Sistemas Distribuidos (Arquitecturas)

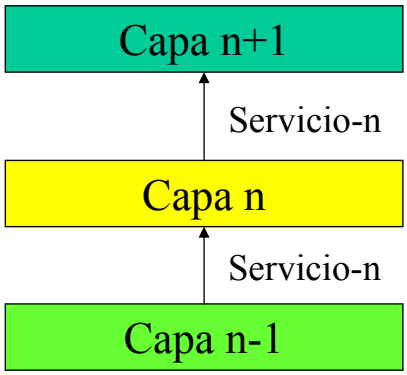
Dr. Víctor J. Sosa Sosa
vjsosa@cinvestav.mx

Arquitecturas

- **Los SD son los sistemas de software más complejos**
  - Nortel Networks crea switches los cuales pueden contener entre 25-30 millones de líneas de código, interviniendo 3000 desarrolladores de software, y con un ciclo de vida de 20 años para actualizar.
  - En Motorola, el 20% de sus ingenieros producen hardware, 80% produce software.
  - En este tipo de software hay materia para toda clase de problemas de ingeniería de software.
- **Investigación de arquitectura de software para tratar los retos de diseño**
  - "... Incluye la organización de un sistema como la composición de componentes; control global de estructuras; los protocolos para comunicación, sincronización, y acceso a datos; la asignación de funcionalidad para diseñar elementos; la composición de diseño de elementos; distribución física; escalamiento y desempeño; dimensiones de evolución; y selección de alternativas de diseño. Este es el nivel de diseño arquitectura de software." [Garlan y Shaw]
- **Algunos paradigmas de arquitecturas pertinentes para SD**
  - Capas
  - Cliente - Servidor

Capas

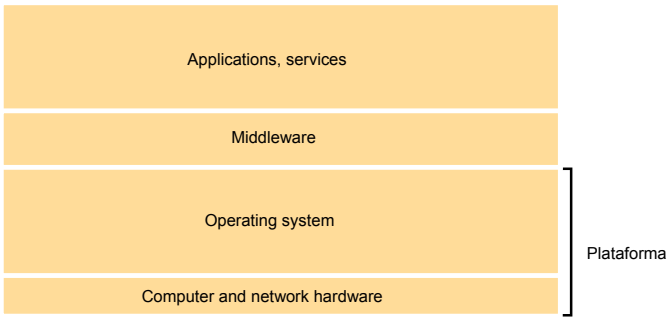
- **Idea básica**
  - Desmembrar la complejidad de sistemas mediante el diseño en capas y servicios
    - Capas: grupo de funcionalidades fuertemente relacionadas y altamente coherentes
    - Servicios: funcionalidades proporcionadas a capas superiores



  - Ejemplos de arquitecturas en capas
    - Sistemas operativos (kernel, otros servicios), históricamente: los sistemas operativos.
    - Arquitecturas de protocolos de red

Cliente-Servidor

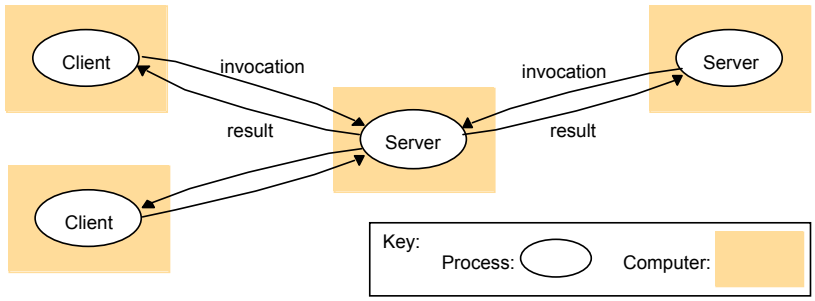
- **Estructura típica en capas de un SD**



  - **Plataforma**: Hardware y sistema operativo
    - Windows NT / Procesador Pentium
    - Solaris / Procesador SPARC
  - **Middleware**: logra trasparencia en la heterogeneidad en el nivel de plataforma
    - Logra comunicación y compartición de recursos
      - Ej. Invocación de métodos remotos
    - Ejemplos
      - CORBA (OMG), DCOM (Microsoft)
      - RM-ODP (ITU-T/ISO)
      - Invocación de Métodos Remotos Java (SUN)
      - Servicios Web
    - Nota: no todas las funciones relacionadas con comunicación puede ser abstractas.

