Capítulo 1

Introducción a las Redes de Dispositivos

Parte 2

Metas de la introducción

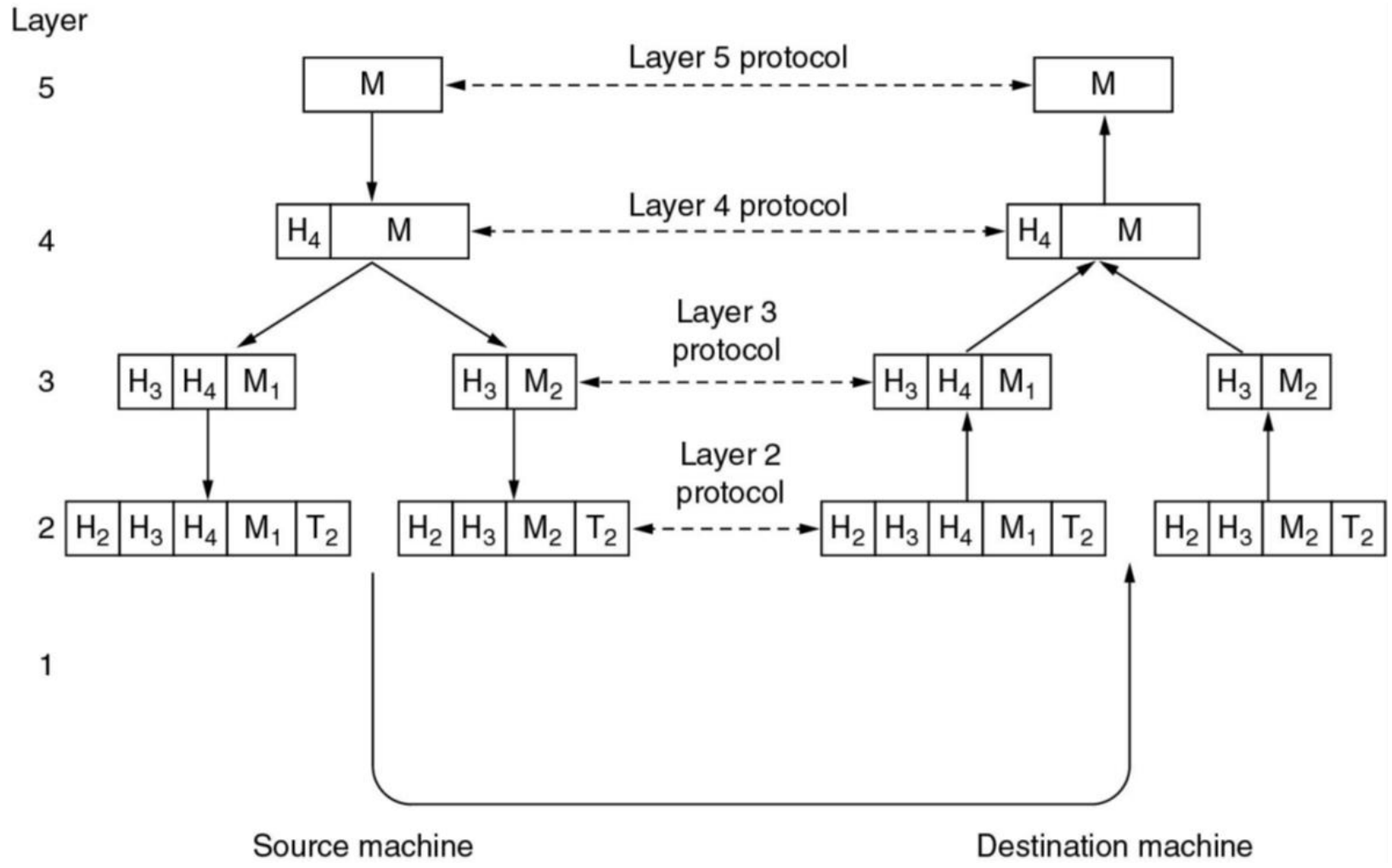
- **Agenda**:

1. Comprender los distintos tipos de redes de dispositivos.
2. Entender **cómo están organizados** los distintos tipos de redes (i.e. internet, la nube, la IoT y las redes blockchain)
3. **Entender la arquitectura de los sistemas operativos de redes (SOR) para los distintos tipos de redes.**
4. Entender algunas **convenciones** a respetar en la materia

Sistemas Operativos de Redes

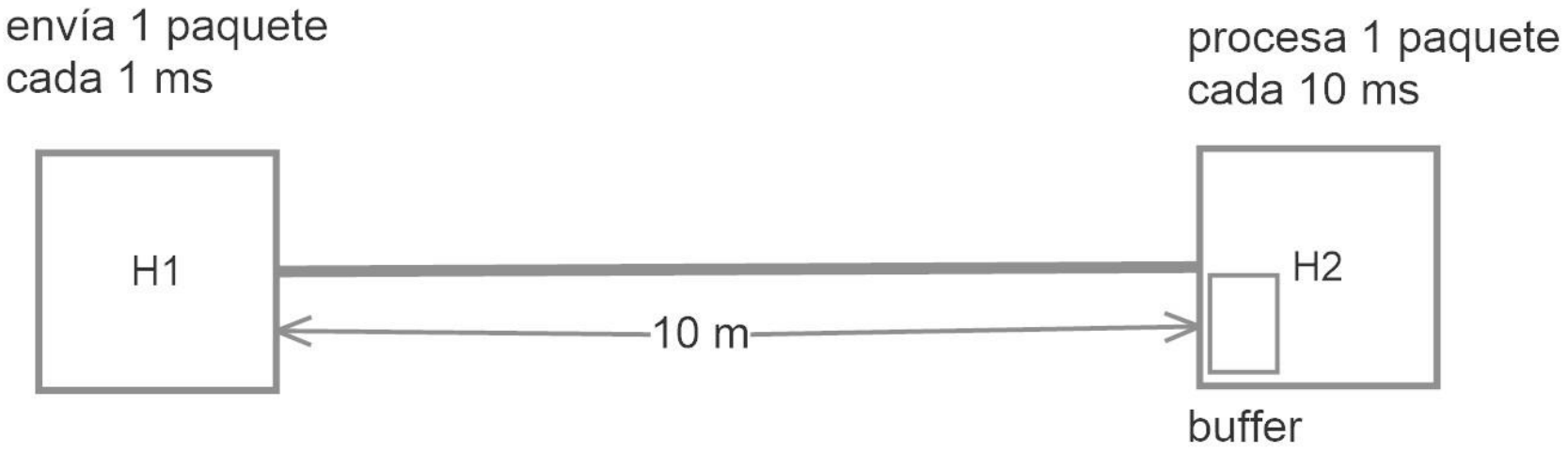
- **SO de red** = pila de capas
- Propósito de una capa
  - Ofrecer servicios
  - Comunicar información
  - Ocultación de implementación
- Interfaz de una capa
- Las capas contienen protocolos
- Comunicaciones entre capas consecutivas

