



Teoría de las Comunicaciones

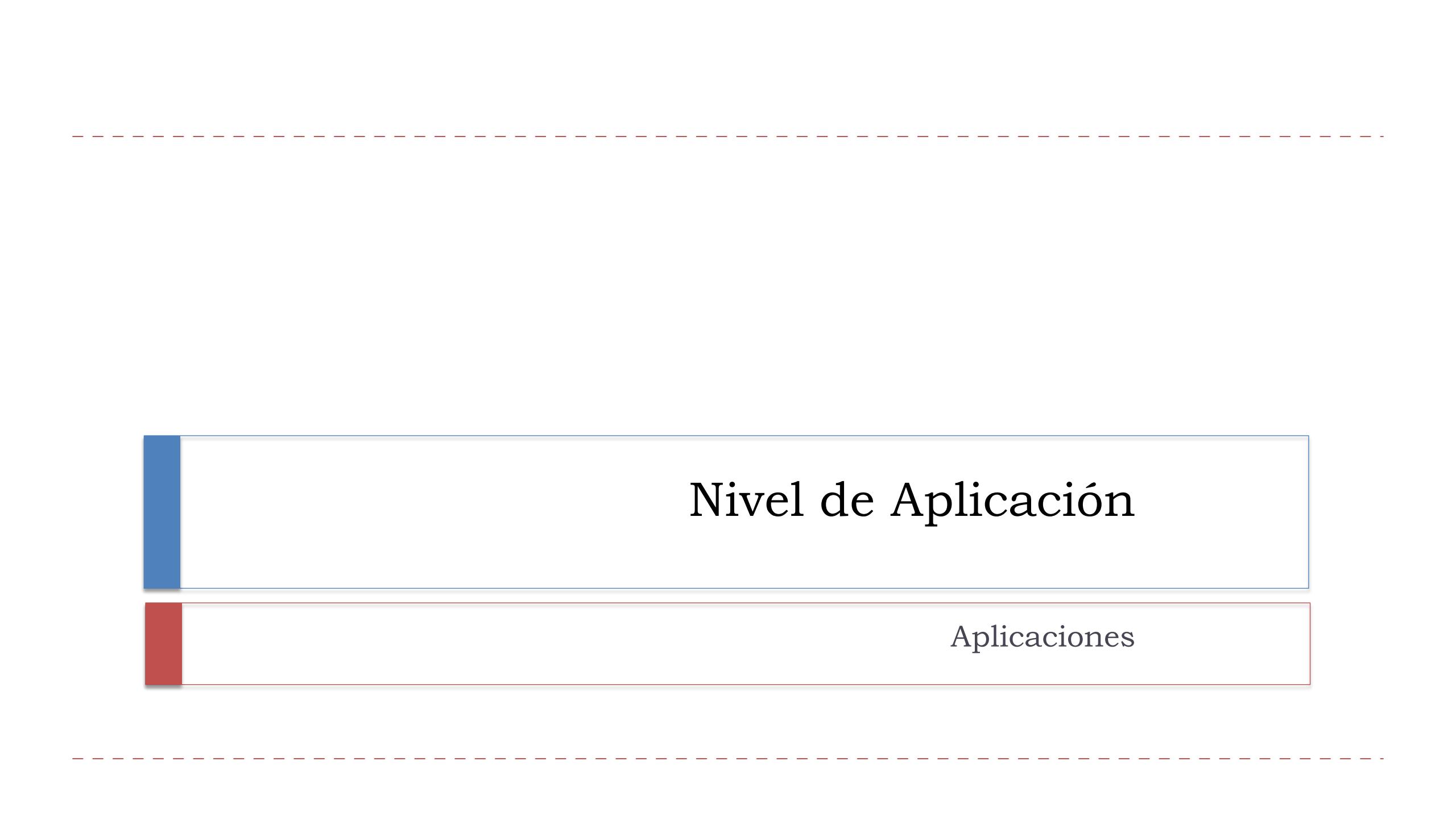
Edición 65 oficial, pero en realidad desde que se comenzó a dictar (a.k.a Redes) es la edición 74. Este es el último cuatrimestre que se dicta.

Dr. Claudio Enrique Righetti

28 octubre 2025

Segundo Cuatrimestre

**Departamento de Computación
Facultad de Ciencias Exactas y Naturales
Universidad de Buenos Aires
Argentina**

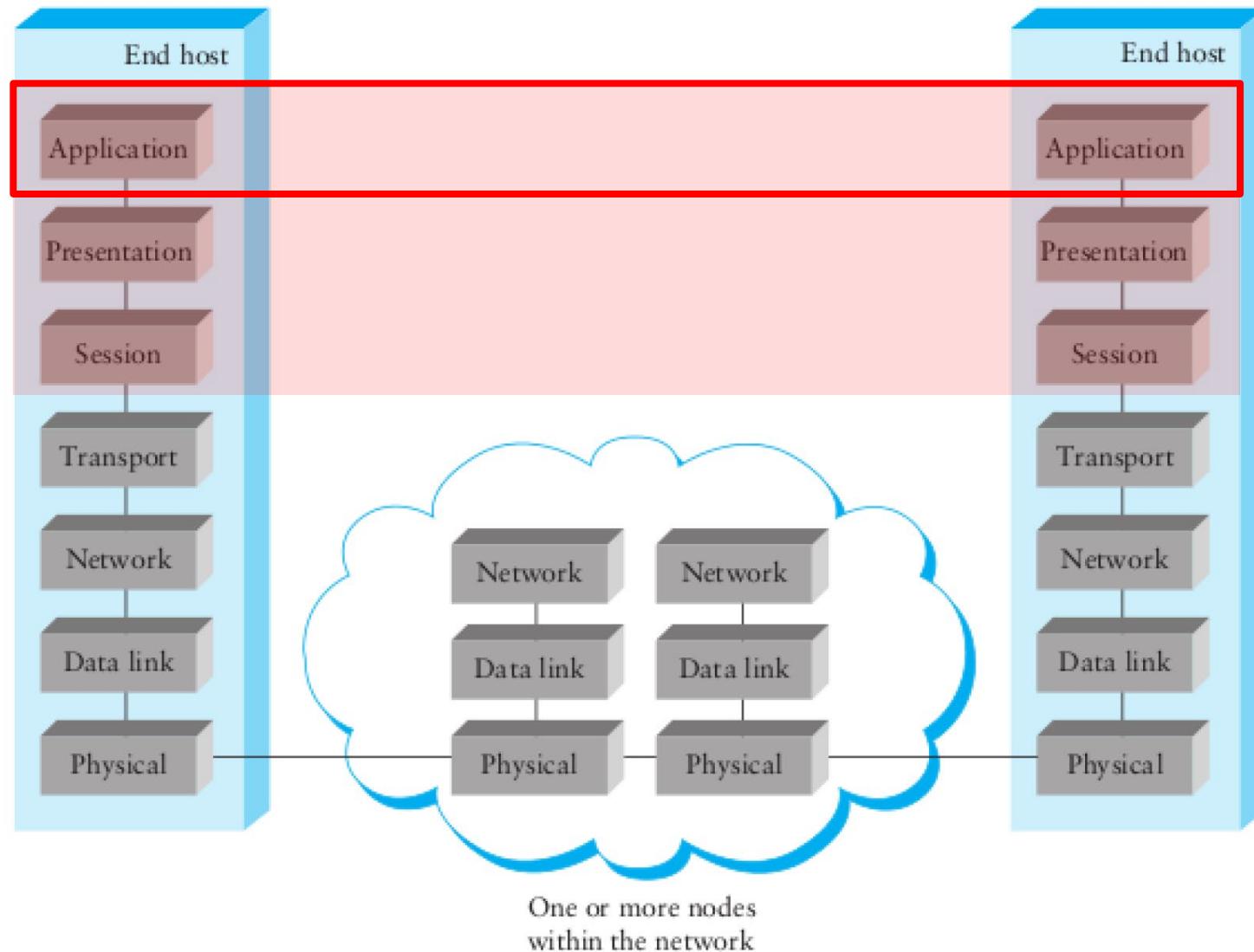


The diagram illustrates a layered architecture. At the top, a horizontal dashed red line spans the width of the slide. Below this line, there is a large white rectangular area. On the left side of this area, there is a vertical blue bar on the far left and a vertical red bar immediately to its right. To the right of the red bar is a wide white space containing the text "Nivel de Aplicación". Further down, another horizontal dashed red line marks the bottom of the main white area. Below this second line, there is a horizontal red-outlined rectangle. On the far left of this rectangle is a small red vertical bar. To the right of this red bar is a white space containing the text "Aplicaciones".

Nivel de Aplicación

Aplicaciones

Arquitectura en capas



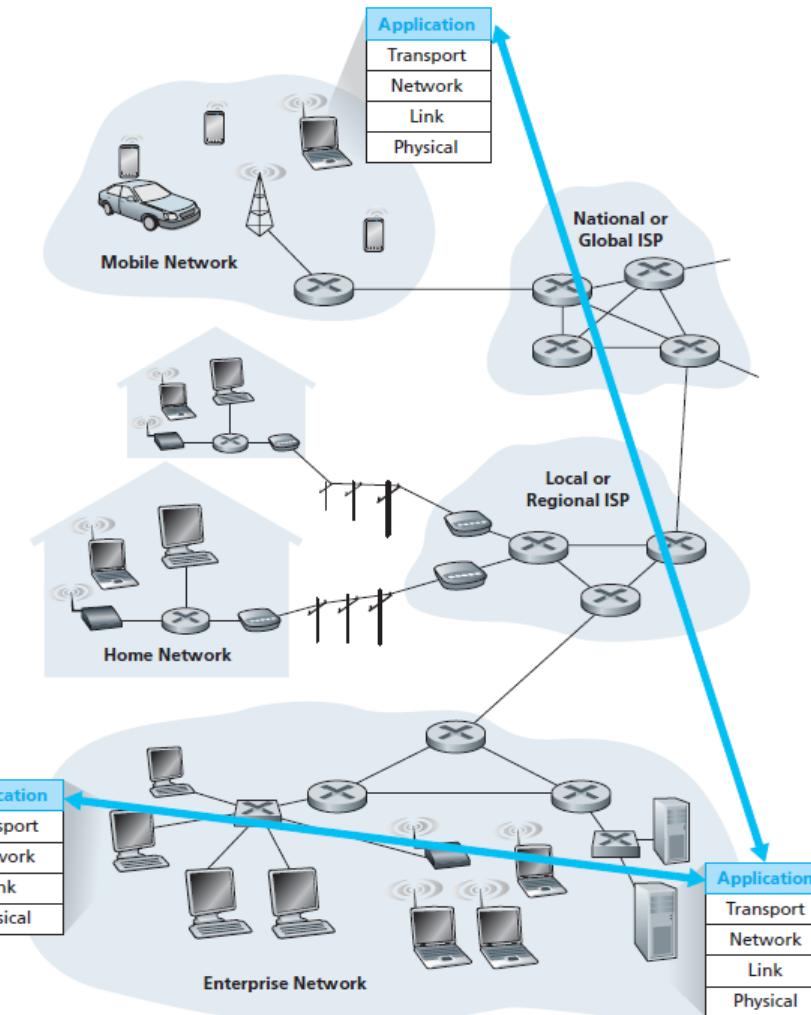
Aplicaciones y protocolos de la capa de aplicación

Aplicación: comunicación + procesos distribuídos.

- ▶ **Ejemplos:** correo electrónico, web, compartir archivos entre pares, mensajería instantánea.
- ▶ Funcionamiento en **sistemas finales** (hosts).
- ▶ Intercambio de **mensajes** para la implementación de aplicaciones.

Protocolos de capa de aplicación

- ▶ Son “una parte” de la aplicación.
- ▶ Definición de **mensajes** intercambiados entre las aplicaciones
 - ▶ Uso de **acciones** ejecutadas por **procesos** en los hosts.
 - ▶ Uso de **servicios** de **comunicación**
 - ▶ Proporcionados por los **protocolos de capas inferiores** (TCP, UDP).



Aplicaciones de red: terminología

Proceso: programas que se ejecutan en un host.

- ▶ En un mismo host, dos procesos se comunican entre sí utilizando comunicación interproceso (definidos por un sistema operativo).
- ▶ Los procesos que se ejecutan en diferentes hosts se comunican con un protocolo de capa de aplicación.

Agentes de usuario:

- interfaces con un usuario “por encima” y una red “por debajo”.
- ▶ Implementa interfaces de usuario y protocolos a nivel de aplicación.
 - ▶ Web: navegador.
 - ▶ Correo electrónico: lector de correo.
 - ▶ Transmisión de audio/vídeo: reproductor multimedia.



Características de los protocolos de capa de aplicación

- ▶ Tipos de mensajes intercambiados: típicamente **de petición y de respuesta**.
- ▶ **Sintaxis** de los tipos de mensajes:
 - ▶ qué campos hay en los mensajes y su estructura
- ▶ **Semántica** de los campos:
 - ▶ significado de la información en los campos.
- ▶ **Reglas** que determinan cuándo y cómo los procesos envían y responden a los mensajes.

Protocolos de dominio público:

- ▶ Definidos en RFCs.
- ▶ Permiten interoperabilidad.
- ▶ Por ejemplo: HTTP, SMTP.



Paradigma tipo cliente-servidor

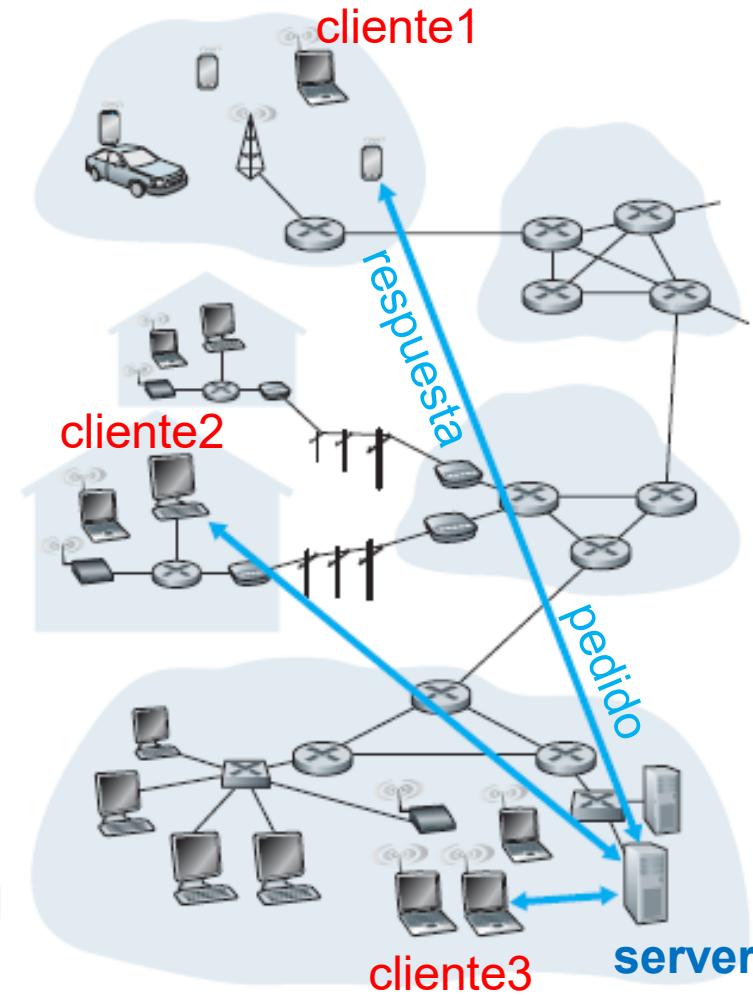
La aplicación de red “típicamente” tiene dos partes: *el cliente* y *el servidor*

Cliente:

- Inicia el contacto con el servidor (“habla primero”).
- Normalmente solicita un servicio del servidor.
- Web: cliente implementado en el navegador.
Correo electrónico: en un lector de correo, etc.

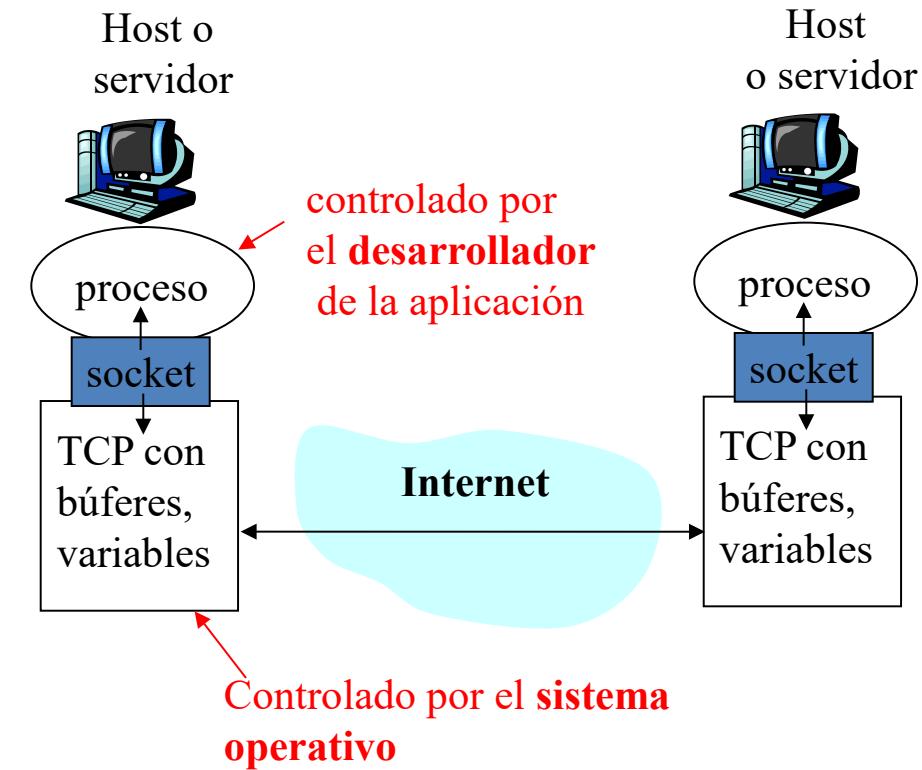
Servidor:

- Proporciona el servicio solicitado al cliente.
- Por ejemplo, el servidor de Web envía la página Web solicitada, el servidor de correo entrega el correo electrónico, etc.



Procesos que se comunican a través de la red

- ▶ El proceso envía/recibe mensajes hacia/desde su socket.
- ▶ El socket es análogo a una puerta:
 - ▶ El **proceso emisor** envía el mensaje por su puerta.
 - ▶ Asume la existencia de una **infraestructura de transporte** al otro lado de la puerta que transportará el mensaje hacia un socket en el **proceso receptor**.



- APIs:
 - Elección del protocolo de transporte
 - Posibilidad de fijar algunos parámetros -opciones- de operación.

Direccionamiento de procesos:

- ▶ Para que un proceso reciba mensajes debe tener un identificador.
- ▶ Cada host tiene una única dirección IP de 32 bits.
- ▶ ¿Basta con la dirección IP del host, en el que el proceso se ejecuta, para identificar el proceso?
 - ▶ No, ya que muchos procesos diferentes pueden estar ejecutándose en el mismo host.
- ▶ El identificador incluye tanto la dirección IP como los **números de puerto** asociados con el proceso del host.
- ▶ Ejemplos típicos de números de puerto:
 - ▶ Servidor HTTP: **80**
 - ▶ Servidor de correo: **25**



Servicios de los protocolos de transporte de Internet

Servicio TCP:

- ▶ Orientado a la conexión: Sistema requerido entre el cliente y el servidor.
- ▶ Transporte fiable entre el proceso emisor y el receptor.
- ▶ Control de flujo: el emisor no debe sobrecargar al receptor.
- ▶ Control de congestión: regulación del emisor si la red se sobrecarga.
- ▶ No proporciona: temporización, garantías de un ancho de banda mínimo.

Servicio UDP:

- ▶ Transferencia de datos no fiable entre el proceso emisor y el receptor.
- ▶ No proporciona: sistema de conexión, control de flujo, control de congestión, temporización ni garantía de ancho de banda.



¿Qué servicio de transporte necesita una aplicación? Criterios de selección

Pérdida de datos

- ▶ Ciertas aplicaciones (por ejemplo, audio) pueden tolerar algunas pérdidas.
- ▶ Otras aplicaciones (por ejemplo, transferencia de archivos, Telnet) requieren un 100% de **transferencia confiable** de datos.

Temporización

- ▶ Algunas aplicaciones (por ejemplo, Telefonía sobre Internet, juegos interactivos) requieren un **retardo artificial** para ser “efectivas” (**eliminar el jitter**)

Ancho de banda (capacidad)

- Algunas aplicaciones (por ejemplo, multimedia) requieren un mínimo de ancho de banda para ser “efectivas”.
- Otras aplicaciones (“flexibles”) hacen uso de cualquier ancho de banda que tengan a su disposición (por ejemplo, email)



Requisitos de los servicios de transporte para aplicaciones comunes

Aplicación	Pérdida de datos	Ancho de banda	Sensible al delay
Transf. de archivos	No pérdida	Flexible	No
Correo electrónico	No pérdida	Flexible	No
Documentos Web	No pérdida	Flexible	No
Audio/vídeo de tiempo real	Tolerante	Audio: 5Kbps-1Mbps Vídeo:10Kbps-	Sí, 100 mseg
Audio/vídeo almacenado	Tolerante	5Mbps	Sí, pocos seg
Juegos interactivos	Tolerante	Igual que el anterior	Sí, 100 mseg
Mensajería instantánea	No pérdida	Pocos Kbps-10Kbps Flexible	Depende



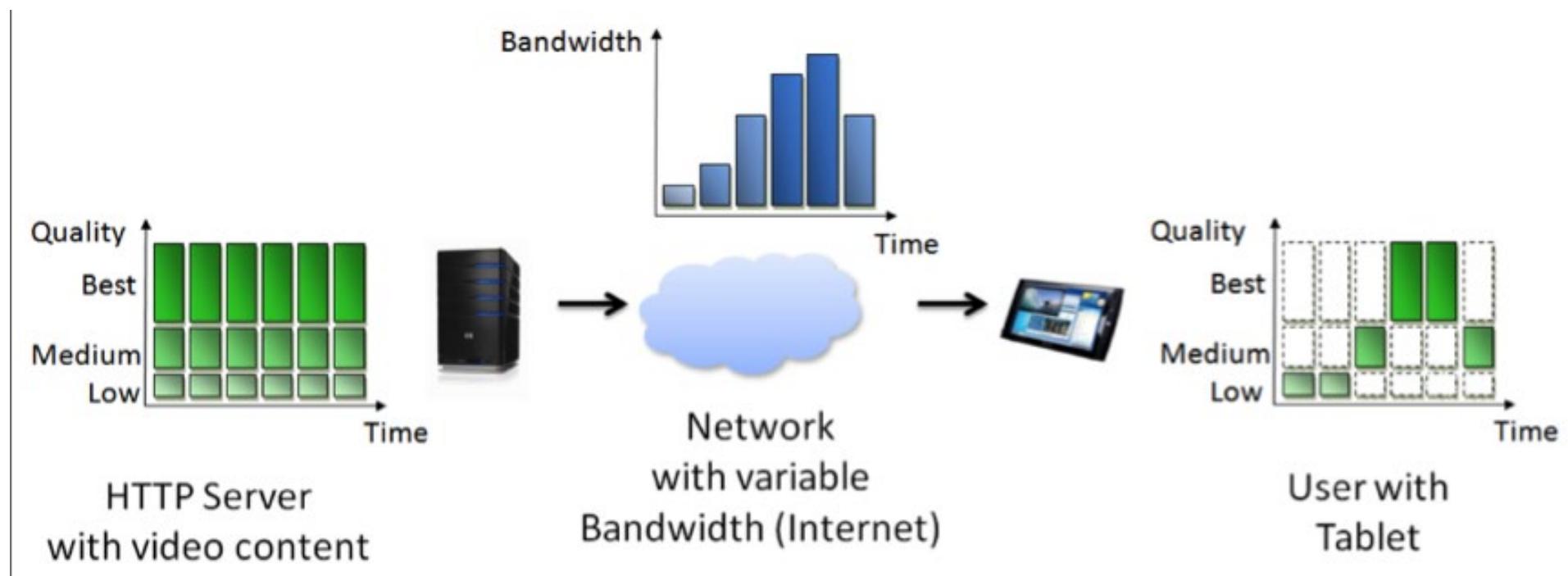
Aplicaciones de Internet: aplicación, protocolos de transporte

<u>Aplicaciones</u>	Protocolo de la capa de aplicación	Protocolo de transporte subyacente
Correo electrónico	SMTP [RFC 2821]	TCP
Acceso a terminales remotos	Telnet [RFC 854]	TCP
Web	HTTP [RFC 2616]	TCP
Transferencia de archivos	FTP [RFC 959]	TCP
Flujo de multimedia	HTTP (YouTube, Netflix) Propietario (Real Networks)	TCP o UDP
Telefonía Internet	SIP [RFC 3261], RTP [RFC 3550], Propietario (Skype)	Típicamente UDP A veces TCP



MPEG DASH

Dynamic Adaptive Streaming over HTTP (DASH), también conocido como MPEG-DASH, es una técnica de streaming de bitrate adaptativo



Fuente : [http://www.streamingmediaglobal.com/Articles/Editorial/Featured-Articles/Dynamic-Adaptive-Streaming-over-HTTP-\(DASH\)-Past-Present-and-Future-93275.aspx](http://www.streamingmediaglobal.com/Articles/Editorial/Featured-Articles/Dynamic-Adaptive-Streaming-over-HTTP-(DASH)-Past-Present-and-Future-93275.aspx)

The diagram illustrates the OSI model layers. At the top, a red dashed horizontal line spans the width of the slide. Below it, a blue rectangular box contains the text "Nivel de Aplicación". To its left is a vertical blue bar. Below the blue box is a red rectangular box containing the text "DNS: Domain Name System". To its left is a vertical red bar. This visual representation maps the application layer (Nivel de Aplicación) to the DNS protocol (DNS: Domain Name System), showing their relationship within the context of the OSI model.

