

I. Introducción a Sistemas Distribuidos

Motivación, conceptos generales y tendencias tecnológicas

Prof. Dr.-Ing. Raúl Monge Anwandter ♦ 2º semestre 2025

@ Prof. Raúl Monge - 2025



Organización del capítulo

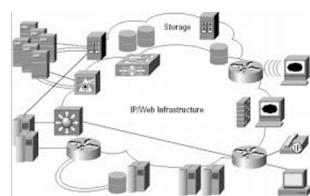
OBJETIVOS:

- Entregar una visión general sobre el área de sistemas distribuidos y sus tendencias.
- Definir conceptos clave como lenguaje básico para tratar los diversos temas.
- Mostrar marco de trabajo para el desarrollo de la asignatura.

CONTENIDOS:

1. Motivación y evolución de los sistemas distribuidos
2. Nociones básicas sobre arquitectura y diseño de sistemas distribuidos
3. Casos y tendencias en sistemas distribuidos

1.1 Motivación y evolución de los sistemas distribuidos



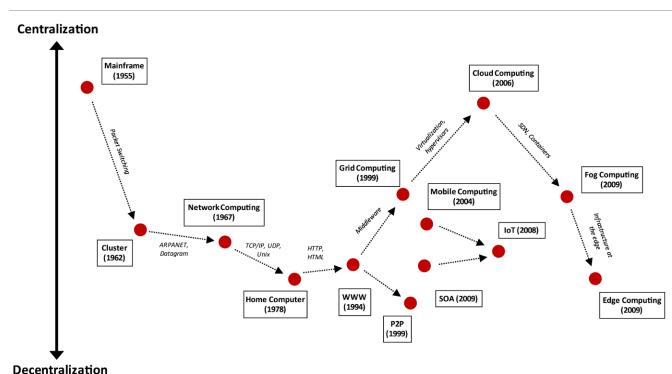
@ Prof. Raúl Monge - 2025

3

Evolución histórica de los Sistemas Distribuidos

Grandes innovaciones tecnológicas

- **1958**→ Redes de computadores (SAGE, teleprocesamiento, Arpanet, TCP/IP).
 - **1974**→ Computación en red (Ethernet, Unix BSD, arquitectura cliente-servidor, sistemas de archivos y DBMS distribuidos)
 - **1983**→ Telefonía móvil (iG, telefonía celular, Internet móvil, Smart phones).
 - **1991**→ Computación ubicua (IoT, redes de sensores)
 - **1993**→ Tecnología WWW (Aplicaciones Web, RR.SS., inteligencia colectiva, servicios Web, SOA y microservicios)
 - **1999**→ Grid computing (SETI, HPC) & P2P (Napster, Gnutella, etc.)
 - **2009**→ Cloud computing & Edge computing
 - **2009**→ Distributed ledger y Blockchain
 - **2017**→ Inteligencia artificial, ML, LLM, RAG



Fuente: Lindsay, D., Gill, S.S., Smirnova, D. et al. "The evolution of distributed computing systems: from fundamental to new frontiers". In *Computing*, Springer, Verlag, vol. 103, pp. 1859–1878 (2021).

Impulsores para el desarrollo de SD

Razones para distribuir sistemas de computación

1. **Alto rendimiento y escalabilidad.** Mayores capacidades de proceso, almacenamiento de datos y uso de ancho de banda. Soportar mayores cargas de trabajo.
2. **Alta disponibilidad y resiliencia.** Operación continua, fiable y tolerante a fallos.
3. **Compartición de recursos y colaboración.** Organizaciones/usuarios comparten mejor recursos y se facilita el trabajo colaborativo; también facilita integración de sistemas.
4. **Distribución geográfica.** Proveer mayor autonomía y menor latencia en ambientes físicamente distribuidos; también, cumplimiento de regulaciones.
5. **Inteligencia ambiental.** Para interactuar “inteligentemente” con el entorno, se requiere mayores capacidades para capturar, comunicar y procesar grandes volúmenes de datos.
6. **Necesidades económicas.** Reducción de costos en inversión y operación.

Perspectivas sobre sistemas distribuidos

“Dominio del Problema” versus “Dominio de la Solución”

Aplicación/negocio:

- Sistemas de comunicación y transferencia de documentos (electrónicos o digitales)
- Sistemas de información de gestión (integrados)
- Telemetría y sistemas de control (Automatización industrial, Smart cities, Agricultura de precisión, etc.)
- Computación de alto desempeño (HPC)
- Redes sociales e inteligencia colectiva
- Juegos multijugador masivos en línea (MMOG)
- Almacenamiento masivo de datos (*Big data*), analítica de datos e inteligencia de negocio (BI)
- Multimedios & *data streaming*
- Criptomonedas, contratos inteligentes

Tecnologías de información y comunicaciones (TIC):

- Redes de computadores y servicios de comunicación (e.g. ISO/OSI, TCP/IP)
- Sistemas cliente-servidor (e.g. sistemas de archivos distribuidos, DDBMS, etc.)
- Tecnología Web (e.g. WWW, aplicaciones Web, servicios Web)
- Computación móvil y comunicación inalámbrica
- Computación ubicua (e.g. IoT, RFID, redes de sensores)
- Cloud computing, Edge computing & Fog computing
- Libro mayor distribuido (DLT: Blockchain)

¿Qué es un sistema distribuido?