Sistemas operativos para redes de computadoras



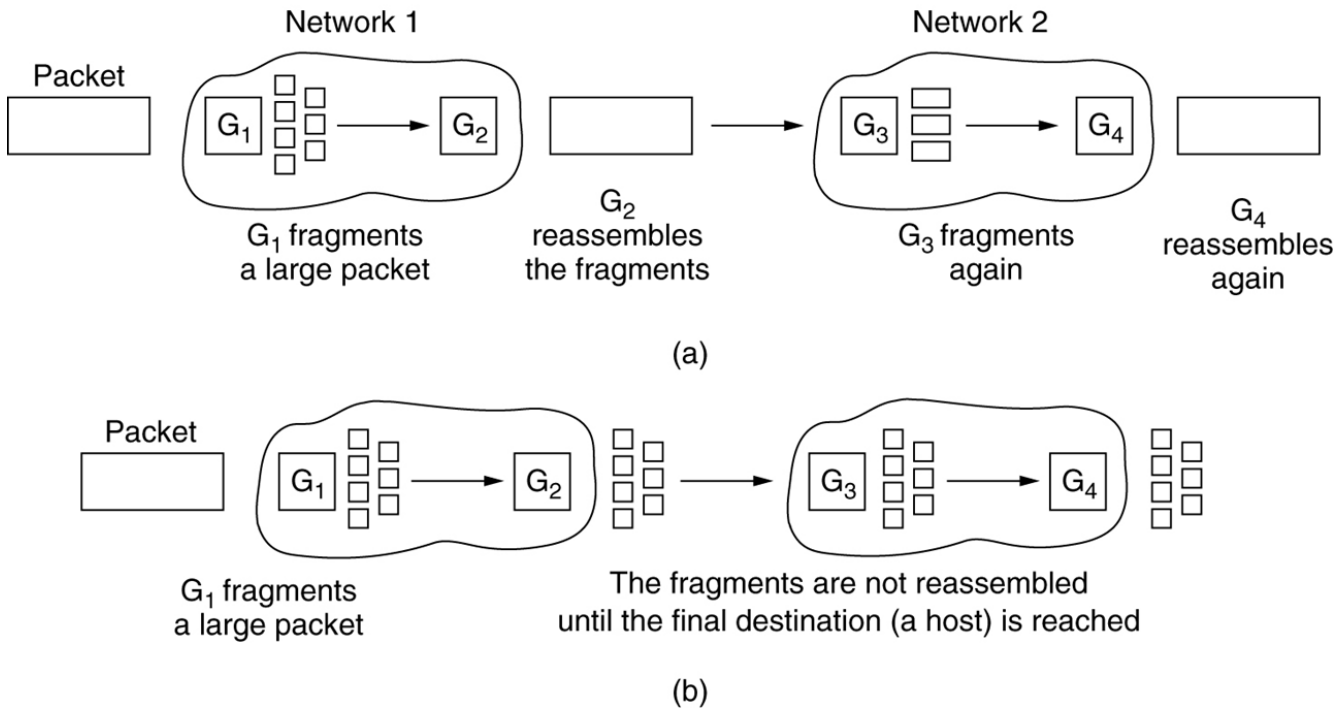
Sistemas operativos para redes de computadoras

- **Problemas de las redes de computadoras:**
  - **Identificación de las máquinas de una red**
    - Dirección IP, MAC
  - **Otras identificaciones**
    - De recursos, de procesos
  - **Control de flujo**
    - Retroalimentación del transmisor



Sistemas operativos para redes de computadoras

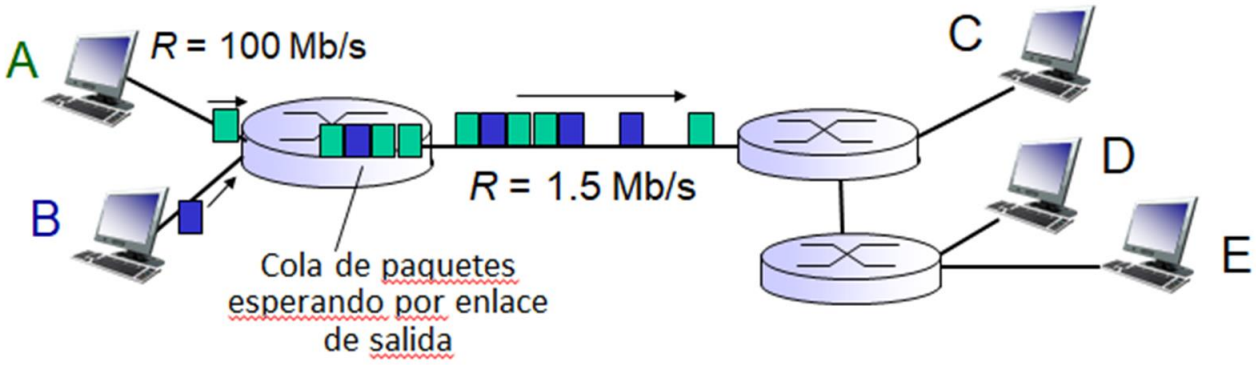
- **Problemas de las redes de computadoras - cont:**
  - Tamaño máximo de mensajes en cada tipo de red.
  - Problema si llega mensaje demasiado grande a una red.
  - **Fragmentación de mensajes.**



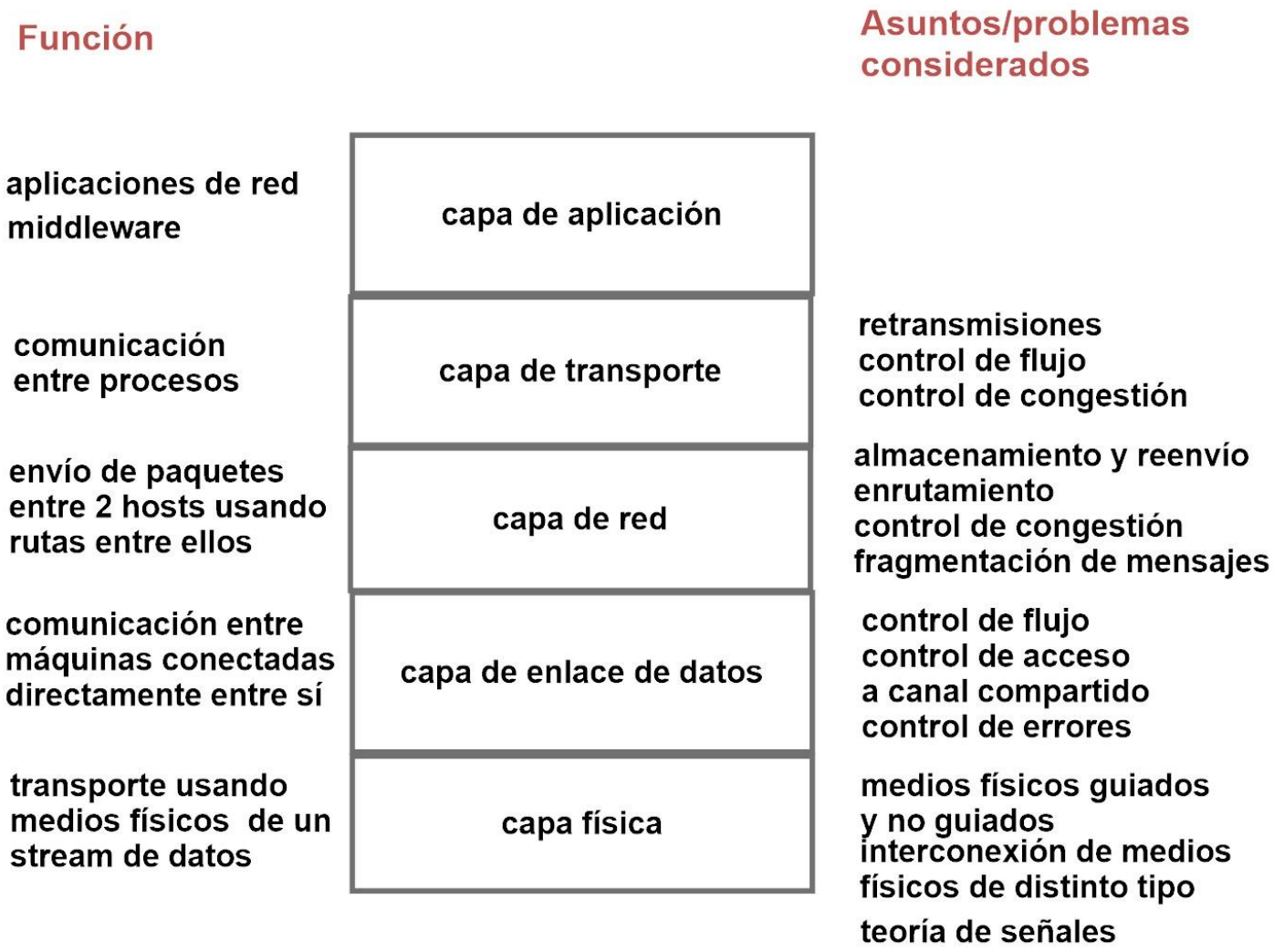
(a) Fragmentación Transparente. (b) Fragmentación no transparente.

