UNIVERSIDAD DE GRANADA

Fundamentos de Redes

Temario de la asignatura

Ismael Sallami Moreno

Recursos Ingeniería Informática y Ade

Licencia

Fundamentos de Redes

Ismael Sallami Moreno

https://elblogdeismael.github.io

Índice general

Capítulo 1

Introducción a los fundamentos de redes

Tema 1 del programa, bloque 1. Qué es una red, por qué se diseña en capas, cómo viaja la información y qué es Internet.

1.1 Sistemas de comunicación y redes

Un sistema de comunicación traslada información de un punto a otro. Sus elementos:

Elemento	Qué hace
Emisor	genera el mensaje
Transmisor	lo convierte en señal apta para el medio
Medio	soporte físico por el que viaja la señal
Receptor	recupera el mensaje a partir de la señal
Destino	consume el mensaje

Una **red** es un conjunto de sistemas de comunicación que permite que muchos equipos se comuniquen entre sí sin un enlace dedicado para cada par. Con n equipos harían falta $n(n-1)/2$ enlaces punto a punto; la red los sustituye por una infraestructura compartida y un mecanismo para dirigir cada mensaje a su destino.

1.1.1 Clasificación

Criterio	Tipos
Alcance	red de área personal, local (LAN), metropolitana (MAN), amplia (WAN)
Topología	bus, estrella, anillo, árbol, malla
Conmutación	de circuitos, de paquetes
Titularidad	privada, pública

1.1.2 Conmutación de circuitos y de paquetes

Es la distinción que más consecuencias tiene, y conviene tenerla clara desde el principio.

	Circuitos	Paquetes
Recursos	reservados durante toda la conexión	compartidos, se piden al usarse
Establecimiento previo	sí	no, en el modo sin conexión
Retardo	constante una vez establecido	variable, por las colas
Si el emisor calla	los recursos se desperdician	quedan para otros
Ante un fallo	la conexión cae	los paquetes se reencaminan
Garantía de capacidad	sí	no, salvo mecanismos añadidos

La telefonía clásica es de circuitos e Internet es de paquetes. La razón del cambio es la naturaleza del tráfico de datos: es **a ráfagas**. Un usuario que navega transmite durante un instante y calla durante segundos, así que reservarle capacidad todo el tiempo desaprovecha casi toda.

La contrapartida es que los paquetes compiten por los mismos enlaces, y de ahí salen las colas, el retardo variable y la congestión que el tema 3 tiene que controlar.

1.2 Diseño funcional en capas

Comunicar dos equipos exige resolver muchos problemas a la vez: modular la señal, detectar errores, encontrar el camino, reordenar lo que llega desordenado, interpretar el contenido. Resolverlos todos en un solo bloque de software sería inmanejable.

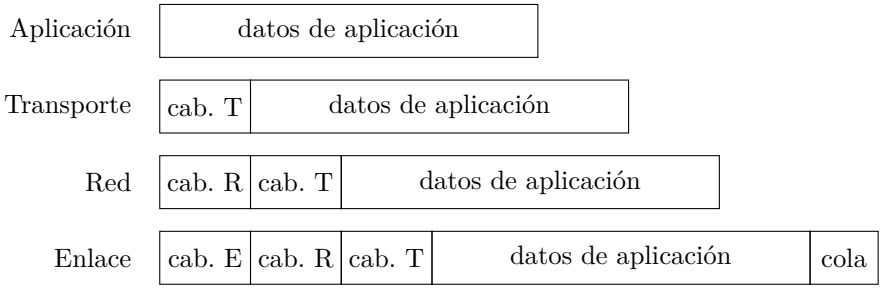
La solución es **estratificar**: cada capa resuelve un problema, usa los servicios de la de abajo y ofrece un servicio a la de arriba.

Principio	Qué significa
Servicio	lo que una capa ofrece a la superior
Protocolo	las reglas que siguen entre sí dos capas del mismo nivel
Interfaz	cómo se pide el servicio a la capa inferior

Servicio y protocolo son cosas distintas, y confundirlos es el error habitual: el servicio dice *qué* se hace y el protocolo *cómo*. Se puede cambiar el protocolo de una capa sin tocar las demás mientras el servicio se mantenga, y esa independencia es toda la ganancia del diseño en capas.

1.2.1 Encapsulamiento

Cada capa añade su propia cabecera a lo que recibe de arriba y trata todo el conjunto como datos.



La unidad de datos de cada capa recibe un nombre propio: **mensaje** en aplicación, **segmento** en transporte, **datagrama** en red y **trama** en enlace. La asignatura los usa con precisión, y conviene hacerlo, porque decir «paquete» para todo esconde en qué capa está el problema.

1.2.2 Los dos modelos

OSI, 7 capas	TCP/IP, 5 capas	Qué resuelve
Aplicación, presentación, sesión	Aplicación	el servicio que ve el usuario
Transporte	Transporte	comunicación extremo a extremo entre procesos
Red	Red	encaminamiento entre redes distintas
Enlace	Enlace	transmisión fiable sobre un enlace
Física	Física	señales sobre el medio

OSI es el modelo de referencia y no llegó a implantarse; TCP/IP es lo que se usa. Sus capas de sesión y presentación no desaparecieron: sus funciones quedaron dentro de la de aplicación, que es donde hoy se cifra y se codifica.

Y una asimetría que conviene registrar: **las capas de red hacia abajo están en todos los nodos, y las de transporte hacia arriba solo en los extremos**. Un encaminador no tiene TCP para el tráfico que reenvía. Por eso el control de congestión del tema 3 lo hacen los extremos, que son los únicos que ven el problema completo.

1.3 Transmisión de información

Lo que viaja por el medio son señales, y sus límites condicionan todo lo demás.

1.3.1 Capacidad del canal

La **velocidad de transmisión** se mide en bits por segundo, y su límite lo fijan el ancho de banda del canal y el ruido. Con ruido, el teorema de Shannon da la capacidad máxima:

$$C = B \log_2\left(1 + \frac{S}{N}\right)$$

con B el ancho de banda en hercios y S/N la relación señal-ruido. La consecuencia práctica es que **la capacidad no se puede aumentar sin límite** subiendo la potencia: crece de forma logarítmica con ella, mientras que crece de forma lineal con el ancho de banda.

1.3.2 Retardos

El tiempo que tarda un paquete en cruzar un enlace tiene cuatro componentes, y mezclarlos es fuente de errores:

Componente	Fórmula	De qué depende
Procesamiento	—	la carga del nodo
Cola	variable	cuánto tráfico hay por delante
Transmisión	L/R	tamaño del paquete y velocidad del enlace

Componente	Fórmula	De qué depende
Propagación	d/v	distancia y velocidad de la señal en el medio

$$T_{\text{total}} = T_{\text{proc}} + T_{\text{cola}} + \frac{L}{R} + \frac{d}{v}$$

Transmisión y propagación son independientes. La primera depende del tamaño del paquete y de la velocidad del enlace; la segunda solo de la distancia. En un enlace corto y lento domina la transmisión, y en un enlace transoceánico rápido domina la propagación, y ahí el ancho de banda no compra latencia.

Solo el retardo de cola es variable, y es el que produce la **fluctuación** que molesta a las aplicaciones en tiempo real del tema 5.

1.3.3 Multiplexación

Compartir un medio entre varias comunicaciones:

Técnica	Cómo reparte
Por división de frecuencia	cada canal, una banda distinta
Por división de tiempo	cada canal, una ranura temporal
Por división de longitud de onda	como la de frecuencia, en fibra óptica
Por división de código	todos a la vez, con códigos ortogonales
Estadística	ranuras asignadas según demanda, no fijas

La **estadística** es la que usa la conmutación de paquetes, y es lo que la hace eficiente: no reserva ranura a quien no tiene nada que enviar.

