Capa Aplicación:Correo Electrónico y DNS

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 3rd* edition. Jim Kurose, Keith Ross Addison-Wesley, 2004.

Capítulo 2: Capa Aplicación

- 2.1 Principios de la aplicaciones de red
- 2.2 Web y HTTP
- 2.3 FTP
- 2.4 Correo Electrónico
  - SMTP, POP3, IMAP
- 2.5 DNS
- 2.6 P2P Compartición de archivos
- 2.7 Programación de Socket con TCP
- 2.8 Programación de socket con UDP
- 2.9 Construcción de un servidor WEB

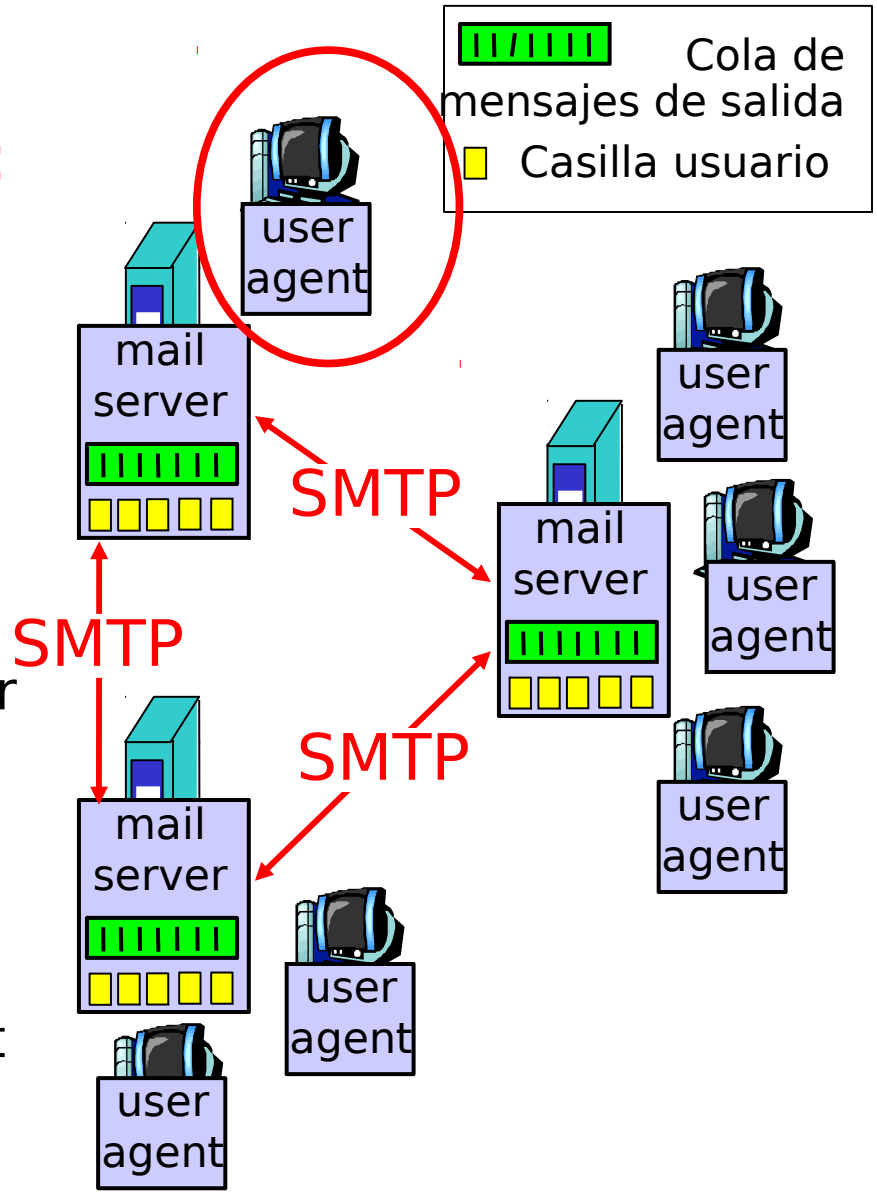
Correo Electrónico

Tres mayores componentes:

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

Agente Usuario

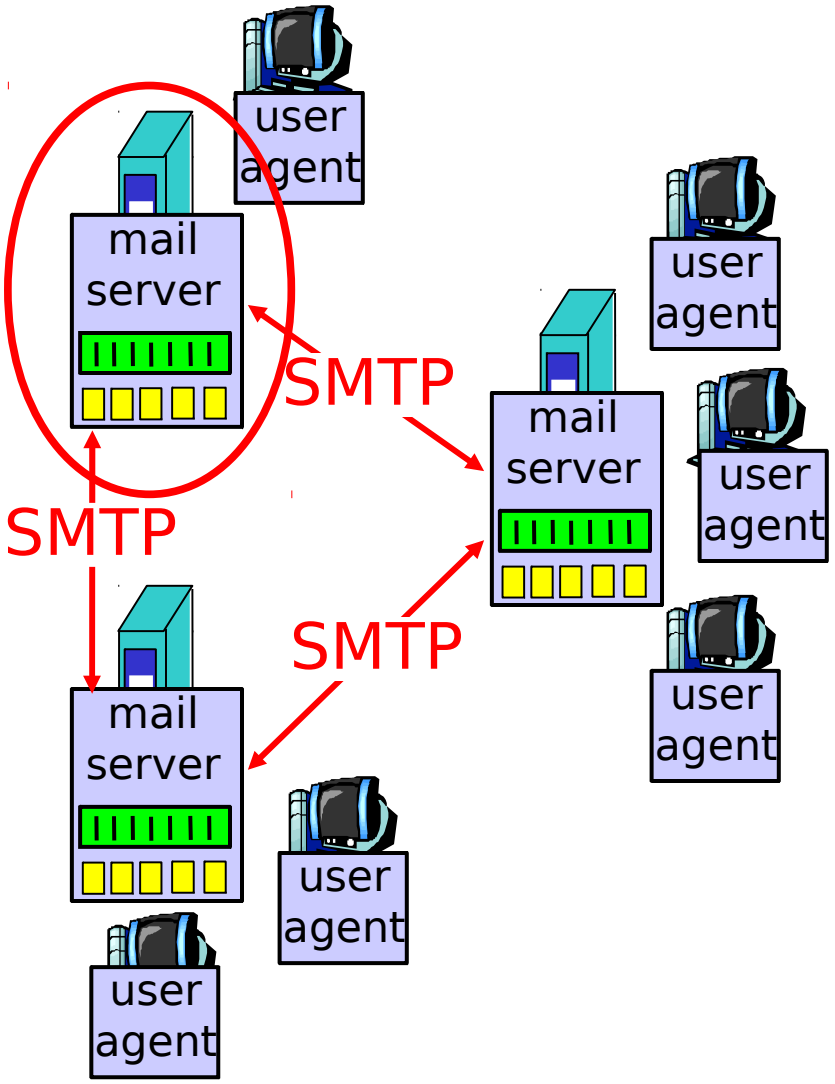
- También conocido como “lector de correo”
- Escritura, edición, lectura de mensajes de correos
- e.g., Eudora, Outlook, Mozilla Thunderbird, Iphone mail client
- Mensajes de salida y entrada son almacenados en servidor



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`

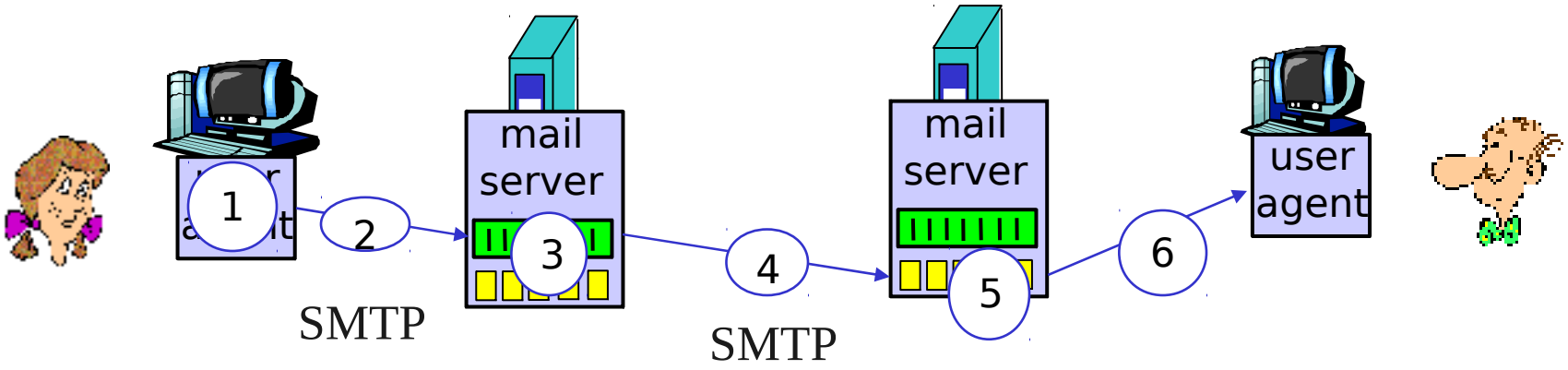
2) El agente de Alicia envía en mensaje a su servidor de correo; el mensaje es puesto en cola de salida

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



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 que no permiten el uso de telnet para envío de correo. La USM y gmail usan TLS (Transport Layer Security)

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 funciona.

Formato de mensajes de correo (comando DATA)

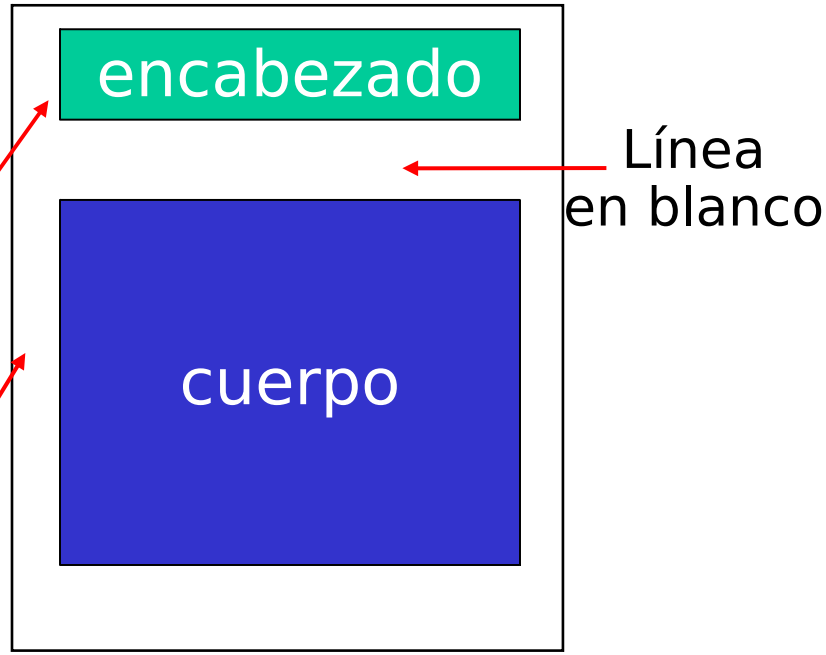
SMTP: protocolo para intercambio de mensajes de correo

