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

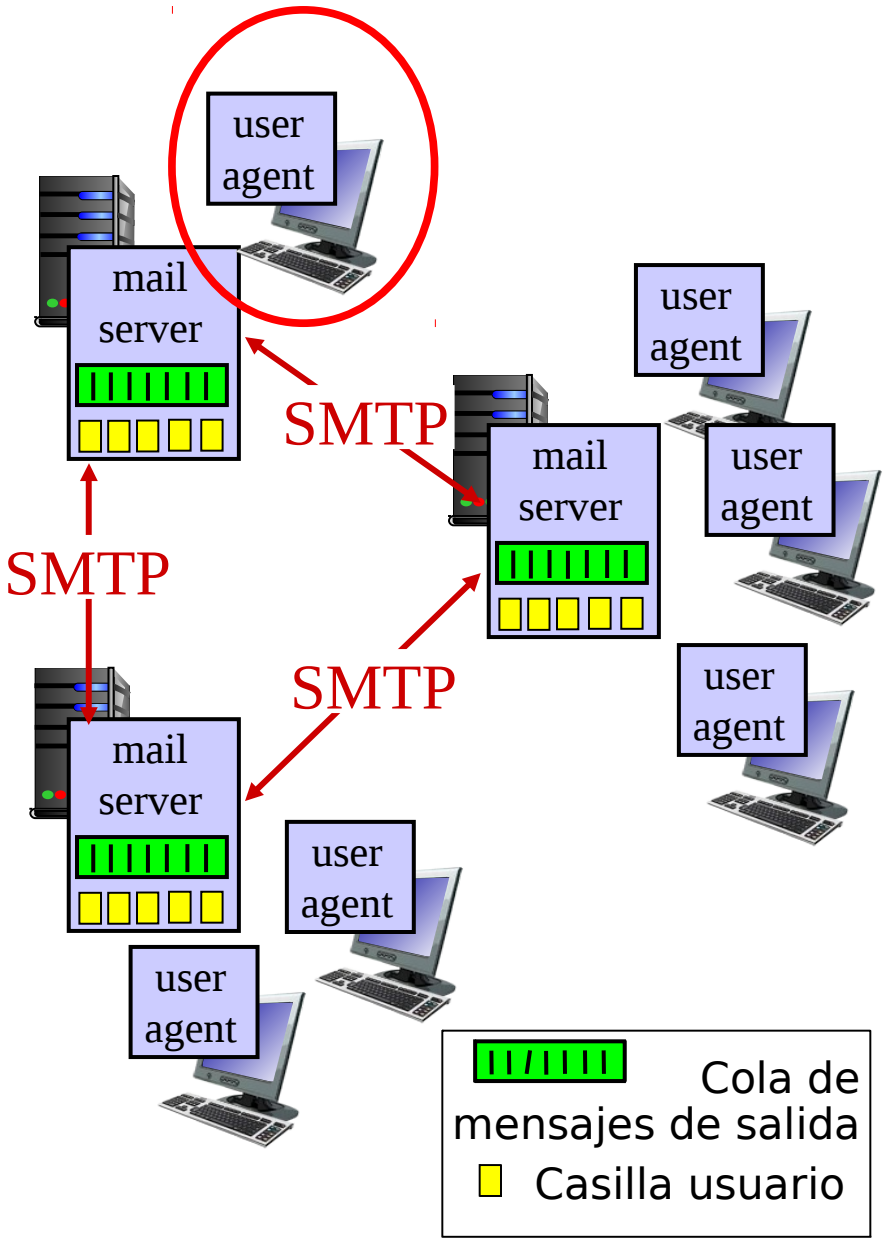
Correo Electrónico

Tres componentes mayores:

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

Agente Usuario

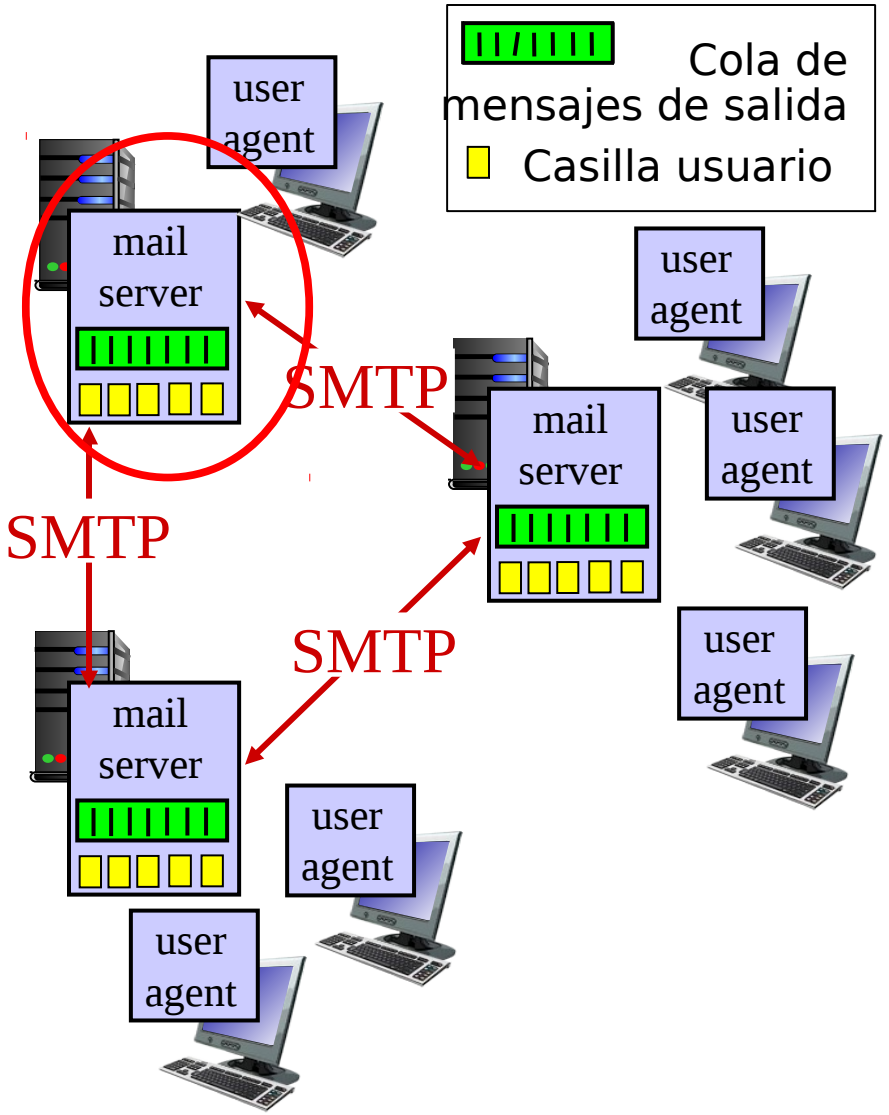
- También conocido como “lector de correo”
- Escritura, edición, lectura de mensajes de correos
- e.g., Outlook, Thunderbird, iPhone mail client
- Mensajes de salida y entrada son almacenados en servidor