Sistemas operativos para redes de computadoras

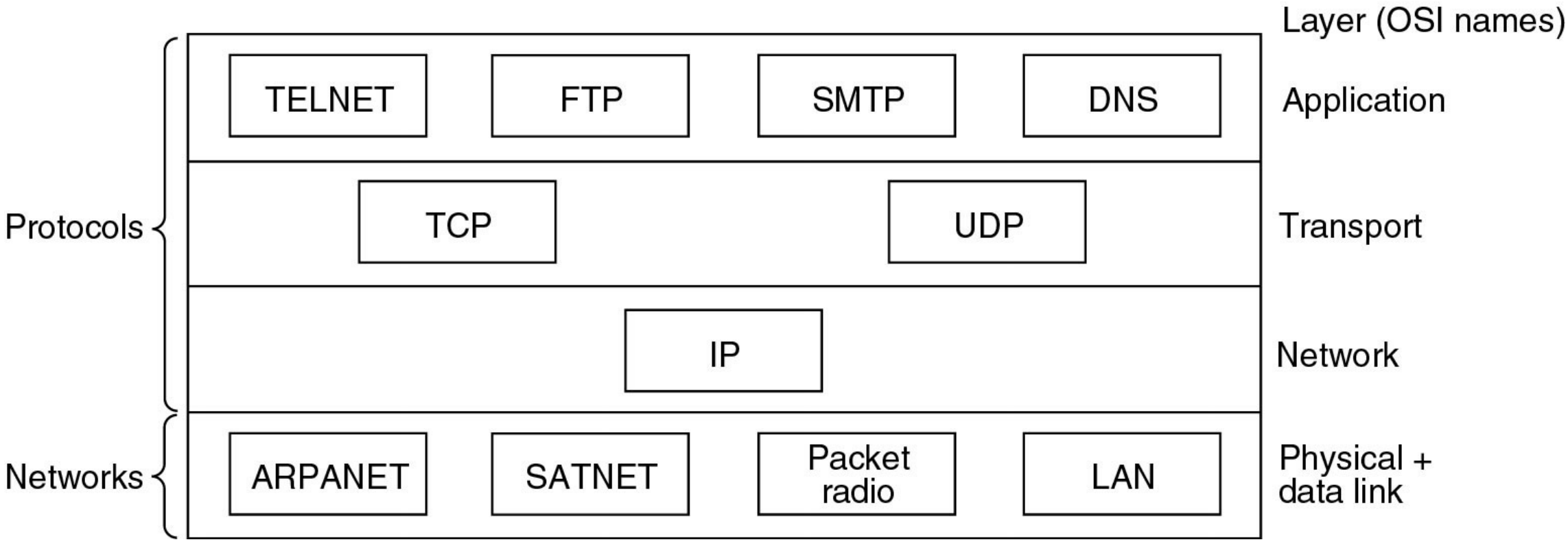
- **Problemas de las redes de computadoras - cont:**
  - Congestión de enrutadores
  - **Control de congestión**
    - ¿Dónde se puede realizar el control de congestión?
    - Ideas de solución



Modelo de referencia de redes de computadoras



Modelo de referencia TCP/IP



Capa de aplicación

- Aplicaciones de red
- **Opciones para desarrollar aplicaciones de red**
  - Socket API
  - Uso de middlewares: la web, RPC
- **La web:**
  - Protocolos HTTP, HTTPS, DNS
  - Sitios web
  - Aplicaciones web
  - Tecnologías de implementación: HTML, JavaScript, PHP, node.js, etc.

Capa de transporte

- **Función**: comunicación entre procesos de hosts diferentes
- Entidad de transporte
- **Problemas resueltos por capa de transporte**
  - Pérdida de mensajes
  - Control de flujo
  - Control de congestión
  - Servicios inactivos
  - Clientes no saben que servicios usar

- **Capa de transporte en Internet**
  - **TCP**:
    - Manejo de stream de bytes
    - Entrega de mensajes confiable y ordenada.
    - Control de flujo y de congestión
    - Manejo de conexiones
  - **UDP**
    - Entrega de mensajes no confiable y desordenada
    - Solo envía mensajes individuales y nada más.
    - Usos de TCP

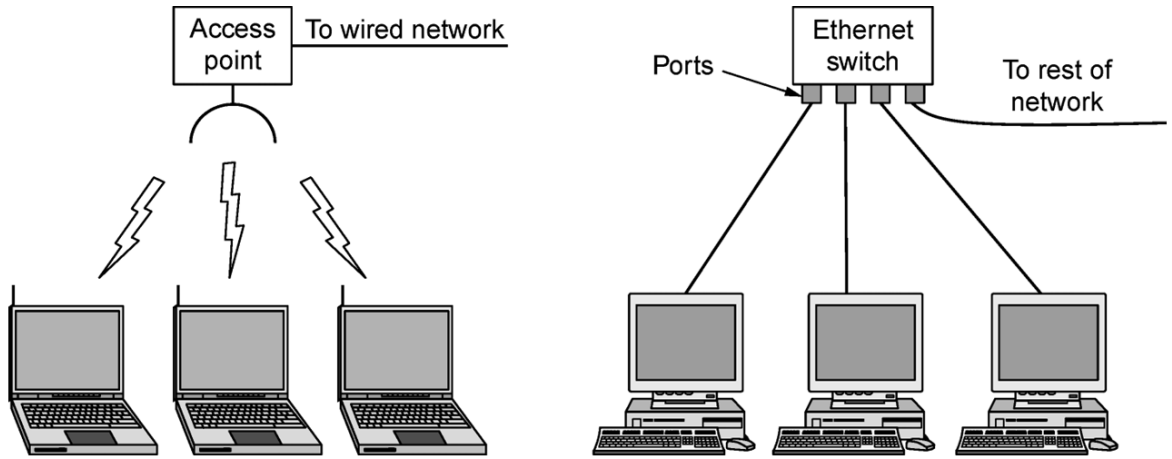
Capa de red

- **Problemas**
  - **Enrutamiento**
    - Elección de la mejor ruta
    - Algoritmo de enrutamiento
  - **Identificación** de las máquinas
  - Control de congestión
- **Interredes - problemas**
  - Manejo de diferencias entre redes diferentes
  - Fragmentación de mensajes

- **Capa de red de Internet**
  - **Internet** = conjunto de sistemas autónomos.
  - **IP**
    - Direcciones IP: IPv4 y IPv6
    - Paquetes IP
    - Protocolos de enrutamiento
      - OSPF – a nivel de Sistema autónomo.
      - BGP – a nivel de interred