Nivel de Aplicación

DNS: Domain Name System

El DNS

- DNS (Sistema de Nombres de Dominio) organiza máquinas dentro de dominios y **resuelve nombres de hosts en direcciones IP**
- DNS ha llegado a ser un sistema de base de datos distribuido generalizado para almacenar una variedad de información relacionada con la elección de un nombre
- Sistema distribuido y escalable.

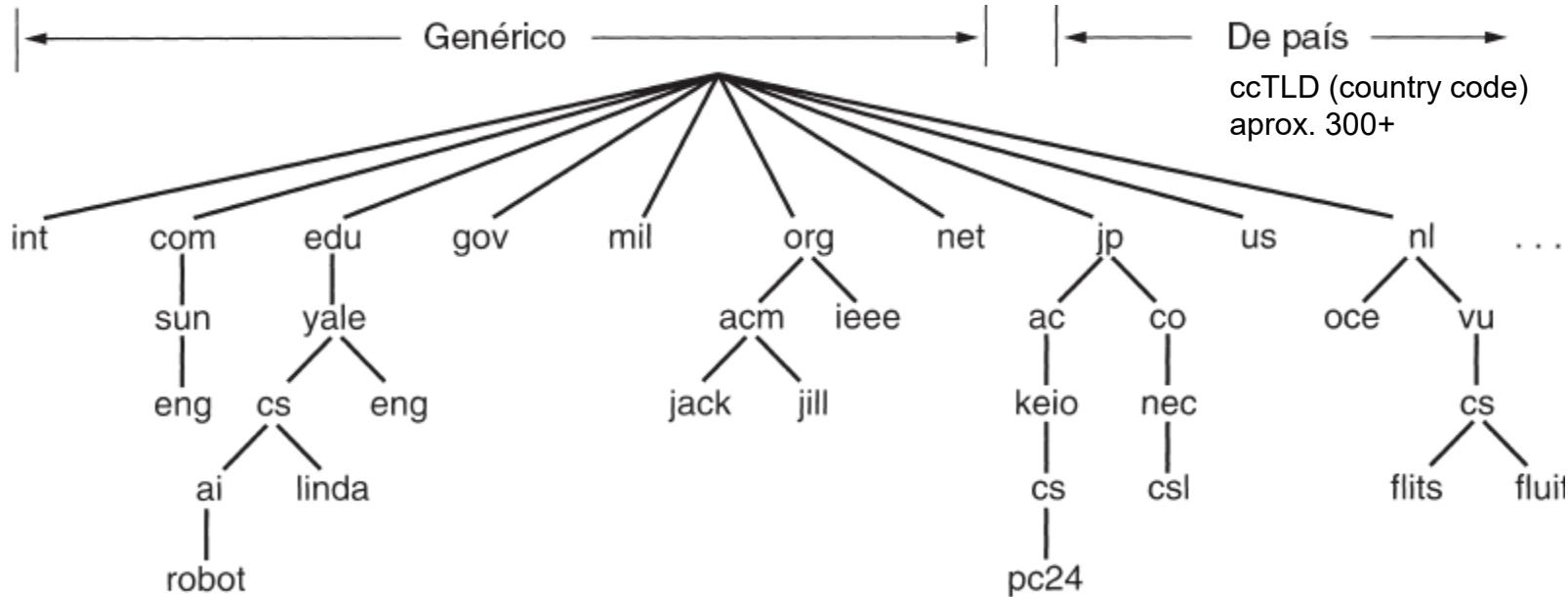


El DNS

- Se define en los RFCs 1034/1035
- Un **proceso de aplicación** llama a un procedimiento de biblioteca llamado **resolvedor** (solicita un servicio)
- Le pasa el nombre como parámetro (por ejemplo *gethostbyname*)
- El **resolvedor**, a su vez, envía un paquete UDP a un **servidor DNS local**, que después busca el nombre y **devuelve la dirección IP** al resolvedor, que entonces lo devuelve al solicitante
- Una vez que obtiene la dirección IP, el programa solicitante puede establecer una conexión TCP con el destino, o enviarle paquetes UDP

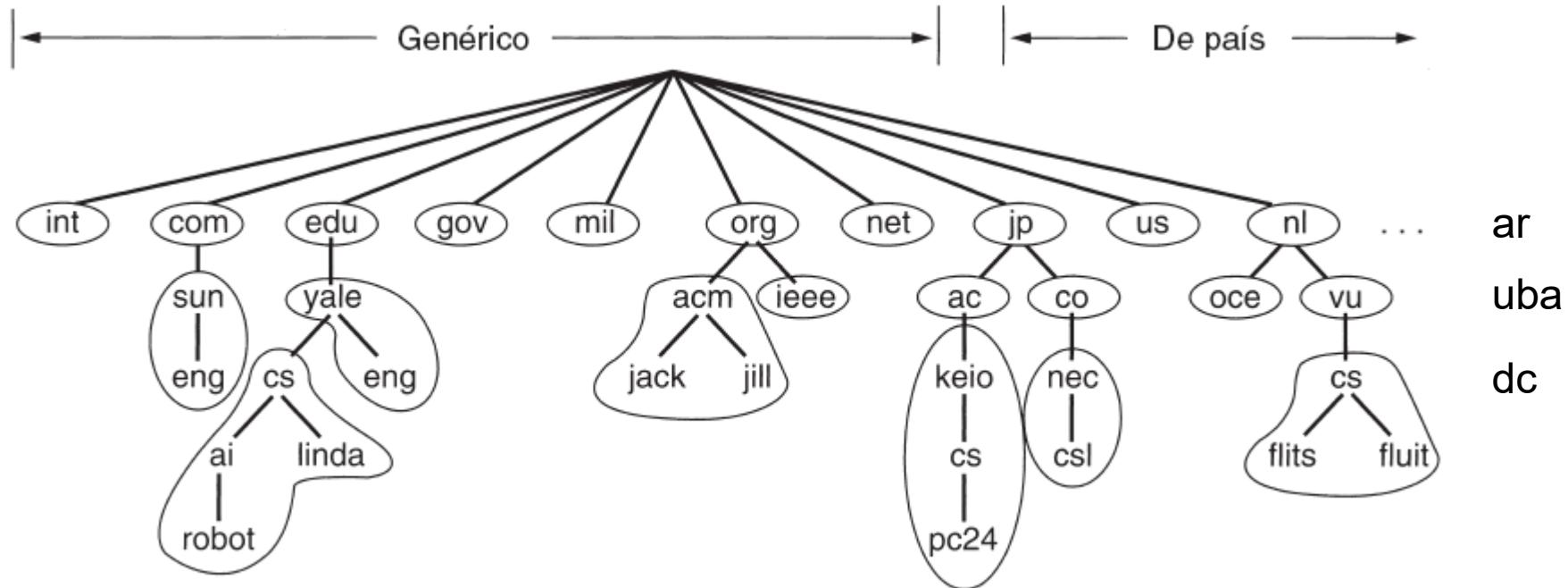


DNS: espacio de nombres



- Conceptualmente, Internet organiza su **espacio de nombres** en el orden de aprox. 1500+ dominios de nivel superior. Cada uno abarca muchos hosts.
- Un **dominio de nivel superior** o TLD (**Top-Level Domain**) es la más alta categoría de los FQDN (**Fully Qualified Domain Name**) que es traducida a direcciones IP por los DNS.
- Cada dominio se divide en **subdominios**, que a su vez se subdividen, etc.
- Los nombres son administrados por la Internet Corporation for Assigned Names and Numbers (**ICANN**).

Servidores de nombres: DNS zones



dc.uba.ar

Registros DNS: tiene cinco tuplas

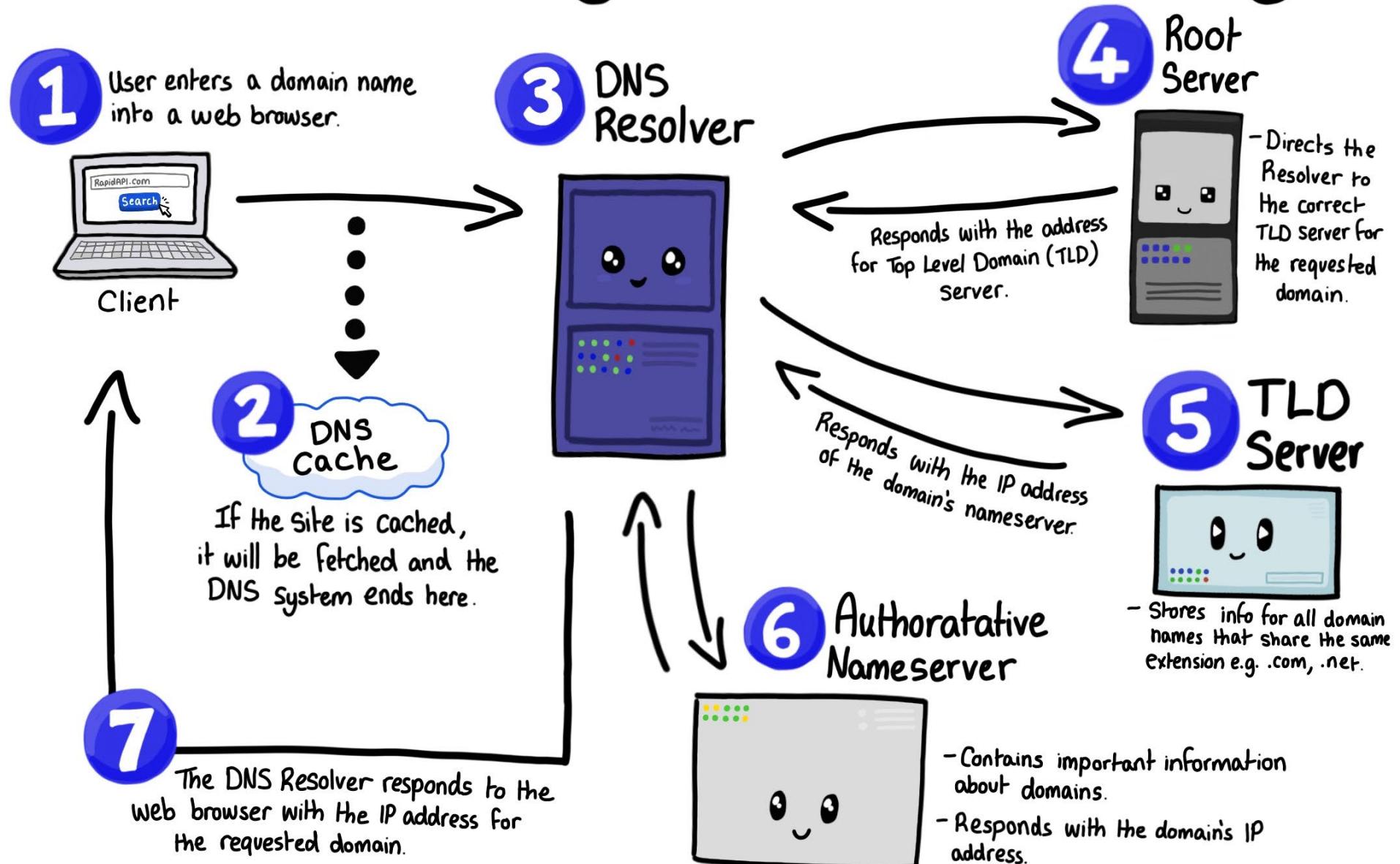
- ▶ **Nombre_dominio, Tiempo_de_vida, Clase**
- ▶ **Tipo, Valor:**

Tipo	Significado	Valor
SOA	Inicio de autoridad	Parámetros para esta zona
A	Dirección IP de un <i>host</i>	Entero de 32 bits
MX	Intercambio de correo	Prioridad, dominio dispuesto a aceptar correo electrónico
NS	Servidor de nombres	Nombre de un servidor para este dominio
CNAME	Nombre canónico	Nombre de dominio
PTR	Apuntador	Alias de una dirección IP
HINFO	Descripción del <i>host</i>	CPU y SO en ASCII
TXT	Texto	Texto ASCII no interpretado

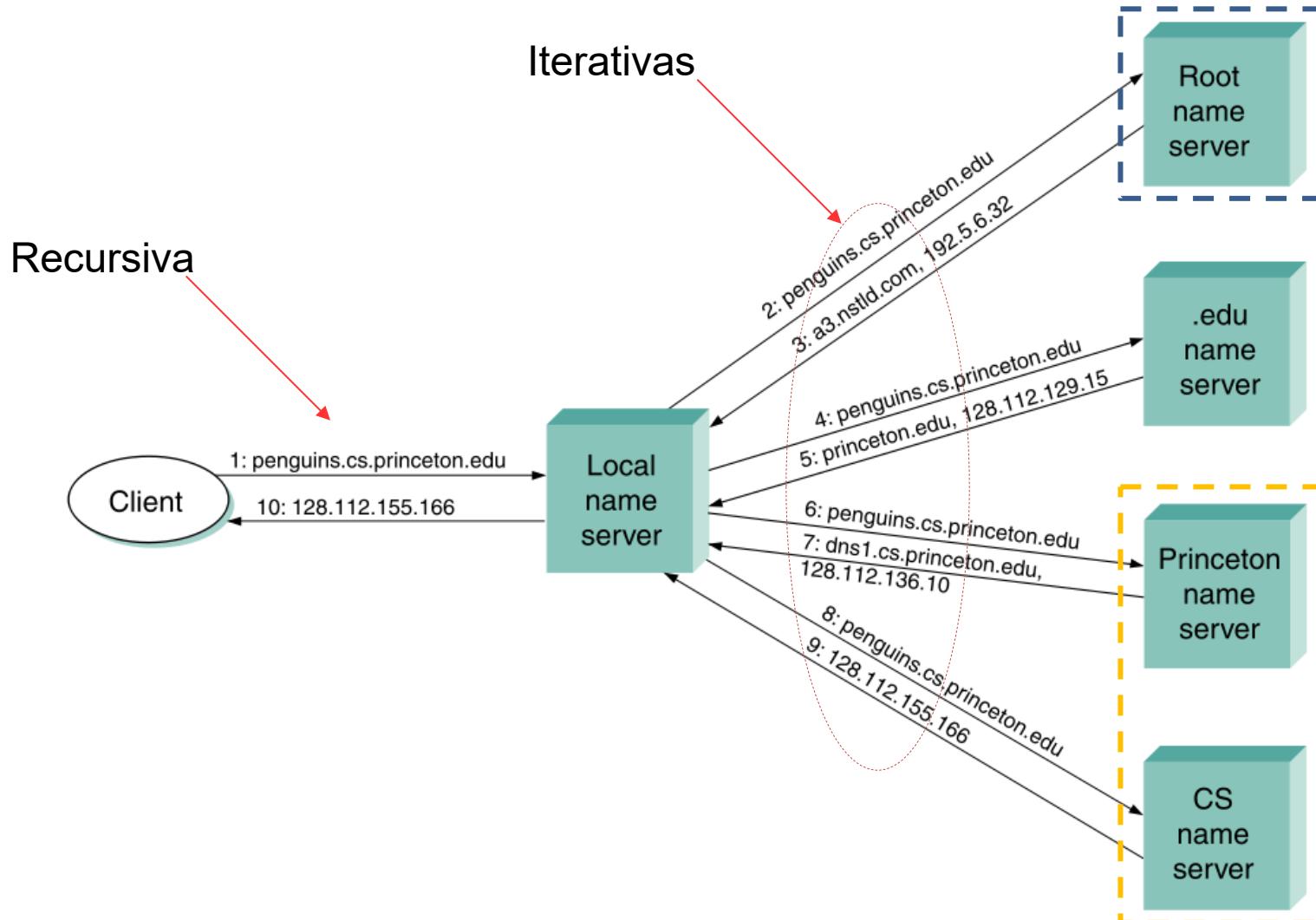
Un registro SOA proporciona el nombre de la **fuente primaria de información** sobre la **zona** del servidor de nombres, la dirección de correo electrónico de su administrador, un número de serie único y varias banderas y temporizadores.



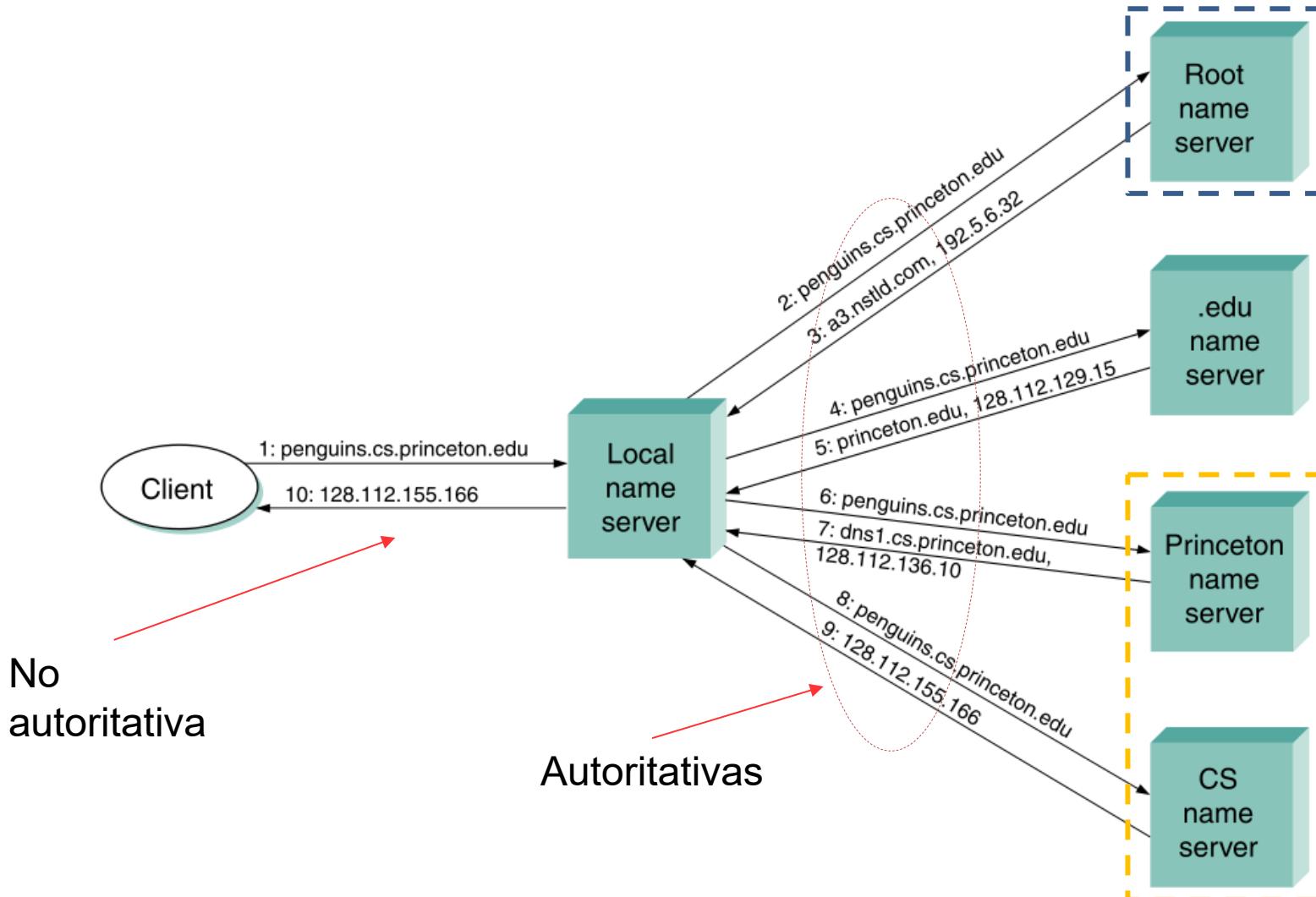
The DNS System Hierarchy



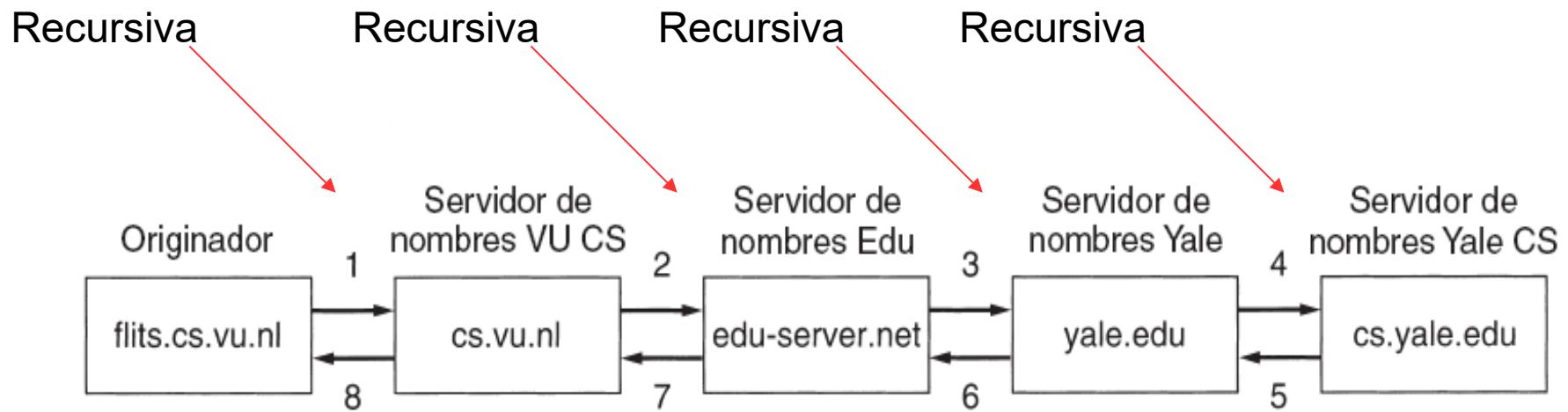
Tipos de consulta: iterativa y recursiva



Tipos de respuesta: autoritativa y no autoritativa



Consulta recursiva



www.cs.yale.edu



Nivel de Aplicación

SMTP: Simple Mail Transfer Protocol

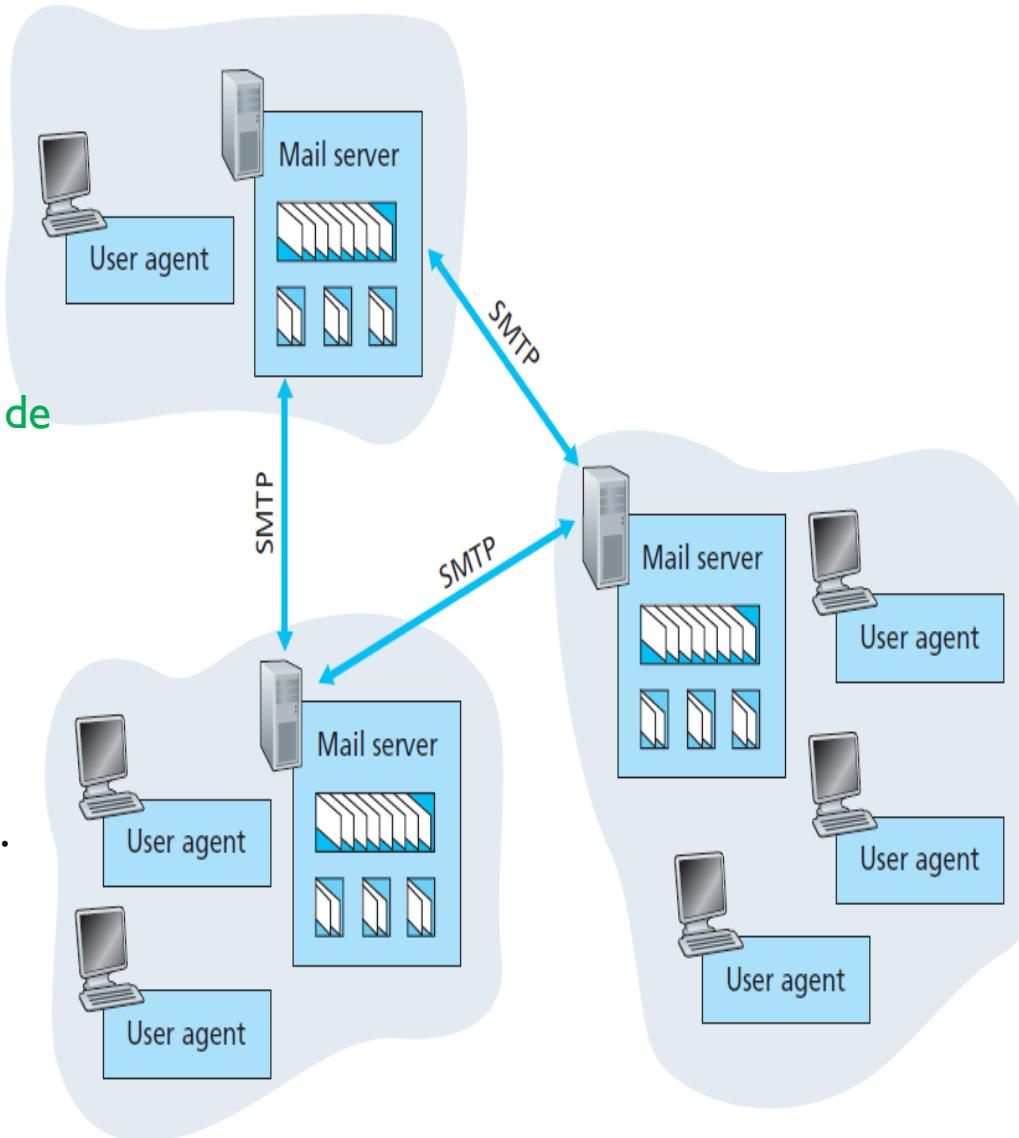
Correo electrónico

Los tres componentes principales:

1. Agentes de usuario.
2. Servidores de correo.
3. Protocolo simple de transferencia de correo: **SMTP**.

Agente usuario

- ▶ También conocido como “lector de correo”.
- ▶ Composición, edición y lectura de mensajes de correo.
- ▶ Por ejemplo: Outlook, Thunderbird.
- ▶ Salida y entrada de los mensajes almacenados en el servidor.