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`

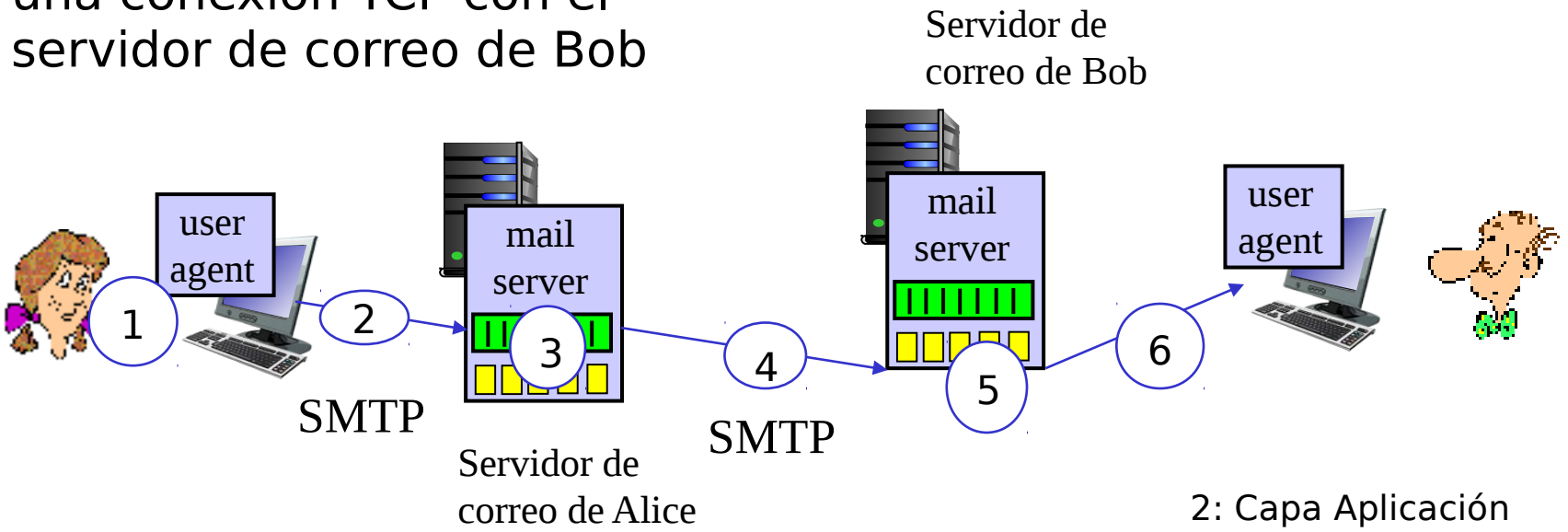
2) El agente de Alicia envía el mensaje a su servidor de correo; el mensaje es puesto en cola de salida

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 esto era posible. Hoy los servidores ocupan conexiones seguras, telnet no es seguro no cifra en mensaje y cualquiera podría suplantar a otro. autenticación. SMTP no encripta

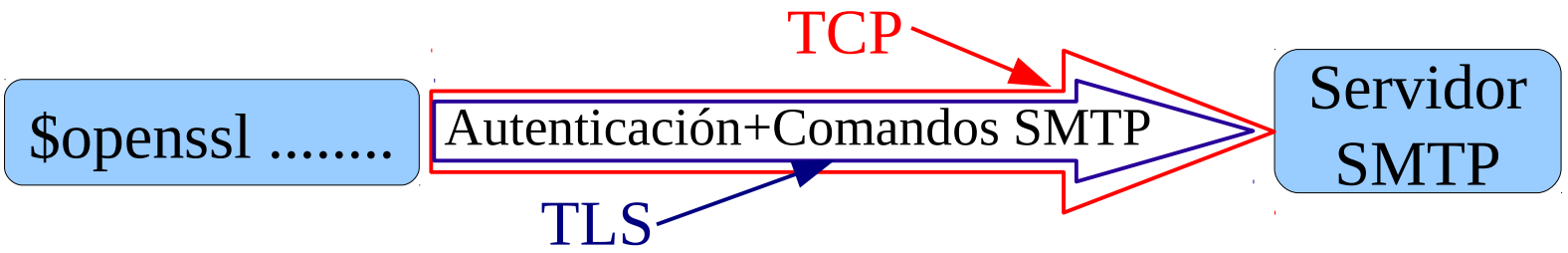
Prueba de interacción SMTP (obsoleta)

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

Lo de arriba nos permitía enviar correo sin usar el 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.

Enviando correo a mano usando gmail

- Para abrir la conexión, en lugar de telnet, usamos:

  $openssl s_client -starttls smtp -crlf -connect smtp.gmail.com:587

- Luego enviamos cuenta de correo y password codificados en formato de 7 bits.
- Para codificar la cuenta y su password en base64 podemos usar:

  http://www.motobit.com/util/base64-decoder-encoder.asp

  - Pero no es seguro pues ellos sabrán correo y password.
- Ahora recién podemos enviar los comandos SMTP para enviar el correo.
- Esto se ve a continuación:

Esto funcionó en algún momento. Las normas de seguridad evolucionan. Hoy puede no funcionar, pero sería algo similar.