Definiciones

1. “Un sistema que consta de componentes de hardware y software localizados en una red de computadores que comunican y coordinan sus acciones sólo por medio de mensajes.
— [Coulouris & otros, 2011]
2. “Conjunto de computadores independientes que se muestra ante sus usuarios como un único sistema coherente.
— [Tanenbaum & van Steen, 2002]
3. “Sistema en el que la falla de un computador que ni siquiera sabías que existía, puede dejar inutilizable tu propio computador.”
— [Leslie Lamport, 1987]

Objetivo principal en un sistema distribuido:

Los procesos del sistema interactúan para coordinarse en el logro de un objetivo común.

¿Qué se distribuye?

Una perspectiva desde diferentes y múltiples dimensiones de distribución

1. **Procesamiento.** Tareas se dividen y distribuyen en múltiples nodos, para mejorar rendimiento y escalabilidad. Paralelismo acelera procesamiento de tareas, pero requiere coordinación. Además, replicación de procesos mejora resiliencia o tolerancia a fallos del sistema.
2. **Datos.** Se pueden particionar y distribuir en múltiples nodos de almacenamiento, para mejorar rendimiento, disponibilidad y resiliencia. Replicación de datos requiere mantener su consistencia.
3. **Control.** Descentralización provee mayor autonomía a los nodos para mejorar rendimiento, resiliencia y escalabilidad. Se requiere siempre de una coordinación global para alcanzar un objetivo común.
4. **Geográfica.** Nodos pueden estar dispersos geográficamente. Acercamiento de componentes al usuario puede mejorar calidad de servicio (menor latencia y mejor tiempo de respuesta).
5. **Gestión.** Nodos pueden estar bajo diferentes dominios administrativos (autoridad descentralizada). Se requiere resolver problemas de integración, interoperabilidad y gobernanza.

Modelo de Enslow: ¡La distribución para un sistema es relativa en cada dimensión!

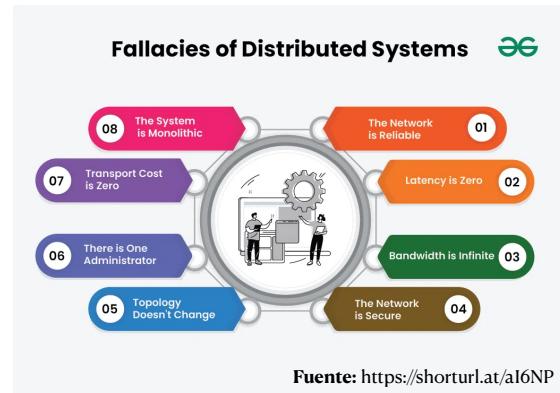
Fuente: P. H. Enslow. 1978. What is a "Distributed" Data Processing System? *IEEE Computer* 11, 1 (January 1978), 13–21.

Las falacias de la computación distribuida

La realidad es diferente al ideal (enfoque ingenuo)

- La red es fiable
- La latencia es cero
- El ancho de banda es infinito
- La red es segura
- La topología no cambia
- Existe un único administrador
- El costo de transporte es cero
- La red es homogénea

Fuente: Peter Deutsch, "Eight Fallacies of Distributed Computing", Sun Microsystems, 1994



Fuente: <https://shorturl.at/ai6NP>

@ Prof. Raúl Monge - 2025

Lectura complementaria: [van Steen & et al., 2023] sección 1.4: Pitfalls. (pp. 52-53)

Características principales de sistemas distribuidos

Propiedades fundamentales que dificultan su desarrollo

1. **Múltiples elementos de computación autónomos.** Multiplicidad de procesos introduce conurrencia y datos pueden quedar inconsistentes (requiere coordinación distribuida).
2. **Subred de comunicación compartida.** Comunicación (básica) por paso de mensajes entre componentes distribuidos (\nexists memoria compartida). Rendimiento y fiabilidad puede degradarse por efecto de la comunicación.
3. **Estado global del sistema.** Al estar distribuido, ningún componente/usuario tiene en un determinado momento conocimiento preciso sobre el estado real de todo el sistema.
4. **Tiempo global impreciso.** Existencia de múltiples relojes (físicos), sin sincronización perfecta. Por lo tanto, no existe un patrón de tiempo preciso y 100% confiable.
5. **Fallas independientes.** Aumento de puntos de falla en el sistema, no necesariamente correlacionados. En consecuencia, se producen fallas parciales, con pérdida (parcial) de estado e inconsistencias.
6. **Aumento de la superficie de ataque.** Las amenazas sobre la seguridad del sistema aumentan al existir más componentes, intermediarios y posibilidades de acceso al sistema.

@ Prof. Raúl Monge - 2025

Grandes desafíos de diseño

Propiedades emergentes ideales en el diseño de sistemas distribuidos

- **Resiliencia.** Garantizar el correcto y continuo funcionamiento del sistema ante posibles fallas, errores y activación de procedimientos de recuperación y/o reparación.
 - Se mide por la fiabilidad ($\Pr[\text{no haya fallado en } t]$) y disponibilidad ($\Pr[\text{esté funcionando correctamente en } t]$).
 - Estrategia es incrementar redundancia del sistema y tolerar fallos.
- **Escalamiento.** Mantener un alto rendimiento y una alta disponibilidad, al aumentar en el sistema la carga de trabajo y/o el número de usuarios.
 - Estrategia es incrementar recursos y distribuir carga.
- **Consistencia.** Mantener consistente el estado del sistema y de los datos, en situaciones de concurrencia, replicación y ocurrencia de fallas.
 - Dado que mantener consistencia es costoso, estrategia es relajar consistencia si es posible.
- **Seguridad.** Proteger al sistema y sus componentes ante posibles amenazas y responder correctamente a posibles ataques y daños.
 - Se debe garantizar Confidencialidad, Integridad y Disponibilidad (CIA) de la información, entre otros.

