# Capa Aplicación: Correo Electrónico

## ELO322: Redes de Computadores

Agustín J. González

Este material está basado en:

- Material de apoyo al texto *Computer Networking: A Top Down Approach Featuring the Internet*. Jim Kurose, Keith Ross.

# Capítulo 2: Capa Aplicación

- 2.1 Principios de la aplicaciones de red
- 2.2 Web y HTTP
- 2.3 Correo Electrónico
  - SMTP, POP3, IMAP
- 2.4 DNS
- 2.5 Aplicaciones P2P
- 2.6 Streaming de video y redes de distribución de contenidos
- 2.7 Programación de socket con UDP y TCP

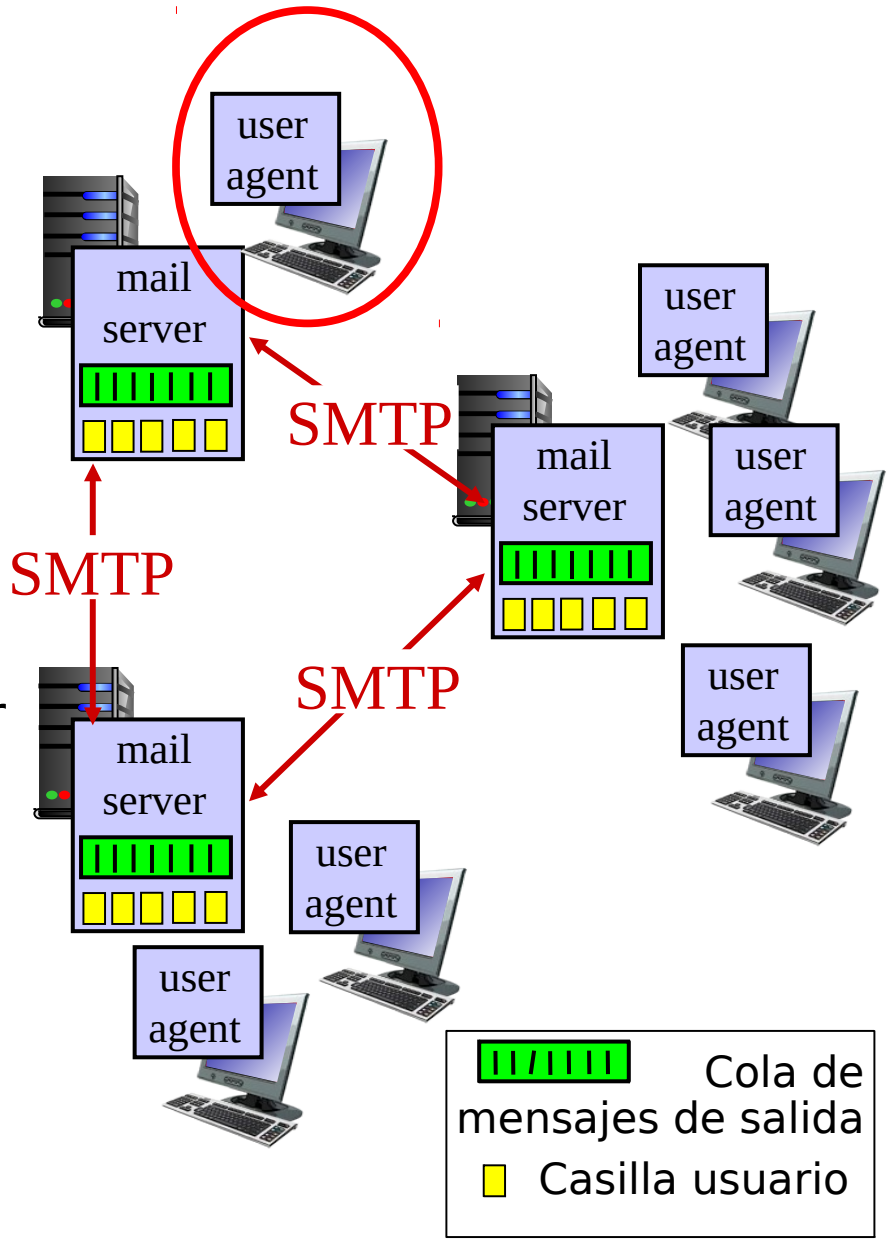
# Correo Electrónico

## Tres componentes mayores:

- Agente usuario o cliente de correo
- Servidor de correo
- Simple Mail Transfer Protocol: SMTP

