Capítulo 1

Introducción a las Redes de Dispositivos

Parte 2

Sistemas Operativos de Redes

- **¿Por qué estudiar los sistemas operativos de redes?**
  - Para manejar los tipos de redes que estudiamos hacen falta **sistemas operativos de redes (SOR).**
  - En cada tipo de red hay **problemas** a ser resueltos si queremos que no tenga un mal desempeño y los SOR se encargan de resolver esos problemas.
    - Estudiarán cómo resolver problemas en los tipos de redes por medio de SOR.
  - Para que las máquinas en un tipo de red se puedan comunicar hacen falta **protocolos de comunicación**; los SOR contienen esos protocolos.
    - Estudiarán cómo diseñar protocolos que permiten la comunicación entre hosts en los distintos tipos de redes.

Sistemas Operativos de Redes

- Los **sistemas operativos de redes (SOR)** están organizadas como una **pila de capas o niveles**, cada una construida arriba de la que está debajo de ella.
- La cantidad de capas, los nombres de las capas, sus contenidos y su función, difieren de un tipo de red a otro.

Jerarquías de Protocolos

- **Pilas de capas** se usan para reducir la complejidad del diseño de los SOR.
- ***¿Han estudiado arquitecturas de capas en alguna materia?¿Cómo era?***

Jerarquías de Protocolos

- **Propósito de una capa en arquitecturas multicapa**
  - ofrecer ciertos servicios a las capas superiores
  - ocultar la implementación a las capas superiores
- **Interfaces entre capas** = operaciones y servicios primitivos ofrecidos por una capa a capa superior.
- **Un SOR se preocupa de la comunicación de información.**
- **Esto afecta al propósito de las capas de la siguiente manera:**
  - Una capa *n* se piensa como una conversación entre la capa *n* de una máquina con la capa *n* de otra máquina,
    - sin tener que preocuparnos de ciertos problemas que resuelven las capas inferiores a la capa *n*.
  - Para especificar cómo es esta conversación se definen **protocolos**

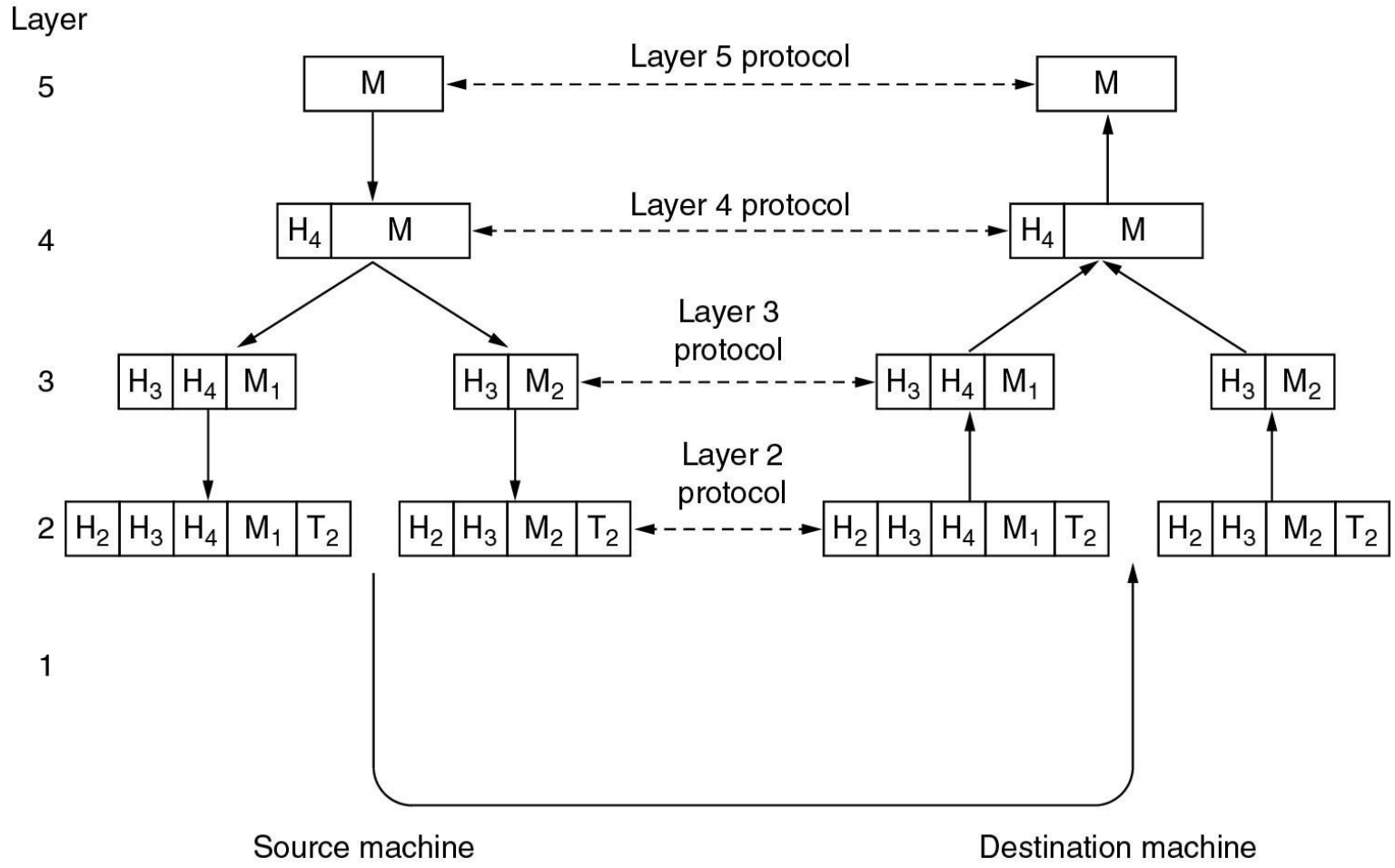
Jerarquías de Protocolos

- **Protocolo de capa *n*** = reglas y convenciones usadas en la **conversación** entre la capa *n* de una máquina y la capa *n* de otra máquina.
- **Comunicaciones entre capas consecutivas ocurren:**
  - **Durante el envío de mensaje**: cada capa pasa los datos y la información de control a la capa inmediatamente inferior, hasta que se alcanza la capa más baja.
  - **Durante la recepción de mensaje**: cada capa pasa cierta información conteniendo los datos a la capa inmediatamente superior hasta que alcanza la capa más alta.
- Debajo de la capa 1 está el **medio físico**.
- **arquitectura de red** = conjunto de capas y protocolos = **pila de protocolos**.

Sistemas operativos para redes de computadoras

- Ahora veremos sistemas operativos para redes de computadoras.
- En una **red de computadoras** la red se usa para la comunicación entre computadoras.
  - Para ello se pueden usar nodos intermediarios como conmutadores, enrutadores y puertas de enlace.
- **Ejemplos de redes de computadoras**: la internet. Una WAN.
- Comenzamos viendo cómo sería una jerarquía de protocolos para una red de computadoras.

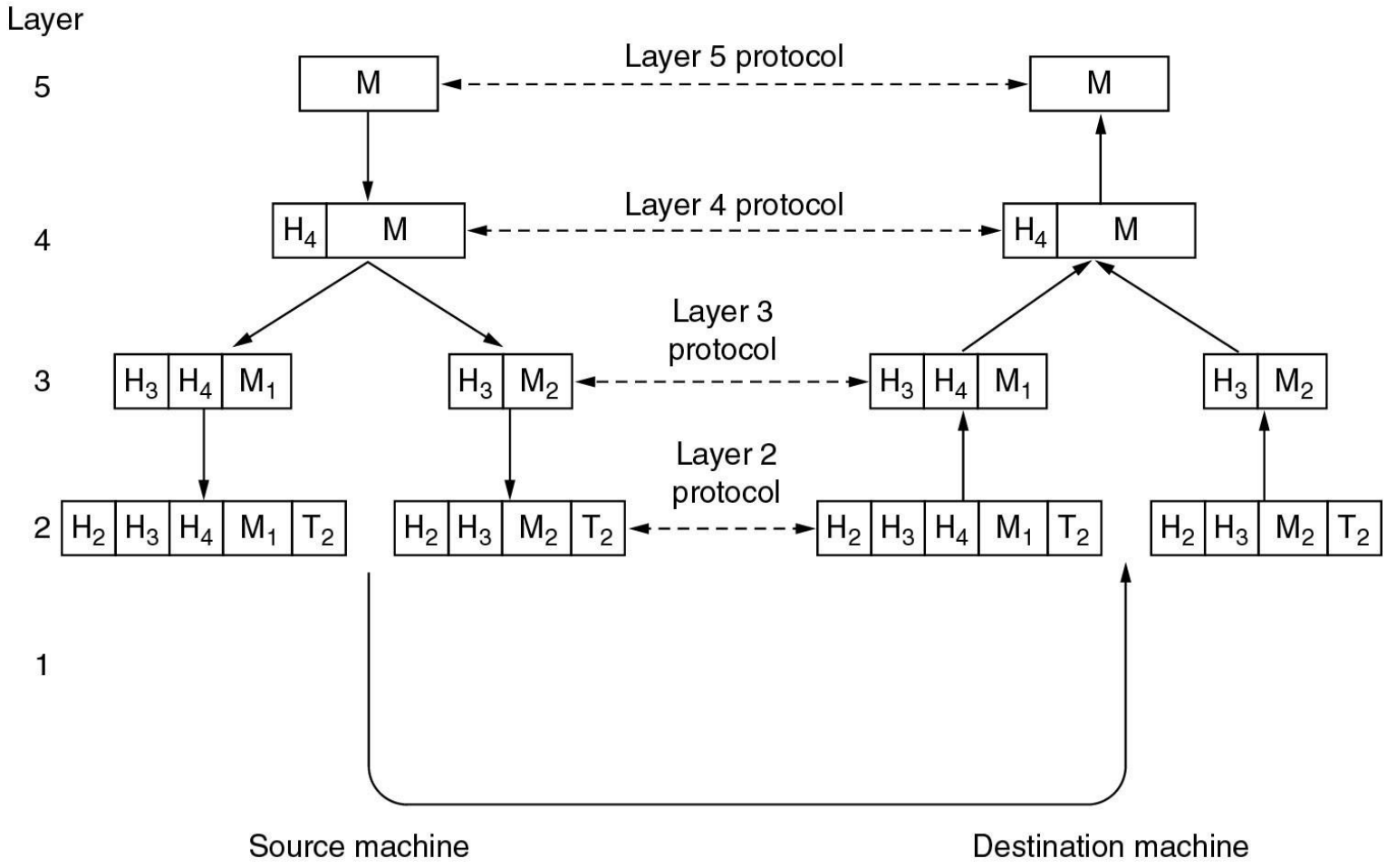
Sistemas operativos para redes de computadoras



Procesos de aplicación (capa 5 o **capa de aplicación**)

- P.ej. browser, e-mail, chat, ftp, etc.
- Produce un mensaje y lo pasa a la capa 4 para su transmisión.

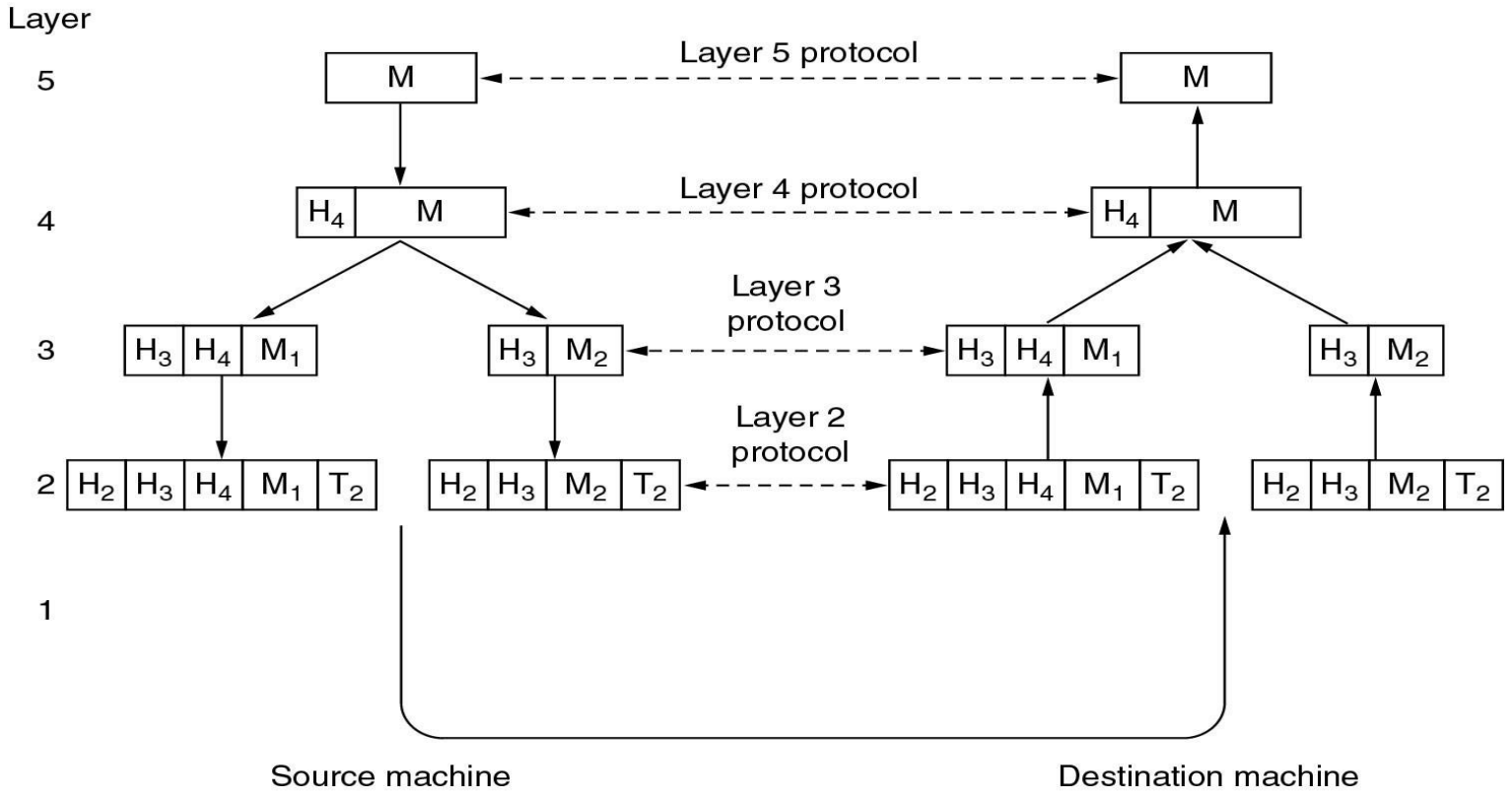
Sistemas operativos para redes de computadoras



La capa 4 (**capa de transporte**)

- pone un encabezado en el mensaje para identificarlo y pasa el resultado a la capa 3.
- El encabezado contiene **números de secuencia** para que la capa 4 en la máquina de destino entregue los mensajes en el orden correcto.

