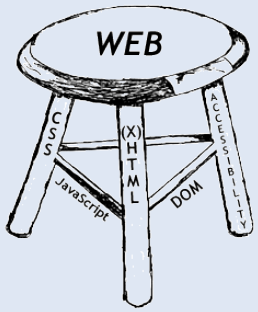
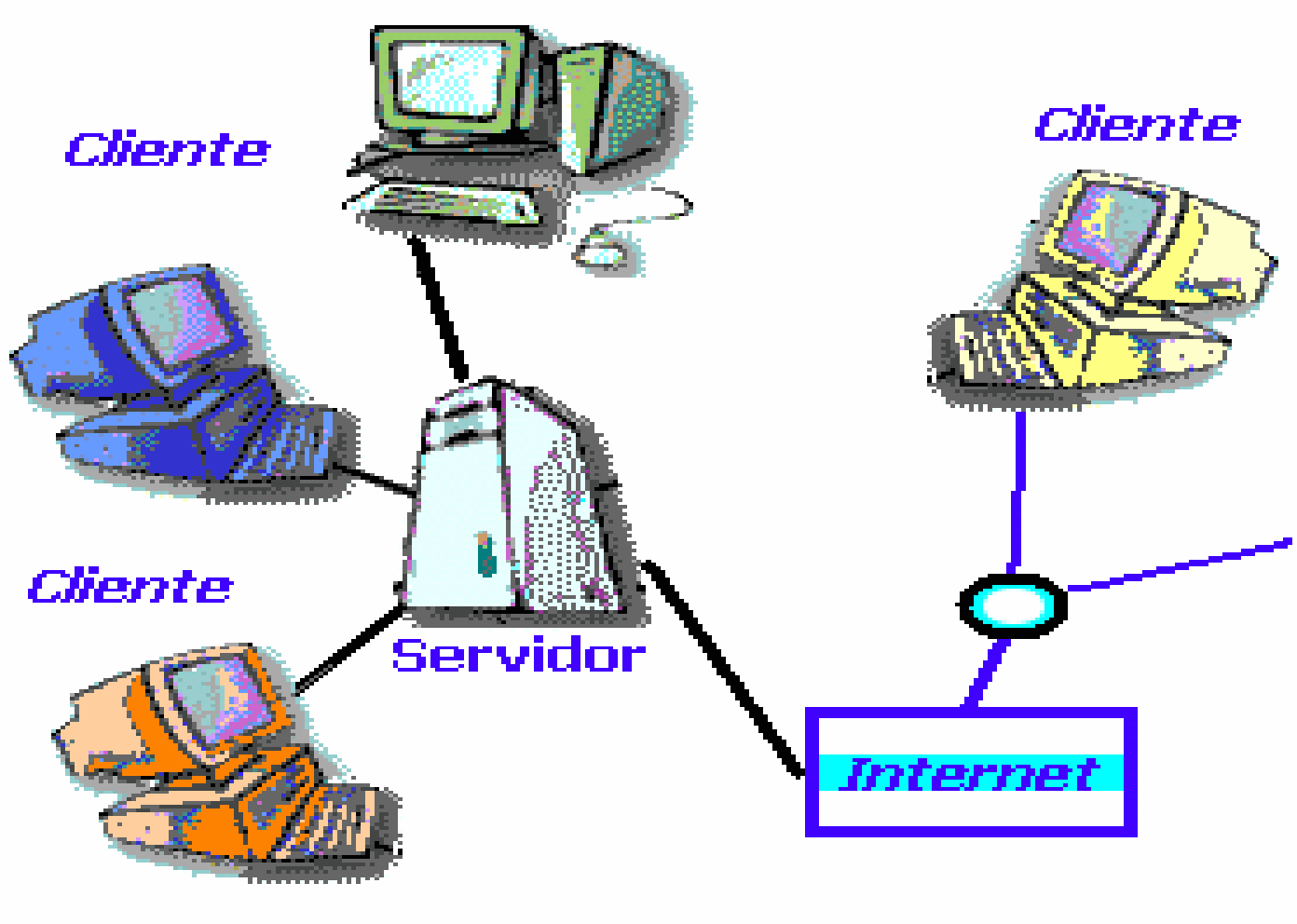
Modelo Cliente - Servidor



Alejandro Soto Ramos

Concepto



Propósito

Proporcionar servicios de cómputo distribuido, por medio de procesos que cooperan entre si.

Cliente/Servidor

Definición: Sistema distribuido entre múltiples procesadores donde hay clientes que solicitan servicios y servidores que los proporcionan.

Separa los servicios situando cada uno en su plataforma más adecuada.

Sistemas Distribuidos

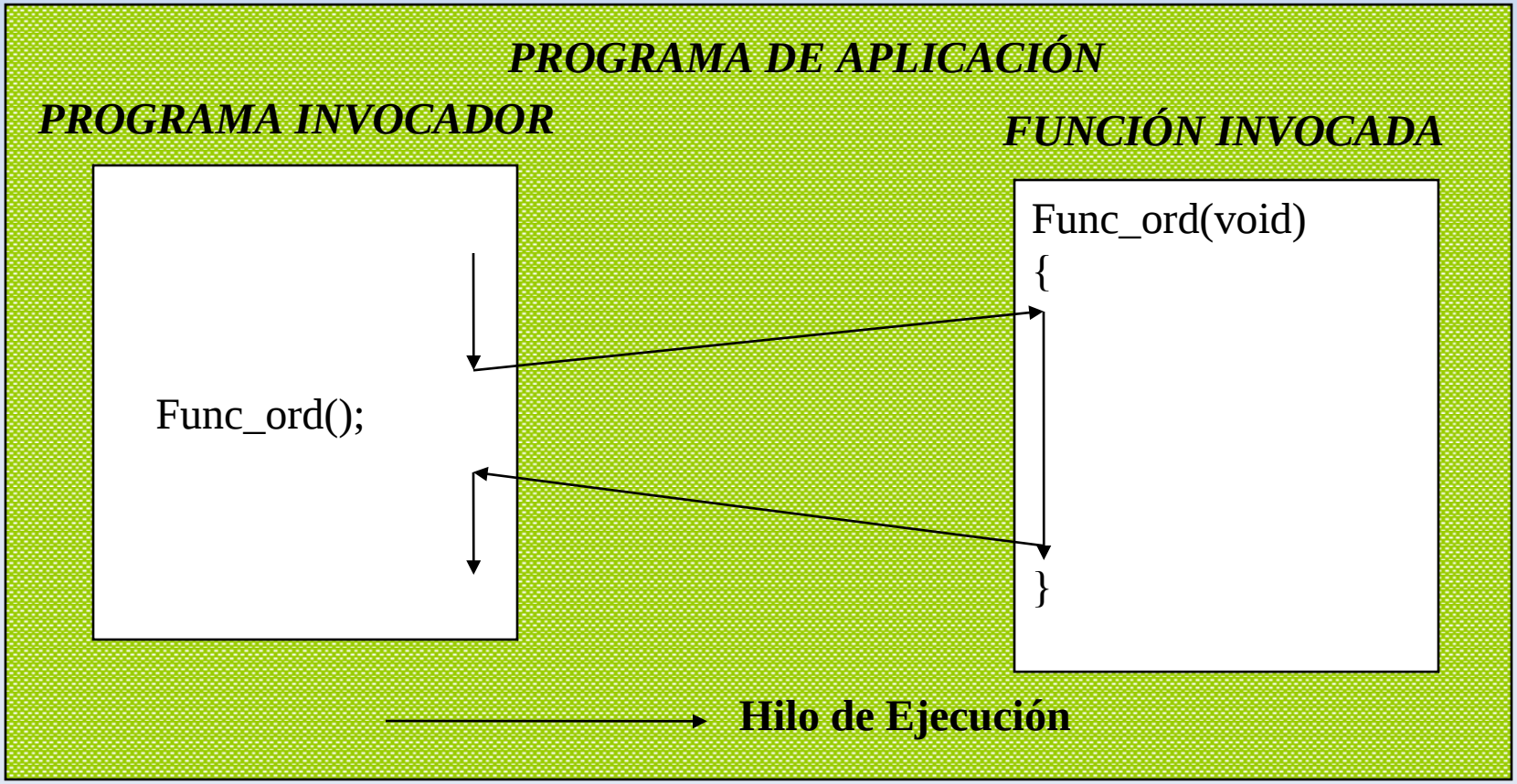
Se distinguen dos esquemas básicos de cómputo distribuido:

- Llamadas a procedimientos remotos
- Paso de mensajes

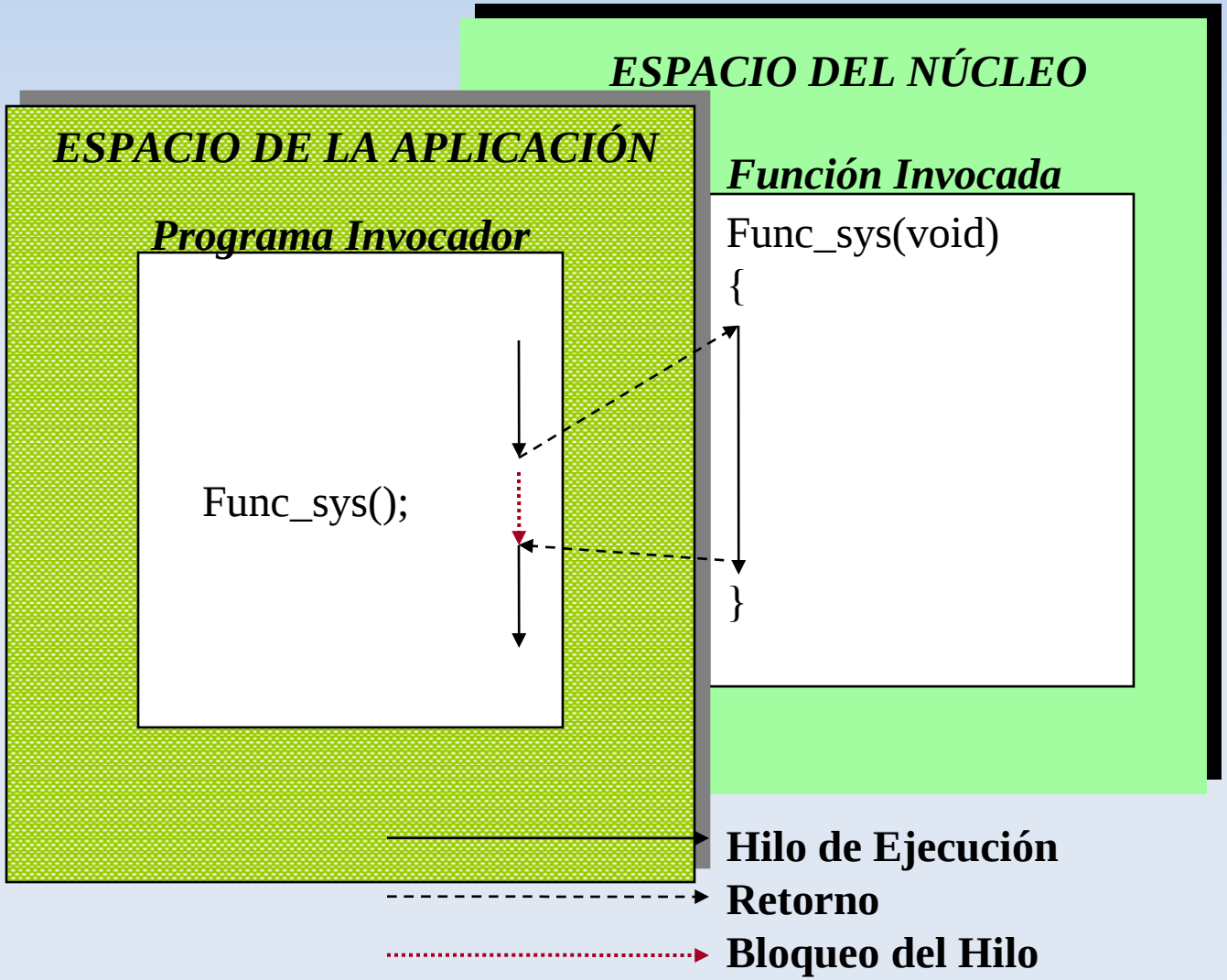
Llamadas a Procedimientos

Las llamadas a procedimientos permiten al programador hacer uso de servicios de cómputo disponibles en bibliotecas de la misma aplicación, en el sistema operativo o en otros equipos conectados a la red