Cliente-Servidor

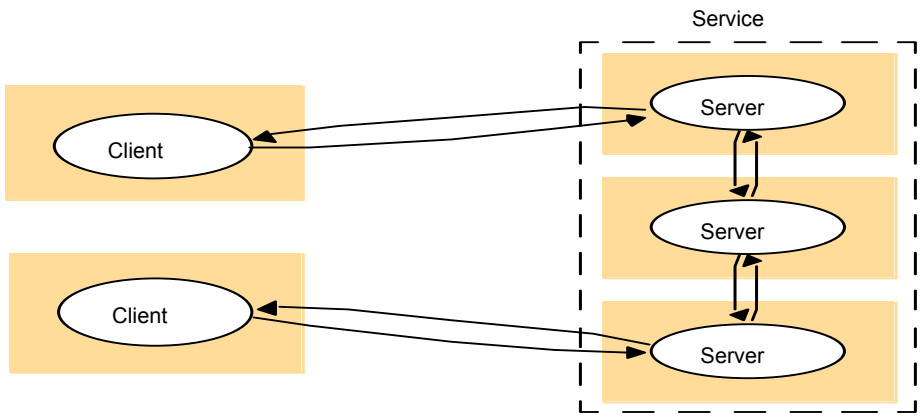
- **Modelo básico**



  - **Cliente**: el proceso requiere acceder datos, utilizar recursos o ejecutar operaciones en una computadora diferente
  - **Servidor**: Proceso maneja datos y otros recursos compartidos, permite al cliente acceder a recursos y ejecutar cómputos
  - **Interacción**: invocación / par de mensajes resultantes
  - Ejemplo
    - Servidor http: cliente (navegador) página solicitada, servidor entrega página
  - Servicios de caching (servidores proxy)
    - Caching de páginas Web frecuentemente utilizadas
  - Procesos pares (no cliente-servidor: peer-to-peer)
    - Procesos que tienen en gran parte similitudes de funcionalidad

Cliente-Servidor

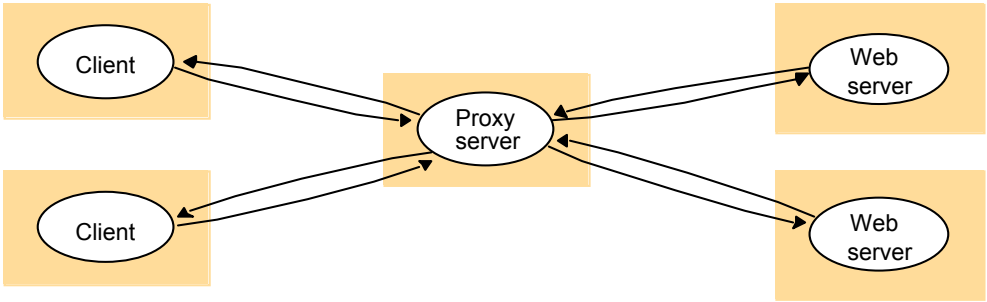
- **Variantes**
  - Servicios proporcionados por múltiples servidores