Arquitectura de software distribuido

Desafíos de diseño, operación y desarrollo

- **Integración.** Resolver problemas de interoperabilidad y heterogeneidad entre componentes y sistemas. Integración de sistemas legados es a veces necesaria.
- **Observabilidad.** En la operación de un sistema: administrar componentes y observar su comportamiento, para reaccionar rápidamente ante la ocurrencia de diferentes anomalías y/o incidentes.
- **Mantenibilidad.** Facilitar la evolución, reparación y adaptación a los cambios tecnológicos y operativos (Evitar que un sistema se convierta en software legado). Diseñar el sistema para que sea extensible, modificable y flexible.

OBSERVACIÓN: Este enfoque es central para metodologías de trabajo tipo DevOps y uso de *pipelining* de CI/CD (Integración continua / Entrega continua)

Modelos fundamentales y limitaciones

Perspectivas sobre el comportamiento esperado de un sistema distribuido

1. **Modelo de sincronismo.** Noción de tiempo en el sistema (Cap. III y IV).
 - Modelo asincrónico vs. sincrónico.
 - Relojes reales y lógicos, causalidad, concurrencia y ordenamiento de eventos.
2. **Modelo de fallas (*failures*).** Tipos de fallos aceptados (Cap. III y VI).
 - Tipos de fallas: procesos, canales de comunicación y medios de almacenamiento de datos.
 - Modelos de fallas: detención, caída (*crash*), omisión, temporales y arbitrarias (*bizantinas*).
3. **Modelo de consistencia.** Semántica de consistencia de estado y datos (Cap. IV, VII y VIII).
 - Estados globales consistentes.
 - Consistencia de réplicas de datos y *caching*.
 - Transacciones, concurrencia y atomicidad.
4. **Modelo de seguridad.** Riesgos de seguridad y contramedidas (fuera del ámbito de la asignatura).
 - Modelos de amenazas y de control de acceso.
 - Modelos de gestión de seguridad (políticas, controles, protecciones).

1.2 Nociones básicas sobre arquitectura y diseño de sistemas distribuidos



Arquitectura en Sistemas Distribuidos

Ejemplos de diferentes perspectivas

- **Arquitectura de computadores**
 - multiprocesador (memoria compartida) vs. multicamputador (memoria distribuida)
- **Arquitectura de redes de computadores**
 - Protocolos y servicios de red; arquitectura de capas o niveles (*multi-layer*)
- **Arquitectura de sistemas operativos**
 - Multiusuario y *multitasking*; estructura de núcleo (*kernel*) y procesos; *microkernel* y procesos distribuidos; virtualización.
- **Arquitecturas de servicios de computación distribuida**
 - cliente-servidor (asimétrico) vs. peer-to-peer (simétrico)
 - Estructura de procesos (e.g. *multi-threading*) y comunicación entre componentes (e.g. IPC, invocaciones, mensajería)
- **Arquitectura de aplicaciones distribuidas**
 - Sistemas operativos distribuidos, *Middleware* y Arquitecturas orientada a servicios (*multi-tiered*)
 - Virtualización, nube y contenedores; Microservicios, *cloud-native* y *serverless*.

Arquitecturas de computación paralela

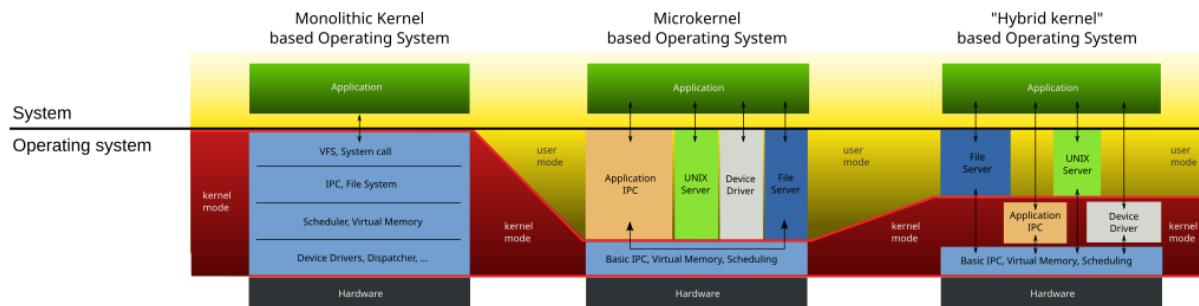
Perspectiva de Computación de alto rendimiento (HPC)