1.4 Internet

Internet no es una red: es una **red de redes**. Miles de redes autónomas, gestionadas por organizaciones distintas, interconectadas y funcionando como si fueran una.

Su arquitectura descansa en tres decisiones:

1. **Un protocolo común en la capa de red, IP.** Cada red puede usar la tecnología de enlace que quiera —Ethernet, wifi, fibra— porque todas hablan IP por encima. Es el modelo del reloj de arena: mucha variedad arriba, mucha variedad abajo, y un solo protocolo en la cintura.
2. **La inteligencia en los extremos.** La red se limita a entregar paquetes lo mejor que puede, y la fiabilidad, el orden y el control de flujo los ponen los extremos. Eso hace la red simple, barata y capaz de crecer.
3. **El servicio del mejor esfuerzo.** IP no garantiza entrega, ni orden, ni retardo máximo. Todo lo que un usuario percibe como fiabilidad lo construye TCP por encima.

1.4.1 Organización

Nivel	Quién
Proveedores de nivel 1	operadores globales, interconectados entre sí
Proveedores regionales	conectan a los de nivel 1
Proveedores de acceso	dan servicio al usuario final
Puntos neutros	donde varios proveedores intercambian tráfico directamente

Un **sistema autónomo** es una red bajo una administración única con una política de encaminamiento propia. La distinción entre encaminar dentro de un sistema autónomo y entre sistemas autónomos es la que ordena el tema 2.

Y los protocolos los publica el **IETF** como documentos RFC. Un RFC no es una norma impuesta: es una especificación que se adopta si funciona y si se implanta, y esa es la forma en que Internet ha ido cambiando sin que nadie la dirija.

El planteamiento general y las magnitudes de transmisión siguen a García Teodoro et al., 2007; la presentación por capas y de Internet, a Kurose y Ross, 2017 y Tanenbaum, 2011.

Capítulo 2

Capa de red

Tema 2 del programa, bloque 2. Llevar un datagrama de un extremo a otro atravesando redes distintas: direcciones, formato, fragmentación, encaminamiento y los protocolos auxiliares.

2.1 Funcionalidades

La capa de red tiene dos funciones, y conviene separarlas porque ocurren en tiempos distintos:

Función	Qué hace	Cuándo
Reenvío	mover un datagrama de la entrada a la salida correcta de un nodo	por cada paquete, en nanosegundos
Encaminamiento	calcular las rutas que llenan la tabla de reenvío	cada cierto tiempo, en segundos o minutos

El reenvío es local y rápido: mirar la dirección, consultar la tabla, sacar el paquete por el puerto que diga. El encaminamiento es global y lento: los nodos intercambian información y calculan caminos.

El servicio que ofrece IP es del **mejor esfuerzo**: sin conexión, sin garantía de entrega, sin garantía de orden y sin garantía de retardo. Los datagramas pueden perderse, duplicarse, llegar desordenados o llegar tarde. Todo eso lo arregla TCP en el tema 3, o no se arregla.

2.2 El protocolo IP

2.2.1 Direcciones IP

Una dirección IPv4 son 32 bits, escritos como cuatro octetos decimales separados por puntos: `192.168.10.5`.

La dirección **no identifica al equipo, sino a su interfaz**. Un encaminador con tres interfaces tiene tres direcciones IP, una por cada red a la que se asoma.

La dirección se parte en dos:

```
192.168.10.5 / 24
|__________|  |__|
  direccion   longitud del prefijo
```

- El **prefijo de red**, los primeros bits, identifica la red.
- El **identificador de host**, el resto, identifica la interfaz dentro de ella.

La **máscara** marca con unos los bits de red: `/24` equivale a `255.255.255.0`. Con máscara de n bits hay 2^{32-n} direcciones en la red, de las cuales dos no se pueden asignar:

Dirección	Qué es
Todo ceros en el host	identifica a la red
Todo unos en el host	difusión dirigida a esa red

Así que una `/24` da $256 - 2 = 254$ direcciones asignables.

Direcciones con significado propio:

Rango	Uso
`10.0.0.0/8`, `172.16.0.0/12`, `192.168.0.0/16`	privadas, no encaminables en Internet
`127.0.0.0/8`	bucle local
`169.254.0.0/16`	autoconfiguración cuando no hay servidor
`224.0.0.0/4`	multidifusión
`255.255.255.255`	difusión limitada, no se reenvía

2.2.2 De las clases al CIDR

El diseño original repartía el espacio en clases fijas: A con `/8`, B con `/16` y C con `/24`. El problema es que las tallas no encajaban con nadie: una organización con 300 equipos no cabía en una C —254 direcciones— y una B le daba 65 534, de las que malgastaba casi todas.

CIDR elimina las clases y admite prefijos de cualquier longitud. Con él llegan dos cosas:

- **Subredes**: partir una red asignada en varias más pequeñas alargando el prefijo.
- **Agregación de rutas**: anunciar varias redes contiguas como un solo prefijo más corto, con lo que las tablas de los encaminadores del núcleo crecen mucho menos.

Ejemplo de subredes. De `192.168.1.0/24` hay que sacar cuatro subredes:

Subred	Rango	Difusión	Asignables
`192.168.1.0/26`	`.1` – `.62`	`.63`	62
`192.168.1.64/26`	`.65` – `.126`	`.127`	62
`192.168.1.128/26`	`.129` – `.190`	`.191`	62
`192.168.1.192/26`	`.193` – `.254`	`.255`	62

Cuatro subredes son dos bits más de prefijo: `/24 + 2 = /26`. La regla general es que para k subredes iguales hacen falta $\lceil \log_2 k \rceil$ bits, y para h hosts en una subred hacen falta $\lceil \log_2 (h+2) \rceil$ bits de host.

Cuando las subredes necesitan tamaños distintos se usan **máscaras de longitud variable**: se asignan primero las más grandes, y cada bloque se coloca en una frontera múltiplo de su tamaño. Ese último requisito es lo que más se falla en los ejercicios: una `/26` no puede empezar en `.32`.

2.2.3 Formato del datagrama

La cabecera IPv4 son 20 octetos sin opciones:

Campo	Bits	Para qué
Versión	4	4 o 6

Campo	Bits	Para qué
Longitud de cabecera	4	en palabras de 32 bits; 5 si no hay opciones
Tipo de servicio	8	prioridad y trato del datagrama
Longitud total	16	cabecera más datos, en octetos; máximo 65 535
Identificación	16	mismo valor en todos los fragmentos de un datagrama
Indicadores	3	`DF` no fragmentar, `MF` hay más fragmentos
Desplazamiento	13	posición del fragmento, en unidades de 8 octetos
Tiempo de vida	8	se decrementa en cada salto; a 0 se descarta
Protocolo	8	qué hay dentro: 1 ICMP, 6 TCP, 17 UDP
Suma de comprobación	16	solo de la cabecera, no de los datos
Direcciones origen y destino	32 + 32	quién y a quién

Tres detalles que se preguntan y se olvidan:

- **El TTL evita los bucles.** Si el encaminamiento se equivoca y un datagrama da vueltas, el TTL lo mata. Sin él, un bucle transitorio saturaría el enlace para siempre.
- **La suma de comprobación cubre solo la cabecera**, y hay que recalcularla en cada salto porque el TTL cambia. Los datos los comprueba la capa de transporte.
- **El desplazamiento va en unidades de 8 octetos**, no de octetos. Es la causa de que todo fragmento salvo el último tenga una longitud de datos múltiplo de 8.

2.2.4 Fragmentación

Cada tecnología de enlace impone una unidad máxima de transferencia: 1500 octetos en Ethernet, menos en otras. Si un datagrama es mayor que la MTU del siguiente enlace, el encaminador lo parte.

