

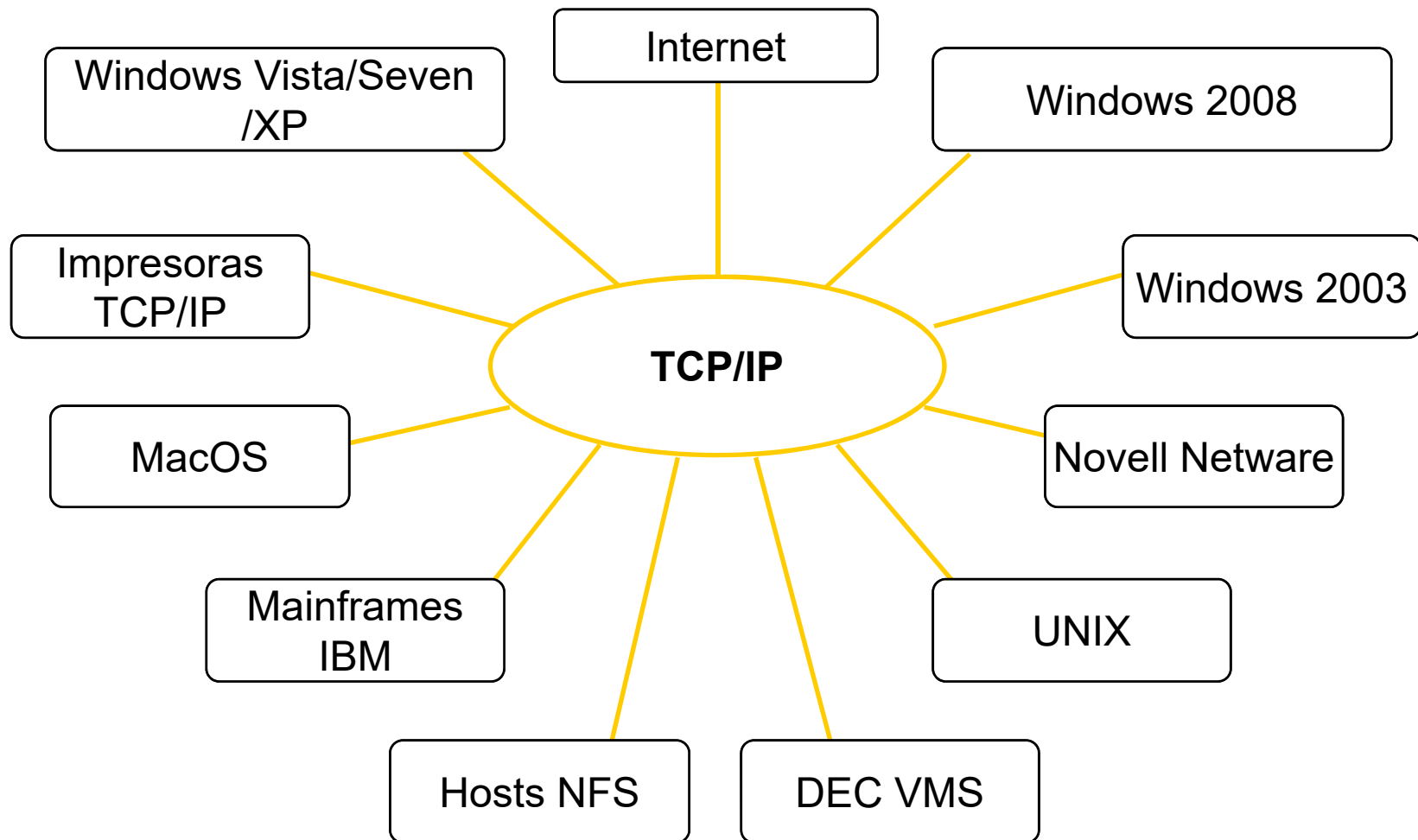
# **Aplicaciones TCP/IP**

# Objetivos



- Estudiar las aplicaciones tradicionales de TCP/IP mas difundidas:
  - Telnet
  - FTP
  - SMTP
  - HTTP

# Conectividad entre Ambientes Heterogeneos con TCP/IP



# Telnet



- Aplicación de Internet más antigua
- Publicada originalmente bajo RFC 97
  - "First Cut at a Proposed Telnet Protocol," Feb. 1971
- 1983 forma final publicada en RFC 854 y RFC 855
- Hoy en día es una aplicación que todavía se utiliza.
- Base para protocolos modernos como HTTP

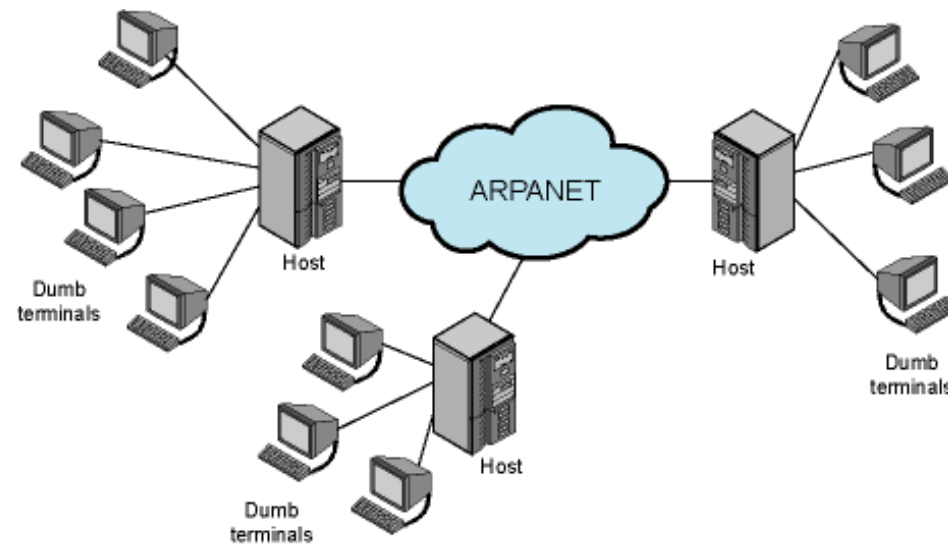
# Acceso remoto



- Acceso a través de terminales **bobos** (“Dumb terminals”)
  - Terminales compuesto por un teclado y una pantalla con interfase de comunicación limitados (normalmente RS-232)
  - “Stream” de caracteres de datos transmitidos en cada dirección.
- Los terminales “bobos” se conectan a host local, el cual se conecta con el host remoto.
- Usuarios locales pueden usar hosts remotos.
- Problema de conversiones de funcionalidades de terminales.

# Telnet: Modalidad de operación

## Original

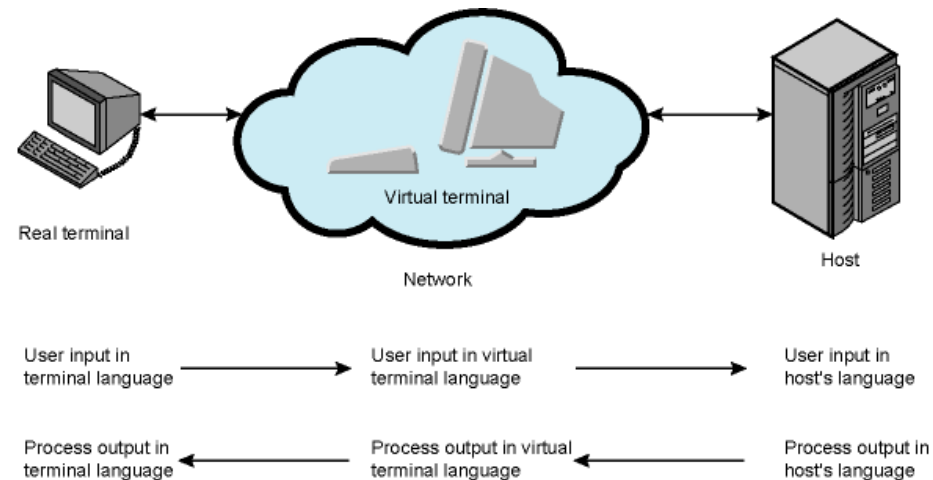


## Actual



# Network Virtual Terminals (NVT)

- Se desarrolló el Virtual terminal protocol (VTP)
- Transforma características de un terminal real en una forma estándar:
  - Network virtual terminal (NVT)
- Ambos extremos generan datos y señales de control en lenguaje nativo.
- Cada lado traduce estos datos en tráfico NVT



# Fases de Operación de VTP



- 4 Fases:
  - Establecimiento de Conexión
    - Se utiliza TCP
  - Negociación
    - Negociación de características de NVT
  - Control
    - Intercambio de información de control y comandos
  - Datos
    - Transferencia de datos
- Para telnet, tanto los datos como control, forman parte del mismo del mismo stream.
  - El "Interpret as Command" (IAC) es un carácter (255=FF) que precede a cualquier comando de telnet.



# File Transfer Protocol (FTP)



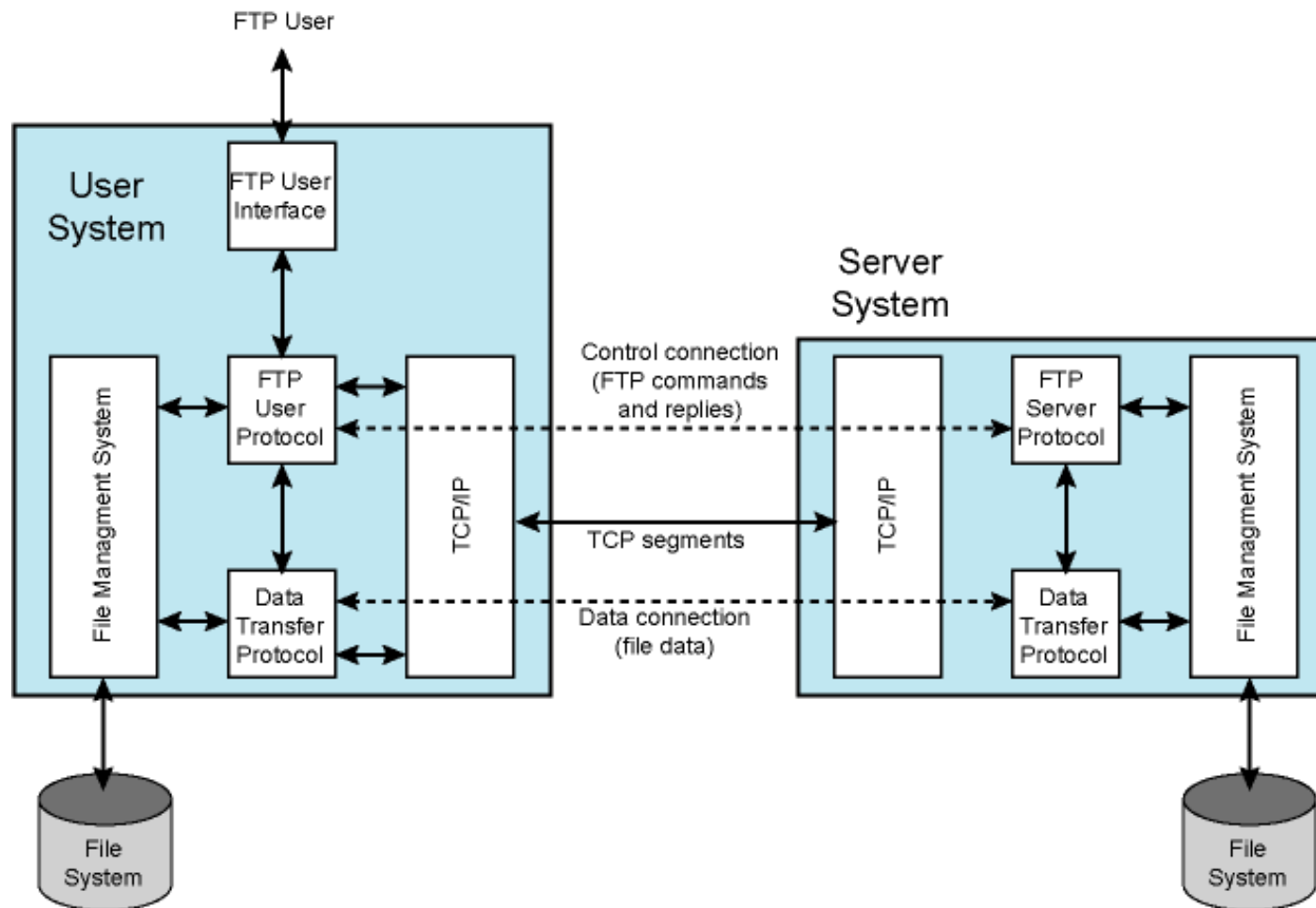
- FTP evolucionó de una época de sistemas diversos, así posee comandos, modos de transferencias y representación de datos.
- RFC 959 (1985) (RFC 795 '80 obsoleto)
- Objetivos:
  - Compartimiento de archivos
  - Comparación de diversas formas de almacenar archivos entre sistemas
  - Transferencia de datos confiable y eficiente.
- FTP trabaja con **File systems** en lugar de archivos independientes
  - TFTP trabaja con archivos independientes: colección de bits con nombre.
  - FTP trabaja con metadatos tales como nombres completos de archivos (pathname), control de acceso, directorios y subdirectorios, etc.