- **Sistema de multiprocesamiento con memoria compartida**
 - Mayor capacidad de procesos, pero memoria compartida es un cuello de botella (*caching* lo mitiga)
 - Ejemplo: SMP (UMA) y Multicore; programación con procesos concurrentes y *multi-threading*.
 - Arquitectura de GPUs (NVIDIA + CUDA) para HPC
- **Sistema multicamputador homogéneo**
 - Multiples procesadores con memoria distribuida, conectados en red
 - Topología de la red determina contención en uso de canales
 - Alto rendimiento, alta disponibilidad y mayor paralelismo (e.g. HPC)
 - Ejemplos: NUMA, Clusters (Beowulf); programación en MPI y OpenMP
- **Sistema Multicamputador heterogéneo**
 - Similar al anterior, pero heterogéneo y posiblemente más masivo y distribuido
 - Ejemplo: Grid Computing (HPC), Big data & data analytics (MapReduce)



Arquitectura del Sistema operativo

Kernel monolítico versus Microkernel



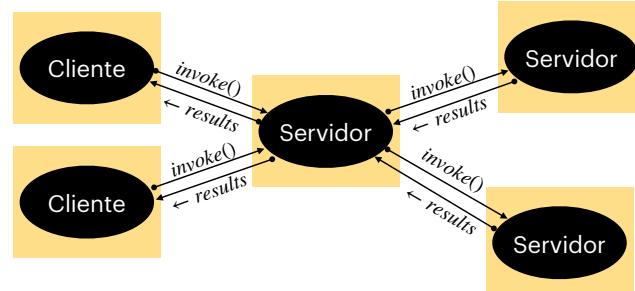
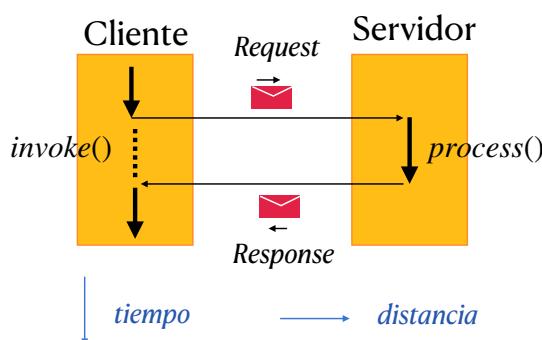
Fuente: Wikipedia (2008). <http://en.wikipedia.org/wiki/OS-structure.svg>

@ Prof. Raúl Monge - 2025

17

Modelo Cliente-Servidor

Modularización y distribución de servicios (acceso remoto)



Interfaz:

- Conjunto de operaciones
- Define un Tipo Abstracto de Datos (ADT)
- Extensible a Orientación a objetos

: proceso

: computador

Lectura complementaria: [van Steen & et al., 2023]
sección 2.3.1: Simple cliente-server architectures

@ Prof. Raúl Monge - 2025

18

Multihebras (*Multithreading*)

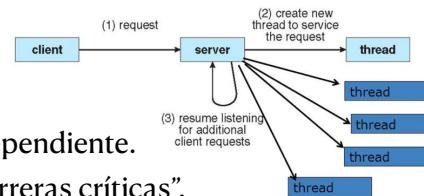
Estructura de proceso con memoria compartida

• Clientes Multihebra

- Permitir manejar en paralelo la comunicación remota, mientras se desarrollan otras actividades locales (soportar concurrencia sin bloquear al proceso).
- Crear múltiples conexiones para mejorar el ancho de banda.

• Servidores Multihebra

- Simplifica desarrollo de software que explota paralelismo.
- Cada petición de cliente se puede procesar por una hebra independiente.
- Estructura de proceso más liviana, pero requiere coordinar “carreras críticas”.



Referencias a Threads:

- Python: <https://docs.python.org/3/library/threading.html>
- Java: <https://docs.oracle.com/javase/tutorial/essential/concurrency/index.html>

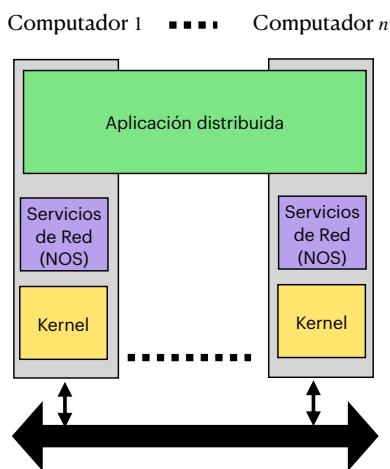
Lectura complementaria: [van Steen & et al., 2023] sección 3.1: *Threads*. (In: Chapter 3. Process)

@ Prof. Raúl Monge - 2025

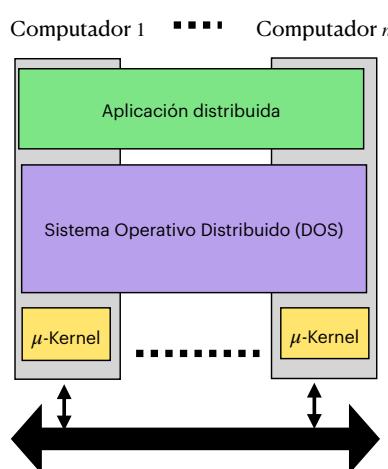
19

Sistemas Operativos en ambientes distribuidos

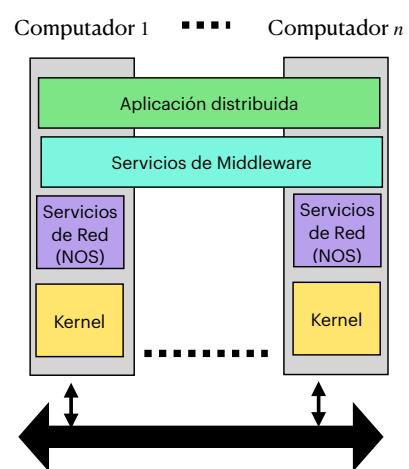
a) Sistema Operativo de Red



b) Sistema Operativo Distribuido



c) Middleware



@ Prof. Raúl Monge - 2025

20

Concepto de Middleware

Facilidad para distribuir aplicaciones

DEFINICIÓN: En un sistema distribuido, el *Middleware* es una capa de software que se encuentra entre el Sistema Operativo y las Aplicaciones en cada sitio del sistema.

