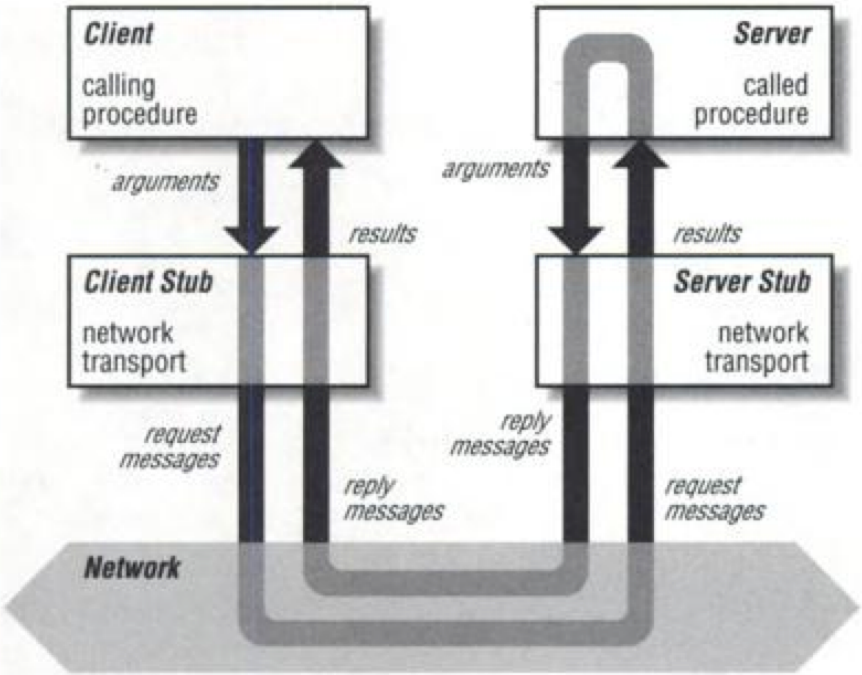
Objetivo

Las rutinas de la biblioteca de interfaz simplificada proporcionan acceso directo a las instalaciones de RPC para los programas que no requieren finos niveles de control.



In a local procedure call, a calling process executes a procedure in its own address space.