# Modelo de FTP



- Modelo cliente servidor como en Telnet
- Host de inicio: Cliente
  - Elige nombre de archivo y opciones.
- Server acepta o rechaza el request.
  - Basado en protección de sistema de archivo y opciones
  - Si acepta, servidor es responsable para **establecer** y manejar la transferencia
- FTP opera en dos niveles
  - Establece conexión FTP para intercambio de información de control (desde el cliente al servidor)
  - Luego de aceptada dicha conexión, se establece otra conexión para transferencia de datos (desde el **servidor al cliente**)

# Modelo FTP

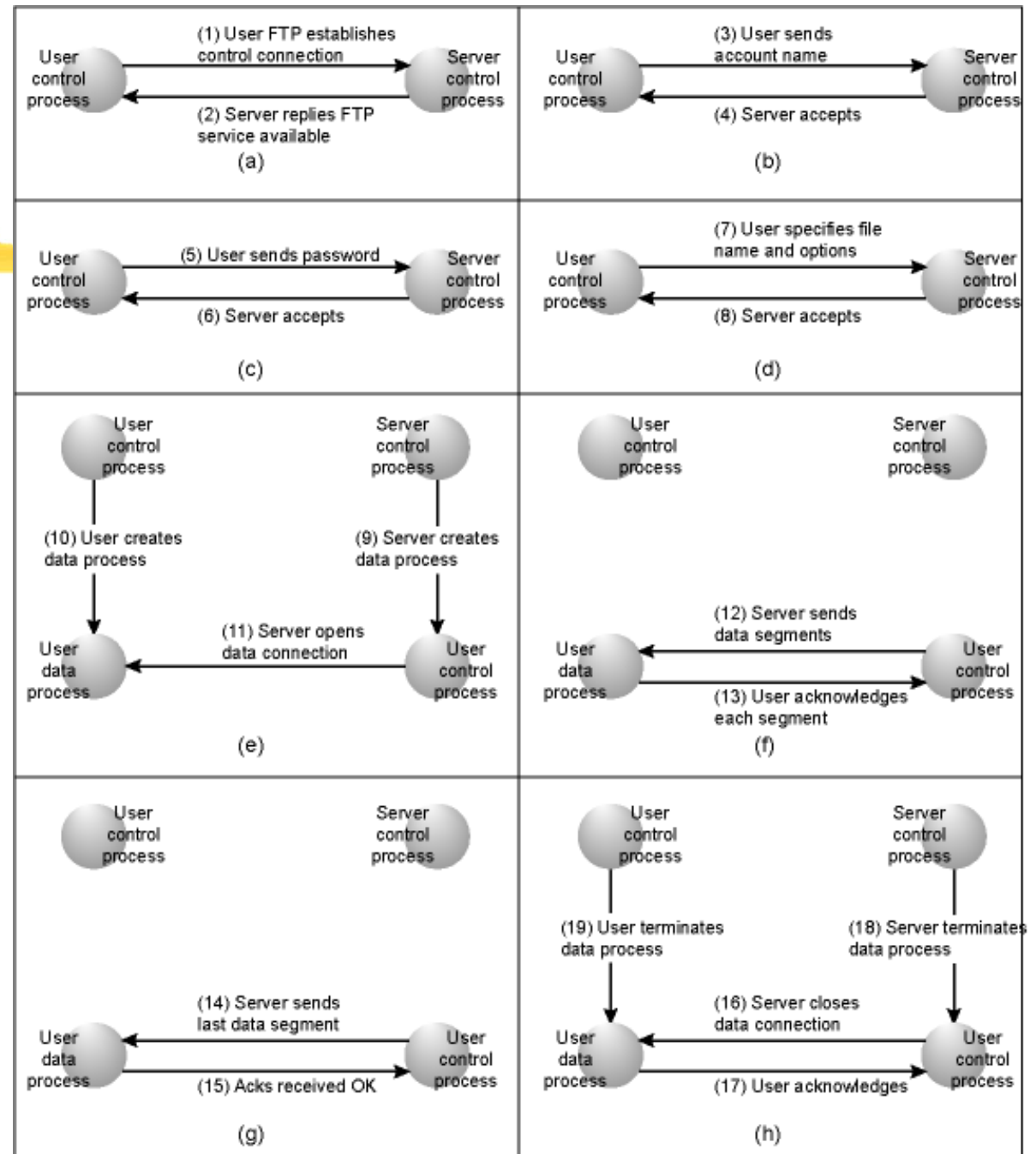


# Comandos FTP

- Especifican parámetros para la **conexión** de datos y la naturaleza de la **operación**
  - **Conexión** como: Modo de transferencia, tipo de representación, estructura
  - **Operación** como: Almacenar, extraer, borrar, etc.
- Ejemplos de comandos (internos):
  - USER {Control de Acceso}
  - PASS {Control de Acceso}
  - PORT host\_port {Parámetros de Transferencia}
  - TYPE type-code {Par. De Transferencia}
  - RETR pathname (server->client) {Servicio}
  - STOR pathname (client->server) {Servicio}
  - CWD pathname {Servicio}
  - LIST {Servicio}
  - NOOP {Servicio}

# Transferencia de datos con FTP

- Ejemplo: Se muestra una transferencia básica de archivo entre un servidor y un cliente.
- Se asume que no se presentan errores.
- No se detallan especificaciones de opciones.
- Observar como la conexión de datos es independiente de la control, abriéndose y cerrándose solo en el momento de transferencia de datos.



# Opciones FTP



- Para soportar hosts y sistemas operativos en general, FTP puede negociar opciones en tres dimensiones:
  - Tipo de Datos ("Datatype")
    - ASCII, EBCDIC, Imagen (bin) o "data byte size"
    - Más usado: **ASCII y Bin**
  - Tipo de Archivos ("Filetype")
    - Estructura de archivos, Estructura de registros y Estructura de Página.
    - Más usado: **Estructura de archivo (prácticamente el único)**
  - Modo de Transferencia
    - "Stream", bloque y comprimido.
    - Más usado: **"stream" (default)**
- Es responsabilidad del usuario especificar estas opciones.

# Operando FTP



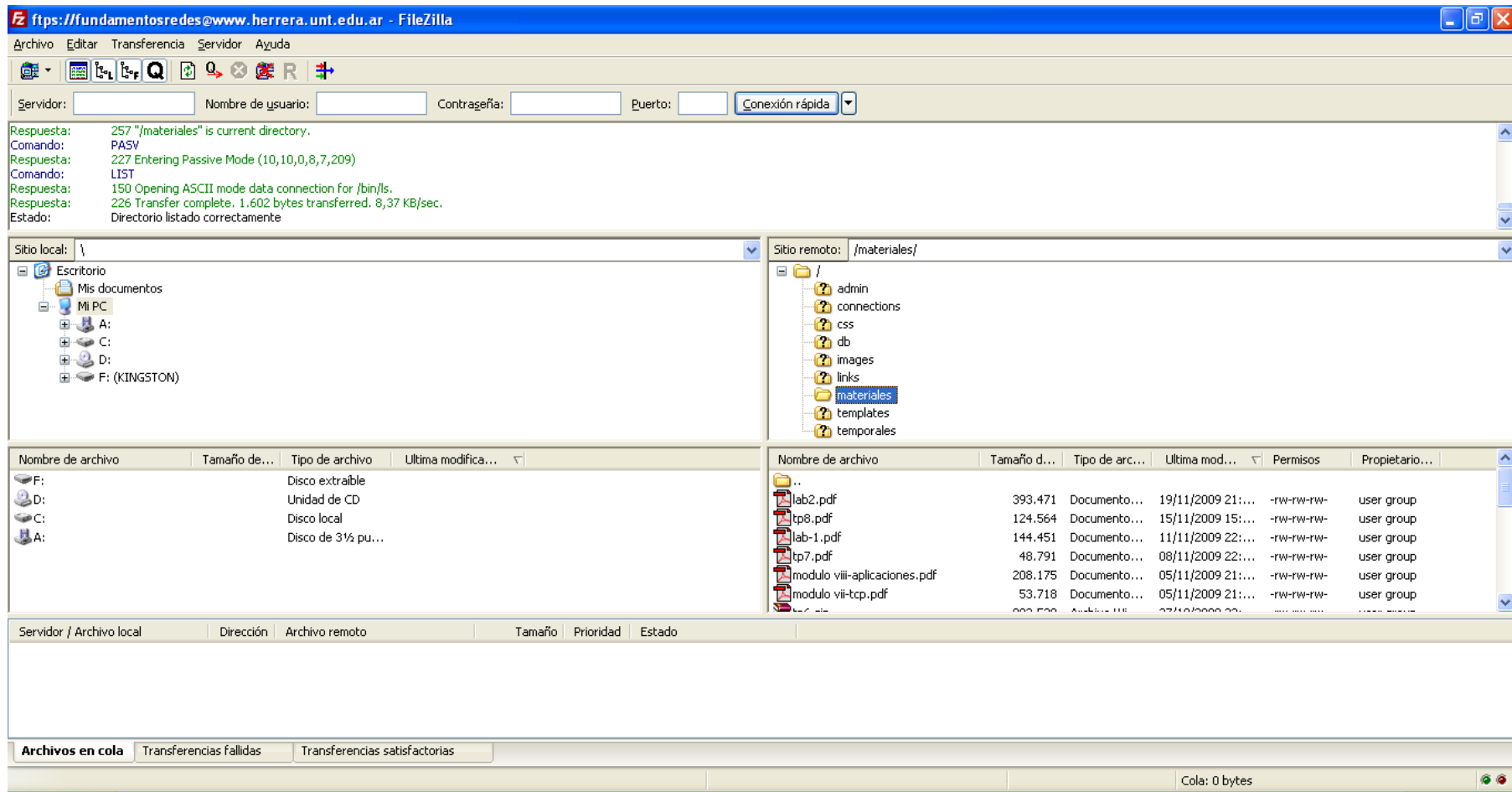
- FTP ofrece una interface de usuario
- Oculta los comandos **internos** de FTP de los comandos a nivel usuario.
- Puede ser:
  - Interface de línea de comando
  - Interface gráfica
- Ejemplos de Comandos:
  - User <user\_name> - Logueo con un nuevo usuario. Se solicita contraseña.
  - ascii – Configura ascii data type para las transferencias subsecuentes.
  - bin – Transferencias binarias
  - Type <data\_type> - configura/muestra tipo de datos para transferencia. (type ascii = ascii)
  - Cd – Cambia de directorio
  - Get <file\_name> [<dest file\_name>] – Transfiere un archivo del servidor al cliente.
  - Put <file\_name> [<dest file\_name>] – Transfiere un archivo del cliente al servidor.
  - Close – Cierra una sesion ftp. Usuario queda logueado
  - Quit, bye, exit – Cierra sesion ftp y abandona la interface.
- Alternativamente FTP puede ser usado con la notación URL e integrado a hipertexto (con un browser se puede utilizar).