Fuente: Sacha Krakowiak. 1988. *Principles of operating systems*. MIT Press, Cambridge, MA, USA.

CARACTERÍSTICAS CLAVE:

- Simplifica el desarrollo de aplicaciones distribuidas complejas, al proveer interoperabilidad basada en una arquitectura distribuida coherente y abierta.
- Facilita en entornos distribuidos la conexión de componentes de software o de personas con las aplicaciones.
- Provee un conjunto de servicios comunes para abstraer y proveer mayor valor a la interacción de procesos distribuidos.

Lectura complementaria: [van Steen & et al., 2023] sección 2.2: *Middleware and distributed systems*

@ Prof. Raúl Monge - 2025

21

Simetría en la Arquitectura de un Sistema Distribuido

Arquitectura C/S vs. Arquitectura P2P

- **Cliente-Servidor** (sistema asimétrico)
 - Estructura simple y muy popular.
 - Servidor controla acceso a recursos o servicios (control centralizado).
 - Facilita distribuir roles y responsabilidades.
- **Peer-to-Peer** (sistema simétrico)
 - Procesos tienen roles y privilegios similares.
 - Cooperación sin coordinación central (control distribuido o descentralizado).
 - Logra mejores tiempos de respuesta y puede ser más robusto.

Lectura complementaria: [van Steen & et al., 2023] sección 2.4: *Symmetrically distributed system architectures*

@ Prof. Raúl Monge - 2025

22

Compartición y gestión de recursos

Abstracciones para mejorar compartición y gestión de recursos

- **Sistemas operativos y ambientes de desarrollo**

- Sistemas operativos de red y distribuidos

- **Virtualización**

- Máquinas virtuales
- Contenedores
- Terminales virtuales (e.g. VDI)
- Redes virtuales (e.g. SDN)



- **Cloud Computing y Contenedores**

- Nuevo paradigma centralizado para compartir y gestionar recursos virtualizados.
- Gestión de *clusters* y orquestación de contenedores (e.g. Kubernetes)
- Servicios de plataforma para desarrollo y despliegue de aplicaciones (e.g. PaaS, Serverless, Cloud-native)

Lectura complementaria: [van Steen & et al., 2023] sección 3.2: Virtualization

Servicios de un Sistema Distribuido

Servicios compartidos comunes

- **Procesamiento.** Modelos de programación concurrente/paralelo para aprovechar altas capacidades de cómputo.
- **Almacenamiento de datos (masivo).** API estándar de acceso, persistencia de datos, alta disponibilidad, *caching*.
- **Comunicación.** Modelos de interacción, coordinación y transferencia de datos.
- **Tiempo.** Sincronización de relojes y marcas de tiempo para ordenar eventos y coordinar.
- **Identidad y descubrimiento.** Identificadores de recursos, servicios de nombre/directorio.
- **Seguridad.** Autenticación, control de acceso, firma digital y asegurar disponibilidad de recursos.
- **Gestión de operaciones.** Cambios de configuración, monitorización, *logging*, rendimiento, contabilidad, predicción/detección de anomalías, respuesta a incidentes, recuperación de servicios.

Metas de diseño de un Sistema distribuido

Requisitos no funcionales (NFR) deseables

- **Compartición.** Compartir recursos y servicios comunes para mejorar desempeño y cooperación.
- **Transparencia:** Ocultar complejidades internas del sistema para facilitar su uso.
- **Apertura (openess):** Facilitar integración de componentes y manejo de heterogeneidad.
- **Escalabilidad:** Soportar mayores cargas, sin degradar desempeño y disponibilidad.
- **Resiliencia:** Aumentar la fiabilidad del sistema. Resistir fallas y ataques; con tolerancia a fallos, capacidad de recuperación y mantención de consistencia.
- **Seguridad.** Proteger activos y reducir riesgos de amenazas/ataques.

Lectura complementaria: [van Steen & et al., 2023] sección 1.2: *Design goals*

a) META: Transparencia

Simplificar el manejo de la complejidad sistémica

- **Acceso.** Oculta diferencias en representación de datos y método de acceso a recursos u objetos remotos.
- **Ubicación.** Oculta dónde están localizados físicamente los objetos al accederlos.
- **Reubicación.** Oculta que un objeto pueda ser movido a otra ubicación por el sistema, mientras está siendo usado y sin afectar la operación.
- **Migración.** Oculta que un objeto se pueda mover a otra ubicación, como iniciativa del usuario.
- **Replicación.** Oculta que un objeto esté replicado, para mejorar la disponibilidad (confiabilidad) o el desempeño (balance de carga y tiempo de respuesta).
- **Concurrencia.** Oculta que un objeto pueda ser compartido entre varios usuarios independientes que compiten por su uso (sin interferencia y garantizando consistencia).
- **Fallas.** Oculta la ocurrencia de una falla y la recuperación de un objeto, para seguir operando y completar tareas.
- **Persistencia.** Oculta la desactivación y reactivación del objeto (para liberar recursos).
- etc.