Hilo de Ejecución de una Llamada a Función Ordinaria



Hilo de Ejecución de una Llamada de Sistema

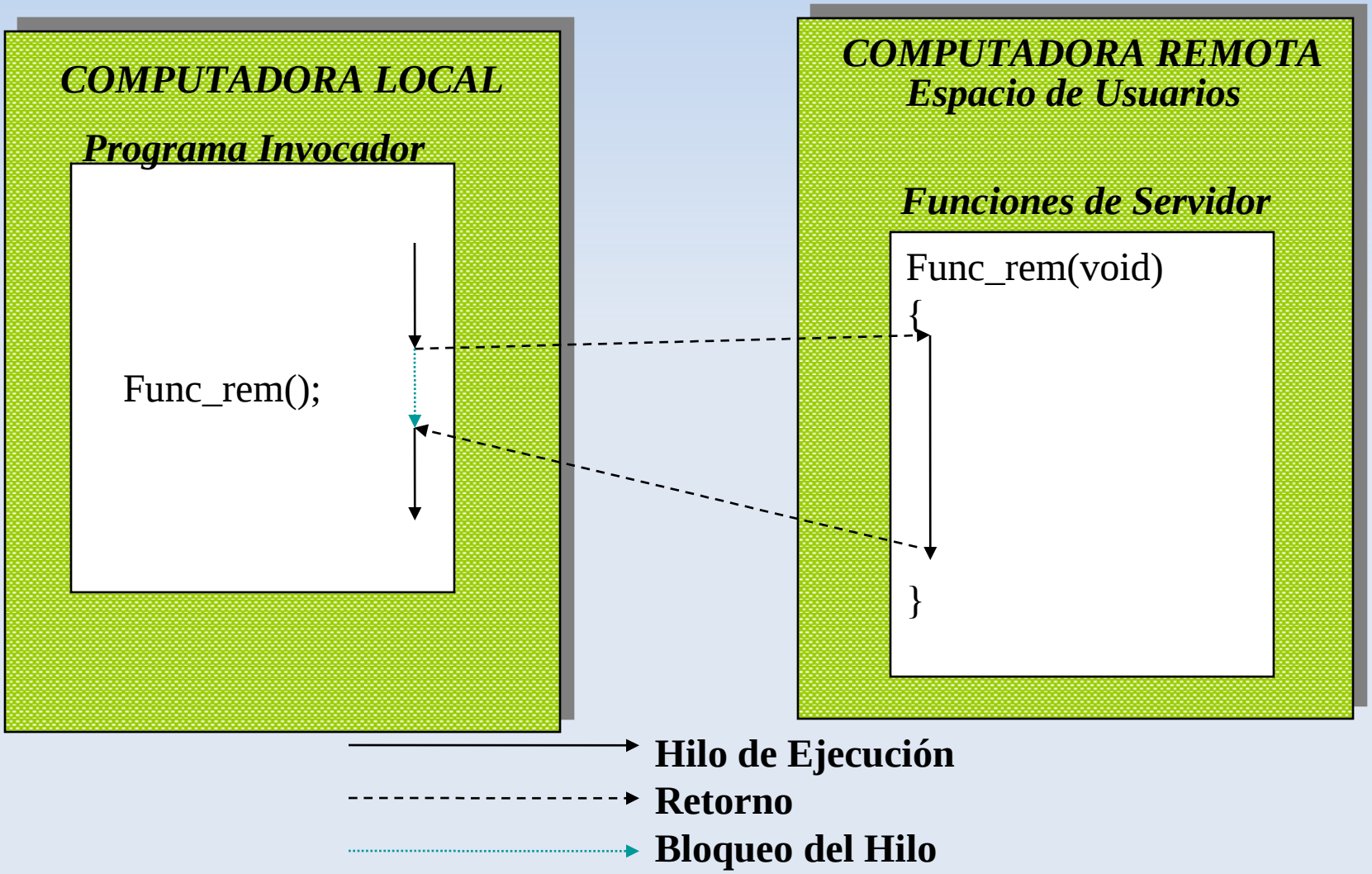


Llamadas a Procesos Remotos RPCs

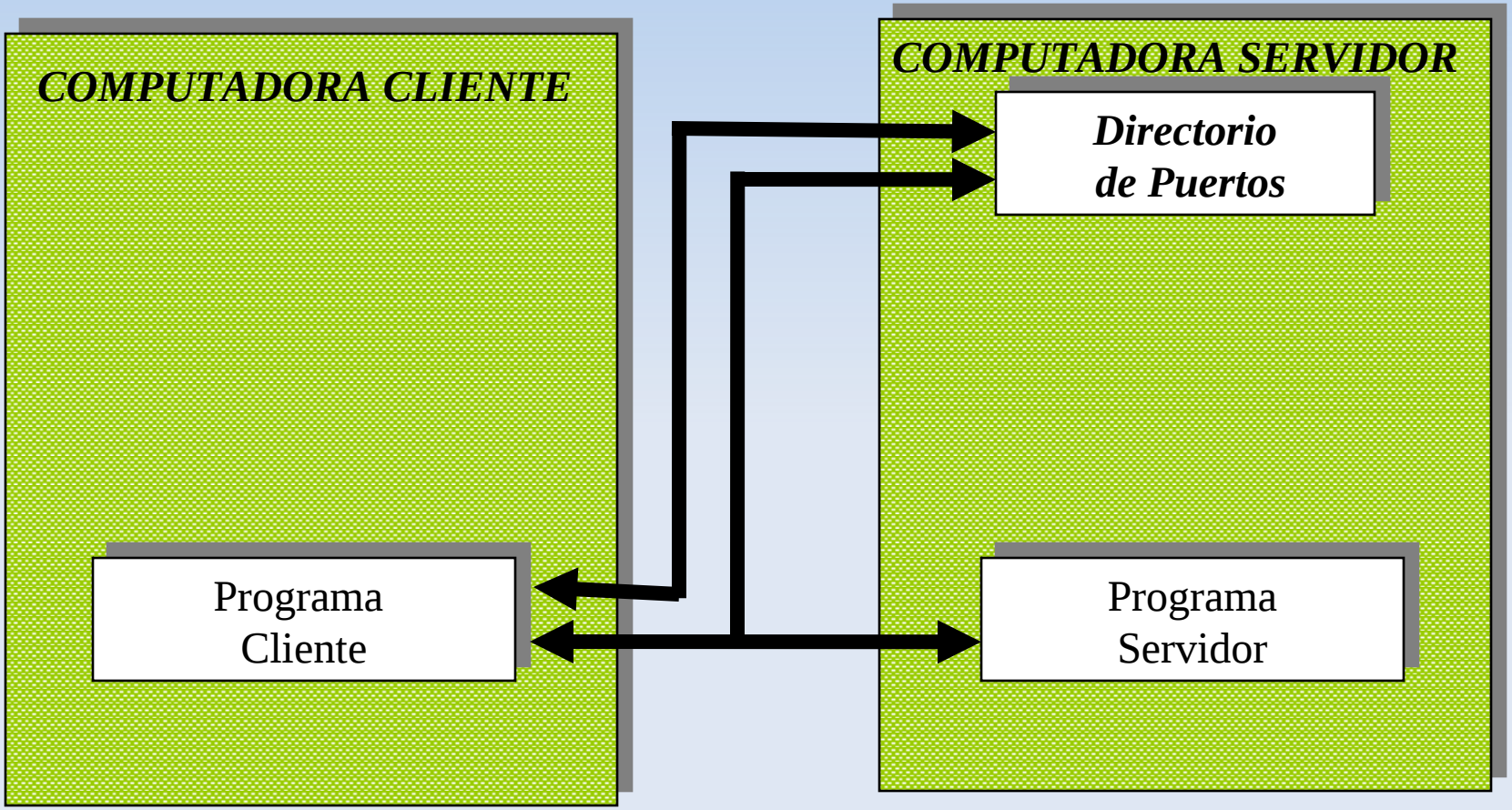
Los protocolos RPC:

- Constituyen una infraestructura de programación de sistemas cliente-servidor que permite desarrollar aplicaciones distribuidas sobre plataformas heterogéneas
- Permite hacer uso de procedimientos disponibles en diversas computadoras de la red
- Permite enviar y recibir estructuras de datos complicadas
- La llamada RPC es transparente para el invocador. El programa cliente ve la llamada remota, como una llamada local ordinaria
- Constituyen un mecanismo de comunicación eficiente y seguro para aplicaciones distribuidas

Hilo de Ejecución de una Llamada a Procedimiento Remoto

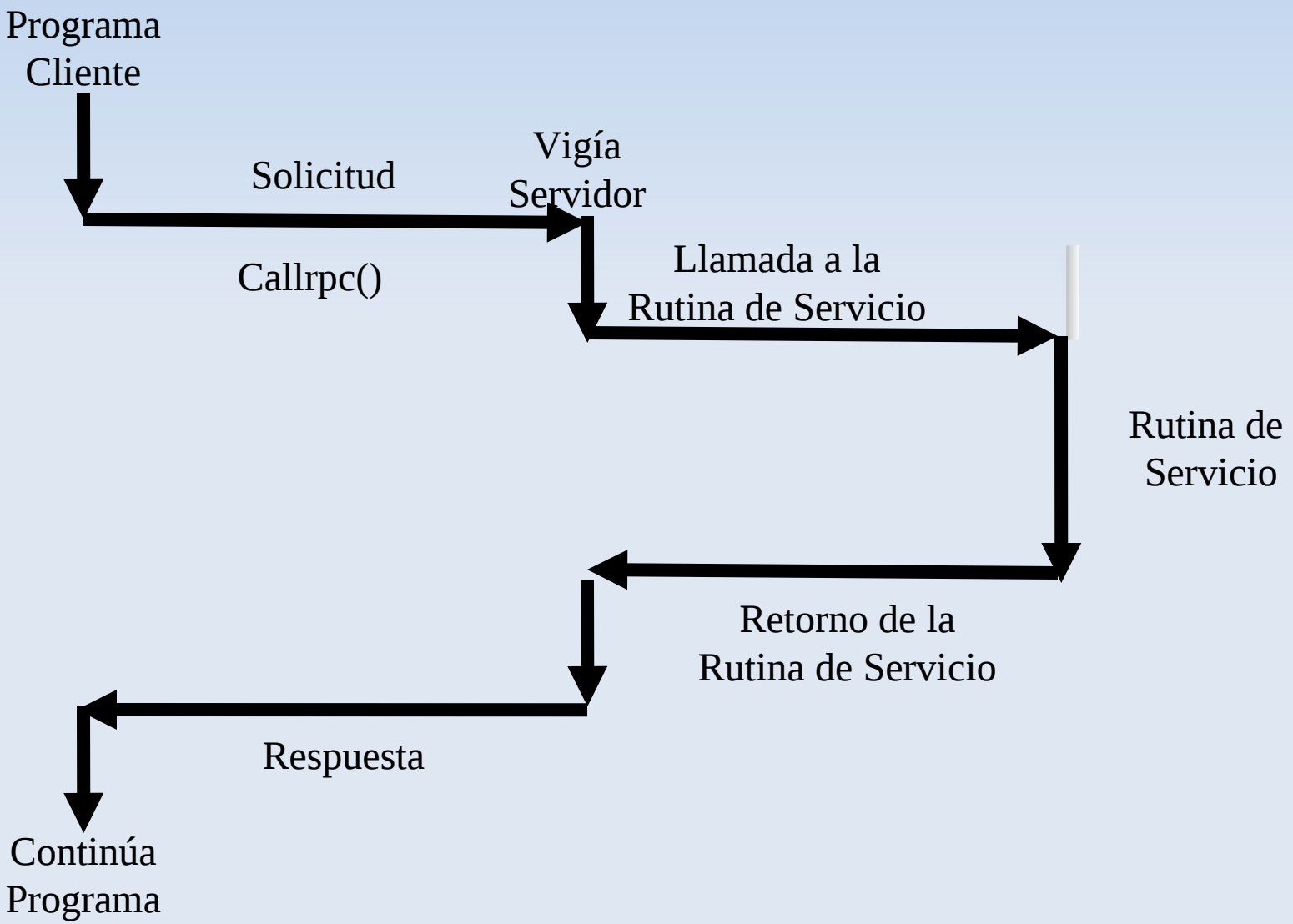


Protocolo de las Llamadas a Procedimientos Remotos



1. El servidor de RPCs establece la dirección en que escucha peticiones y registra el número de puerto en el Directorio de Puertos
2. El Cliente consulta el Directorio de Puertos de la computadora servidor e identifica el número de puerto para recibir peticiones
3. El cliente y el servidor abren una ruta de comunicación para realizar la ejecución de la llamada remota. El cliente realiza la petición y el servidor envía la respuesta

Hilos de Procesamiento de las Llamadas a Procedimientos Remotos



Paso de Mensajes

Los sistemas operativos actuales cuentan con recursos para que los procesos se intercomuniquen y colaboren.