# Ejemplo de Operación (Línea de Comando)

```
ftp> open www.herrera.unt.edu.ar
Conectado a www.herrera.unt.edu.ar.
220 Microsoft FTP Service
Usuario (www.herrera.unt.edu.ar:(none)): tcpip
331 Password required for tcpip.
Contraseña:
230 User tcpip logged in.
ftp> dir
200 PORT command successful.
150 Opening ASCII mode data connection for /bin/ls.
drwxrwxrwx   1 owner   group           0 Mar 11  0:40 links
-rwxrwxrwx   1 owner   group       12262 Mar 11 17:41 links.asp
-rwxrwxrwx   1 owner   group       18124 Apr 20 12:53 login.asp
drwxrwxrwx   1 owner   group           0 May  5 19:51 materiales
-rwxrwxrwx   1 owner   group       17959 Mar 11 17:41 materiales.asp
226 Transfer complete.
ftp: 2911 bytes recibidos en 0,01 segundos 194,07 a KB/s.
ftp> bin
200 Type set to I.
ftp> get login.asp
200 PORT command successful.
150 Opening BINARY mode data connection for login.asp(18124 bytes).
226 Transfer complete.
ftp: 18124 bytes recibidos en 0,03 segundos 584,65 a KB/s.
ftp> bye
C:>
```



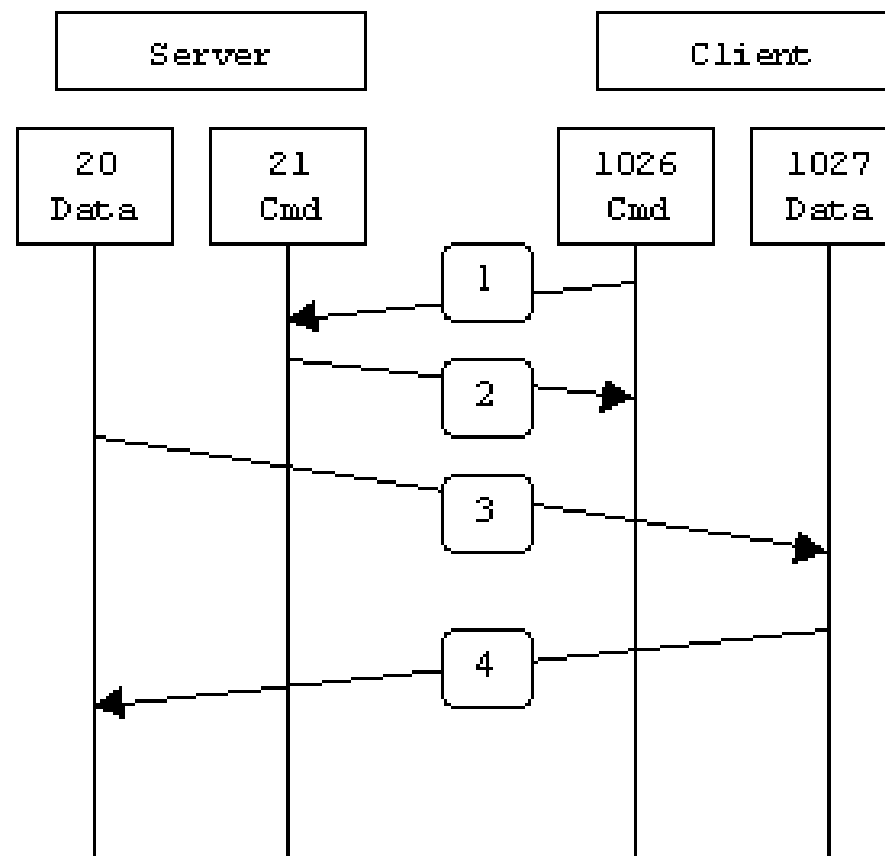
# Ejemplo de Interfaz Gráfica (Filezilla)



# FTP Activo y Pasivo

- Cada vez que se debe transmitir archivos o datos, se debe crear una conexión de datos.
- FTP especifica dos métodos para crear una conexión de datos:
  1. Método Activo de Conexiones de Datos (Normal o **Default**)
  2. Método Pasivo
- Activo:
  - Servidor inicia la conexión de datos (desde port 20)
  - Puerto Cliente: Efímero usado para la conexión de control
  - Normalmente cambia dicho puerto con comando **PORT** para evitar ciertas inconsistencias (va a tener dos puertos iguales abiertos).
- Pasivo:
  - El cliente le avisa al servidor que el va abrir una conexión hacia él
  - El servidor le deberá proveer la IP y Puerto sobre el cual se abrirá la conexión
- El modo Pasivo se utiliza por un tema de seguridad (Firewall puede bloquear conexiones entrantes)

# FTP Activo - Diagrama



# FTP Activo – Secuencia de Comandos

testbox1: {/home/p-t/slacker/public\_html} % **ftp -d testbox2 (-d: debugging)**

Connected to testbox2.slacksite.com.

220 testbox2.slacksite.com FTP server ready.

Name (testbox2:slacker): **slacker**

---> **USER slacker**

331 Password required for slacker.

Password: **TmpPass**

---> **PASS XXXX**

230 User slacker logged in.

---> **SYST 215**

**UNIX Type: L8**

Remote system type is UNIX.


Using binary mode to transfer files.

ftp> **ls**

---> **PORT 192,168,150,80,14,178 (4 prim. Octetos: ip – 14\*256+178=3762 – port )**

200 PORT command successful.

# FTP Activo – Secuencia de Comandos



---> LIST

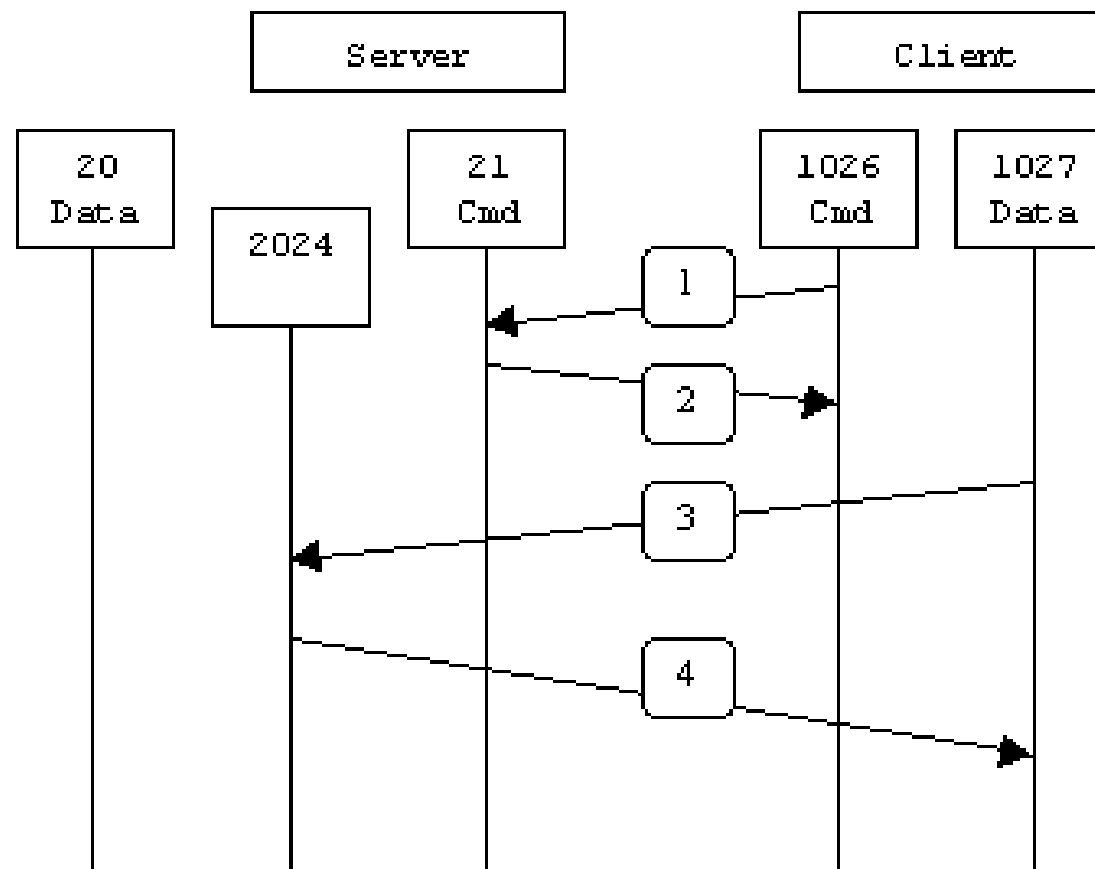
150 Opening ASCII mode data connection for file list.  
drwx----- 3 slacker users 104 Jul 27 01:45 public\_html  
226 Transfer complete.

ftp> **quit**


---> QUIT 221

Goodbye.