## Agente Usuario

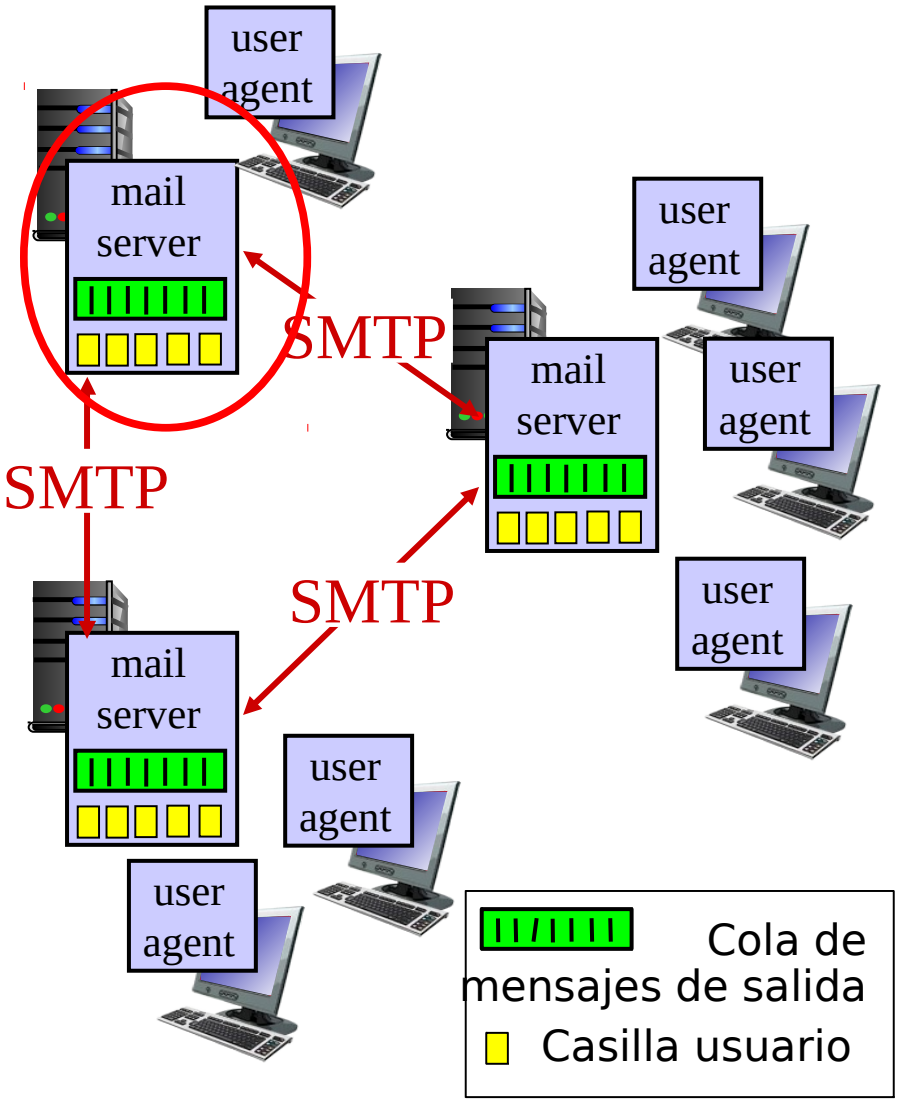
- También conocido como “lector de correo”
- Escritura, edición, lectura de mensajes de correos
- e.g., Outlook, Thunderbird, iPhone mail client
- Mensajes de salida y entrada son almacenados en servidor



# Correo Electrónico: Servidor de correo

## Servidor de Correo

- Casilla contiene mensajes de entrada para el usuario
- Cola de mensajes de los correos de salida
- SMTP: Protocolo entre servidores de correo para enviar mensajes e-mail
  - cliente: servidor que envía el correo
  - "servidor": servidor que recibe el correo
- También lo usa el agente usuario para enviar correo.



# Correo Electrónico: SMTP [RFC 2821]

- Usa TCP para transferir confiablemente mensajes e-mail desde el cliente al servidor, puerto 25 en servidor.
- Transferencia directa: servidor envía correos al servidor receptor
- Tres fases de la transferencia
  - handshaking (apretón de manos para establecer conexión)
  - transferencia de mensajes
  - cierre
- Interacción comandos/respuestas
  - comandos: Texto ASCII
  - respuesta: código de estatus y frase.
- Mensajes deben ser enviados en ASCII de 7-bits ¿Qué pasa con las fotografías y archivos binarios?