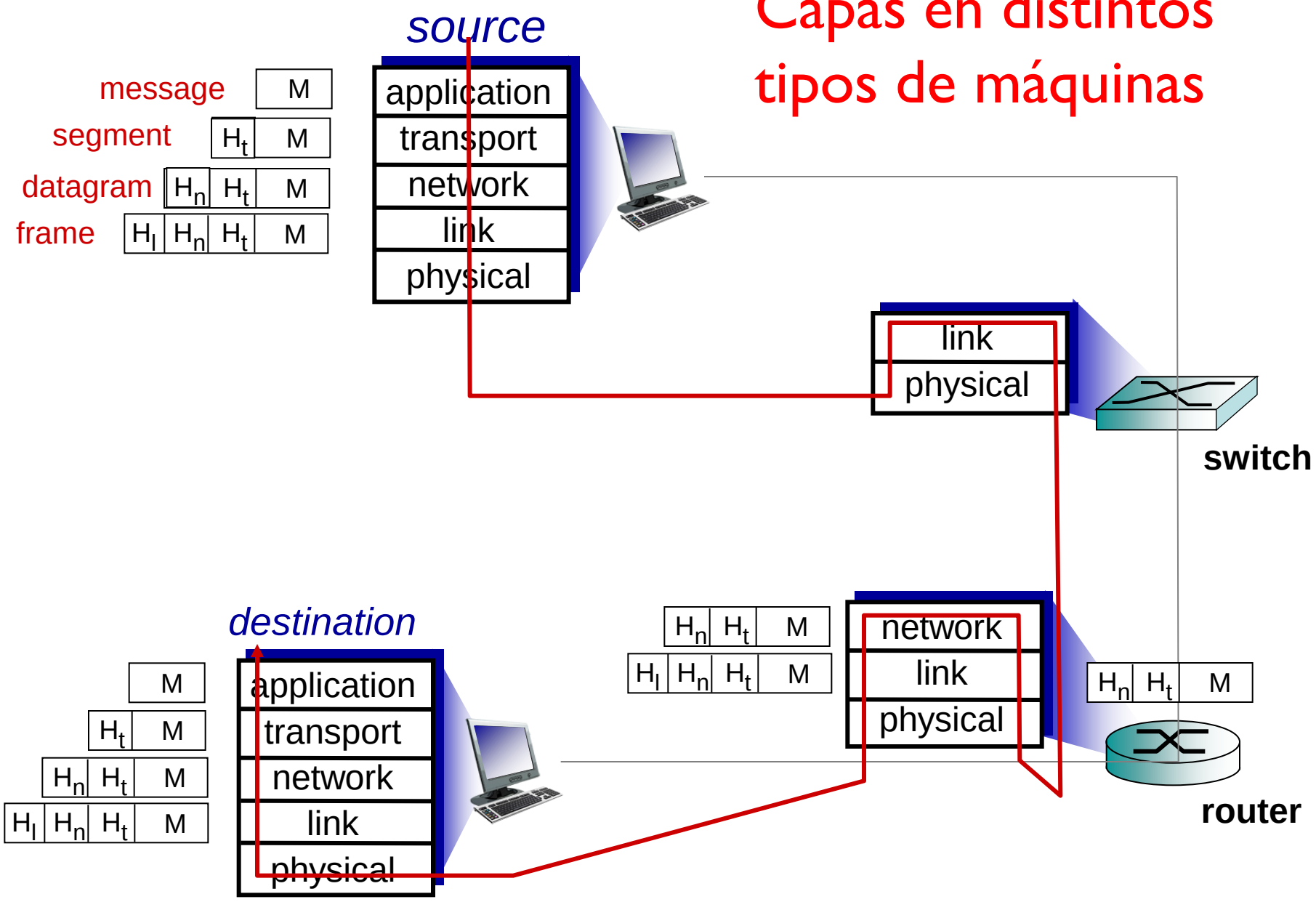
Capa de enlace de datos

- Propósito
- **Problemas**
  - División de paquetes entramas
  - Manejo de pérdidas de tramas
  - Control de flujo
  - Control de acceso a canal compartido
    - Control de colisiones: enfoques
  - Control de errores
    - Enfoques
- **Tecnologías de capa de enlace de datos**
  - Ethernet
  - WiFi



Wireless and wired LANs. (a) 802.11 (WIFI). (b) Switched Ethernet.

Capas en distintos tipos de máquinas

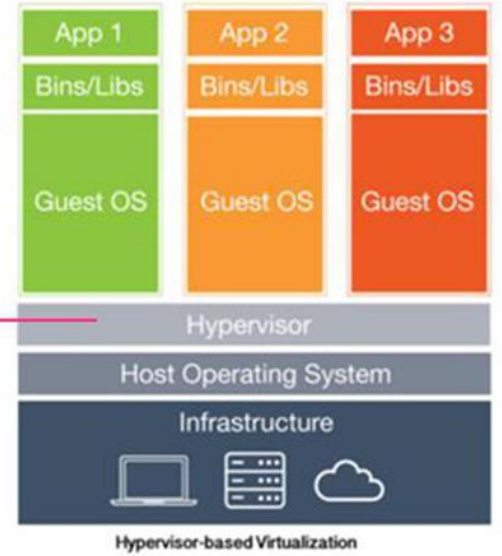


Sistema operativo de la nube

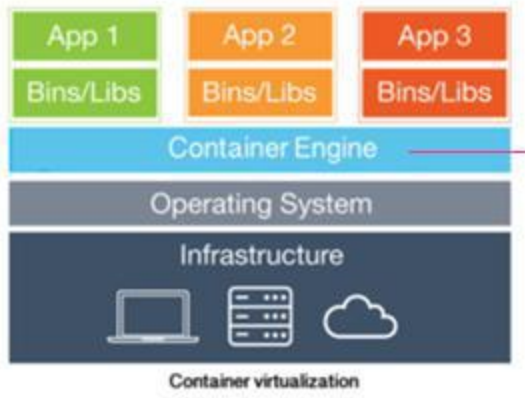
- **Servicios de la nube**
  - **Infraestructura como servicio (IaaS)**
    - Manejo de recursos como almacenamiento, máquinas virtuales, servidores, redes
    - El usuario setea, configura los recursos.
    - El usuario instala, gestiona y monitorea el software
  - **Plataforma como servicio (PaaS)**
    - El usuario desarrolla, testea, despliega y gestiona aplicaciones
    - La nube provee infraestructura requerida, midlewares y otros recursos necesitados.
    - La nube provee herramientas y servicios para desarrollo de aplicaciones
  - **Software como servicio (SaaS)**
    - Acceso a aplicaciones mediante suscripción

Sistema operativo de la nube

- **Virtualización**
  - Varias máquinas virtuales en servidor físico.
  - **Máquinas virtuales**
    - ¿Y si falla una?
  - Hipervisor
  - Host OS
- **Containerización**
  - Aplicaciones en contenedores portables
  - **Contenedores en un servidor**
    - ¿y si falla uno?
  - Máquina de contenedores



