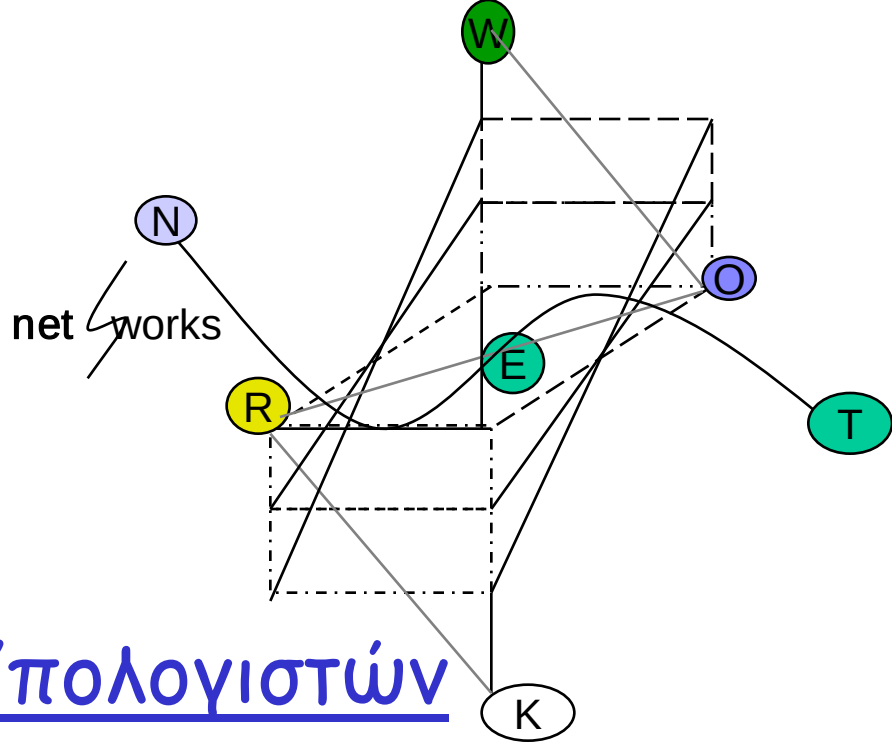


ΗΥ-335 : Δίκτυα Υπολογιστών

Επίπεδο Εφαρμογής

Μαρία Παπαδοπούλη

Τμήμα Επιστήμης Υπολογιστών
Πανεπιστήμιο Κρήτης
Χειμερινό εξάμηνο 2012-2013

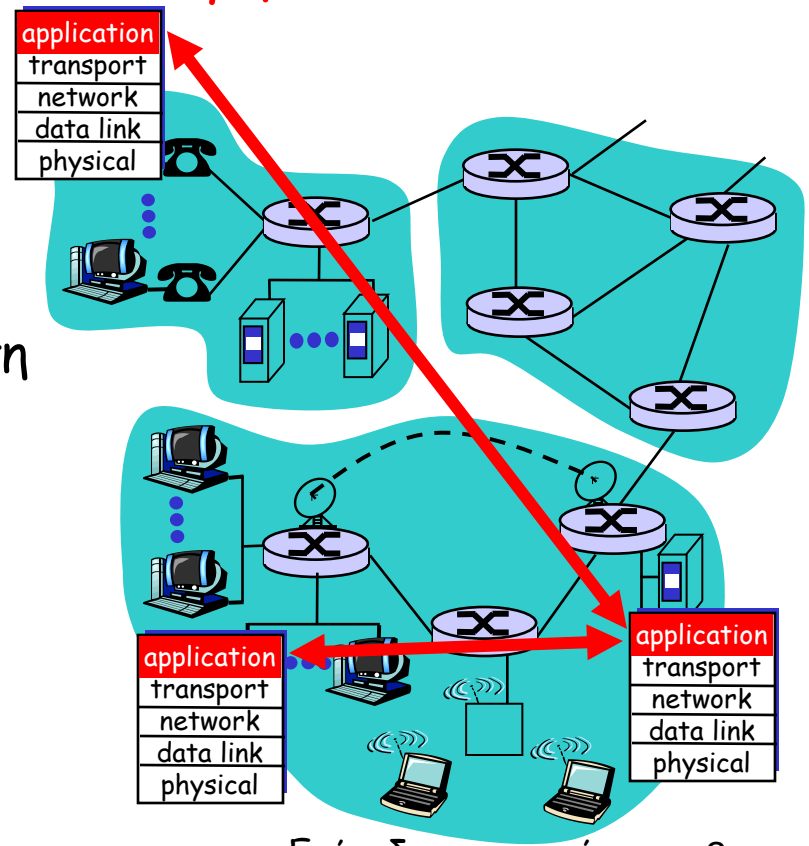
Δημιουργώντας μια δικτυακή εφαρμογή

Γράφουμε προγράμματα τα οποία:

- Τρέχουν σε διαφορετικά τερματικά συστήματα και επικοινωνούν μέσω δικτύου
  - Π.χ., το λογισμικό του web server επικοινωνεί με το λογισμικό του browser

Λίγο λογισμικό έχει γραφτεί για συσκευές στον πυρήνα του δικτύου

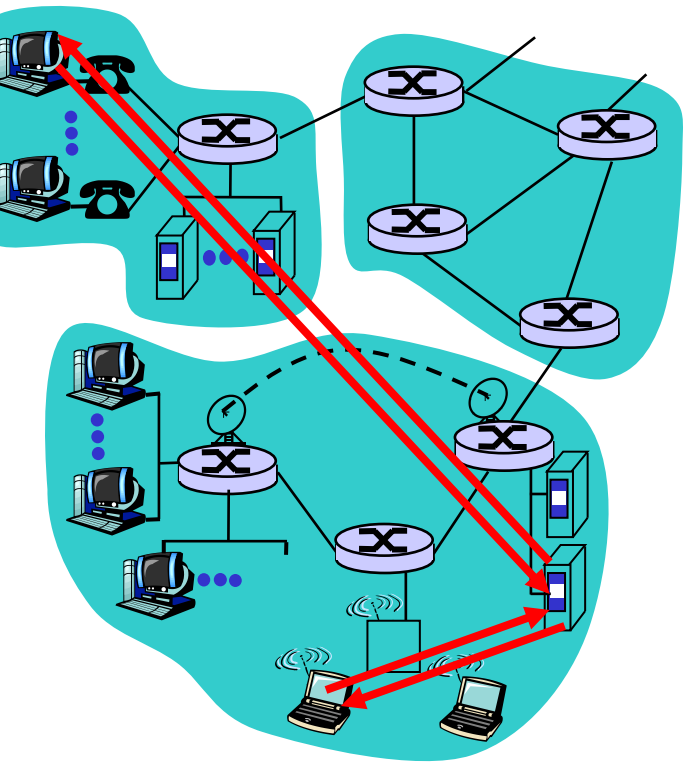
- Οι συσκευές στον πυρήνα του δικτύου δεν τρέχουν κώδικα εφαρμογής χρήστη
- Η εφαρμογή στα τερματικά συστήματα επιτρέπει τη ραγδαία ανάπτυξη και διάδοση εφαρμογών