# FTP Pasivo - Diagrama



# FTP Pasivo – Secuencia de Comandos



```
testbox1: {/home/p-t/slacker/public_html} % ftp -d testbox2
```

```
Connected to testbox2.slacksite.com.
```

```
220 testbox2.slacksite.com FTP server ready.
```

```
Name (testbox2:slacker): slacker
```

```
---> USER slacker
```

```
331 Password required for slacker.
```

```
Password: TmpPass
```

```
---> PASS XXXX
```

```
230 User slacker logged in.
```


```
---> SYST
```

```
215 UNIX Type: L8
```

```
Remote system type is UNIX.
```

```
Using binary mode to transfer files.
```

# FTP Pasivo – Secuencia de Comandos



ftp> **passive** {Ejecutado por el cliente}

Passive mode on.

ftp> **ls**

---> PASV {cliente envía comando PASV y recibe línea abajo del servidor}

227 Entering Passive Mode (192,168,150,90,195,149). {server espera conexión en el port 50069 }

---> LIST

150 Opening ASCII mode data connection for file list

drwx----- 3 slacker users 104 Jul 27 01:45 public\_html

226 Transfer complete.

ftp> **quit**

---> QUIT

221 Goodbye.



# Correo Electrónico (email)



- Email en TCP/IP no está implementado como un único protocolo o aplicación.
- Es un conjunto de componentes relacionados que definen:
  - Protocolos de comunicación de correos
    - SMTP – POP - IMAP
  - Especificaciones de Direcciones
    - Relacionado con DNS
  - Especificaciones de Formato de Mensajes
    - RFC 822 - MIME

# Proceso de Comunicación



- Consiste de cinco etapas.
  1. Composición del Mail
  2. Entrega del Mail al Servidor
  3. Transmisión del Mail desde el Servidor de Mail
  4. Recepción del Mail por parte del Servidor (remoto)
  5. Acceso al Servidor Mail y recuperación del Mail.
- En algunos casos no se realizan todos los pasos, por ejemplo si el usuario que envía el mail esta cargado en el servidor, el paso 2 es omitido

# 1. Composición del Mail



- Usuario crea un mensaje
- El mensaje tiene un formato estándar definido por RFC 822 (ASCII) y MIME.
- Todo mensaje tiene un **"header"** y un **"body"**
  - "Header": Contiene datos que describen el mensaje y controlan el proceso de entrega y comunicación del mensaje.
    - Debe especificar la dirección (o direcciones) destino.
    - Equivalente al sobre de un correo común
  - "Body": Contiene el mensaje propiamente dicho.
    - Equivalente a la carta propiamente dicha en un correo común.

## 2. Entrega del Mail al Servidor



- La aplicación de correo electrónico, es diferente de otras aplicaciones en el sentido que el Tx y Rx no necesariamente están conectados simultáneamente en la red.
- El sistema está diseñado para que el usuario cree el mensaje y lo deposite en un servidor para su entrega.
- Para este proceso se utiliza el protocolo SMTP (Simple Mail Transfer Protocol)
- Esto es similar a depositar una carta en una oficina de correo.

### 3. Entrega del Correo (desde el servidor)



- El email es aceptado en el servidor (local) SMTP para ser transmitido a través del sistema de mail, al servidor remoto.
- Este proceso de entrega, necesita los servicios de DNS para resolver la dirección del servidor SMTP remoto.
- Una vez resuelta la dirección, se establece una conexión entre los servidores SMTP y se transfiere el correo.
- Esto es similar al transporte de la carta desde la oficina de correo local a la oficina remota.

## 4. Recepción del Mail (Servidor Remoto)



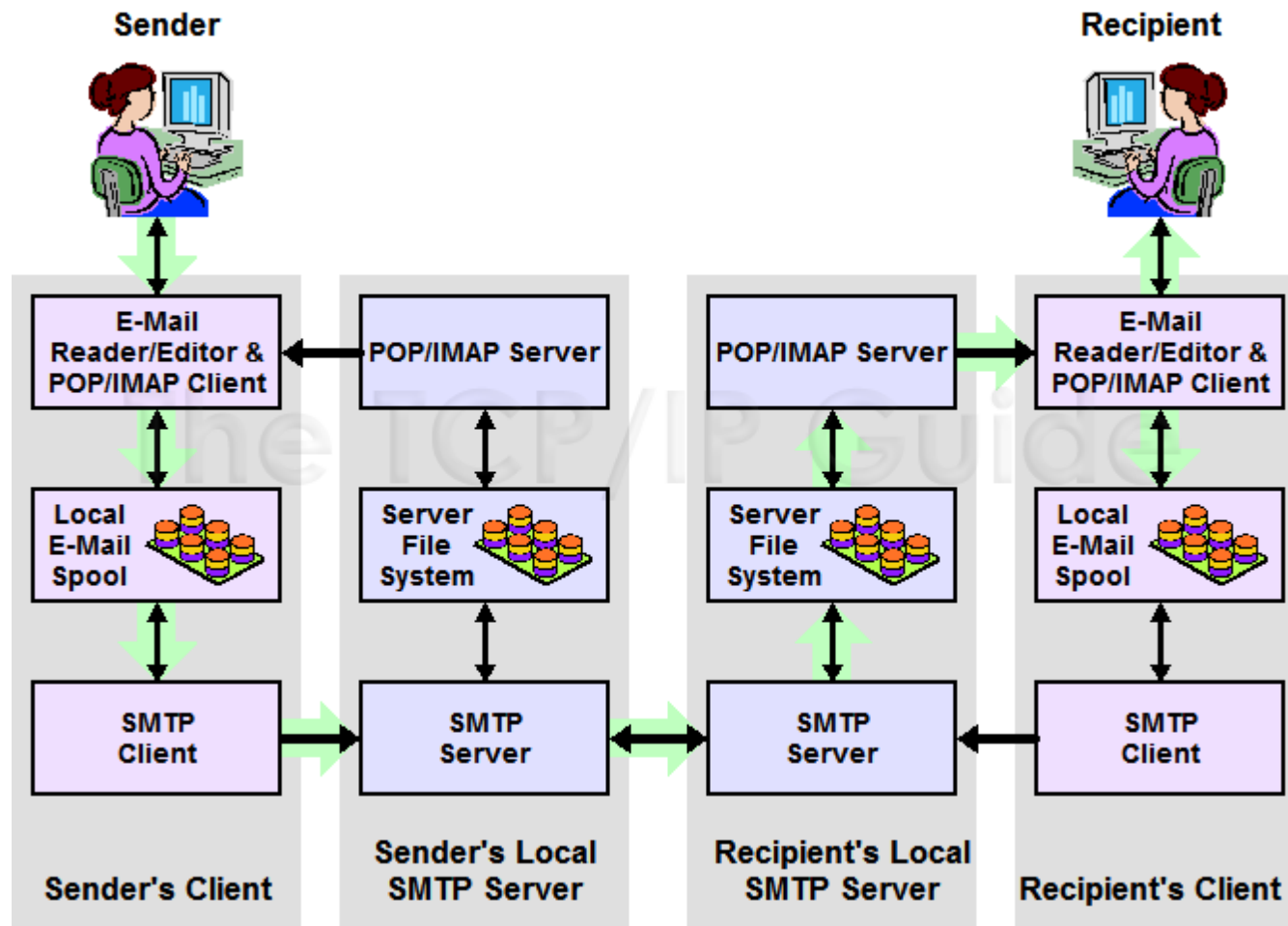
- El servidor SMTP acepta el mensaje y lo procesa.
- Ubica el mensaje en la casilla de correo del usuario, esperando que el usuario lo recoja de allí.
- Aquí termina la acción de SMTP (SMTP es solo responsable de entregar el correo en el servidor remoto)
- En la analogía con el sistema de correo común, esto es similar a cuando en la oficina de correo, se guarda la correspondencia en la casilla de correo o se acomoda para su reparto.

## 5. Recuperación del Mail



- El receptor chequea periódicamente con el servidor SMTP para ver si tiene correos nuevos.
- Si posee, lo recupera y puede leerlo.
- Para este proceso, no utiliza SMTP sino otro protocolo (IMAP, POP3)
- Este paso en la analogía sería la entrega propiamente dicha del correo al destinatario.

# Proceso de Comunicación con email





# Formatos del Mensaje



- El sistema de email trabaja con formatos de mensajes estándar para poder interpretarlos
- Existen dos formatos:
  - RFC 822
    - Utilizado para transporte de mensajes codificados en ASCII de 7 bits.
  - MIME (Multipurpose Internet Mail Extensions)
    - Utilizado para transporte de mensajes que transportan texto de ASCII 8 bits, texto enriquecido, imágenes, videos, applets, etc.
- En ambos casos, el mensaje está compuesto por el “header” y el “body”.
- SMTP normalmente trabaja con el “header” de los mensajes excepto por dos condiciones que impone sobre el “body”
  - **El set de caracteres debe ser ASCII – 7 Bits**
  - SMTP adiciona información del camino recorrido por el mail al comienzo del mensaje (luego de finalizar el header)

# Formato RFC 822



- No utiliza formato binario (como el resto de los protocolos de la suite). Se utiliza ASCII (incluso para "Header")
- Cada línea termina con un CR y un LF (CRLF)
- Cada campo del **header** tiene la forma:
  - <Header Name>:<Header Value>
- Ventaja:
  - Hace que el header sea legible
  - Flexibilidad
- Al finalizar el "header" (con todos sus campos), se debe insertar una línea **vacía** (CRLF)
- El texto a continuación constituye el **cuerpo** del mensaje.
- Cada línea del header o cuerpo, no debe poseer más de **1000** caracteres (incluyendo el CRLF) (se recomienda que sean líneas de 80 o menos caracteres para legibilidad)

# RFC 822: Campos del Header



- Divididos en varios grupos. Ejemplo:
  - Campo de Fecha ("Originator Date")
  - Campo de Remitente ("Originator Fields")
  - Campo de Destino(s) ("Destination Address Fields")
  - Campos de Identificación
  - Campos de Información
  - Etc.
- Algunos campos son obligatorios y otros opcionales
  - Ejemplos:
    - Date: (Obligatorio)
    - From: (Obligatorio)
    - Cc: (Opcional)
    - Subject: (Opcional – Normalmente presente)

# MIME



- Limitaciones que presenta formato RFC 822 (principalmente por usar ASCII 7 bits):
  - No se pueden transmitir archivos ejecutables o binarios
  - Texto con caracteres internacionales
  - Servidores SMTP pueden rechazar emails superiores a un tamaño máximo.
  - Problemas con gateways que traducen de ASCII a EBCDIC
- MIME resuelve estos problemas y aporta funcionalidades, **manteniendo compatibilidad con RFC 822.**
- Estandarizado por RFC 2045 al 2049.

# MIME (Multipurpose Internet Mail Exchange)



- La especificación MIME incluye:
  - Cinco campos de cabecera **nuevos**. Proveen información del **cuerpo** del mensaje.
    - MIME Version (1.0)
    - Tipo de Contenido ("content-type"): describe dato contenido en el cuerpo, para activar un agente apropiado en el cliente.
      - Por ejemplo, un mail que contiene un documento HTML puede contener un "content-type": tct/html o si contiene una imagen: image/jpeg
    - Codificación de Transferencia de Contenido ("content transfer-encoding"): convierte a ASCII-7
      - Con este campo el recipiente puede decodificar el contenido del mensaje a su representación normal.
    - ID del contenido ("content-ID"): identifica entidades MIME
      - Similar al Campo Message-ID de RFC 822, pero es específico para MIME.
    - Descripción del Contenido ("Content-Description"): descripción en texto del objeto.