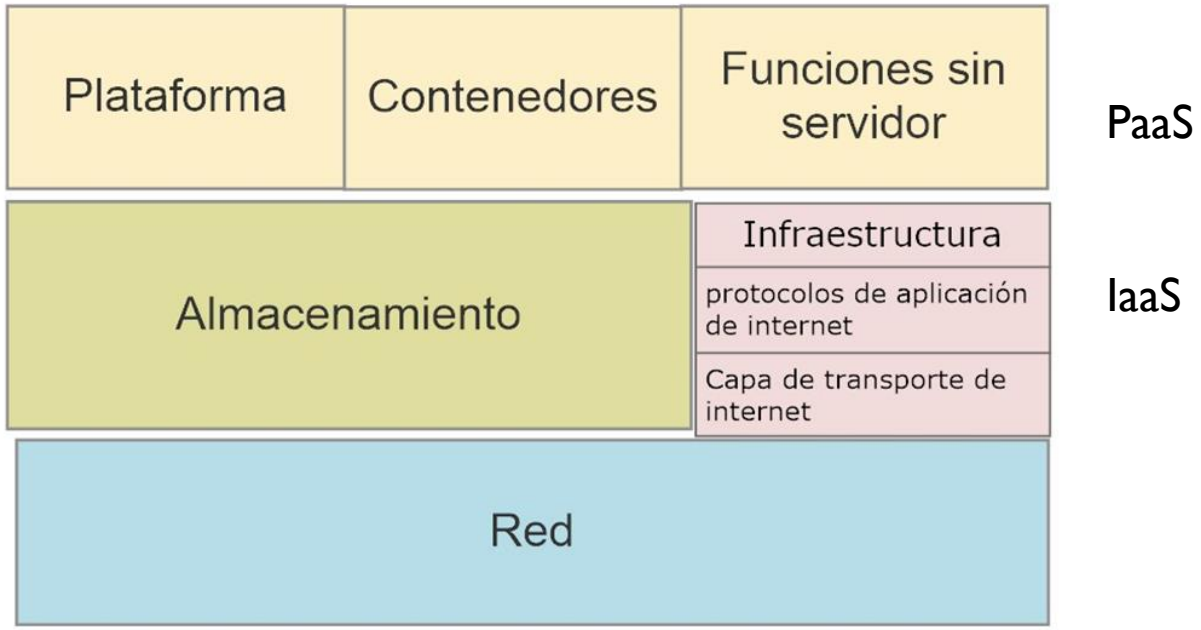
- Virtual Box
- VMware



- Docker
- Kubernetes
- Docker compose

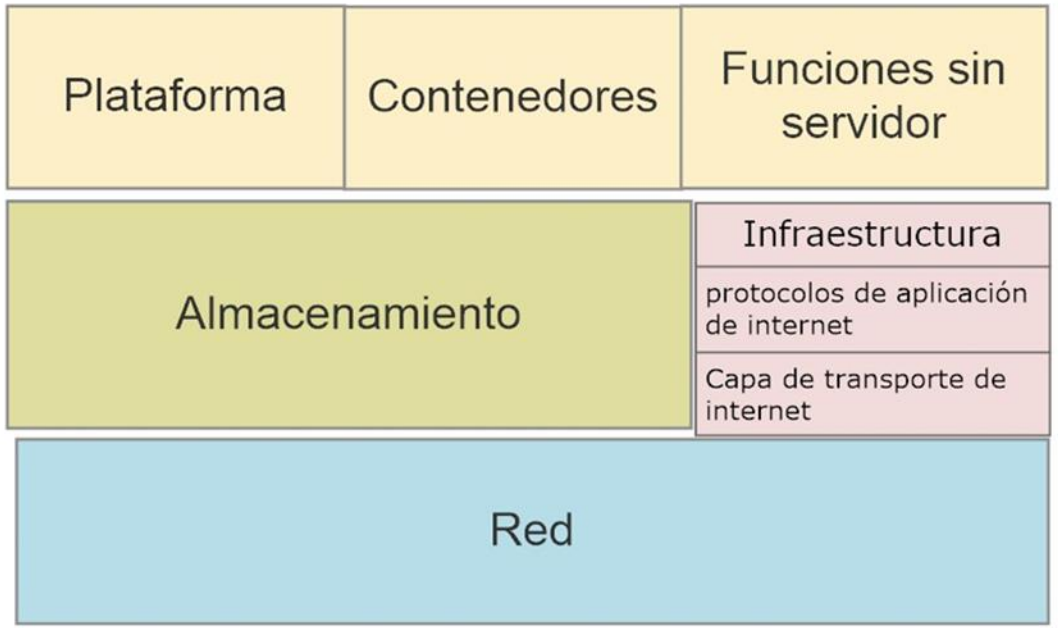
Sistemas operativos para la nube

- **La capa de Red**
  - Propósito
  - **Tipos de protocolo**
    - Protocolos de Internet
    - **Protocolos redes privadas virtuales**: conexiones seguras usando la internet
      - Ejemplos: OpenVPN,WireGuard.
    - **Protocolos para conexiones privadas:**
      - no usan internet
      - Son más rápidas y seguras
      - **Ejemplos**: MPLS , Direct Connect de AWS, ExpressRoute de Azure.



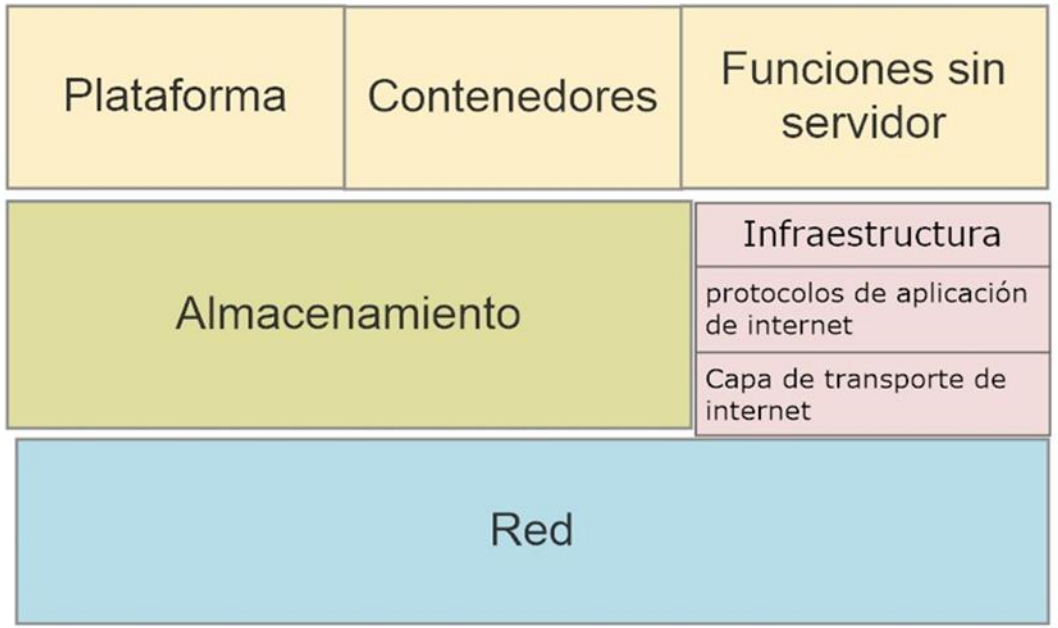
Sistemas operativos para la nube

- **Capa de irfraestructura:**
  - **Propósito**: comunicación y transferencia de datos entre servidores y clientes.
  - **Protocolos**:
    - De capa de aplicación de internet
    - De capa de transporte de internet



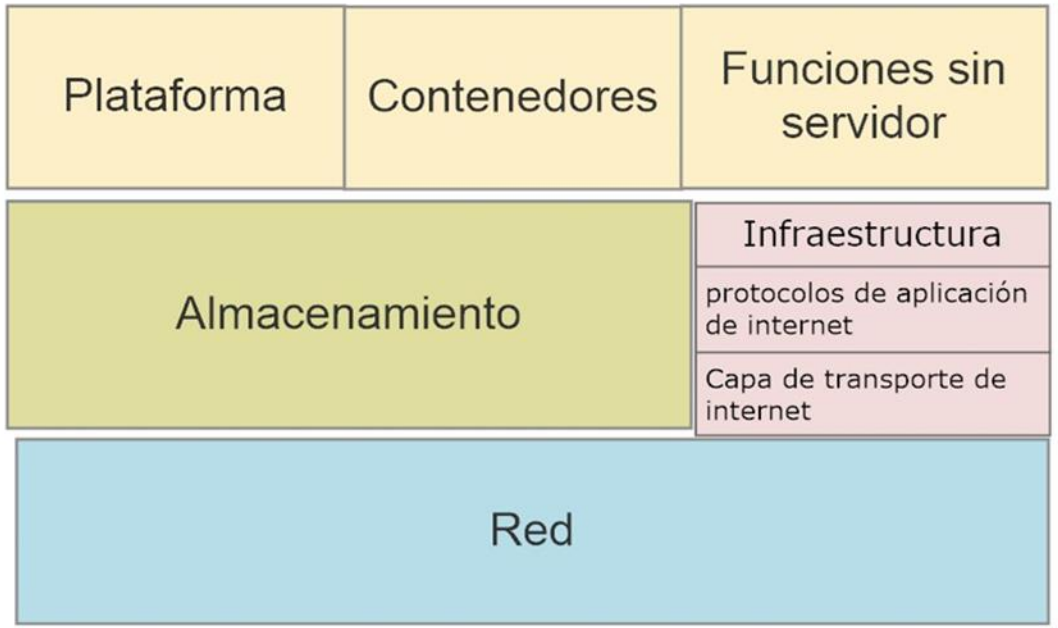
Sistemas operativos para la nube

- **Capa de almacenamiento:**
  - Propósito
  - **Tipos de almacenamiento**
    - **En bloque**: p.ej. iSCSI.
    - **De archivos**: acceso a sistema de archivos. p.ej. NFS.
    - **De objetos**: archivos multimedia, imágenes de disco, back-up de BD, datos IoT, logs.