Ejemplo. Un datagrama de 4000 octetos —20 de cabecera y 3980 de datos— hacia un enlace con MTU 1500:

Fragmento	Datos	Identificación	MF	Desplazamiento
1	1480	777	1	0
2	1480	777	1	185
3	1020	777	0	370

Los datos por fragmento son $1500 - 20 = 1480$, y 1480 es múltiplo de 8. Los desplazamientos son 0, $1480/8 = 185$ y $2960/8 = 370$.

Dos reglas que causan casi todos los errores en los ejercicios:

- **El reensamblado solo lo hace el destino**, nunca un encaminador intermedio. Los fragmentos pueden seguir caminos distintos, así que ningún nodo del camino los ve todos.
- **Perder un fragmento equivale a perder el datagrama entero**, porque no hay retransmisión de fragmentos sueltos. Por eso la fragmentación es cara y se evita: con el bit `DF` puesto, el emisor descubre la MTU mínima del camino y envía datagramas que caben. En IPv6 la fragmentación en tránsito directamente no existe.

2.3 Encaminamiento

Calcular por dónde va cada datagrama. La tabla de reenvío asocia prefijos de destino con la interfaz de salida y el siguiente salto:

Destino	Máscara	Siguiente salto	Interfaz
`192.168.1.0`	`/24`	directo	eth0
`10.0.0.0`	`/8`	`192.168.1.254`	eth0
`0.0.0.0`	`/0`	`192.168.1.1`	eth0

La búsqueda usa la regla del **prefijo más largo**: entre todas las entradas que casan, gana la de máscara más larga. La última fila, con `/0`, casa siempre y es la ruta por defecto, así que solo se usa cuando ninguna otra sirve.

2.3.1 Algoritmos

Familia	Cómo	Qué conoce cada nodo	Protocolo
Vector de distancias	cada nodo anuncia a sus vecinos su distancia a cada destino	solo lo que le cuentan sus vecinos	RIP
Estado del enlace	cada nodo inunda la red con el estado de sus enlaces	el mapa completo	OSPF
Vector de caminos	se anuncia el camino completo, no solo la distancia	los sistemas autónomos que atraviesa	BGP

Vector de distancias aplica la ecuación de Bellman-Ford:

$$d_x(y) = \min_{v \in N(x)} \{\, c(x,v) + d_v(y) \,\}$$

Es simple y sufre el problema de la **cuenta a infinito**: si un enlace cae, dos nodos pueden estar anunciándose el uno al otro una ruta que ya no existe, incrementando el coste de uno en uno hasta alcanzar el infinito convenido. Se mitiga con horizonte dividido —no anunciar una ruta al vecino del que se aprendió— y con un valor de infinito bajo: en RIP, 16 saltos.

Estado del enlace hace lo contrario: cada nodo difunde a toda la red el estado de sus enlaces, todos construyen el mismo mapa y cada uno ejecuta Dijkstra por su cuenta. Converge más rápido y

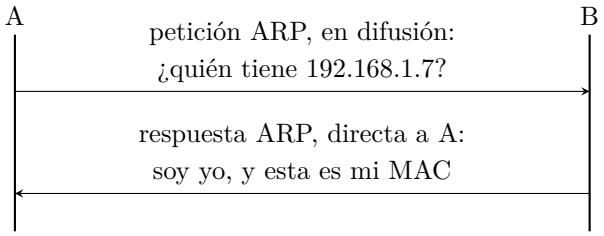
no tiene cuenta a infinito, a cambio de más tráfico de control y más cálculo. Es lo que usa OSPF, que además divide el dominio en áreas para que ese coste no crezca sin límite.

BGP encamina entre sistemas autónomos y su criterio no es la distancia sino la **política**: un operador elige por dónde sale su tráfico según los acuerdos comerciales que tenga, aunque el camino sea más largo. Anunciar el camino completo sirve además para detectar bucles: si un sistema autónomo se ve a sí mismo en la ruta anunciada, la descarta.

De ahí la división que ordena el encaminamiento en Internet: **dentro de un sistema autónomo se optimiza el coste; entre sistemas autónomos se aplica la política**.

2.4 ARP: la asociación con la capa de enlace

IP encamina hasta la última red, y en ella el datagrama tiene que ir dentro de una trama dirigida a una dirección física de 48 bits. **ARP** traduce una dirección IP de la misma red en su dirección física.



La petición va en **difusión**, porque no se sabe a quién preguntar; la respuesta va **directa**, porque el que responde ya conoce la dirección física del que preguntó, que venía en la petición. El resultado se guarda en una caché con caducidad, para no preguntar por cada datagrama.

ARP solo funciona dentro de la misma red. Si el destino está fuera, el emisor no pregunta por él: pregunta por la dirección física de su encaminador por defecto y le entrega la trama a él. Es el error de concepto más frecuente del tema.

2.5 ICMP

El protocolo de control y de errores de la capa de red. Viaja **dentro** de datagramas IP aunque sea parte de la misma capa, y sirve para informar de lo que IP no puede resolver.

Tipo	Mensaje	Cuándo
0 y 8	respuesta y petición de eco	`ping`
3	destino inalcanzable	red, host, protocolo o puerto no accesible
5	redirección	hay un encaminador mejor en esta red
11	tiempo excedido	el TTL llegó a cero
12	problema de parámetro	cabecera mal formada

Todo mensaje de error incluye la cabecera IP del datagrama que lo provocó más sus primeros 8 octetos de datos, que es justo lo necesario para que el origen sepa a qué conexión corresponde: en esos 8 octetos están los puertos de TCP o UDP.

`traceroute` se construye con esto: se envían datagramas con TTL 1, 2, 3…, y cada encaminador del camino devuelve un «tiempo excedido» que lo identifica.

Y una limitación: **ICMP no hace fiable a IP**. Informa de errores cuando puede, y sus mensajes también se pueden perder. Además muchas redes los filtran, y por eso un `ping` sin respuesta no prueba que el destino esté caído.

2.6 NAT

La traducción de direcciones permite que una red entera con direcciones privadas salga a Internet con una sola dirección pública.

El encaminador que hace NAT mantiene una tabla y reescribe las cabeceras:

Interno	Puerto	Externo	Puerto traducido
`192.168.1.10`	5000	`88.20.30.40`	40001
`192.168.1.11`	5000	`88.20.30.40`	40002

Al salir, sustituye la dirección y el puerto de origen; al volver, hace lo contrario. Como distingue las conexiones por el puerto traducido, con una sola dirección pública caben miles de conexiones simultáneas.

Ventajas	Inconvenientes
ahorra direcciones públicas	rompe la conectividad extremo a extremo
oculta la estructura interna	un equipo interno no es alcanzable desde fuera sin configuración
se puede cambiar de proveedor sin renumerar	los protocolos que llevan direcciones dentro de los datos se rompen

El segundo inconveniente contradice el principio de que la inteligencia está en los extremos: NAT guarda estado de las conexiones en un nodo intermedio, y con ello un elemento de la red pasa a mirar dentro de la capa de transporte. La tercera fila es su consecuencia práctica, y es la razón de que las aplicaciones entre pares necesiten mecanismos añadidos para atravesarlo.

El tratamiento de IP, del direccionamiento y del encaminamiento sigue a García Teodoro et al., 2007 y Kurose y Ross, 2017; los algoritmos de encaminamiento y sus análisis, a Tanenbaum, 2011 y Stallings, 2006.

Capítulo 3

Capa de transporte

Tema 3 del programa, bloque 3. Comunicar procesos, no máquinas: UDP y TCP, y los tres controles que hacen fiable un servicio que no lo es.

3.1 Qué añade la capa de transporte

La capa de red entrega datagramas de una máquina a otra. La de transporte los entrega **de un proceso a otro**, y solo existe en los extremos.