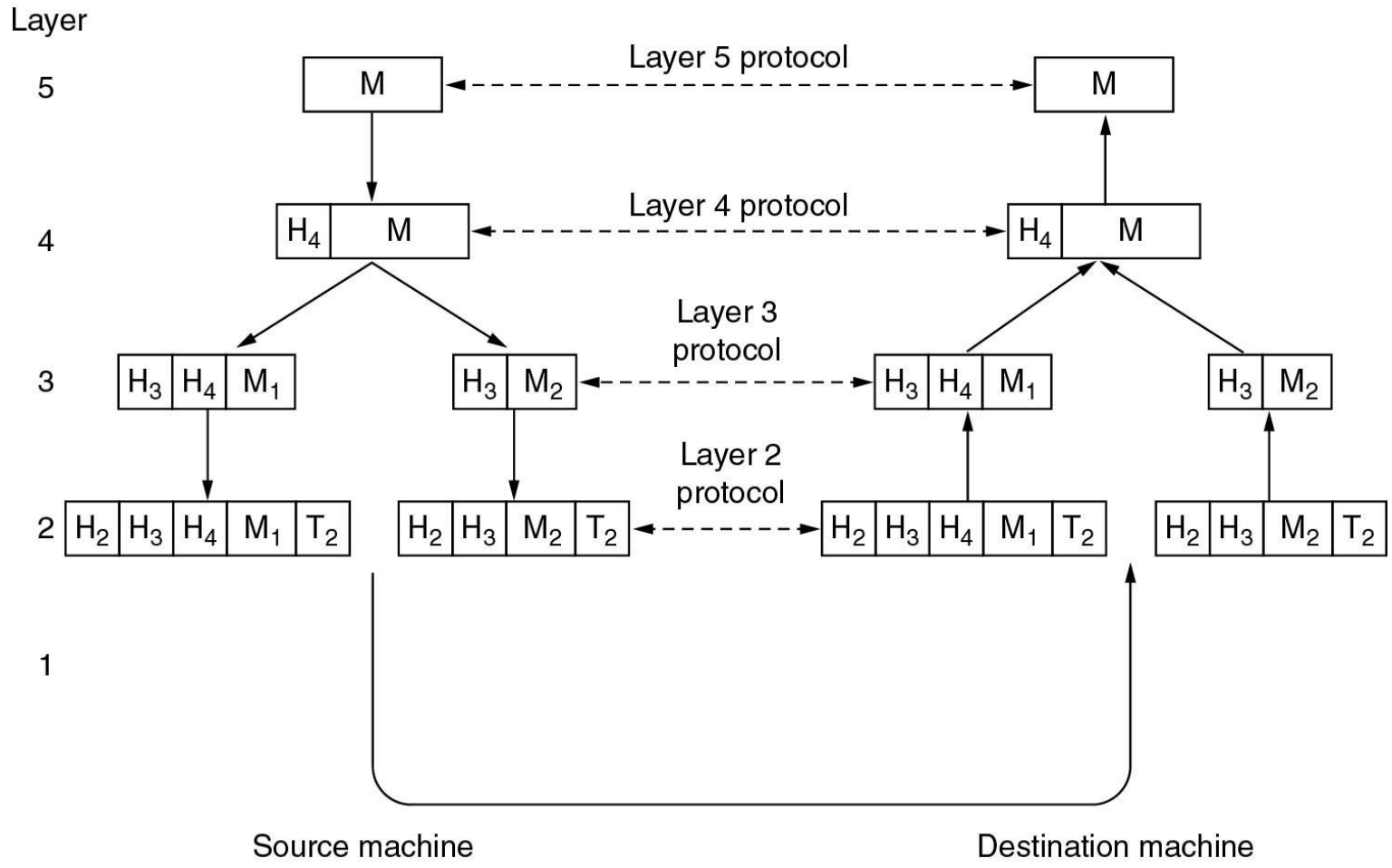
Sistemas operativos para redes de computadoras



Capa 3 (**capa de red**):

- Hay limitaciones en el tamaño de los mensajes de capa 3.
- Divide en **paquetes** los mensajes que llegan.
- A cada paquete se le coloca un encabezado.
- Decide cuál de las líneas que salen usar (cuando la máquina es un enrutador).
- Pasa los paquetes a la capa 2.

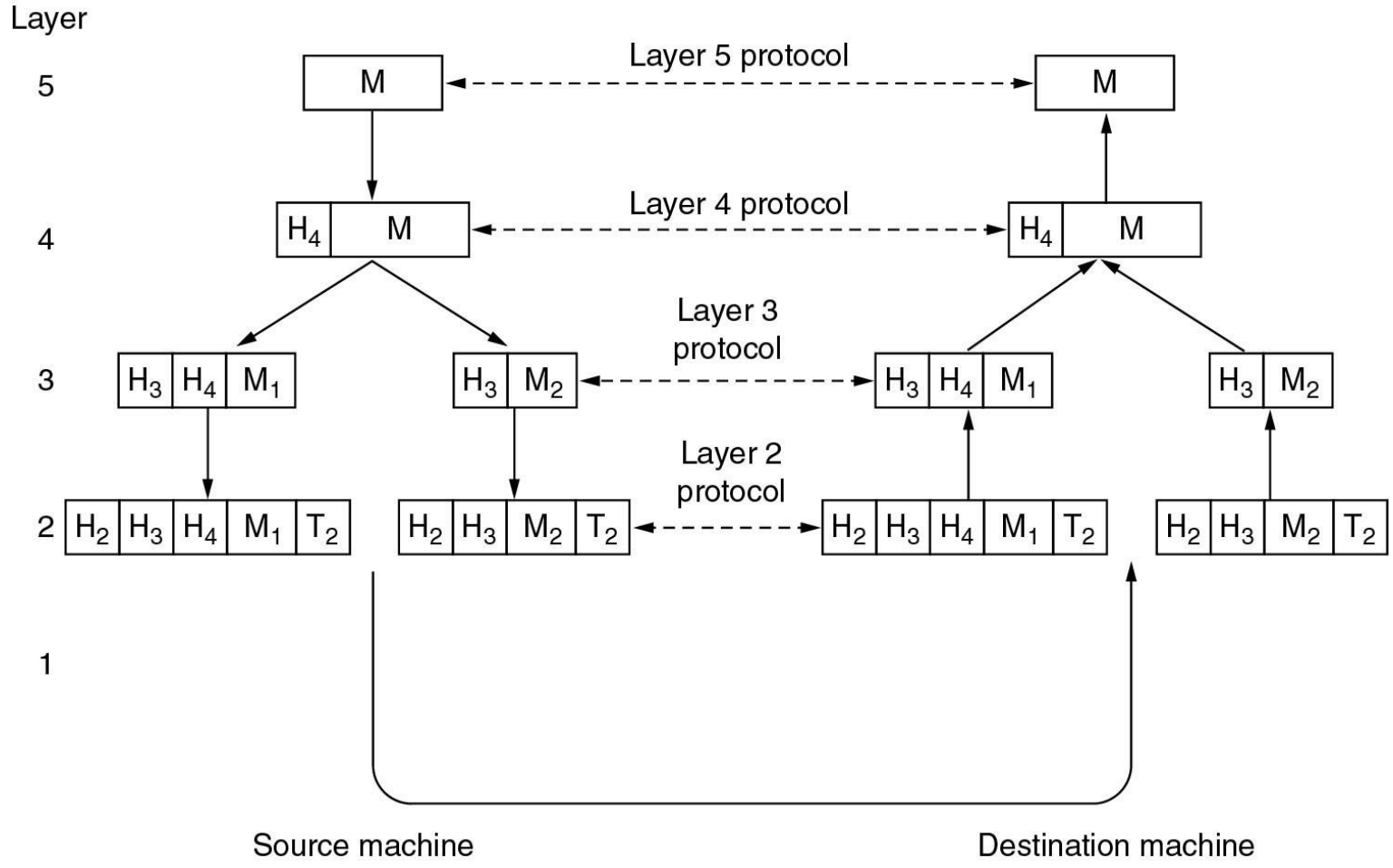
Sistemas operativos para redes de computadoras



La capa 2 (**capa de enlace de datos**)

- agrega un encabezado y un terminador, a cada pieza
- pasa la unidad resultante a la capa 1 para su transmisión.

Sistemas operativos para redes de computadoras



En la máquina receptora el mensaje pasa hacia arriba de capa en capa, perdiendo los encabezados conforme avanza.

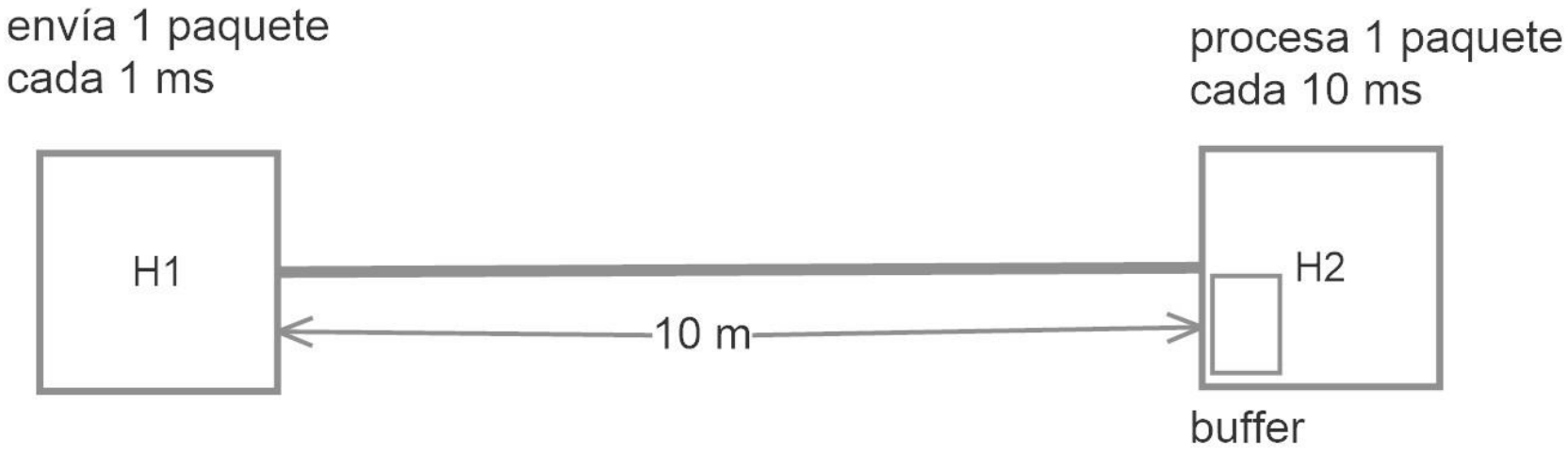
Problemas de las redes de computadoras

- Ahora vemos distintos **problemas a resolver** en las redes de computadoras.
- **Problema**: Hace falta un mecanismo para identificar a las máquinas de una red.
- **Solución**: Se usan direcciones para las máquinas.

Problemas de las redes de computadoras

- **Control de flujo**:
  - **Situación indeseable**: mensajes que llegan al receptor y se pierden.
  - **Esto sucede porque:**
  - un emisor rápido satura de datos al receptor hasta que este ya no puede almacenar más datos que le llegan y comienza a perder datos.

Problemas de las redes de computadoras



Problemas de las redes de computadoras

- **Control de flujo**:
  - **Problema: ¿Cómo evitar que un emisor rápido sature de datos a un receptor lento?**
  - **Idea de Solución**: Uso de **retroalimentación al emisor**.
    - O sea, indicarle cuándo y cuánto puede enviar.

Problemas de las redes de computadoras

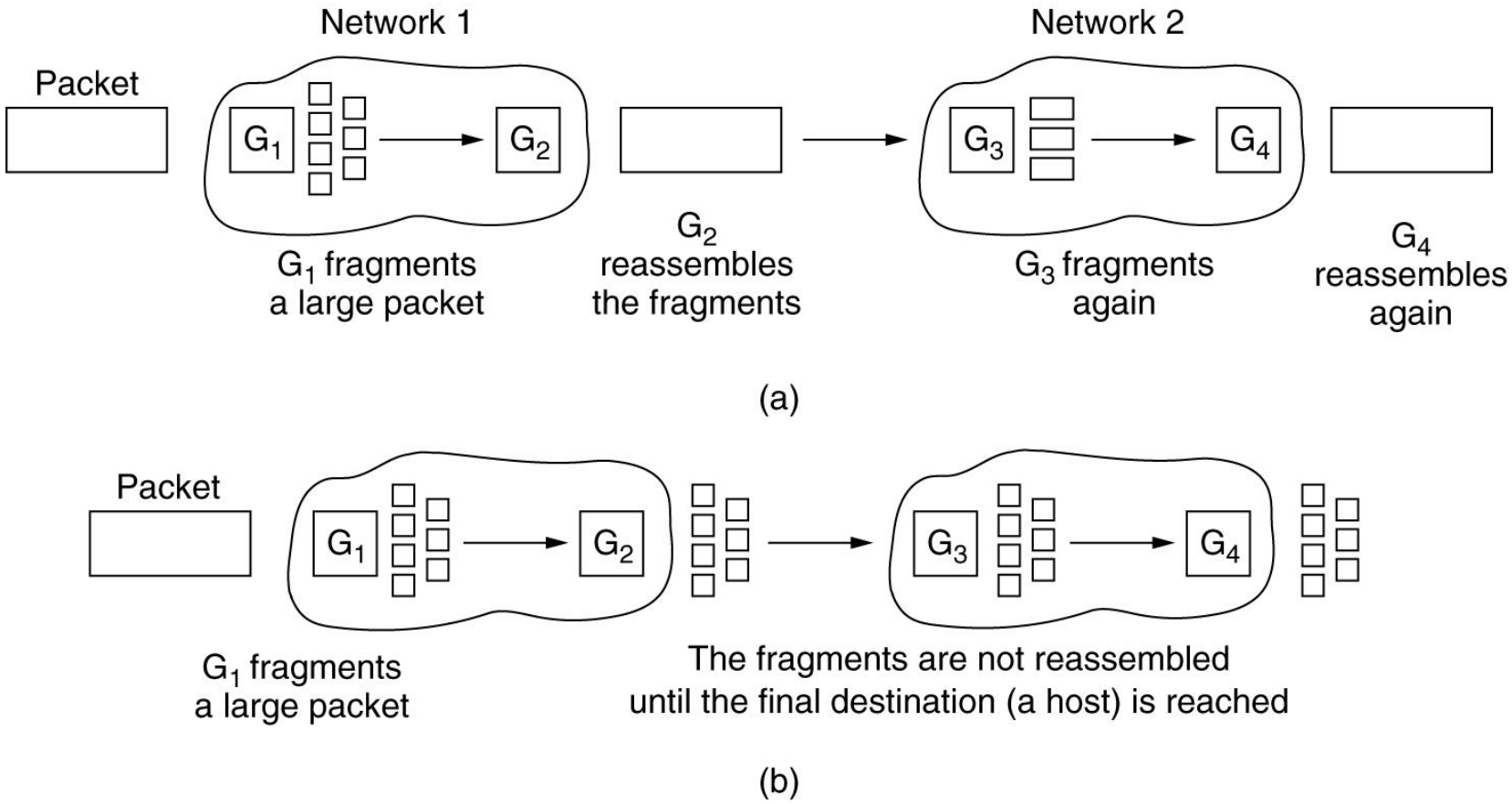
- **Situación**: Es común que las capas impongan un **tamaño máximo** a los mensajes.
  - En los encabezados de los protocolos suele haber campos de tamaño de mensaje.
  - Estos campos suelen tener una longitud fija, lo que implica un tamaño máximo de mensaje.
- **Problema**: los mensajes que llegan no pueden ser aceptados por un protocolo de una capa por ser demasiado grandes.
- **Esto puede pasar porque**
  - el mensaje demasiado grande llega de una red diferente que tiene un tamaño máximo de mensaje mayor al de la red actual.
    - Porque los procesos de la red actual son incapaces de aceptar mensajes que superan una cierta longitud.
  - También puede pasar cuando capas consecutivas aceptan distintos tamaños de mensajes.