# MIME (Continúa)



- Define formatos de contenidos
  - Se estandariza la representación de contenidos multimediales.
- Codificación de la Transferencia
  - Convierte el contenido del mensaje en una forma que se transmitan en ASCII de 7 Bits.

# SMTP (Simple Mail Transfer Protocol): RFCs



- Protocolo derivado de MTP (Mail Transport Protocol – RFC 722: Sept. 1980)
- SMTP está estandarizado por RFC 821 (1982).
- En 1993 se publica RFC 1425 ("SMTP Service Extension") y luego una serie de RFC relacionados (RFC 1651, 1869)
- En 2001 se publica RFC 2821 que es el que se utiliza actualmente.

# SMTP: Modo de Operación



- SMTP se utiliza para dos transferencias:
  - Desde el host del remitente al servidor SMTP del remitente
  - Desde dicho servidor SMTP al servidor SMTP del destinatario
- Originalmente SMTP utilizaba un mecanismo de “relay” para la comunicación con otros servidores SMTP.
- Actualmente, con la inclusión de registros DNS (MX), el ruteo se hace automático (RFC 2821).
- Un servidor SMTP que desea enviar un correo a otro, comienza haciendo un DNS “lookup” del registro MX del dominio que corresponde al nombre de dominio del destinatario.
- SMTP utiliza TCP para el transporte de correos, usando como puerto (servidor) al 25.



# RR MX (Mail Exchange) de DNS

- El Registro **MX** especifica un servidor de mail que manejará el tráfico de email para un dominio particular.
- Ejemplo de uso del RR MX:
  - Enviamos un mail a [ssaade@herrera.unt.edu.ar](mailto:ssaade@herrera.unt.edu.ar)
  - El cliente de email (SMTP) invoca al resolver (DNS) y al servidor DNS para realizar la resolución de herrera.unt.edu.ar
  - El servidor DNS configurado en el resolver buscará el servidor autoritativo para la zona herrera.unt.edu.ar (por ejemplo ns1.herrera.unt.edu.ar)
  - ns1.herrera.unt.edu.ar buscará el registro **MX** para la zona.
  - Se devolverá a **mail.herrera.unt.edu.ar** (suponiendo que este es el contenido del registro MX) como servidor de mail para la zona y también se enviará su dirección IP (sección **adicional** de DNS).
  - Es posible especificar múltiples registros MX para un dominio, cada uno apuntando a un servidor de mail. De esta forma se ofrece redundancia. DNS permite incluir **prioridades** (o preferencias) entre los servidores de Mail.

# RR MX (Mail Exchange) de DNS

## ➤ Formato de MX RR:

➤ [dominio] IN MX [preferencia] mail\_server

## ➤ Ejemplo:

➤ herrera.unt.edu.ar. IN MX 0 mail.herrera.unt.edu.ar.

➤ mail.herrera.unt.edu.ar. IN A 138.15.25.27

➤ El valor de preferencia más bajo, denota un servidor de mail de Mayor Prioridad

# Transacciones SMTP



- Las transacciones de MAIL con SMTP consta de tres fases:
  - Inicio de la Transacción e Identificación del Sender
    - SMTP “Sender” le comunica al SMTP “receiver” que quiere enviar un mensaje.
    - Identifica al Remitente del Correo
  - Identificación del Destinatario
    - El “sender” indica al “receiver” la dirección de email del destinatario
  - Transferencia del Mensaje
    - Transferencia del mensaje en si en formato RFC 822 o MIME.
- Observar que la especificación del remitente y del destinatario, no se lo obtiene del mensaje en si.
  - Se gana en eficiencia (no se debe procesar el mensaje para el proceso de envío del mismo, es más el SMTP Receiver puede aceptar o no el mensaje antes de ser enviado)
  - Mayor control en como distribuir los mensajes (por ejemplo inicio de dos sesiones SMTP en servidores diferentes debido a dos destinatarios de dos dominios diferentes).
  - Es mas simple implementar seguridad

# SMTP: Modo de Operación



- Operación de SMTP basada en serie de comandos y respuestas entre SMTP "sender" y SMTP "receiver" (en RFC 2821 se habla de "client" / "server")
- La iniciativa la toma el "sender" (Tx)
  - Establece una conexión TCP
  - Luego envía comandos al receptor
  - Ejemplo:
    - HELO<SP><domain><CRLF>
    - Este comando envía la identificación del Sender
- Cada comando del "sender" genera exactamente una respuesta del "Receiver" (Rx). Esta respuesta comienza con un código de 3 dígitos
  - Ejemplo: 250 (acción de Mail solicitada OK; completada)

# SMTP: Comandos

- Los comandos SMTP consisten de una única línea de texto, que comienza con un código de 4 letras, seguido en algunos casos por campo de argumento:
  - Ejemplos:
    - **HELO**<SP><domain><CRLF>
      - Envío de identificación del Sender
    - **MAIL**<SP>FROM:<reverse-path><CRLF>
      - Identifica originador del mail.
    - **RCPT**<SP>TO:<forward-path><CRLF>
    - **DATA**<CRLF>
      - Transferencia de mensaje (texto)
  - Referencias:
    - SP: Space - CRLF: Carriage Return y Line Feed

# SMTP: Respuestas



- Cada respuesta comienza con un código de tres dígitos.
- El **primer** dígito del código indica la categoría de la respuesta:
  - Dígito 1: "Preliminary Positive Response":
    - Si servidor inició la acción solicitada, espere otra respuesta antes de continuar con un comando nuevo.
    - Ejemplo (FTP): 150 Opening ASCII mode data connection for /bin/l
  - Dígito 2: "Positive Completion Reply"
    - Comando aceptado – Se realizó la acción.
    - Ejemplo: 220: (Servicio ready)
  - Dígito 3: "Positive Intermediate Reply"
    - Se acepta el comando, pero se solicita comandos adicionales.
    - Ejemplo: Comando DATA (luego del comando aceptado, se solicita que se envíen datos adicionales, que es el DATA ppiamente dicho)
  - Dígito 4: "Transient Negative Completion Reply"
    - Comando **no** aceptado. Situación de error es temporaria y se puede solicitar la acción nuevamente.
    - Ej.: 451 (error local en procesamiento del servidor SMTP) ó 452 ("Insufficient **System** Storage", Disco del Servidor SMTP lleno)
  - Dígito 5: "Permanent Negative Completion Reply"
    - Comando **no** aceptado. La acción no se llevó a cabo.
    - Ej.: 550 (No existe la casilla de correo especificada) ó 552 (Exceeded Storage allocation: Casilla de correo del **usuario** llena)

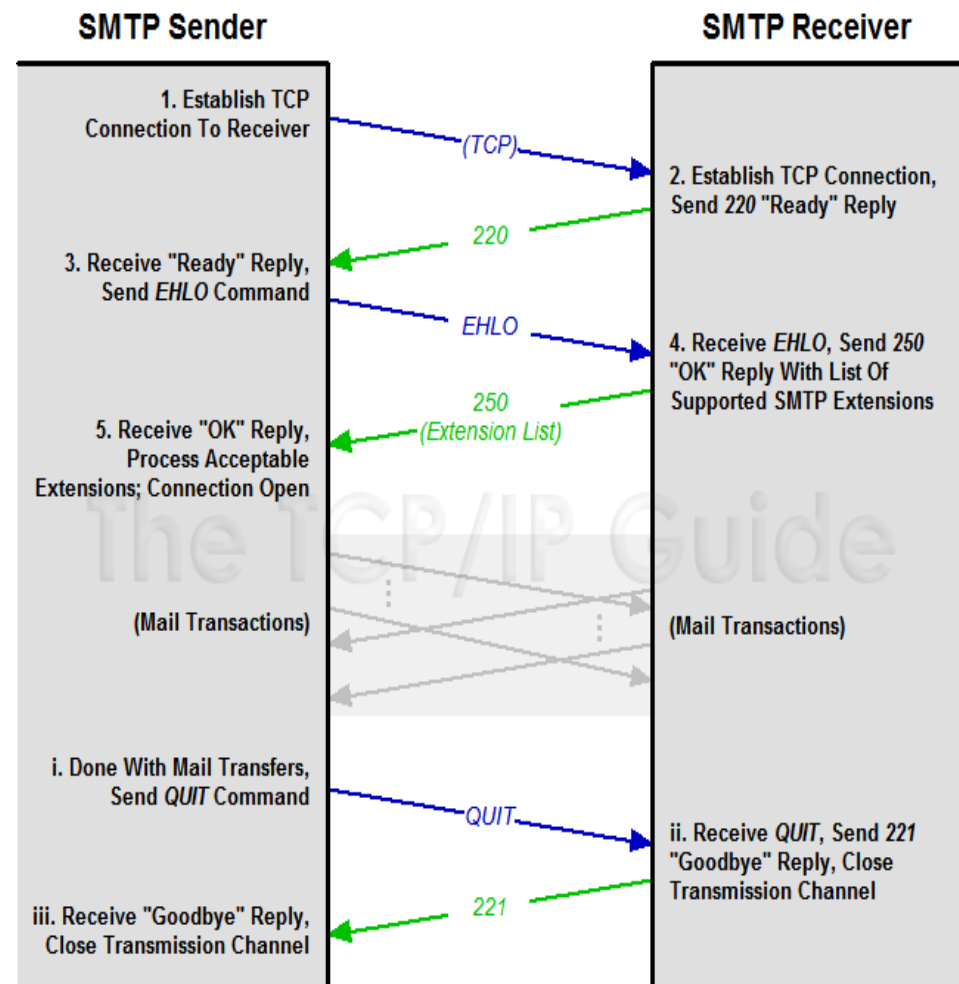
# SMTP: Modo de Operación.



- La operación básica de SMTP consiste de tres fases:
  - Establecimiento de la conexión
    - SMTP Sender establece conexión **TCP** cuando tiene uno o mas mensajes para enviar.
    - Luego se abre una conexión **SMTP**
  - Intercambio de uno o más comandos-respuestas
    - Consta mínimamente de 3 fases de comandos:
      - Comando MAIL (identifica al remitente)
      - Uno o más comandos RCPT (identifica los destinatarios)
      - Comando DATA (transfiere el mensaje)
  - Terminación de la Conexión.
    - Se cierra la conexión SMTP y luego la conexión TCP.

# SMTP: Modo de Operación

- Una vez establecida la conexión, TCP, SMTP Rx envía comando Ready (220)
- SMTP Tx envía comando EHLO "Extended Hello" (soporta SMTP Ext.)
- Si SMTP Rx soporta ext. envía OK sucesivamente con extensiones soportadas.
- Si SMTP Rx no soporta ext., rechaza comando EHLO
- Comienza la transacción de mails (Comandos MAIL, RCPT y DATA)
- Una vez finalizada, SMTP Tx envía comando Quit
- SMTP Rx contesta con un Goodbye.
- Se concluye la comunicación.