Función	UDP	TCP
Multiplexación por puertos	sí	sí
Detección de errores	sí, opcional en IPv4	sí
Orientado a conexión	no	sí
Entrega fiable	no	sí
Entrega ordenada	no	sí
Control de flujo	no	sí
Control de congestión	no	sí

3.1.1 Puertos y sockets

Un **puerto** es un número de 16 bits que identifica al proceso dentro de la máquina.

Rango	Uso
0 – 1023	bien conocidos: 80 HTTP, 25 SMTP, 53 DNS
1024 – 49151	registrados
49152 – 65535	dinámicos, para el extremo cliente

Cómo se identifica una comunicación es distinto en cada protocolo, y es una diferencia de fondo:

- **UDP**: el socket es la pareja (dirección destino, puerto destino). Dos clientes distintos que envían al mismo servidor llegan al mismo socket.
- **TCP**: el socket es la **cuaterna** (IP origen, puerto origen, IP destino, puerto destino). Dos clientes distintos producen dos conexiones distintas aunque el puerto de destino sea el mismo.

Por eso un servidor web atiende a miles de clientes en el puerto 80: cada conexión es una cuaterna diferente.

3.2 El protocolo UDP

Cabecera de 8 octetos: puerto origen, puerto destino, longitud y suma de comprobación. Nada más.

Lo que hace es multiplexar y comprobar errores. Lo que no hace es todo lo demás: no establece conexión, no retransmite, no ordena, no controla el flujo ni la congestión.

Y eso es una ventaja en algunos casos:

Motivo	Dónde importa
Sin establecimiento previo	DNS: una consulta y una respuesta, sin gastar un ida y vuelta en abrir
Cabecera de 8 octetos frente a 20	tráfico de muchos mensajes muy pequeños
Sin retransmisiones ni control de congestión	voz y vídeo: llegar tarde es peor que no llegar
Sin estado de conexión	un servidor aguanta muchos más clientes

En tiempo real, retransmitir un paquete de voz perdido es inútil: cuando llegue, el instante que representaba ya ha pasado. Es preferible el hueco.

La **suma de comprobación** de UDP cubre los datos y una pseudocabecera con las direcciones IP. Esa pseudocabecera es una violación deliberada de la separación entre capas, y está para detectar un datagrama entregado a la máquina equivocada.

3.3 El protocolo TCP

Servicio fiable, ordenado y orientado a conexión, sobre un IP que no garantiza nada.

3.3.1 Cabecera

Campo	Bits	Para qué
Puertos origen y destino	16 + 16	multiplexación
Número de secuencia	32	posición del primer octeto del segmento en el flujo
Número de asentimiento	32	siguiente octeto que se espera recibir
Longitud de cabecera	4	en palabras de 32 bits
Indicadores	6	`URG`, `ACK`, `PSH`, `RST`, `SYN`, `FIN`
Ventana	16	cuántos octetos puede recibir el emisor de este segmento
Suma de comprobación	16	cabecera, datos y pseudocabecera
Puntero de urgencia	16	con `URG`

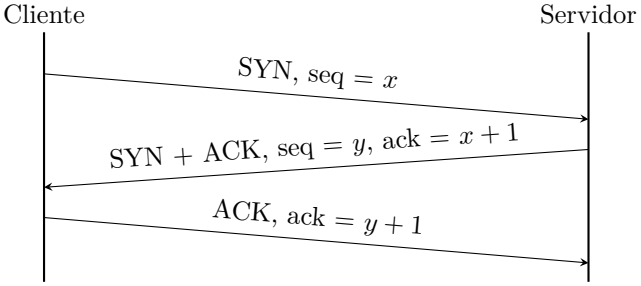
Dos cosas que hay que fijar desde el principio:

– **La numeración es por octetos, no por segmentos.** El número de secuencia es la posición del primer octeto de datos dentro del flujo.

– **El asentimiento es acumulativo y dice lo que se espera, no lo que se recibió.** Un `ACK = 5001` significa «tengo todo hasta el 5000, mándame desde el 5001».

3.3.2 Control de conexión

Apertura, en tres pasos:



Hacen falta **tres** y no dos porque los dos extremos tienen que sincronizar su numeración inicial y saber que el otro la ha recibido. Y los números iniciales se eligen al azar, no en cero: así un segmento retrasado de una conexión anterior entre los mismos puertos no se cuela en la nueva.

Cierre, en cuatro pasos, porque la conexión es dúplex y cada sentido se cierra por separado: `FIN`, `ACK`, `FIN`, `ACK`. Entre el primer par y el segundo, un extremo puede haber terminado de enviar y seguir recibiendo.

El que cierra primero queda en espera durante el doble del tiempo máximo de vida de un segmento antes de liberar la conexión. Es para poder reenviar el último `ACK` si se pierde, y para que ningún segmento rezagado aparezca en una conexión nueva con la misma cuaterna.

3.3.3 Control de errores

La fiabilidad se construye con tres piezas: numeración, asentimientos y retransmisión por temporizador.

El **temporizador de retransmisión** se calcula a partir del tiempo de ida y vuelta medido, suavizado y con un margen por su variabilidad:

$$\mathrm{RTT}_{\mathrm{est}} \leftarrow (1-\alpha)\,\mathrm{RTT}_{\mathrm{est}} + \alpha\,\mathrm{RTT}_{\mathrm{medido}}$$

$$\mathrm{RTO} = \mathrm{RTT}_{\mathrm{est}} + 4\,\mathrm{DesvRTT}$$

con $\alpha = 0{,}125$. Un temporizador demasiado corto retransmite lo que aún no se ha perdido y empeora la congestión; demasiado largo deja el enlace parado tras cada pérdida. Y al retransmitir se duplica el valor, que es la forma de no insistir sobre una red ya saturada.

Retransmisión rápida. Esperar al temporizador es lento. Si llegan tres asentimientos duplicados —el receptor pidiendo el mismo octeto una y otra vez—, el emisor deduce que ese segmento se perdió y lo reenvía sin esperar.

Que los `ACK` duplicados indiquen pérdida y no otra cosa es una inferencia: podrían venir de un reordenamiento. Tres seguidos es el compromiso entre reaccionar rápido y no reaccionar ante un simple desorden.

3.3.4 Control de flujo

Impide que el emisor desborde el **búfer del receptor**. El receptor anuncia en cada segmento su ventana disponible, y el emisor no envía más octetos sin asentir de los que quepan en ella.

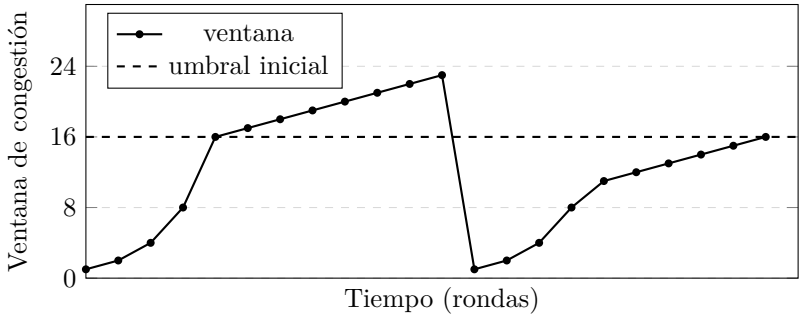
Si la ventana llega a cero, el emisor se detiene. Y para no quedarse bloqueado si el anuncio de reapertura se pierde, envía periódicamente un segmento de un octeto que fuerza al receptor a repetir su ventana.

El control de flujo protege al receptor. El de congestión protege a la red. Son mecanismos distintos con causas distintas, y confundirlos es el error clásico del tema.

3.3.5 Control de congestión

El emisor mantiene además una **ventana de congestión** que estima cuánto aguanta la red. Lo que puede enviar sin asentir es el mínimo de las dos ventanas.

TCP no recibe ninguna notificación de la red: **deduce la congestión de las pérdidas**. Y para deducirla tiene que provocarla, aumentando hasta que algo se pierde.