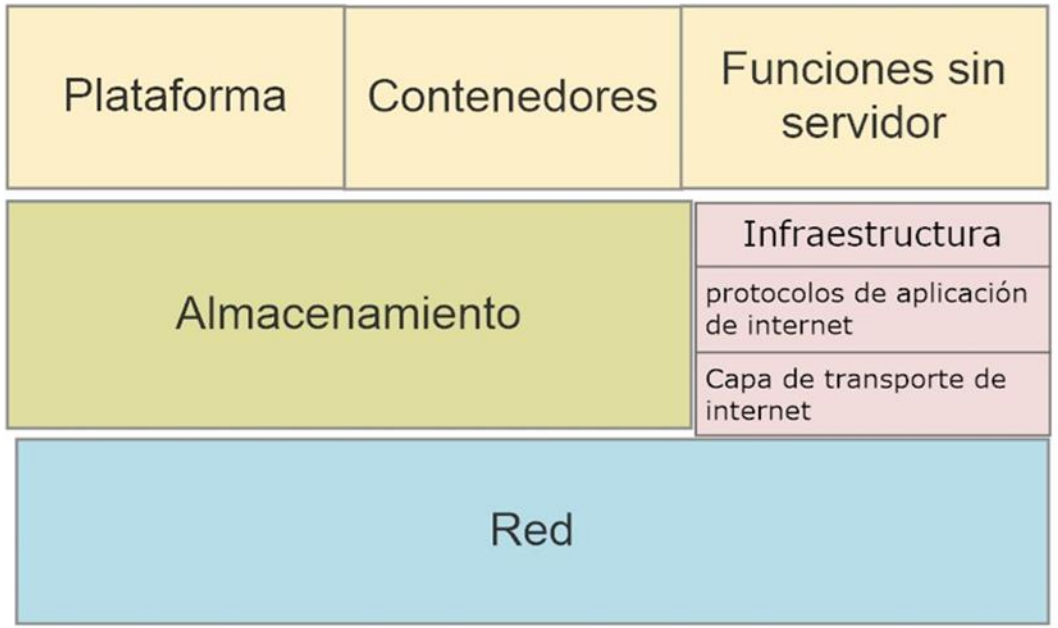
Sistemas operativos para la nube

- **Capa de plataforma:**
  - **Propósito**:
    - Herramientas y servicios para construir y ejecutar apps.
    - Uso de APIs para que apps conecten y trabajen con bases de datos, almacenamiento o procesamiento.
  - **Protocolos**:
    - REST
    - gRPC



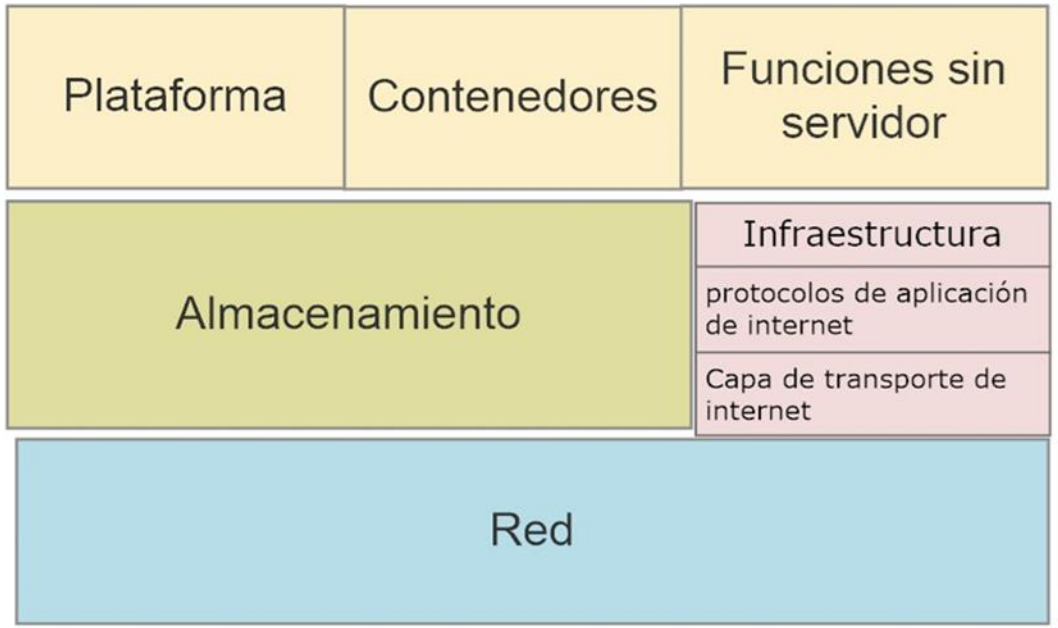
Sistemas operativos para la nube

- **Capa de funciones sin servidor:**
  - **Propósito**: si ocurren eventos, ejecutar funciones.
  - **Tipos de eventos**: pedidos HTTP, modificaciones en BD, eventos de almacenamiento, eventos de apps externas, intervalos regulares, etc.
  - **Ejemplos**: Azure Functions, AWS Lambda.