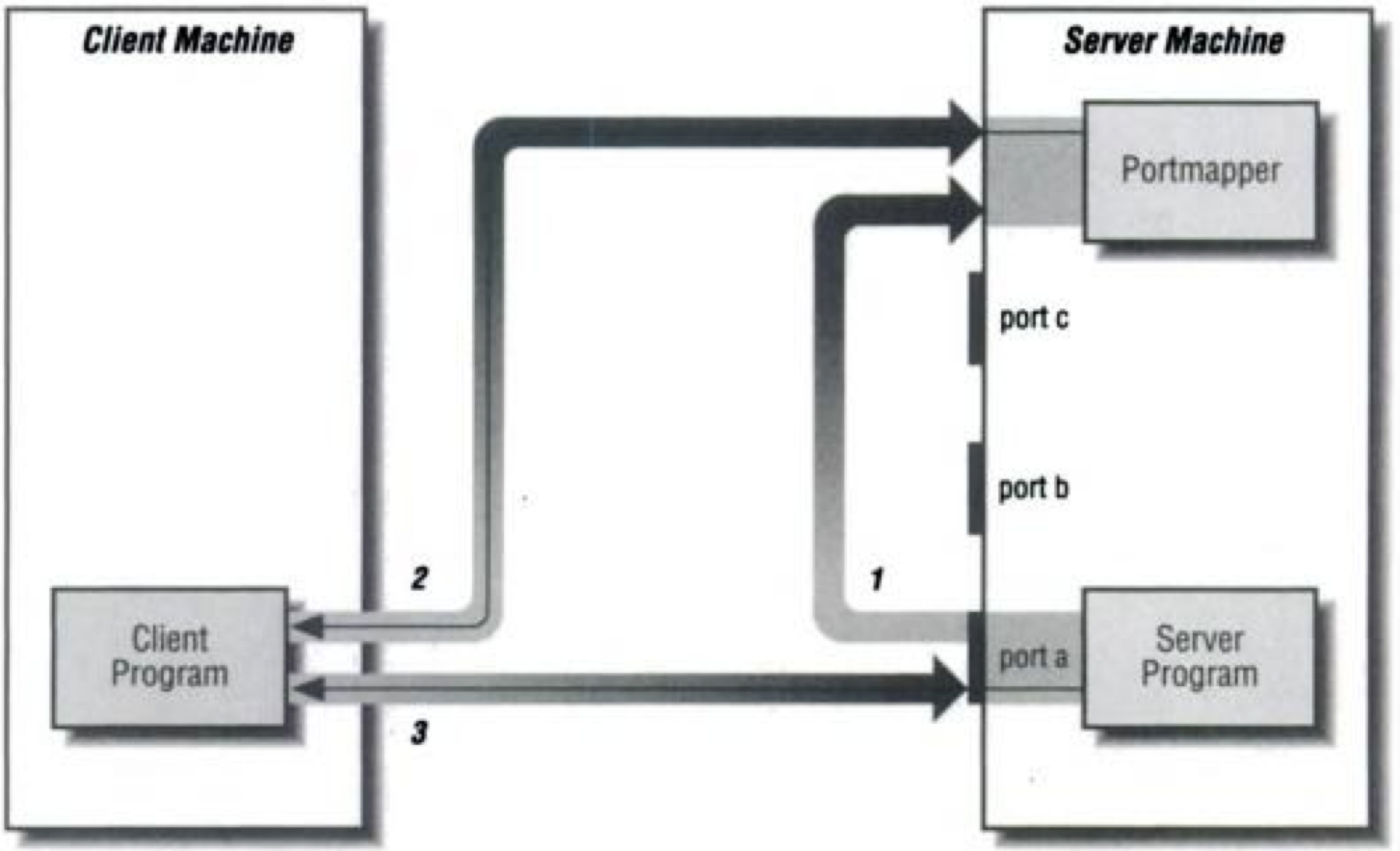
Local Procedure Call



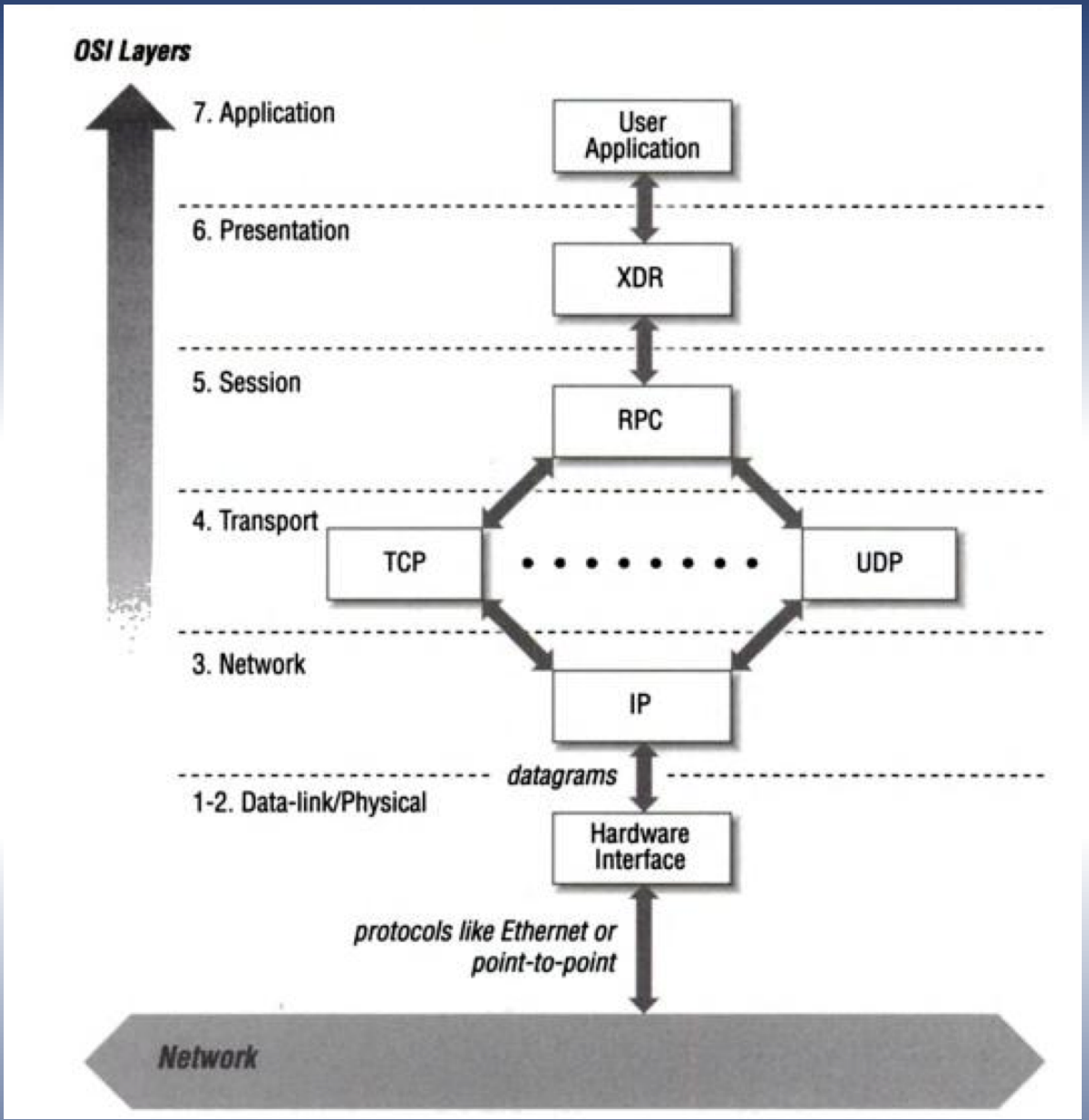
In a remote procedure call, the client and server run as two separate processes. It is not necessary for them to run on the same machine.

The two processes communicate through stubs, one each for the client and server. These stubs are pieces of code that contain functions to map local procedure calls into a series of network RPC function calls.

Remote Procedure Call



1. RPC server establishes address at which it listens for requests. It registers the port number (address) with the Portmapper, in this instance port a.
2. The client consults the Portmapper of the server machine to identify the port number that is to receive the RPC request.
3. The client and server now open a communication path to perform remote procedure execution. The client makes requests; the server sends replies.


OSI Layers
7. Application
User Application
6. Presentation
XDR
5. Session
RPC
4. Transport
TCP
UDP
3. Network
IP
datagrams
1-2. Data-link/Physical
Hardware Interface
protocols like Ethernet or point-to-point
Network

Llamadas a procedimientos remotos

- Muchos sistemas distribuidos se han basado en el intercambio explícito de mensajes entre procesos.
- Los procedimientos *send* y *receive* no ocultan en lo absoluto la comunicación, lo cual es importante para lograr la transparencia en los sistemas distribuidos.
- *Birrel* y *Nelson* sugirieron que procesos invocaran a procedimientos que se encontraran en otros *hosts*.
- Ningún mensaje de paso es visible para el desarrollador.

Problemas

1. Los procesos se ejecutan en distintos espacios de direcciones.
2. El paso de parámetros es complicado si las hosts son de diferentes arquitecturas.
3. Un host puede fallar por diferentes causas y cada una de ellas genera diferentes problemas.