Las fases:

Fase	Cómo crece	Cuándo
Arranque lento	se dobla cada ida y vuelta: crecimiento exponencial	al empezar, y tras un vencimiento del temporizador
Evitación de la congestión	un segmento por ida y vuelta: crecimiento lineal	por encima del umbral
Recuperación rápida	se reduce a la mitad y se sigue en lineal	tras tres `ACK` duplicados

La reacción **no es la misma según cómo se detecte la pérdida**, y esa asimetría es deliberada:

- **Tres `ACK` duplicados** significan que los segmentos siguientes sí llegaron, o sea, que la red aún entrega. La ventana se reduce a la mitad.
- **Vencimiento del temporizador** significa que no llega nada. Se supone congestión grave y la ventana vuelve a 1.

El nombre «arranque lento» despista: es la fase que **más rápido crece**. Se llama así porque empieza en un solo segmento, frente a la alternativa previa de empezar enviando toda la ventana anunciada.

Y el efecto de conjunto: cuando muchas conexiones comparten un enlace, todas suben hasta perder, todas se recortan y todas vuelven a subir. Eso reparte la capacidad entre ellas de forma aproximadamente equitativa, sin que ningún nodo de la red arbitre nada. Es el mismo principio del tema 1: la inteligencia en los extremos.

El tratamiento de UDP, de TCP y de sus tres controles sigue a Kurose y Ross, 2017 y García Teodoro et al., 2007; los algoritmos de ventana y su análisis, a Tanenbaum, 2011 y Stallings, 2006.

Capítulo 4

Seguridad en redes

Tema 4 del programa, bloque 4. Cifrado, autenticación, funciones resumen, firma y certificados, y los protocolos que los combinan.

4.1 Qué hay que proteger

Propiedad	Qué garantiza	Con qué se consigue
Confidencialidad	que solo el destinatario lea el contenido	cifrado
Integridad	que el contenido no se ha alterado	funciones resumen
Autenticación	que el interlocutor es quien dice ser	firma, certificados
No repudio	que el emisor no pueda negar haberlo enviado	firma digital
Disponibilidad	que el servicio siga funcionando	fuera del alcance de la criptografía

Las cuatro primeras se consiguen con criptografía. La quinta no: contra un ataque de denegación de servicio no protege ningún cifrado, y por eso se trata con filtrado, dimensionado y limitación de tasa.

Los ataques que este tema considera:

Ataque	Qué hace	Contra qué propiedad
Escucha	leer el tráfico ajeno	confidencialidad
Modificación	alterar los mensajes en tránsito	integridad
Suplantación	hacerse pasar por otro	autenticación
Repetición	reenviar un mensaje válido capturado antes	autenticación
Intermediario	situarse entre los dos extremos	todas

4.2 Cifrado

4.2.1 Cifrado simétrico

La misma clave cifra y descifra.

$$C = E_K(M), \qquad M = D_K(C)$$

Ventajas	Inconvenientes
muy rápido, apto para grandes volúmenes	hay que compartir la clave por un canal seguro
claves cortas: 128 o 256 bits	n interlocutores necesitan $n(n-1)/2$ claves
	no da no repudio: los dos tienen la misma clave

Algoritmos: **AES** es el estándar actual, con claves de 128, 192 o 256 bits; DES está retirado, porque sus 56 bits de clave se recorren por fuerza bruta.

El problema del reparto de claves es el que motiva todo lo que sigue: **para compartir una clave en secreto haría falta un canal secreto, que es lo que se quería construir**.

4.2.2 Cifrado asimétrico

Cada participante tiene un par de claves: una **pública**, que se publica, y una **privada**, que no sale de su poder. Lo cifrado con una solo se descifra con la otra.

Se cifra con	Solo descifra	Sirve para
clave pública del destino	el destino, con su privada	confidencialidad
clave privada del emisor	cualquiera, con su pública	autenticación y no repudio

La segunda fila es la base de la firma digital: si algo se descifra con la clave pública de alguien, es que se cifró con su privada, y esa solo la tiene él.

Se apoya en problemas matemáticos difíciles de invertir: factorizar un número grande en RSA, el logaritmo discreto en Diffie-Hellman y en curva elíptica.

Su inconveniente es la **velocidad**: es varios órdenes de magnitud más lento que el simétrico, y las claves son mucho más largas para un nivel de seguridad equivalente.

4.2.3 El esquema híbrido

Es lo que se usa en la práctica, y combina lo bueno de los dos:

1. Se genera una clave simétrica **de sesión**, al azar y para un solo uso.
2. Se cifra esa clave con la **pública del destinatario** y se le envía.
3. Todo el tráfico se cifra con la clave de sesión, en simétrico.

El asimétrico resuelve el reparto de claves, que era su cometido, y el simétrico hace el trabajo pesado. Es lo que hacen TLS, SSH y el correo cifrado.

4.2.4 Diffie-Hellman

Permite que dos extremos acuerden una clave secreta **hablando por un canal público**, sin haber compartido nada antes.

Paso	Alicia	Público	Bruno
1	elige a	p, g	elige b

Paso	Alicia	Público	Bruno
2	calcula $A = g^a \bmod p$	A, B	calcula $B = g^b \bmod p$
3	$K = B^a \bmod p$		$K = A^b \bmod p$

Los dos llegan a $K = g^{ab} \bmod p$. Quien escucha ve p, g, A y B, y para obtener K tendría que resolver el logaritmo discreto.

Por sí solo no autentica, y es su límite: un atacante que se sitúe en medio puede acordar una clave con cada extremo y retransmitir descifrando y recifrando. Por eso el intercambio se firma, y ahí entran los certificados.

4.3 Funciones resumen

Una función resumen convierte un mensaje de cualquier tamaño en una cadena de longitud fija. Sus propiedades:

Propiedad	Qué significa
Unidireccional	del resumen no se puede recuperar el mensaje
Efecto avalancha	cambiar un bit cambia todo el resumen
Resistente a preimagen	dado h, no se puede encontrar m con $H(m) = h$
Resistente a colisiones	no se pueden encontrar $m_1 \neq m_2$ con $H(m_1) = H(m_2)$

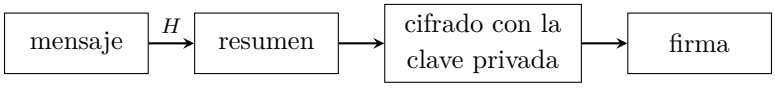
Algoritmos: **SHA-256** y SHA-3 son los recomendados; MD5 y SHA-1 están rotos, porque se saben construir colisiones, y no deben usarse para nada que dependa de la integridad.

Un resumen a secas **no protege contra modificación**: quien altera el mensaje recalcula el resumen. Hacen falta dos construcciones distintas según lo que se quiera:

- **HMAC**: resumen con una clave secreta compartida. Da integridad y autenticación de origen entre quienes comparten la clave, y no da no repudio.
- **Firma digital**: el resumen se cifra con la clave privada del emisor. Da además no repudio, porque solo él pudo producirla.

4.4 Firma digital y certificados

4.4.1 Firma



Para verificar, el receptor descifra la firma con la clave pública del emisor, calcula por su cuenta el resumen del mensaje recibido y compara. Si coinciden, el mensaje no se ha alterado y solo pudo firmarlo quien tiene la clave privada.

Se firma el resumen y no el mensaje por dos razones: el asimétrico es lento y solo opera sobre bloques del tamaño de la clave, y el resumen es corto y de tamaño fijo.

Firmar **no cifra**. Un mensaje firmado se lee sin problema; lo que garantiza es que no ha cambiado y de quién viene. Para las dos cosas hay que firmar y cifrar.

4.4.2 Certificados

Queda un problema abierto: **cómo saber que una clave pública es de quien dice**. Un atacante puede publicar una clave a nombre de otro.