Fuente original: ISO/IEC 10746: "RM-ODP (Reference Model of Open Distributed Processing)"
Lectura complementaria: [van Steen & et al., 2023] sección 1.2.2: *Distribution transparency*

b) META: Apertura (*openness*)

Facilidad de integración, interoperabilidad y portabilidad (con heterogeneidad)

- **Fuentes de heterogeneidad:**

- Computadores, sistemas operativos, lenguajes de programación, protocolos de comunicación, diferentes proveedores y vendedores.

- **Algunas estrategias:**

- Middleware (e.g. Corba, DCOM y RMI; SOA)
- Máquinas virtuales y código móvil interpretado (e.g. JVM, JS)

- **Preocupaciones mayores:**

- Estandarización de interfaces (API): componentes y conectores
- Extensiones, evolución e integración

Lectura complementaria: [van Steen & et al., 2023] sección 1.2.3: Openness

@ Prof. Raúl Monge - 2025

c) META: Escalabilidad

Capacidad para crecer y soportar mayores cargas con buen desempeño

- **Particionamiento (sharding) y replicación** (agregar más recursos para soportar mas carga)

- Estructuras de múltiples servidores cooperativos
- Distribución y balance de carga entre servidores

- **Acercamiento del sistema a los usuarios** (reducir latencia y tiempos de respuesta; distribuir carga)

- Uso de técnicas de *Proxies*, *Caching* y *Edge computing*.
- Movilidad de código, acercando procesamiento al usuario.

- **Explotación del asincronismo** (mejorar tiempo de respuesta y distribuir carga en el tiempo)

- Esconder latencias de comunicación, sin esperar respuesta inmediata.
- Procesamiento *batch*, colas de mensajes y *buffering*.

- **Reducción de sobrecarga de coordinación** (disminuir carga y mejorar desempeño)

- Uso de componentes sin estado.
- Relajar condiciones de sincronismo y consistencia, usando protocolos más livianos.

Lectura complementaria: [van Steen & et al., 2023] sección 1.2.6: Scalability

@ Prof. Raúl Monge - 2025

d) META: Resiliencia

Asegurar continuidad operacional en ambientes hostiles

- Tolerancia a fallos y redundancia (replicación)
- Modelos de fallas
- Atomicidad y enfoques de recuperación de errores.
- Detección de fallas y recuperación de errores

Lectura complementaria: [van Steen & et al., 2023] sección 1.2.5: *Dependability*

Profundización: Capítulo VI “Confiabilidad en Sistemas Distribuidos”

e) META: Seguridad

Tema transversal que incluye Seguridad de Información y Ciberseguridad

- Confidencialidad, Integridad y Disponibilidad (CIA)
- Autenticación de entidades y mensajes/documentos
- Control de acceso, autorización y auditoría
- No repudio y firma digital
- Arquitectura de seguridad y protección de activos
- Detección de anomalías e intrusiones
- etc.

OBSERVACIÓN: Esta asignatura no profundizará en estos temas.

1.3 Casos y tendencias en sistemas distribuidos



@ Prof. Raúl Monge - 2025

31

a) Tecnología Web en Sistemas Distribuidos

Arquitectura Cliente-servidor abierta

Características:

- Arquitectura abierta implementada sobre Internet creada por Tim Berners-Lee (\approx 1990) para el CERN (Suiza).
- Su fuerte desarrollo comienza en 1994 con la creación de W3C (<https://www.w3.org/>).
- Modelo de recursos compartidos usando URL/URI como referencia y protocolo HTTP(S) para conexión y transporte (seguro) de datos.
- Integración con aplicaciones (*backend*) para generación de páginas dinámicas e interactivas (*front-end*).
- Provee acceso universal a aplicaciones usando sólo el navegador (*browser*) y HTML como lenguaje de presentación.

Desafíos relevantes:

- Desarrollo e integración de sistemas y servicios (e.g. Frameworks y lenguajes, conectores y protocolos)
- Usabilidad, latencia y capacidad de respuestas a usuarios.
- Representación estándar de datos (XML), para intercambio y búsqueda de información.
- Escalabilidad, resiliencia y consistencia (e.g. *caching*, replicación y *clustering*)
- Seguridad (e.g. DDoS, inyección de código, conexiones seguras y movilidad de código).



@ Prof. Raúl Monge - 2025

32

b) Cloud Computing

Emerge a fines de la década de los 2000

DEFINICIÓN: Modelo de acceso cómodo y bajo demanda a un conjunto compartido de recursos computacionales configurables (e.g. redes, servidores, almacenamiento, aplicaciones y servicios), que puede ser rápidamente aprovisionados y liberados con un mínimo esfuerzo de gestión o interacción con el proveedor de servicios.