Key:



Outgoing
message queue

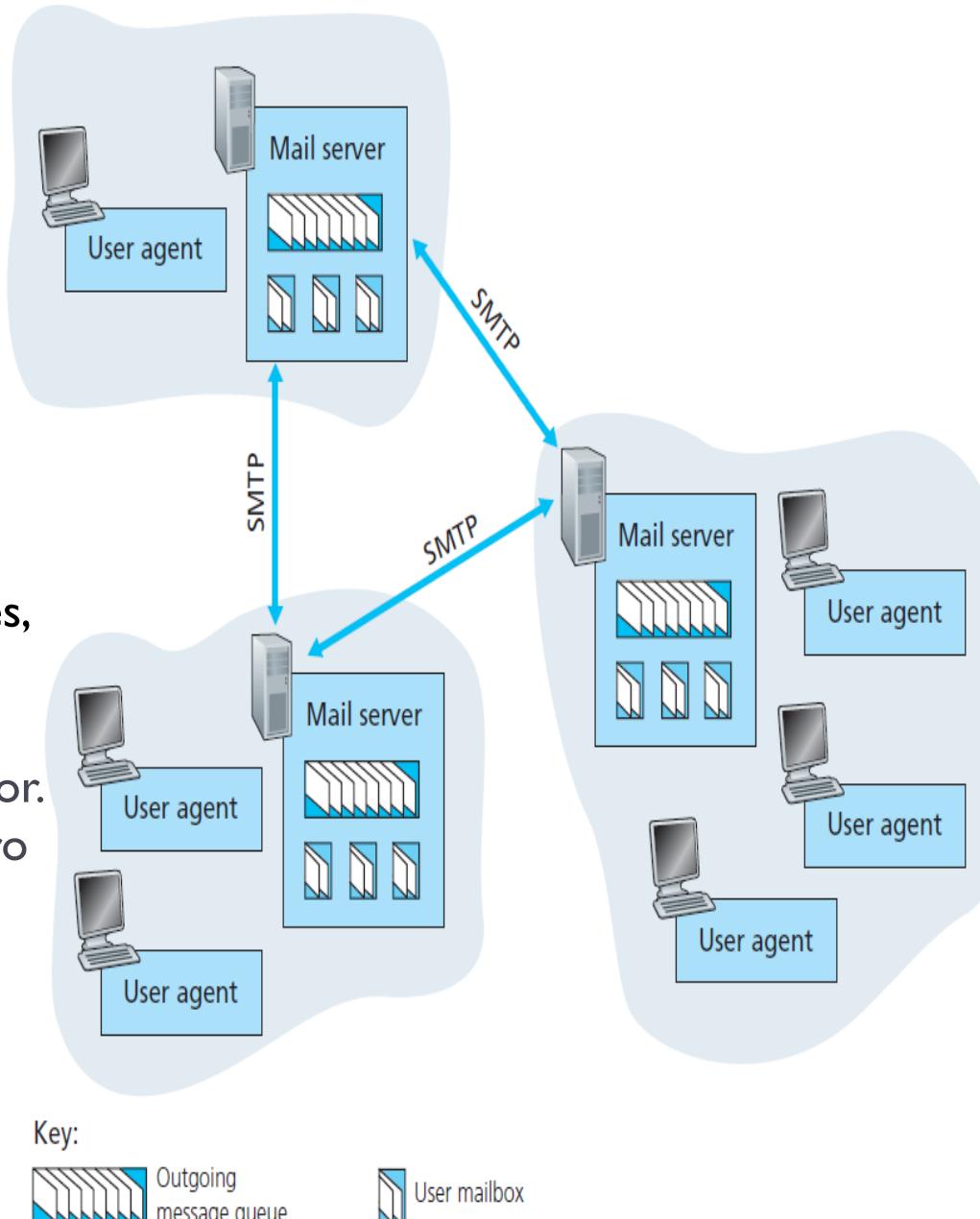


User mailbox

Correo electrónico: servidores de correo

Servidores de correo:

- ▶ **Buzón de correo:** contiene los mensajes de entrada del usuario.
- ▶ **Cola de mensajes:** mensajes de correo de salida (para ser enviados).
- ▶ **Protocolo SMTP:** entre servidores, para intercambiar mensajes de correo electrónico.
 - ▶ **Cliente:** envía correo al servidor.
 - ▶ **Servidor:** recibe correo de otro servidor.



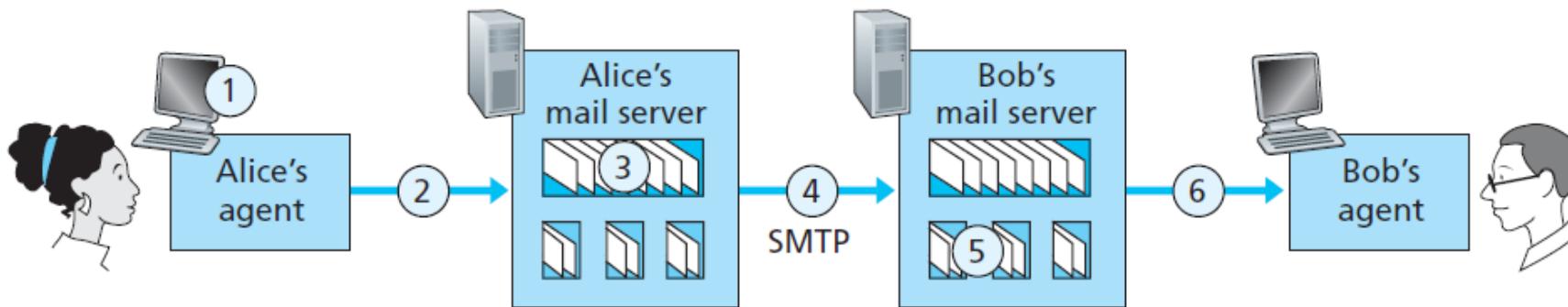
Correo electrónico: SMTP [RFC 2821]

- ▶ Utiliza TCP para transferir confiablemente el mensaje de correo electrónico **del cliente al servidor**, usando el **Puerto 25**.
- ▶ Transferencia directa: del **servidor** que envía al **servidor** que recibe.
- ▶ Las tres fases de la transferencia son:
 - ▶ “Acuerdo” (saludo).
 - ▶ Transferencia de mensajes.
 - ▶ Cierre.
- ▶ Interacción comando/respuesta:
 - ▶ Comandos: texto ASCII.
 - ▶ Respuesta: código de status y frase.
- ▶ Los mensajes deben codificarse con caracteres de siete bits en ASCII.



Ejemplo: Alicia envía un mensaje a Roberto

- 1) Alicia utiliza su agente usuario para componer el mensaje “a” bob@escuela.edu
- 2) El agente de usuario de Alicia envía un mensaje a su servidor de correo; y el mensaje es ubicado en la cola de mensajes.
- 3) El lado cliente del SMTP abre una conexión TCP con el servidor de correo de Roberto.
- 4) El cliente SMTP envía el mensaje de Alicia usando la conexión TCP.
- 5) **El servidor de correo de Roberto deposita el mensaje en el buzón de correo de Roberto.**
- 6) Roberto recurre a su agente de usuario para leer el mensaje.

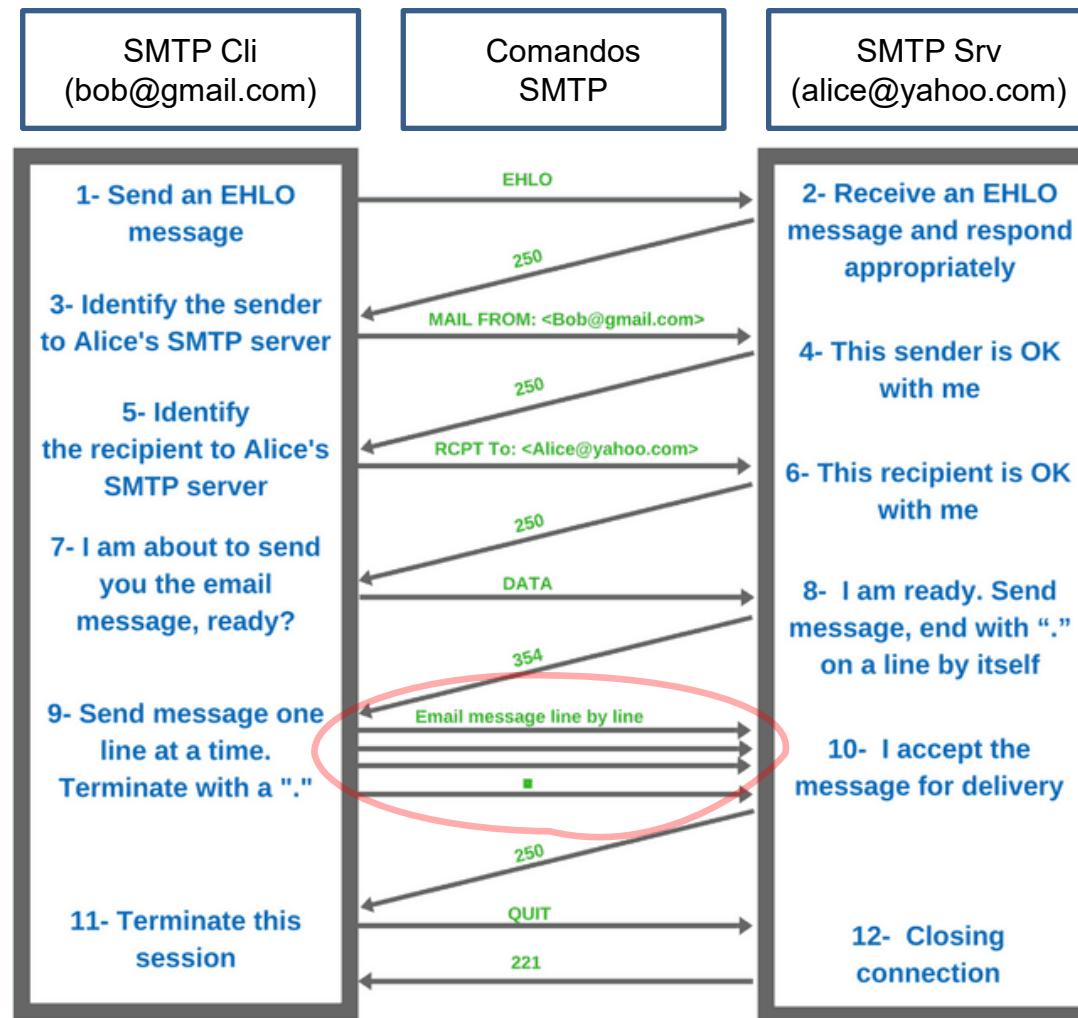


Ejemplo de interacción SMTP

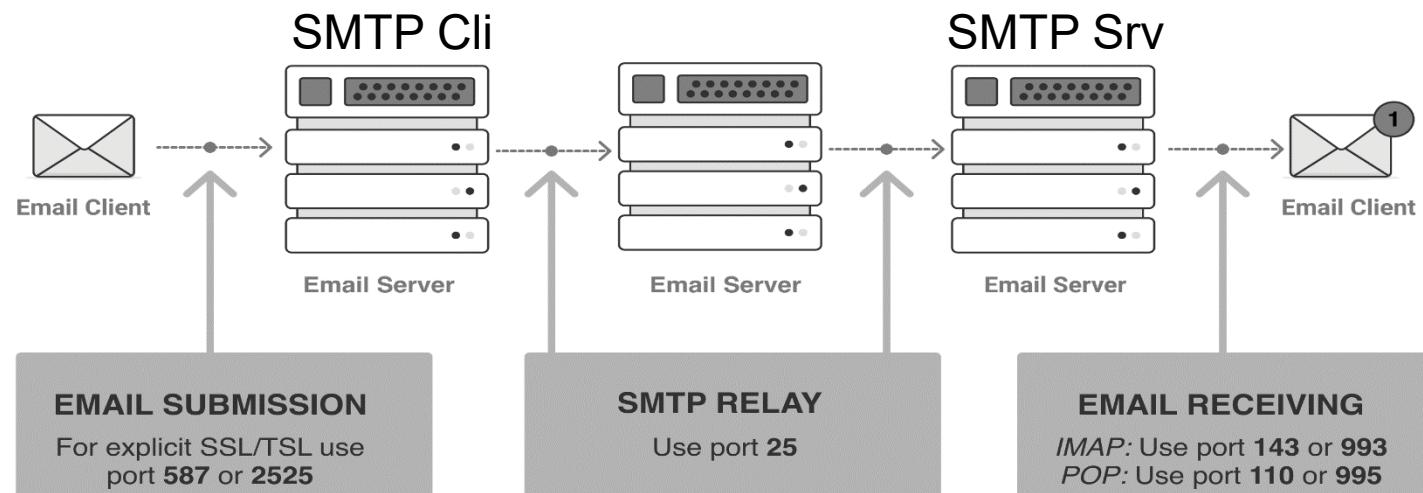
```
C: HELO crepes.fr
S: 250 Hello crepes.fr, pleased to meet you
C: MAIL FROM: <alice@crepes.fr>
S: 250 alice@crepes.fr ... Sender ok
C: RCPT TO: <bob@hamburger.edu>
S: 250 bob@hamburger.edu ... Recipient ok
C: DATA
S: 354 Enter mail, end with "." on a line by itself
C: Do you like ketchup?
C: How about pickles?
C: .
S: 250 Message accepted for delivery
C: QUIT
S: 221 hamburger.edu closing connection
```



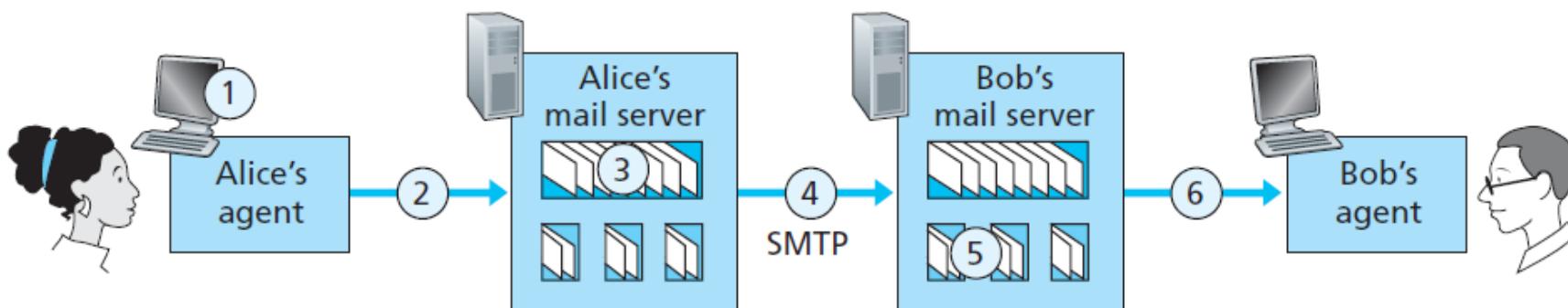
Ejemplo de interacción SMTP



Email Ports



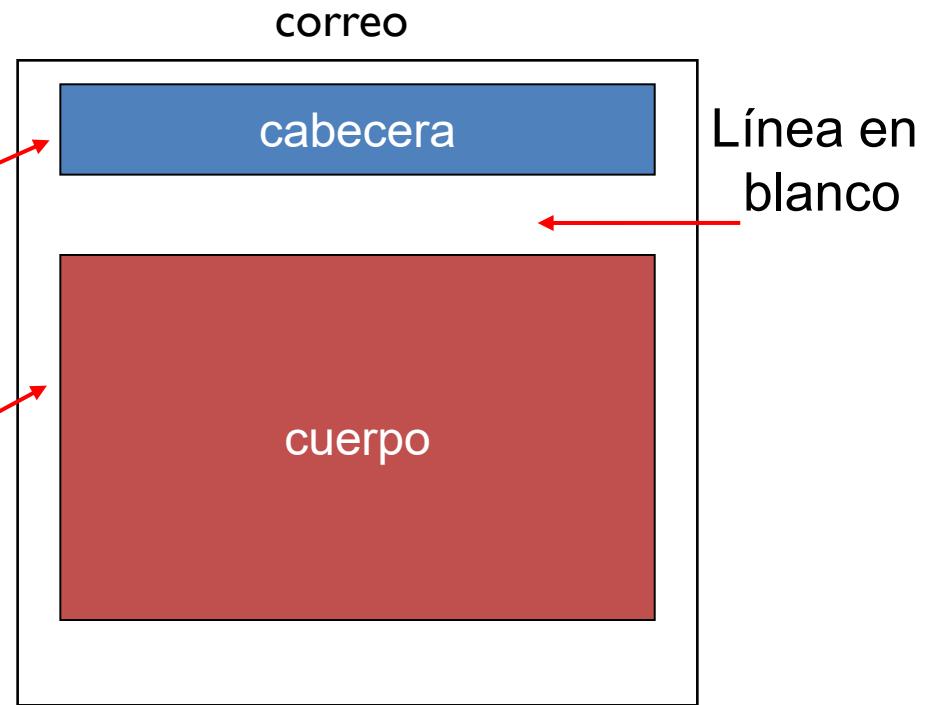
Puertos recomendados usualmente utilizados



Formato de los mensajes de correo

RFC 822: estándar para el formato de texto del mensaje:

- ▶ Cabecera:
 - ▶ Para:
 - ▶ De:
 - ▶ Asunto:
 - ▶ ...
- ▶ Cuerpo:
 - ▶ el “mensaje”, sólo caracteres ASCII.



RFC 822

Encabezado	Significado
To:	Direcciones de correo electrónico de los destinatarios primarios
Cc:	Direcciones de correo electrónico de los destinatarios secundarios
Bcc:	Direcciones de correo electrónico para las copias ocultas
From:	Persona o personas que crearon el mensaje
Sender:	Dirección de correo electrónico del remitente
Received:	Línea agregada por cada agente de transferencia en la ruta
Return-Path:	Puede usarse para identificar una ruta de regreso al remitente



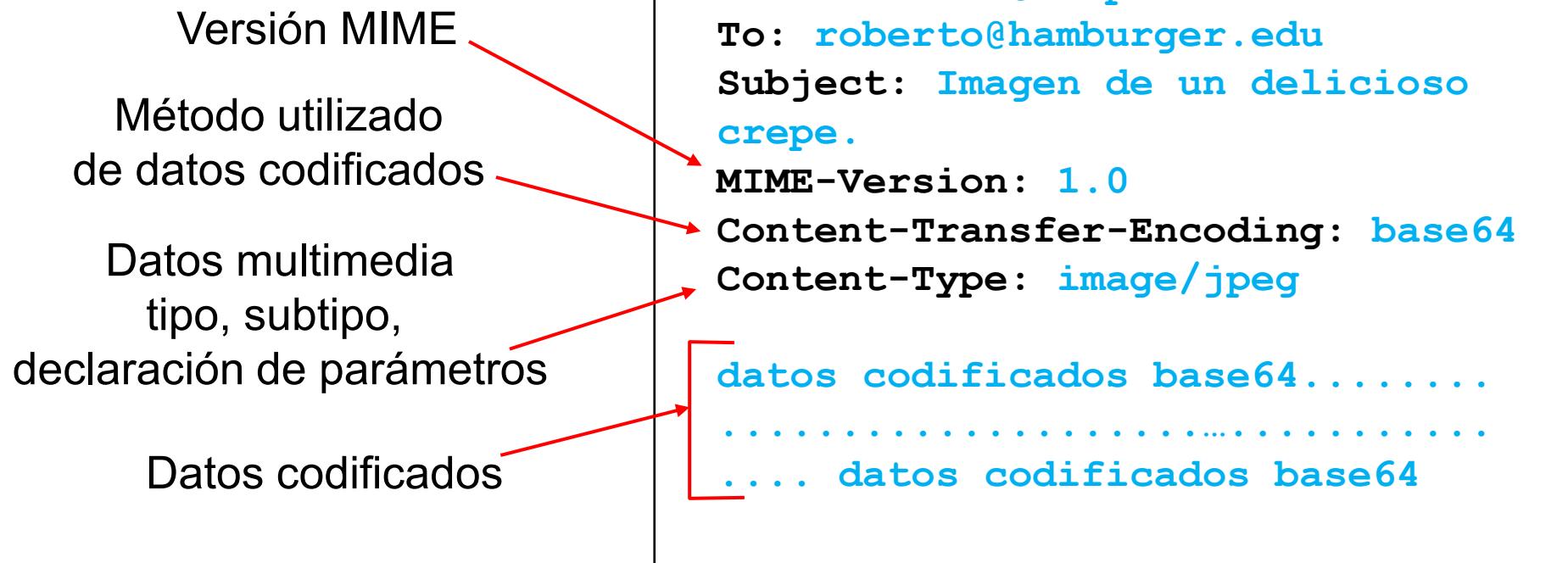
RFC 822

Encabezado	Significado
Date:	Fecha y hora de envío del mensaje
Reply-To:	Dirección de correo electrónico a la que deben enviarse las contestaciones
Message-Id:	Número único para referencia posterior a este mensaje
In-Reply-To:	Identificador del mensaje al que éste responde
References:	Otros identificadores de mensaje pertinentes
Keywords:	Claves seleccionadas por el usuario
Subject:	Resumen corto del mensaje para desplegar en una línea



MIME: Multipurpose Internet Mail Extensions

- ▶ MIME: extensiones de **correo multimedia**, RFC 2045, 2056
- ▶ Líneas adicionales en la cabecera del mensaje declaran el tipo de **contenido MIME**.



MIME: Multipurpose Internet Mail Extensions

- ▶ RFC 2045: hay siete tipos definidos, cada uno de los cuales tiene uno o más subtipos.
- ▶ El tipo y el subtipo se separan mediante una barra
 - ▶ Por ejemplo: Content-Type: video/mpeg

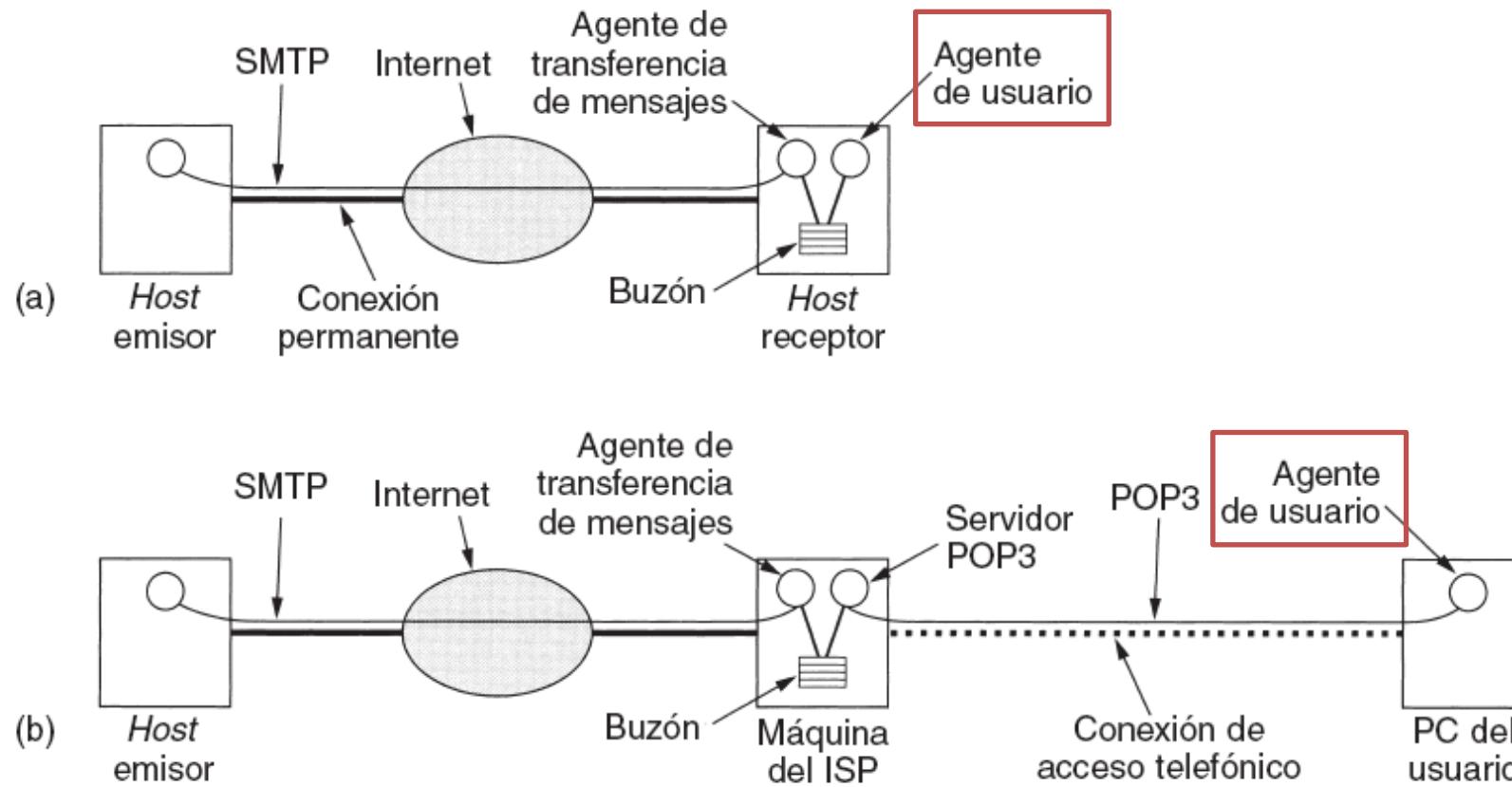
Encabezado	Significado
MIME-Version:	Identifica la versión de MIME
Content-Description:	Cadena de texto que describe el contenido
Content-Id:	Identificador único
Content-Transfer-Encoding:	Cómo se envuelve el mensaje para su transmisión
Content-Type:	Naturaleza del mensaje