Un **certificado digital** es un documento que asocia una identidad con una clave pública, firmado por una **autoridad de certificación** en la que ambos confían. Su contenido, en el formato X.509:

Campo	Qué lleva
Sujeto	a quién identifica
Clave pública	la que se certifica
Emisor	qué autoridad lo firma
Validez	fechas de inicio y de fin
Número de serie	identificador único dentro de esa autoridad
Firma	de la autoridad, sobre todo lo anterior

Los certificados forman una **cadena**: el del servidor lo firma una autoridad intermedia, a esta la firma otra, y arriba hay una autoridad raíz cuyo certificado está **autofirmado** y viene preinstalado en el navegador o el sistema.

La confianza descansa por completo en esa raíz preinstalada. Si una autoridad raíz se ve comprometida o emite certificados indebidos, todo lo que cuelga de ella deja de valer, y esa es la debilidad conocida del modelo.

Un certificado se puede **revocar** antes de caducar, si su clave privada queda expuesta. Se comprueba con listas de revocación o consultando en línea el estado del certificado.

4.5 Protocolos seguros

Combinan lo anterior, y cada uno actúa en una capa distinta:

Protocolo	Capa	Qué protege
TLS	entre transporte y aplicación	el tráfico de una conexión TCP
IPsec	red	todo el tráfico IP entre dos extremos o dos redes
SSH	aplicación	acceso remoto y túneles
S/MIME y PGP	aplicación	el correo, de extremo a extremo
WPA2 y WPA3	enlace	el acceso a una red inalámbrica

TLS es el más presente: es lo que convierte HTTP en HTTPS. Su negociación reúne todo el tema:

1. El cliente propone versiones y suites criptográficas.
2. El servidor elige y envía su **certificado**.
3. El cliente lo valida contra las autoridades en las que confía.
4. Se acuerda una clave de sesión, hoy con **Diffie-Hellman efímero**.
5. El resto de la comunicación va cifrada en simétrico, con HMAC para la integridad.

El paso 4 con claves efímeras da **secreto hacia el futuro**: como la clave de sesión no se deriva de la clave privada del servidor, quien grabe hoy el tráfico cifrado no podrá descifrarlo aunque esa clave privada se filtre mañana.

La capa importa. IPsec protege todo el tráfico entre dos puntos, incluidas las cabeceras de transporte, y es la base de una red privada virtual; TLS protege una conexión concreta y deja

a la vista con quién se habla. Lo que se cifra en una capa no protege lo que va por debajo: **con HTTPS, un observador no ve la página pedida y sí ve la dirección IP del servidor**.

Los fundamentos de criptografía aplicada a redes siguen a García Teodoro et al., 2007 y Stallings, 2006; los protocolos seguros y su encaje por capas, a Kurose y Ross, 2017 y Tanenbaum, 2011.

Capítulo 5

Capa de aplicación

Tema 5 del programa, bloque 5. Los protocolos que ve el usuario: nombres, web, correo y multimedia.

5.1 El paradigma cliente/servidor

Un proceso **servidor** espera en un puerto conocido; un proceso **cliente** inicia la comunicación cuando la necesita.

	Cliente	Servidor
Quién empieza	él	espera
Dirección	puede ser dinámica, y estar tras NAT	fija y conocida
Disponibilidad	intermitente	permanente
Con quién habla	con el servidor	con muchos clientes

La alternativa es el modelo **entre pares**, donde cada nodo es cliente y servidor a la vez. Su ventaja es que la capacidad crece con el número de participantes, en vez de repartirse entre ellos; sus problemas son encontrar quién tiene qué, y atravesar los NAT del tema 2.

5.1.1 Sockets

La interfaz que ofrece el sistema operativo a la capa de transporte. El esquema de un servidor TCP:

```
int s = socket(AF_INET, SOCK_STREAM, 0);
bind(s, direccion, longitud);      /* se asocia al puerto */
listen(s, cola);                   /* pasa a modo de espera */
int c = accept(s, NULL, NULL);     /* bloquea hasta que llega un cliente */
read(c, buffer, n);                /* c es la conexion, s sigue escuchando */
write(c, respuesta, m);
close(c);
```

Y el del cliente: `socket`, `connect`, `write`, `read`, `close`.

Dos detalles que se aprenden equivocándose:

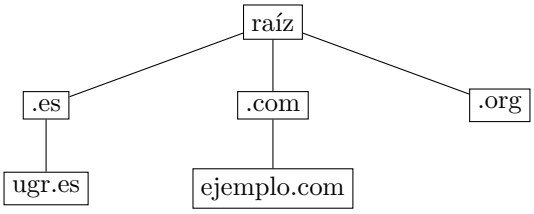
- **`accept` devuelve un socket nuevo.** El original sigue escuchando; el nuevo es esa conexión concreta. Cerrar el que no toca deja al servidor sin poder atender a nadie más.
- **TCP es un flujo, no un mensaje.** Un `write` de 1000 octetos puede llegar en tres `read`. Si el protocolo necesita delimitar mensajes, lo tiene que hacer él: con una longitud por delante, o con un separador.

Con UDP no hay `connect` ni `accept`: se usan `sendto` y `recvfrom`, y ahí cada datagrama sí es una unidad.

5.2 DNS

Traduce nombres de dominio en direcciones IP. Sin él habría que recordar direcciones, y ninguna máquina podría cambiar de dirección sin romper sus enlaces.

5.2.1 La jerarquía



Cada nivel delega en el siguiente: la raíz sabe quién gestiona `.es`, el servidor de `.es` sabe quién gestiona `ugr.es`, y ese último conoce los nombres de dentro. Ningún servidor conoce todo, y por eso el sistema escala.

Tipo de servidor	Qué sabe
Raíz	qué servidor gestiona cada dominio de primer nivel
De primer nivel	qué servidor gestiona cada dominio bajo él
Autoritativo	los registros de su propio dominio
Local o recursivo	nada propio; consulta por el cliente y guarda en caché

5.2.2 Resolución

- **Recursiva**: el cliente pregunta a su servidor local y recibe la respuesta final. El trabajo lo hace el servidor.
- **Iterativa**: cada servidor responde con la dirección del siguiente al que preguntar, y quien pregunta va recorriendo la jerarquía.

En la práctica el cliente hace una consulta recursiva a su servidor local, y ese resuelve de forma iterativa hacia arriba.

La **caché** es lo que hace el sistema viable: sin ella, los trece servidores raíz recibirían todas las consultas del mundo. Cada registro lleva un tiempo de vida que dice cuánto se puede conservar, y ese valor es también el retardo con el que se propaga un cambio: bajarlo antes de mover un servicio es la práctica habitual.

5.2.3 Registros

Tipo	Qué asocia
`A`	nombre a dirección IPv4
`AAAA`	nombre a dirección IPv6
`NS`	dominio al servidor que lo gestiona
`MX`	dominio al servidor de correo, con prioridad
`CNAME`	nombre a otro nombre

Tipo	Qué asocia
`PTR`	dirección a nombre, para la resolución inversa
`TXT`	texto libre; se usa para verificaciones y políticas de correo

DNS usa **UDP en el puerto 53**, porque una consulta y una respuesta caben en un datagrama y abrir una conexión TCP costaría más que la consulta. Pasa a TCP cuando la respuesta es grande o para las transferencias de zona entre servidores.

Y una limitación de origen: **DNS no autentica nada**. Una respuesta falsificada lleva al usuario a otro servidor sin que lo note. Eso es lo que DNSSEC corrige, firmando los registros con la cadena de confianza del tema 4.

5.3 La navegación web y HTTP

5.3.1 Direcciones

```
https://www.ugr.es:443/estudios/index.html?id=7#seccion
|___|   |_________||_||__________________||_____||______|
esquema      host  puerto      ruta      consulta  fragmento
```

El **fragmento** no se envía al servidor: lo usa el navegador para posicionarse dentro del documento recibido.