Γενικές αρχιτεκτονικές εφαρμογών

- Πελάτη-εξυπηρετητή (client-server)
- Διομότιμο σύστημα (peer-to-peer P2P)
- Υβριδικό του πελάτη-εξυπηρετητή και του διομότιμου συτήματος

Client-server αρχιτεκτονική



server:

- Συνεχής διαθεσιμότητα (always-on host)
- Μόνιμη IP διεύθυνση
- Ομάδα από servers (server farms) για κλιμάκωση (scaling)

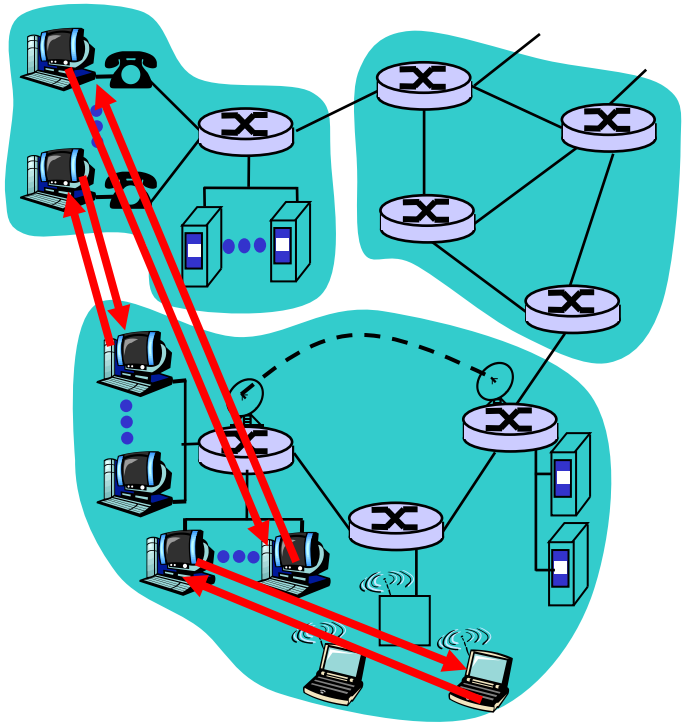
clients:

- Επικοινωνούν με τον server
- Πιθανότατα χωρίς συνεχή σύνδεση
- Πιθανότατα με δυναμικές IP addresses (που λαμβάνουν από έναν DCHP server)
- Δεν επικοινωνούν άμεσα ο ένας client με τον άλλον

Καθαρή P2P αρχιτεκτονική

- Δεν υπάρχει server που να είναι συνεχώς διαθέσιμος
- Τερματικά συστήματα επικοινωνούν το ένα με το άλλο
- Οι peers έχουν διακοπτόμενη σύνδεση στο Ιντερνετ και οι IP διευθύνσεις τους μπορεί να αλλάζουν
- Παράδειγμα: Gnutella

☞ Είναι κλιμακώσιμα αλλά όχι εύκολο να τα διαχειριστείς



Υβριδικό client-server και P2P

Skype

- Εφαρμογή διαδικτυακής τηλεφωνίας
- Εύρεση διεύθυνσης απομακρυσμένου προσώπου: κεντρικοποιημένος server(s)
- Η σύνδεση από client σε client είναι άμεση (όχι μέσω server)

Άμεση ανταλλαγή μυνημάτων (instant messaging)

Η συνομιλία μεταξύ δύο χρηστών είναι P2P

Εντοπισμός παρουσίας/θέσης κεντρικοποιημένη:

- Ο χρήστης καταχωρεί την IP διεύθυνσή του με κεντρικό server όταν μπαίνει online
- Ο χρήστης επικοινωνεί με τον κεντρικό server για να βρει τις IP διευθύνσεις των επαφών του

Διεργασίες που επικοινωνούν

Διεργασία: ***πρόγραμμα*** που τρέχει σε ένα host

Στον ίδιο host:

δύο διεργασίες επικοινωνούν μεταξύ τους με διαδεργασιακή επικοινωνία (*inter-process communication*) (ορίζεται στο OS)

Διεργασία client:

η διεργασία που ξεκινάει την επικοινωνία

Διεργασία server :

η διεργασία που περιμένει για επικοινωνία

☞ Διεργασίες σε **διαφορετικούς hosts επικοινωνούν με την ανταλλαγή μηνυμάτων**

Σημείωση: εφαρμογές με P2P αρχιτεκτονικές έχουν ***διεργασίες client & server***

Διεργασία στον δέκτη

Ο sender προσδιορίζει τη διεργασία στη συσκευή του δέκτη χρησιμοποιώντας τα παρακάτω δύο στοιχεία:

όνομα ή διεύθυνση της συσκευής → IP address

προσδιορίζει μοναδικά το network interface του δέκτη
μοναδικότητα στο Internet

ένα identifier που προσδιορίζει την διεργασία στον δέκτη

Port number

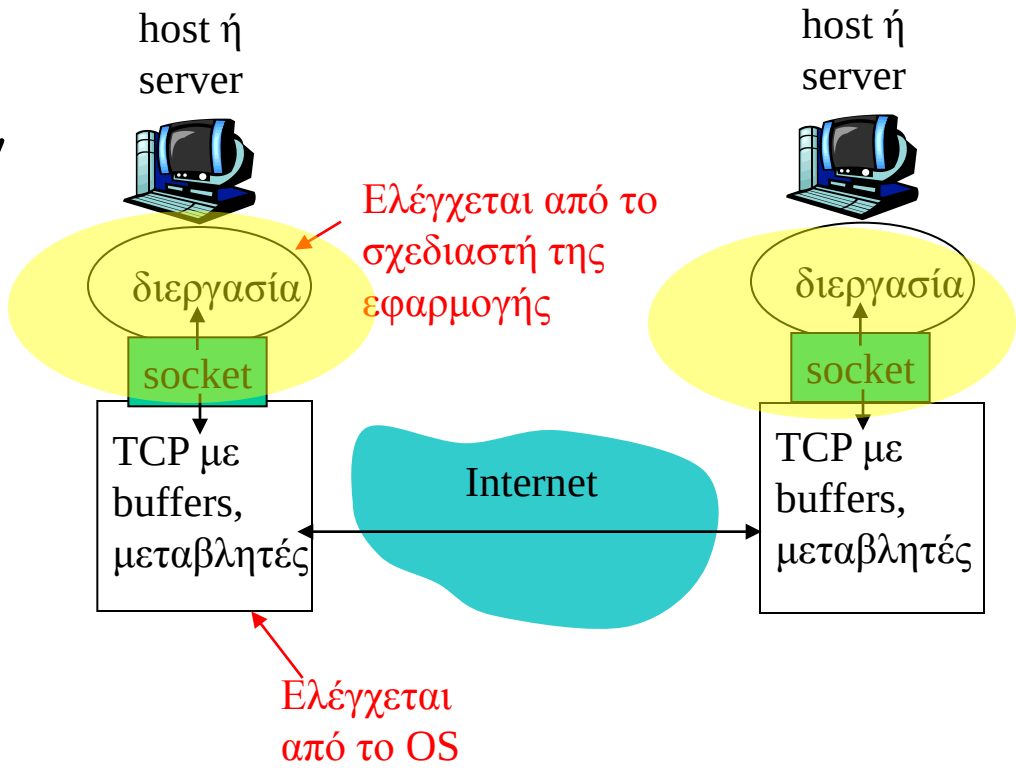
Υπενθύμιση: Sockets

Το socket είναι η **διεπαφή** μεταξύ του επιπέδου εφαρμογής & επιπέδου μεταφοράς μέσα σε μία συσκευή καθώς επίσης μεταξύ της εφαρμογής & δικτύου

Διεργασίες *στέλνουν/λαμβάνουν μηνύματα προς/από* ***socket***

Socket μοιάζει με μια "πόρτα"

- Η αποστέλουσσα διεργασία στέλνει το μήνυμα έξω από την πόρτα
- Η αποστέλλουσα διεργασία υποθέτει υποδομή μεταφοράς πίσω από την πόρτα που φέρνει το μήνυμα στο socket της λαμβάνουσας διεργασίας



API - επιλέγει:

- Το πρωτόκολλο μεταφοράς
- τιμές σε παραμέτρους

Διευθυνσιοδότηση διεργασιών

Για να λάβει μηνύματα η διεργασία πρέπει να έχει έναν *identifier*

- Ο host έχει μία μοναδική 32-bit IP διεύθυνση

- Αρκεί η IP διεύθυνση του host στον οποίο τρέχει η διεργασία για τον προσδιορισμό της διεργασίας;

  Απάντηση: ΟΧΙ, πολλές διεργασίες μπορεί να τρέχουν στον ίδιο host

- *Ο identifier* περιέχει IP διεύθυνση & αριθμούς θυρών (port numbers) της διεργασίας στο host

  Παραδείγματα αριθμών θυρών:

  - HTTP server: 80
  - Mail server: 25

Για να στείλουμε ένα HTTP μήνυμα στον gaia.cs.umass.edu web server:

- IP διεύυθνση: 128.119.245.12
- Αριθμός θύρας: 80

Πρωτόκολλα επιπέδου εφαρμογής

- ***Τύποι μηνυμάτων*** που ανταλλάσσονται
  π.χ., αίτηση, απάντηση
- ***Σύνταξη μηνύματος***:
  Τι πεδία υπάρχουν στο μήνυμα και πως αυτά διαχωρίζονται
- ***Σημασιολογία μηνύματος***
  Σημασία της πληροφορίας στα πεδία
- ***Κανόνες για το πότε και πώς*** οι ***διεργασίες*** στέλνουν & απαντούν σε μηνύματα

Πρωτόκολλα δημοσίου τομέα:

- Ορίζονται σε RFCs
- Επιτρέπουν διαλειτουργικότητα
- π.χ., HTTP, SMTP

Ιδιοταγή πρωτόκολλα:

π.χ., KaZaA

Τι υπηρεσία μεταφοράς χρειάζεται μία εφαρμογή;

Απώλεια δεδομένων

- Κάποιες εφαρμογές (π.χ., ήχου) μπορούν να ανεχθούν κάποιες απώλειες
- Άλλες εφαρμογές (π.χ., μεταφορά αρχείων, telnet) απαιτούν 100% αξιόπιστη μεταφορά δεδομένων

Εύρος ζώνης

Κάποιες εφαρμογές (π.χ., πολυμεσικές) απαιτούν κάποιο ελάχιστο εύρος ζώνης για να είναι αποτελεσματικές

Οι **"ελαστικές εφαρμογές"** χρησιμοποιούν οποιοδήποτε εύρος ζώνης είναι διαθέσιμο

Χρονισμός

Κάποιες εφαρμογές (π.χ., διαδικτυακή τηλεφωνία, διαδραστικά παιχνίδια) απαιτούν χαμηλή καθυστέρηση για να είναι αποτελεσματικές

Απαιτήσεις υπηρεσίας μεταφοράς για συνήθεις εφαρμογέ

Εφαρμογή	Απώλεια δεδομένων	Εύρος ζώνης	Ευαισθησία σε καθυστερήσεις
Μεταφορά αρχείων	Καμία απώλεια	ελαστική	όχι
e-mail	Καμία απώλεια	ελαστική	όχι
Έγγραφα Web	Καμία απώλεια	ελαστική	όχι
φωνή/video πραγματικού χρόνου	Ανοχή σε απώλειες	ήχος: 5kbps-1Mbps video:10kbps-5Mbps	ναι, της τάξης των100 msec
Αποθηκευμένη φωνή/video	Ανοχή σε απώλειες	Όπως πριν	ναι, μερικά secs
Διαδραστικά παιχνίδια	Ανοχή σε απώλειες	Λίγα kbps πάνω	ναι, της τάξης των 100 msec
Άμεση ανταλλαγή μηνυμάτων	Καμία απώλεια	ελαστική	Ναι και όχι

Υπενθύμιση:
Πρωτόκολλα μεταφοράς υπηρεσιών στο Internet

TCP υπηρεσία:

- *Συνδεσιστρεφής:* απαιτείται εγκαθίδρυση μεταξύ διεργασιών των client & server
- *Αξιόπιστη μεταφορά* μεταξύ διεργασιών των sender & receiver
- *Έλεγχος ροής:* ο sender δεν κατακλύζει τον receiver
- *Έλεγχος συμφόρησης:* μειώνει την ροή του sender στην περίπτωση υπερφόρτωσης του δικτύου
- *Δεν παρέχει χρονικές* ή ελάχιστου bandwidth εγγυήσεις

UDP υπηρεσία:

- Αναξιόπιστη μεταφορά δεδομένων μεταξύ της αποστέλλουσας και της λάμβάνουσας διεργασίας
- Δεν παρέχει: εγκαθίδρυση σύνδεσης, αξιοπιστία, έλεγχος ροής, έλεγχος συμφόρησης, χρονισμός, ή εγγύηση εύρους ζώνης

Q: Γιατί υπάρχει τότε η UDP σύνδεση?