Problemas de las redes de computadoras

- **Ejemplos del problema anterior**:
  - El tamaño de paquete de capa de enlace de datos es menor que el tamaño de paquete de la capa de red.
  - En una inter-red distintas redes aceptan distintos tamaños de paquetes máximos
- **Solución**: fragmentar los mensajes, transmitir fragmentos y reensamblar los mensajes.
- En una interred hay dos tipos de soluciones:

Problemas de las redes de computadoras

Fragmentación de mensajes



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

Problemas de las redes de computadoras

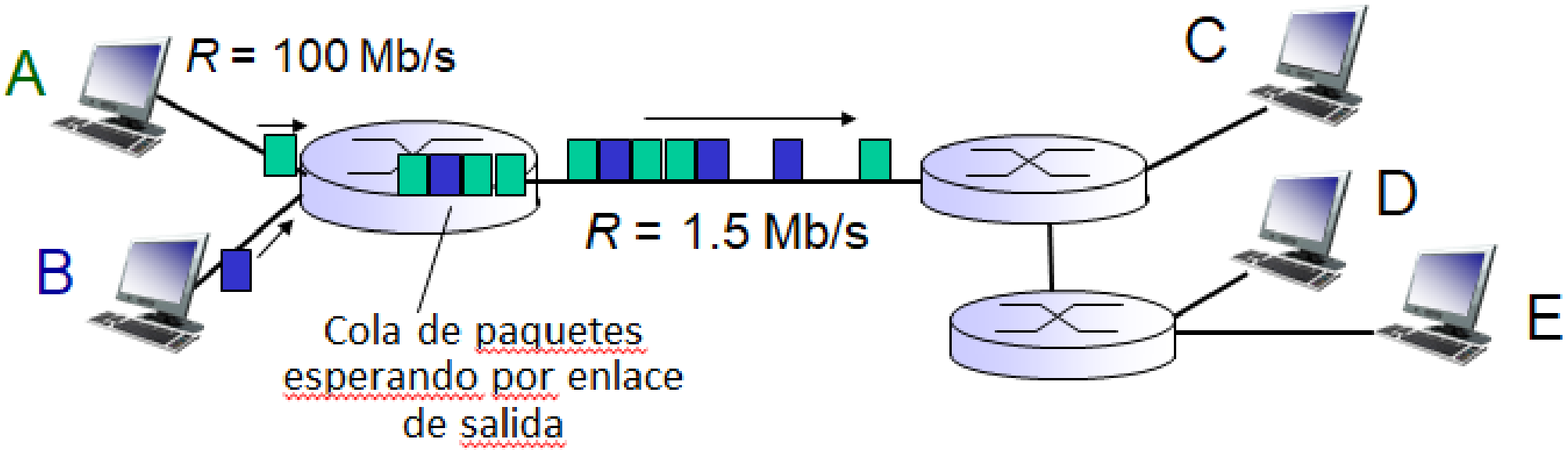
- **Ejemplo**: capa de red maneja tamaño de paquete más grande que capa de enlace de datos.
  - En el emisor el paquete de red es fragmentado en tramas al ser pasado a la capa de enlace de datos.
  - En el receptor las tramas pueden ser reensambladas para ser pasadas a la capa de red.

Problemas de las redes de computadoras

- **Problema**: A veces en la red hay que enviar demasiados mensajes por la misma línea de salida de un enrutador y esta se pone más lenta o no puede mandarlos a todos estos mensajes.
  - A esto se le llama **congestión**.
  - La congestión ocurre cuando la red no puede manejar la carga de paquetes que recibe de manera aceptable (esperas inaceptables o pérdida de paquetes)

Problemas de las redes de computadoras

Ejemplo: a un enrutador le llegan mensajes por una línea de entrada a una velocidad mayor que lo que puede enviarlos por la línea de salida.



Problemas de las redes de computadoras

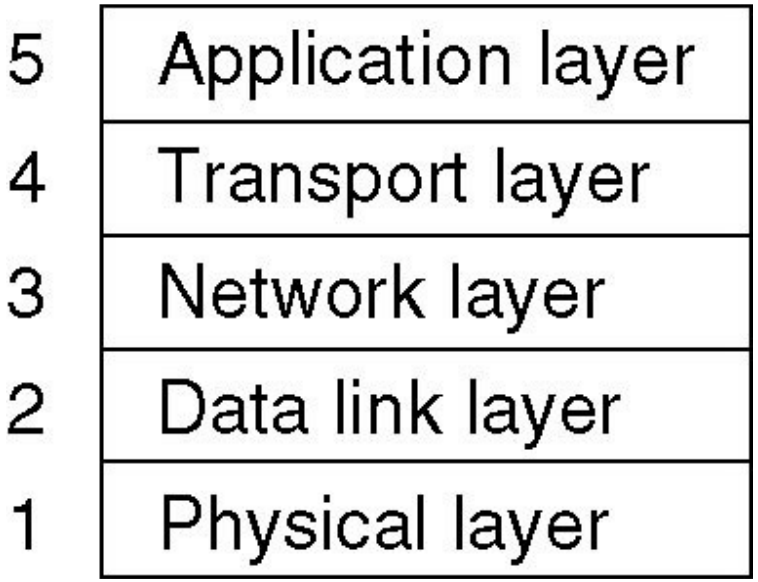
- **¿Cómo controlar la congestión?**
- **Idea de solución:** las computadoras emisoras se enteran de la congestión y reducen el tráfico de salida.

Capas de un SO de redes de computadoras

- A continuación vemos algunos **modelos de capas** de referencia para redes de computadoras.
  - Veremos el modelo híbrido de capas usado en la materia y el modelo de referencia TCP/IP.
- Después de eso estudiaremos las **capas de un sistema operativo de una red de computadoras** en **más detalle**.
  - Ejemplificaremos algunas de esas capas con la **internet** (**modelo de referencia TCP/IP**).
    - Veremos **por encima** algunos **protocolos de la internet**.
    - Más adelante en la materia veremos varios protocolos de la internet en **más detalle**.

Modelos de referencia de redes de computadoras

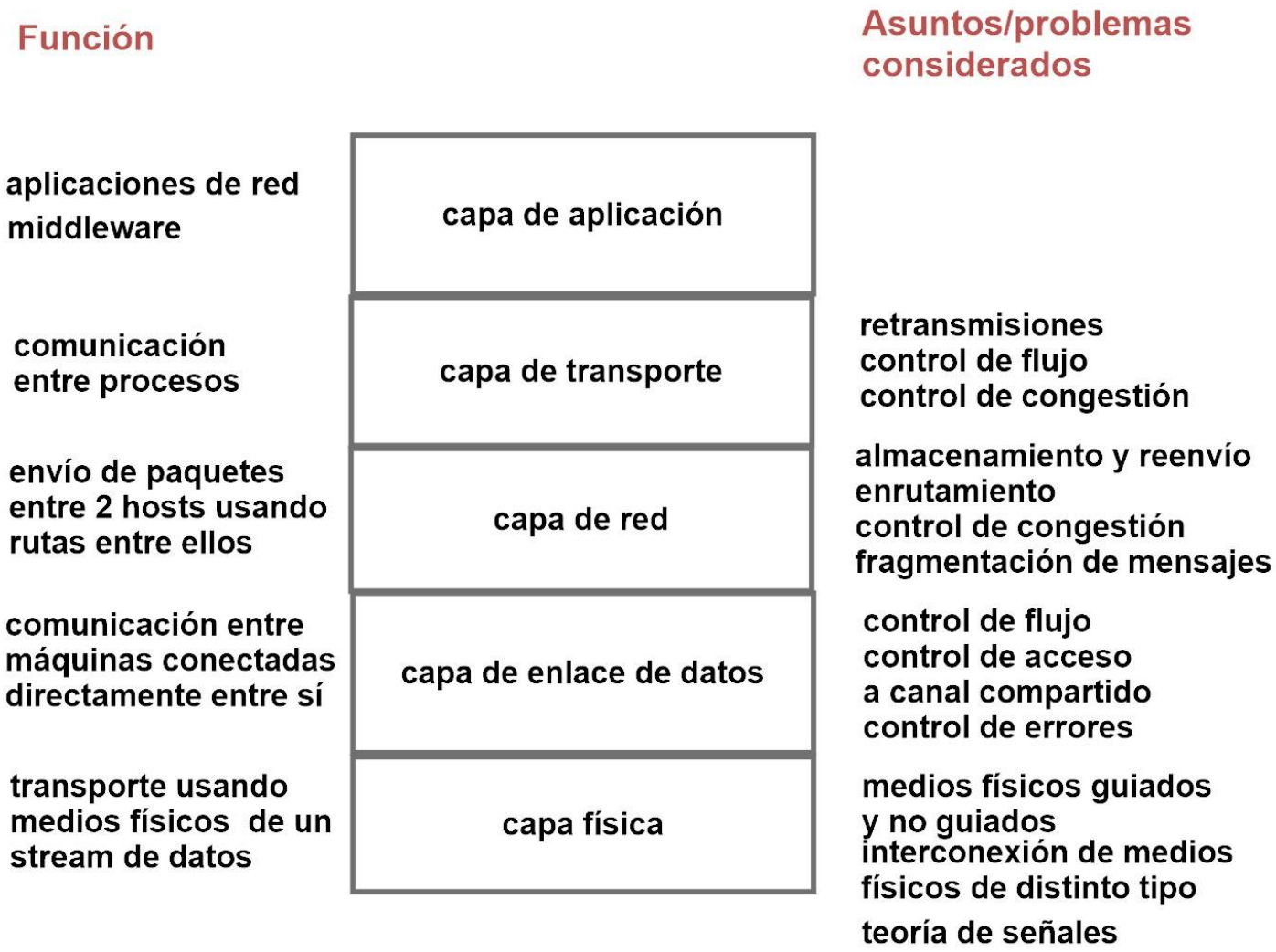
Modelo Híbrido



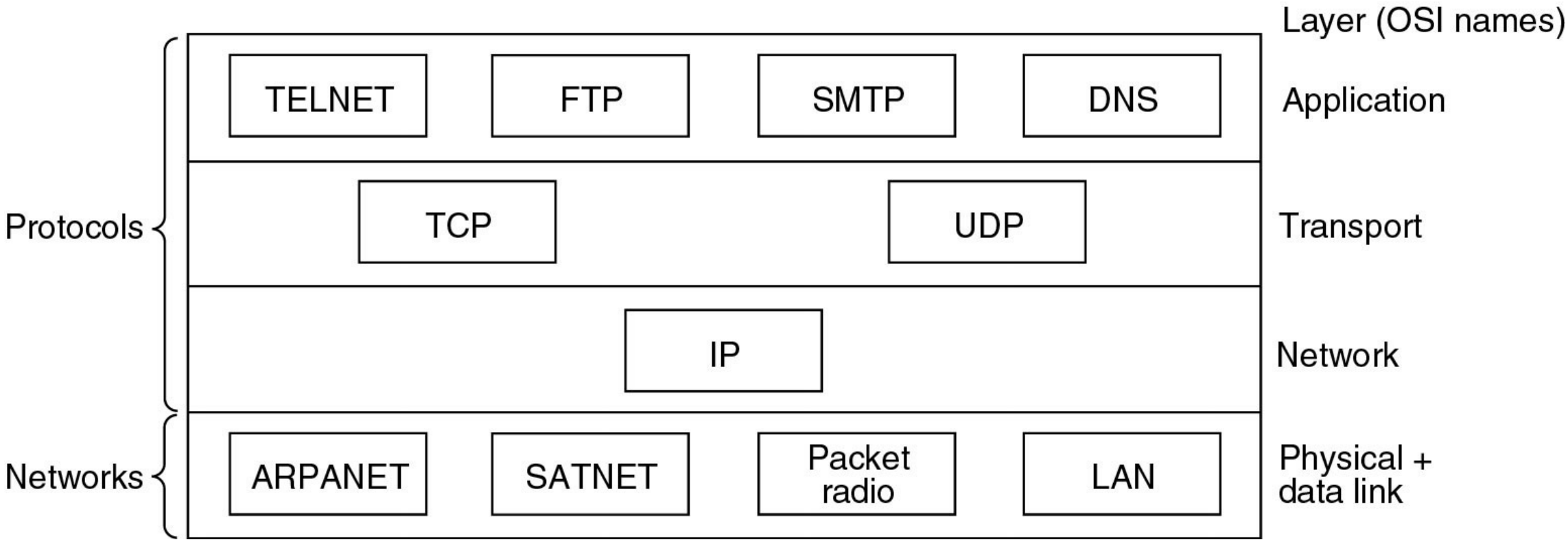
- En este curso usaremos un modelo híbrido con las siguientes capas:
  - Física, de enlace de datos, de red, de transporte y de aplicación.
- Nos concentraremos principalmente en el modelo TCP/IP para capas 3, 4 y 5.
  - **Orden a seguir en la materia: De arriba hacia abajo (Top_down).**
  - capa de aplicación -> capa de transporte -> capa de red-> capa de enlace de datos -> capa física.

Modelos de referencia de redes de computadoras

Modelo Híbrido



Capas de un SO de redes de computadoras



Modelo de referencia TCP/IP

Capa de aplicación

- En la capa de aplicación tenemos las **aplicaciones de red.**
- Cada aplicación de red ofrece un servicio específico con su propia forma de interfaz con el usuario.

Capa de aplicación

- Hay **dos opciones para desarrollar aplicaciones de red**:

1. El programador para especificar la comunicación usa **una interfaz para programas de aplicación (API)**.
   - Una API es conjunto básico de funciones a ser usadas.
   - La **socket API** es el estándar de facto para el software que se comunica sobre la internet.
     - Será tema de estudio en el taller de la materia.
   - **¿Qué utilidad tiene saber sobre el SOR subyacente?**
   - El conocimiento del SOR subyacente permite al programador escribir mejor código y desarrollar aplicaciones más eficientes.

Capa de aplicación

2. El programador se apoya en **middlewares** para construir la aplicación de red.
   - **Por ejemplo**: la web, y llamada a procedimientos remotos.
   - Una **middleware** provee servicios al software de la aplicación que
     - hacen más fácil a los desarrolladores implementar la comunicación y la entrada/salida de modo que
     - se pueden enfocar en el propósito específico de la aplicación.

La capa de aplicación: TCP/IP

- La **capa de aplicación en TCP/IP** contiene varios protocolos de nivel mas alto: transferencia de archivos (FTP), correo electrónico (SMTP), para resolución de nombres de host en sus direcciones de red (DNS), para la web (HTTP), etc.

La capa de aplicación: TCP/IP

- En la capa de aplicación de Internet es muy importante **la web** y los protocolos que la soportan: HTTP, HTTPS, DNS.
  - En la web están **los sitios web** que son conjuntos de páginas web entrelazadas.
    - **Por ejemplo:** es muy común que las empresas tengan sus sitios web.
  - También están las **aplicaciones web** a las que se accede por medio de un navegador (browser) y realizan funciones similares a las aplicaciones de escritorio.
    - **Por ejemplo**: generar páginas web, consultar datos y mostrarlos en pantalla, hacer operaciones que modifican el estado de los datos de la aplicación.

Capa de transporte

- La capa de red provee comunicación **entre hosts**
  - los paquetes de la comunicación entre los hosts siguen rutas elegidas por la capa de red.
- ***¿Con esto alcanza?***
- No, en la práctica la comunicación ocurre entre procesos.
  - La **capa de transporte** (**CT**) provee comunicación **entre procesos**.
- La CT **mejora** los servicios de la CR como veremos.
- La CT se ejecuta por completo en los hosts.
- **Entidad de transporte (ET)** = software/hardware de la CT.

Capa de transporte

- **Problemas que debería solucionar la CT:**
  - Uso de **temporizadores** y las **retransmisiones de paquetes**.
  - Uso de búferes y control de flujo.
  - Evitar congestionar la red poniendo demasiados paquetes en ella.
    - Cuando la CR pierde paquetes, la CT puede solucionarlo.
  - ***Si consideramos que hay procesos cliente y procesos servidor, surgen otros problemas:***
    - **El direccionamiento explícito de los destinos.**
      - ¿Cómo hacer para que un **proceso** adecuado **atienda a las necesidades** de una máquina **cliente**?
      - El proceso podría **no estar activo**, el cliente podría **no saber** cuál proceso usar, etc.