5.3.2 El protocolo

HTTP es **sin estado**: cada petición se atiende sin memoria de las anteriores. Eso simplifica enormemente el servidor y obliga a construir la sesión por encima, con cookies.

Una petición:

```
GET /estudios/index.html HTTP/1.1
Host: www.ugr.es
User-Agent: navegador/1.0
Accept: text/html
Connection: keep-alive
```

Y una respuesta:

```
HTTP/1.1 200 OK
Date: Mon, 16 Aug 2026 10:00:00 GMT
Server: nginx
Content-Type: text/html; charset=utf-8
Content-Length: 4096

<!doctype html> ...
```

Método	Qué hace
`GET`	pide un recurso
`HEAD`	pide solo la cabecera
`POST`	envía datos al servidor

Método	Qué hace
`PUT`	crea o sustituye un recurso
`DELETE`	lo borra
`OPTIONS`	pregunta qué métodos admite

Código	Familia	Ejemplos
1xx	informativo	100 Continue
2xx	éxito	200 OK, 204 No Content
3xx	redirección	301 permanente, 304 no modificado
4xx	error del cliente	400, 403, 404
5xx	error del servidor	500, 502, 503

La cabecera **`Host` es obligatoria en HTTP/1.1**, y es lo que permite servir muchos dominios desde una misma dirección IP: sin ella el servidor no sabría cuál de sus sitios se pide.

5.3.3 Conexiones persistentes

En HTTP/1.0 cada objeto abría y cerraba su propia conexión TCP. Una página con treinta imágenes pagaba treinta aperturas en tres pasos, y treinta arranques lentos del tema 3.

HTTP/1.1 reutiliza la conexión para varios objetos, y es de las mejoras que más se notan. HTTP/2 va más allá y multiplexa varias peticiones sobre una sola conexión, con lo que una respuesta lenta deja de bloquear a las que van detrás.

5.3.4 Caché y cookies

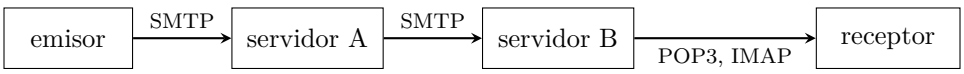
La **caché** evita transferir de nuevo lo que no ha cambiado. El cliente pregunta con `If-Modified-Since` y el servidor responde `304 Not Modified` sin cuerpo, o envía el recurso completo. Ahorra tráfico y latencia.

Las **cookies** dan estado a un protocolo que no lo tiene: el servidor envía `Set-Cookie`, el cliente lo devuelve en cada petición siguiente, y así el servidor reconoce al usuario entre peticiones. Es lo que sostiene las sesiones de usuario, y también el seguimiento entre sitios.

5.4 Correo electrónico

Tres piezas, y el reparto de tareas importa:

Pieza	Qué es
Agente de usuario	el programa de correo del usuario
Servidor de correo	recibe, almacena y reenvía
Protocolos	SMTP para enviar, POP3 e IMAP para recoger



SMTP empuja y POP3 o IMAP tiran. El correo llega hasta el servidor del destinatario por iniciativa del emisor, y de ahí al buzón del usuario por iniciativa de este, que puede estar apagado cuando el mensaje llegó.

5.4.1 SMTP

Puerto 25 entre servidores, sobre TCP, con órdenes en texto:

```
HELO cliente.ejemplo.com
MAIL FROM: <ismael@ejemplo.com>
RCPT TO: <destino@otro.com>
DATA
Subject: prueba
...
.
QUIT
```

Un punto solo en una línea marca el final del mensaje. Y una consecuencia de haber nacido como protocolo de texto de siete bits: **para enviar adjuntos hay que codificarlos**, y eso es lo que hace MIME, declarando el tipo de contenido y codificando lo binario en base 64, con un 33 % de sobrecoste.

SMTP no autentica al remitente por diseño. El `MAIL FROM` lo pone quien envía y nadie lo comprueba, y de ahí viene la suplantación en el correo. Las defensas —SPF, DKIM y DMARC— se montaron después, y funcionan publicando en el DNS del dominio, con registros `TXT`, qué servidores pueden enviar en su nombre y con qué clave se firman sus mensajes.

5.4.2 POP3 e IMAP

	POP3	IMAP
Puerto	110	143
Modelo	descargar y borrar	mantener en el servidor
Carpetas en el servidor	no	sí
Varios dispositivos	mal: cada uno se lleva lo suyo	bien: todos ven el mismo estado
Uso sin conexión	completo	parcial

IMAP es el que encaja con varios dispositivos, que es la situación normal hoy. POP3 sigue teniendo sentido cuando se quiere un único archivo local y no depender del servidor.

Y las versiones cifradas de los tres —puertos 465 o 587, 995 y 993— son los mismos protocolos sobre TLS, con lo visto en el tema 4.

5.5 Aplicaciones multimedia

Voz y vídeo tienen requisitos distintos de los de la web:

Requisito	Web	Multimedia en tiempo real
Pérdidas	intolerables	tolerables en pequeña proporción
Retardo	tolerable	acotado, en torno a 150 ms en voz
Fluctuación	irrelevante	crítica
Ancho de banda	a ráfagas	sostenido

Por eso usan **UDP** con RTP por encima: TCP retransmitiría lo perdido, y ese paquete llegaría cuando ya no sirve, además de introducir una espera que rompe la fluidez.

Las técnicas para funcionar sobre una red del mejor esfuerzo:

- **Búfer de reproducción**: se acumulan unos cientos de milisegundos antes de empezar, y así la fluctuación del retardo no produce cortes. Es el compromiso entre latencia y continuidad.
- **Ocultación de pérdidas**: se rellena el hueco repitiendo o interpolando la trama anterior.
- **Adaptación de la tasa**: se cambia la calidad según la capacidad disponible. Es lo que hace la difusión adaptativa sobre HTTP, que trocea el vídeo y ofrece cada trozo en varias calidades.
- **Redes de distribución de contenido**: se replica el contenido cerca del usuario, con lo que baja el retardo de propagación del tema 1, que es el único que no se puede reducir de otra forma.

La distinción práctica: en **difusión** —vídeo bajo demanda— el búfer puede ser de segundos y ahí TCP y HTTP sirven perfectamente; en **conversación** el retardo total tiene que caber en unos 150 ms, y por eso hace falta UDP.

Los protocolos de aplicación siguen a Kurose y Ross, 2017 y García Teodoro et al., 2007; la programación con sockets, a Comer y Stevens, 2001; el tratamiento de DNS y del correo, también a Tanenbaum, 2011.

Capítulo 6

Seminarios y prácticas de laboratorio

El temario práctico de la guía docente: seis seminarios y tres prácticas.

6.1 Los seminarios

Seminario	Contenido
S1	creación de un laboratorio virtual de redes
S2	diagnóstico y resolución de fallos en redes
S3	resolución de problemas del tema 2: IP y encaminamiento
S4	creación de aplicaciones cliente/servidor
S5	resolución de problemas del tema 3: TCP y UDP
S6	resolución de problemas del tema 4: seguridad, DNS y SMTP

6.1.1 S1 · Laboratorio virtual

Montar varias máquinas y encaminadores sobre un solo equipo, con redes virtuales entre ellos. Lo que hay que dejar claro antes de empezar son los modos de red:

Modo	Qué permite
NAT	la máquina sale a Internet y no es alcanzable desde fuera
Red interna	las máquinas se ven entre sí y no salen
Adaptador puente	la máquina aparece como un equipo más de la red física
Solo anfitrión	se comunica con el anfitrión y no con el exterior

Una topología típica del laboratorio son dos redes internas unidas por una máquina con dos interfaces que actúa de encaminador. Y ahí el fallo de novato es olvidar habilitar el reenvío de paquetes en esa máquina: con `net.ipv4.ip_forward` a cero, todo está bien configurado y nada pasa de una red a la otra.