Εφαρμογές διαδικτύου: πρωτόκολλα εφαρμογής, μεταφοράς

Εφαρμογή	Πρωτόκολλο επιπέδου εφαρμογής	Πρωτόκολλο μεταφοράς
e-mail	SMTP [RFC 2821]	TCP
Απομακρυσμένη προσπέλαση τερματικού	Telnet [RFC 854]	TCP
Web	HTTP [RFC 2616]	TCP
Μεταφορά αρχείων	FTP [RFC 959]	TCP
Πολυμέσα συνεχούς ροής	Ιδιοταγές (π.χ. RealNetworks)	TCP ή UDP
Διαδικτυακή τηλεφωνία	ιδιοταγές (π.χ., Vonage,Dialpad)	συνήθως UDP

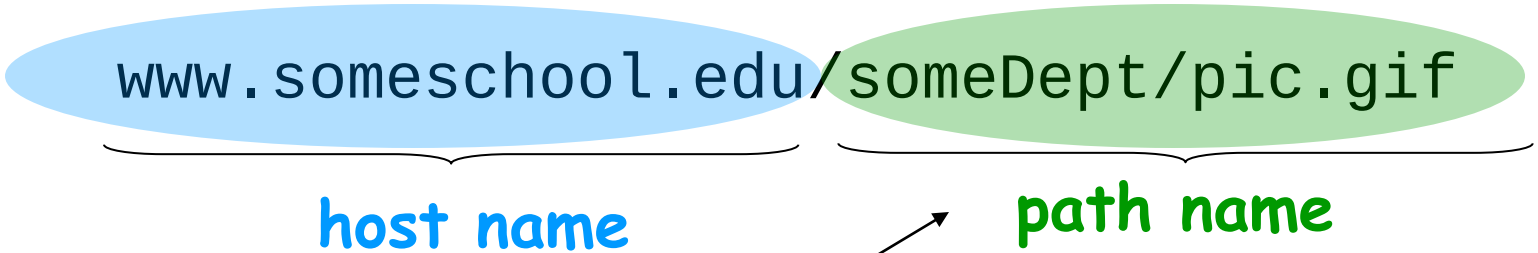
Web και HTTP

Ορολογία

- Μία **ιστοσελίδα** αποτελείται από αντικείμενα (***objects***)
- Ένα object μπορεί να είναι: HTML file, JPEG image, Java applet, audio file,…
- Μία ιστοσελίδα αποτελείται από **base HTML-file** που περιλαμβάνουν μερικά ***referenced objects***

☞ Κάθε object γίνεται ***addressable*** από ένα URL

Παράδειγμα URL:



Path name: shows from where (in the RAM or disk) to retrieve the object

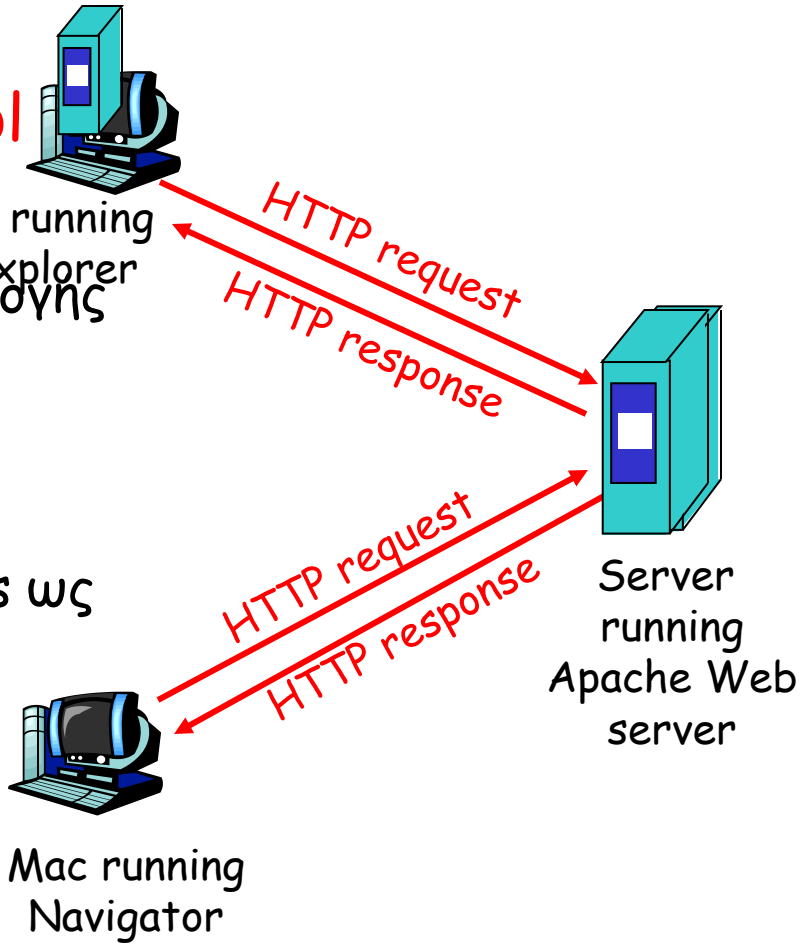
HTTP overview

HTTP: hypertext transfer protocol

- Πρωτόκολλο του Web επιπέδου εφαρμογής
- client/server model
  - *client:* browser που ζητά, λαμβάνει, "εμφανίζει" Web objects
  - *server:* Web server στέλνει objects ως απάντηση σε αιτήματα που λαμβάνει

HTTP 1.0: RFC 1945

HTTP 1.1: RFC 2068



HTTP overview (συνέχεια)

- Χρήση του TCP

Ο client:
ξεκινά μία *TCP σύνδεση*: δημιουργεί socket στο server, port 80

Ο server:
δέχεται τη TCP σύνδεση από τον client

- **Ανταλλαγή HTTP μηνυμάτων** (application-layer protocol messages) μεταξύ browser (HTTP client) & Web server (HTTP server)

- **Κλείσιμο TCP σύνδεσης**

HTTP overview (συνέχεια)

Χρήση του TCP:

- Ο client ξεκινά μία TCP σύνδεση (δημιουργεί socket) στο server, port 80
- Ο server δέχεται τη TCP σύνδεση από τον client
- Ανταλλαγή HTTP μηνυμάτων (application-layer protocol messages) μεταξύ browser (HTTP client) & Web server (HTTP server)
- Κλείσιμο TCP connection

HTTP ***δεν*** διατηρεί πληροφορία σχετικά με την κατάσταση του συστήματος (είναι "***stateless***")

Ο server δεν διατηρεί πληροφορίες από ***παλιότερα αιτήματα*** των clients (requests)

☹ Πρωτόκολλα που διατηρούν **καταστάσεις του συστήματος** ("***state***") είναι **περίπλοκα** !