# Escenario: Alicia envía mensaje a Bob

1) Alicia usa agente usuario para componer el mensaje para `bob@someschool.edu`

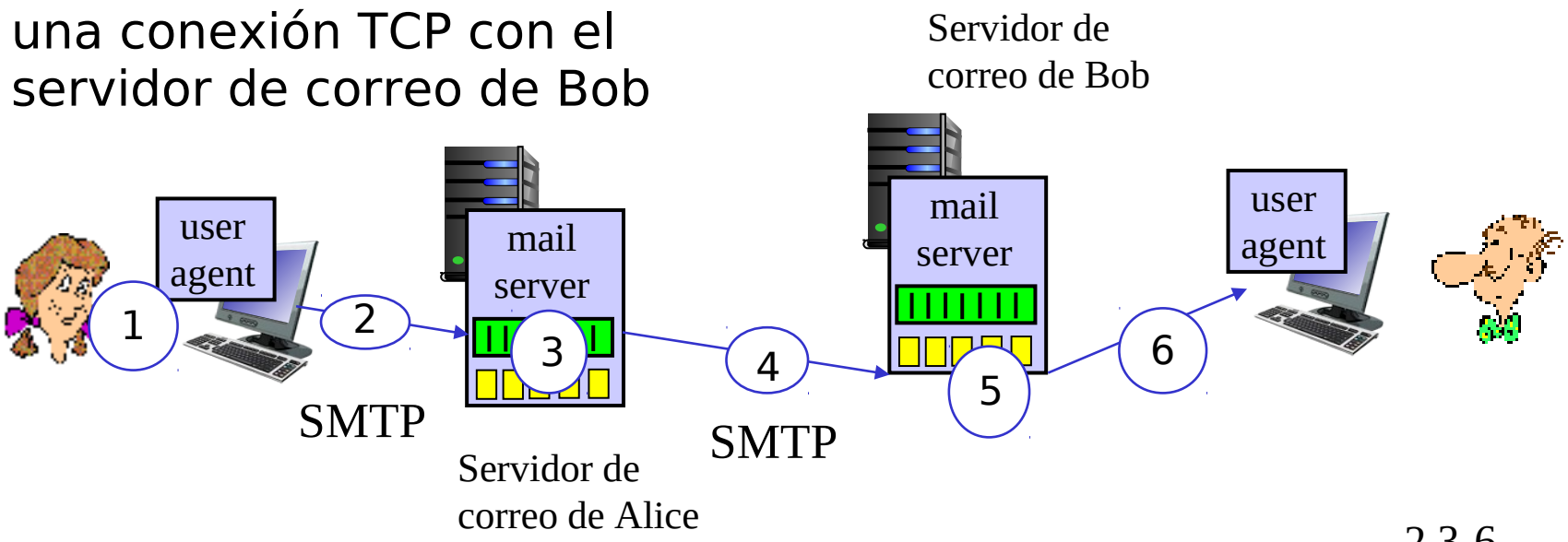
2) El agente de Alicia envía el mensaje a su servidor de correo; el mensaje es puesto en cola de salida

3) Lado cliente de SMTP abre una conexión TCP con el servidor de correo de Bob

4) El cliente SMTP envía el mensaje de Alicia por la conexión TCP

5) El servidor de correo de Bob pone el mensaje en su casilla

6) Bob invoca su agente usuario para leer el mensaje



# Ejemplo de Interacción SMTP

**Luego de: $telnet hamburger.edu 25 <enter>**

**S: 220 hamburger.edu**
**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**

En el pasado era posible comunicarse con servidor de correo usando telnet. Hoy los servidores ocupan conexiones seguras (con encriptación).