MIME : RFC 2045

Tipo	Subtipo	Descripción
Texto	Plano	Texto sin formato
	Enriquecido	Texto con comandos de formato sencillos
Imagen	Gif	Imagen fija en formato GIF
	Jpeg	Imagen fija en formato JPEG
Audio	Básico	Sonido
Vídeo	Mpeg	Película en formato MPEG
Aplicación	Octet-stream	Secuencia de bytes no interpretada
	Postscript	Documento imprimible en PostScript
Mensaje	Rfc822	Mensaje MIME RFC 822
	Parcial	Mensaje dividido para su transmisión
	Externo	El mensaje mismo debe obtenerse de la red
Multipartes	Mezclado	Partes independientes en el orden especificado
	Alternativa	Mismo mensaje en diferentes formatos
	Paralelo	Las partes deben verse en forma simultánea
	Compendio	Cada parte es un mensaje RFC 822 completo

Transferencia y entrega de mensajes



POP3 - IMAP

Característica	POP3	IMAP
En dónde se define el protocolo	RFC 1939	RFC 2060
Puerto TCP utilizado	110	143
En dónde se almacena el correo electrónico	PC del usuario	Servidor
En dónde se lee el correo electrónico	Sin conexión	En línea
Tiempo de conexión requerido	Poco	Mucho
Uso de recursos del servidor	Mínimo	Amplio
Múltiples buzones	No	Sí
Quién respalda los buzones	Usuario	ISP
Bueno para los usuarios móviles	No	Sí
Control del usuario sobre la descarga	Poco	Mucho
Descargas parciales de mensajes	No	Sí
¿Es un problema el espacio en disco?	No	Con el tiempo podría serlo
Sencillo de implementar	Sí	No
Soporte amplio	Sí	En crecimiento



Nivel de Aplicación



HTTP: HyperText Transfer Protocol

La Web y HTTP

En primer lugar, un poco de jerga

- ▶ Una **página web** consta de **objetos**.
- ▶ Un **objeto** puede ser un archivo HTML, una imagen JPEG, un applet Java, un archivo de audio, un código Javascript, etc.
- ▶ Una **página web** está formada por un **archivo HTML base**, que incluye a los diversos **objetos** referenciados.
- ▶ Cada **objeto** es direccionable por una **URL**.
- ▶ Ejemplo de URL:

www.escuela.edu/departamento/imagen.gif

nombre de host

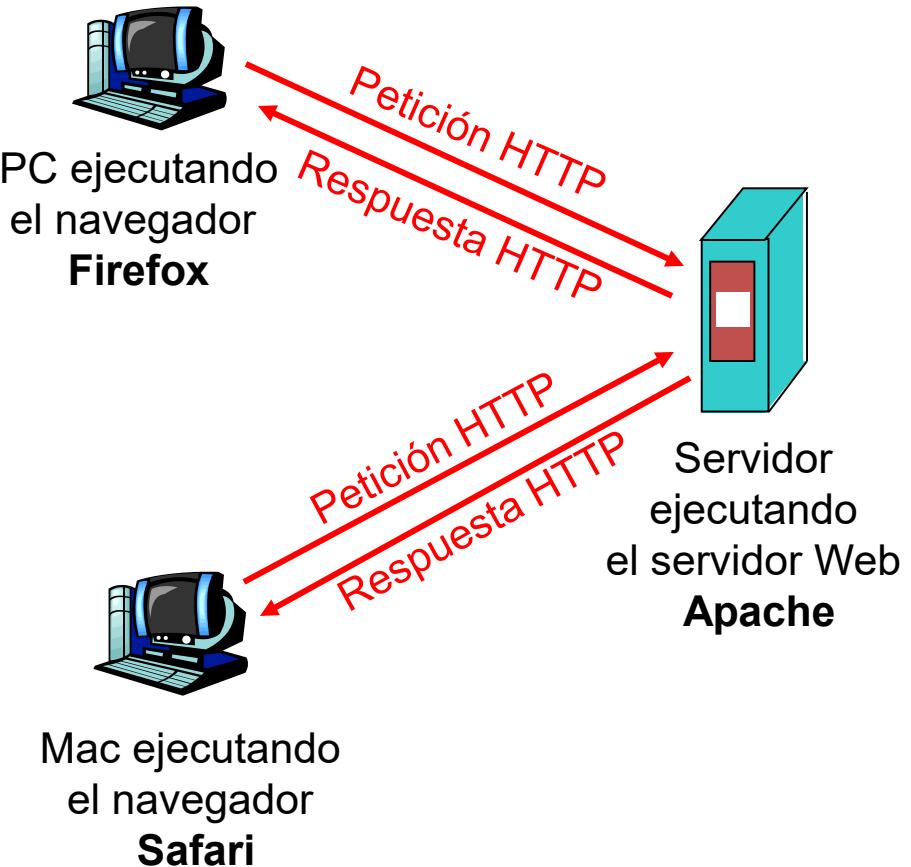
nombre de ruta



Introducción a HTTP

HTTP: protocolo de transferencia de hipertexto

- ▶ Protocolo de la **capa de aplicación de la Web**.
- ▶ Modelo cliente/servidor:
 - ▶ **Cliente**: navegador que solicita, recibe y “descarga” objetos Web.
 - ▶ **Servidor**: servidor Web que devuelve los objetos solicitados en respuesta a las peticiones.
- ▶ HTTP 1.0: RFC 1945
- ▶ HTTP 1.1: RFC 2068



Introducción a HTTP

Usos de TCP:

- ▶ El **cliente** inicia la conexión TCP (crea un socket) al servidor, puerto 80.
- ▶ El **servidor** acepta la conexión TCP del cliente.
- ▶ Los **mensajes HTTP** (mensajes de protocolo de la capa de aplicación) son intercambiados entre el navegador (cliente HTTP) y el servidor Web (servidor HTTP)
- ▶ La conexión TCP se cierra.

HTTP opera “sin estado”

- ▶ El servidor no conserva ninguna información sobre las peticiones previas de clientes.

Los protocolos que conservan “estado” son complicados

- La historia previa (estado) debe conservarse.
- Si el servidor/cliente se bloquea, sus visiones del “estado” pueden ser inconsistentes y deben ser recomuestas.



Mensaje HTTP de petición

- ▶ Hay dos tipos de mensajes HTTP: *de petición, y de respuesta.*
- ▶ Mensaje HTTP de petición:
 - ▶ ASCII (formato legible por personas)

Línea de petición
(GET, POST,
comandos HEAD)

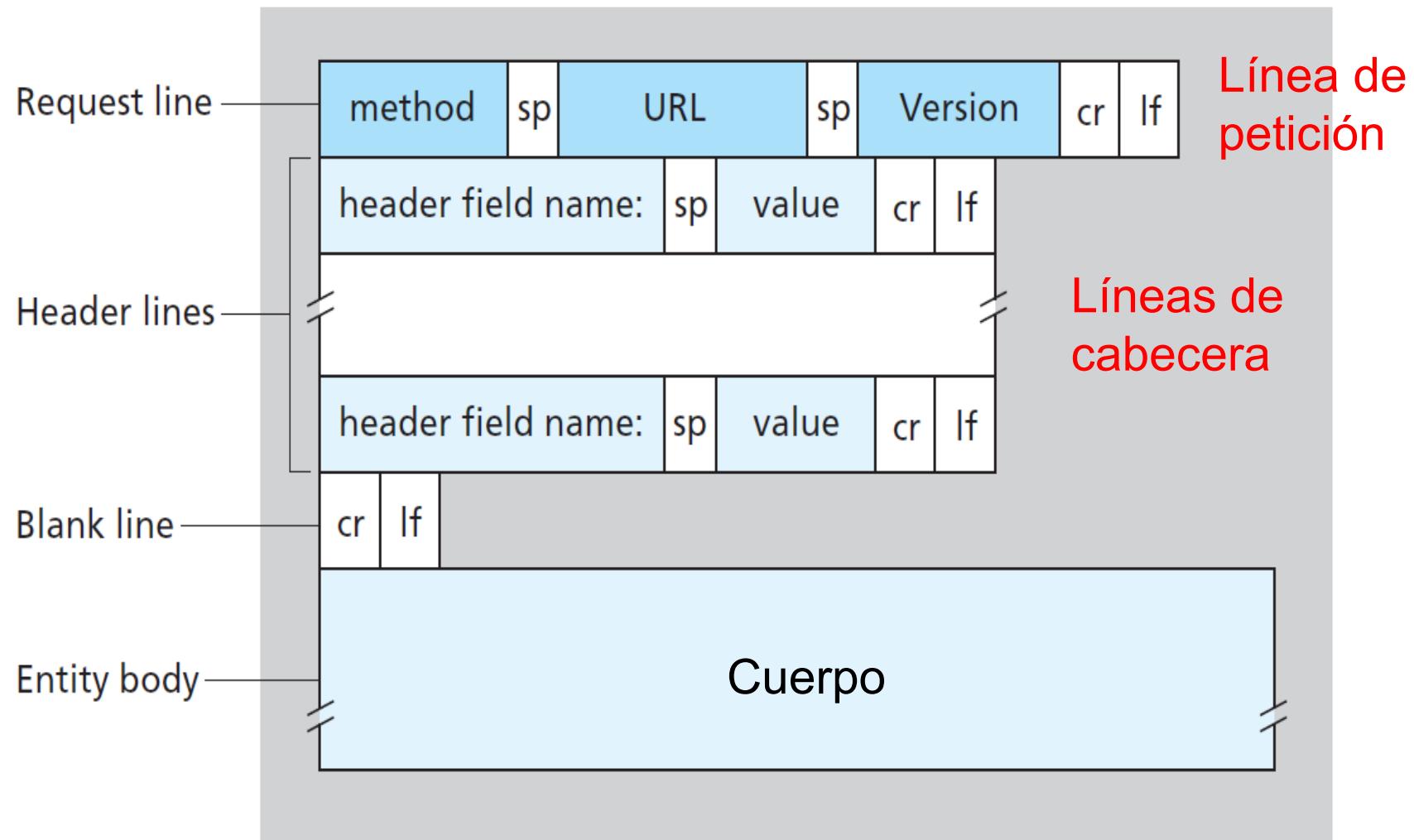
Líneas de
cabecera

Retorno de carro y
avance de línea
que indican el final
del mensaje

```
GET /dir/pagina.html HTTP/1.1
Host: www.escuela.edu
User-agent: Mozilla/4.0
Connection: close
Accept-language: fr
```

(Retorno de carro extra, avance de línea)

Mensaje HTTP de petición: formato general



Tipos de mensajes/métodos

HTTP/1.0

- ▶ GET
- ▶ POST
- ▶ HEAD
 - ▶ Pide al servidor que excluya el objeto solicitado de la respuesta. El interés está en la cabecera (metadatos) de la respuesta, para validación.

HTTP/1.1

- ▶ GET, POST, HEAD.
- ▶ PUT:
 - ▶ Descarga el archivo especificado en el cuerpo de entidad en una ruta especificada en el campo URL.
- ▶ DELETE:
 - ▶ Borra el archivo especificado en el campo URL.



Mensaje HTTP de respuesta

Línea de status
(protocolo del
código de estatus
y la frase
de estatus)

Líneas de
cabecera

Datos, por ejemplo,
el archivo
HTML solicitado

HTTP/1.1 200 OK
Connection: close
Date: Thu, 06 Aug 1998 12:00:15 GMT
Server: Apache/1.3.0 (Unix)
Last-Modified: Mon, 22 Jun 1998 ...
Content-Length: 6821
Content-Type: text/html

datos datos datos datos datos ...



Código de status HTTP de respuesta

En la primera línea en el mensaje de respuesta servidor -> cliente.

Algunos ejemplos de códigos:

200 OK

- ▶ petición exitosa, el objeto requerido aparece posteriormente en este mensaje.

301 Moved Permanently

- ▶ el objeto demandado ha sido movido, su nueva localización se especifica posteriormente en este mensaje (Location:).

400 Bad Request

- ▶ el servidor no comprendió el mensaje de petición.

404 Not Found

- ▶ el documento pedido no existe en este servidor.

505 HTTP Version Not Supported



Ponga a prueba el HTTP (como cliente) usted mismo

I. Telnet a su servidor web favorito:

```
telnet www.eurecom.fr 80
```

Abre la conexión TCP al puerto 80.
(puerto por defecto servidor HTTP) en
www.eurecom.fr

Cualquier cosa que se escriba es enviada a
www.eurecom.fr

2. Escriba una petición HTTP tipo GET:

```
GET /~ross/index.html HTTP/1.0
```

Escribiendo esto (con doble retorno
de carro), está enviando esta petición
GET mínima (pero completa) al servidor
HTTP.

3. Mire el mensaje de respuesta enviado por el servidor HTTP.

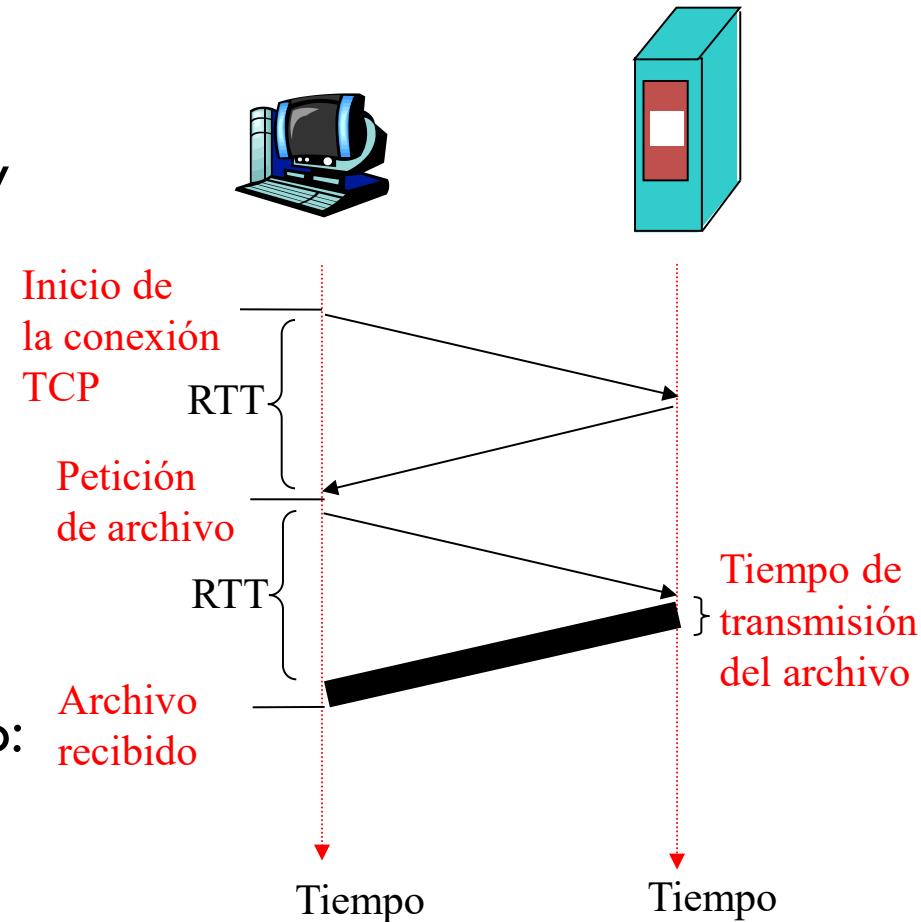


Modelo del tiempo de respuesta

Definición de RTT: tiempo necesario para enviar un paquete pequeño desde el cliente hasta el servidor y después de vuelta al cliente.

Tiempo de respuesta:

- ▶ Un RTT para iniciar la conexión TCP.
- ▶ Un RTT para la petición HTTP y los primeros bytes de respuesta HTTP de vuelta.
- ▶ Tiempo de transmisión del archivo:
 $\text{Total} = 2 \times \text{RTT} + \text{tiempo de transmisión}$



Conexiones HTTP

Conexiones HTTP no persistentes

- ▶ Se envía un objeto como máximo con una conexión TCP.
- ▶ HTTP/1.0 utiliza conexiones HTTP no persistentes.

Conexiones HTTP persistentes

- ▶ Se pueden enviar múltiples objetos con una sola conexión TCP entre el cliente y el servidor.
- ▶ HTTP/1.1 utiliza conexiones persistentes en su modo por defecto.



Conexiones HTTP no persistentes

Supongamos que el usuario entra en el URL:

www.escuela.edu/departamento/home.index

(consta de texto
y referencias a 10
imágenes .jpeg)

Ia. El cliente HTTP inicia la conexión TCP con el servidor HTTP (proceso) de www.escuela.edu en el puerto 80.

2. El cliente HTTP envía un *mensaje HTTP de petición* (que contiene el URL) a través del socket de la conexión TCP.
El mensaje indica que el cliente quiere el objeto departamento/home.index.

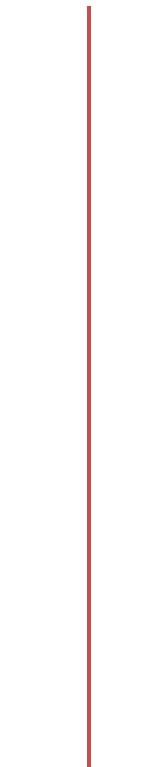
Ib. El servidor HTTP en el host www.escuela.edu espera la conexión TCP del puerto 80 y “acepta” la conexión, notificando al cliente.

3. El servidor HTTP recibe el mensaje de petición, compone un *mensaje de respuesta* que contiene el objeto solicitado, y lo envía al socket.

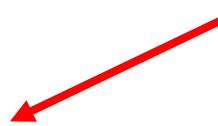
Tiempo



Conexiones HTTP no persistentes



4. El servidor HTTP cierra la conexión TCP.



5. El cliente HTTP recibe el mensaje de respuesta que contiene el archivo html y descarga el html. Analizando el archivo html, se encuentran referenciados 10 objetos jpeg.

6. Se repiten los pasos del 1 al 5 para cada uno de los 10 objetos jpeg.



Conexiones HTTP persistentes

Particularidades de

HTTP no persistente:

- ▶ Requieren dos RTT por objeto.
- ▶ El sistema operativo debe asignar los recursos del host para cada conexión TCP.
- ▶ Sin embargo, los navegadores suelen abrir **conexiones TCP paralelas** para traer los objetos referenciados.

HTTP persistente:

- ▶ El servidor deja la conexión TCP abierta tras enviar la respuesta.
- ▶ Los mensajes HTTP posteriores entre el mismo cliente/servidor se envían por la misma conexión.

Conexiones persistentes sin

entubamiento:

- ▶ El cliente sólo emite una nueva petición una vez que ha recibido la anterior respuesta.
- ▶ Un RTT por cada objeto referenciado.

Conexiones persistentes con

entubamiento:

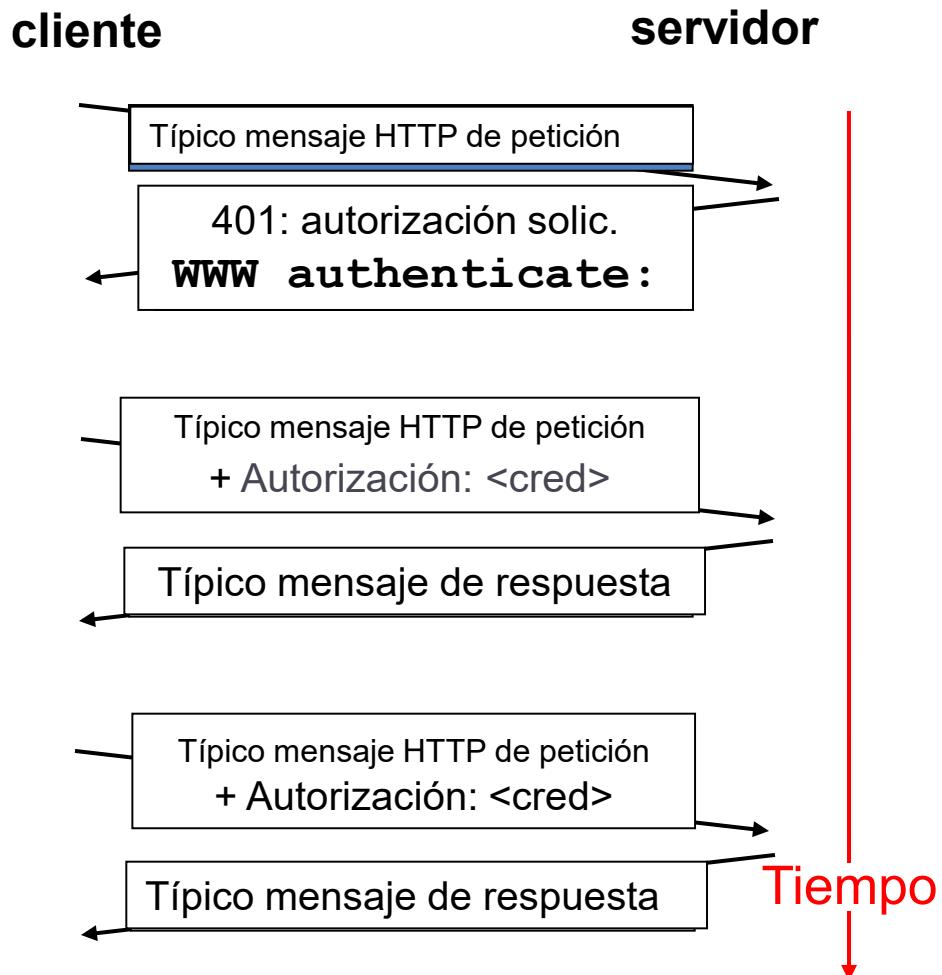
- ▶ Por defecto en HTTP/1.1
- ▶ El cliente hace su petición **tan pronto como** encuentra un objeto referenciado.
- ▶ Tan sólo un RTT para todos los objetos referenciados.



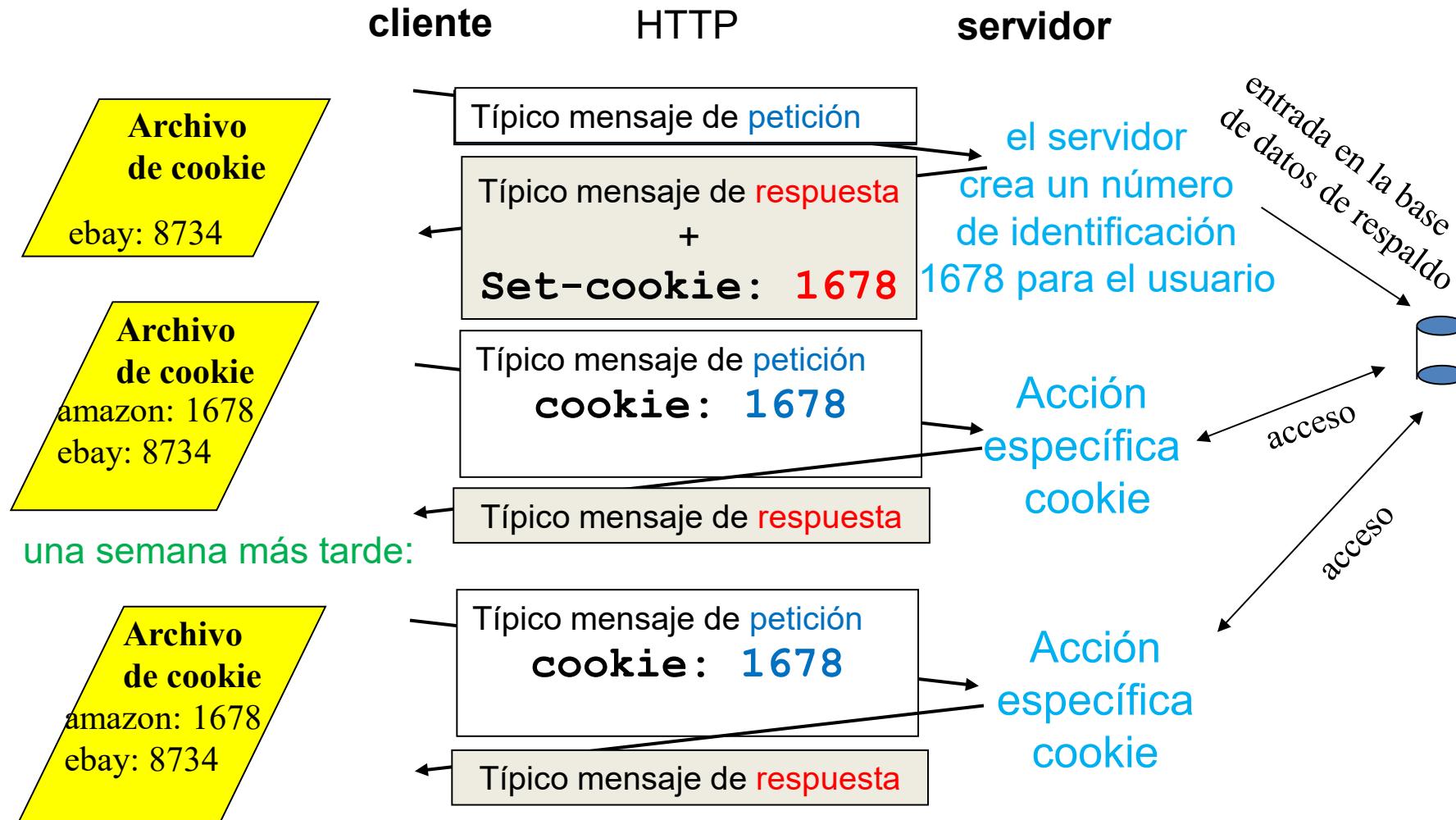
Interacción usuario-servidor: autorización

Autorización: control al acceso del contenido del servidor.