RFC 822: estándar para el formato de los mensajes:

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

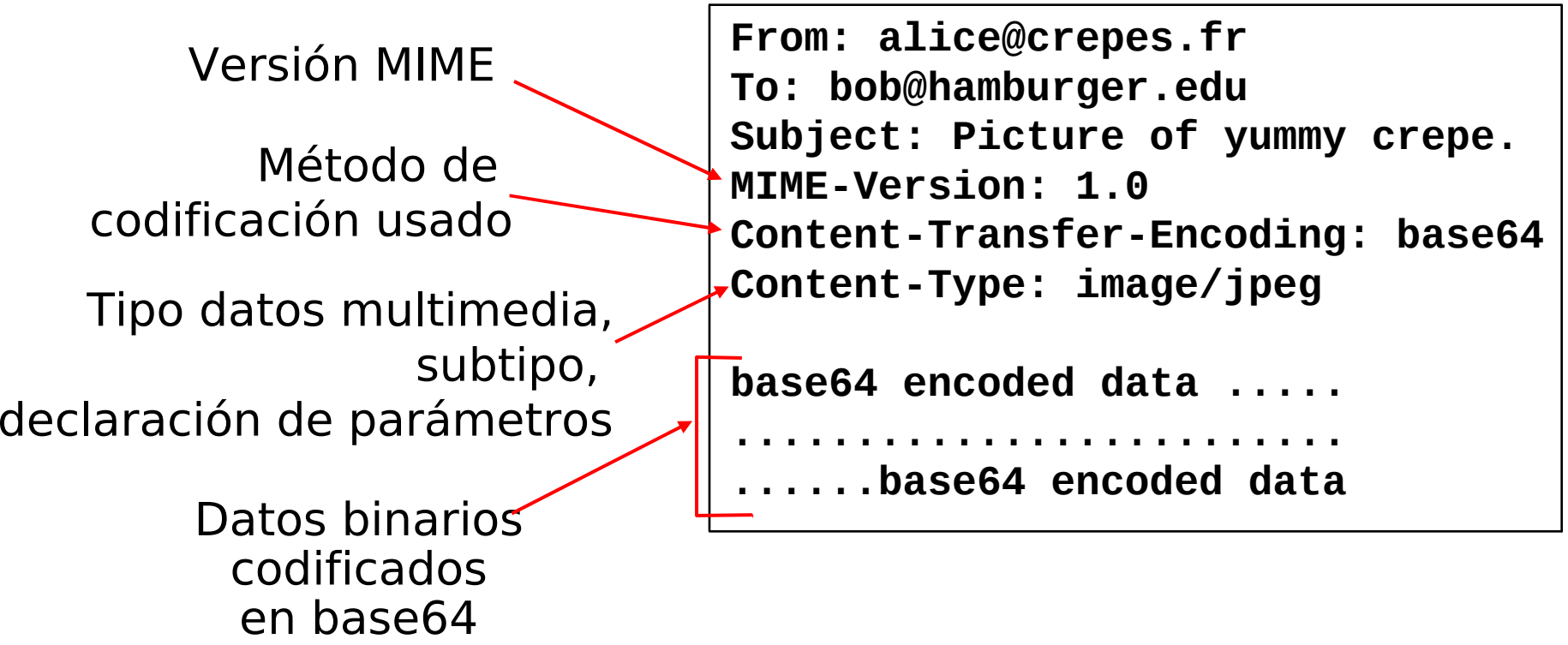
diferente a los comandos SMTP!

- Cuerpo
  - El “mensaje”, sólo caracteres ASCII

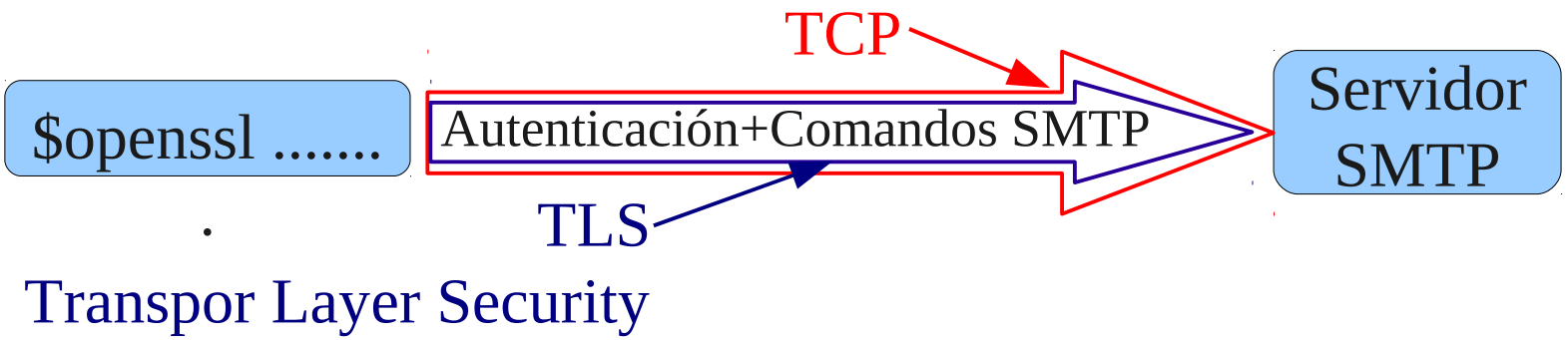


Formato de mensaje: extensiones multimedia

- 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 +/=



Prueba SMTP actual con gmail



- Ver datos para comunicación con gmail en: http://mail.google.com/support/bin/answer.py?hl=en&answer=77662
- 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. Usaremos comando openssl de linux.
  - Luego debo enviar 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:

  % perl -MMIME::Base64 -e 'print encode_base64("\000elo322\@gmail.com\000tu.password")'

- Ahora recién podemos enviar los comandos SMTP para enviar el correo.
- Esto se ve a continuación:

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("\000agustin.j.gonzalez\@gmail.com\000elo322_2010")'

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

---

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+wnuklpj4MdF35rIkgQIDAQABo4GmMIGjMB0GA1UdJQQWMBQGCCsG

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.)

```
250 PIPELINING
AUTH PLAIN AGFndXN0aW4uai5nb256YWxlekBnbWFpbC5jb20AZWxvMzIyXzIwMTA=
235 2.7.0 Accepted
mail from: <agustin.j.gonzalez@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: ELO322 <elo322@usm.cl>
Subject: Ejemplo de envío de correo vía gmail server.
Hola Muchachos!
UN saludo cordial del profe,
Agustín
.
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
```

Opción para codificar en base64

- En lugar de usar perl para codificar en base64, se puede usar el utilitario base64 en linux.

  agustin@agustin-laptop:~$ base64

  ^@agustin.j.gonzalez@gmail.com^@elo322_2010

  AGFndXN0aW4uai5nb256YWxlekBnbWFpbC5jb20AZWxvMzIyXzIwMTAK

  agustin@agustin-laptop:~$

- Con Control-@ se ingresa el carácter null (ASCII 00) necesario como parte del formato para enviar la cuenta y la password.

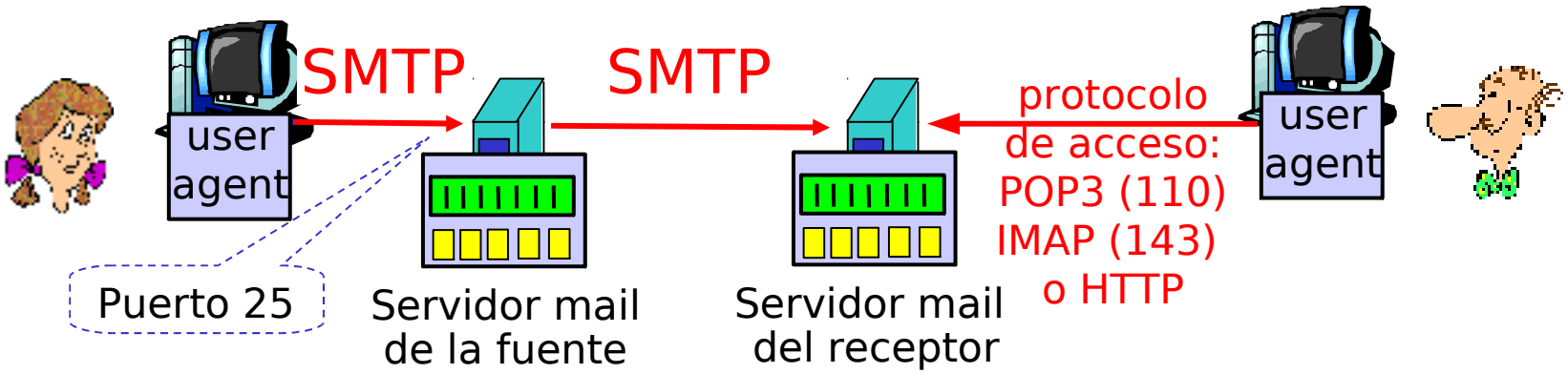
SMTP: palabras finales

- SMTP usa conexiones persistentes
- SMTP requiere que el mensaje (encabezado y cuerpo) sean en 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

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
  - POP: Post Office Protocol [RFC 1939]
    - autenticación (agent <-->server) y bajada
  - IMAP: Internet Mail Access Protocol [RFC 1730]
    - Más características (más complejo)
    - Permite manipulación de los mensajes almacenados en el servidor
  - HTTP: 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
- **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 mapeo entre Ids (identificadores) de mensajes y nombres de carpetas.

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

Capítulo 2: Capa Aplicación

- 2.1 Principios de la aplicaciones de red
- 2.2 Web y HTTP
- 2.3 FTP
- 2.4 Correo Electrónico
  - SMTP, POP3, IMAP
- 2.5 DNS
- 2.6 P2P Compartición de archivos
- 2.7 Programación de Socket con TCP
- 2.8 Programación de socket con UDP
- 2.9 Construcción de un servidor WEB

DNS: Domain Name System (Sistema de nombres de dominio)

Personas: muchos identificadores:

- ROL, RUT, name, # pasaporte

Host y router en Internet:

- Dirección IP (32 bit) – usada para direccionar datagramas (ideal para router)
- "nombre", e.g., www.yahoo.com – son usados por humanos

Q: ¿Quién mapea entre direcciones IP y nombres?