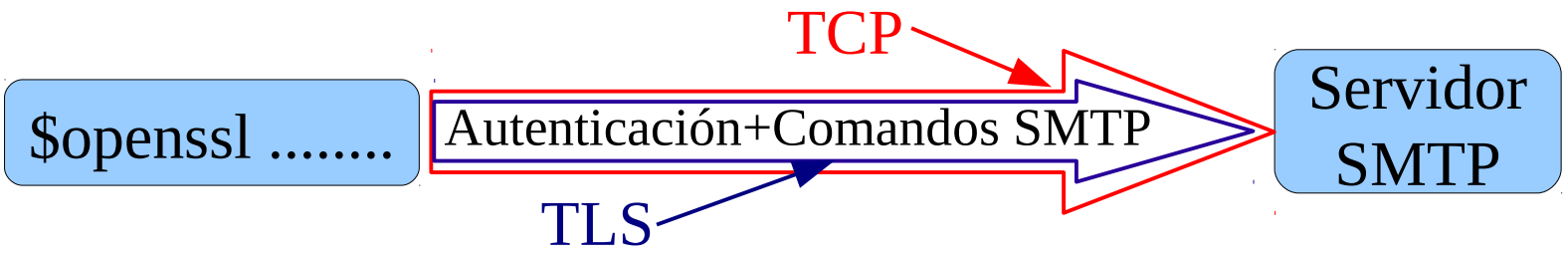
# Prueba de interacción SMTP (obsoleta)

- **`telnet <servername> 25`**
- Luego de respuesta 220 desde el servidor
- Ingresar los comandos HELO, MAIL FROM, RCPT TO, DATA, QUIT

Lo de arriba nos permitía enviar correo sin usar un cliente de correo.


- Hoy muchos servidores están configurados para aceptar sólo conexiones seguras y no permiten el uso de telnet para envío de correo. La USM y gmail usan TLS (Transport Layer Security)

# Prueba SMTP actual con gmail




Transpor Layer Security

- Ver datos para comunicación con gmail en su página.
- Servidor: smtp.gmail.com, puerto TLS:587
- Para crear la conexión segura al servidor smtp de gmail:
  - Primero debo hacer una conexión TLS hasta el servidor. Se puede usar comando openssl de linux.
  - Luego se envía autenticación al servidor gmail.

Ver caso gmail: https://taufanlubis.wordpress.com/2016/02/08/how-to-send-email-via-gmail-server-using-openssl/

# SMTP: palabras finales

- SMTP usa conexiones persistentes

  => Varios mensajes pueden ser enviados usando la misma conexión
- SMTP requiere que el mensaje (encabezado y cuerpo) estén en código ASCII de 7-bits
- Servidor SMTP usa `CRLF.CRLF` para terminar el mensaje; es decir, una línea con sólo un punto en ella.

## Comparación con HTTP:

- HTTP: pull (saca contenido desde servidor)
- SMTP: push (pone contenido en servidor)
- Ambos tienen interacción comando/respuesta en ASCII, y tienen códigos de estatus
- HTTP: cada objeto es encapsulado en su propio mensaje
- SMTP: múltiples objetos son enviados en un mensaje multiparte

# Formato de mensajes de correo (comando DATA)

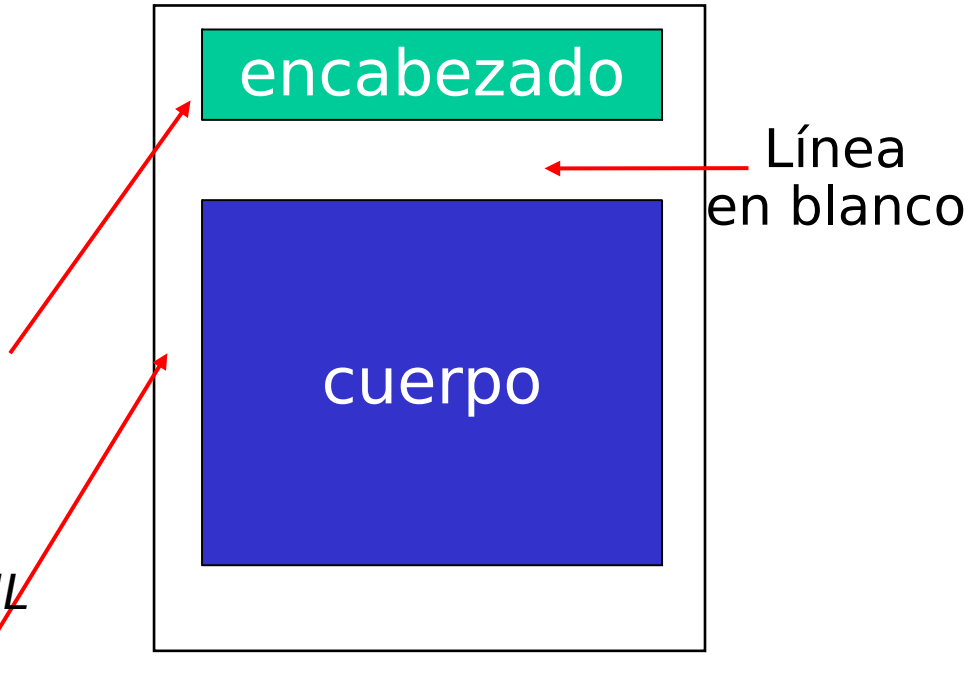
SMTP: protocolo para intercambio de mensajes de correo