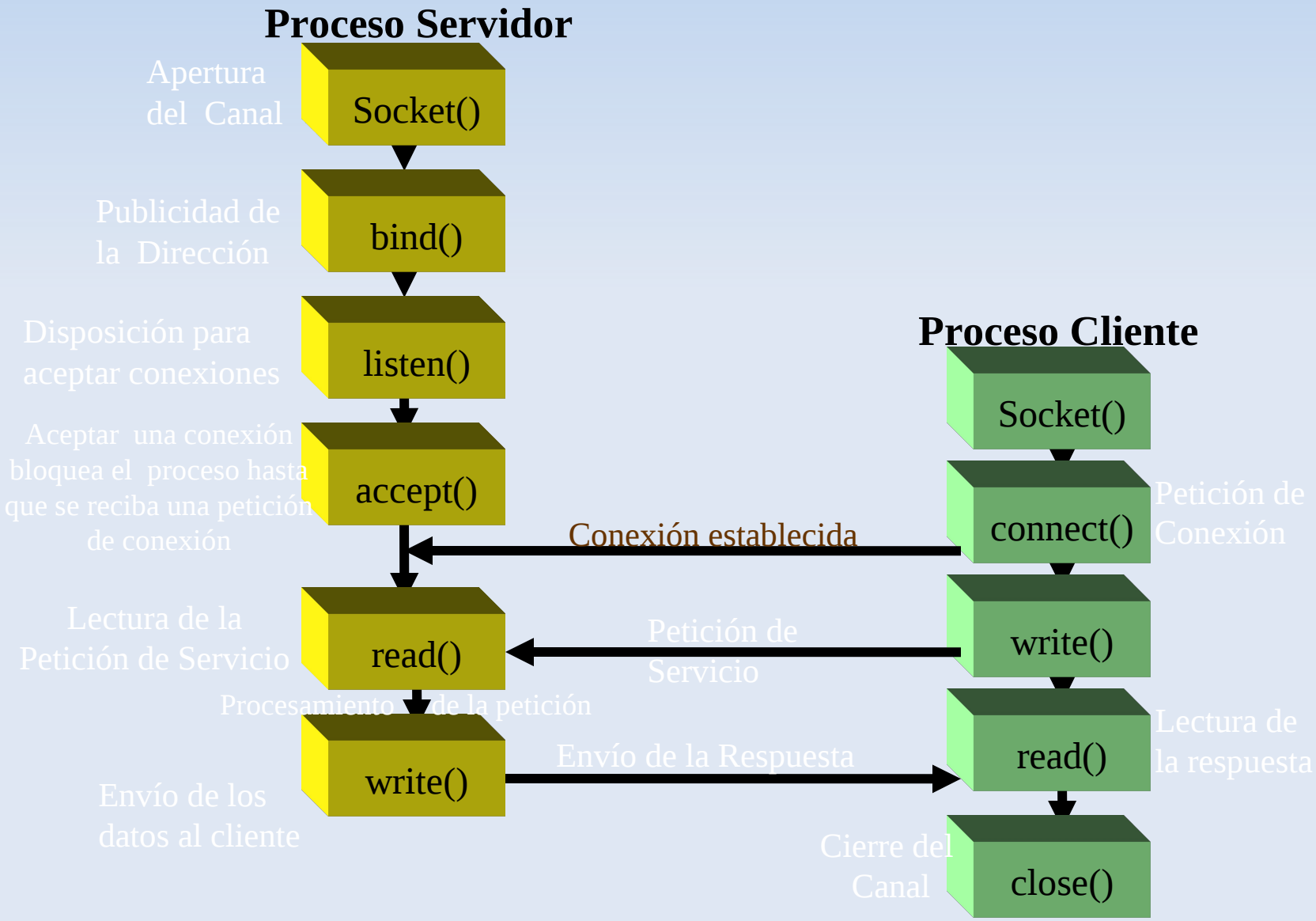
Intercomunicación entre Procesos (IPCs)

- Los IPCs son un conjunto de interfaces de software que permiten a un programador crear y manejar procesos individuales.
- Hacen posible la comunicación entre los procesos
- Incluyen los conceptos de:
  - Canales (“Pipes”) y Canales Nominales (“Named Pipes”)
  - Colas de Mensajes (“Queues”)
  - Semáforos
  - Memoria Compartida (“Shared Memory”)
  - Sockets

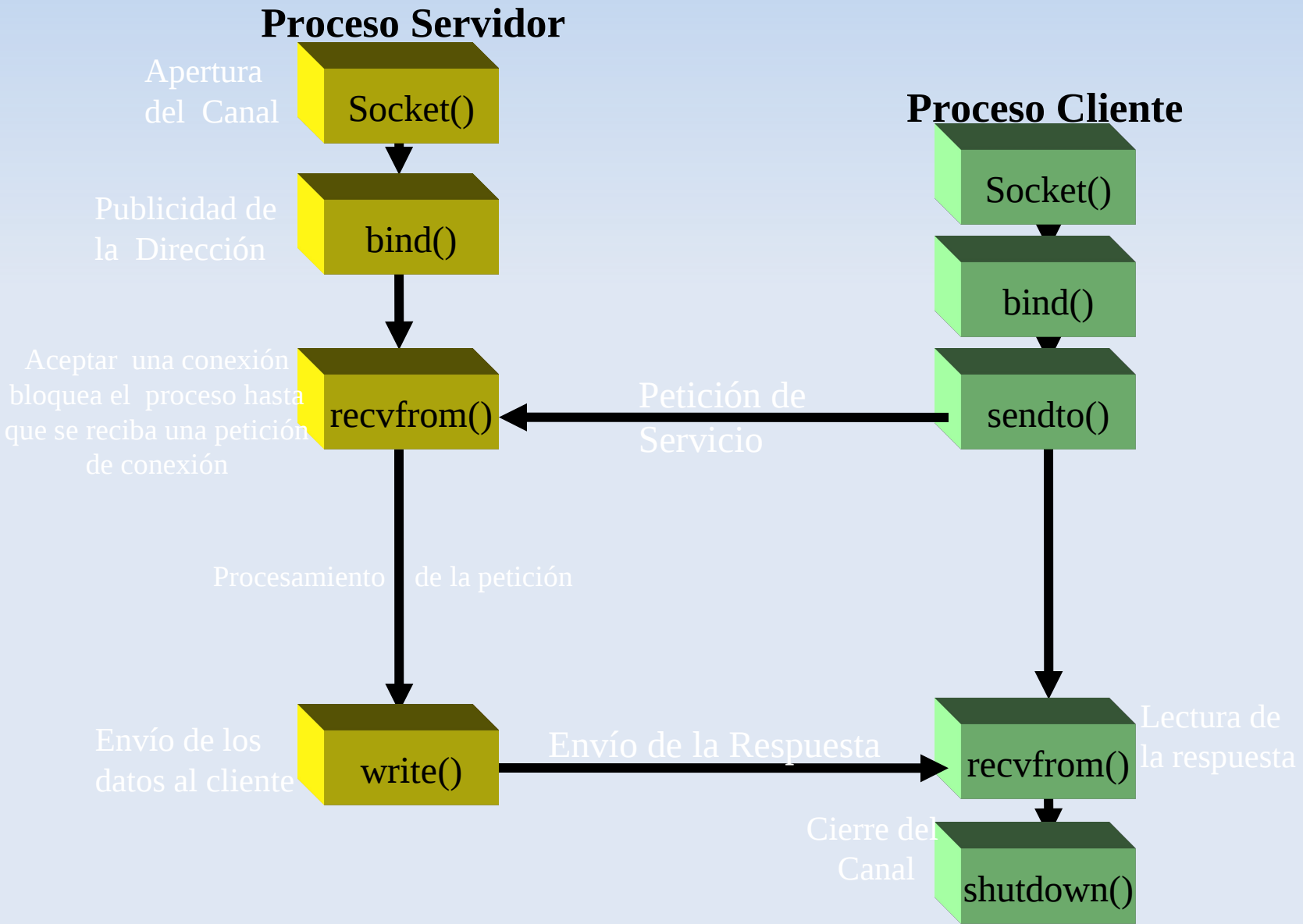
SOCKETS

Los sockets son los recursos que permiten la comunicación entre procesos corriendo en equipos diferentes, haciendo uso de los protocolos del nivel de transporte: TCP y UDP.