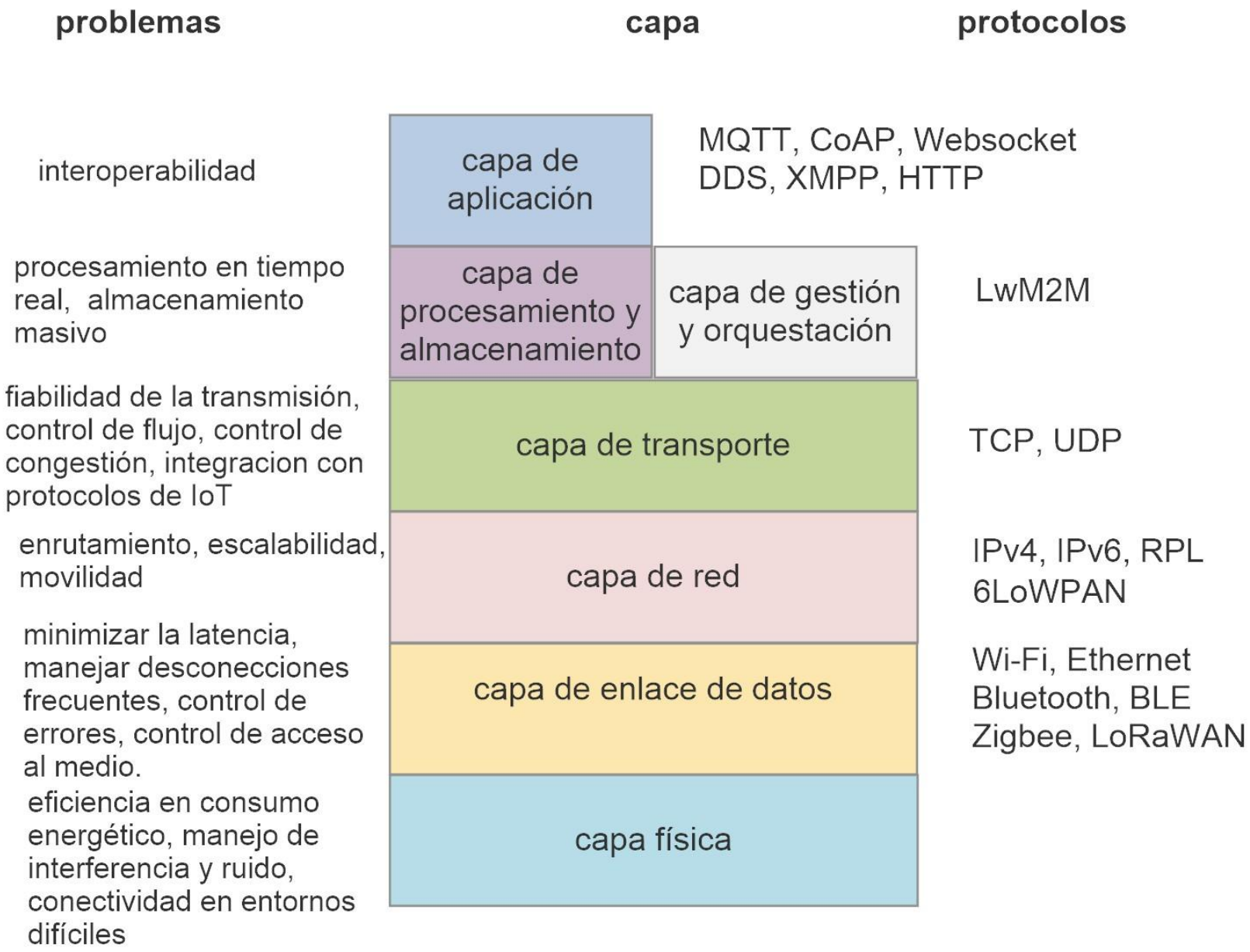


Sistemas operativos para la nube

- **Capa de contenedores:**
  - **Función**
  - **Ejemplos**: Kubernetes y Docker



Sistema Operativo para Internet de las Cosas



Sistema Operativo para Internet de las Cosas

- **La capa física**
  - **Sugerir definición.**
  - **Problemas atacados:**
    1. Problemas que vienen de la capa física de las redes de computadoras
    2. Eficiencia en el consumo energético
    3. Tratar efectivamente con la interferencia y ruido
    4. Mantener la conectividad en entornos difíciles
       - Concepto de conectividad
  - **Protocolos**:
    - **Zigbee**: diseñado para atacar todos los problemas anteriores.
    - **LoRaWAN**: diseñado para problemas 2 y 4, comunicación de largo alcance, permite gran cantidad de dispositivos.

Sistema Operativo para Internet de las Cosas

- **La capa de enlace de datos:**
  - Sugerir definición en base a lo que saben.
  - **Problemas considerados:**
    1. Problemas que vienen de las redes de computadoras
    2. Manejar retrasos y variaciones en tiempo de transmisión
    3. Manejo de desconexiones frecuentes
    4. Manejo de seguridad de la comunicación
  - **Protocolos**:
    - Ya mencionados: Wi-Fi, Ethernet, Bluetooth
    - Zigbee, LoRaWan
    - **Bluetooth Low Energy (BLE)**:
      - Para dispositivos con baterías pequeñas.
    - **6LoWPAN**:
      - Permite encapsular paquetes IPv6 en redes LAN inalámbricas de baja potencia.
      - Compresión de encabezados y fragmentación (para P2).

Sistema Operativo para Internet de las Cosas

- **La capa de red:**
  - Dar la definición que saben
  - **Problemas atacados:**
    1. Ya conocidos de antes
    2. Escalabilidad
    3. Movilidad afuera y dentro de la red

  - **Protocolos**:
    - IPV4 e IPv6
    - **6LoWPAN**
      - Integración con redes IP tradicionales.
      - Escalabilidad: uso de direcciones IPv6, compresión de cabeceras.
      - Maneja movilidad.
    - **RPL**:
      - Para dispositivos de baja potencia y alta tasa de pérdida de paquetes.
      - Escalable
      - Adapta las rutas de acuerdo a las condiciones de la red.
      - No fue hecho para movilidad.

Sistema Operativo para Internet de las Cosas

- **La capa de transporte**:
  - Recordar definición anterior
  - **Problemas atacados:**
    - Los mismos de antes.
  - **Protocolos**:
    - TCP, UDP

- **MQTT**
  - Para ancho de banda limitado, minimiza consumo de energía durante transmisiones
  - Formato de mensaje compacto.
  - Se apoya en TCP
  - Comunicación en tiempo real; comunicación M2M.
- **CoAP**
  - Para dispositivos con hardware limitado
  - Formato de mensaje compacto
  - Se apoya en UDP

Sistema Operativo para Internet de las Cosas

- **Capa de procesamiento y almacenamiento**
  - Procesamiento, almacenamiento y análisis de datos recopilados por dispositivos IoT.
  - **Problemas**:
    - Procesamiento rápido de datos
      - Edge computing
    - Almacenamiento masivo
      - Cómputo en la nube
  - **Componentes**: servidores locales, Edge computing, la nube.

- **Capa de gestión y orquestación:**
  - Gestión y orquestación de recursos en la red IoT
  - **Funciones**:
    - Configuración, monitoreo, actualización, administración de dispositivos IoT..
  - **Problemas**:
    - Configuración y monitoreo eficiente.
    - Actualización de firmware de manera remota.
  - **Protocolos**:
    - LwM2M.

Sistema Operativo para Internet de las Cosas

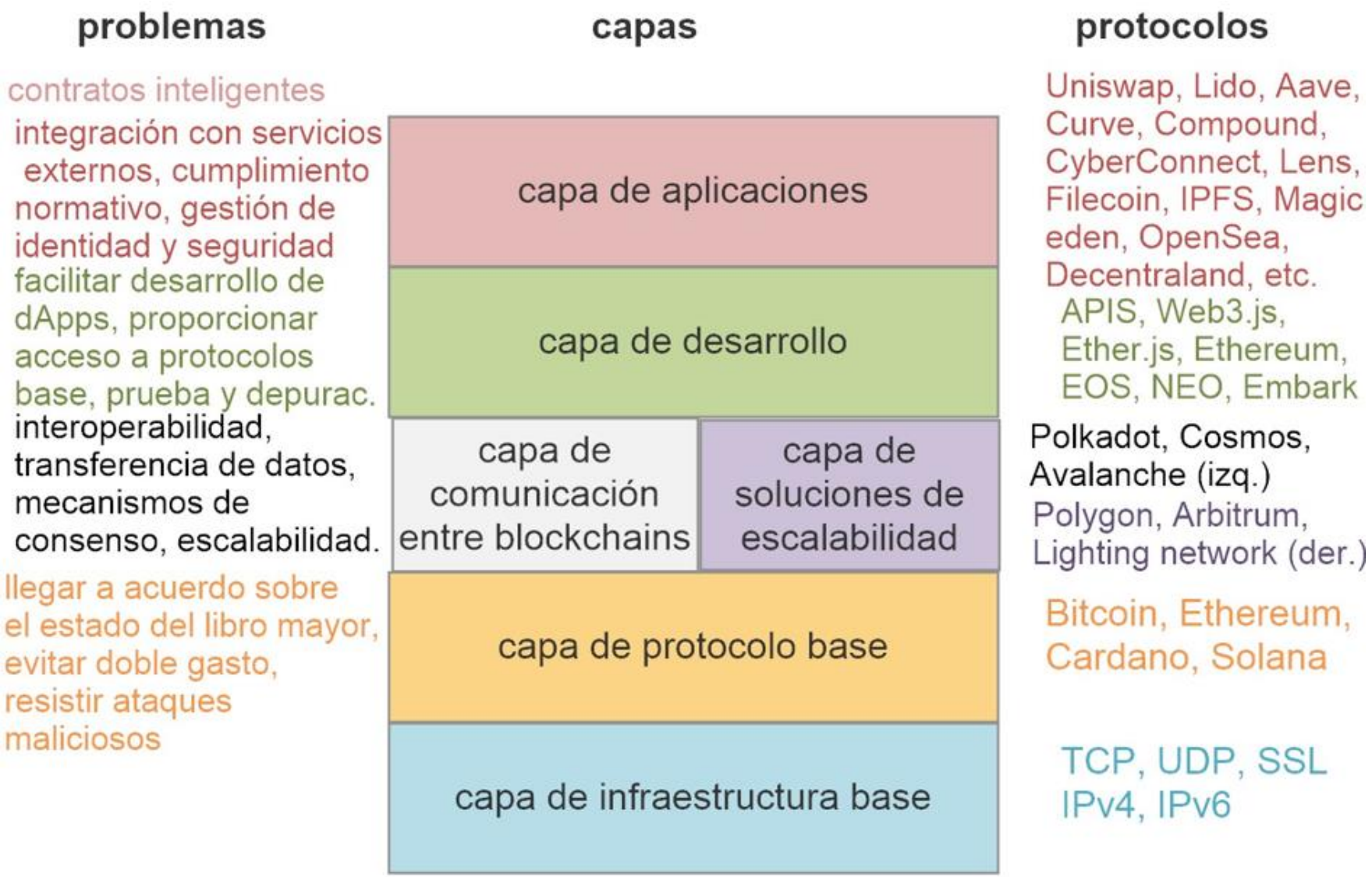
- **La capa de aplicación:**
  - Middlewares pero para IoT.
  - **Problemas**:
    1. Interoperabilidad
    2. Gestión de dispositivos
    3. Seguridad
    4. Eficiencia energética
    5. Entrega confiable de mensajes
    6. Escalabilidad y ancho de banda
    7. Simplicidad para dispositivos con recursos limitados
    8. Complejidad suficiente para manejar las necesidades de las aplicaciones