La capa de transporte: TCP/IP

- **Capa de transporte de internet:** Tiene dos protocolos:
  - **TCP**
    - TCP divide el flujo de bytes entrantes en mensajes discretos y pasa cada uno de ellos a la capa de interred.
    - TCP proporciona entrega confiable y en orden de los mensajes.
      - Esto significa que un flujo de bytes que se origina en una máquina se entregue **sin errores** en otra máquina en la interred.
    - **Reensamblado** de los mensajes recibidos en el receptor.
    - **Para la entrega confiable de mensajes**
      - los mensajes recibidos deben ser confirmados.
    - TCP también maneja el **control de flujo** y **el control de congestión.**

La capa de transporte: TCP/IP

- **UDP**
  - UDP proporciona entrega de mensajes no confiable y desordenada.
  - **Esto significa que**
    - Un mensaje puede entregarse con errores, o no entregarse, o varios mensajes pueden entregarse en forma desordenada.
  - **Cosas de TCP que no tiene UDP**
    - Uso de confirmaciones de recepción para los mensajes.
    - Control de flujo, control de congestión, retransmisiones cuando se recibe mensaje erróneo.
  - **UDP se usa para los siguientes tipos de aplicaciones:**
    - Aplicaciones que no usan el control de flujo ni la secuenciación de mensajes.
    - Aplicaciones que involucran consultas de solicitud-respuesta.
    - Aplicaciones de transmisión de voz y video.

Capa de Red

- Objetivos de la **capa de red (capa 3)**.
  - Algoritmos de almacenamiento y reenvío
  - **Control de congestión.**
  - Resolver problemas que surgen cuando un mensaje tiene que viajar por redes de distinta tecnología para llegar a destino.

Capa de Red

- Objetivos de la **capa de red (capa 3) - cont**.
  - **Enrutamiento**
    - **Situación indeseable**: un mensaje demora demasiado en llegar.
    - **Esto sucede porque**
      - en determinadas redes (p.ej. WAN, internet, etc.) hay múltiples rutas entre el origen y el destino
        - Y justo se toma una ruta demasiado lenta/larga entre origen y destino
    - **Entonces hay que resolver el siguiente problema:**
    - **Problema**: ¿Cuando hay múltiples rutas entre el origen y el destino cómo elegir la mejor o las mejores?
      - De esto se encargan los **algoritmos de enrutamiento.**

La capa de red: TCP/IP

- **Capa de interred**:
  - Permite que los hosts inyecten paquetes dentro de cualquiera de las redes en la interred.
  - Los paquetes diferentes entre dos host viajan a su destino de manera independiente.
  - **Una consecuencia de esto es que**
    - los paquetes pueden llegar en un orden distinto al cual fueron enviados.
  - **Cuando los paquetes llegan fuera de orden**
    - las capas mas altas deberán ordenarlos, si se desea una entrega ordenada.

La capa de red: TCP/IP

- **Para distinguir entre las diferentes máquinas que tienen una conexión a internet se usan direcciones IP.**
  - **Direcciones IPv4 (de 32 bits)**
    - 4 números entre 0 y 255 separados por '.' .
    - P. ej. 200.45.191.35
  - **Direcciones IPv6 (de 128 bits)**
- **Se envían paquetes IP (paquetes IPv4 o paquetes IPv6).**
- **Paquetes IP.** (tienen su propio formato como veremos más adelante.)
- **Para hacer el enrutamiento**
  - hay protocolos de enrutamiento: se usan **OSPF** y **BGP** para enrutamiento de paquetes.

Capa de Enlace de Datos

- Objetivo de la **capa de enlace de datos – CED- (capa 2)**
  - transformar un medio de transmisión puro en una línea de comunicación que aparezca libre de errores de transmisión.
- **Problemas de diseño que se consideran:**
  - Fragmentación de paquetes en **tramas**, cuando un paquete es demasiado grande para ser aceptado por la CED.
    - Transmisión de las tramas de manera secuencial.
  - **Tramas de confirmación de recepción** son usadas cuando el servicio es confiable.
  - **Control de flujo**.
    - Para evitar que un emisor rápido sature a un receptor lento.
  - **Control de acceso a un canal compartido**: se busca manejar y minimizar o evitar colisiones.
    - Una colisión sucede cuando varias máquinas comparten un mismo canal de comunicaciones y envían tramas en simultáneo usando ese canal.

Capa de Enlace de Datos

- **Control de errores**
  - **Situación indeseable**: los mensajes llegan con errores
  - **Esto sucede porque**
    - el medio físico de comunicaciones es imperfecto y ocasiona errores
  - **Enfoques para encarar el problema**
    - Identificación errores y retransmisión de mensajes erróneos
    - Corrección de errores en el mensaje recibido
  - Para poder aplicar uno de los enfoques los mensajes deben llevar cierta **información extra** con el propósito de control de errores.

Capa Física

- **El propósito de la capa física (CF) es**
  - transportar un stream de datos de una máquina otra usando medios físicos.
- **¿Cuáles eran los medios de transmisión físicos?**
  - **Por ejemplo**, cable de cobre, fibra óptica, microondas, celular, etc.
- **La CF no consiste solo de medios físicos**
  - los medios físicos **se conectan entre sí** usando dispositivos como codecs, modems, multiplexores, demultiplexores, etc.

Capa Física: medios físicos

- bit: se propaga entre pares de transmisor/receptor
- Enlace físico: lo que yace entre el transmisor & receptor
- Medios guiados:
  - Las señales se propagan en medios sólidos: cobre, fibra óptica, coaxial.
- Medios no guiados:
  - Las señales se propagan libremente, e.g., radio

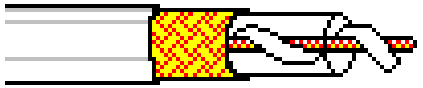
Par trenzado (TP)

- 2 cables de cobre aislados
  - Category 5: 100 Mbps, 1 Gbps Ethernet
  - Category 6: 10Gbps

Capa Física: medios físicos

coaxial cable:

- 2 conductores concéntricos de cobre
- bidireccionales
- broadband:
  - Múltiples canales en el cable



Cable de fibra óptica:

- Fibra de vidrio que transporta pulsos de luz, cada pulso es un bit
- Operan a alta velocidad:
  - high-speed point-to-point transmission (e.g., 10' s-100' s Gbps transmission rate)
- Baja tasa de errores:
  - repeaters spaced far apart
  - immune to electromagnetic noise

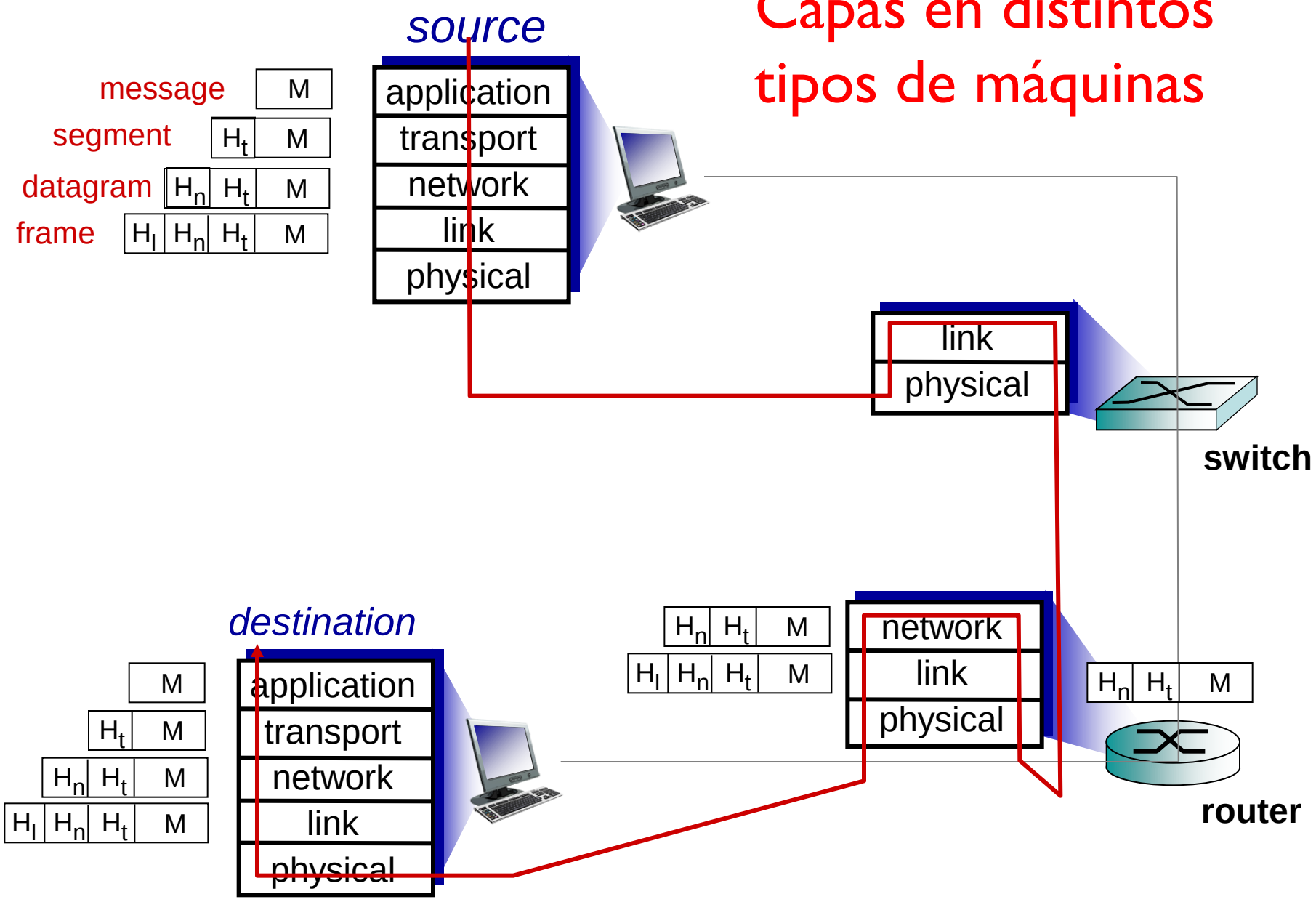
Capa Física: medios físicos

- **Radio**: señal transportada en el espectro electromagnético.
- No se usa cable físico.
- Es bidireccional.
- Efectos de propagación en el entorno:
  - Reflección
  - Obstrucción por objectos.
  - Interferencia

Tipos de enlaces de radio:

- Microondas terrestres
  - Por ej. canales de hasta 45 Mbps.
- LAN (e.g., Wi-Fi)
  - 54 Mbps
- Área amplia (e.g., cellular)
  - 4G cellular: ~ 10 Mbps
- satélite
  - Canal de 45Mbps (o varios canales más chicos)
  - 270 msec de demora entre extremos.
  - Geosíncronos versus baja altitud.

Capas en distintos tipos de máquinas



Sistemas operativos para la nube

- Antes de ver los detalles de la pila de protocolos en la nube,
  - veremos algunos **conceptos importantes** sobre el cómputo en la nube.
  - Esto va a **facilitar la comprensión** de las diferentes capas.

Cómputo en la Nube (Cloud)

- **Infraestructura como-servicio (IaaS)**:
  - Ambiente formado por *recursos informáticos básicos* que pueden ser accedidos/manejados vía interfaces basadas en servicios de la nube y en herramientas.
  - **Incluye**: servidores, almacenamiento, redes.
  - Los usuarios pueden aprovisionar, configurar y gestionar sus propios recursos informáticos.
  - Se pueden escalar los recursos según la necesidad del negocio.
  - Los proveedores del IaaS son responsables del mantenimiento del hardware.
- **Plataforma como servicio (PaaS)**:
  - Plataforma completa para que los desarrolladores creen, desplieguen y gestionen aplicaciones sin preocuparse por la infraestructura subyacente
  - Proporciona herramientas y servicios para el desarrollo de aplicaciones, como bases de datos, middleware y entornos de ejecución.
  - El proveedor se encarga del mantenimiento del entorno de desarrollo.
  - P.ej: Google Cloud Compute Engine ofrece infraestructura escalable para ejecutar aplicaciones en la nube.
  - P.ej: Microsoft Azure Virtual Machines: permite crear y gestionar máquinas virtuales en la nube.

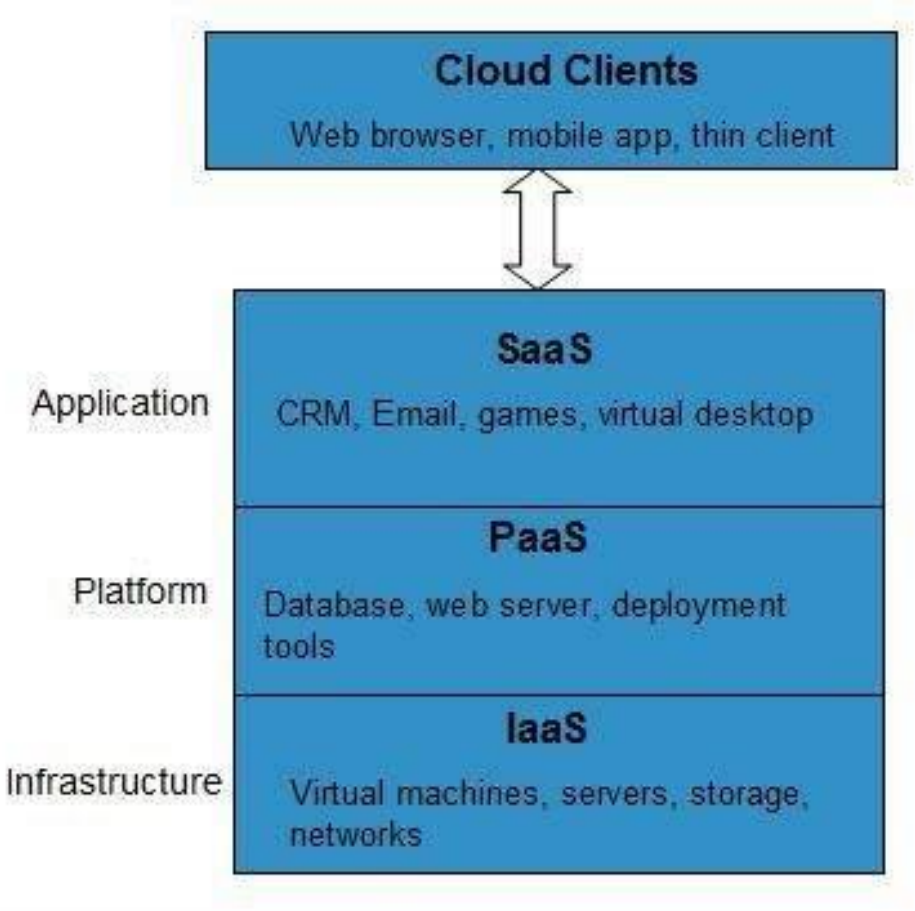
Cómputo en la Nube (Cloud)

- **Software como servicio:**
  - Permite a los usuarios acceder a aplicaciones completas alojadas en la nube mediante una suscripción, sin necesidad de instalación o mantenimiento local.
  - Los usuarios pueden acceder al software desde cualquier dispositivo con conexión a internet.
  - El proveedor se encarga de todas las actualizaciones, mantenimiento y seguridad del software.
  - P.ej: MS Office 365, Google Apps, etc.