# SMTP: Ejemplo modo de operación

```
telnet mail.herrera.unt.edu.ar 25
220-herrera.unt.edu.ar ESMTP MDaemon 7.2.1; Tue, 21 Jun
    2005 10:27:40 -0300
220 All transactions and IP addresses are logged
EHLO herrera.unt.edu.ar
250-herrera.unt.edu.ar Hello herrera.unt.edu.ar, pleased
    to meet you
250-ETRN          #POSIBILIDAD DE SOLICITAR UN 2DO. MAIL SERVER
250-AUTH=LOGIN    #SOPORTE DE AUTENT. MEJORADA
250-AUTH LOGIN CRAM-MD5    #SOPORTE DE ENCRIP. LOGIN
250-8BITMIME      #SOPORTE DE MIME
250 SIZE 0        #DECLARACION DE TAMAÑO PREVIO A ENVIO DE MAIL
MAIL FROM:ssaade@herrera.unt.edu.ar
250 <ssaade@herrera.unt.edu.ar>, Sender ok
```

# SMTP: Ejemplo modo de operación

**RCPT TO:**infoiec@herrera.unt.edu.ar

250 <infoiec@herrera.unt.edu.ar>, Recipient ok

**RCPT TO:**ssaade@sistelco.com

250 <ssaade@sistelco.com>, Recipient ok

**DATA**

354 Enter mail, end with <CRLF>.<CRLF>

Prueba de Email enviado

Desde un cliente Telnet

Conectado por puerto 25 (SMTP) al servidor de herrera.

.

250 Ok, message saved <Message-ID: >

**QUIT**

# Acceso a "Mail Boxes"



- Según se vió un intercambio de emails involucra normalmente 4 dispositivos
  - Host del remitente (donde se compone el mail)
  - Servidor SMTP del remitente.
  - Servidor SMTP del destinatario.
  - Host del destinatario (donde se lee el mail)
- En los tres primeros dispositivos se utiliza como protocolo de comunicación SMTP
- En el cuarto se debe utilizar otro protocolo de acceso (es decir para recibir un correo del servidor SMTP).
- Esto es así porque:
  - SMTP fue diseñado para transporte de mail solamente, no para acceso a casillas de correos de clientes. Adicionar esa funcionalidad, haría más complejo el protocolo.
  - Se ofrece mayor flexibilidad en el método de recoger el correo (por ejemplo en clientes móviles, o dejando copias sobre el servidor).

# Modos de Recuperación de Mails



- Existen tres modelos:
  - Modelo de Acceso Online
    - Acceso continuo al mailbox del SMTP server.
    - Requiere acceso continuo a Internet
    - Clientes Unix (IMAP también soporta este modelo)
  - Modelo de Acceso Offline
    - Acceso al servidor SMTP donde reside el mail box
    - Descarga y borrado de correos
    - Lectura y procesamiento de mails en forma offline por parte del usuario.
    - POP
  - Modelo de Acceso Desconectado
    - Híbrido entre los dos modelos
    - Se bajan mails del servidor para su manipulación offline pero no se borran del servidor.
    - En el futuro el usuario se conecta y puede sincronizar cambios entre el mailbox local y el remoto.
    - IMAP

# Post Office Protocol (POP)



- POP (informalmente conocido como POP3) es un protocolo antiguo pero el más utilizado para obtener correos en forma offline.
- La versión usada actualmente (versión 3) está estandarizada por el RFC 1081 (1988), con algunas mejoras introducidas por RFC's posteriores
- Es un protocolo simple y eficiente
  - Cliente y servidor siguen tres fases durante la interacción
  - Número mínimo de comandos.
- POP3 trabaja sobre TCP.
- El servidor POP3 que ejecuta sobre el servidor SMTP donde residen las cuentas de correo, trabaja sobre el puerto **110**.

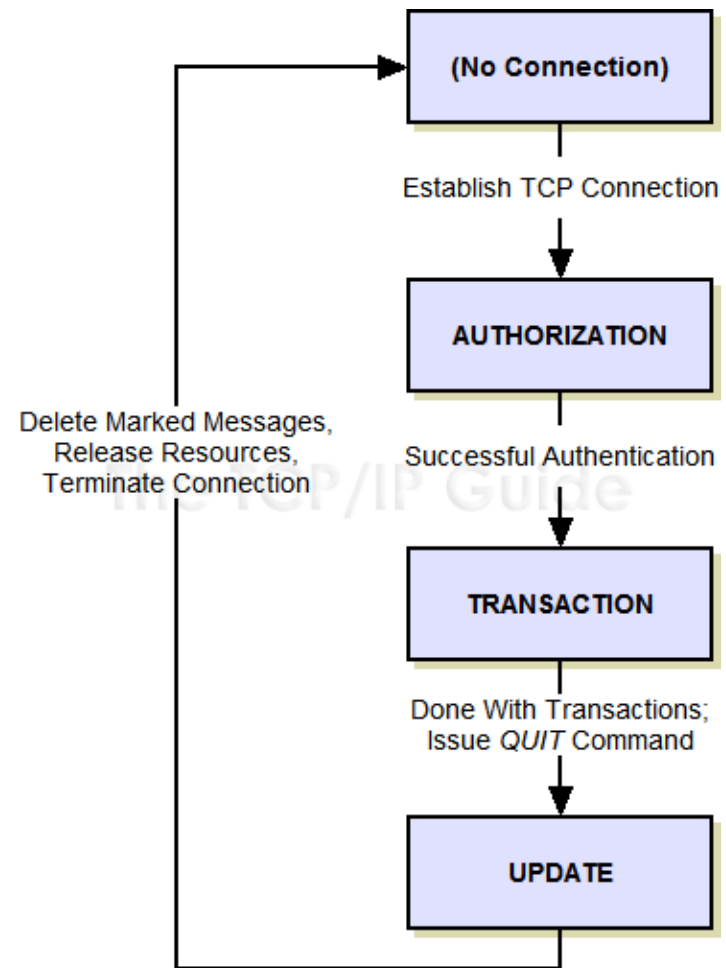
# Comandos y Respuestas POP



- Comandos POP3 son de 3 o 4 letras (ASCII) terminadas por un CRLF
- Las respuestas también son de texto, pero solamente usan las palabras claves:
  - +OK: Respuesta Positiva
  - -ERR: Respuesta Negativa
- Estos mensajes pueden ser acompañados por un mensaje explicativo (generalmente en el caso de -ERR)

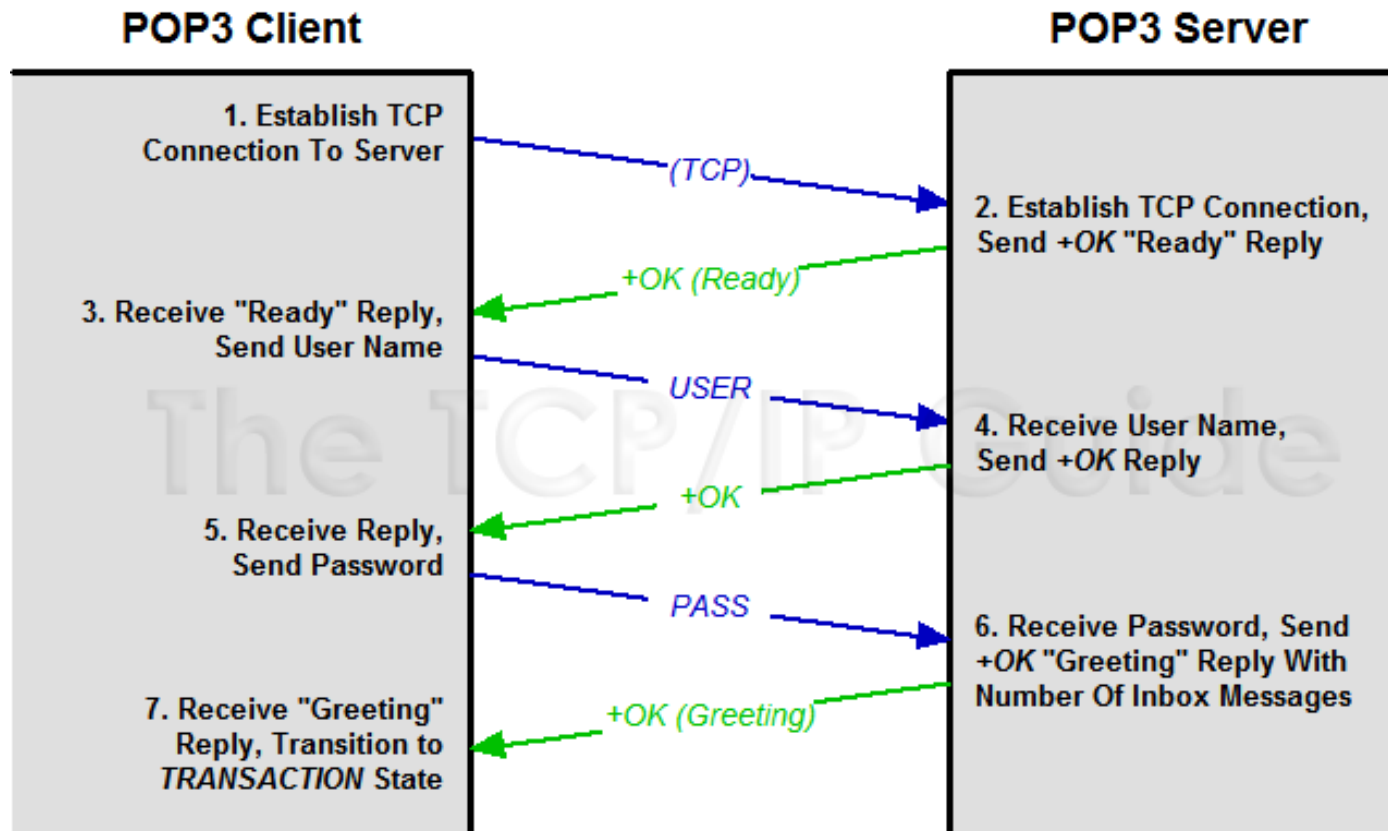
# Máquina de Estado Finito POP3

- Se observan tres estados:
  - **Autorización**: Servidor listo para recibir comandos. Cliente provee información de autenticación para permitir acceso a mailbox.
  - **Transacción**: Se permite realizar al cliente varias operaciones en el mailbox.
  - **Actualización**: Cuando el cliente termina, ejecuta el comando QUIT, la sesión entra a este estado, borrándose los mensajes indicados en el estado Transacción.
- Luego del estado "actualizar", se termina la conexión TCP



# Proceso de Autorización POP3

- Se comienza cuando el cliente inicia la conexión TCP con el servidor.
- Se envía USER & PASS – Clear Text
  - Existen esquemas más seguros, usando el comando APOP que usa encriptación MD5