Domain Name System:

- *Base de datos distribuida* implementada en una jerarquía de muchos *servidores de nombres*
- *Protocolo de capa aplicación* permite a host, routers, y servidores de nombre comunicarse para *resolver* nombres (traducción dirección/nombre)
- No está orientado al uso directo de los usuarios, ellos usan nombres.
  - DNS es función central de la Internet implementada como protocolo de capa aplicación
  - La idea de diseño de Internet es dejar la complejidad en la "periferia" de la red.

DNS

Servicios DNS

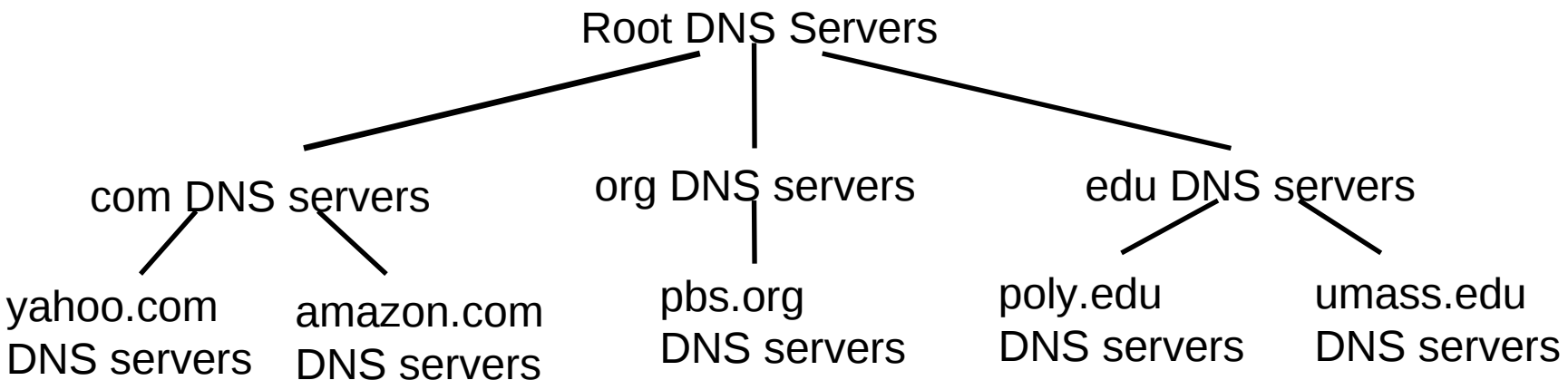
- Traducción de nombre de host a dirección IP
- Alias para host
  - Nombre canónico o alias
  - Nombre canónico: CNAME en RFC 1035
- Alias para servidor de correo
- Distribución de carga
  - Servidores Web replicados: conjunto de direcciones IP para un nombre canónico (e.g. relay1.west-coast.amazon.com), servidore DNS rota entre direcciones IP

¿Por qué no centralizar DNS?

- Único punto de falla
- Volumen de tráfico, muchos necesitan el DNS
- Sería una base de datos centralizada distante con grandes retardos de acceso.
- Mantención, es mejor que cada dominio gestione sus nombres

Porque no sería *escalable!*

Base de datos jerárquica y distribuida



Cliente desea IP de www.amazon.com; 1ra aprox. :

- Cliente consulta al servidor raíz para encontrar servidor DNS de com
- Cliente consulta servidor DNS TLD (Top Level Domain) de com para obtener servidor DNS de amazon.com
- Cliente consulta servidor DNS amazon.com para obtener dirección IP de www.amazon.com

DNS: servidores de nombre en raíz

- Son contactados por servidor de nombre local cuando no puede resolver un nombre
- Servidor nombre raíz:
  - Contacta servidor de nombre autoritario de la zona superior (e.g. com) si mapeo del nombre es desconocido para él
  - Obtiene mapeo (propio o desde otro servidor raíz)
  - Retorna mapeo al servidor de nombre local

13 servidores de nombre raíz en todo el mundo

TLD y Servidores Autoritarios

- Top-level domain (TLD) servers: responsable por com, org, net, edu, etc., y todos los dominios superiores de cada país: uk, fr, ca, jp, cl, etc..
  - Network solutions mantiene servidores para el TLD de com
  - Educause para el TLD de edu
  - Nic (network information center) para el TLD de cl (www.nic.cl)
- Servidores DNS autoritarios: son servidores DNS de las organizaciones y proveen mapeos autoritarios entre hostname e IP (e.g., Web y mail).
  - Éstos pueden ser mantenidos por la organización o el proveedor de servicio