- Παλιότερη ιστορία (state) πρέπει να διατηρείται &
- Εάν οι server/client καταρρεύσουν ⇨ η εικόνα που έχουν για την κατάσταση τους μπορεί να *διαφέρει* ⇨ το πρωτόκολλο πρέπει να ***εξασφαλίσει*** μια "***κοινή***" εικόνα

HTTP συνδέσεις

Nonpersistent HTTP

- Το *πολύ ένα* αντικείμενο στέλνεται πάνω από *μία* TCP σύνδεση
- HTTP/1.0 χρησιμοποιεί nonpersistent HTTP

Persistent HTTP

☞ **Πολλαπλά** αντικείμενα μπορούν να σταλούν πάνω από *μία* TCP σύνδεση μεταξύ client & server

HTTP/1.1 χρησιμοποιεί persistent συνδέσεις σε default κατάσταση

Nonpersistent HTTP

Υποθέστε ο user βάζει το URL (περιέχει κείμενο, αναφορές σε 10 jpeg εικόνες)

`www.someSchool.edu/someDepartment/home.index`

1a. HTTP client ξεκινάει μια TCP σύνδεση στον HTTP server (διεργασία) @ www.someSchool.edu στο port 80

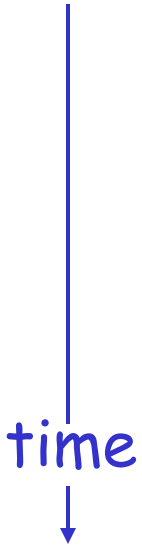
1b. HTTP server στο host www.someSchool.edu περιμένει για TCP σύνδεση στο port 80, "**δέχεται**" **τη σύνδεση, ενημερώνοντας** τον client

2. HTTP client στέλνει HTTP *μήνυμα αιτήματος* (περιλαμβάνοντας το URL) στο socket της TCP σύνδεσης. Το μύνυμα υποδεικνύει ότι ο client ζητάει το αντικείμενο someDepartment/home.index

3. HTTP server λαμβάνει το μήνυμα αιτήματος, κατασκευάζει ένα *μήνυμα απάντησης* περιλαμβάνοντας το ζητούμενο αντικείμενο, & στέλνει τι μήνυμα στο socket

time

Nonpersistent HTTP (συνέχεια)



4. HTTP server κλείνει TCP σύνδεση

5. HTTP client λαμβάνει το μήνυμα απάντησης που περιέχει το html αρχείο, και το δείχνει. Αναλύοντας το html αρχείο, βρίσκει 10 αναφερόμεναr jpeg αντικείμενα

6. Τα βήματα 1-5 επαναλαμβάνονται για κάθε ένα από τα 10 jpeg αντικείμενα

☞ In this example, 11 TCP connections will be created!

Non-Persistent HTTP Connection

- A brand-new **TCP connection** must be established & maintained ***for each requested object***

- For each of these TCP connections:

TCP buffers must be allocated and
TCP variables must be
kept in ***both the client and server***

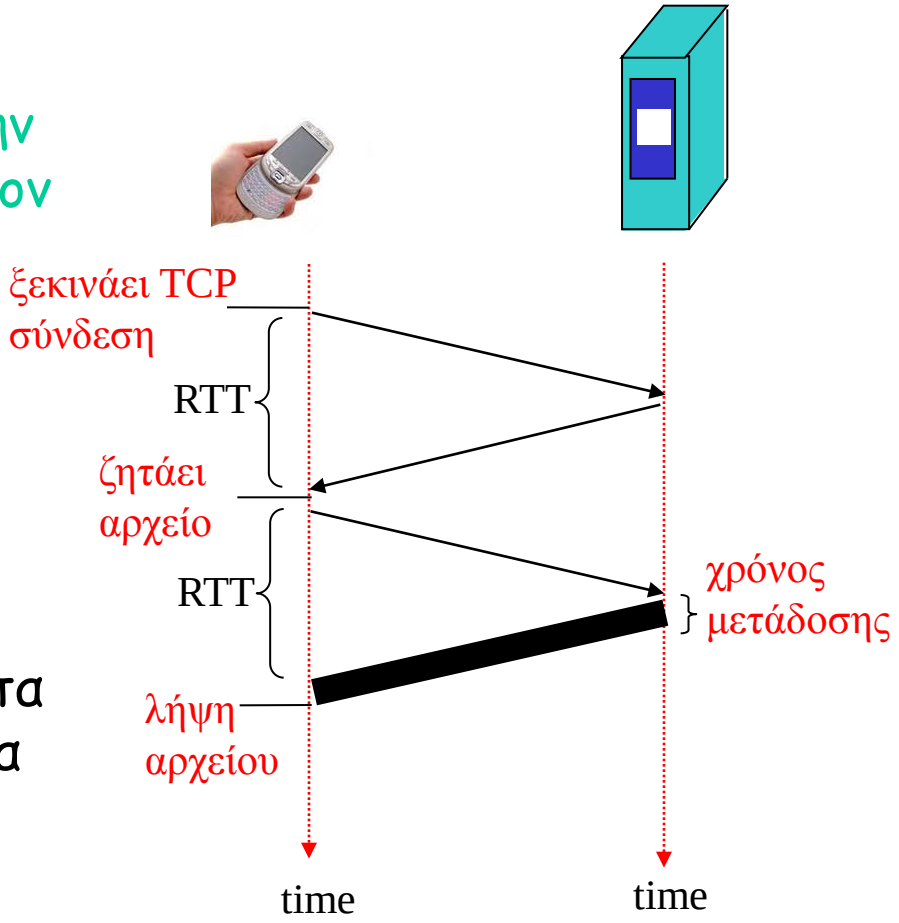
☹ ***This*** can place a serious burden on the Web server

Furthermore, each object suffers a delivery delay of 2RTTs:
one RTT to ***establish the TCP connection***, and
one RTT to **request and receive the object**

Non-Persistent HTTP: Χρόνος απόκρισης

Ορισμός του RRT:

Συνολικός χρόνος που απαιτείται για την αποστολή ενός μικρού πακέτου από τον client στον server και αμέσως μετά από τον server στον client.



Χρόνος απόκρισης:

- ένα RTT για να ξεκινήσει την TCP σύνδεση
- ένα RTT για το HTTP αίτημα και για τα πρώτα bytes τής HTTP απάντησης να επιστρέψει
- Χρόνος για να μεταδοθεί το αρχείο

σύνολο = 2RTT + χρόνος μετάδοσης

Note: RTT includes packet propagation delays, packet queuing delays in intermediate routes and switches, and packet processing delays

Persistent HTTP

Nonpersistent HTTP :

- ☞ απαιτεί 2 RTTs **ανά αντικείμενα**
- OS overhead για ***κάθε*** **TCP σύνδεση**
- browsers συχνά ανοίγουν **παράλληλες TCP** συνδέσεις για να φέρουν αναφερόμενα αντικείμενα

Persistent HTTP

- ο server **αφήνει τη σύνδεση ανοιχτή** μετά την αποστολή της απάντησης
- τα επακόλουθα HTTP μηνύματα μεταξύ ίδιου client/server **στέλνονται πάνω από την ανοιχτή σύνδεση**

Persistent ***χωρίς*** **pipelining**:

- ο client εκδίδει νέο αίτημα ***μόνο*** **όταν η προηγούμενη απάντηση ληφθεί**
- ☞ 1 RTT για ***κάθε*** αναφερόμενο αντικείμενο