Enviando correo vía gmail: Comandos

```
Script started on Wed 21 Apr 2010 10:04:24 PM CLT
agustin@agustin-laptop:~$ perl -MMIME::Base64 -e 'print
encode_base64("\000elo322utfsm\@gmail.com\000elo322_USM")'
AGFndXN0aW4uai5nb256YWxlekBnbWFpbC5jb20AZWxvMzIyXzIwMTA=
agustin@agustin-laptop:~$ openssl s_client -starttls smtp -crlf -connect smtp.gmail.com:587
CONNECTED(00000003)
depth=0 /C=US/ST=California/L=Mountain View/O=Google Inc/CN=smtp.gmail.com
verify error:num=20:unable to get local issuer certificate
verify return:1
depth=0 /C=US/ST=California/L=Mountain View/O=Google Inc/CN=smtp.gmail.com
verify error:num=27:certificate not trusted
verify return:1
depth=0 /C=US/ST=California/L=Mountain View/O=Google Inc/CN=smtp.gmail.com
verify error:num=21:unable to verify the first certificate
verify return:1
---
Certificate chain
 0 s:/C=US/ST=California/L=Mountain View/O=Google Inc/CN=smtp.gmail.com
   i:/C=ZA/ST=Western Cape/L=Cape Town/O=Thawte Consulting cc/OU=Certification Services Division/CN=Thawte
Premium Server CA/emailAddress=premium-server@thawte.com
---
```