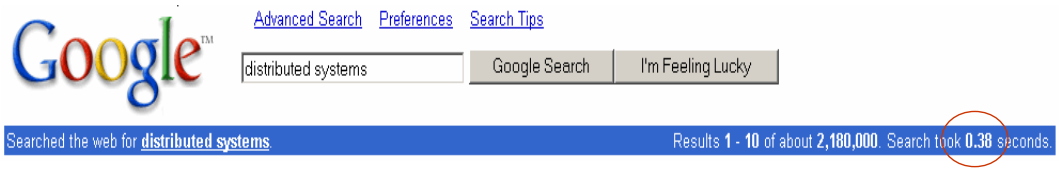
  - Ejemplos: muchos servicios de comercio Web están implementados en diferentes servidores
  - Motivación
    - Desempeño (ej. cnn.com, servidores para descarga, etc.)
    - Confiabilidad
  - Los servidores mantienen bases de datos replicadas o distribuidas

Cliente-Servidor

- **Variantes**
  - Servidores proxy: suministrar replicación/distribución trasparente



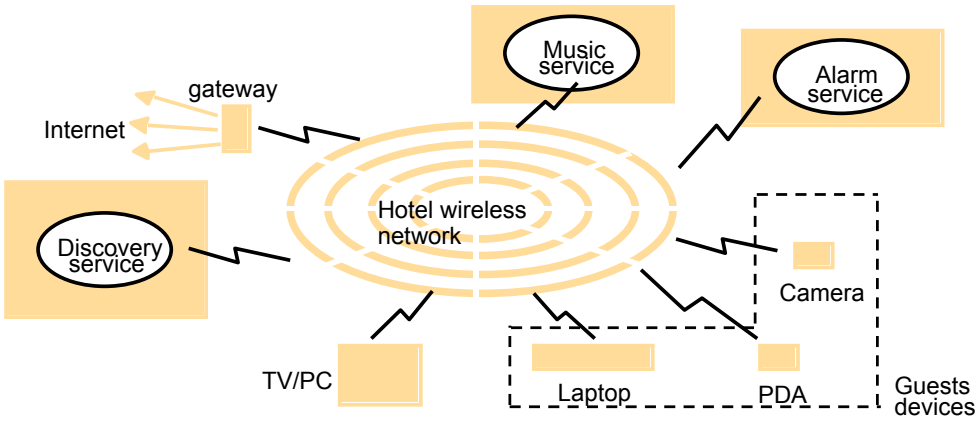
  - Caching
    - Los servidores proxy mantienen *caches*, como almacenes de recursos solicitados recientemente
    - Utilizados frecuentemente en motores de búsqueda:



Cliente-Servidor

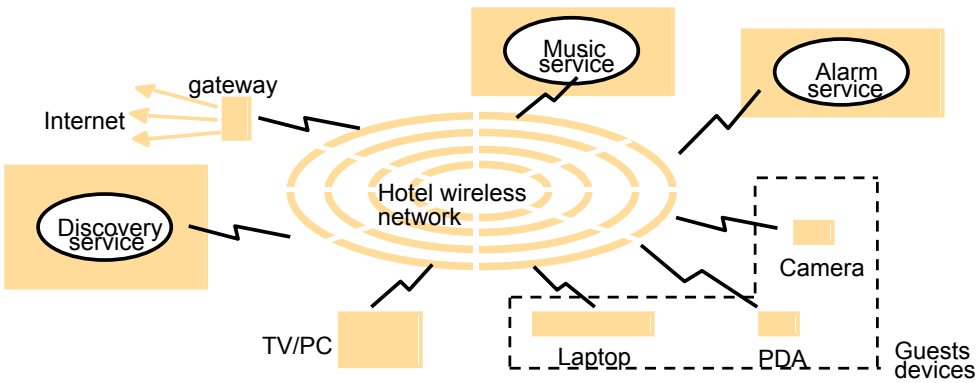
- **Más variantes de modelo Cliente- Servidor**
  - Código Móvil
    - Código enviado a un proceso cliente para realizar una tarea específica
    - Ejemplos
      - Applets
      - Mensajes Activos(contiene código de protocolo de comunicación)
- **Agentes Móviles**
  - Programa ejecutado (código + datos), migración entre procesos, realizando una tarea autónoma, frecuentemente en representación de otro proceso
  - ventajas: flexibilidad, ahorro en costo de comunicación
  - Merados virtuales, programas gusano
- **Clientes delgados**
  - Ejecutar interfaces de ventanas localmente mientras la aplicación se ejecuta en el servidor
  - ejemplo: servidores X11 (corren del lado de la aplicación cliente)
- **Dispositivos portátiles para cómputo móvil**
  - personal digital assistants (PDAs)
  - Como se conectan a internet
    - wireless LANs/ MANs
    - wireless Personal Area Networks