- ▶ Credenciales de autorización, normalmente son: *nombre, contraseña*.
- ▶ **Sin estado:** el cliente debe presentar la autorización para *cada* petición:
 - ▶ **Autorización:** línea de cabecera en cada petición.
 - ▶ Si no hay **autorización:** cabecera, el servidor rechaza el acceso y envía **WWW authenticate:** línea de cabecera en respuesta.



Cookies: mantenimiento del “estado”



Cookies: mantenimiento del “estado”

Muchos sitios web importantes utilizan “cookies”.

Cuatro componentes:

- 1) Línea de cabecera de cookie en el mensaje HTTP de **respuesta**.
- 2) Línea de cabecera de cookie en el mensaje HTTP de **petición**.
- 3) **Archivo de cookie** que se almacena en el host del usuario y que es gestionado por el navegador del usuario.
- 4) **Base de datos** de respaldo en el sitio Web.

Ejemplo:

- ▶ Susana siempre accede a Internet desde el mismo PC.
- ▶ Visita un sitio de comercio electrónico por primera vez.
- ▶ Cuando la petición HTTP inicial llega al sitio, éste crea un número de identificación único y también crea en su base de datos una entrada indexada por el número de identificación.



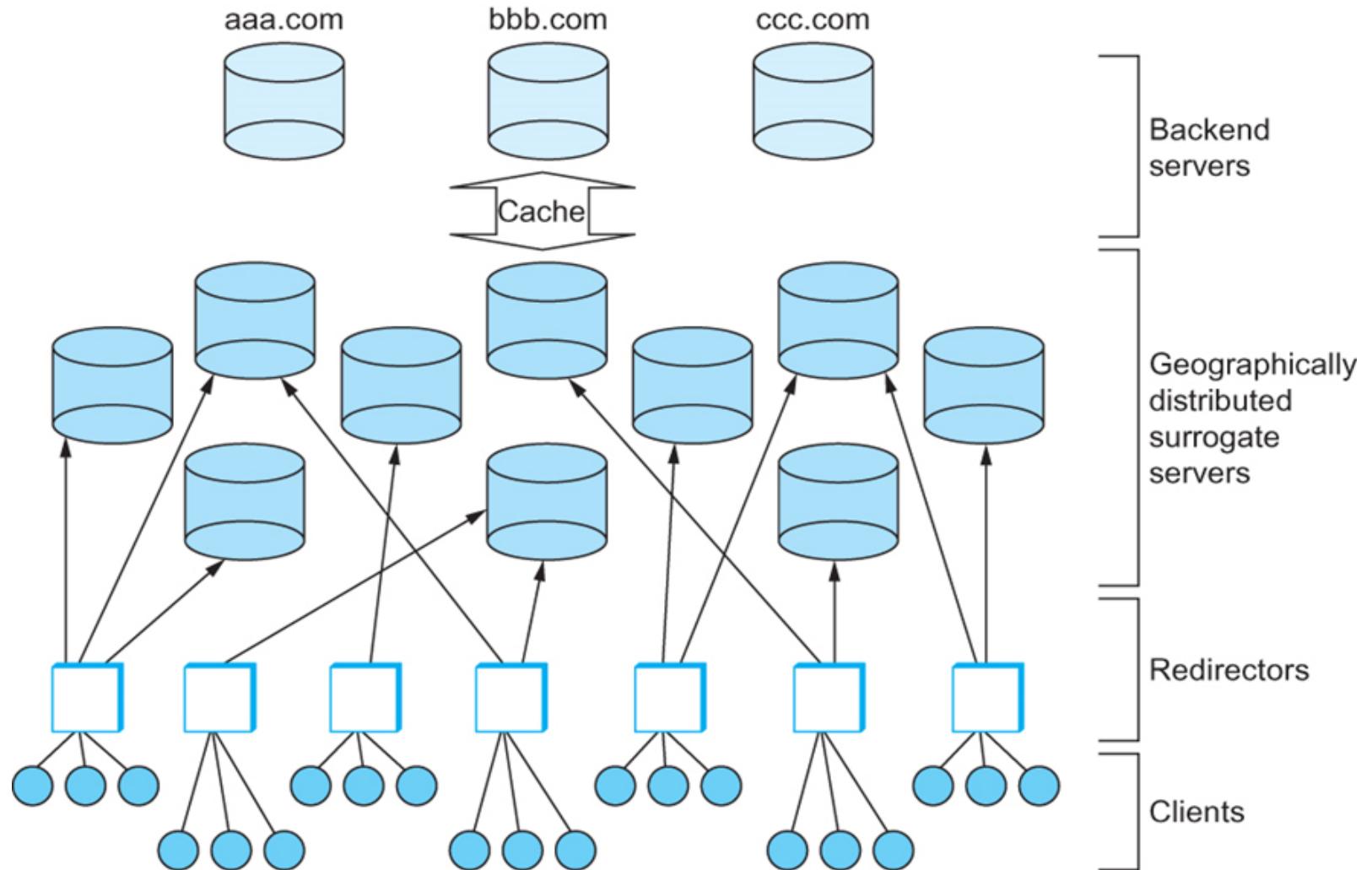


CDN



Content Distribution Networks

Modelo genérico de una CDN



CDN

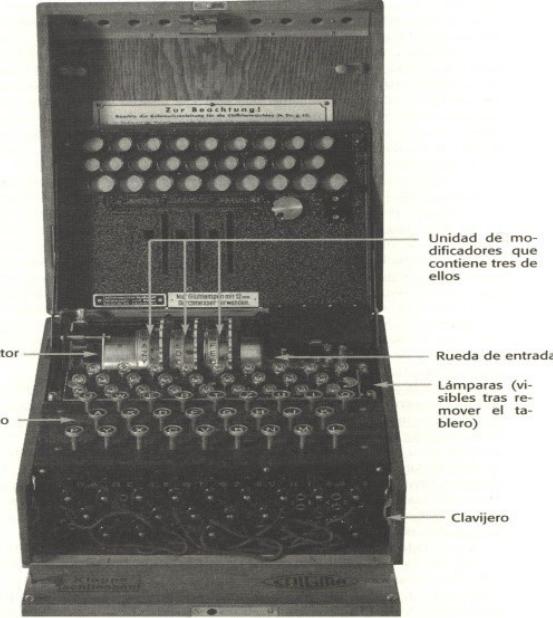
¹⁰Las CDNs, como su nombre lo indica, son redes de distribución de contenido. Funcionan como nexo entre los proveedores, por ejemplo los sitios Web de canales de televisión, y los consumidores en los ISPs. Colocando servidores propios en los ISPs, servidores que cachean o almacenan enteramente el contenido, las CDNs se aseguran de que los clientes lo accedan de manera local. Esto es especialmente importante para la distribución de video, en la cual una latencia razonable es importante para que el video no se vea entrecortado.

La distribución de usuarios a servidores se realiza mediante redirecciones DNS. Cuando un usuario solicita contenido almacenado en una CDN, la dirección IP que recibe para una URL dada se resuelve teniendo en cuenta su localización geográfica y en la red. De esta manera, accederá a la copia del contenido presente en el servidor más cercano (geográficamente, o en términos de latencia) a él.

Bibliografía Básica

- ▶ Computer Networks, Fifth Edition:A Systems Approach (The Morgan Kaufmann Series in Networking) [Larry L. Peterson , Bruce S. Davie 2011](#)
- ▶ Capítulo 9 Nivel de Aplicación: Mail - Web
págs. 700-718 y DNS págs. 745-755
 - ▶ Se explican arquitecturas, fundamentos en pizzarrón + slides de la clase “práctica”
- ▶ Capítulo 9 Nivel de Aplicación: Overlays Networks págs. 759-789



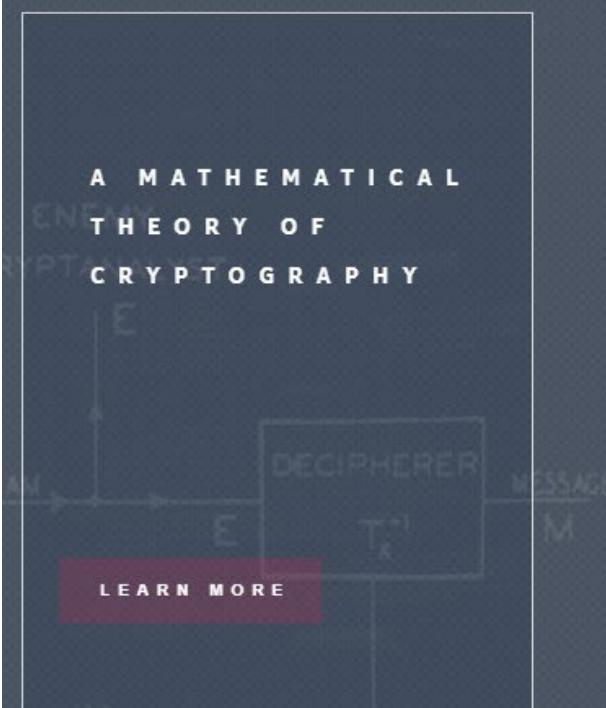


Seguridad en Redes

Fundamentos

<http://www.moonmentum.com/blog/tag/alan-mathison-turing>

Una teoría de la criptografía



- Provided an analysis of possible “unbreakable cryptographical methods”.
- Determined which secrecy methods would be ideal or most practical if an unbreakable system could not be implemented.
- Provided a proof that all theoretically unbreakable ciphers must have the same requirements as the one-time pad - a foundational treatment of modern cryptography.
- Made astute observation that language, particularly English, was redundant and predictable.
- Proved that in a point-to-point error-free link where an eavesdropper observes the transmitted message with no error, with enough common randomness between the transmitter and the receiver, a perfectly secure communication link is achievable.

Reporte interno Bell Labs:

<https://www.iacr.org/museum/shannon/shannon45.pdf>

Paper público:

C. E. Shannon, "Communication theory of secrecy systems" in *The Bell System Technical Journal*, vol. 28, no. 4, pp. 656-715, Oct. 1949

Una teoría de la criptografía

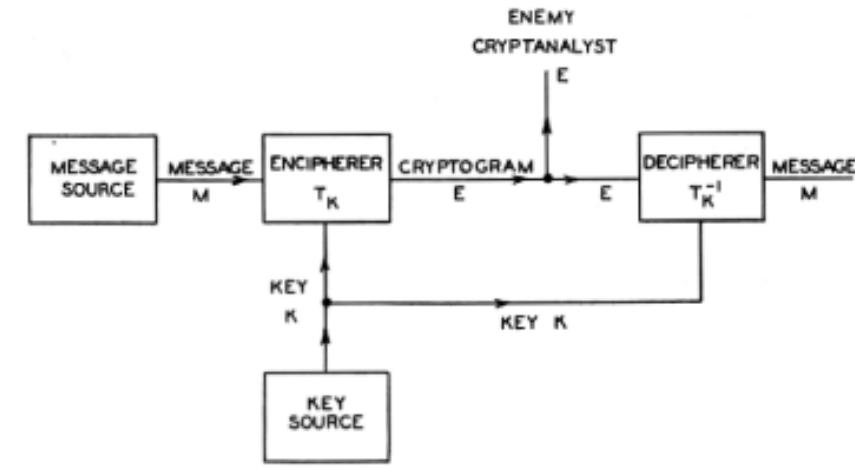
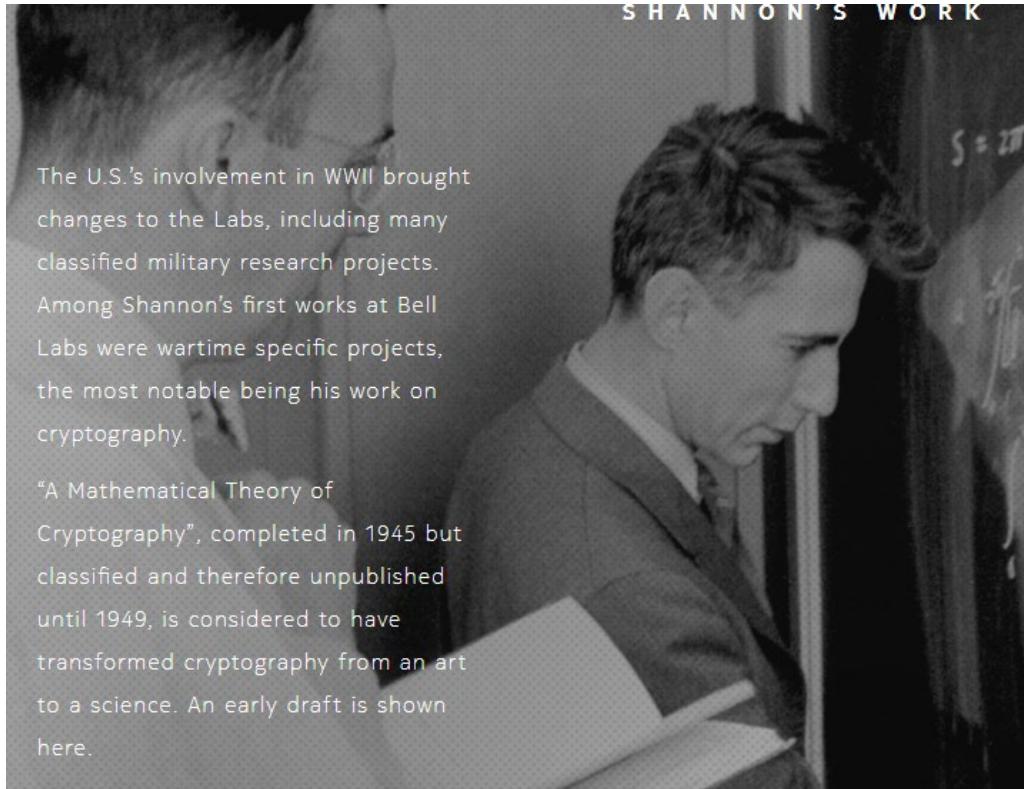


Fig. 1—Schematic of a general secrecy system.

Agenda

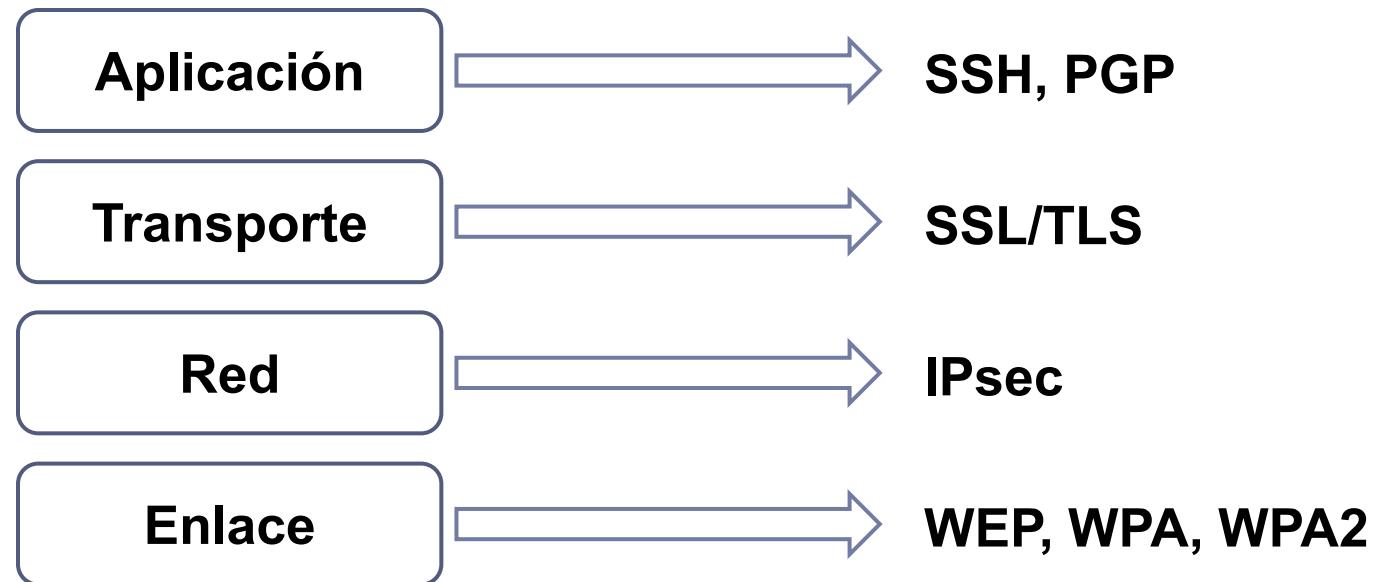
- ▶ Marco de Trabajo
- ▶ Introducción a la Criptografía
- ▶ Algoritmos de Simétricos y de Clave Pública
- ▶ Firma Digital – Integridad

Marco de Trabajo - Conceptos

- ▶ **Confidencialidad:** Los mensajes sólo deben poder **ser entendidos** por las partes especificadas en la comunicación.
- ▶ **Integridad:** Los mensajes enviados no pueden ser modificados durante su transmisión.
 - ▶ Originalidad: El mensaje **no es una copia artificial** repetida
 - ▶ Temporalidad: El mensaje **no fue demorado** maliciosamente
- ▶ **Autenticación:** Ninguna parte puede asumir en forma no autorizada **la identidad** de otra parte.
- ▶ **Control de acceso (Autorización):** Solo ciertos usuarios remotos pueden realizar ciertas acciones permitidas
- ▶ **Disponibilidad:** Todo usuario potencial tendrá su oportunidad de ser considerado y eventualmente admitido (ataque DoS)
- ▶ **No Repudiación:** Ninguna de las partes puede **negar haber participado** en una transacción.

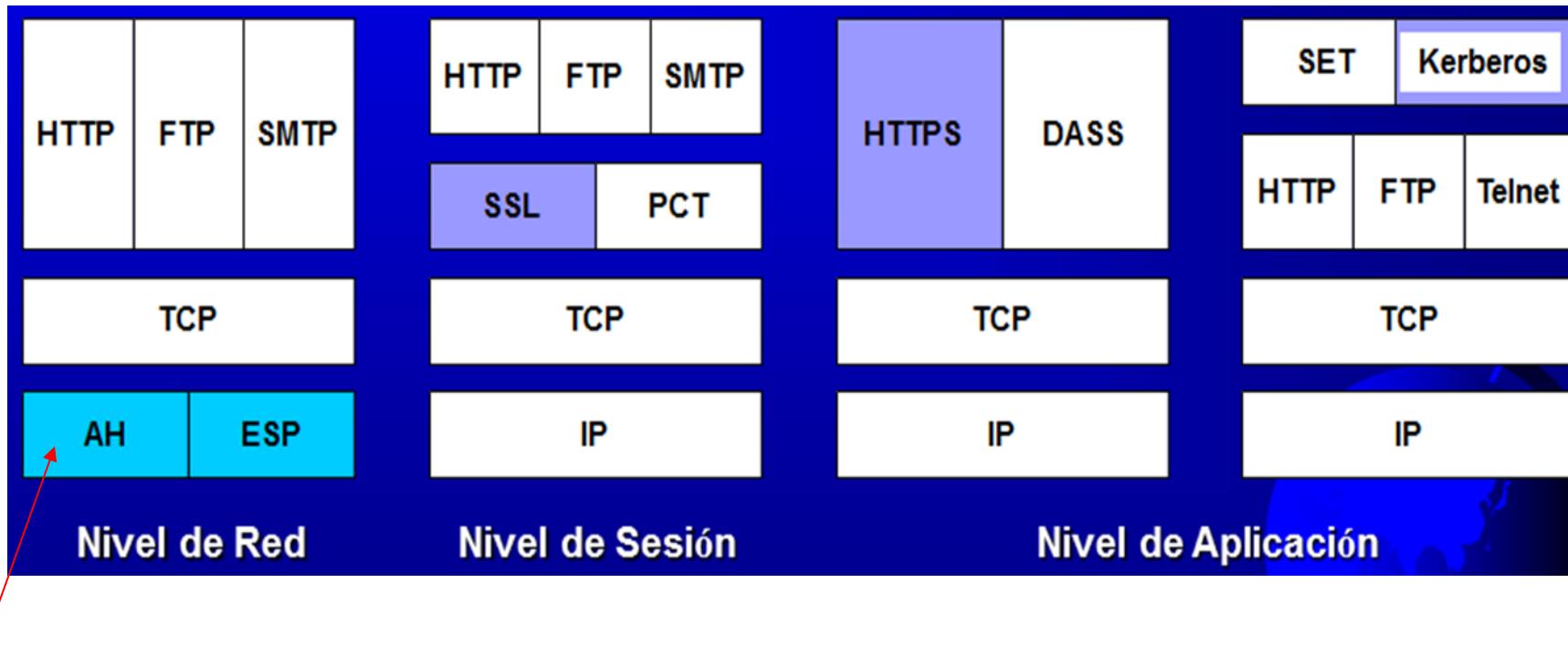
Ante cada tecnología de Seguridad Informática debemos preguntarnos: **¿Cuál subconjunto de propiedades provee?**

Protocolos y Capas



- ▶ La seguridad se implementa para resolver diferentes potenciales **problemas en diferentes capas** de la arquitectura de una red.

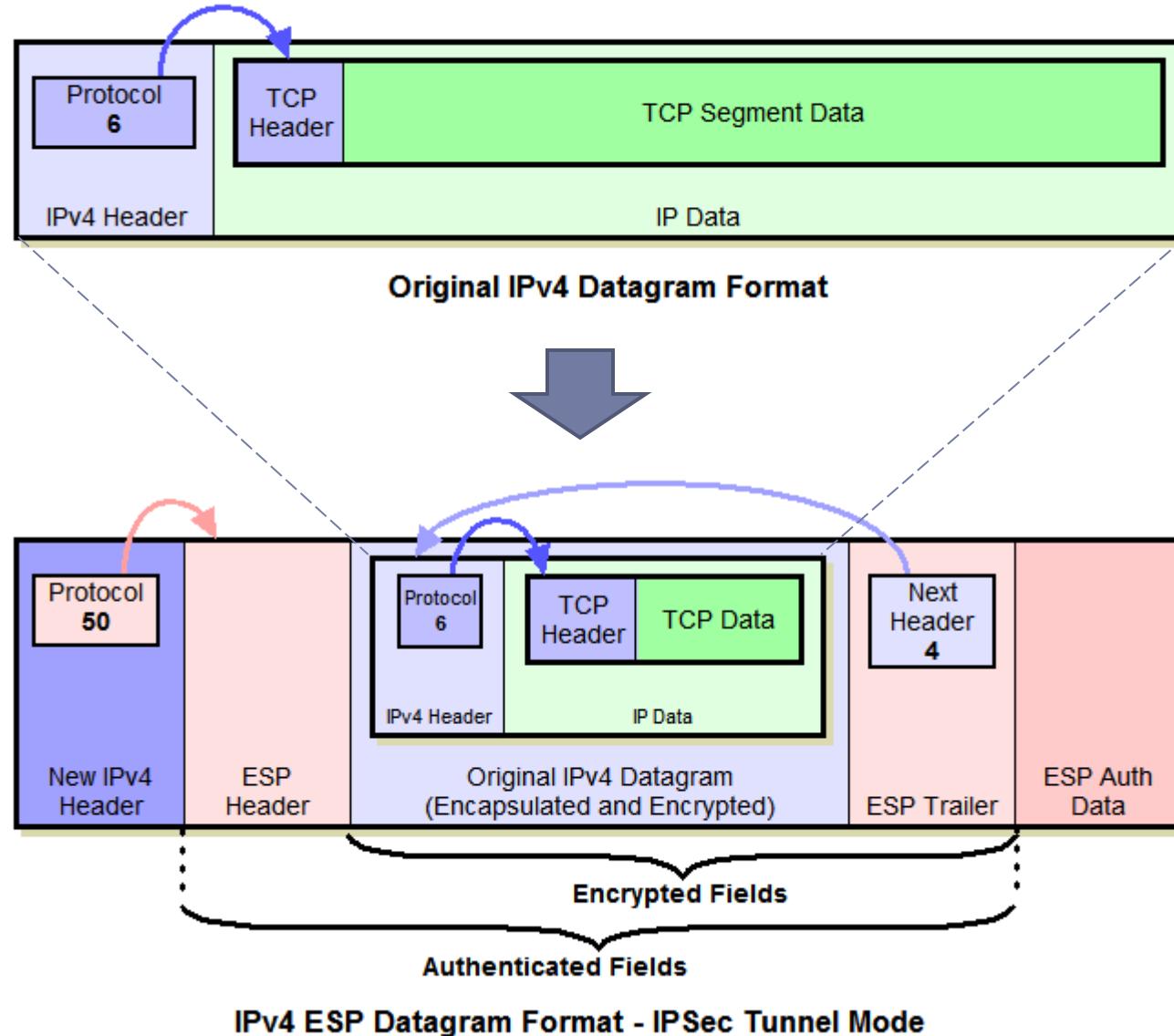
Seguridad en las Capas



Ejemplo: **Conjunto IPsec** de protocolos

- Para la **Capa de Red**
- **Authentication Header (AH)** proporciona integridad, autenticación y no repudio.
- **Encapsulating Security Payload (ESP)** proporciona confidencialidad

IPSec (AH y ESP)



Introducción a la Criptografía

κρύπτω *krypto*, «**oculto**» y γράφως *graphos*, «**escribir**»
literalmente «escritura oculta»

Un recorrido rápido

- ▶ Generación de números al azar
- ▶ Hashing
- ▶ Claves Simétricas
- ▶ Claves Asimétricas

Definiciones

▶ Cifrado

- ▶ Transformación **carácter por carácter, o bit por bit**, sin importar la estructura lingüística del mensaje

▶ Código

- ▶ Reemplaza **una palabra con otra** palabra o símbolo

- ▶ Ej.: código basado en el idioma indígena Navajo, usado por los EEUU en el Pacífico durante la II Guerra Mundial

<https://youtu.be/loDvpds64gk?t=23>

<https://youtu.be/-iTdQGjj3eI?t=33>

- ▶ Un indio Navajo a ambos lados de la línea. Su idioma no tiene reglas, no es muy conocido, no es escrito.