Fuente: "The NIST Definition of Cloud Computing", NIST SP 800-145

CARACTERÍSTICAS:

- Auto-servicio por demanda (consumidor se aprovisiona)
- Acceso a banda ancha (capacidades disponibles por la red)
- Recursos compartidos (entre múltiples consumidores)
- Elasticidad rápida (capacidades aprovisionadas elásticamente)
- Servicios medidos (se paga por lo consumido)



LÍDERES DE LA INDUSTRIA MUNDIAL:

- AWS de Amazon, Google Cloud Platform (GCP), Microsoft Azure (3 grandes)
- IBM Cloud, Oracle Cloud Infrastructure (OCI), Alibaba Cloud, Rackspace, etc.

Modelos de Cloud Computing

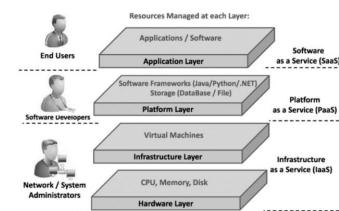
Provisión de servicios y despliegue

MODELO DE SERVICIO (*layers*)

- **Hardware (Data center)**
- **IaaS (Infraestructura)**
 - Virtualización de recursos (máquinas, almacenamiento y redes virtuales)
- **PaaS (Plataforma)**
 - Herramientas de desarrollo, despliegue, integración continua y gestión de aplicaciones.
- **SaaS (Software)**
 - Aplicaciones para el usuario final.

MODELO DE DESPLIEGUE

- Cloud público
- Cloud privado
- Cloud híbrido
- Cloud comunitario



Lectura complementaria: van Steen (2023). Sección 2.5.1 pp. 98-103.

Aplicaciones Cloud nativas

Aplicaciones desarrolladas específicamente para la nube

CARACTERÍSTICAS:

- Descomposición en microservicios (acoplamiento débil y desarrollo independiente)
- Servicios empaquetados en contenedores (e.g. Docker)
- Orquestación con despliegue, escalado y gestión automática (e.g. Kubernetes)
- DevOps & CI/CD (*pipeline* para desarrollo y despliegue rápido y mejora continua)
- Escalabilidad elástica (según demanda)

BENEFICIOS:

- Resiliencia (tolerancia a fallos y sistemas de auto-sanado / *self-healing*)
- Portabilidad (consistencia de ejecución entre nubes públicas, privadas o híbridas)
- Eficiencia (optimiza el uso de recursos)
- Velocidad (ciclos de desarrollo y despliegue más rápidos)

Referencia: Proyecto “Cloud Native Computing Foundation” (CNCF) de Linux Foundation.
<https://www.cncf.io/>

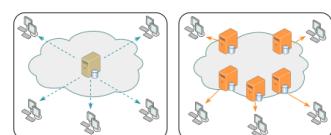
c) Content Delivery Networks (CDN)

Optimización de entrega de contenidos principalmente en la Web

DEFINICIÓN: Red de servidores distribuida geográficamente, diseñada para entregar eficientemente contenidos (e.g. páginas web, videos, imágenes), reduciendo latencia y tiempos de carga.

Mecanismos principales:

- *Caching* (guardar copias cerca del usuario)
- Enrutamiento de solicitudes de usuarios a servidor CDN más cercano



Beneficios:

- Reducir latencia y uso de ancho de banda
- Mejorar escalamiento y disponibilidad.

Desafíos:

- Consistencia de *cache*
- Seguridad (e.g. DDoS)
- Costos de despliegue

d) Computación ubicua

Se relaciona con Pervasive Computing & IoT

DEFINICIÓN: Paradigma que integra capacidades computacionales en objetos y el entorno natural, para proveer servicios e interactuar con ellos (*Context-Awareness*).

- Involucra integración masiva de objetos con limitaciones tales como: bajas capacidades computacionales, energía, inestabilidad de comunicación, seguridad, fallas, etc.
- Se relaciona con *Big data* e IA , al concentrar y procesar (inteligentemente) grandes volúmenes de datos.

Tecnologías habilitantes:

- Comunicación inalámbrica: Bluetooth, Wi-Fi, RFID, Zigbee y 5G.
- Sistemas embebidos (embedded): Microcontroladores, sensores y actuadores.
- IoT (Internet de las cosas): Objetos interconectados usando Internet.
- Middleware. Para integrar componentes y desarrollar aplicaciones distribuidas.
- Edge y Fog Computing (mas reciente): Extensión de las capacidades de la nube al borde de la red.

Referencia: Mark Weiser (1999). "The computer for the 21st century". *SIGMOBILE*, Rev. 3, 3 (July 1999), pp. 3-11.

Lectura complementaria: van Steen (2023). Sección 1.3.3 pp. 43-53.

e) Edge Computing & Fog Computing

Se complementa con computación ubicua

DEFINICIONES:

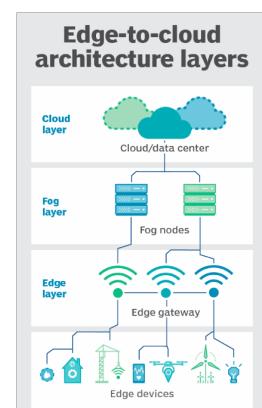
- **Edge Computing**: El procesamiento de datos se produce directamente en los dispositivos (e.g. sensores, dispositivos IoT) o cerca donde se generan, lo que reduce la necesidad de enviar datos "crudos" a una nube central.
- **Fog Computing**: Amplía modelo anterior, añadiendo una capa de dispositivos intermediarios, para un procesamiento y coordinación más complejos, pero más eficientes.