Codificación Base 64

Conexión al servidor

Texto desde el servidor

Enviando correo vía gmail: comandos

Server certificate

-----BEGIN CERTIFICATE-----

MIIDYzCCAsygAwIBAgIQUR2EgGT4+hGKEhCgLMX2sjANBgkqhkiG9w0BAQUFADCB

zjELMAkGA1UEBhMCWkExFTATBgNVBAgTDFdlc3Rlcm4gQ2FwZTESMBAGA1UEBxMJ

Q2FwZSBUb3duMR0wGwYDVQQKExRUaGF3dGUgQ29uc3VsdGluZyBjYzEoMCYGA1UE

CxMfQ2VydGlmaWNhdGlvbiBTZXJ2aWNlcyBEaXZpc2lvbjEhMB8GA1UEAxMYVGhh

d3RlIFByZW1pdW0gU2VydmVyIENBMSgwJgYJKoZIhvcNAQkBFhlwcmVtaXVtLXNl

cnZlckB0aGF3dGUuY29tMB4XDTA3MDczMDAwMDAwMFoXDTEwMDcyOTIzNTk1OVow

aDELMAkGA1UEBhMCVVMxEzARBgNVBAgTCkNhbGlmb3JuaWExFjAUBgNVBAcTDU1v

dW50YWluIFZpZXcxEzARBgNVBAoTCkdvb2dsZSBJbmMxFzAVBgNVBAMTDnNtdHAu

Z21haWwuY29tMIGfMA0GCSqGSIb3DQEBAQUAA4GNADCBiQKBgQD+RiG+G3Mo9Q9C

tcwDjpp6dJGifjiR5M2DbEbrsIOlth80nk5A7xstKCUfKobHkf/G9Y/DO24JP5yT

s3hWep05ybyiCmOzGL5K0zy3jIq0vOWy+4pLv2GsDjYi9mQBhobAAx3z38tTrTL+

WF4p0/Kl014+wnukIpj4MdF35rIkgQIDAQABo4GmMIGjMB0GA1UdJQQWMBQGCCsG

AQUFBwMBBggrBgEFBQcDAjBABgNVHR8EOTA3MDWgM6Axhi9odHRwOi8vY3JsLnRo

YXd0ZS5jb20vVGhhd3RlUHJlbWl1bVNlcnZlckNBLmNybDAyBggrBgEFBQcBAQQm

MCQwIgYIKwYBBQUHMAGGFmh0dHA6Ly9vY3NwLnRoYXd0ZS5jb20wDAYDVR0TAQH/

BAIwADANBgkqhkiG9w0BAQUFAAOBgQBeNYOZwMVQ7bd6b4sueAkgm57Cyv2p1Xv1

52e8bLnWqd03mWgn/+TQtrwbE1E6pVuQaZJY33ILpt8IfzwVf2TGQI+M5yazZ2fC

xwArHo20iAss3MLQR8tDXWfBoH2Lk9BBsEKDRP4hp83yfpZgdY3pinHTCbqHpsiS

v97epiiFBA==

-----END CERTIFICATE-----

Enviando correo vía gmail (cont.)

subject=/C=US/ST=California/L=Mountain View/O=Google Inc/CN=smtp.gmail.com

issuer=/C=ZA/ST=Western Cape/L=Cape Town/O=Thawte Consulting cc/OU=Certification Services Division/CN=Thawte Premium Server CA/emailAddress=premium-server@thawte.com

---

No client certificate CA names sent

SSL handshake has read 1230 bytes and written 335 bytes

New, TLSv1/SSLv3, Cipher is RC4-MD5

Server public key is 1024 bit

Compression: NONE

Expansion: NONE

SSL-Session:

Protocol : TLSv1

Cipher : RC4-MD5

Session-ID: 48813969EF40970160190D9B1FB40388942A5CC55E97C10EAD31CFDE74E435A1

Session-ID-ctx:

Master-Key: 876D3A355068325B07BDE5BD23243E6293AFEDD421395C31E0F00B4ACED7AF63F6B3ED8CBABF203E0C5397B4260AAE2B

Key-Arg : None

Start Time: 1271902106

Timeout : 300 (sec)

Verify return code: 21 (unable to verify the first certificate)

Enviando correo vía gmail (cont.)

Servidor espera autenticación

```
250 PIPELINING
AUTH PLAIN AGFndXN0aW4uai5nb256YWxlekBnbWFpbC5jb20AZWxvMzIyXzIwMTA=
235 2.7.0 Accepted
mail from: <elo322utfsm@gmail.com>
250 2.1.0 OK 20sm535661ywh.15
rcpt to: <agustin.gonzalez@usm.cl>
250 2.1.5 OK 20sm535661ywh.15
data
354  Go ahead 20sm535661ywh.15
From: Batmann <batman@rectoria.usm.cl>
To: SuperMan <superman@cualquier.cosa.cl>
Subject: Ejemplo de envio de correo via gmail server.
Hola Muchachos!
UN saludo cordial del profe,
Agustin
.
250 2.0.0 OK 1271902394 20sm535661ywh.15
quit
221 2.0.0 closing connection 20sm535661ywh.15
read:errno=0
agustin@agustin-laptop:~$ exit
exit
Script done on Wed 21 Apr 2010 10:13:35 PM CLT
```

Protocolo SMTP

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)