Conceptos

- ▶ Los mensajes a ser encriptados se denominan Texto Plano o “**plaintext**”.
- ▶ Son tomados como una **entrada** y son transformados por una **función** (algoritmo) parametrizada por una **clave**.
- ▶ La **salida** del **proceso de encripción (cifrado)** es conocida como “**ciphertext**”, o texto encriptado, y constituye los datos a transmitir.

- ▶ Se asume que un *intruso* puede escuchar y copiar el ciphertext completo del canal de comunicación (**intruso pasivo**).
- ▶ La técnica de descifrar mensajes es llamada *criptoanálisis*.
- ▶ Las técnicas de **crear ciphers (criptografía)** y de **decifrarlos (criptoanálisis)** forman la *criptología*.

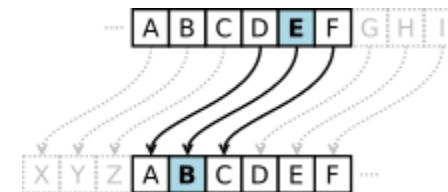
Métodos Básicos de Cifrado

- ▶ **Sustitución:** cada letra o grupo de letras se reemplazan por otra letra o grupo de letras para enmascarar.

Se preserva el **orden** del *plaintext*.

- ▶ Ejemplo:

- Cifrado del César (Julius Caesar)
Rotación del alfabeto una cantidad fija
 - Sustitución mono alfabética
(cada letra se mapea con otra cualquiera; $26! = 4 \times 10^{26}$ llaves posibles)

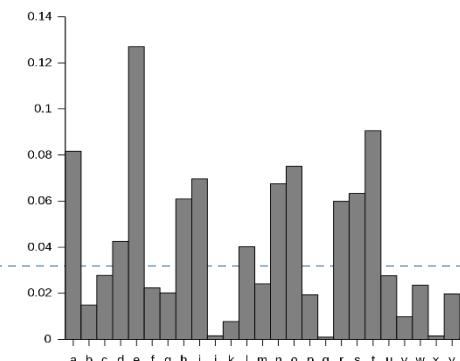


- ▶ Ejemplo:

texto plano: a b c d e f g h i j k l m n o p q r s t u v w x y z

texto cifrado: Q W E R T Y U I O P A S D F G H J K L Z X C V B N M

- ▶ Es fácil quebrarlo?
- ▶ Propiedades estadísticas
de los lenguajes naturales?



Sustitución
Polialfabética
(1 alfabeto diferente por
cada letra del mensaje)

Métodos Básicos de Cifrado

- ▶ **Transposición:** Se reordenan las letras pero no se “enmascaran”
- ▶ La clave de este cifrado es una palabra que no contiene letras repetidas. Por ejemplo: MEGABUCK.
- ▶ El propósito de la clave es numerar las columnas, estando la columna número 1 bajo la letra clave más cercana al inicio del alfabeto, y así sucesivamente.
 - ▶ El texto plano se escribe horizontalmente, en filas, las cuales se llenan para completar la matriz si es necesario.
 - ▶ El texto cifrado se lee por columnas, comenzando por la columna cuya letra clave es la más baja.

<u>M</u>	<u>E</u>	<u>G</u>	<u>A</u>	<u>B</u>	<u>U</u>	<u>C</u>	<u>K</u>
<u>7</u>	<u>4</u>	<u>5</u>	<u>1</u>	<u>2</u>	<u>8</u>	<u>3</u>	<u>6</u>
p	l	e	a	s	e	t	r
a	n	s	f	e	r	o	n
e	m	i	l	l	i	o	n
d	o	l	l	a	r	s	t
o	m	y	s	w	i	s	s
b	a	n	k	a	c	c	o
u	n	t	s	i	x	t	w
o	t	w	o	a	b	c	d

Plaintext

please transfer one million dollars to
my swiss bank account six two two

Ciphertext

AFLLSKSOSELAWAIATOOSCTCLNMOMANT
ESILYNTWRNNTSOWDPAEDOBUEIRICXB

Principio de Kerckhoffs

“Todos los *algoritmos* deben ser *públicos*,
sólo las *claves* deben ser *secretas*”

KERCKHOFF,A.:“La Cryptographie Militaire,” J. des Sciences Militaires, vol. 9, pp.5-38, Jan. 1883 and pp. 161-191

- Idea: tratar de mantener secreto **el algoritmo** nunca funciona (es decir, la “seguridad por desconocimiento”)
- Es el principio usado por la mayoría de los sistemas prácticos de criptografía moderna.

Más información y material en petitcolas.net/kerckhoffs

Longitud de la clave, Factor de Trabajo

- ▶ Si el **secreto reside en la clave**, entonces su **longitud** es un factor de diseño crítico.
- ▶ Clave de 2 dígitos decimales → 100 combinaciones.
 - ▶ 6 dígitos → 1 millón de combinaciones
- ▶ A mayor longitud de clave, mayor “Factor de Trabajo” o *tiempo requerido del criptoanalista para romper la clave*.
 - ▶ El **Factor de Trabajo crece exponencialmente** con el tamaño de la clave → El **“secreto” se logra con un algoritmo robusto y una clave suficientemente larga para que el tiempo sea impráctico**.
- ▶ Cuanto es “suficiente” para el tamaño de clave?
 - ▶ 64 bits previene lectura de mails por parte de hermanos menores
 - ▶ Claves de 128/256 bits uso comercial rutinario
 - ▶ >= 256 bits para sistemas críticos (ej. Defensa)

One-Time Pads (“rellenos de una sola vez”)

Cifrado Inviolable.

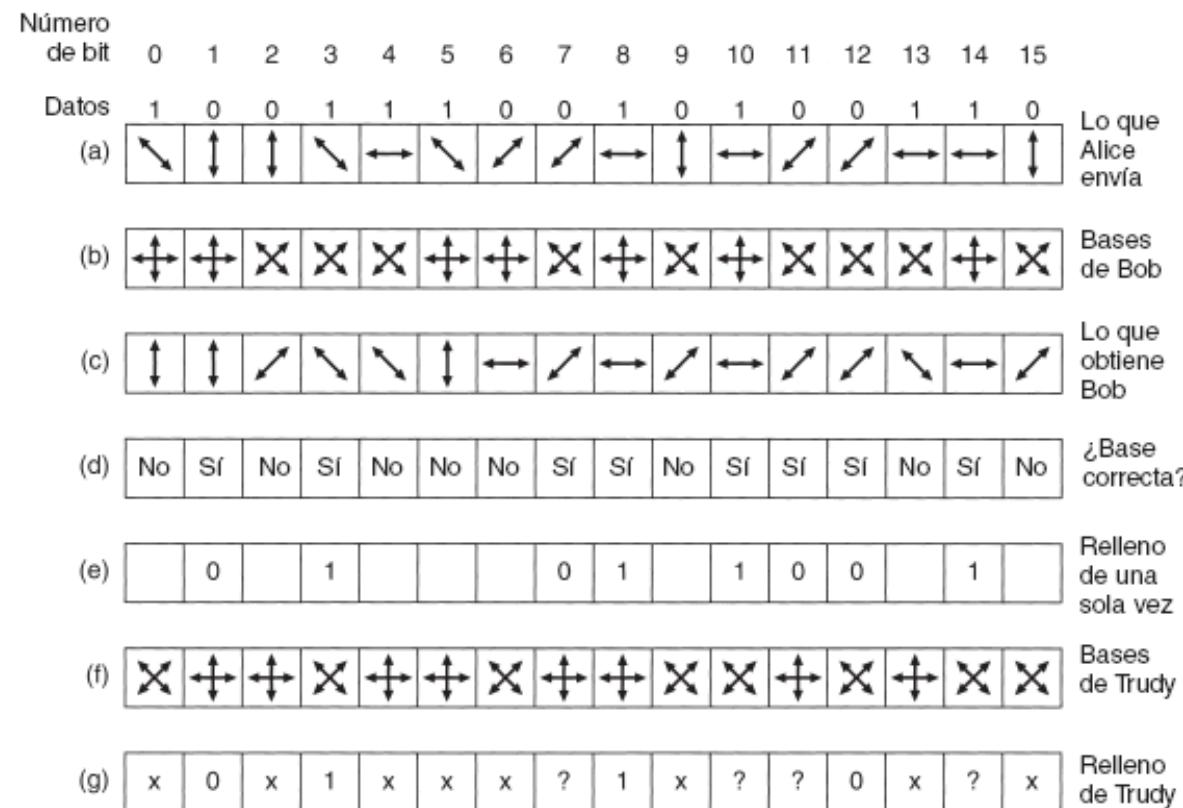
- Elegir un gran **random bit string** como clave (= longitud del texto?)
- Usar Bit XOR para encriptar y desencriptar.
- “*inmune a todos los ataques actuales y futuros sin importar cuánta potencia computacional tenga el intruso*“
(resultado basado en Teoría de la Información, probado por Claude Shannon en 1945/1949)
- Problema: como distribuir y proteger la clave? **Poco práctico.**

Mensaje 1: 1001001 0100000 1101100 1101111 1110110 1100101 0100000 1111001 1101111 1110101 0101110

Relleno 1: 1010010 1001011 1110010 1010101 1010010 1100011 0001011 0101010 1010111 1100110 0101011

Texto cifrado: 0011011 1101011 0011110 0111010 0100100 0000110 0101011 1010011 0111000 0010011 0000101

Criptografía Cuántica (BB84)

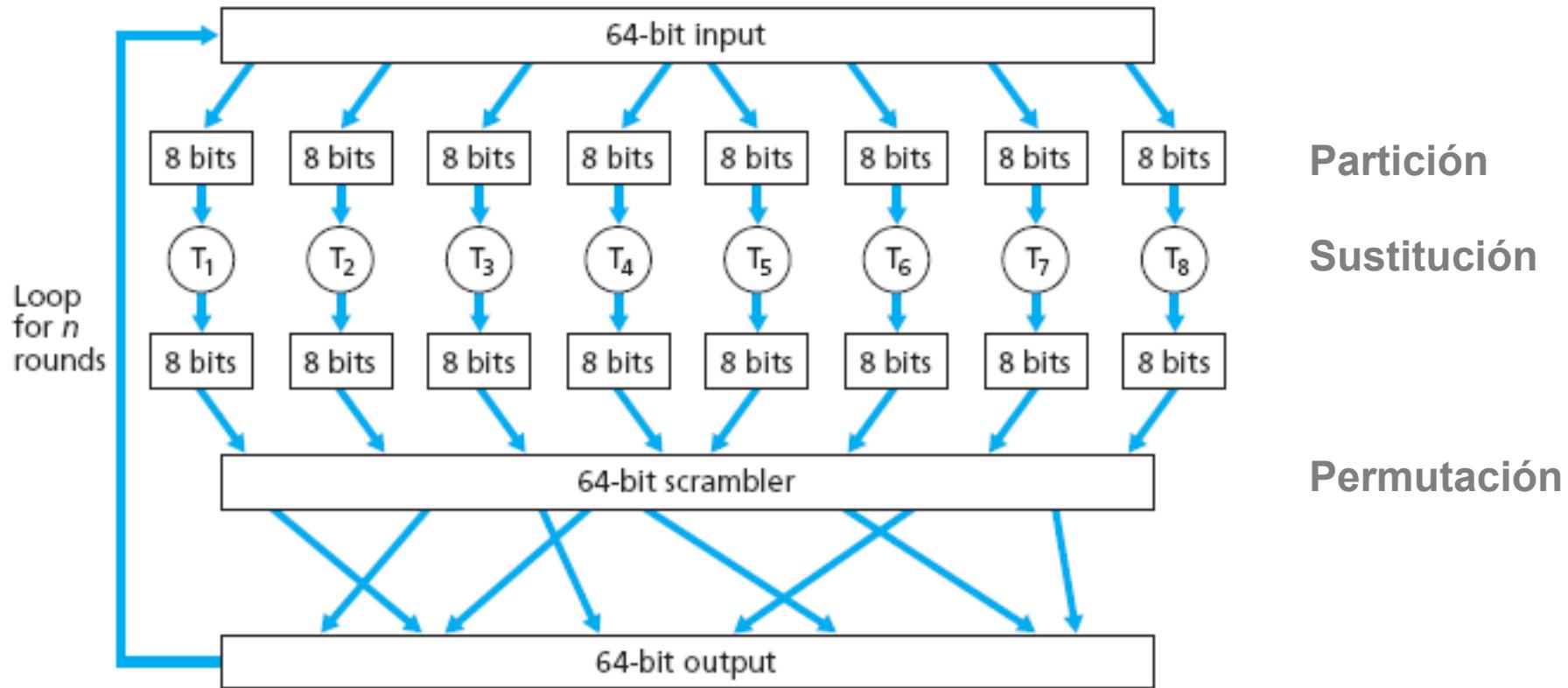


C.H. Bennett and G. Brassard (1984), "Quantum cryptography: public key distribution and coin tossing", Proceedings of IEEE International Conference on Computers, Systems and Signal Processing, IEEE press., pp. 175-179.

Criptografía Cuántica

- “Quantum cryptography was proposed first by Stephen Wiesner, then at Columbia University in New York, who, in the early 1970s, introduced the concept of quantum conjugate coding. His seminal paper titled "Conjugate Coding" was **rejected** by IEEE Information Theory but was eventually **published in 1983 in SIGACT News** (15:1 pp. 78–88, 1983). In this paper he showed how to store or transmit two messages by encoding them in two "conjugate observables", such as linear and circular polarization of light, so that either, but not both, of which may be received and decoded. He illustrated his idea with a design of unforgeable bank notes.
- A decade later, building upon this work, Charles H. Bennett, of the IBM Thomas J. Watson Research Center, and Gilles Brassard, of the Université de Montréal, proposed a method for secure communication based on Wiesner's "conjugate observables". In 1990, independently and initially unaware of the earlier work, Artur Ekert, then a Ph.D. student at Wolfson College, University of Oxford, developed a different approach to quantum key distribution based on peculiar quantum correlations known as quantum entanglement.”
- http://en.wikipedia.org/wiki/Quantum_cryptography

Cifrado de Bloque iterativo

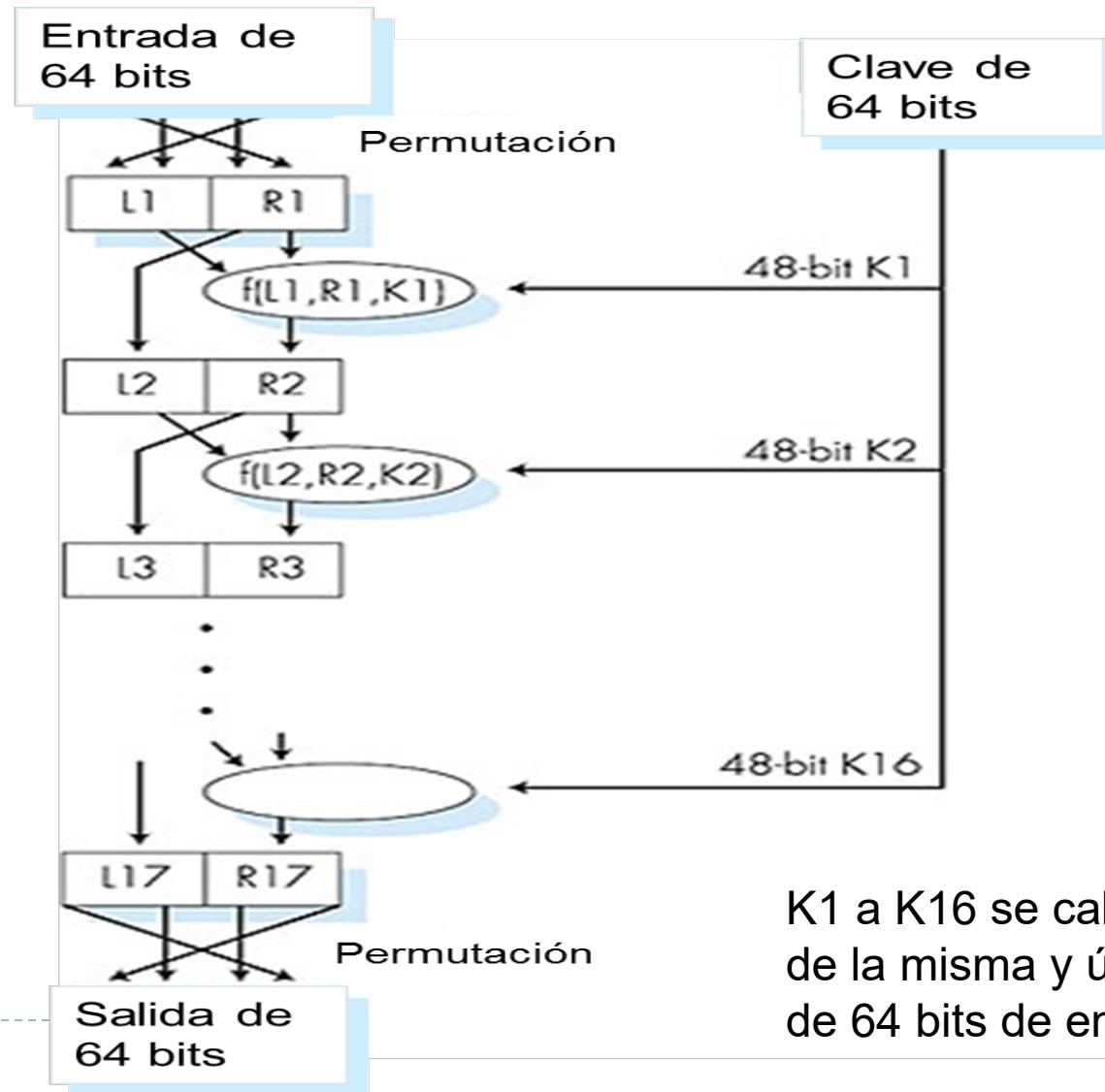


Se toma un **bloque de n bits de texto plano** como entrada y se lo transforma (T) utilizando la clave en un **bloque de n bits de texto cifrado**

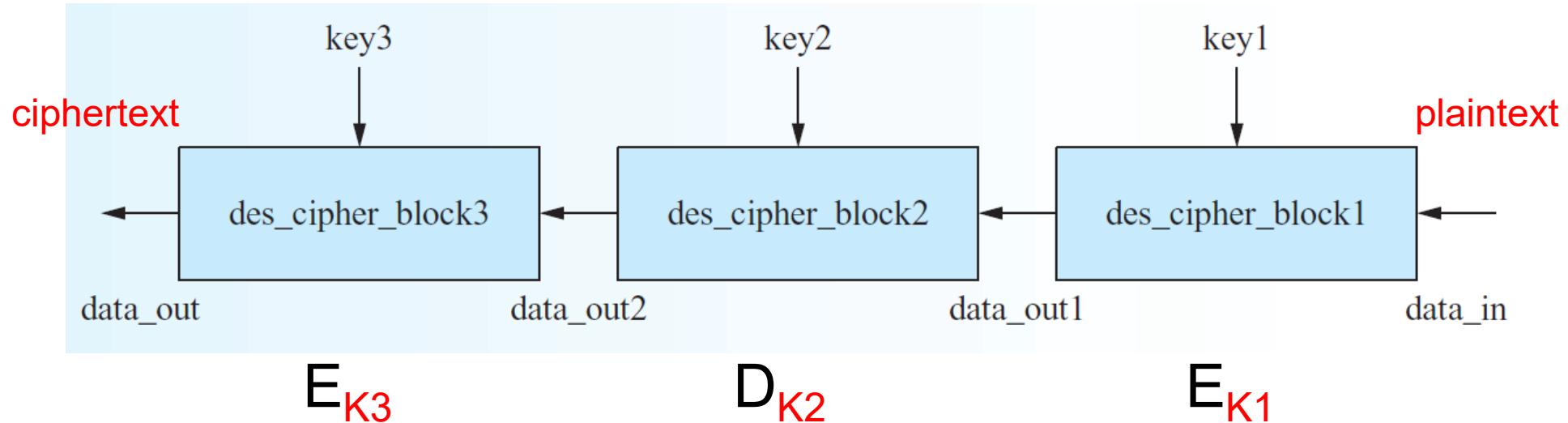
DES (Data Encryption Standard - IBM 1977)

- ▶ Se basa en Cifrado de Bloques.
- ▶ Orientado a bits, no a caracteres.
- ▶ Usa tanto **transposición** como **sustitución**.
- ▶ Trabaja en **bloques de a 64 bits**.
- ▶ Cada bloque ejecuta **16 iteraciones con distintas claves**.
- ▶ Con el tiempo fue quedando obsoleto el largo de la clave y **los ataques por fuerza bruta comenzaron a ser computacionalmente más rápidos**.
- ▶ También se le encontraron vulnerabilidades para algunas claves específicas.
- ▶ AES es lo mínimo recomendado hoy en día.
- ▶ Claves mas largas resuelven vulnerabilidades en DES.

DES - Detalles



3DES



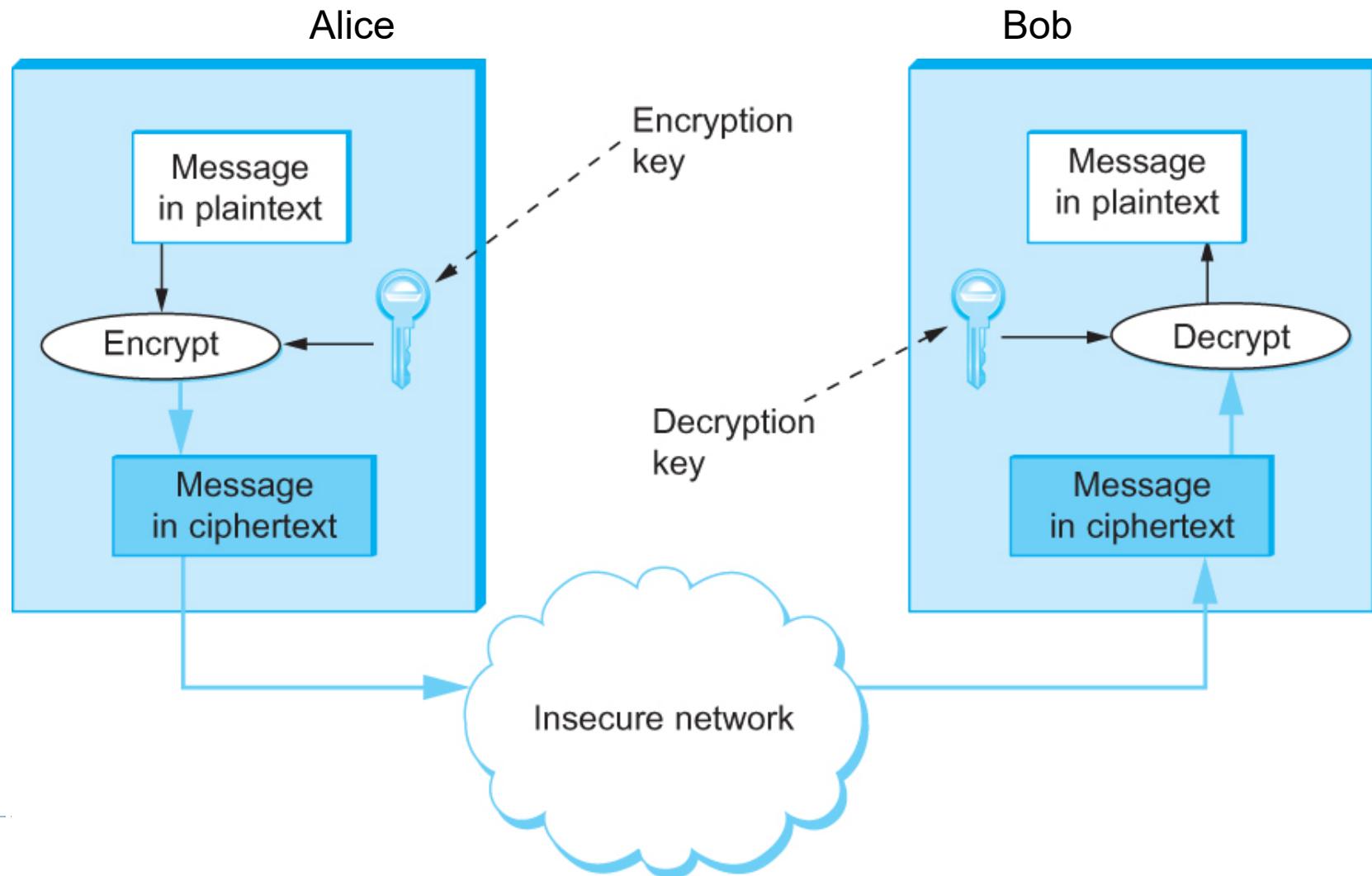
- ▶ Encripción: $ciphertext = E_{K3}(D_{K2}(E_{K1}(\text{plaintext})))$
- ▶ Desencripción: $\text{plaintext} = D_{K1}(E_{K2}(D_{K3}(ciphertext)))$