Cloud Delivery Model	Common Cloud Consumer Activities	Common Cloud Provider Activities
SaaS	uses and configures cloud service	implements, manages, and maintains cloud service monitors usage by cloud consumers
PaaS	develops, tests, deploys, and manages cloud services and cloud-based solutions	pre-configures platform and provisions underlying infrastructure, middleware, and other needed IT resources, as necessary monitors usage by cloud consumers
IaaS	sets up and configures bare infrastructure, and installs, manages, and monitors any needed software	provisions and manages the physical processing, storage, networking, and hosting required monitors usage by cloud consumers

Cómputo en la Nube (Cloud)

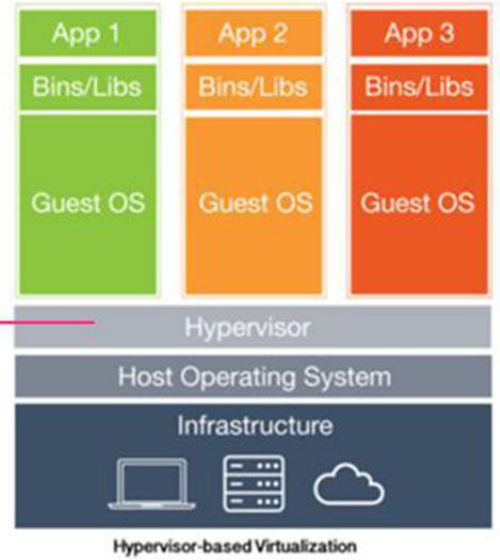
- **Servicios**
  - Infrastructure-as–a-Service (**IaaS**)
    - Amazon EC2, W. Azure, Google Compute Engine
  - Platform-as-a-Service (**PaaS**)
    - Heroku, Apache Stratos
  - Software-as-a-Service (**SaaS**)
    - Google Apps, Microsoft Office 365



Cómputo en la Nube (Cloud)

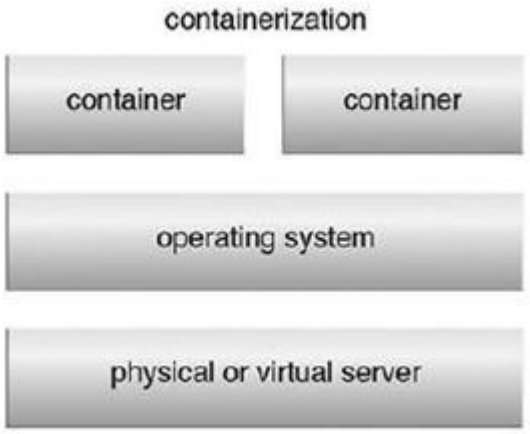
- **Virtualización**:
  - La virtualización permite dividir un servidor físico en varias máquinas virtuales, donde cada una es capaz de ejecutar su propio sistema operativo (sistema operativo invitado) y aplicaciones.
  - **Hipervisor**: software especializado que permite que múltiples instancias (o máquinas virtuales) se ejecuten en un solo servidor físico.
  - Tanto el sistema operativo invitado y el software de aplicación ejecutando en servidor virtual no son conscientes del proceso de virtualización.
  - Si una máquina virtual falla, no afecta a las demás.
  - Se pueden agregar o eliminar máquinas virtuales según se necesite.

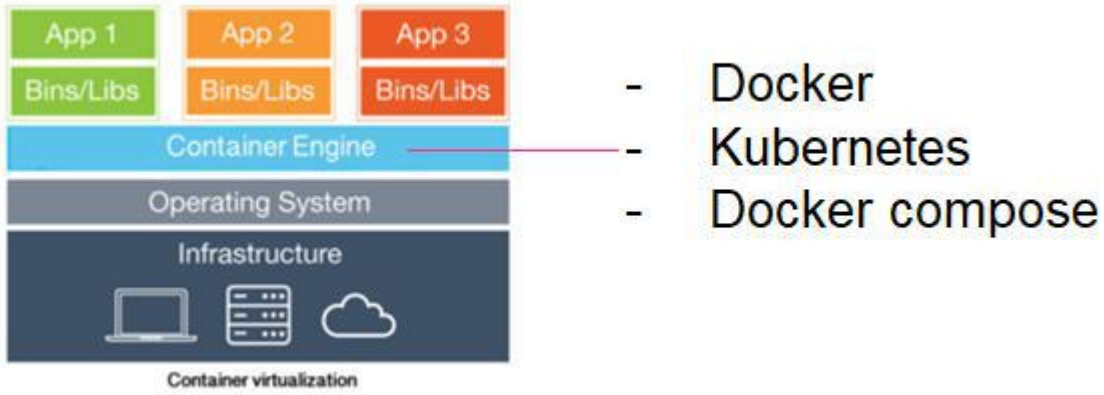
- Virtual Box
- VMware



Cómputo en la Nube (Cloud)

- **Containerización**:
  - Se **empaqueta** el código de la aplicación junto con los archivos de configuración relacionados, librerías y dependencias requeridas en para que pueda ejecutar.
    - Este paquete de software o **contenedor** se *abstrae del SO* y es portable.
  - Las aplicaciones son *desplegadas en contenedores*. Cada contenedor ejecuta en un proceso.
  - Usar contenedores permite a varios servicios de la nube ejecutarse como un servidor (físico o virtual) único mientras se accede al mismo SO.
  - Los contenedores pueden ejecutarse en cualquier plataforma que soporte el motor de contenedores.
  - Si un contenedor falla, no afecta a los otros.
  - Como comparten el núcleo de un sistema operativo, los contenedores requieren menos recursos que las máquinas virtuales.

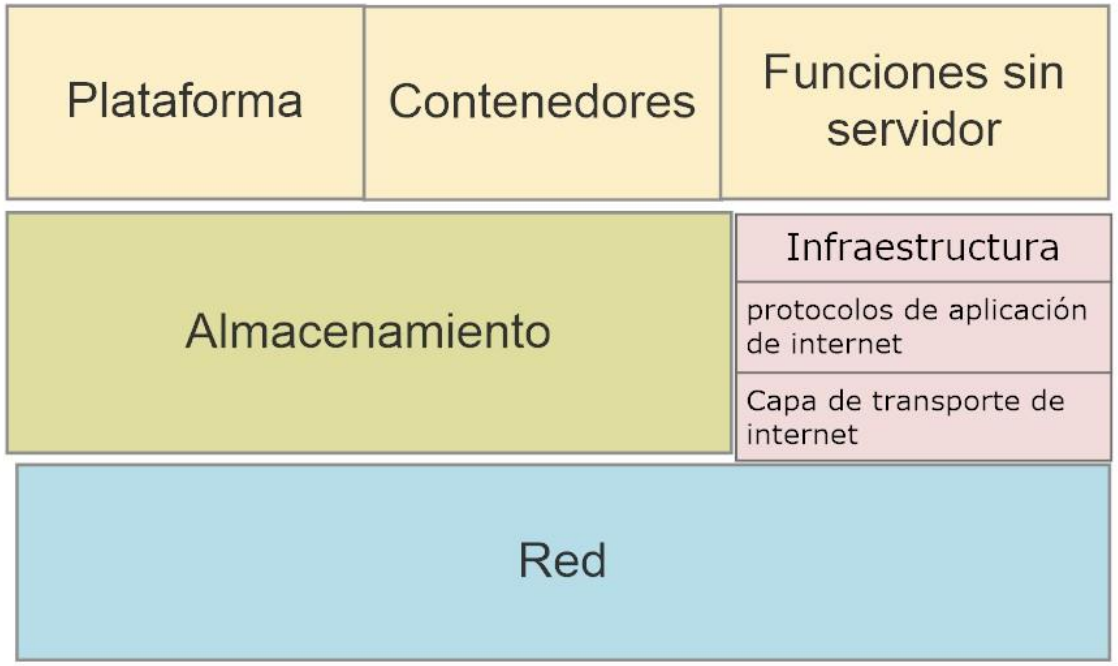




- Docker
- Kubernetes
- Docker compose

Sistemas operativos para la nube

- **La capa de red** está formada por protocolos que facilitan la conectividad y el enrutamiento dentro de la infraestructura de red de cada proveedor.
  - También se preocupa de la seguridad de estas redes.
  - **Tipos de protocolos usados:**
    - **Protocolos de internet**: Son importantes para que lo datos puedan moverse entre redes conectadas a través de internet.
      - Se usan IP y BGP.



Sistemas operativos para la nube

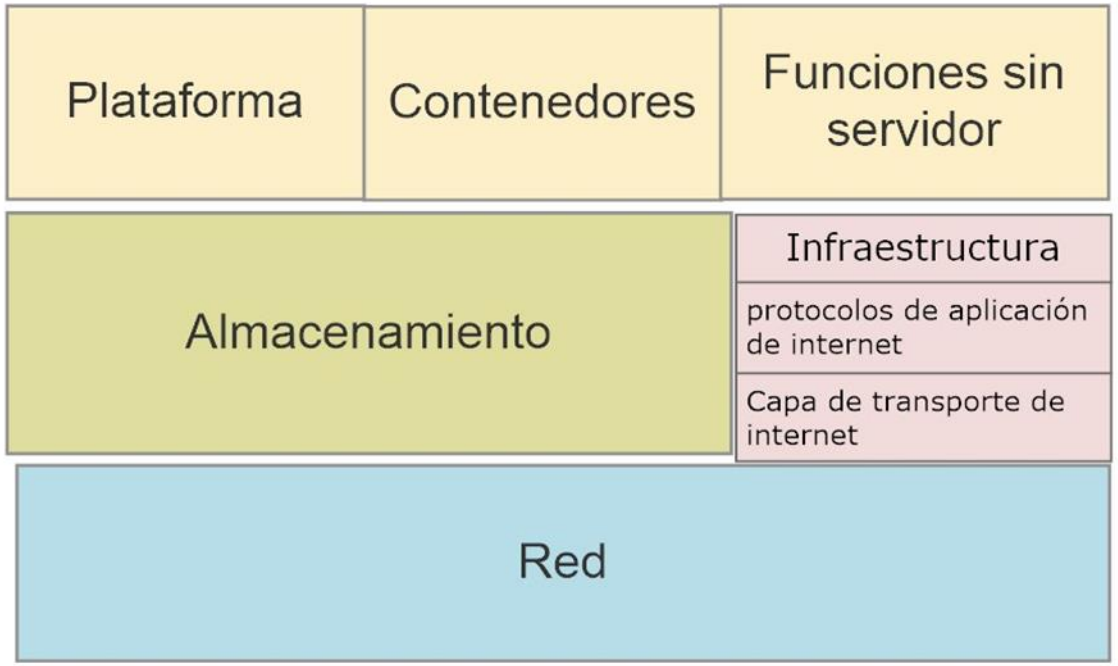
La capa de red – cont:

- **Tipos de protocolos usados - cont:**
  - **Protocolos para redes privadas virtuales (VPN):** Las VPN permiten establecer conexiones seguras entre las redes del cliente y la nube a través de Internet. Los protocolos más comunes incluyen:
    - **OpenVPN**: Cifra los datos y asegura que viajen protegidos a través de una conexión pública.
    - **WireGuard**: también cifra las conexiones VPN, ofreciendo mayor velocidad y simplicidad.

- **Protocolos para conexiones privadas entre el cliente y proveedor de la Nube**: Cuando las empresas necesitan conexiones más rápidas y seguras, pueden optar por métodos privados que no usan Internet. Por ejemplo:
  - **MPLS**: crea rutas privadas dedicadas para enviar datos directamente entre las instalaciones del cliente y el proveedor de nube
  - **Túneles VPN:** permiten crear una conexión segura sobre Internet público hacia el proveedor de nube, utilizando protocolos como IPsec o L2TP.
  - **Conexiones dedicadas**: establecen líneas privadas entre el centro de datos del cliente y la nube.
    - Estas conexiones son más rápidas, confiables y seguras que las basadas en Internet público.
    - **Por ejemplo**: Direct Connect de AWS, ExpressRoute de Azure.

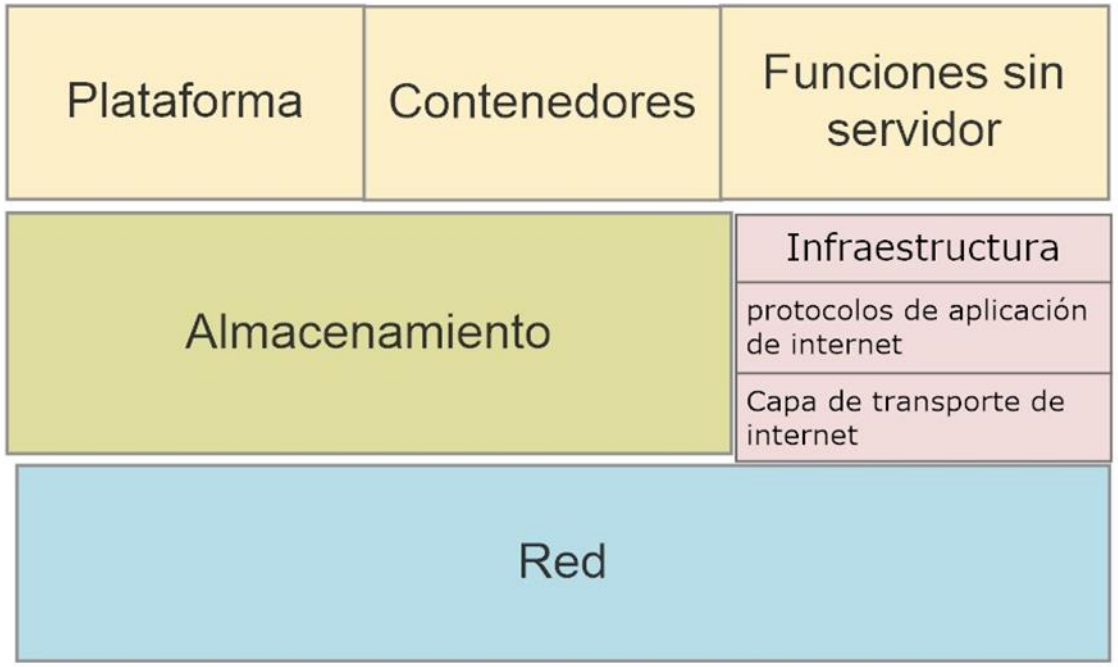
Sistemas operativos para la nube

- La **capa de infraestructura**: Se refiere a los protocolos básicos que permiten la comunicación y transferencia de datos entre servidores y clientes.
  - Incluye los siguientes tipos de protocolos:
    - Protocolos de capa de aplicación de internet como HTTP, HTTPS, SFTP, FTP.
    - Protocolos de capa de transporte de internet como TCP, UDP.



Sistemas operativos para la nube

- **Capa de almacenamiento**: consiste de protocolos usados para acceder y gestionar **datos almacenados** en la nube.
  - Se usan protocolos diferentes para diferentes tipos de almacenamiento.
  - Aclarado en filmina siguiente.

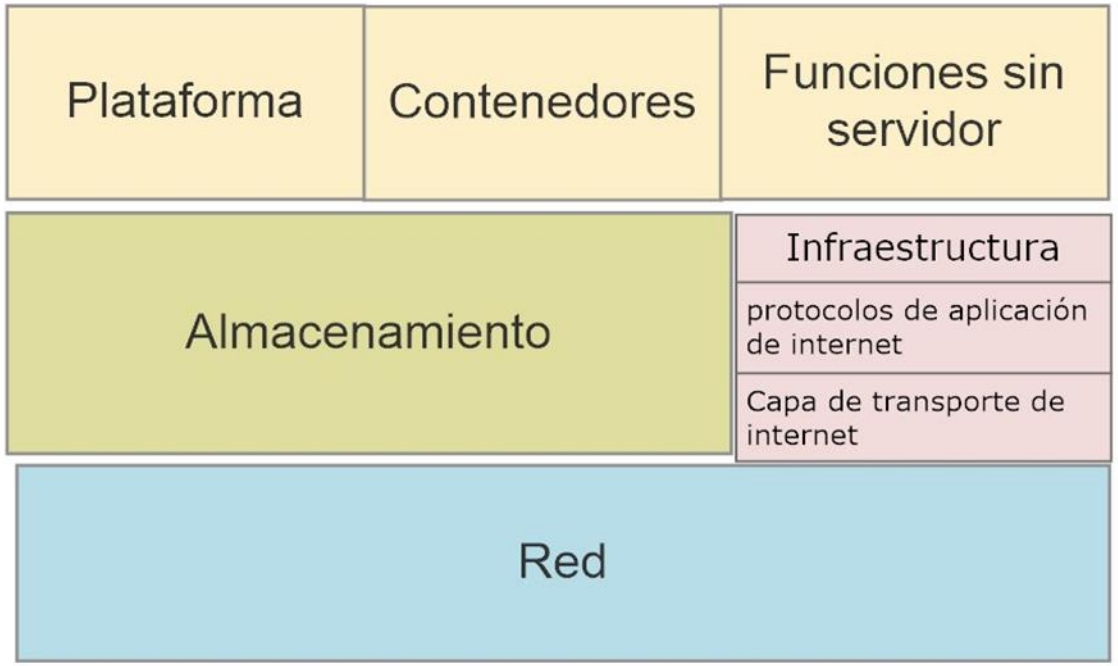


Sistemas operativos para la nube

- **Capa de almacenamiento – cont**:
  - Los **tipos de almacenamiento** pueden ser:
    - **En bloque**: se almacenan datos en bloques individuales. P.ej: para bases de datos o sistemas de gestión de contenido. P.ej. iSCSI permite acceder a dispositivos de almacenamiento en bloque a través de redes TCP/IP.
    - **De archivos**: permite acceso a sistema de archivos (archivos y carpetas). P.ej. NFS.
    - **De objetos**: Para grandes cantidades de datos no estructurados como:
      - Archivos multimedia (audio, video) – p.ej. se usa HTTP/HTTPS y REST API.
      - pdf, hojas de calculo, presentaciones,
      - **Copias de seguridad**: imágenes de disco, backups de bases de datos, etc.
      - **Datos de IoT**: como sensores y cámaras.
      - Archivos de registro (log)

Sistemas operativos para la nube

- **Capa de plataforma**:
  - es un **conjunto de herramientas y servicios** que los desarrolladores pueden usar para *construir y ejecutar aplicaciones*
    - sin tener que preocuparse por los detalles técnicos de cómo funcionan los servidores o la infraestructura detrás de escena.
  - Esta capa permite que las aplicaciones se *conecten y trabajen con servicios en la nube* (como bases de datos, almacenamiento o procesamiento) utilizando **APIs**.
    - Una API es simplemente un "puente" que permite que dos sistemas se comuniquen entre sí.