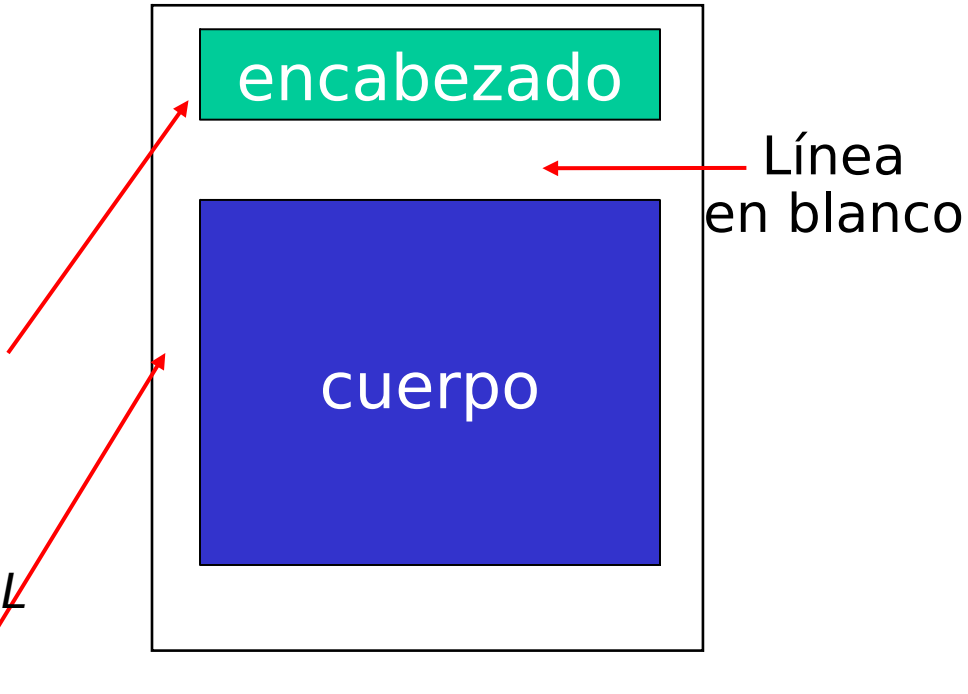
SMTP: protocolo para intercambio de mensajes de correo

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 +/=

Versión MIME

Método de codificación usado

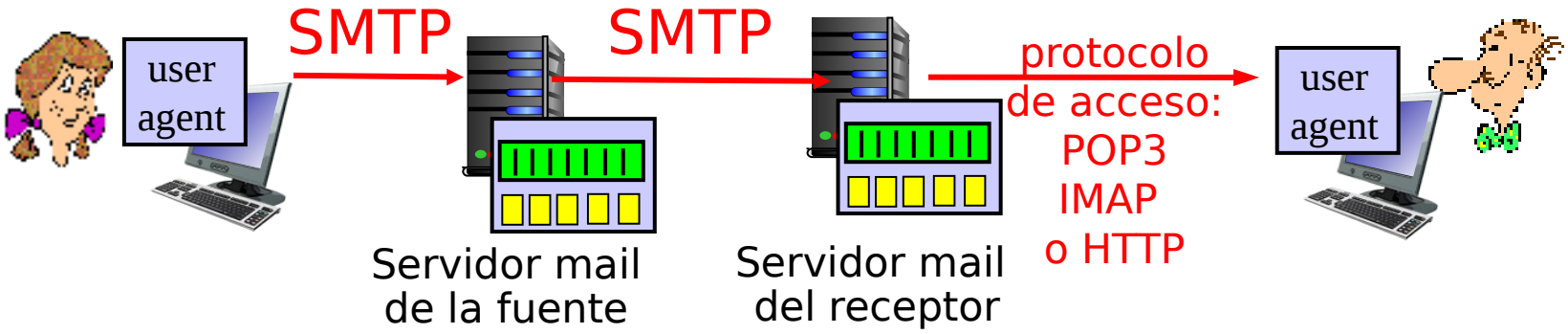
Tipo datos multimedia, subtipo, declaración de parámetros

Datos binarios codificados en base64

```
From: alice@crepes.fr
To: bob@hamburger.edu
Subject: Picture of yummy crepe.
MIME-Version: 1.0
Content-Transfer-Encoding: base64
Content-Type: image/jpeg

base64 encoded data .....
.........................
......base64 encoded data
```

Protocolos de acceso de correo



- SMTP: permite envío y almacenamiento de correo en servidor del destinatario
- Protocolo de acceso a correo: permite extraer correo desde el servidor destinatario
  - POP: Post Office Protocol [RFC 1939]
    - autenticación (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
- “bajada y conserva”: obtiene copia de los mensajes en diferentes clientes.
- POP3 no mantiene el estado de una sesión a otra (“stateless”)

IMAP

- Puede mantener 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 y lanzaron la primera versión 7 Meses más tarde (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.

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