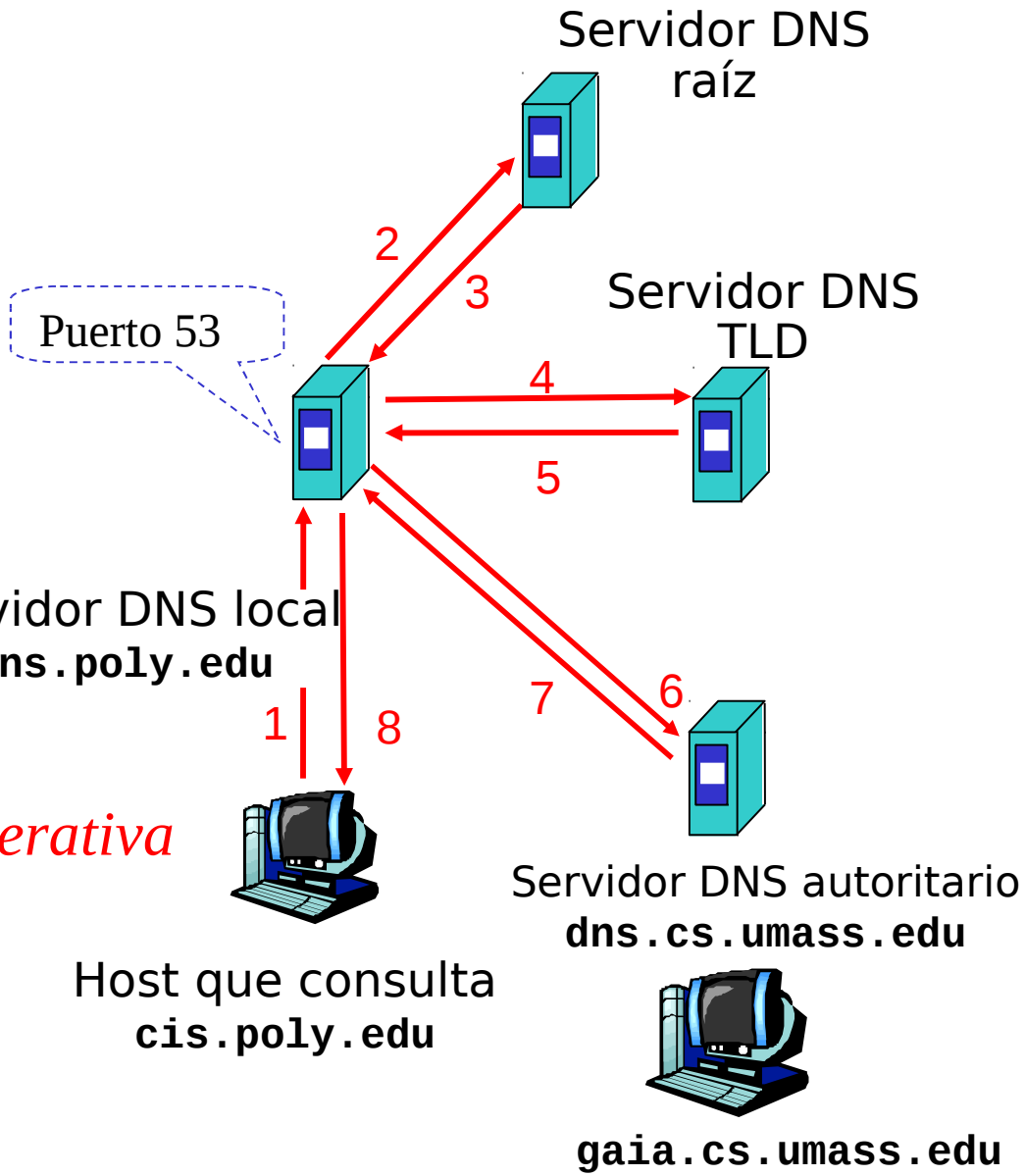
Servidor de nombre local

- No pertenece estrictamente a la jerarquía
- Cada ISP (ISP residencial, compañía, universidad) tiene uno.
  - También son llamados “servidor de nombre por omisión” (default name server)
- Cuando un host hace una consulta DNS, ésta es enviada a su servidor DNS local
  - Actúa como proxy, re-envía consulta dentro de la jerarquía.