SECUENCIA DE LLAMADAS PARA UNA COMUNICACIÓN ORIENTADA A CONEXIÓN MEDIANTE SOCKETS



SECUENCIA DE LLAMADAS PARA UNA COMUNICACIÓN ORIENTADA A MENSAJES MEDIANTE SOCKETS



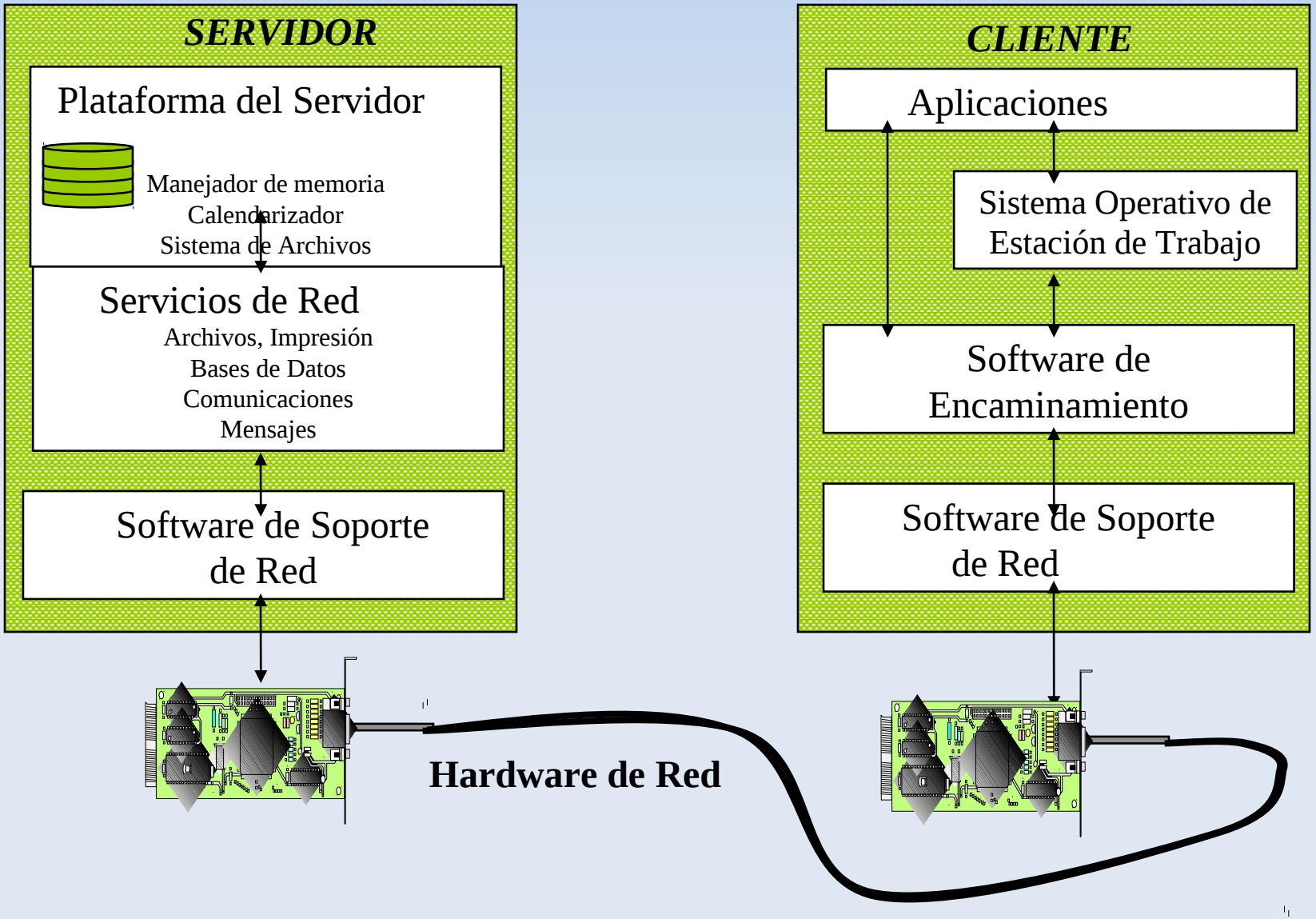
Modelo Cliente Servidor

- Fue introducido con los sistemas Unix
- El servidor proporciona un servicio de cómputo como:
  - Acceso a bases de datos
  - Impresión
  - Acceso a un canal de comunicación
  - Acceso a la memoria de trabajo de una máquina, etc.

Objetivos del Modelo Cliente-Servidor

- Localización transparente.
- Recursos compartidos.
- Escalabilidad
  - Horizontal: > nº de estaciones.
  - Vertical: migración a diversas plataformas.
- Interoperatividad entre distintos tipos de Hw y Sw.