Cliente-Servidor



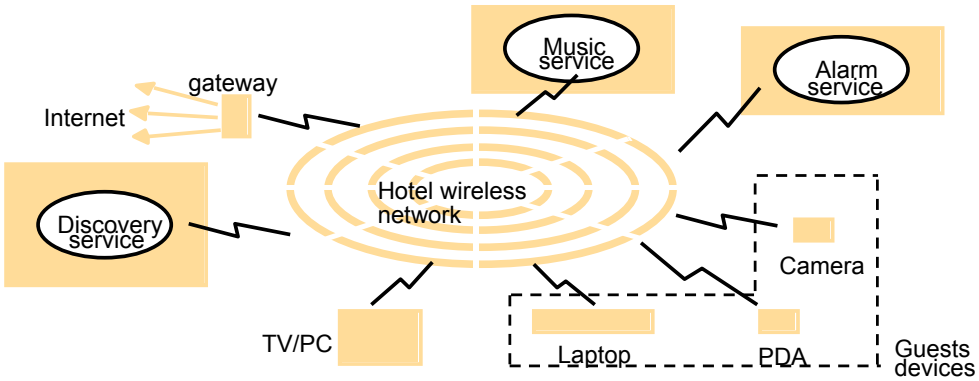
- **Más variantes del modelo Cliente- Servidor**
  - Gestión espontánea de red
    - Características
      - W-LAN se enfrentan a constantes cambios de dispositivos móviles heterogéneos
      - Dispositivos vagando en ambientes W-LAN heterogéneos
    - Beneficios
      - no se requiere conexión con cable
      - Fácil acceso a servicios disponibles localmente

Cliente-Servidor



- **Más variantes del modelo Cliente- Servidor**
  - Gestión espontánea de red
    - Retos
      - Soporte para conexiones convenientes e integración:
        - » Internet asume dispositivos con dirección IP en redes fijas
        - » Posible solución: asignación dinámica de direcciones IP
        - » Problemas: como encontrar dispositivos si estos son servidores
      - Conexión intermitente de dispositivos
      - Privacidad
      - Seguridad

Cliente-Servidor



- **Más variantes del modelo Cliente- Servidor**
  - Gestión espontánea de red
    - Descubrimiento de servicios
      - Servicios disponibles en la red
      - Sus propiedades, y como accederlos (incluyendo información específica de drivers)
    - Interfaces para descubrir servicios
      - Servicios de registro
        - » Acepta solicitudes de registro de servidores, almacena propiedades en BDs de servicios disponibles
    - Servicio lookup
      - Servicios de solicitudes equivalente con servidores disponibles

Cliente-Servidor

- **Interfaces**
  - Utilizar arquitecturas cliente-servidor impacta en el uso del software
    - ¿cuál es el mecanismos de sincronización entre cliente y servidor?
    - ¿tipos permitidos de solicitudes/respuestas?
- **Retos de diseño**
  - Calidad de Servicio
    - desempeño
      - Tiempos de respuesta
      - caudal
      - puntualidad

    ** dependen de la latencia de la red y tiempo de cómputo (incluyendo planificación) **
    - confiabilidad
    - adaptabilidad
  - Dependencia
    - Tolerancia a fallas: se espera que el sistema siga funcionando correctamente a pesar de presentarse fallas
    - seguridad