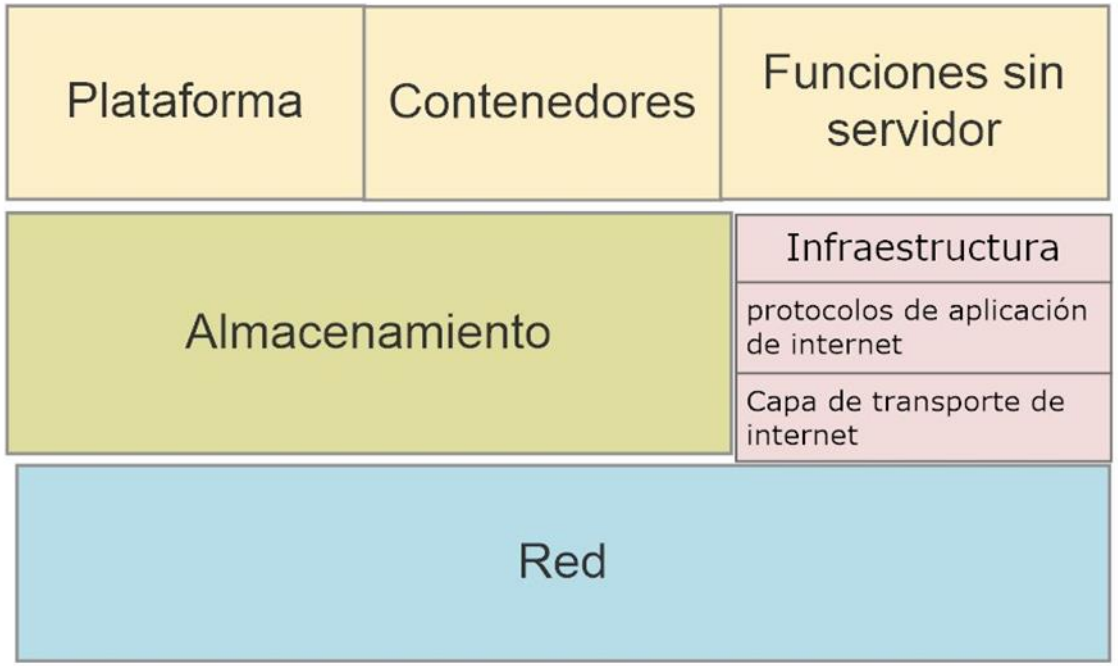


Sistemas operativos para la nube

- **Capa de plataforma – cont:**
  - **Facilidades**: Esta capa proporciona todo lo necesario para que los desarrolladores puedan:
    - **Escribir el código** de sus aplicaciones.
    - **Probarlas** para asegurarse de que funcionan correctamente.
    - **Desplegarlas** (es decir, ponerlas en funcionamiento para los usuarios).
  - **Utilidad:** Gracias a esta capa, los desarrolladores no tienen que preocuparse por configurar servidores, gestionar redes o resolver problemas complejos de infraestructura.
    - Pueden enfocarse únicamente en crear las funcionalidades y características de su aplicación.
  - **Protocolos usados en esta capa**:
    - **REST**: se usa para que las aplicaciones pidan o envíen información a los servicios en la nube. P.ej. se puede usar REST para guardar datos en una base de datos en la nube.
    - **gRPC** (Google remote procedure call): permite a los programas ejecutar funciones o procedimientos en servidores remotos como si estuvieran en la misma máquina local.

Sistemas operativos para la nube

- **Capa de funciones sin servidor**: Involucra protocolos que permiten ejecutar código en respuesta a eventos sin necesidad de gestionar servidores, ni la infraestructura..
  - Es ideal para aplicaciones que requieren escalabilidad rápida y eficiente.
  - **Ejemplos**: Microsoft Azure Functions, AWS Lambda.

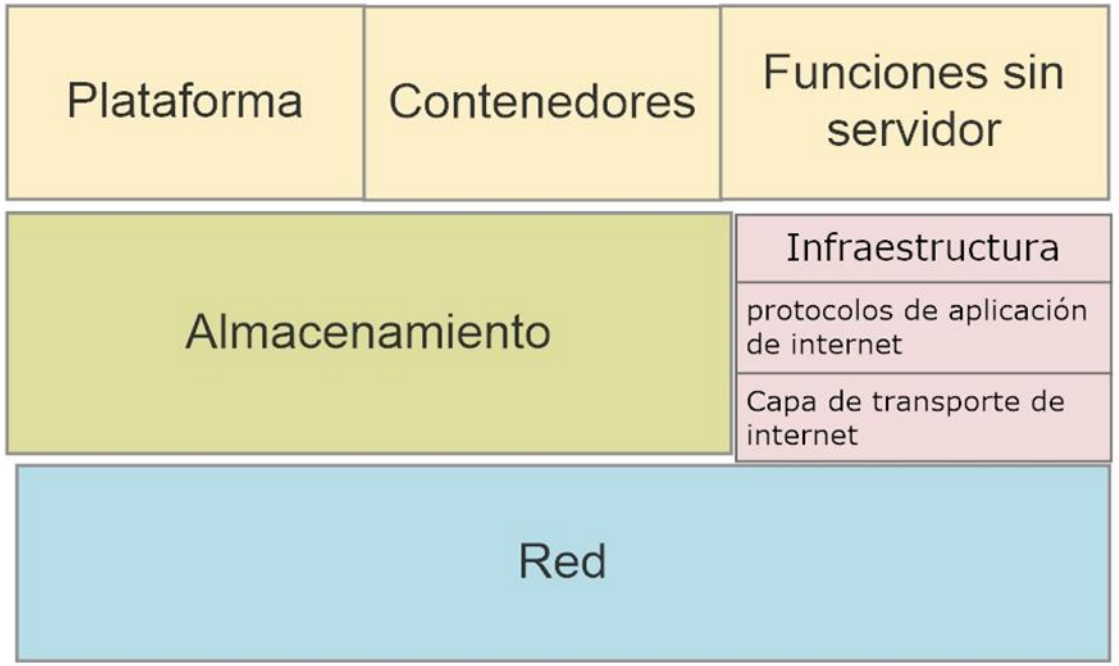


Sistemas operativos para la nube

- **Capa de funciones sin servidor – cont**: los eventos son fundamentales para activar y ejecutar las funciones.
  - **Tipos de eventos más comunes:**
    - **Eventos HTTP**: cuando usuario hace *solicitud a través de una API o un browser*.
    - **Cambios en la base de datos**: cuando hay *modificaciones en los datos*.
    - **Eventos de almacenamiento en la nube**: cuando se hacen *acciones sobre objetos almacenados;* p.ej: como cargar, eliminar o modificar archivos.
    - **Eventos de mensajería**: sistemas de mensajería o colas que mandan notificaciones sobre nuevos mensajes o tareas.
    - **Eventos de IoT:** cuando dispositivos IoT *mandan datos o notificaciones*.
    - **Eventos a intervalos regulares**: evento de que pasó un tiempo *T.*
    - **Eventos de aplicaciones externas**: aplicaciones externas *mandan notificaciones sobre eventos relevantes.*

Sistemas operativos para la nube

- **Capa de contenedores**: protocolos y tecnologías usadas para la *gestión y orquestación de contenedores* en la nube.
  - Los **contenedores** permiten *empaquetar las aplicaciones y sus dependencias*, lo que facilita su despliegue y escalabilidad.
  - Se refiere a la gestión y orquestación de aplicaciones en la nube usando tecnologías como **Kubernetes** y **Docker**.



Ejemplos de proveedores de la nube

Proveedor	Capa	Protocolos Usados
Amazon Web Services (AWS)	**Infraestructura**	TCP/IP, HTTP, HTTPS, FTP, SFTP
	Plataforma	REST, SOAP, WebSocket
	Almacenamiento	S3 API, NFS, SMB, iSCSI
	Redes	VPC (Virtual Private Cloud), Direct Connect, VPN, Route 53 (DNS)
	Seguridad	IAM (Identity and Access Management), KMS (Key Management Service), SSL/TLS
	Contenedores	Docker, Kubernetes (EKS), AWS Fargate
	Funciones sin servidor	AWS Lambda (event-driven architecture)

Ejemplos de proveedores de la nube

Microsoft Azure	**Infraestructura**	TCP/IP, HTTP, HTTPS, FTP
	Plataforma	REST, OData, gRPC
	Almacenamiento	Blob Storage REST API, Azure Files (SMB), Azure Data Lake Storage
	Redes	Azure Virtual Network, ExpressRoute, Azure DNS
	Seguridad	Azure Active Directory (AAD), Key Vault
	Contenedores	Azure Kubernetes Service (AKS), Azure Container Instances
	Funciones sin servidor	Azure Functions

Ejemplos de proveedores de la nube

Google Cloud Platform (GCP)	**Infraestructura**	TCP/IP, HTTP, HTTPS
	Plataforma	REST, gRPC
	Almacenamiento	Cloud Storage JSON API, Filestore (NFS)
	Redes	Google Virtual Private Cloud (VPC), Cloud Interconnect
	Seguridad	Cloud IAM, Cloud KMS
	Contenedores	Google Kubernetes Engine (GKE), Cloud Run
	Funciones sin servidor	Google Cloud Functions

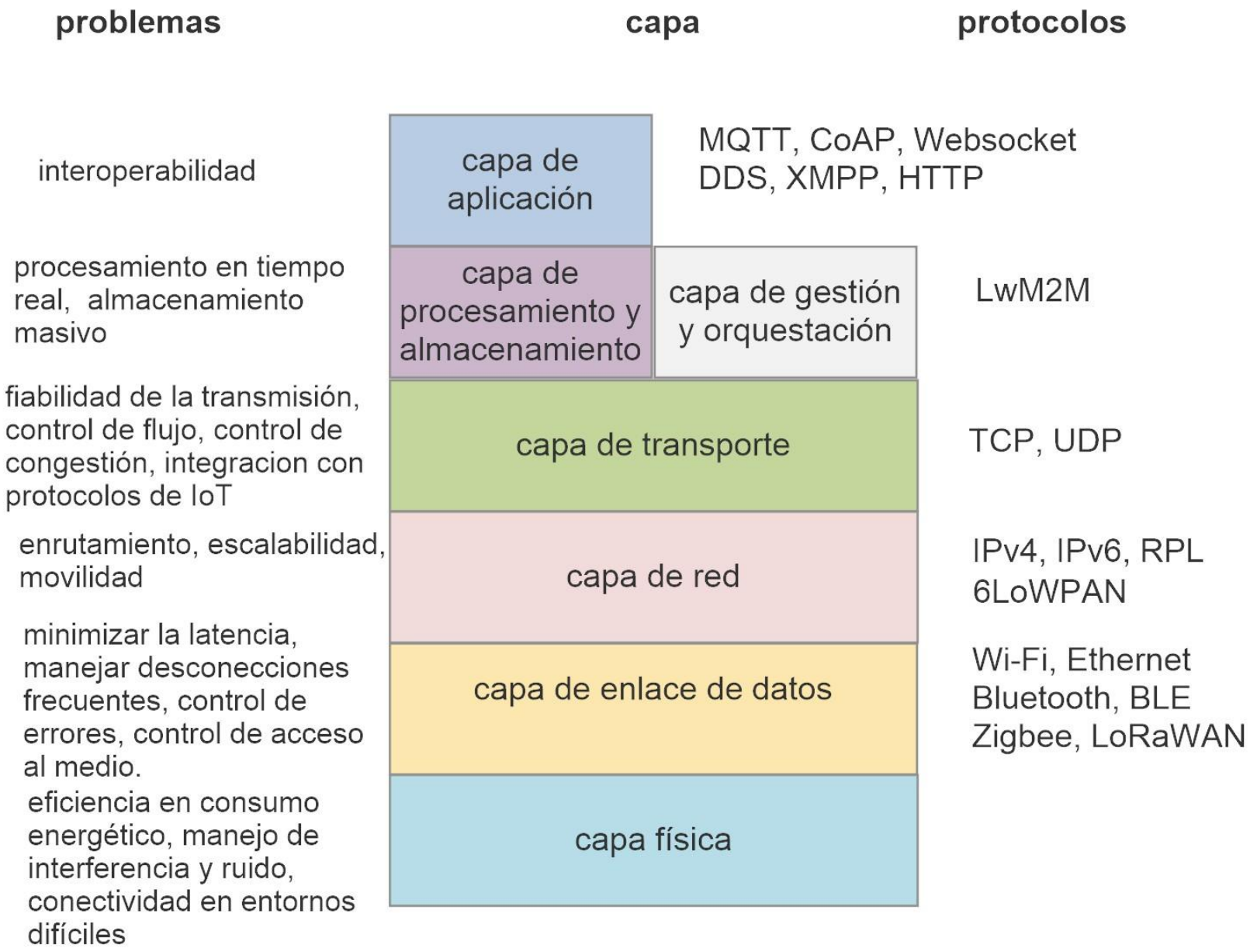
Sistema Operativo para Internet de las Cosas

- **Diferencias de los protocolos de IoT con los protocolos de Internet:**
  - Muchos dispositivos IoT son de **baja potencia** y tienen **recursos limitados**. Por lo tanto los protocolos para IoT están diseñados para ser ligeros y eficientes en el uso de energía y recursos.
    - Los dispositivos conectados a internet, generalmente tienen recursos relativamente abundantes en términos de energía, capacidad de procesamiento y memoria.
  - La IoT se enfoca en la **comunicación entre dispositivos IoT heterogéneos** en entornos específicos (p.ej. Hogares inteligentes, fábricas automatizadas, ciudades inteligentes, etc.)
    - La internet está diseñada para la comunicación entre computadoras y servidores, facilitando el intercambio de información a gran escala y el acceso a servicios en línea.
  - Se necesitan protocolos para **escalar a una gran cantidad de dispositivos IoT**, gestionando **transmisiones de datos frecuentes** y a menudo **en tiempo real**.
    - Los protocolos de internet permiten escalar a muchas máquinas y gestionar grandes volúmenes de tráfico de datos, pero solo para computadoras y servidores, y no para dispositivos IoT.

Sistema Operativos para Internet de las Cosas

- Para la internet de las cosas usaremos un **modelo de capas de referencia** con las mismas capas que las redes de computadoras, pero **se añaden dos capas más**.
- Además, las capas con el mismo nombre resuelven **problemas adicionales** típicos de internet de las cosas.
- **Veremos para cada capa**: su función, los problemas que resuelve, los protocolos que usa y para cada protocolo sus características más salientes.
- Más adelante se estudiarán algunos protocolos en más detalle.