Persistent *με pipelining*:

- **default στο HTTP/1.1**
- ο client στέλνει αιτήματα ***μόλις*** **αντιμετωπίσει ένα αναφερόμενο αντικείμενο**
- ☞ 1 RTT για ***όλα*** τα αναφερόμενα αντικείμενα

Persistent HTTP

- An entire web page (in the previous example, the base HTML file and ten images) can be sent over a single persistent TCP connection
- Moreover multiple web pages residing on the same server can be sent from the server to the same client over a single persistent TCP connection
- Typically the HTTP server close a connection when it isn't used for a certain time (a configurable timeout interval)
- With persistent HTTP with pipelining, it is possible for only one RTT to be expended for all the referenced objects, rather than one RTT per referenced object when pipelining isn't used

HTTP μήνυμα αιτήματος

- δύο τύποι HTTP μηνυμάτων: *αίτημα*, απάντηση
- HTTP μήνυμα αιτήματος:

  ASCII (format που διαβάζονται από τον άνθρωπο)

γραμμή αιτήματος (*GET*, *POST*, *HEAD* εντολές)

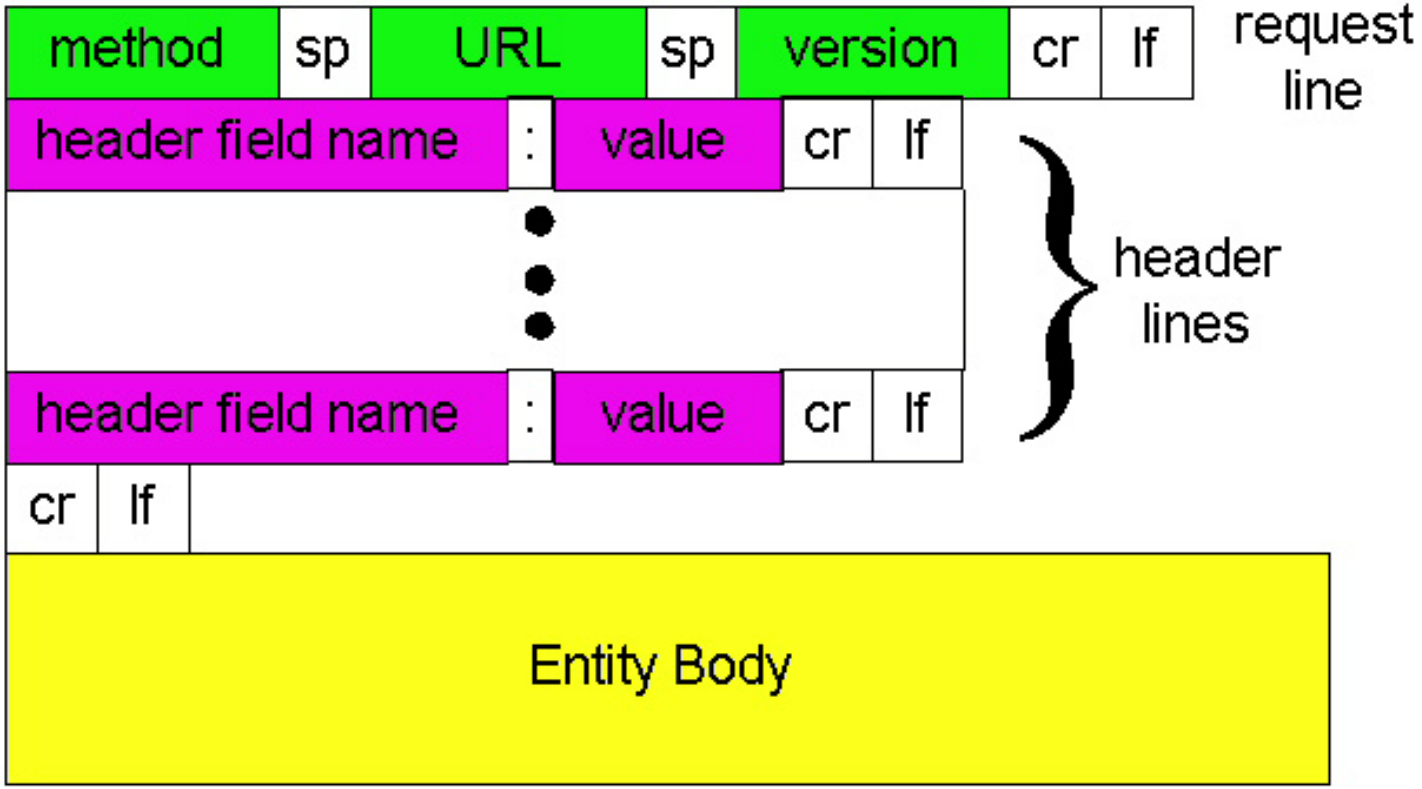
Γραμμές επικεφαλίδων

```
GET /somedir/page.html HTTP/1.1
Host: www.someschool.edu
User-agent: Mozilla/4.0
Connection: close
Accept-language:fr
```

Carriage return, line feed indicates end of message

(extra carriage return, line feed)