Fundamentos del Modelo de Interacción

- **Sistemas Distribuidos**
  - Procesos múltiples
  - Conectados mediante canales de comunicación
- **Algoritmos distribuidos**
  - Pasos a realizar por cada proceso
  - Comunicación entre procesos
    - sincronización
    - Flujo de información
- **Paradigmas generales para captar aspectos de comportamiento de un sistema distribuido basado en mensajes, algoritmos de ejecución**
  - Comunicando máquinas de estado finito extendidas [Brand y Zafiropoulo]
  - Autómatas de E/S [Lynch]

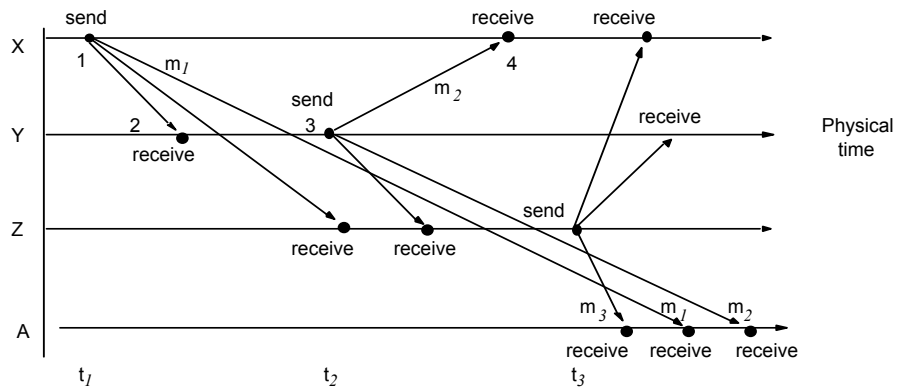
Fundamentos del Modelo de Interacción

- **Características de desempeño del canal de comunicación**
  - latencia: retardo entre el envío y recepción del mensaje
    - Tiempo de acceso a la red (ej., retardos de transmisión Ethernet)
    - Tiempo para que el primer bit viaje desde la interfaz de la red transmisora hasta la interfaz de red receptora
    - Tiempo procesado dentro del proceso de envío y recepción
  - caudal: número de unidades (ej., paquetes) entregadas por unidad de tiempo
  - Ancho de banda: cantidad de información (ej., bits) transmitida por unidad de tiempo
  - Variación de retardo: variación en retardos entre diferentes mensajes del mismo tipo (ej., cuadros de video en redes ATM)

Fundamentos del Modelo de Interacción

- **Sistemas Distribuidos Síncronos**
  - el tiempo para ejecutar cada paso de un proceso tiene establecidos limites inferiores y superiores
  - los tiempo de entrega de mensajes tienen limites establecidos
  - cada proceso tiene un reloj que deriva rangos en tiempo real con limites establecidos
- **Sistemas Distribuidos Asíncronos : sin límites**
  - Tiempos de ejecución de procesos
  - Tiempo de entrega de mensajes
  - Tasa de movimiento del reloj
- **Nota**
  - los sistemas distribuidos síncronos son fáciles de manejar, pero determinar limites realistas puede ser difícil o imposible
  - Los sistemas asíncronos son más abstractos y genéricos: un algoritmo distribuido ejecutado en un sistema es probable que también trabaje en otro