Sistema Operativo para Internet de las Cosas



Sistema Operativo para Internet de las Cosas

- **La capa física**: se encarga de la transmisión de bits a partir de medios físicos.
  - Sensores y actuadores suelen tener capa física.
  - **Problemas:**
    - **El consumo energético**: hay dispositivos IoT que son alimentados por batería y necesitan ser eficientes en el uso de energía.
      - Se implementan técnicas como el modo de sueño, para que los dispositivos entren en estado inactivo cuando no están transmitiendo datos.
    - **Interferencias y ruido**: las comunicaciones inalámbricas pueden ser afectadas por interferencias y ruido.
      - Se usan **técnicas de modulación adaptativa** para optimizar la transmisión según las condiciones del canal.
      - Se **ajusta el tipo de modulación** en función del nivel de interferencia o ruido.
    - **Conectividad**: garantizar que los dispositivos pueden mantenerse conectados en entornos difíciles. P.ej: entornos subterráneos, algunos entornos industriales, condiciones ambientales extremas, etc.
      - Se puede implementar una **malla** de modo que todos los dispositivos se comunican entre sí directamente.

Sistema Operativo para Internet de las Cosas

- **La capa física – cont:**
  - **Protocolos**:
    - **Zigbee**: diseñado para ser eficiente en el consumo de energía; ideal para dispositivos alimentados por baterías.
      - Usa técnicas de modulación y control de acceso al medio para minimizar la interferencia.
      - Proporciona conectividad fiable en entornos de corto alcance.
    - **LoRaWAN**: permite la comunicación de largo alcance con un bajo consumo de energía.
      - Diseñado para dispositivos de baja potencia, que pueden permanecer en modos de sueño la mayor parte del tiempo.
      - Permite conectividad en entornos difíciles como en áreas rurales y en entornos urbanos densos.
      - Admite una gran cantidad de dispositivos conectados.
      - Los dispositivos solo transmiten cuando tienen datos que enviar.
    - **NB-IoT**: permite baja potencia y larga duración de la batería;
      - proporciona una amplia cobertura, incluso en áreas de difícil acceso.
      - Reduce los costos operativos y de implementación.

Sistema Operativo para Internet de las Cosas

- **Capa de enlace de datos:** responsable de la transmisión de datos entre dispositivos dentro de la misma red local.
  - **Problemas considerados:**
    - **Control de errores**: Igual que antes.
    - **Control de acceso al medio**: Igual que antes.
    - **Retrasos y variaciones en el tiempo de transmisión** pueden afectar las aplicaciones en tiempo real.
      - **Idea de solución**: ajustar el tamaño y la estructura de las tramas para minimizar la latencia.
    - **Desconexiones frecuentes:** Los dispositivos IoT al ser móviles o ubicarse en áreas con mala cobertura pueden sufrir desconexiones frecuentes.
    - **Seguridad de la comunicación:** Los datos transmitidos no deben ser interceptados ni alterados.
      - Se puede aplicar cifrado y autenticación en la capa de enlace de datos para asegurar la comunicación.

Sistema Operativo para Internet de las Cosas

- **Capa de enlace de datos – cont:**
  - **Protocolos**:
    - **Ya vistos o conocidos**: Wi-Fi, Ethernet, Bluetooth
    - **También en capa física**: Zigbee, LoRaWan
    - **Bluetooth Low Energy (BLE)**: muy eficiente en consumo de energía, ideal para dispositivos que necesitan durar mucho tiempo con baterías pequeñas.
      - Menor costo de implementación en comparación con otras tecnologías de corto alcance.
    - **6LoWPAN**: permite encapsular y enviar paquetes IPv6 sobre redes de baja potencia.
      - Por ejemplo, se puede usar sobre redes de área personal inalámbricas de baja potencia.
      - Se puede usar en sensores y actuadores.
      - Implementa técnicas de compresión de encabezado y fragmentación para permitir que los paquetes IPv6 se transmitan eficientemente en redes WPAN de baja potencia.

Sistema Operativo para Internet de las Cosas

Característica	Bluetooth	Wi-Fi	Zigbee
Tecnología de Conexión	Inalámbrica de corto alcance	Inalámbrica de alta velocidad	Inalámbrica de baja potencia
Consumo de Energía	Muy bajo a moderado (según versión)	Moderado a alto	Muy bajo
Rango de Cobertura	Corto (1-100 metros, según versión)	Medio (30-100 metros)	Corto (10-100 metros)
Velocidad de Datos	Baja a moderada (hasta 3 Mbps en BLE)	Alta (hasta varios Gbps)	Baja (hasta 250 kbps)
Topología de Red	Punto a punto, estrella	Punto de acceso	Malla, estrella, punto a punto
Costo de Implementación	Bajo	Moderado a alto	Bajo
Interferencia	Moderada (especialmente en entornos saturados)	Alta (debido a otras redes Wi-Fi y dispositivos)	Baja
Casos de Uso Comunes	Dispositivos personales, auriculares, domótica	Redes domésticas y empresariales, streaming	Domótica, redes de sensores, IoT
Estándares	IEEE 802.15.1	IEEE 802.11 (a/b/g/n/ac/ax)	IEEE 802.15.4

Sistema Operativo para Internet de las Cosas

- **Capa de red**: Gestiona el direccionamiento y el enrutamiento de los dados entre diferentes redes.
  - **Problemas**:
    - **Enrutamiento**: en redes con dispositivos móviles y topologías cambiantes.
    - **Escalabilidad**: gestión eficiente de un gran número de dispositivos.
    - **Movilidad en redes inalámbricas**: soporte para dispositivos que se mueven fuera y dentro de la red.

Sistema Operativo para Internet de las Cosas

- **Capa de red – cont:**
  - **Protocolos**:
    - **De internet**: IPv4, IPv6
    - **6LoWPAN**: permite que los dispositivos de baja potencia usen IPv6.
      - Optimizado para *dispositivos de baja potencia*.
      - Resuelve problemas de enrutamiento y direccionamiento.
      - Facilita la integraciones con redes IP tradicionales.
      - Utiliza IEEE 802.15.4 como protocolo de enlace de datos, que es adecuado para *comunicaciones de corto alcance y baja potencia.*
    - **RPL**: Optimizado para *redes de dispositivos con baja potencia y alta tasa de pérdida de paquete*s. Soporta redes con gran número de nodos.
      - RPL adapta las rutas de transmisión en función de las condiciones de la red, optimizando el uso de los recursos y reduciendo el consumo de energía.

Sistema Operativo para Internet de las Cosas

- **Capa de transporte:** Asegura la transmisión de datos confiable y ordenada entre dispositivos.
  - **Problemas**:
    - **Fiabilidad de la transmisión**: asegurar que los datos lleguen de manera confiable, en especial en redes con alta tasa de pérdida de paquetes.
    - Control de flujo y control de congestión
    - **Compatibilidad de protocolo**: integración con protocolos específicos de IoT.

Sistema Operativo para Internet de las Cosas

- **Capa de transporte – cont:**
  - **Protocolos**
    - **De internet:** TCP, UDP
    - **MQTT**: optimizado para redes con *ancho de banda limitada*; diseñado para *minimizar el consumo de energía durante la transmisión de datos*, proporciona varios niveles de calidad de servicio para garantizar la entrega de mensajes.
      - Usa *formato de mensaje compacto* para minimizar el tamaño de los datos enviados, esto reduce consumo de energía.
      - MQTT usa TCP como base para la transmisión de datos.
      - Se centra en la *comunicación de mensajes en tiempo real*, comunicación máquina a máquina.
    - **CoAP**: optimizado para *dispositivos con recursos limitados*, como baja capacidad de procesamiento y memoria. Proporciona confirmaciones de entrega y retransmisión de mensajes perdidos.
      - Optimizado para minimizar el uso de ancho de banda.
      - También usa formato binario de mensaje compacto.
      - Generalmente se implementa sobre UDP.
      - Por ejemplo: se usa en redes de área personal.
      - Por ejemplo, se usa en redes de monitorización y control.

Sistema Operativo para Internet de las Cosas

- **Capa de aplicación**: define los protocolos de comunicación usados por las aplicaciones de IoT.
  - Esta capa facilita la interacción entre el usuario y el sistema IoT.
  - **Problemas**:
    - **Interoperabilidad**: asegurar que los dispositivos y aplicaciones de diferentes fabricantes puedan comunicarse entre si.
    - **Seguridad y privacidad**: garantizar que los datos sean transmitidos y almacenados de manera segura.
      - Se puede usar SSL y TLS para asegurar las comunicaciones.
    - **Gestión de dispositivos**: Administración eficiente de un gran número de dispositivos.
      - Se pueden usar plataformas de gestión que facilitan la gestión y monitoreo de dispositivos.
- **Eficiencia energética** para dispositivos con recursos limitados y consumo de energía bajo.
- **Fiabilidad y calidad de servicio**: entrega de mensajes confiables para aplicaciones críticas.
- **Escalabilidad y ancho de banda**: poder manejar la demanda cuando el número de dispositivos IoT aumenta.
- **Simplicidad**: simples de implementar en dispositivos con capacidades limitadas.
- **Complejidad**: suficientemente complejos como para manejar las necesidades de las aplicaciones.

Sistema Operativo para Internet de las Cosas

- **Capa de aplicación – cont:**
  - **Protocolos**:
    - **De la web**: HTTP, HTTPS
    - **MQTT**: facilita la comunicación de dispositivos y servidores; facilita la comunicación entre diferentes fabricantes; soporta grandes cantidades de nodos y transmisiones de datos frecuentes.
      - Para eso usa un **modelo de publicación/suscripción** para minimizar el uso de ancho de banda y energía. Se mandan datos solo cuando es necesario en lugar de mantener conexiones constantes.
      - Usa **topicos** para direccionar los mensajes: un tópico representa un canal de comunicación a través del cual se publican y suscriben mensajes.
      - Los clientes publican mensajes en tópicos y los suscriptores reciben los mensajes de esos tópicos.
      - El **bróker** es un servidor que actúa como intermediario en la comunicación entre dispositivos IoT. Es intermediario entre dispositivos que quieren enviar datos (publicadores) y dispositivos que quieren recibir datos (suscriptores)
      - Mantiene comunicación persistente con el bróker, reduciendo la sobrecarga de establecer nuevas conexiones.

Sistema Operativo para Internet de las Cosas

- **Capa de aplicación – cont:**
  - **Protocolos – cont:**
    - **CoAP**: proporciona un enfoque ligero para la comunicación entre dispositivos y servidores. Es ideal para dispositivos con baterías. Ligero y fácil de implementar.
      - Sigue un modelo de arquitectura de cliente-servidor.
      - Usa métodos similares a HTTP para interactuar con los recursos.
      - Envía códigos de estado para las respuestas.
      - Incluye un mecanismo de descubrimiento de recursos.
      - Usa URIs, pero adaptados para dispotivos IoT. Los URIs representan recursos alojados en dispositivos IoT.
    - **Websocket**: WebSocket se basa en TCP y permite streams de mensajes a ser enviados en ambos sentidos entre cliente y servidor, mientras se mantiene la conexión TCP abierta.
    - **DDS** (Data distribution service): es un middleware centrado en datos para la comunicación de dispositivo-a-dispositivo o máquina-a-máquina.
    - **XMPP** (Extensible Messaging and Presence Protocol) es un protocolo para comunicación de tiempo real y streaming de datos XML entre entidades de red. XMPP soporta caminos de comunicación cliente-a-servidor y servidor-a-servidor.

Sistema Operativo para Internet de las Cosas

- **Capa de procesamiento y almacenamiento**: Es responsable de procesar, almacenar y analizar los datos recopilados por los dispositivos IoT.
  - **Problemas:**
    - **Procesamiento en tiempo real:** Capacidad para procesar y analizar datos rápidamente.
      - **Edge computing**: procesamiento de datos cercanos a la fuente para reducir la latencia. Reduce la carga de la red al procesar datos localmente antes de enviarlos a la nube.
    - **Almacenamiento masivo:** manejo eficiente del almacenamiento de grandes volúmenes de datos.
      - **Cómputo en la nube:** uso de servicios de la nube para almacenar y analizar grandes volúmenes de datos. Permite el acceso a datos desde cualquier lugar.
  - **Componentes**: Servidores locales, edge computing, las nubes.
  - **Tecnologías**: Bases de datos, sistemas de procesamiento de datos en tiempo real, análisis de grandes cantidades de datos.

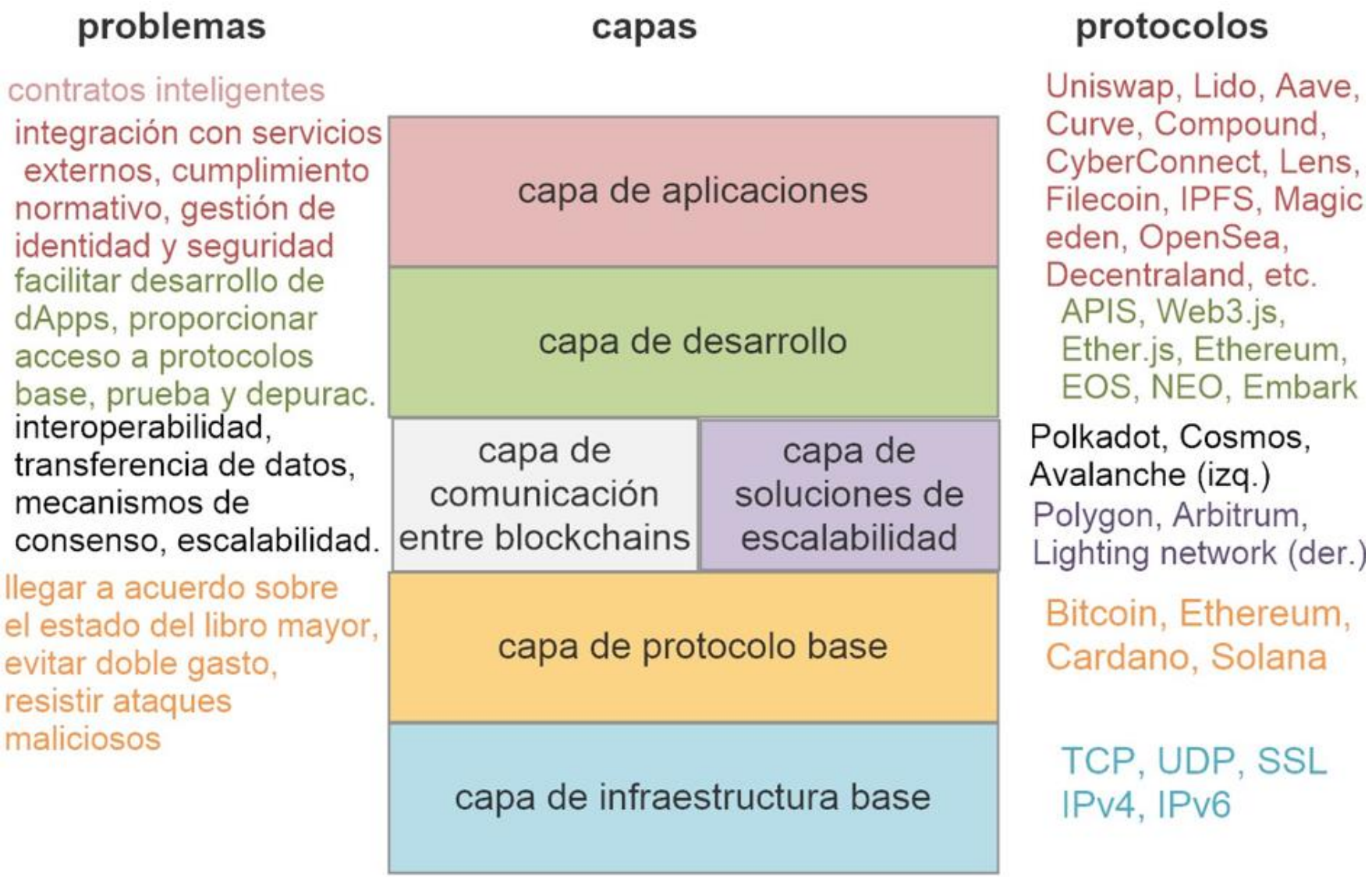
Sistema Operativo para Internet de las Cosas

- **Capa de gestión y orquestación**: se encarga de la gestión y orquestación de recursos en la red IoT.
  - Proporciona herramientas para la *configuración*, *monitoreo*, *actualización* y *administración* de dispositivos y aplicaciones IoT.
  - **Funciones**: gestión de dispositivos, seguridad, actualizaciones de firmware, monitoreo de rendimiento.
  - **Plataformas IoT**: proveen herramientas para la gestión centralizada de dispositivos
  - **Problemas**:
    - **Configuración y monitoreo**: necesidad de configurar y monitorear dispositivos de manera eficiente.
    - **Actualización de firmware**: capacidad de actualizar el software de los dispositivos de manera remota.
  - **Protocolos de gestión de dispositivos**:
    - LwM2M: proporciona herramientas para la configuración, monitoreo y administración de dispositivos IoT. Facilita la administración remota del firmware de los dispositivos.