6.1.2 S2 · Diagnóstico

El método que ordena todo el diagnóstico es **subir por las capas**: no tiene sentido mirar el DNS si el enlace está caído.

Capa	Comprobación	Órdenes
Enlace	¿la interfaz está levantada y con dirección?	`ip link`, `ip addr`
Enlace	¿se ve al vecino de la misma red?	`ping` al encaminador, `ip neigh`
Red	¿hay ruta al destino?	`ip route`, `ping`, `traceroute`
Red	¿por dónde se va y dónde se corta?	`traceroute`, `mtr`
Transporte	¿el puerto está abierto y escuchando?	`ss -tlnp`, `nc -vz`
Aplicación	¿el nombre resuelve?	`dig`, `host`
Aplicación	¿el servicio responde?	`curl -v`, cliente propio

Dos casos que aparecen siempre y se distinguen así:

- **`ping` a la IP funciona y al nombre no**: el problema es DNS, no la red.
- **`ping` funciona y el servicio no responde**: la capa de red está bien; mirar si el proceso escucha, y después el cortafuegos.

Y la herramienta que resuelve lo que las demás no: **capturar el tráfico** con `tcpdump` o Wireshark. Un filtro concreto —`tcp port 80 and host 10.0.0.5`— evita ahogarse en tráfico irrelevante, y las capturas enseñan el saludo en tres pasos, las retransmisiones y la evolución de la ventana del tema 3 mejor que ninguna explicación.

Cuidado con lo que un `ping` sin respuesta demuestra, que es poco: muchas redes filtran ICMP a propósito, y un destino que no contesta puede estar perfectamente vivo. Es la limitación del tema 2.

6.1.3 S3, S5 y S6 · Problemas

Los tres seminarios de problemas cubren lo que más cae:

Seminario	Tipos de ejercicio
S3	subredes con máscara variable, tablas de reenvío, fragmentación, camino de un datagrama
S5	diagramas de tiempos de TCP, cálculo de números de secuencia y de asentimiento, evolución de la ventana de congestión, rendimiento
S6	esquemas de cifrado y firma, resolución DNS paso a paso, diálogo SMTP

Errores que se repiten, y que conviene comprobar antes de dar por bueno un resultado:

- **En subredes, cada bloque empieza en una frontera múltiplo de su tamaño.** Una `/26` no puede empezar en `.32`.
- **En fragmentación, el desplazamiento va en unidades de 8 octetos**, y los datos de todo fragmento salvo el último son múltiplo de 8.
- **En TCP, el asentimiento dice el siguiente octeto esperado**, no el último recibido.
- **Los segmentos `SYN` y `FIN` consumen un número de secuencia** aunque no lleven datos.

– **En rendimiento, distinguir transmisión de propagación.** Es el reparto del tema 1 y es lo que decide si el resultado depende del tamaño del paquete o de la distancia.

6.1.4 S4 · Aplicaciones cliente/servidor

Programar con sockets lo del tema 5. El servidor iterativo atiende a un cliente cada vez, y el concurrente lanza un proceso o un hilo por conexión.

Tres cosas que no se ven en el código y sí en la ejecución:

– **TCP entrega un flujo, no mensajes.** Hay que delimitar por longitud o por separador; suponer que un `write` corresponde a un `read` funciona en pruebas locales y falla en cuanto hay latencia.
– **`SO_REUSEADDR` evita el «dirección ya en uso»** al reiniciar el servidor. Viene del estado de espera del cierre de TCP, que el tema 3 explica.
– **El orden de los octetos.** Los números en la red van en orden de red, así que `htons` y `ntohs` no son opcionales entre máquinas distintas.

6.2 Las prácticas de laboratorio

6.2.1 P1 · Encaminamiento y cortafuegos

Encaminamiento **estático**: escribir a mano las rutas de cada equipo con `ip route add`, y comprobar el camino con `traceroute`. Es donde se ve que la regla del prefijo más largo del tema 2 decide de verdad, y que una ruta por defecto tapa los errores hasta que aparece un destino que sí tenía ruta específica.

Encaminamiento **dinámico**: levantar RIP u OSPF y observar cómo convergen las tablas, cómo reaccionan al caer un enlace y cuánto tardan. Es la comprobación experimental de la comparación entre vector de distancias y estado del enlace.

Cortafuegos con `iptables` o `nftables`. Lo que hay que entender es el orden de evaluación y la política por defecto:

Cadena	Cuándo se aplica
`INPUT`	el paquete va dirigido a esta máquina
`OUTPUT`	lo genera esta máquina
`FORWARD`	pasa por ella hacia otra red

Dos reglas prácticas: **la primera regla que casa decide**, así que el orden importa tanto como el contenido; y una política por defecto de descarte exige acordarse de permitir el tráfico de retorno, o la máquina puede enviar y no recibir respuestas. También aquí se configura el NAT del tema 2, con la cadena de posencaminamiento.

6.2.2 P2 · Servicios básicos

Servicio	Qué se configura	Del temario
DHCP	rangos, tiempos de concesión, reservas	direccionamiento del tema 2

Servicio	Qué se configura	Del temario
DNS	zona directa e inversa, registros `A`, `NS`, `MX`, `PTR`	tema 5
Web	servidor HTTP, sitios virtuales por nombre	tema 5

La comprobación no es «arranca el servicio», es **que responde lo que debe**: `dig` contra el servidor propio, `curl -v` contra cada sitio virtual, y una captura para ver la consulta y la respuesta.

Y el sitio virtual por nombre es la comprobación práctica de por qué la cabecera `Host` es obligatoria: dos sitios en la misma dirección IP solo se distinguen por ella.

6.2.3 P3 · Servicios avanzados

Servicio	Qué se configura	Del temario
Correo	SMTP para envío y recepción, IMAP para el buzón	tema 5
HTTPS	certificado, clave y redirección desde HTTP	tema 4
Túneles y red privada virtual	cifrado entre dos redes	tema 4

Al montar el correo aparece el problema de fondo del tema 5: **un servidor recién instalado es un retransmisor abierto** si no se restringe quién puede enviar a través de él, y eso lo convierte en emisor de correo no deseado en cuestión de horas. Restringir el reenvío y exigir autenticación es parte de la práctica, no un extra.

Con HTTPS se comprueba lo del tema 4 sobre un certificado real: un certificado autofirmado funciona criptográficamente y el navegador avisa, porque no hay cadena hasta una autoridad en la que confíe. Es exactamente el problema que los certificados vienen a resolver, visto desde el otro lado.

6.3 Lo que ya publica esta ficha

En `Resumenes/` están los resúmenes de los temas 1 a 5, en `Teoria/` las preguntas de examen resueltas paso a paso, y en `TestsFR/` los tres cuestionarios de las prácticas, hechos con `/md2html/`.

Los guiones y la orientación práctica siguen a García Teodoro et al., 2007; la programación con sockets, a Comer y Stevens, 2001, y las herramientas de diagnóstico y su lectura, a Kurose y Ross, 2017.

Bibliografía

Comer, D. E., & Stevens, D. L. (2001). *Internetworking with TCP/IP, Vol. III: Client-Server Programming and Applications, Linux/POSIX Sockets Version*. Prentice Hall International.

García Teodoro, P., Díaz Verdejo, J., & López Soler, J. M. (2007). *Transmisión de datos y redes de computadores*. Pearson.

Kurose, J. F., & Ross, K. W. (2017). *Computer Networking: A Top-Down Approach* (7.ª ed.). Addison-Wesley.

Stallings, W. (2006). *Comunicaciones y redes de computadores* (7.ª ed.). Prentice Hall International.

Tanenbaum, A. S. (2011). *Computer Networks* (6.ª ed.). Prentice Hall International.