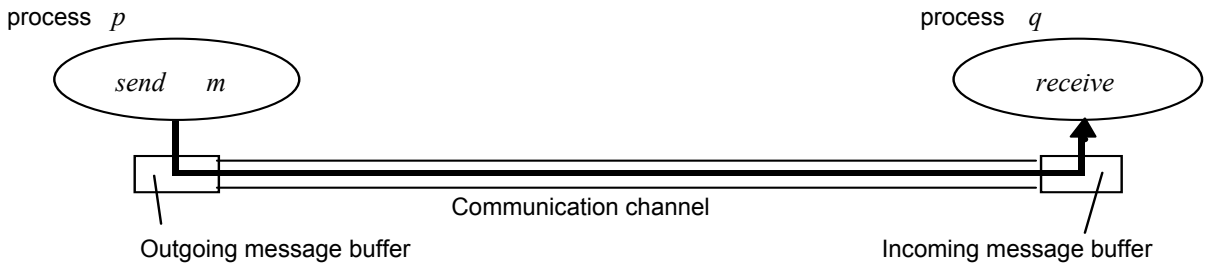
Fundamentos del Modelo de Interacción



- **Ordenamiento de eventos**
  - en un sistema distribuido es imposible que un proceso tenga una vista del estado global del sistema
  - posiblemente para registrar localmente información de tiempos,
  - reglas de ordenamiento de eventos
    - si *e1* y *e2* suceden en el mismo proceso, y *e2* sucede después de *e1*, entonces $e1 \rightarrow e2$
    - si *e1* es el emisor del mensaje *m* y *e2* es el receptor del mismo mensaje *m*, entonces $e1 \rightarrow e2$

    Por lo tanto, $\rightarrow$ describe una relación de ordenamiento parcial del conjunto de eventos en un sistema distribuido

Fallas

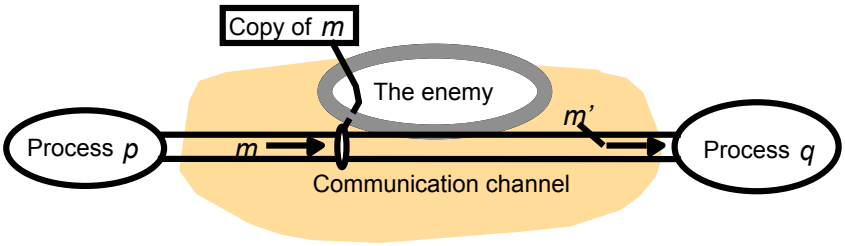


- **Fallas por Omisión**
  - Fallas por omisión de proceso: caída de proceso
    - detección con timeouts
    - la caída es del tipo fail-stop si otro proceso puede detectar con certeza que el proceso ha caído
  - Fallas por omisión de comunicación: el mensaje no ha sido entregado (pérdida de mensajes
    - Posibles causas:
      - error de trasmisión de red
      - Sobrecarga de buffer de recepción de mensajes
- **Fallas arbitrarias**
  - proceso: omite pasos esperados del proceso o lleva a cabo no deseados
  - Canal de comunicación: ej., sin entrega, corrupción o duplicidad

Fallas

Tipo de falla	*Efecto*	*Descripción*
Fail-stop	Proceso	El proceso para y permanece así. Otros procesos pueden detectar este estado
Crash	Proceso	El proceso para y permanece así. Otros procesos pueden no ser capaces de detectar este estado.
Omission	Canal	Un mensaje insertado en un buffer de mensajes de salida no llega al siguiente buffer de llegada de mensajes.
Send-omission	Proceso	Un proceso completa un envío, pero el mensaje no es puesto en su buffer de mensajes de salida.
Receive-omission	Proceso	Un mensaje es puesto en el buffer de mensajes de llegada de un proceso, pero éste no lo recibe
Arbitrary (Byzantine)	Proceso o canal	El proceso/canal muetra un comportamiento arbitrario: podría enviar/transmitir arbitrariamente mensajes en tiempos arbitrarios, comete omisiones; un proceso puede detenerse o tomar un paso incorrecto

Seguridad



- **Protección de acceso a objetos**
  - derechos de acceso
  - en sistemas cliente-servidor: involucra autentificación de clientes
- **Protección de procesos e interacciones**
  - amenazas a procesos: problemas de solicitudes / réplicas no autentificadas
  - amenazas a canales de comunicación: el enemigo puede copiar, alterar o introducir mensajes cuando estos viajan por la red
    - uso de canales “seguros”, basados en métodos criptográficos
- **Negación de servicio**
  - Generación de redes débiles o cargar al servidor para que los servicios estén por de facto no disponibles
  - “pings” para cnn.com
- **Código móvil**
  - requiere privilegios de ejecución en la máquina objetivo
  - el código puede ser malicioso (ej, correos gusano)