RFC 822: estándar para el formato de los mensajes:

- E.g. líneas de encabezado (opcional), entre otros:
  - To:
  - From:
  - Subject:

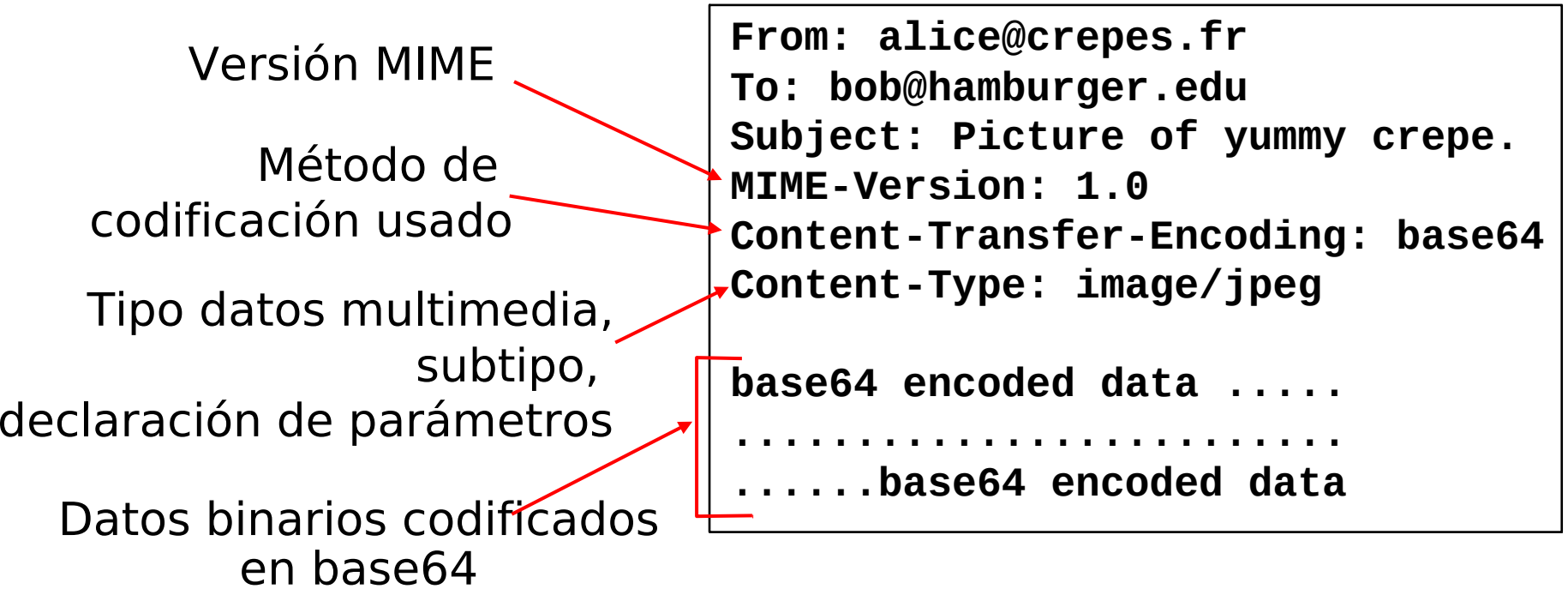
*diferente a los comandos SMTP MAIL FROM, RCPT TO. !!*