Seguridad en Redes

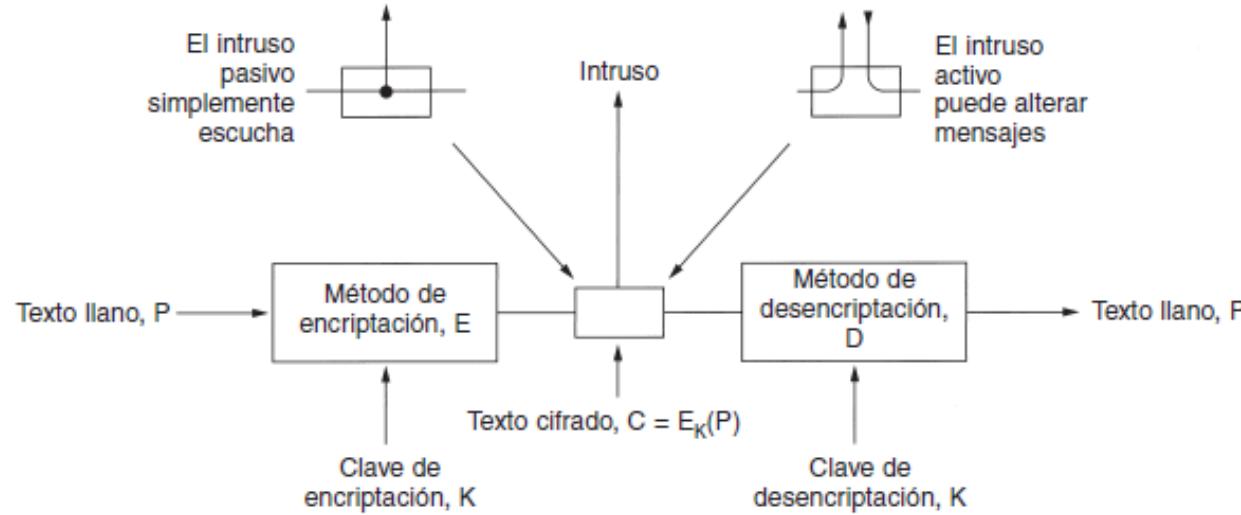
Algoritmos de clave Simétrica – Cifrado por Clave Privada

Criptografía Simétrica



Modelo de Encriptación

Modelo genérico
para clave simétrica



- ▶ $C=E_K(P)$ para representar la encripción del plaintext P usando la **clave K** con el método de encripción E , generando el ciphertext C
- ▶ En forma similar; $P=D_K(C)$ representa la desencripción de C para obtener, nuevamente, el plaintext P , usando la **clave K**
- ▶ En consecuencia $P=D_K(E_K(P))$
- ▶ Esto sugiere que E y D son funciones matemáticas de dos parámetros, uno de los cuales representa la clave



Seguridad en Redes



Algoritmos de clave Asimétrica - Cifrado por Clave Pública

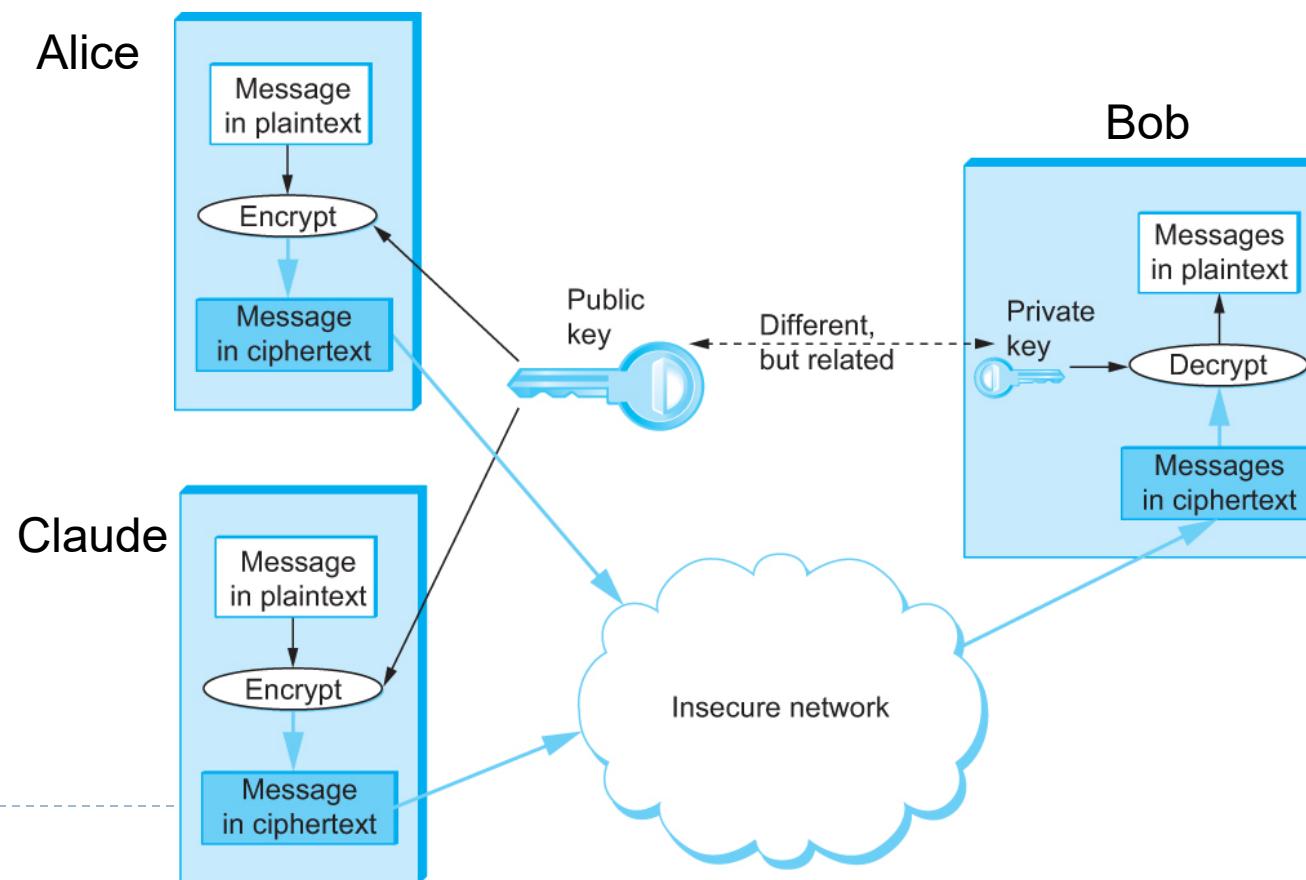
Criptografía de clave Asimétrica

- ▶ Criptografía de **clave simétrica**
 - ▶ Requiere que el emisor y el receptor conozca la **clave secreta** compartida.
 - ▶ ¿Cómo ponerse de acuerdo en la clave, especialmente si nunca se han visto?
- ▶ Criptografía de **clave pública**
 - ▶ Enfoque radicalmente distinto
 - ▶ [Diffie-Hellman 1976, RSA (Rivest–Shamir–Adleman) 1978]
 - ▶ Emisor y receptor **no comparten una clave secreta**
 - ▶ Clave de **encriptación pública** conocida por todos
 - ▶ Clave de **desencriptación privada**, conocida sólo por el receptor

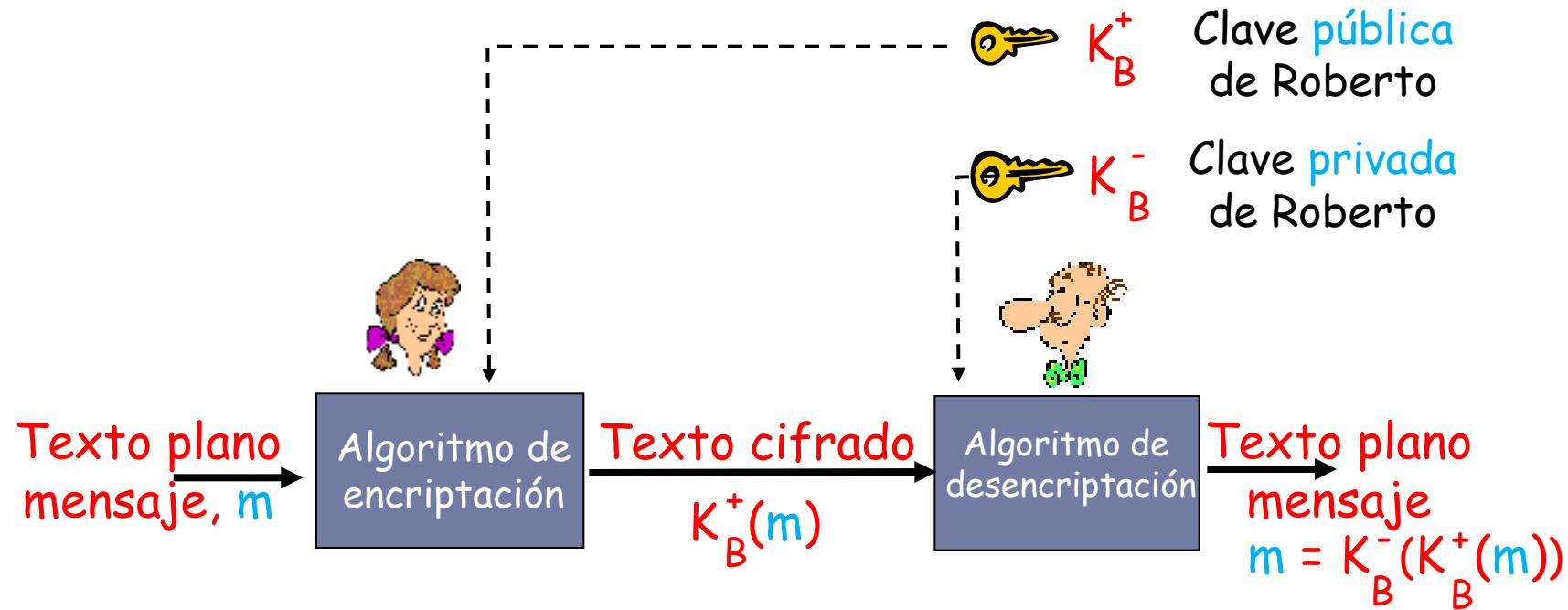


Criptografía de clave pública

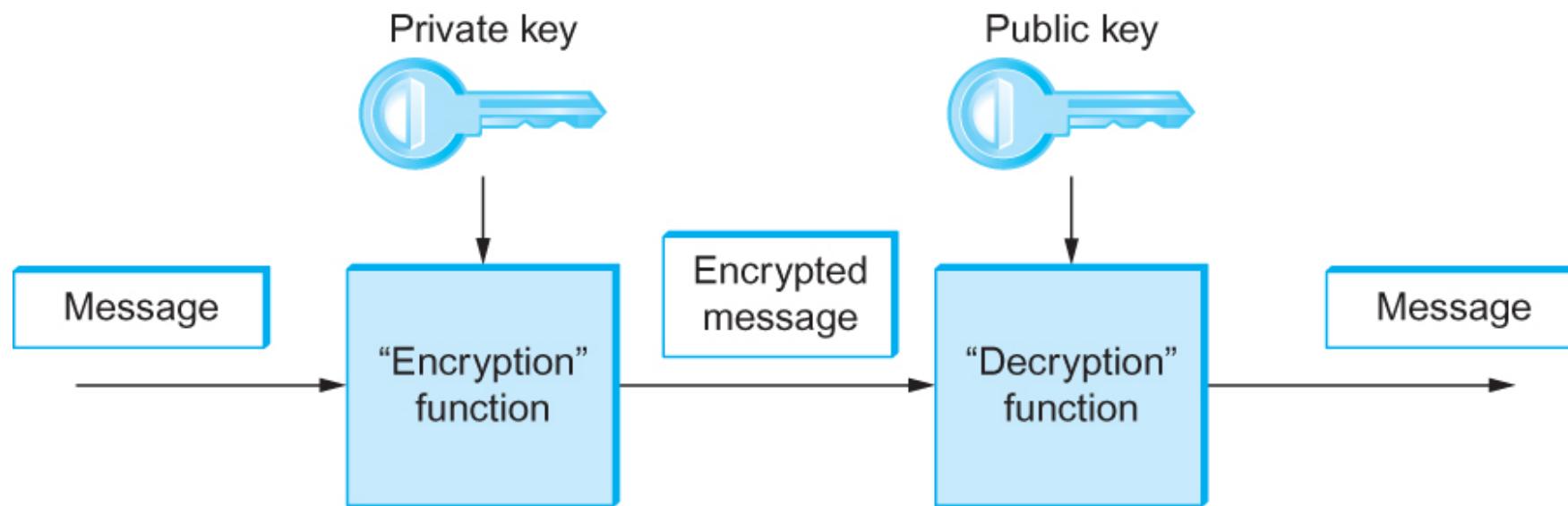
- Las claves de encriptado (**pública**) y desencriptado (**privada**) son lo suficientemente diferentes como para que la segunda no pueda calcularse a partir de la primera.



Criptografía de clave pública



Autenticación



Requisitos

- ▶ Debe ser fácil cifrar o descifrar, dada la clave adecuada
- ▶ Debe ser **inviable computacionalmente** derivar la clave privada a partir de:
 - ▶ La clave pública
 - ▶ Un texto que ha sido descifrado (plaintext)

RSA - Rivest–Shamir–Adleman

- ▶ Los conceptos teóricos que sustentan este algoritmo son **propiedades matemáticas respecto a los números primos**, módulos y exponenciación.
 - ▶ Factorizar números muy grandes **en números primos** es muy costoso
 - ▶ $(x^a)^b = (x^b)^a$
- ▶ Existe un algoritmo capaz de producir **un par de claves**, que debido a estas propiedades **no son derivables una de la otra**, pero **tienen la propiedad de ser la inversa**.

RSA: elegir claves

1. Elegir dos **números primos grandes, p y q** .
(por ejemplo, 1024 bits cada uno)
2. Calcular $n = p \cdot q$ y $z = (p-1) \cdot (q-1)$
3. Elegir e (con e pequeño y $e < n$) que no tenga factores comunes con z
(e y z son primos relativos).
4. Encontrar un número d , tal que $e \cdot d \equiv 1 \pmod{z}$ sea divisible de forma exacta entre z (en otras palabras, $e \cdot d \bmod z = 1$).
5. La **clave pública** es (n, e) . La **clave privada** es (n, d) .

$$\begin{array}{c} \overbrace{n, e}^{\text{K}^+} \\ \text{B} \end{array} \quad \begin{array}{c} \overbrace{n, d}^{\text{K}^-} \\ \text{B} \end{array}$$

RSA: encriptación, desencriptación

1. Dados (n, e) y (n, d) calculados anteriormente.

2. **Para encriptar** un patrón de bits m , o plaintext, calcular:

$$c = m^e \bmod n \quad (\text{es decir, el resto cuando } m \text{ se divide por } n).$$

3. **Para desencriptar** el patrón de bits recibidos c , o ciphertext, calcular:

$$m = c^d \bmod n \quad (\text{es decir, el resto cuando } c \text{ se divide por } n).$$

$$m = \underbrace{(m^e \bmod n)}_c^{d \bmod n} \bmod n$$

RSA: encriptación, desencriptación

$$n = 33 \quad (p=3, q=11)$$

$$e = 3$$

$$\textcolor{blue}{d = 7}$$

$$m = (\textcolor{green}{m}^{\textcolor{red}{e}} \bmod n)^{\textcolor{blue}{d}} \bmod n$$

Plaintext (P)		Ciphertext (C)		After decryption	
Symbolic	Numeric	P^3	$P^3 \pmod{33}$	C^7	$C^7 \pmod{33}$
S	19	6859	28	13492928512	19
U	21	9261	21	1801088541	21
Z	26	17576	20	1280000000	26
A	01	1	1	1	01
N	14	2744	5	78125	14
N	14	2744	5	78125	14
E	05	125	26	8031810176	05

Sender's computation *Receiver's computation*

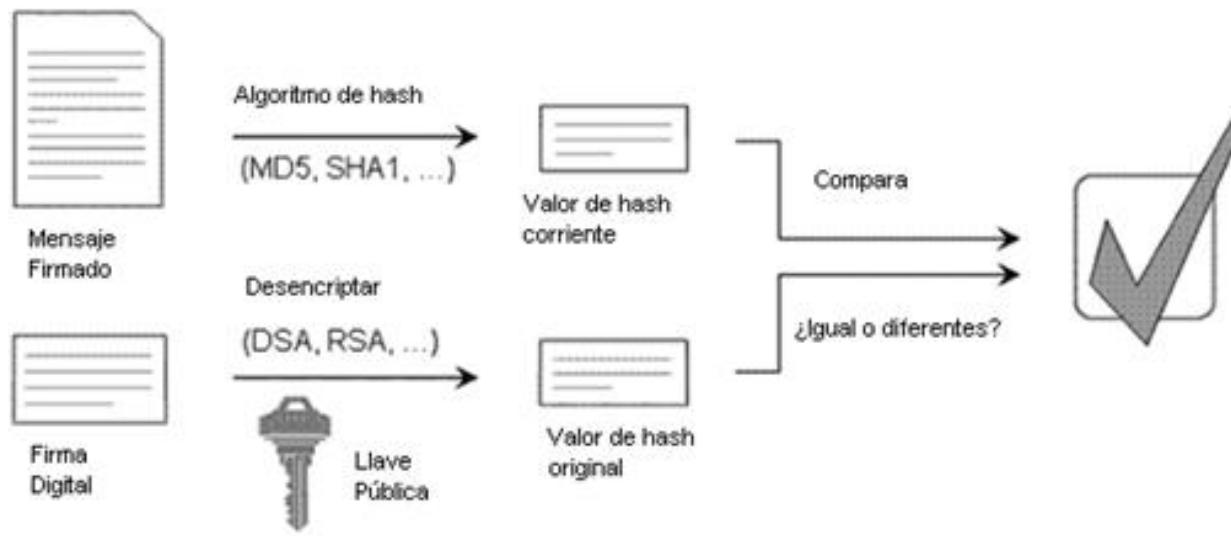
RSA: otra propiedad importante

$$\underbrace{K_B^{-}(K_B^{+}(m))}_{\text{Usar primero clave pública, seguida de clave privada}} = m = \underbrace{K_B^{+}(K_B^{-}(m))}_{\text{Usar primero clave privada, seguida de clave pública}}$$

Usar primero clave pública, seguida de clave privada

Usar primero clave privada, seguida de clave pública

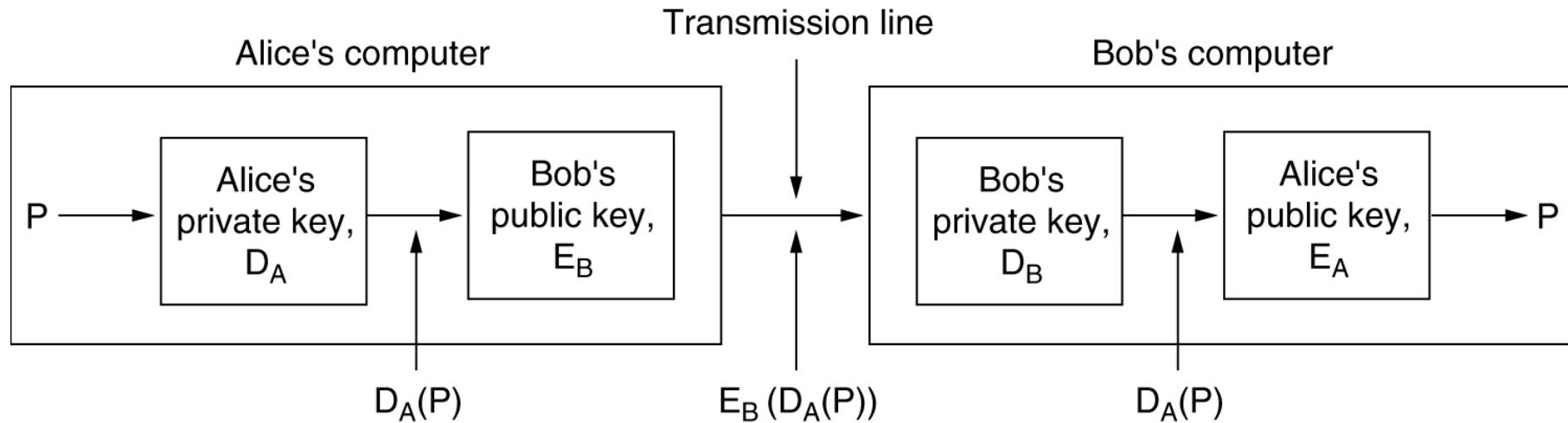
- Por ejemplo útil para firma digital



Seguridad en Redes

Firma Digital

Firma Digital con Criptografía Asimétrica

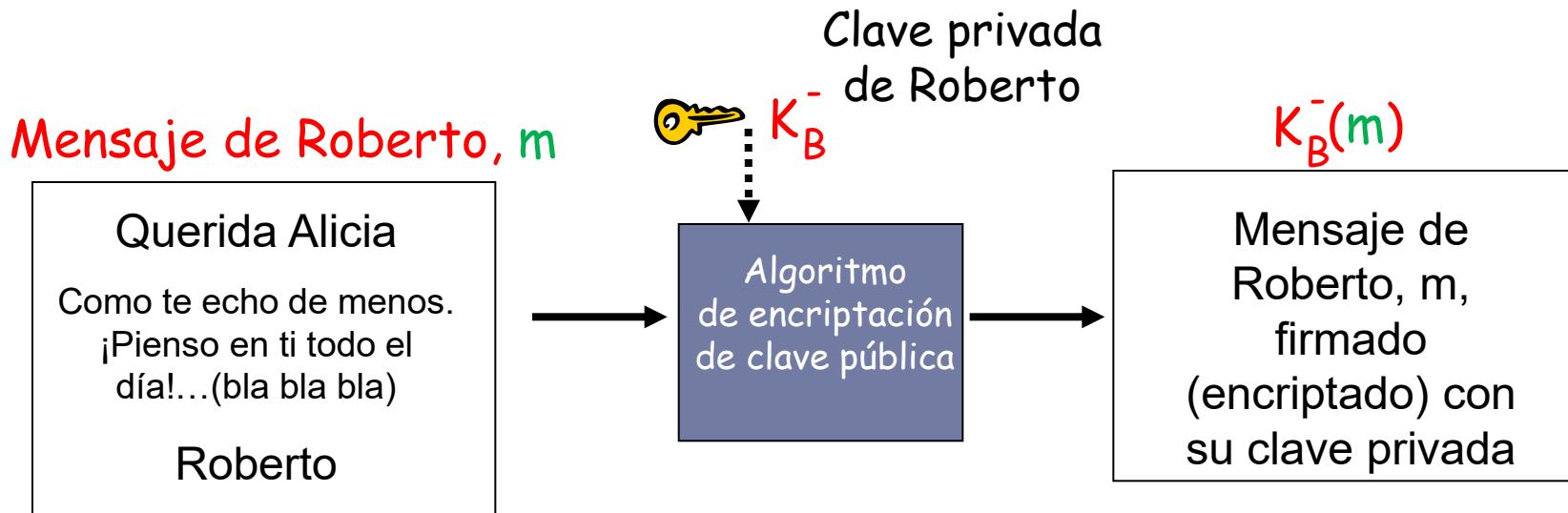


- ▶ Mensaje P
- ▶ No repudiación
 - ▶ Mediante **cifrado** con la Clave Privada
 - ▶ Bob puede producir P y $D_A(P)$, cosa que **solo Alice pudo haber hecho**.

Firma Digital

Firma digital simple para mensaje m

- ▶ Roberto firma m encriptándolo con su clave privada K_B , creando un mensaje “firmado” $K_B(m)$



Resumir el mensaje – Message Digest

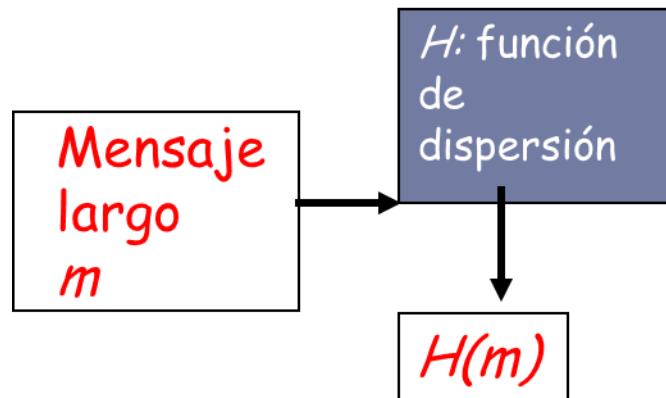
Encriptar con clave pública es computacionalmente caro para mensajes largos.

Objetivo: longitud fija, fácil de computar.

- ▶ Aplicar función de dispersión H a m , obtener resumen del mensaje de tamaño fijo, $H(m)$.

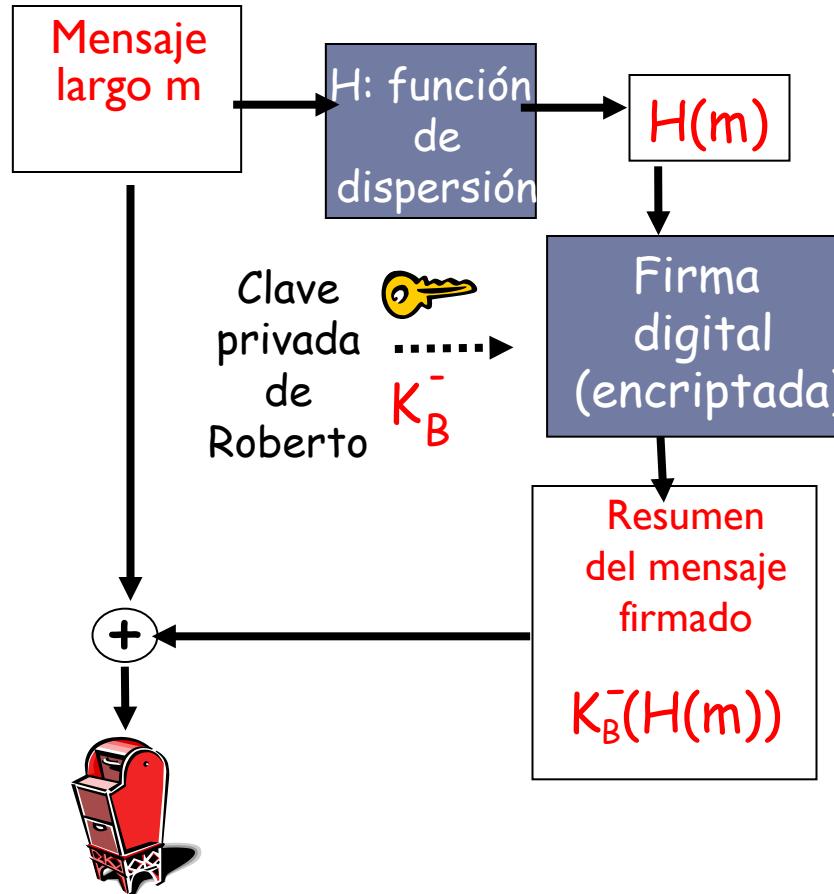
Propiedades de la función de Hash.