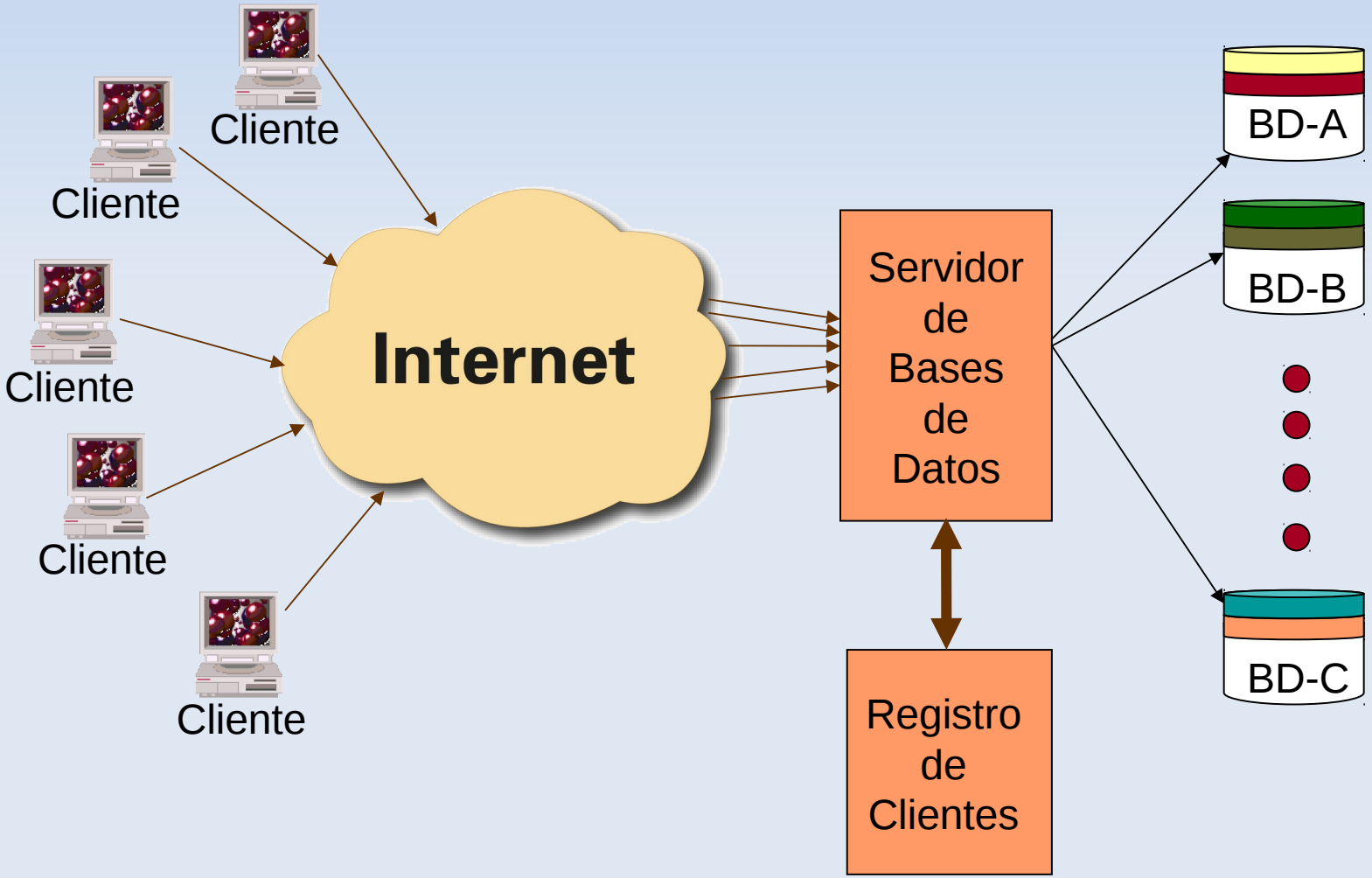
Arquitectura Cliente-Servidor



El modelo de cliente/servidor básico

- Una aplicación en un ambiente de ejecución cliente/servidor hace su consulta a una aplicación intermedia llamada "Servidor". A las aplicaciones de usuario se les denomina "Clientes".
- Las bases de datos residen típicamente en una computadora servidor a la que se tiene acceso a través de una red de área local. una intranet, o la Internet, permitiendo que las bases de datos puedan utilizarse desde localizaciones remotas.
- El servidor "sirve" los datos que pide el cliente y escribe la información pertinente en la base de datos, de acuerdo a la instrucción del cliente.
- El diseño cliente/servidor permite que múltiples instancias de una aplicación accedan a la misma base de datos, al mismo tiempo.
- La seguridad de red se mejora porque el acceso del usuario puede controlarse más estrictamente y las comunicaciones entre cliente y servidor pueden cifrarse.

El modelo de cliente/servidor básico



Características del Modelo Cliente-Servidor.

- Flexibilidad:
  - Middleware.
  - Separación de funciones:
    - Lógica de presentación.
    - Lógica de negocio.
    - Lógica de datos.
  - Encapsulación de servicios.
  - Portabilidad - reubicación.
  - Operación síncrono - asíncrono.

Características del Modelo Cliente-Servidor (II).

- Entorno de aplicaciones incremental.
  - Añadir un nuevo servidor.
  - Añadir un nuevo cliente.
  - Modificar un cliente para usar un nuevo servidor.
- Integración: por la GUI.

Ventajas de la Arquitectura Cliente-Servidor

- *La computación cliente-servidor permite a las organizaciones pasar de sistemas basados en maxicomputadoras y minicomputadoras a sistemas en servidores basados en redes locales y de área amplia*
- *La carga de trabajo impuesta por las aplicaciones se divide entre el sistema computacional, en donde se ubica el cliente y el servidor, cumpliendo así con el procesamiento distribuido*
- *Los servidores manejan tareas relacionadas con datos centralizados y con ello minimizan la cantidad de información que se cursa en la red, ya que sólo envían a los clientes los resultados requeridos*

Ventajas de la Arquitectura Cliente-Servidor

- *La información está más segura en una sola localización. Los almacenes de datos ponen a disposición de servidores de grupos de trabajo la información, mientras que mantienen el control y la seguridad de ella.*
- *Las aplicaciones que corren mejor en las estaciones de trabajo pueden cargarse y procesarse en ellas*
- *Facilitan el procesamiento paralelo*

Requisitos de la Arquitectura Cliente-Servidor

- ***Seguridad***: Debe controlarse el acceso a los datos
- ***Integridad***: Debe asegurarse que las transacciones se den por concluidas si están completas y se reprocesen si están incompletas.
- ***Concurrencia y Disponibilidad***: Es necesario asegurar que muchos usuarios tengan acceso y actualicen los datos
- ***Confiabilidad y Recuperabilidad***: Es necesario contar con respaldos y características de tolerancia a fallas

Evolución

- 1ª ÉPOCA:
  - LAN.
  - LAN con MAINFRAMES.
  - Comunicaciones homogéneas (LU, SNA, APPC).
- 2ª ÉPOCA:
  - Herramientas de desarrollo del Modelo Cliente-Servidor.
  - Proveedores DBMS con el Modelo Cliente-Servidor.
  - Downsizing: migración a PCs.
  - S.O. de red con servidores de servicios.

Evolución (II)

- 3ª ÉPOCA: ACTUAL.
  - PWS: Estaciones de trabajo programables gráficamente.
  - GUI: Interfaz gráfico de usuario. Alta resolución.
  - Nuevas tecnologías: Ratón, lápiz óptico, scanner, multimedia.
  - Tecnología de componentes: DDE y OLE.
  - Conectividad de BDs: ODBC, JDBC
  - Objetos Distribuidos: CORBA, COM, COM+, DCOM
  - Internet: HTML, CGI, Applet, ActiveX, JAVA, JAVASCRIPT
  - Arquitecturas C/S de 2 y 3 niveles.
  - Middleware.

Tecnología de componentes: DDE y OLE

- DDE: (Dynamic Data Exchange) (*Microsoft*).
  - Enlaces de datos dinámicos.
  - Información automáticamente actualizada entre aplicaciones.
- OLE: (Object Linking and Embeding) (*Microsoft*).
  - Objetos enlazados y embebidos.
  - Enlazado: Guardando una referencia.
  - Embebido: Insertando un documento.