  - **Protocolos:**
    - **HTTP, HTTPS**
    - **MQTT**
      - Modelo de publicación-suscripción (P4). Tópicos, Broker.
      - Usa TLS/SSL para seguridad
      - Niveles de servicio (P5)
      - Manejo de grandes volúmenes de datos (P6)
    - **WebSocket**:
      - Basado en TCP
      - Stream de mensajes bidireccional durante conexión entre cliente y servidor.

Sistema Operativo para Internet de las Cosas

- **La capa de aplicación - cont:**
  - **Protocolos - cont:**
    - **CoAP:**
      - Usa UDP y es ideal para dispositivos con baterías
      - Usa modelo cliente-servidor
      - Está diseñado para ser ligero y fácil de implementar.
      - Usa métodos similares a HTTP
      - Envía códigos de estado para las respuestas
      - Tiene mecanismo de descubrimiento de recursos
      - Usa URIs para recursos alojados en dispositivos IoT
      - Usa DTLS o IPSec para seguridad.
      - Usa confirmaciones de mensajes.

Sistemas Operativos para Redes Blockchain



Sistemas Operativos para Redes Blockchain

- **Capa de protocolo base:**
  - Responsible de seguridad y funcionamiento operative de la red blockchain.
  - **Funciones**:
    - Diseño de **cadena de bloques**
    - Establece **reglas de consenso**
      - para llegar a un acuerdo sobre el estado del libro mayor.
      - Previene doble gasto
      - Resistencia a ataques maliciosos
    - Reglas de comunicación entre nodos.
    - Procesamiento, validación y ordenamiento de transacciones.
    - Protección de transacciones y autorización de usuarios.
    - Gestión de tarifas y manejo de congestión.
    - Soporte de contratos inteligentes
  - **Problemas**:
    - Escalabilidad y rendimiento
    - Seguridad contra ataques
    - Eficiencia energética
    - Costos de transacciones predecibles
  - **Protocolos**:
    - **Bitcoin**: con minería, sin contratos inteligentes, limitación de suministro, reserva de valor y medio de pago.
    - **Ethereum**: no usa minería, con contratos inteligentes, programabilidad de apps sobre Ethereum, aplicaciones de tokenización.
    - **Cardano**: Eficiencia energética, escalabilidad, contratos inteligentes. Sin minería. Tarifas relativamente bajas. Exchanges descentralizados.
    - **Solana:** alto rendimiento, baja latencia, tarifas muy bajas, aplicaciones de finanzas descentralizadas. Contratos inteligentes.

Sistemas Operativos para Redes Blockchain

- **Capa de comunicación entre blockchains:**
  - Interoperabilidad entre diferentes redes blockchain (i.e. comunicación e intercambio de información entre ellas.).
  - **Problemas que resuelve**
    - Eliminación del aislamiento
    - Incompatibilidad de protocolos
    - Evita usar intermediarios para transferir activos entre redes
      - Transacciones entre distintas blockchain.
    - Mejora la escalabilidad
    - Permite apps que integran varias redes blockchain.

- **Protocolos**:
  - **Polkadot**: para intercambio de datos
  - **Cosmos**: para intercambio de activos y datos.
  - **Avalanche**: tiene 3 tipos de cadenas que se comunican entre si.

Sistemas Operativos para Redes Blockchain

- **Capa de soluciones de escalabilidad:**
  - Mejora el rendimiento y la capacidad de procesamiento de transacciones.
  - Se construye sobre una blockchain existente.
  - **Ventaja**: Reduce carga sobre la blockchain y disminuye costos asociados.
  - **Soluciones**:
    - **Rollups**: agrupa en un paquete varias transacciones que se validan en red secundaria y
      - registra resultado final en blockachain base.
- **Cadenas laterales**: blockchains independientes se conectan a blockchain principal.
  - Cuando blockchain principal esta congestionada se usa cadena lateral para procesar transacciones, ejecutar contratos y ejecutar apps.
  - Cadenas laterales con sus propios protocolos y mecanismos de consenso. Tienen tarifas menores.
- **Ejemplos**:
  - **Polygon**: para Ethereum, usa cadenas laterales y rollups
  - **Arbitrum**: para Ethereum, usa rollups.

Sistemas Operativos para Redes Blockchain

- **Capa de aplicaciones**:
  - Aplicaciones descentralizadas (dApps.)
  - **Características de las dApps:**
    - No dependen de servidores.
    - Usan contratos inteligentes.
    - Usan APIs para interactuar con la blockchain.
    - Los usuarios interactúan directamente con una dApp.
  - **Ejemplos de dApps:**
    - Finanzas descentralizadas
    - Gestión de cadenas de suministro
    - Votación
    - Almacenamiento descentralizado
    - Mercados NFT
    - Juegos

Convenciones a respetar

- B mayúscula = 1 byte = 8 bits (= 2^3 bits)
- 1KB = 2^10 B = 1024 B (=2^13 bits = 8192 bits) - **kibi**byte
- 1MB = 2^20 B = 1.048.576 B
- 1GB = 2^30 B
- En resumen, para los casos anteriores se usan potencias de 2 junto con bytes.
- En cambio, b minúscula = 1 bit
- 1Kb = 10^3 b = 1000 b – **Kilo** bit
- 1Mb = 10^6 b = 1.000.000 b – **Mega** bit
- 1Gb = 10^9 b = 1000.000.000 b – **Giga** bit
- En resumen, se usan potencias de 10 junto con bits.
- Para expresar velocidades de transmisión:
  - 1Kbps = 1000 bits por segundo
  - 10Mbps = 10^6 bps = 10.000.000 bits por segundo
  - 10Gbps = 10^9 bps

Convenciones a respetar

Exp.	Explicit	Prefix	Exp.	Explicit	Prefix
10^{-3}	0.001	milli	10^{3}	1,000	Kilo
10^{-6}	0.000001	micro	10^{6}	1,000,000	Mega
10^{-9}	0.000000001	nano	10^{9}	1,000,000,000	Giga
10^{-12}	0.000000000001	pico	10^{12}	1,000,000,000,000	Tera

Las unidades de medida pequeñas pueden usarse para medir tiempo
Los unidades de medida grandes pueden usarse para expresar cantidades de bits