- ▶ Muchos a uno.
- ▶ Produce resumen de mensaje de tamaño fijo
- ▶ Dado el **resumen de mensaje x** , es **computacionalmente inviable hallar m para que $x = H(m)$** .

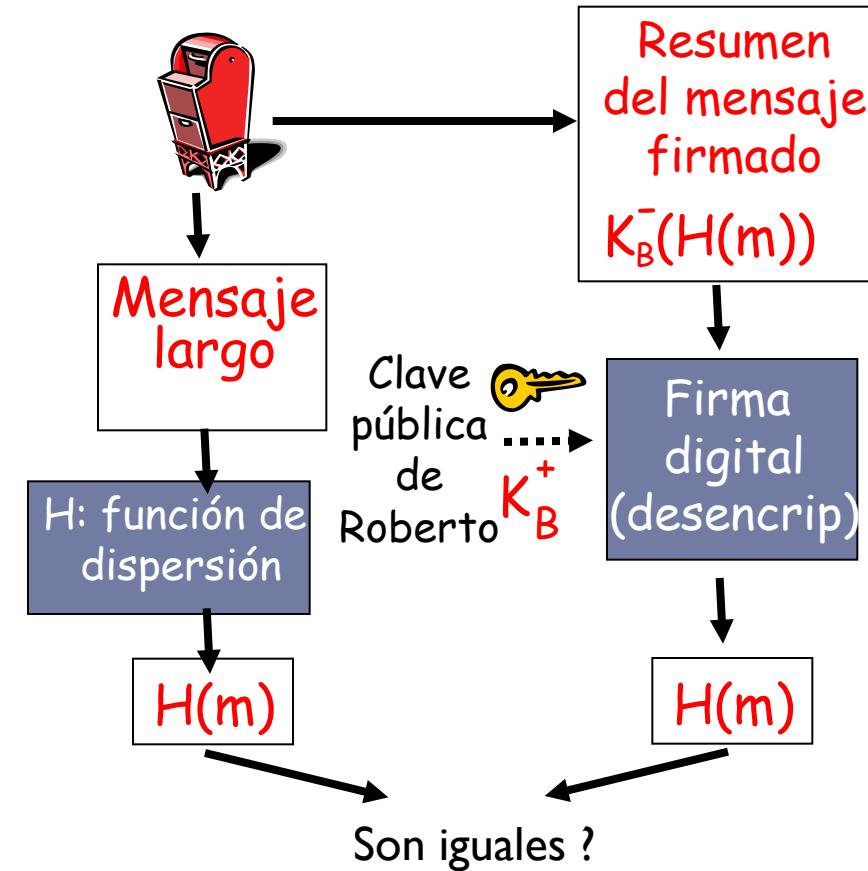


Firma Digital: Hash y Message Digest

Bob envía un mensaje largo



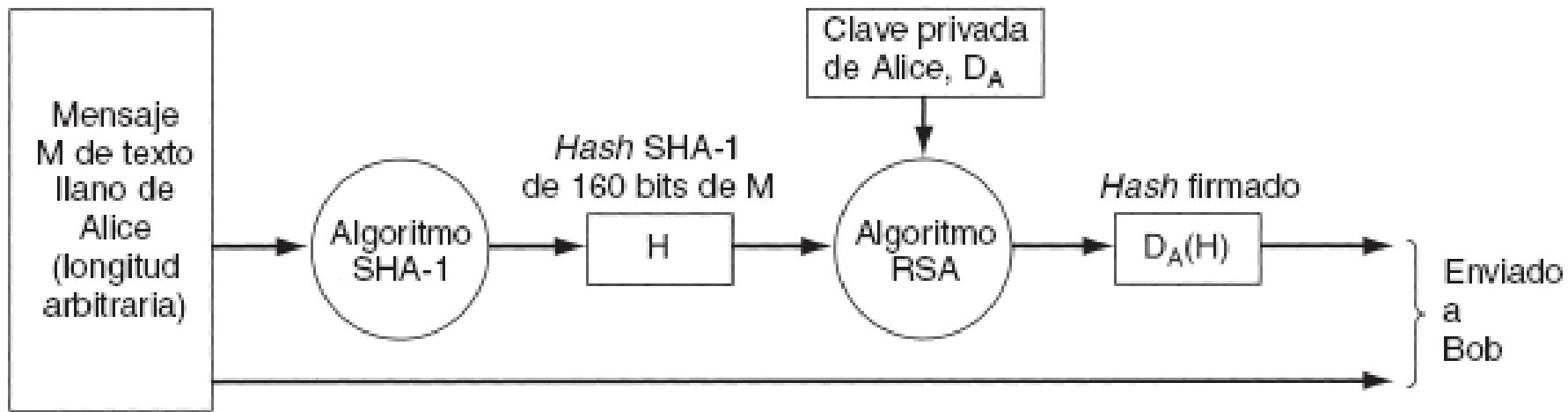
Alicia verifica la **firma** y la **integridad** del mensaje firmado digitalmente



Algoritmos para la función HASH

- ▶ **MD5 (RFC 1321):**
 - ▶ Calcula un resumen de mensaje de 128 bits en un proceso de cuatro pasos.
 - ▶ Cadena x arbitraria de 128-bit, parece difícil construir mensaje m cuya dispersión MD5 sea igual a x .
 - ▶ Colisiones encontradas
- ▶ **SHA-1:**
 - ▶ Estándar de EE.UU. [NIST, FIPS PUB 180-1].
 - ▶ Resumen de mensaje de 160 bits.
 - ▶ Colisiones encontradas
- ▶ **SHA-2**
 - ▶ Resumen de mensaje de 256 y 512 bits.

Firma Digital con RSA y SHA-1



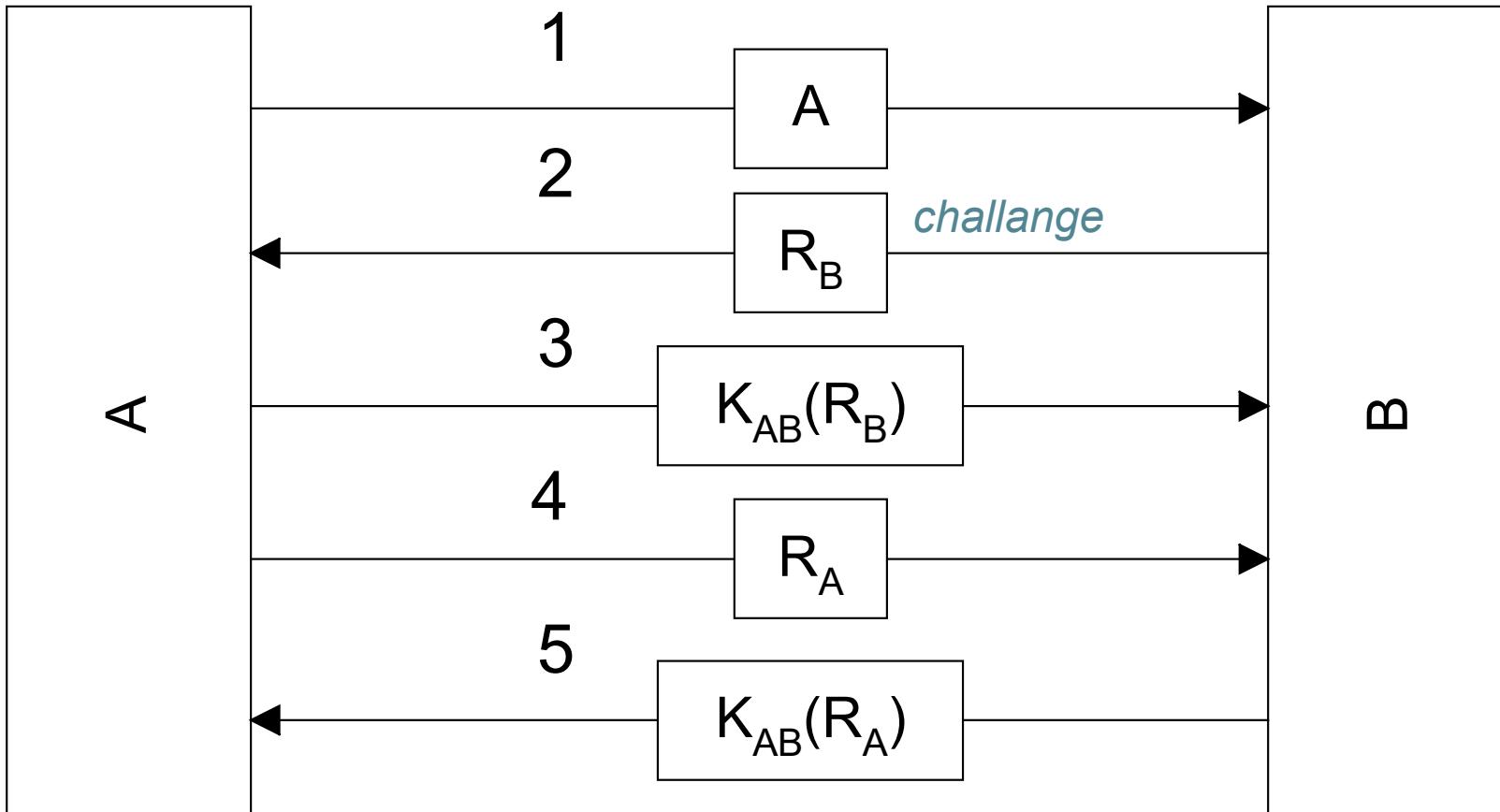


Seguridad en Redes



Autenticación

Autenticación de dos vías



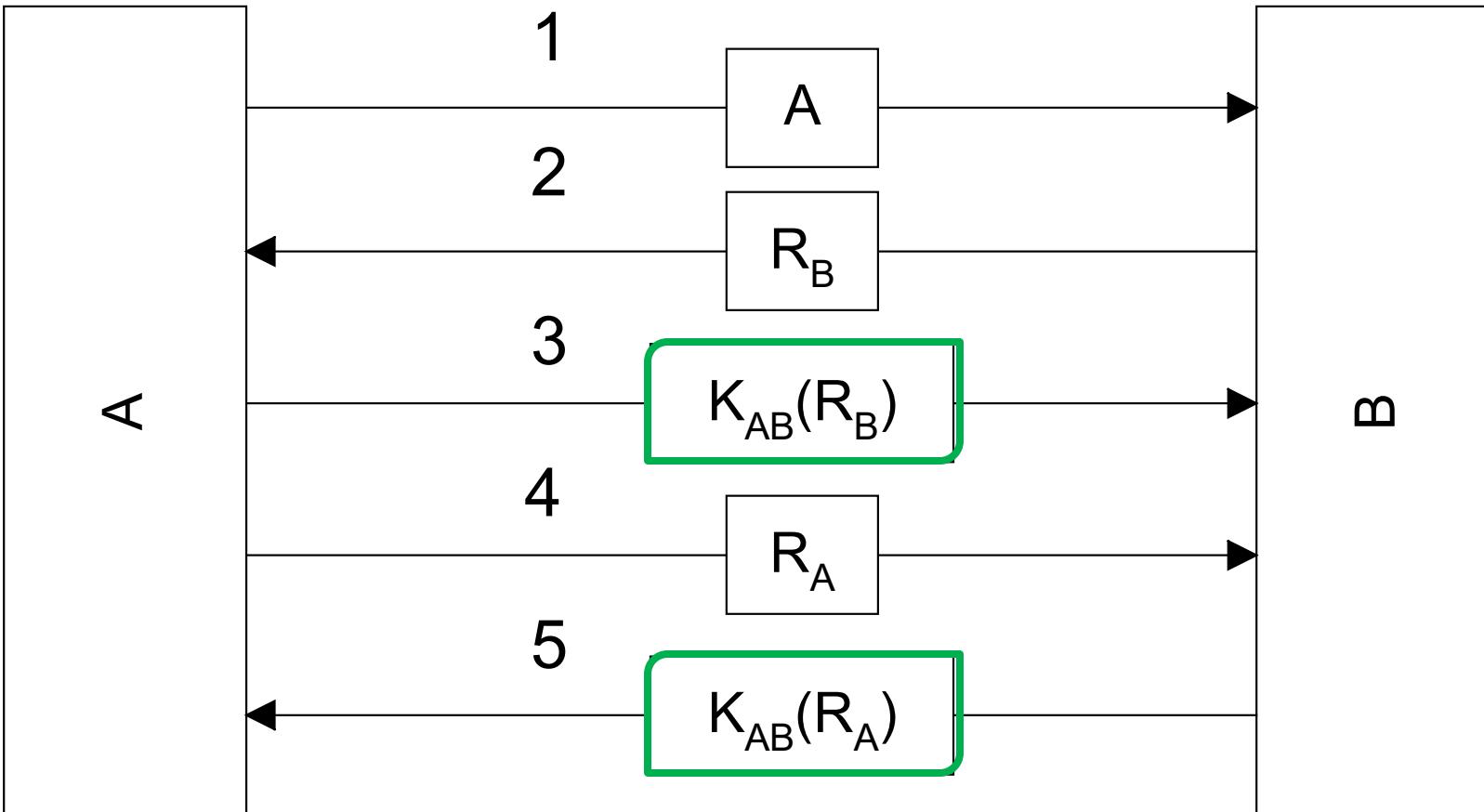
Protocolo “challenge-response”

K_{AB} : Clave Compartida

Autenticación basada en Claves Secretas Compartidas

- ▶ 1) A envía su identidad a B.
- ▶ 2) Dado que todavía B no puede determinar si el mensaje recibido es realmente de A o de un tercero T, B elige un *challenge* R_B , (un **número aleatorio suficientemente grande**) y se lo envía a A sin encriptar.
- ▶ 3) A encripta el mensaje 2 con la **clave que comparte** con B, $K_{AB}(R_B)$, y se lo devuelve a B.
- ▶ 4) Cuando B recibe este texto cifrado inmediatamente sabe que proviene de A, dado que es el único que conoce la clave K_{AB} .

Autenticación de dos vías



Protocolo “challenge-response”

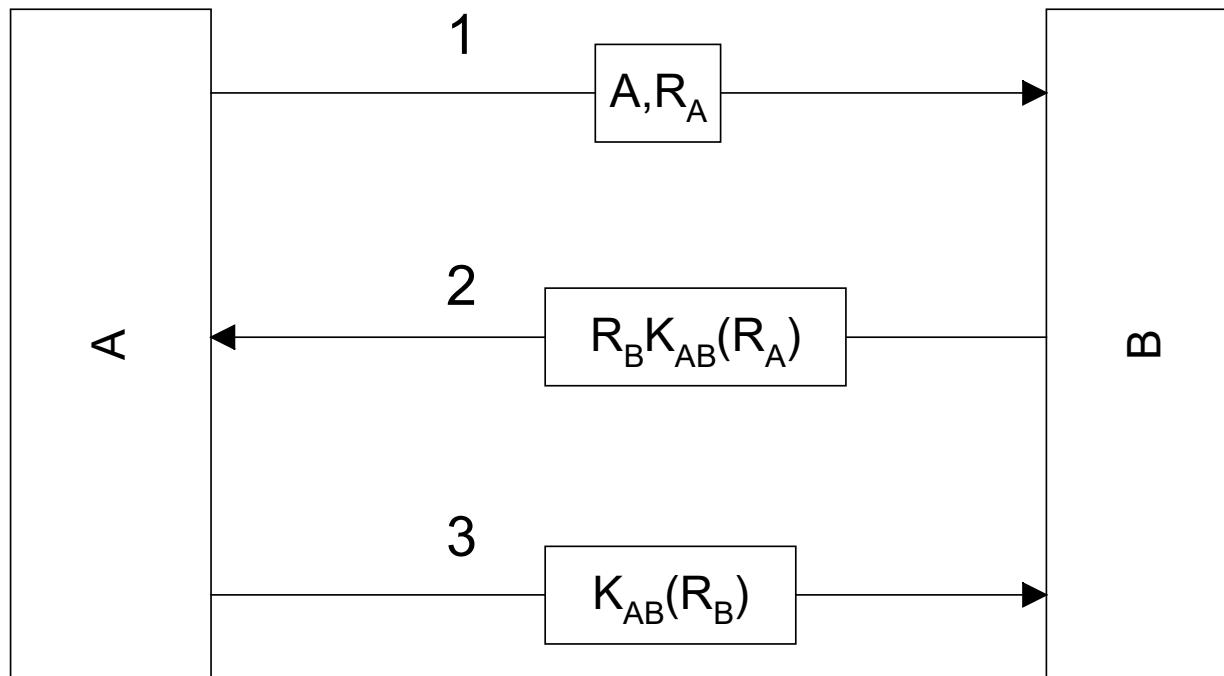
K_{AB} : Clave Compartida

Autenticación basada en Claves Secretas Compartidas

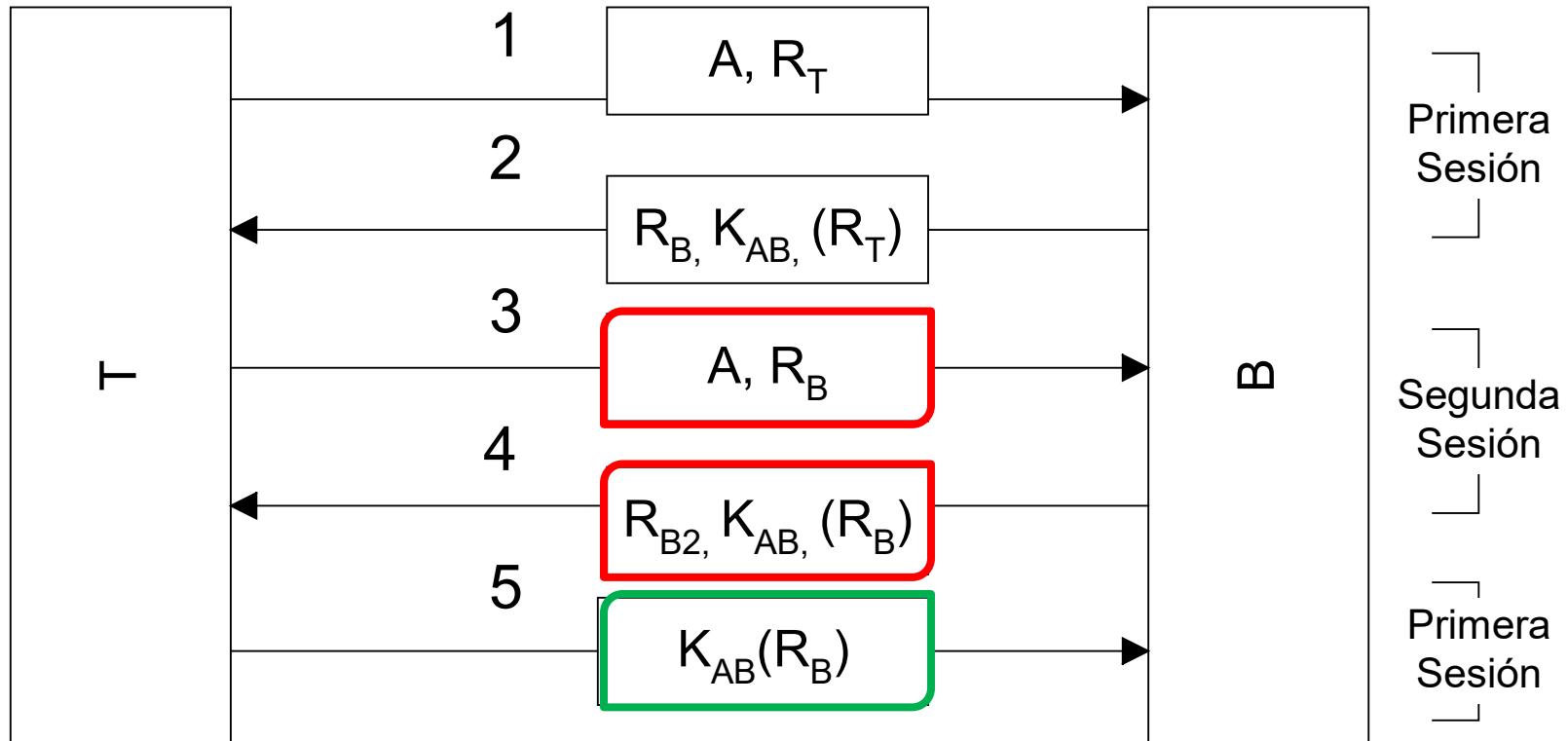
- ▶ La elección random y la longitud del **challenge** (e.g. 128 bits) hace realmente improbable que un **tercero tome R_B** y su **respuesta $K_{AB}(R_B)$** de una sesión previa.
- ▶ En este punto B está seguro de la identidad de A, pero este último no **posee ninguna garantía acerca de B**.
- ▶ A efectos de verificar la identidad de B, A elige un número al azar, R_A , y se lo envía a B sin encriptar (mensaje 4).
- ▶ Cuando B **responde con $K_{AB}(R_A)$** , A se asegura de la identidad de B.
- ▶ En este momento, si ambos desean establecer **una clave de sesión**, entonces A seleccionará una K_S y lo enviará a B encriptado con K_{AB}

Protocolo Autenticación simplificado

Una forma de simplificar la secuencia anterior de envío de mensajes es haciendo que cada participante transmita su identidad y el challenge elegido en el mismo mensaje, sin esperar al envío correspondiente de la otra parte.



Ataque por sesiones



Protocolo “challenge-response”

T: tercero

Ataque por sesiones

Si resulta posible establecer **sesiones múltiples** entre los participantes, podría darse la siguiente secuencia:

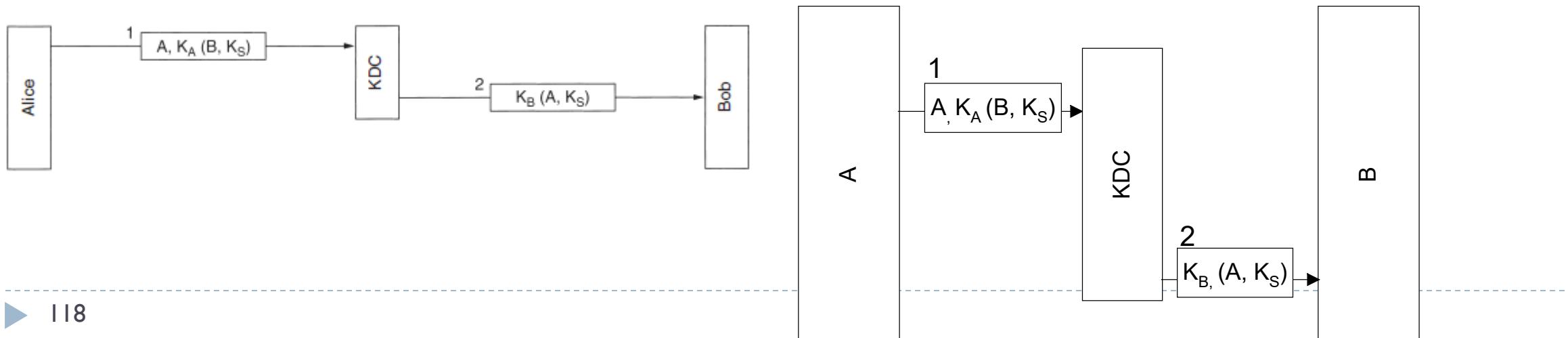
- ▶ En el mensaje 1, un tercero T, simula ser A, enviando la identidad de A y R_T .
- ▶ En el mensaje 2, B responde como siempre con su challenge R_B , y espera a que A se lo devuelva encriptado.
- ▶ En el mensaje 3, dado que T no conoce la clave de encripción, no puede devolver el challenge de B encriptado, entonces inicia una segunda sesión y envía como su challenge R_B .
- ▶ Cuando B le devuelve el challenge encriptado $K_{AB}(R_B)$, en el mensaje 4, T utiliza esto como respuesta al mensaje 2 de la primer sesión, logra engañar a B, y aborta la segunda.

Reglas de diseño

- ▶ El participante que inicia la transmisión debe probar su identidad en forma previa al participante receptor.
- ▶ Ambos participantes deben usar claves diferentes para la verificación de identidades, aún cuando esto signifique tener dos claves compartidas K_{AB} y K'_{AB} .
- ▶ Deben elegir los challenges de conjuntos diferentes, por ejemplo el que inicia la transmisión del set de números pares y quien contesta a partir de los impares.
- ▶ Que resista ataques que involucren una segunda sesión paralela, en la que la información obtenida se use en una sesión diferente.

Distribución de claves confiable (KDC - *Key Distribution Center*)

- ▶ A selecciona una clave de sesión K_S y le comunica al KDC su intención de hablar con B.
- ▶ Este mensaje es encriptado utilizando la clave secreta K_A que A comparte solamente con el KDC.
- ▶ El KDC desencripta el mensaje tomando la identidad de B y la clave de sesión y construye un nuevo mensaje conteniendo la identidad de A y la clave de sesión, y lo envía a B encriptado con K_B , la clave secreta que B comparte con el KDC.
- ▶ Cuando B desencripta el mensaje, sabe que A se quiere comunicar con él y la clave que desea usar.



Autenticación

Top 4 Most Used Authentication Mechanisms

blog.bytebytogo.com