Conectividad de BDs

- ODBC: (Open DataBase Conectivity) (*Microsoft*).
  - Conectividad abierta entre BDs.
  - Interfaz de conexión entre BDs (especialmente Microsoft)

- JDBC: (Java DataBase Conectivity) (*Java*).
  - Conectividad abierta entre BDs versión Java.
  - Abierto.

Objetos Distribuidos

- CORBA (Common Object Request Broker Architecture) (*Object Management Group*): Estándar de programación distribuida basada en objetos.
- COM (*Microsoft*): Interface estándar para objetos (no importa cómo están programados).
- COM+ (*Microsoft*): Extensión de COM en el que se añade un modelo para la programación de objetos.
- DCOM (*Microsoft*): Extensión de COM que permiten crear objetos clientes y servidores utilizando COM aunque creando transparencia sobre la localización física del objeto (es decir que puede encontrarse en otra máquina). La gestión de la comunicación está embebida.

INTERNET

- HTML (HyperText Markup Language): Lenguaje basado en el estándar de etiquetado SGML para la creación de páginas Web visibles desde un cliente remoto o navegador.
- CGI (Common Gateway Interface): Interfaz para el proceso de ejecutables en el servidor (remoto) a petición de clientes. Rápido y muy modular.
- ActiveX (*Microsoft*): Objetos visuales de control (desde botones hasta mini-aplicaciones) embebidos en un documento (o página Web) que se descargan y se ejecutan en el visor del cliente.
- JAVA (*Sun Microsystems*): Lenguaje de programación específico para Internet.
- APPLET: Objetos visuales embebidos en una página Web (versión abierta de ActiveX).
- JAVABEANS (*Sun Microsystems*): Especificación de objetos Java.
- JAVASCRIPT (*Netscape*): Lenguaje utilitario para HTML.

Evolución (III)

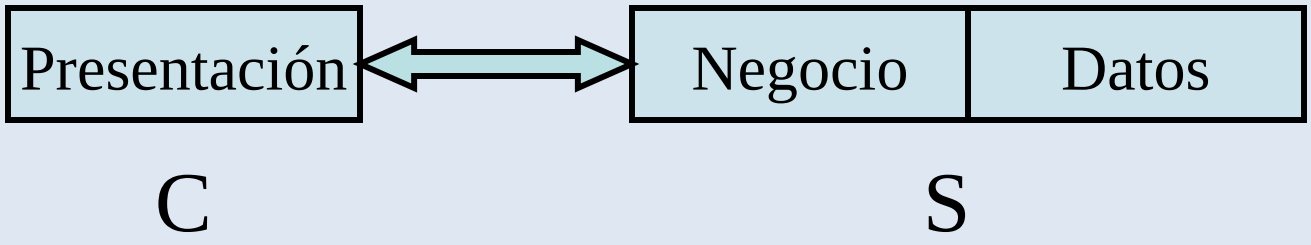
- EL FUTURO.
  - Facilidad de uso de las aplicaciones.
  - Accesos a datos distribuidos en cualquier lugar del mundo (y del espacio).

MIDDLEWARE

- Conecta procesos para constituir aplicaciones.
- Conjunto de funciones + servicios.
- Actúa en el nivel bajo del Sistema Informático Distribuido:
  - Comunicación.
  - Directorios.
  - Integridad.
- Define la plataforma de transparencia en la localización.

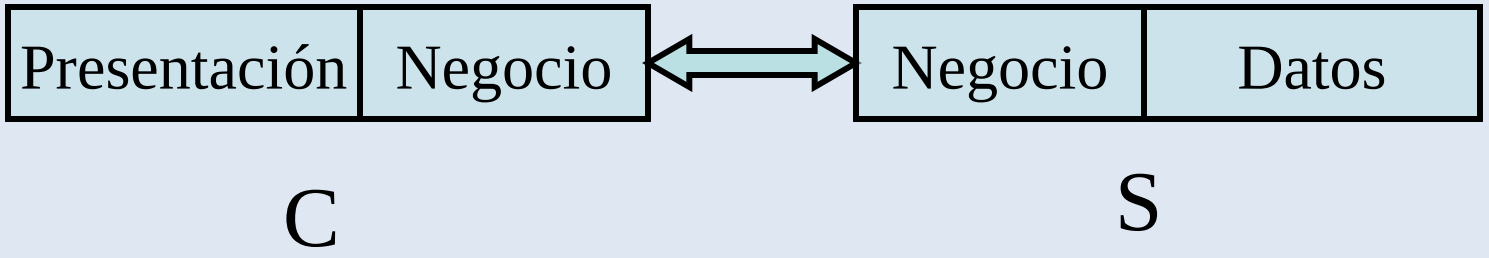
Modelos Cliente-Servidor

- Presentación distribuida
  - Proporciona un API que separa la programación de ventanas del resto.
  - Ejemplo: X-Windows System en UNIX o Windows95 y NT.



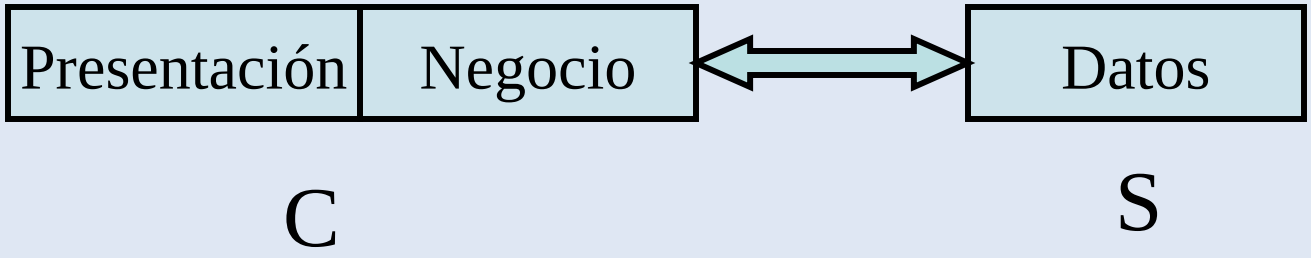
Modelos Cliente-Servidor(II)

- Función distribuida
  - Máxima flexibilidad.
  - Lógicas de negocio separadas.



Modelos Cliente-Servidor (III)

- Datos distribuidos
  - Archivos distribuidos.
  - Bases de datos distribuidas.



Aplicaciones de 2 niveles

- 2 niveles:
  - Generalmente usa los modelos de función distribuida o datos distribuidos.
  - Muy productivo.
  - Distribución no flexible.
  - Dependiente del suministrador.

Aplicaciones de 3 niveles

- 3 niveles:
  - Modelo presentación-negocio-datos
  - Distribución flexible.
  - Sistema abierto. No dependiente.