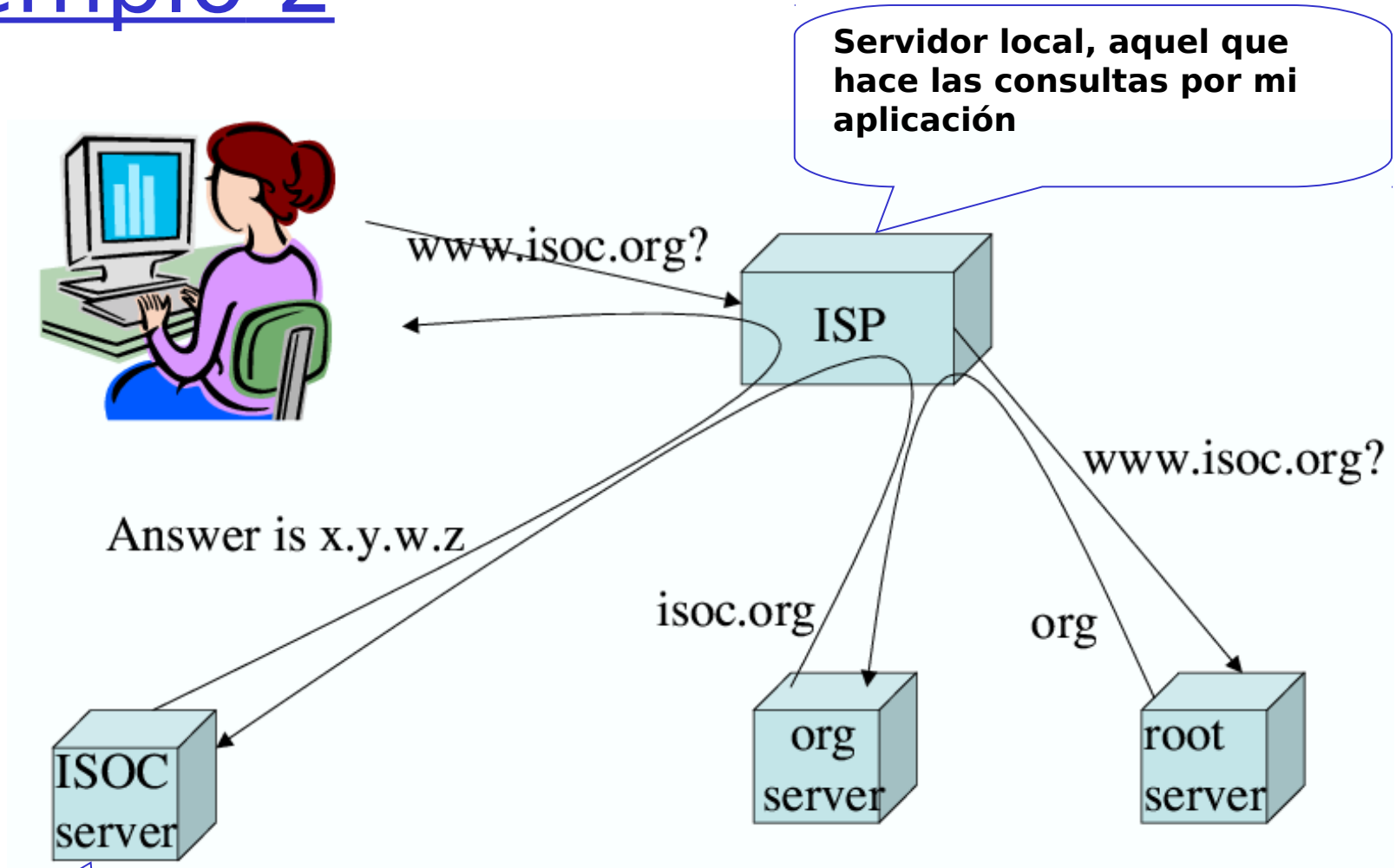
Ejemplo 1

Consulta iterativa:

- Host en cis.poly.edu quiere la dirección IP de gaia.cs.umass.edu
- Servidor contactado responde con el nombre del servidor a contactar
- "Yo no conozco este nombre, pero pregunta a este servidor"



Ejemplo 2



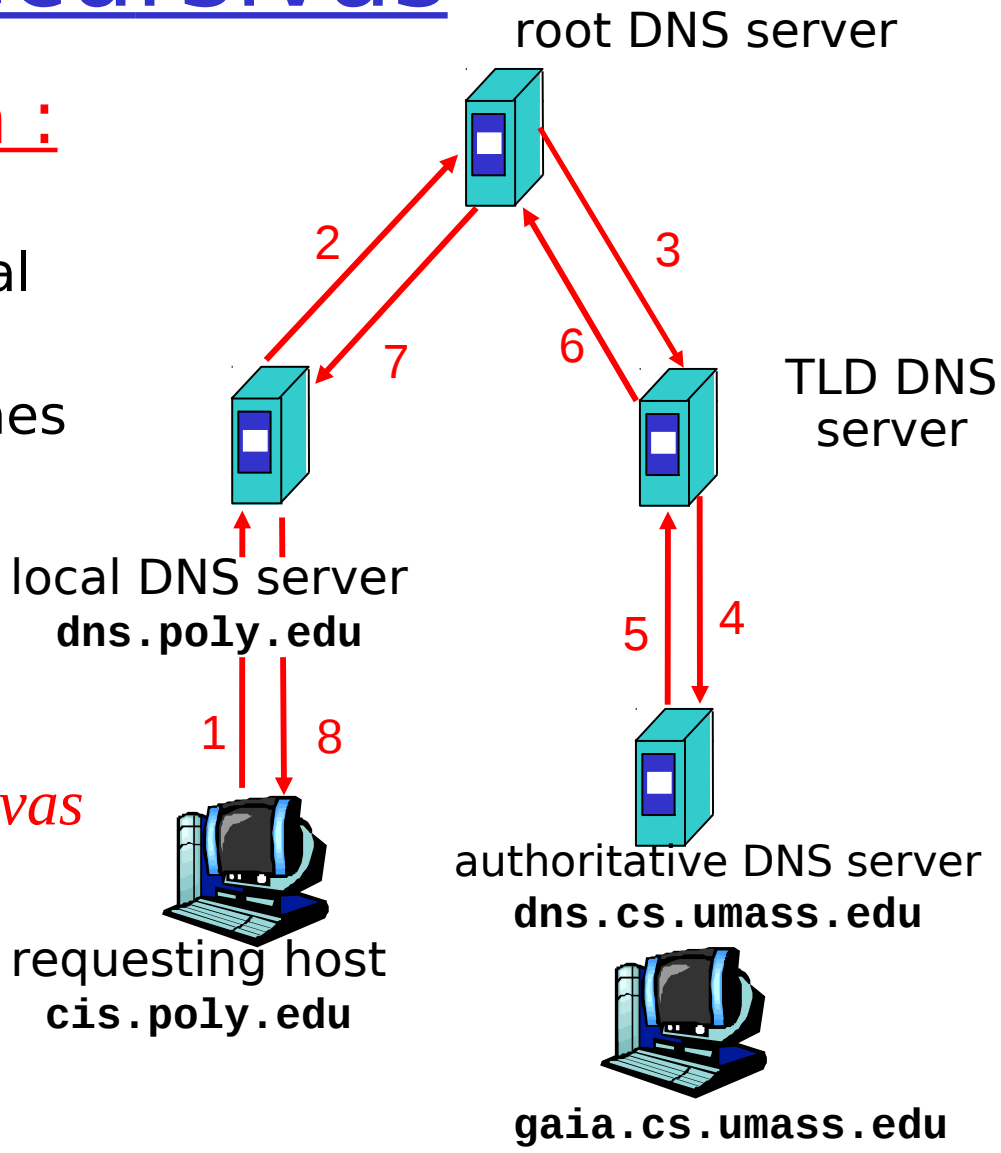
Consulta iterativa

Servidor autoritario, aquel que define el mapeo nombre <-> IP

Consultas Recursivas

Consulta recursiva :

- Pone la carga de la resolución de nombre al servidor contactado.
- ¿Qué pasa en situaciones de alta carga?