Sistema Operativos para redes blockchain

- Para las redes blockchain usaremos un **modelo de capas de referencia**. La organización en capas es bastante diferente que en las redes de computadoras.
- Además, la capa de aplicación resuelve **problemas adicionales** típicos de las redes blockchain.
- **Veremos para cada capa**: su función, los protocolos que usa.
- Más adelante se estudiarán algunos protocolos en más detalle.

Sistemas Operativos para Redes Blockchain



Sistemas Operativos para Redes Blockchain

- **Capa de Infraestructura Base**
  - Proporciona la infraestructura subyacente para la creación y operación de blockchains.
  - Incluye componentes físicos y de red.
  - Por ejemplo: internet (TCP/IP, HTTP, SSL, etc.)
  - Por ejemplo: enrutadores.

Sistemas Operativos para Redes Blockchain

- **Capa de Protocolo Base**
  - Es la **blockchain** en si misma.
    - Tiene su propia **cadena de bloques** con un diseño específico, incluyendo su propio token nativo.
    - Es responsable de la *seguridad* y el *funcionamiento operativo* de la red blockchain.
    - Establece las **reglas fundamentales de consenso** y la estructura de datos principal.
    - Facilita la comunicación entre nodos y el envío de transacciones.
  - Aquí se llevan a cabo las **transacciones**.
    - Se procesan las transacciones y se validan los datos.

Sistemas Operativos para Redes Blockchain

- **Capa de Protocolo Base – Cont:**
  - Un **mecanismo de consenso** establece las reglas y mecanismos mediante los cuales los nodos llegan a un acuerdo sobre el estado del libro mayor. **Beneficios**:
    - Usar mecanismo de consenso previene problemas como el **doble gasto** al garantizar que solo una versión del libro mayor sea aceptada por todos los nodos.
    - Usar mecanismo de consenso aumenta la **resistencia a ataques maliciosos** al requerir que un número significativo de nodos coincida en el estado del sistema.

Sistemas Operativos para Redes Blockchain

- **Capa de Protocolo Base – Cont:**
  - **Ejemplos de mecanismos de consenso (a estudiar más adelante).**
    - prueba de participación (PoS),
    - prueba de trabajo (PoW),
    - prueba de historia (PoH)
  - **Ejemplos de protocolos**:
    - Bitcoin (usa PoW),
    - Ethereum (usa PoS),
    - Cardano (usa versión de PoS),
    - Solana (combina PoS con PoH).

Sistemas Operativos para Redes Blockchain

- **Capa de Protocolo Base – Cont:**
  - **Limitaciones**: Las blockchains de esta capa suelen enfrentar limitaciones en su capacidad para procesar un gran número de transacciones por segundo.
    - Esto puede resultar en tiempos de espera prolongados y tarifas elevadas, especialmente en períodos de alta demanda.
    - Para resolver esto se definen **soluciones de escalabilidad**.

Sistemas Operativos para Redes Blockchain

- **Capa de Comunicación entre Redes Blockchain**:
  - Se usan protocolos que facilitan la interoperabilidad y la comunicación entre diferentes redes blockchain.
  - Facilita la creación de redes interconectadas.
  - Aborda problemas como la escalabilidad e interoperabilidad.
  - **Ejemplos**:
    - Polkadot: usa la arquitectura de relay chain que conecta diversas cadenas, permitiendo la interoperabilidad entre ellas.
      - Su protocolo cross-chain messaging passing (XCMP) permite la comunicación y el intercambio de datos entre diferentes blockchains.
    - **Cosmos** (**la internet de las blockchains**): facilita el intercambio de activos y datos sin necesidad de soluciones centralizadas.
      - Permite que distintas blockchain se comuniquen entre sí usando el **protocolo Inter-Bllockchain Communication (IBC)**

Sistemas Operativos para Redes Blockchain

- **Capa de Comunicación entre Redes Blockchain - cont**:
  - **Ejemplos - cont**:
    - **Avalanche**: ofrece cadenas como P-Chain, X-Chain, y C-chain, cada una optimizada para tareas específicas.
      - Usa el protocolo **Avalanche Warp Messaging** para facilitar la comunicación entre estas cadenas.
    - **LayerZero**: permite la interoperabilidad entre blockchains, facilitando la transferencia de activos y datos sin comprometer la seguridad.
    - **Wormhole**: protocolo de puente que permite la transferencia de activos y datos entre diferentes blockchains.

Sistemas Operativos para Redes Blockchain

- **Capa de Soluciones de Escalabilidad**
  - Mejora el rendimiento y la capacidad de procesamiento de transacciones al construirse sobre una blockchain existente.
  - Permiten **realizar un mayor número de transacciones fuera de la cadena principal**.
    - Esto reduce la carga sobre la blockchain y disminuye los costos asociados.
    - Se procura no comprometer la seguridad proporcionada por la blockchain.

Sistemas Operativos para Redes Blockchain

- **Capa de Soluciones de Escalabilidad – cont:**
  - **Protocolos**:
    - **Rollups**: agrupan varias transacciones en un paquete; este paquete se valida en una red secundaria y luego registran solo los resultados finales (como un resumen criptográfico del paquete) en la blockchain del protocolo base.
    - **Cadenas laterales**: blockchains independientes que están conectadas a una blockchain principal.
      - Una cadena lateral puede procesar transacciones y ejecutar aplicaciones de manera independiente.
      - Algunas cadenas laterales pueden ejecutar contratos inteligentes.
      - Las cadenas laterales cuando ejecutan una transacción, la registran en su propio libro mayor.
      - Las cadenas laterales pueden usar sus propios protocolos y mecanismos de consenso.

Sistemas Operativos para Redes Blockchain

- **Capa de Soluciones de Escalabilidad – cont:**
  - **Protocolos – cont:**
    - **Canales de estado**: permiten transacciones rápidas y privadas entre dos partes sin necesidad de registrar cada transacción en la blockchain principal.
      - Un **canal de estado** es un *entorno temporal* donde dos o mas partes pueden ejecutar varias transacciones directamente entre ellas sin involucrar a la blockchain principal.
      - Las transacciones se llevan a cabo **fuera de la blockchain principal**, lo que reduce los tiempos y costos asociados con cada operación.
      - El canal de estado lleva el registro del **estado actual** de las transacciones entre las partes involucradas.
      - Al finalizar las interacciones, el **estado final del canal se registra en la blockchain** principal.
      - Solo se **pagan tarifas** por abrir y cerrar el canal.
      - Las transacciones dentro del canal no son visibles públicamente; solo el estado inicial y final son visibles.
      - Cada transacción dentro del canal debe ser **firmada por todos los participantes**, lo que garantiza que no se pueda alterar el estado final sin el consentimiento mutuo.
      - En un canal de estado, los participantes crean un **contrato inteligente** que define las reglas y condiciones bajo las cuales se llevarán a cabo las transacciones. Este contrato actúa como un "juez" que asegura que las transacciones sean válidas y que todos los participantes cumplan con lo acordado

Sistemas Operativos para Redes Blockchain

- **Capa de Soluciones de Escalabilidad – cont:**
  - **Ejemplos**:
    - **Lighting Network** (para Bitcoin): usa canales de estado.
    - **Polygon**: usa cadenas laterales y rollups.
    - **Arbitrum** (para Ethereum): usa rollups.

Sistemas Operativos para Redes Blockchain

- **Capa de Aplicaciones:**
  - Permite la ejecución y el acceso a las aplicaciones que interactúan con la blockchain.
  - Aquí se encuentran las aplicaciones descentralizadas (dApps) que operan sobre la blockchain, así como aplicaciones no descentralizadas.
  - **Características de las dApps:**
    - Las dApp no dependen de servidores y operan sobre la blockchain.
    - Las dApp usan **contratos inteligentes** y **APIs** (ver capa de desarrollo) para la interacción con la blockchain.
    - Las dApp pueden seguir funcionando incluso si una parte de la red es atacada o censurada.
    - Las dApp operan sin la necesidad de intermediarios, permitiendo a los usuarios interactuar directamente con la aplicación.
    - Muchas dApp tienen código abierto.

Sistemas Operativos para Redes Blockchain

- **Capa de Aplicaciones – cont:**
  - Los **contratos inteligentes** permiten la ejecución automática de acuerdos cuando se cumplen ciertas ***condiciones predefinidas***.
    - Esto elimina la necesidad de intermediarios, lo que reduce costos y tiempos de transacción.
    - Cada contrato inteligente desplegado ***tiene una dirección única*** usada para identificarlo.
  - Los contratos inteligentes están registrados en la blockchain, por lo que son **inmutables** y **transparentes**.
    - Las partes involucradas pueden verificar las condiciones y resultados del contrato, lo que fomenta la confianza entre ellas.
  - Los contratos inteligentes pueden gestionar transacciones complejas que involucran múltiples partes y condiciones.
  - Los contratos inteligentes pueden interactuar con otros contratos y dApps en la blockchain.
  - Los contratos inteligentes minimizan el riesgo de errores humanos.

Sistemas Operativos para Redes Blockchain

- **Capa de Aplicaciones - cont**
  - **Ejemplos de aplicaciones descentralizadas:**
    - **Finanzas descentralizadas (DeFi):** Aquí los contratos inteligentes son fundamentales para gestionar préstamos, intercambios y otros servicios financieros sin intermediarios.
      - **Ejemplos**: Uniswap, Lido, MakerDAO, Aave, Curve finance, Compound.
    - **Gestión de cadenas de suministro**: Pueden rastrear productos a lo largo de la cadena de suministro, registrando cada paso en la blockchain para garantizar autenticidad y trazabilidad. Es una dApp siempre que use contratos inteligentes.

Sistemas Operativos para Redes Blockchain

- **Capa de Aplicaciones - cont**
  - **Ejemplos de aplicaciones descentralizadas - cont:**
    - **Redes sociales**: ofrecen mayor privacidad y control sobre los datos personales, a menudo recompensando a los usuarios con criptomonedas por su participación.
      - Ejemplos: CyberConnect, Lens Protocol.
    - **Votación**: redes que usan la blockchain para mejorar la transparencia y la seguridad en los procesos de votación.
    - **Almacenamiento descentralizado**: permiten el almacenamiento y la distribución de datos de manera descentralizada aumentando la seguridad y resistencia frente a fallos.
      - Ejemplos: Filecoin, IPFS (InterPlanetary File System)

Sistemas Operativos para Redes Blockchain

- **Capa de Aplicaciones - cont**
  - **Ejemplos de aplicaciones descentralizadas - cont:**
    - **Mercados de NFTs**: plataformas que permiten a los usuarios comprar, vender y coleccionar tokens no fungibles (NFT).
      - Los **NFT** son **tokens criptográficos** que **representan activos únicos**, ya sea digitales o versiones tokenizadas de activos del mundo real.
        - » El uso de la tecnología blockchain garantiza la autenticidad y unicidad.
      - **Ejemplos**: OpenSea y Magic Eden. Otro ejemplo es tokenización de bienes raíces.
    - **Juegos**: permiten a los usuarios jugar y ganar recompensas en forma de criptomonedas o NFTs.
      - **Ejemplos**: Decentraland, Splinter Lands, CryptoKities.

Sistemas Operativos para Redes Blockchain

- **Capa de aplicaciones – cont:**
  - **Ejemplos de aplicaciones no descentralizadas**:
    - **Interfaces de usuario**: permiten interactuar con la blockchain de manera intuitiva.
      - Por ejemplo: aplicaciones web y móviles que permiten la gestión de criptomonedas, visualización de datos, etc.
    - **Wallets** (**Billeteras**): Aplicaciones que permiten a los usuarios almacenar, enviar y recibir criptomonedas.
      - **Ejemplos**: MetaMask, Trust Wallet, billeteras de hardware como Ledger.

Sistemas Operativos para Redes Blockchain

- **Capa de Desarrollo**
  - Proporciona las herramienta necesarias para desarrollar y desplegar aplicaciones sobre blockchain.
  - **Tecnologías específicas usadas:**
    - **APIs**: permiten a los desarrolladores construir aplicaciones que se comunican con la blockchain sin necesidad de entender completamente su funcionamiento interno. Ejemplo: APIs RESTFUL.
    - **Bibliotecas**: Facilitan el desarrollo de aplicaciones descentralizadas sobre blockchains especificas. Ejemplos:
      - **Web3.js** es una biblioteca JavaScript que permite a los desarrolladores interactuar con Ethereum desde aplicaciones web.
      - **Ether.js:** es una biblioteca JavaScript que facilita la interacción con Ethereum, enfocándose en la simplicidad y seguridad.

Sistemas Operativos para Redes Blockchain

- **Capa de desarrollo – cont:**
  - **Tecnologías específicas usadas - cont:**
    - **Plataformas de desarrollo**: Entornos que permiten a los desarrolladores crear y desplegar dApps.
      - **Ethereum** es la plataforma más popular para desarrollar dApps, permitiendo crear contratos inteligentes.
      - **Hyperledger Fabric**: es una plataforma empresarial que permite el desarrollo de aplicaciones blockchain privadas y permissioned.
      - **EOS** y **NEO**: son plataformas que facilitan el desarrollo de dApps con un enfoque en escalabilidad y rendimiento.
    - **Frameworks para dApps**: estructura que proporciona un conjunto de herramientas y bibliotecas para facilitar el desarrollo de dApps.
      - **Ejemplo**: Embark facilita el desarrollo y la implementación de dApps usando tecnologías descentralizadas.

Sistemas Operativos para Redes Blockchain

- **Lenguajes de programación** usados para desarrollar aplicaciones en la blockchain:
  - **Solidity**: usado para escribir contratos inteligentes en Ethereum.
  - **JavaScript**: usado junto con Web3.js para inrteractuar con la blockchain.
  - **Python**: Usado en plataformas como Modum para desarrollar contratos inteligentes y scripts de interacción.
  - **C++:** usado para desarrollar nodos y aplicaciones en la Blockchain como Bitcoin y Ethereum.
  - **Go**: usado para el desarrollo de aplicaciones para varias plataformas blockchain, incluyendo Hyperledger.

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

Misión

- **Misión de la materia**:
  - Adquirir habilidades con el fin de poder:
    - comprender, evaluar, usar: redes, SOR, middlewares y aplicaciones de red;
    - desarrollar aplicaciones de red.
  - Comprender se refiere a conceptos, principios, problemas y protocolos.
    - Con esta finalidad se usan ejercicios de razonamiento y cuentas que usan matemática elemental.
    - Además se consideran preguntas teóricas.
    - El alumno necesitará poder leer y entender protocolos y aplicaciones de red no enseñados en clase (habrá tareas de este tipo).
    - Puede haber algunos labs que ayuden a comprender en profundidad algunos protocolos.
  - Evaluar se refiere a: comparar, ver semejanzas y diferencias entre distintas alternativas de solución a un problema.
    - Incluye poder decidir cuál solución adoptar para un problema.