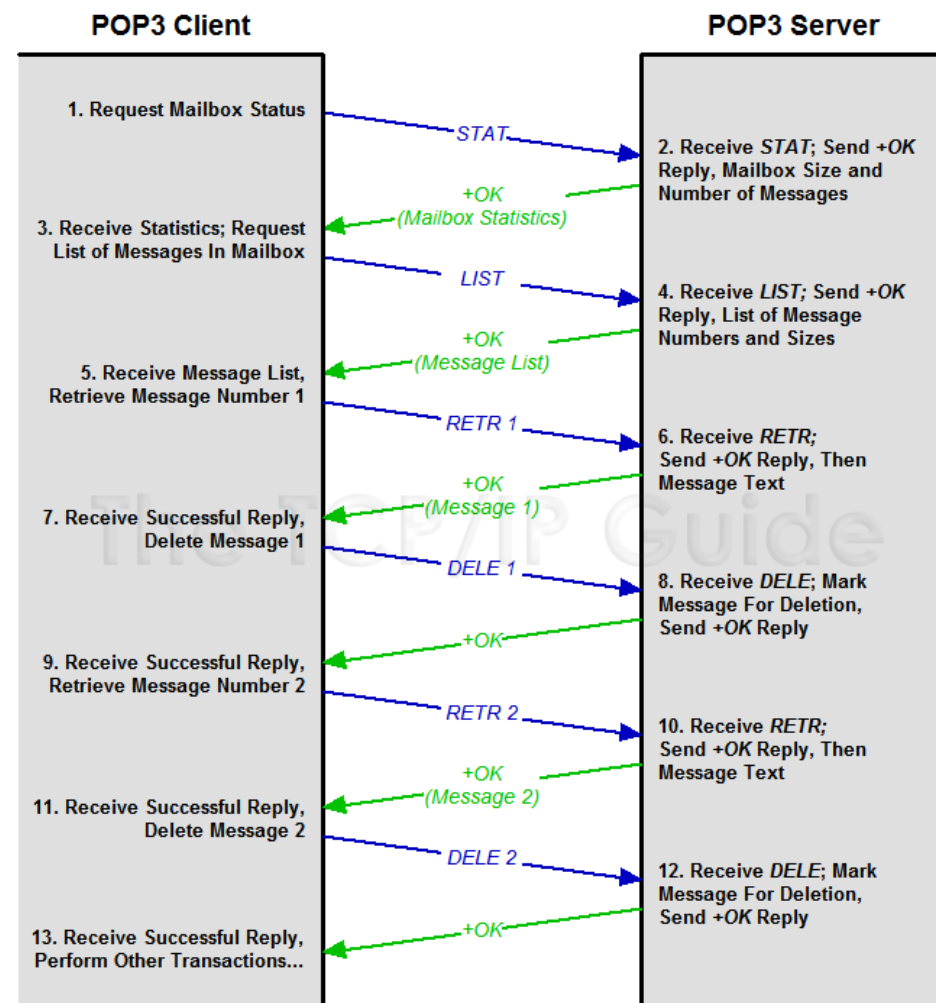
# Estado de Transacción



- La mayoría de los comandos POP3 son válidos en este estado.
- Algunos comandos son:
  - STAT: Estado del mailbox
  - LIST: Lista mensajes
  - RETR: Obtiene un dado mensaje del mail box
  - DELE: Marca un mensaje como borrado
  - ...

# Estado de Transacción

- La figura muestra las transacciones típicas entre un cliente y un servidor POP3
- El cliente envía STAT para conocer el número de mensajes en el mailbox
- Luego envía LIST para saber los números de mensajes a traer
- A continuación comienza a traer y luego borrar mensajes
- En algunos clientes POP3 se puede configurar para que no realice el DELE luego del RETR



# Estado de Actualización



- Se inicia cuando el cliente POP3 ejecuta el comando QUIT.
- Una de las acciones que realiza el servidor en este estado, es borrar todos los mensajes marcados (por el comando DELE)
- Una vez terminado, el servidor envía un ACK al cliente de la forma +OK o -ERR.
- Si se finaliza correctamente, se termina la conexión TCP
- El servidor POP3 normalmente implementa un timer de inactividad, en el cual se cierra la conexión TCP luego de 10 min de inactividad. Si esto ocurre (normalmente porque el ACK del servidor devolvió -ERR), no se borran mensajes del servidor y se cierra la conexión TCP.

# HTTP (Hypertext Transfer Protocol)



- HTTP/1.0 – 1996 (RFC 1945)
- HTTP/1.1 – 1999 (RFC 2616)
- HTTP está implementado bajo un esquema Cliente (“Browser” o Explorador) – Servidor (“Web Server”)
- Ejemplo de clientes: MS Explorer, Google Chrome, Mozilla Firefox
- Ejemplo de servidores: MS IIS, Apache
- El Servidor contiene documentos o páginas Web, clasificado como **hipermedial** (**hiper** porque puede referenciar a otro documento y **medial**, porque puede contener objetos no solo de texto (imagen, video, etc.))
- HTTP es un protocolo que define la estructura de los mensajes enviados entre ambas aplicaciones y como se realiza ese intercambio.

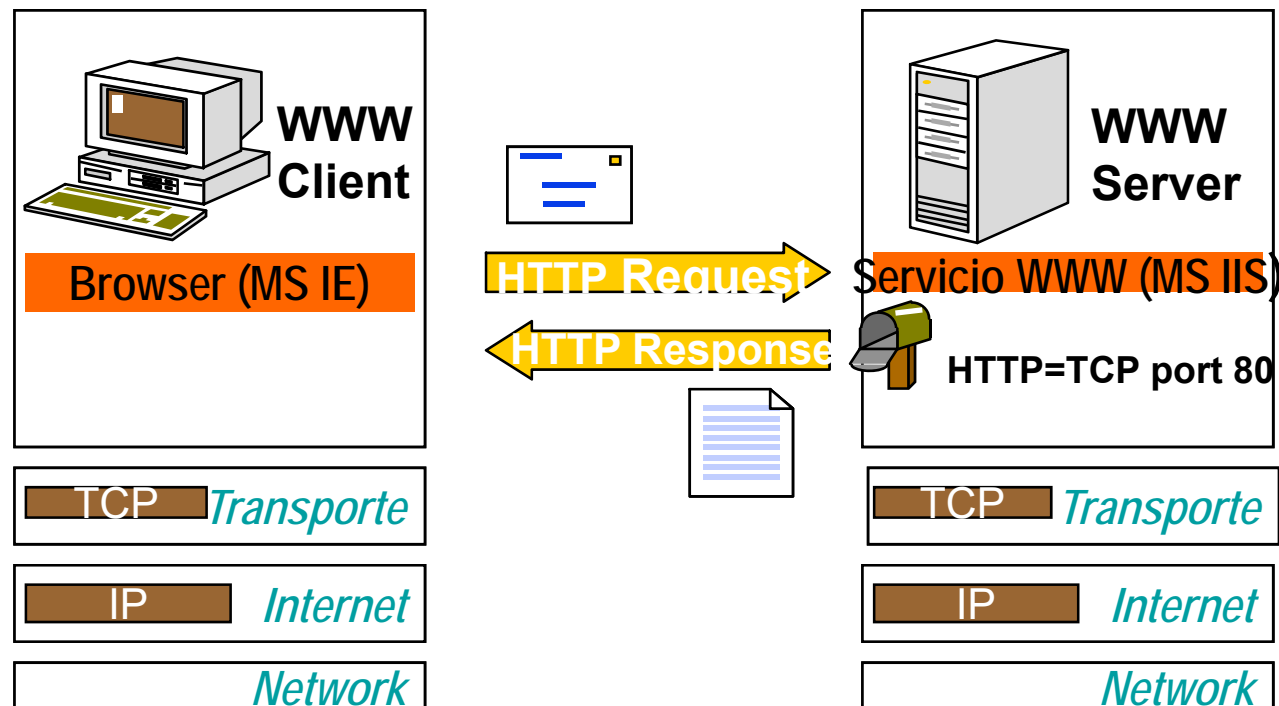
# Terminología web



- Página web
  - Compuesta de objetos
  - Objeto: archivo (texto, imagen, sonido, etc.) direccionable por un único URL
  - Páginas web compuestas por un archivo HTML base y varios objetos referenciados
  - Ejemplo: 1 página con texto HTML y 5 imágenes JPEG, tiene en total 6 objetos.
  - El archivo base referencia al resto de los objetos con URL
  - Un URL (“Uniform Resource Locator”) posee dos componentes:
    - Nombre de servidor que lo contiene (Dónde reside)
    - Nombre de la ruta al objeto.
  - Ejemplo: [www.unt.edu.ar/ejemplo/grafico.gif](http://www.unt.edu.ar/ejemplo/grafico.gif)

# HTTP

- HTTP define como los clientes web solicitan páginas web y como los servidores transfieren las páginas a los clientes.
- El cliente envía al servidor HTTP Request.
- El servidor responde con un HTTP Response.



# HTTP: Características generales



- **Protocolo de Aplicación:** HTTP es un protocolo de capa de aplicación que utiliza como protocolo de transporte a TCP.
  - TCP ofrece a HTTP un servicio confiable, por lo que HTTP no se preocupa por errores o pérdidas de datos.
  - Cliente HTTP inicia una conexión TCP con el servidor.
- **Request/Response:** Una vez iniciada una conexión, cliente envía "Request" al Servidor, el cual responde con un "Response" al Cliente.
- **"Stateless":** Servidor HTTP envía archivos solicitados sin almacenar información de estado. Se dice que HTTP es un protocolo **sin estado** ("stateless").
- **Transferencia bidireccional:** Por lo gral. servidor envía un documento al cliente. En algunos casos cliente tb. envía al servidor ("Forms")
- **Negociación de capacidades:** Se permite la negociación entre cliente y servidor de ciertos detalles como el set de caracteres.
- **Soporte de Caching:** Los clientes soportan caching de páginas.
- **Soporte de Intermediarios:** Soporte de servidores Proxies.

# Conexiones no persistentes y persistentes



## ➤ Conexiones no persistentes

- Pueden ser utilizadas por HTTP/1.0 y HTTP/1.1 (por compatibilidad hacia atrás)
- Pueden transmitir un único objeto web sobre una conexión TCP

## ➤ Conexiones persistentes

- Son las utilizadas por defecto en HTTP/1.1
- HTTP/1.0 no soporta conexiones persistentes.
- Se establece y se mantiene una única conexión TCP para la transferencia de múltiples objetos web.



# Conexiones no persistentes

- Ejemplo: URL del archivo base html:  
[www.unt.edu.ar/depto/home.index](http://www.unt.edu.ar/depto/home.index)
- Supongamos que este documento consta de 11 objetos web
- Los pasos que se siguen son:
  1. Cliente HTTP inicia conexión TCP con el servidor [www.unt.edu.ar](http://www.unt.edu.ar) sobre el puerto 80.
  2. El cliente envía mensaje de petición, el cual incluye la ruta /depto/home.index
  3. Servidor recibe mensaje petición, recupera el objeto y lo encapsula en el mensaje de respuesta, enviándolo al cliente.
  4. Servidor HTTP dice a TCP que cierre la conexión.
  5. Cliente recibe mensaje respuesta. Finaliza la conexión TCP. Cliente extrae el archivo HTML del mensaje de respuesta, examina y encuentra referencia a 10 objetos adicionales.
  6. Se repiten pasos 1. al 4.
- Se observa que cada conexión TCP transporta un mensaje de petición y uno de respuesta.