Consultas Recursivas

Ejemplo

- Hacer algo del tipo:
  $ nslookup www.elo.utfsm.cl
- Luego:
  $ nslookup 200.1.17.5
- Finalmente:
  $nslookup www.google.com
- Y
  $ nslookup 64.233.161.99
- Estando en aragorn hacer:
  $ nslookup 200.1.17.195

Es común que las máquinas tengan asignados alias; por ejemplo www.elo.utfsm.cl es un alias para vega.elo.utfsm.cl. ¿Expliqué por qué no conviene usar www.elo.utfsm.cl como nombre canónico de la máquina?

- Usando www.elo.utfsm.cl como alias es posible configurar otra máquina para reemplazar el servidor WEB y una vez que esté lista, sólo hacemos el cambios del alias en el DNS para que los accesos futuros se dirijan al nuevo servidor. Esta operación resulta transparente para los usuarios. Si fuera nombre canónico, mientras configuramos el nuevo servidor WEB, no podríamos tener dos máquinas con igual nombre.

DNS: Cache y actualización de registros

- Una vez que un servidor de nombre conoce un mapeo, éste *guarda* (caches) el mapeo
  - Las entradas del cache expiran (desaparecen) después de algún tiempo
  - Servidores TLD típicamente están en cache de los servidores de nombre locales
    - Así los servidores de nombre raíz no son visitados con frecuencia
- Mecanismos de Actualización/notificación están bajo diseño por el IETF (Internet Engineering Task Force)
  - RFC 2136
  - http://www.ietf.org/html.charters/dnsind-charter.html

Registros DNS

DNS: es una base de datos distribuida que almacena registros de recursos (resource records, RR)

Formato RR: **(name, value, type, ttl)**

- Type=A
  - **name** es un hostname (nombre real o canónico)
  - **value** es una dirección IP
- Type=NS
  - **name** es un dominio (e.g. foo.com)
  - **value** es la dirección IP (nombre) del servidor autoritario que sabe cómo obtener las direcciones IP de este dominio.
- Type=CNAME
  - name es un alias para algún nombre real (indicado en type A)

    www.ibm.com es realmente servereast.backup2.ibm.com

  value es el nombre real (canónico)
- Type=MX

  value es el nombre del servidor de correo asociado con name

Inserción de registros en DNS

- Ejemplo: Recién se crea una empresa “Network Utopia”
- Debemos registrar el nombre networkuptopia.com en un administrador de dominio (e.g., Network Solutions)
  - Necesitamos proveer el nombre y la dirección IP de nuestro servidor de nombre autoritario (primario y segundario)
  - Administrador del dominio inserta dos RRs en el servidor TLD com:

```
(networkutopia.com, dns1.networkutopia.com, NS)
(dns1.networkutopia.com, 212.212.212.1, A)
```

- Incorporar en el servidor autoritario un registro Tipo A para www.networkuptopia.com y un registro Tipo MX para networkutopia.com
- En Chile debemos acceder a NIC Chile para arrendar un nombre de dominio.

Explique por qué los resultados de varios PING a www.youtube.com muestran direcciones IPs distintas:
agustin@agustin-laptop:~$ ping www.youtube.com
PING youtube-ui.l.google.com (74.125.224.76) 56(84) bytes of data.
64 bytes from 74.125.224.76: icmp_seq=1 ttl=52 time=162 ms
.... Luego:
agustin@agustin-laptop:~$ ping www.youtube.com
PING youtube-ui.l.google.com (74.125.224.42) 56(84) bytes of data.
64 bytes from 74.125.224.42: icmp_seq=1 ttl=52 time=160 ms
.... Luego:
agustin@agustin-laptop:~$ ping www.youtube.com
PING youtube-ui.l.google.com (74.125.224.79) 56(84) bytes of data.
64 bytes from 74.125.224.79: icmp_seq=1 ttl=52 time=175 ms
...
Explique cómo esto se hace posible.

?

- El ping a un mismo nombre lógico condujo a tres máquinas distintas por ello tres IPs distintas. Esto se explica porque el servicio de youtube es atendido por un conjunto de máquinas para poner servir a más usuarios a la vez.
- Esto es posible gracias al servidor DNS. Cuando el ping consulta por la IP de www.youtube.com, el servidor que maneja este nombre identifica la máquina adecuada de entre el conjunto para atender la petición y retorna esa dirección IP.

Capítulo 2: Capa Aplicación

- 2.1 Principios de la aplicaciones de red
- 2.2 Web y HTTP
- 2.3 FTP
- 2.4 Correo Electrónico
  - SMTP, POP3, IMAP
- 2.5 DNS
- 2.6 P2P Compartición de archivos
- 2.7 Programación de Socket con TCP
- 2.8 Programación de socket con UDP
- 2.9 Construcción de un servidor WEB