BENEFICIOS (similar a CDN):

- Reduce latencia y uso de ancho de banda
- Mejora escalabilidad y resiliencia
- Descentraliza la gestión

APLICACIONES:

- IoT, redes de sensores y CDN.



Lectura complementaria: van Steen (2023). Sección 2.5.2 pp. 98-99

f) Streaming de multimedios

Entrega de contenidos multimediales en tiempo real por Internet

APLICACIONES

- **Categorías:**

- Video por demanda (e.g. Netflix, Youtube)
- Videoconferencias (e.g. Zoom, Google Meet, Microsoft Teams)
- *Streaming en vivo (live)*

- **Desafíos:**

- Variabilidad del ancho de banda
- Latencia (minimizarla en casos interactivos)
- Escalabilidad (Nº de usuarios concurrentes)
- Calidad de servicio (reproducción fluida)

TECNOLOGÍA

- **Protocolos de streaming:**

- Real-Time Messaging Protocol (RTMP) & WebRTC
- HTTP Live Streaming (HLS), Dynamic Adaptive Streaming over HTTP (DASH)

- **Content Delivery Networks (CDNs):** entrega eficiente de contenidos

- **Servidores de medios:** Almacenamiento; manejo de codificación, QoS, etc.

- **Clientes** (diferentes tipos de dispositivos, según categoría de aplicación)

Referencia: Jeng-Neng Hang (2009). "Multimedia Networking From Theory to Practice", Cambridge University Press.

g) Distributed ledger (DLT) & Blockchain

- Tiene su origen en Bitcoin (criptomoneda), basado en una cadena de bloques con registro inmutable de transacciones (S. Nakamoto, 2009).
- Blockchain es un tipo de tecnología “Distributed ledger” (DLT), que domina al menos el 80-90% del mercado DLT.
- No existe un administrador central; en consecuencia, no tiene un punto de falla único.
- Requiere de una red P2P y algoritmos de consenso para poder replicar el registro (*ledger*) en múltiples nodos de la red (con una política de recompensa en el minado).
- Se usa criptografía para la seguridad de los datos y privacidad de usuarios (claves, firmas y *hashes*).



Referencia: Satoshi Nakamoto (2009). "Bitcoin: A Peer-to-Peer Electronic Cash System".

Lectura complementaria: M. van Steen (2023). Sección 2.5.3 pp. 104-108.

1.4 Conclusiones

Resumen de temas cubiertos

- Revisión breve de evolución e impulsores en el desarrollo de Sistemas Distribuidos.
- Caracterización de Sistemas distribuidos. Definición de “Sistemas Distribuido”; problema y limitaciones (e.g. falacias de P. Deutsch); modelos fundamentales (sincronismo, fallas, consistencia y seguridad).
- Conceptos sobre arquitecturas de sistemas (C-S, Threads, Middleware, servicios); propiedades emergentes/deseables (metas) y estrategias de implementación (transparencia, apertura, escalabilidad, resiliencia, seguridad)
- Revisión de algunas aplicaciones, arquitecturas y tecnologías específicos: Tecnología web, Cloud computing, CDN, Computación ubicua, Edge & Computing, Streaming de multimedios, DLT & Blockchain.

Material de estudio complementario del capítulo

- van Steen, et al. (2023). Texto guía.
 - Cap I. Introduction (pp. 1-54)
 - Cap II: Architecture: Sección 2.5 “Hybrid System architectures” (pp. 98-109)
 - Cap III. Processes (pp. 111-179)
- Coulouris, et al. (2012).
 - Cap. I: Characterization of Distributed Systems (pp. 1-34)
 - Cap II: Systems Models (pp. 37-77)

Sinopsis de la asignatura

- **Capítulo II: Arquitectura en Sistemas distribuidos (3).** Estilos y patrones. Arquitectura de servicios. Arquitectura basada en eventos. Comunicación y descubrimiento de componentes.
- **Capítulo III: Comunicación en Sistemas distribuidos (4).** Paradigmas. TCP/IP y sockets. Intercambio y representación de datos. Invocaciones remotas. Servicios Web. Sistemas de mensajería.
- **Capítulo IV: Fundamentos teóricos de computación distribuida (4).** Modelos de sincronismo. Sincronización de relojes. Reloj lógicos y vectoriales. Estados globales y consistencia.
- **Capítulo V: Algoritmos distribuidos de coordinación (4).** Elección de líder y exclusión mutua. Otros algoritmos básicos.
- **Capítulo VI: Confiabilidad en Sistemas distribuidos (4).** Tolerancia a fallos. Fiabilidad y disponibilidad. Consenso distribuido. *Checkpointing* y *rollback*. Replicación y *multicasting*.
- **Capítulo VII: Replicación de datos (3).** Modelos de replicación. Teorema CAP. Semántica de consistencia. Protocolos de consistencia. Consistencia eventual. *Caching* y distribución geográfica.
- **Capítulo VIII: Distribución de datos y transacciones (3).** Modelos y propiedad ACID. Control de concurrencia. Recuperación errores. Protocolos de compromiso.

I. Introducción a Sistemas Distribuidos

Motivación, conceptos generales y tendencias tecnológicas

Prof. Dr.-Ing. Raúl Monge Anwandter ♦ 2º semestre 2025

@ Prof. Raúl Monge - 2025