# Conexiones Persistentes



- Conexiones no persistentes son poco eficientes:
  - Establecimiento, manutención y terminación de conexiones TCP por cada objeto de un documento.
  - Retardo en la recepción de cada objeto HTTP (dos Round Trip Time mínimo).
- Mejora: permitir varias conexiones abiertas en forma simultánea
  - Desventaja: Múltiples conexiones TCP -> Consume recursos en explorador y ppalmente. en el servidor web!
- Con conexiones persistentes se deja **abierta** la conexión TCP luego de que el servidor envía el primer objeto.
  - De esa forma una página completa, con todos sus objetos pueden ser enviados por la misma conexión.
  - También podrían ser enviadas múltiples páginas al mismo cliente desde ese servidor
- Existen dos versiones de conexiones persistentes:
  - Sin "pipelining"
    - Cliente envía un nuevo request cuando la respuesta previa ha sido completada.
  - Con "pipelining" (entubamiento)
    - Se solicitan varios objetos en forma simultánea.
    - Este el modo por defecto en HTTP 1.1

# Formatos de Mensaje HTTP – “Request”: Ejemplo

- En HTTP existen dos tipos de mensajes:

- Petición (“Request”)
- Respuesta (“Response”)

- Ejemplo:

GET /dir/pagina.html	HTTP/1.1	<- Línea de petición
Host: <a href="http://www.unt.edu.ar">www.unt.edu.ar</a>		<- Líneas
Connection: close		<- de
User-agent: Mozilla/4.0		<- cabecera ...
Accept-language: en		
<CRLF>		

- Observamos:

- Mensaje en texto **ASCII** de 5 líneas (cada una finaliza con un CRLF, excepto la última que termina con un **CRLF** adicional).
- En realidad puede contener 1 sola línea o mas líneas que las del ejemplo.
- Primera línea: **Línea de Petición** (“Request”)
  - 3 campos:
    - **Método** (GET). Otros métodos: POST, HEAD, etc.
    - **URL**
    - Versión HTTP
- Siguiendo líneas: **Líneas de Cabecera**.

# Formatos de Mensaje HTTP – “Request”: Ejemplo



## ➤ Líneas de cabecera

### ➤ Host:

- Indica donde reside el objeto (nombre del servidor web)

### ➤ Connection: close.

- El cliente le dice al servidor que no use conexiones persistentes (default es utilizar conexiones persistentes).

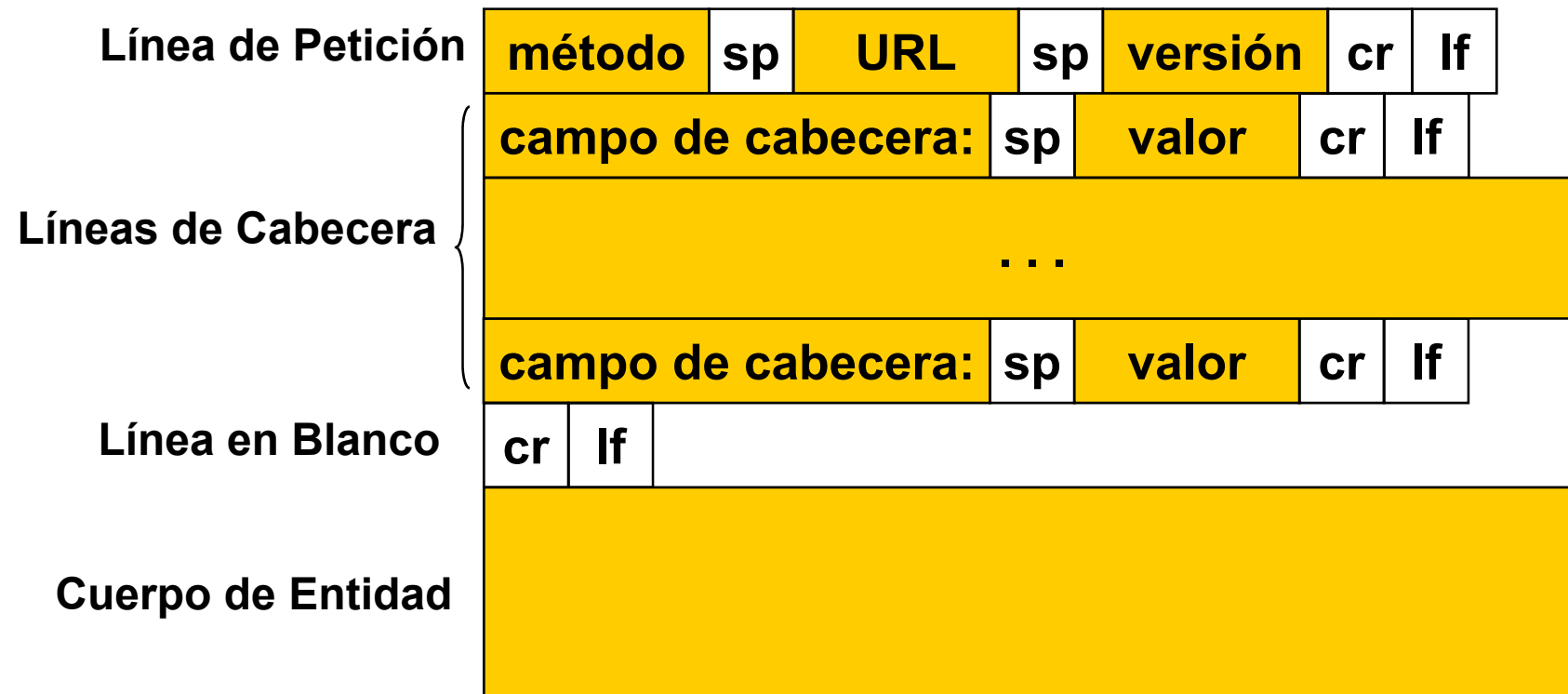
### ➤ User-agent: Mozilla 4.0

- Tipo de navegador del cliente.

### ➤ Accept-Language:

- Cabecera de negociación que explicita el lenguaje que desea obtener el cliente la página (si es que existe)
- Esta es una de las tantas cabeceras de negociación que maneja HTTP.

# Formatos de Mensaje HTTP – "Request": Genérico.



**Nota:** el cuerpo de entidad está vacío en el método get

# Formatos de Mensaje HTTP – "Response": Ejemplo

## ➤ Ejemplo:

HTTP/1.1 200 OK	<-Estado
Connection: close	<-Cabecera
Date: Thu, 19 May 2005 18:15:24 GMT	<- ...
Server: Apache/1.3.0 (Unix)	
Last-Modified: Mon, 5 Apr 2005 12:26:55 GMT	
<b>Content-Length: 6821</b>	
Content-Type: text/html	<- ...
(datos ...)	<- Cuerpo
(datos ...)	<- de entidad

## ➤ Consta de tres secciones

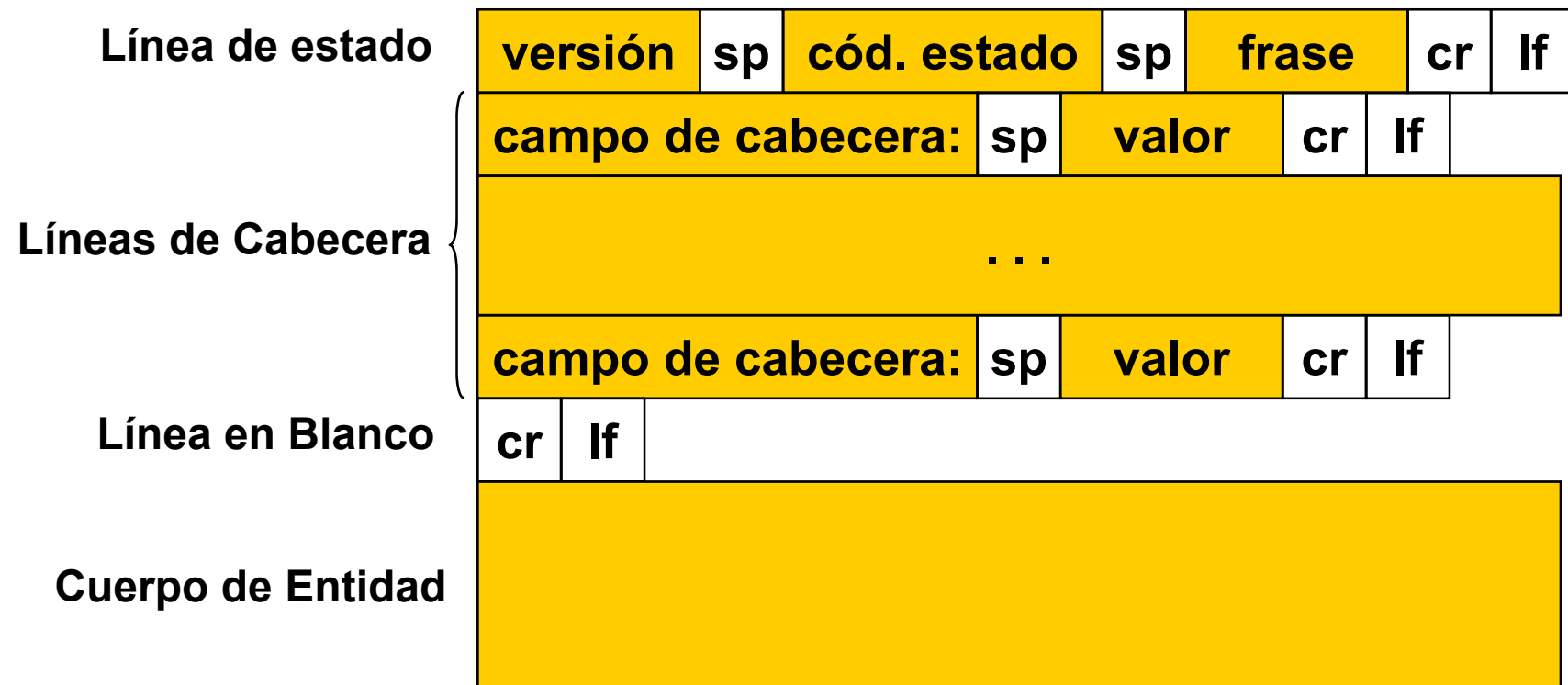
- Línea de estado
- Líneas de Cabecera (6)
- Cuerpo de Entidad

# Formatos de Mensaje HTTP – “Response”: Ejemplo



- Línea de Estado
  - Tres campos
    - Versión del protocolo (HTTP/1.1)
    - Código de Estado (200)
    - Mensaje de Estado (OK)
- Líneas de cabecera
  - Connection close: va a cerrar la conexión TCP luego de enviar el mensaje HTTP
  - Date: Fecha en que se creó y envió la **respuesta** HTTP
  - Server: Tipo de servidor que genera la respuesta
  - Last-Modified: Fecha en que el **objeto** fue creado o modificado
    - Usada para mantenimiento del cache.
  - Content-Length: Longitud (en Bytes) del objeto enviado.
  - Content-Type: Tipo de objeto en el cuerpo de la entidad (en este caso, texto HTML)

# Formatos de Mensaje HTTP – "Response": Genérico.





# Códigos y Frases de Estado



- **200 OK:** Petición exitosa
- **301 Moved Permanently:** El objeto solicitado ha sido movido permanentemente. El nuevo URL es indicado en la cabecera **Location:**
- **400 Bad Request:** Código genérico de error. Indica que la petición no pudo ser entendida por el servidor.
- **404 Not Found:** Documento solicitado no existe en el servidor
- **505 HTTP Version Not Supported:** La versión de HTTP pedida no es soportada por el servidor.
- Obviamente existe una gran cantidad de códigos de estados.