- Cuerpo
  - El “mensaje”, sólo caracteres ASCII

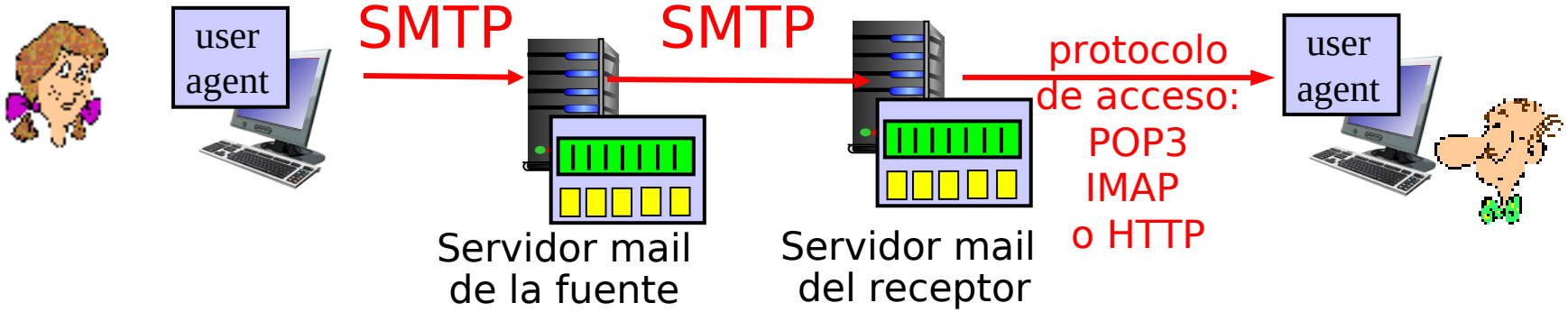


# Formato de mensaje: extensiones multimedia

- MIME: “multimedia mail extension”, RFC 2045, 2056
- Líneas adicionales en el encabezado del mensaje declaran el tipo de contenido MIME
- La codificación Base64 usa sólo los caracteres: A-Z, a-z, 0-9 y +/=



# Protocolos de acceso al correo




- SMTP: permite envío y almacenamiento de correo al servidor del destinatario
- Protocolo de acceso a correo: permite extraer correo desde el servidor destinatario
  - POP: Post Office Protocol [RFC 1939]
    - autorización (agent <-->server) y bajada
  - IMAP: Internet Mail Access Protocol [RFC 1730]
    - Más características que POP (IMAP es más complejo)
    - Permite manipulación de los mensajes almacenados en el servidor
  - HTTP: gmail, Hotmail , Yahoo! Mail, etc.

# Protocolo POP3

## Fase de autorización

- Comandos del cliente:
  - **user:** declara username
  - **pass:** password
- Respuestas del servidor:
  - **+OK**
  - **-ERR**

## Fase transaccional, cliente:

- **list:** lista números de mensajes
- **retr:** extrae mensajes por su número (retrieve)
- **dele:** borra
- **quit**

```
S: +OK POP3 server ready
C: user bob
S: +OK
C: pass hungry
S: +OK user successfully logged on
```

```
C: list
S: 1 498
S: 2 912
S: .
C: retr 1
S: <message 1 contents>
S: .
C: dele 1
C: retr 2
S: <message 1 contents>
S: .
C: dele 2
C: quit
S: +OK POP3 server signing off
```

Tamaño del mensaje

# POP3 (más) e IMAP

## Más sobre POP3

- Ejemplo previo usa modo “bajar y borrar”.
- Bob no puede releer el correo si cambia el cliente
- Modo “bajada y conserva”: obtiene copia de los mensajes en diferentes clientes.
- POP3 no mantiene el estado de una sesión a otra (“stateless”)

## IMAP

- Mantiene los mensajes en el servidor
- Permite que el usuario organice sus correos en carpetas
- IMAP mantiene el estado del usuario de una sesión a otra:
  - Nombre de carpetas y mapeo entre Ids (identificadores) de mensajes y nombres de carpetas.

/* Si usted sabe programar sockets,
usted puede escribir un cliente de correo. */

# Origen de Web Mail

- Diciembre 1995: Dos personas aparecen con la idea frente a un inversionista. Formaron Hotmail.
- Tres empleados y 14 part-times desarrollaron la primera versión en 7 meses (Julio 1996).
- En menos de 1 año y medio Hotmail tenía 12 millones de cuentas y fue comprado por Microsoft en 400 millones de dólares.
- Éxito se logra por haber sido los primeros y por tratarse de una aplicación que se difunde sola. Es de interés de muchos. Buscar concepto “Killer application”

# Capítulo 2: Capa Aplicación

- 2.1 Principios de la aplicaciones de red
- 2.2 Web y HTTP
- 2.3 Correo Electrónico
  - SMTP, POP3, IMAP
- 2.4 DNS
- 2.5 Aplicaciones P2P
- 2.6 Streaming de video y redes de distribución de contenidos
- 2.7 Programación de socket con UDP y TCP