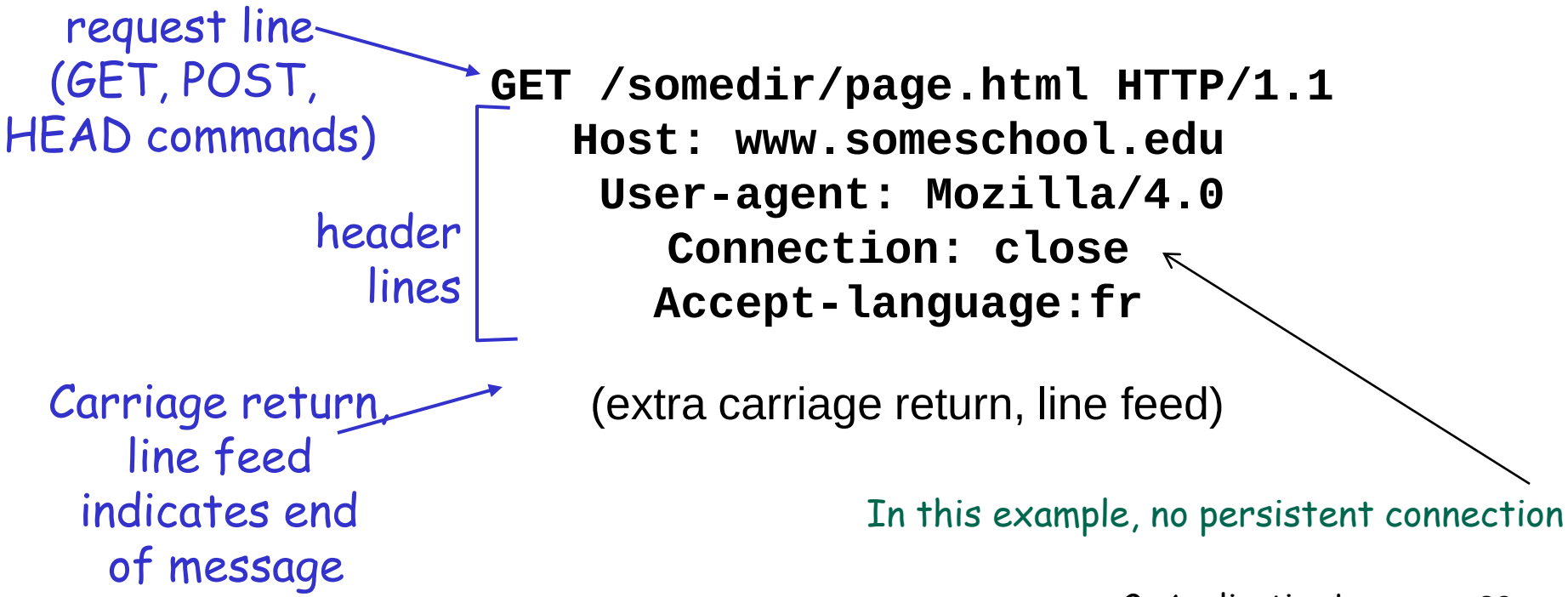
HTTP μήνυμα μηνύματος: γενικό format



HTTP request message

- two types of HTTP messages: *request, response*
- HTTP request message:

  ASCII (human-readable format)



Μοντελοποίηση Καθυστέρησης

Q: Πόσος χρόνος απαιτείται για τη λήψη ενός αντικειμένου από έναν Web server μετά της αποστολή μίας αίτησης;

Αγνοώντας τη συμφόρηση, η καθυστέρηση επηρεάζεται από:

- την **εγκαθίδρυση της TCP σύνδεσης**
- την **καθυστέρηση αποστολής δεδομένων**
- **αργή εκκίνηση**

Συμβολισμοί, παραδοχές:

- Θεωρείστε μία ζεύξη μεταξύ του client και του server ρυθμού R
- S: MSS (bits)
- O: μέγεθος object (bits)
- Δεν υφίστανται επαναποστολές(δεν υπάρχουν απώλειες (loss) ή αλλοιώσεις (corruption))

Μέγεθος παραθύρου:

- Καταρχήν, θεωρείστε σταθερό παράθυρο συμφόρησης, W segments
- Στη συνέχεια, δυναμικό παράθυρο, μοντελοποιώντας την αργή εκκίνηση.

Μοντελοποίηση HTTP Καθυστέρησης

Assume Web page consists of:

- *1* base HTML page (of size *O* bits)
- *M* images (each of size *O* bits)

Non-persistent HTTP:

- *M+1* TCP connections in series
- *Response time = (M+1)O/R + (M+1)2RTT + sum of idle times*

Persistent HTTP with pipeling:

- *2 RTT* to request and receive base HTML file
- *1 RTT* to request and receive M images
- *Response time = (M+1)O/R + 3RTT + sum of idle times*

Uploading form input

Post method:

Still the user requests a Web page

- ☞ Used when the user fills out a form (e.g., user provides search words to a search engine)

- Web page often includes form input
- Input is uploaded to server in entity body

URL method:

- Uses GET method
- Input is uploaded in URL field of request line:

`www.somesite.com/animalsearch?monkeys&banana`

Method types

HTTP/1.0

- GET
- POST
- HEAD
  - asks server to leave requested object out of response
  - Used for debugging

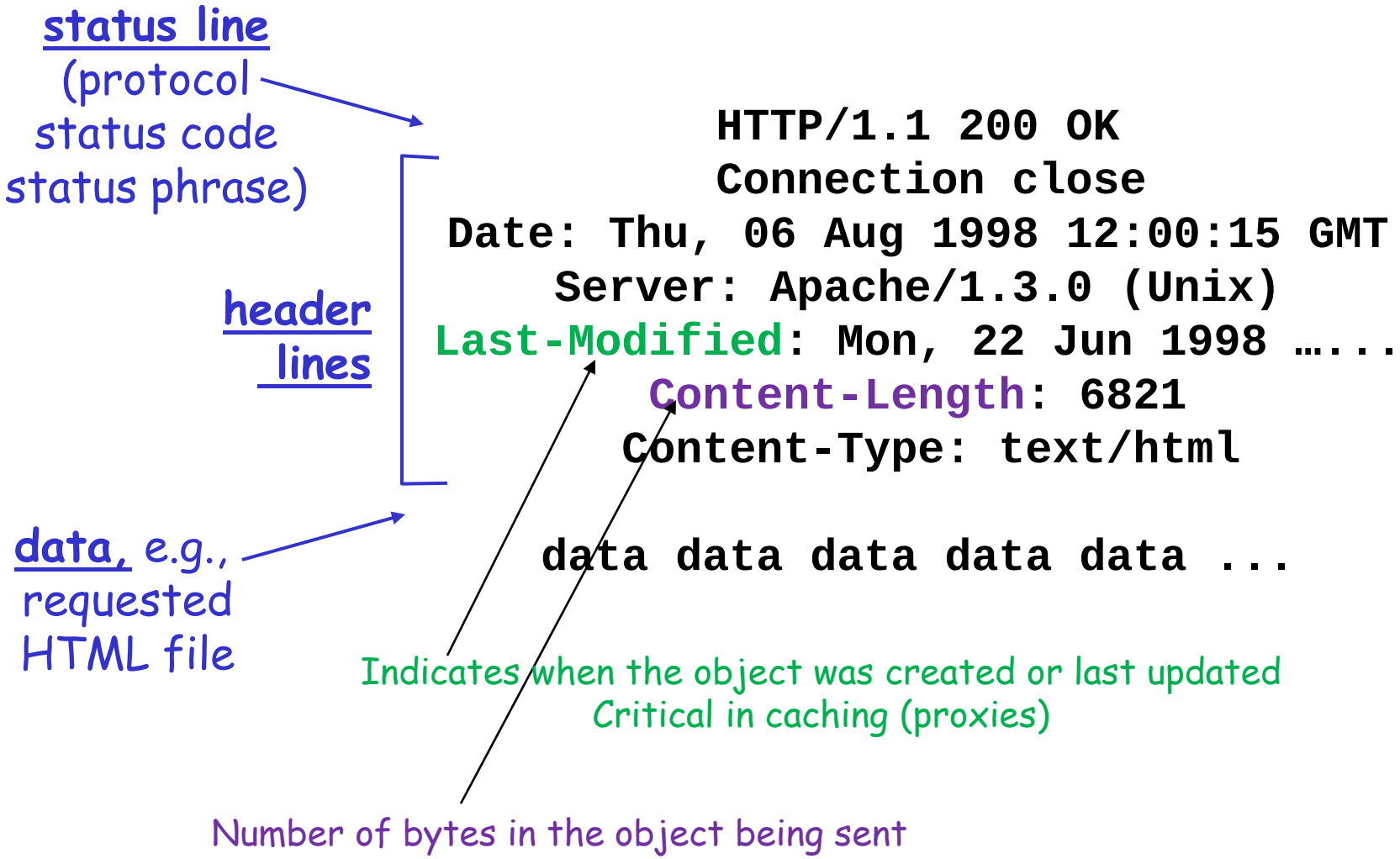
HTTP/1.1

- GET, POST, HEAD
- PUT

  uploads file in entity body to path specified in URL field
- DELETE

  deletes file specified in the URL field

HTTP response message



HTTP response status codes

In first line in server->client response message.
A few sample codes:

200 OK

request succeeded, requested object later in this message

301 Moved Permanently

requested object moved, new location specified later in this message (Location:)

400 Bad Request

request message not understood by server

404 Not Found

requested document not found on this server

505 HTTP Version Not Supported

401 Authorization Required

the user needs to provide a user name/password

HTTP & statefullness

- Simplified HTTP servers are stateless
- Often desirable for a web site to identify users because the server wishes to **restrict user access** or to **serve content as a function of user identity**

Mechanisms to achieve that:

- Authorization
- Cookies

Trying out HTTP (client side) for yourself

1. Telnet to your favorite Web server:

```
telnet cis.poly.edu 80
```

Opens TCP connection to port 80 (default HTTP server port) at cis.poly.edu. Anything typed in sent to port 80 at cis.poly.edu

2. Type in a GET HTTP request:

```
GET /~ross/ HTTP/1.1
Host: cis.poly.edu
```

By typing this in (hit carriage return twice), you send this minimal (but complete) GET request to HTTP server

3. Look at response message sent by HTTP server!

User-server state: cookies

Many major Web sites use cookies

Four components:

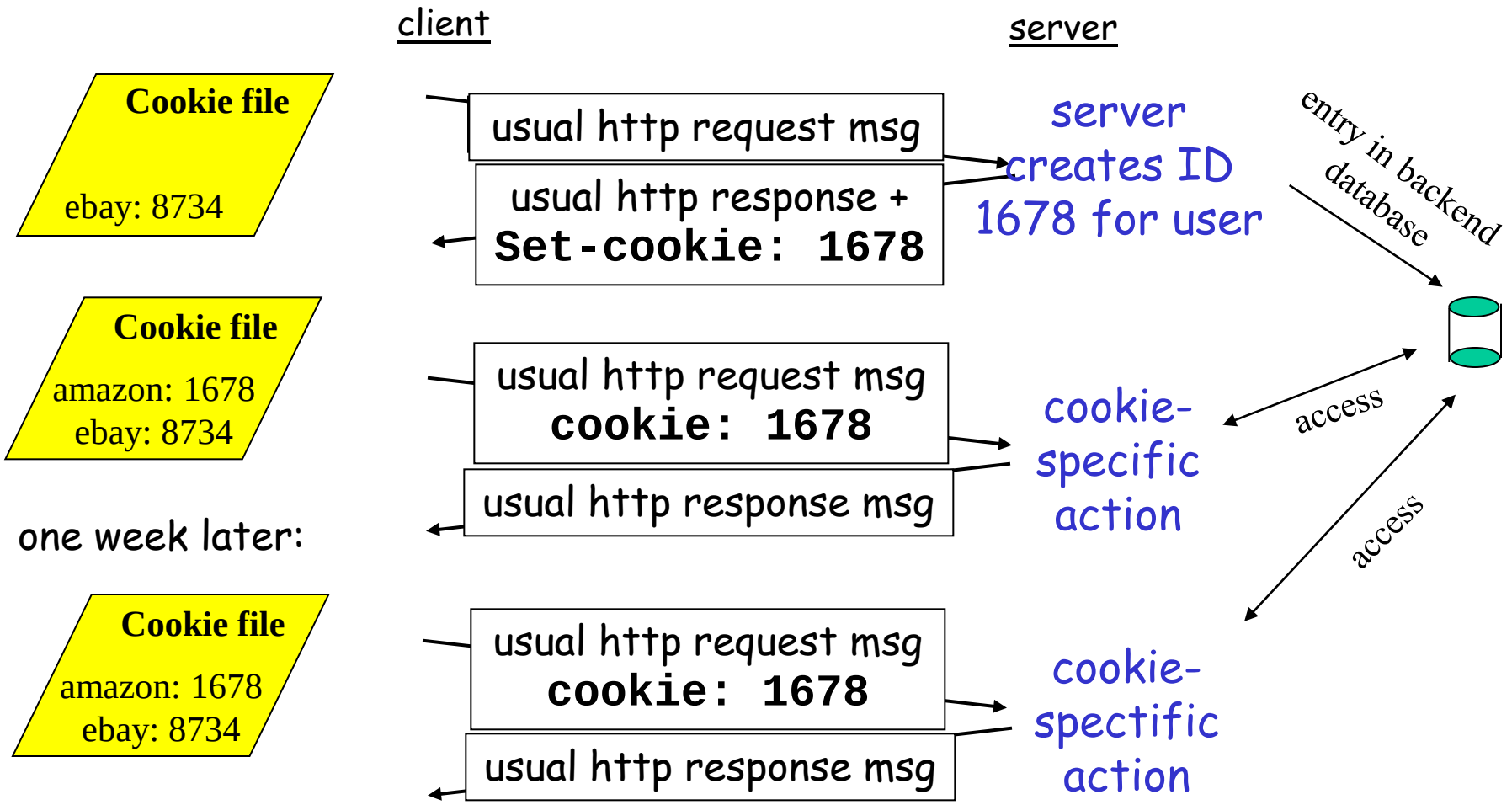
1) **cookie header line** of HTTP *response* **message**
2) cookie header line in HTTP *request* **message**
3) cookie **file kept on user's host**, managed by user's browser
4) **back-end database @ Web site**

Example:

- Susan access Internet always from same PC
- She visits a specific e-commerce site for first time
- When initial HTTP requests arrives at site, site creates a unique ID and creates an entry in backend database for ID

Cookies: keeping "state" (cont.)

There is an identification number for a web site
the browser consults this identification number



Cookies (continued)

What cookies can bring:

- authorization
- shopping carts
- recommendations
- user session state (Web e-mail)

Cookies are controversial!

Privacy issues

aside

Cookies and privacy:

- ☐ cookies permit sites to learn a lot about you
- ☐ you may supply name and e-mail to sites
- ☐ search engines use redirection & cookies to learn yet more
- ☐ advertising companies obtain info across sites

HTTP Content

- Data carried in HTTP response messages are objects from web pages, that is, HTML files, GIFs, JPEGs, java applets, XML files
- XML files are structured data often used in electronic commerce applications (and not only)
- HTTP is also used as the file transfer protocol for peer-to-peer applications, streaming stored audio and video content

Caching

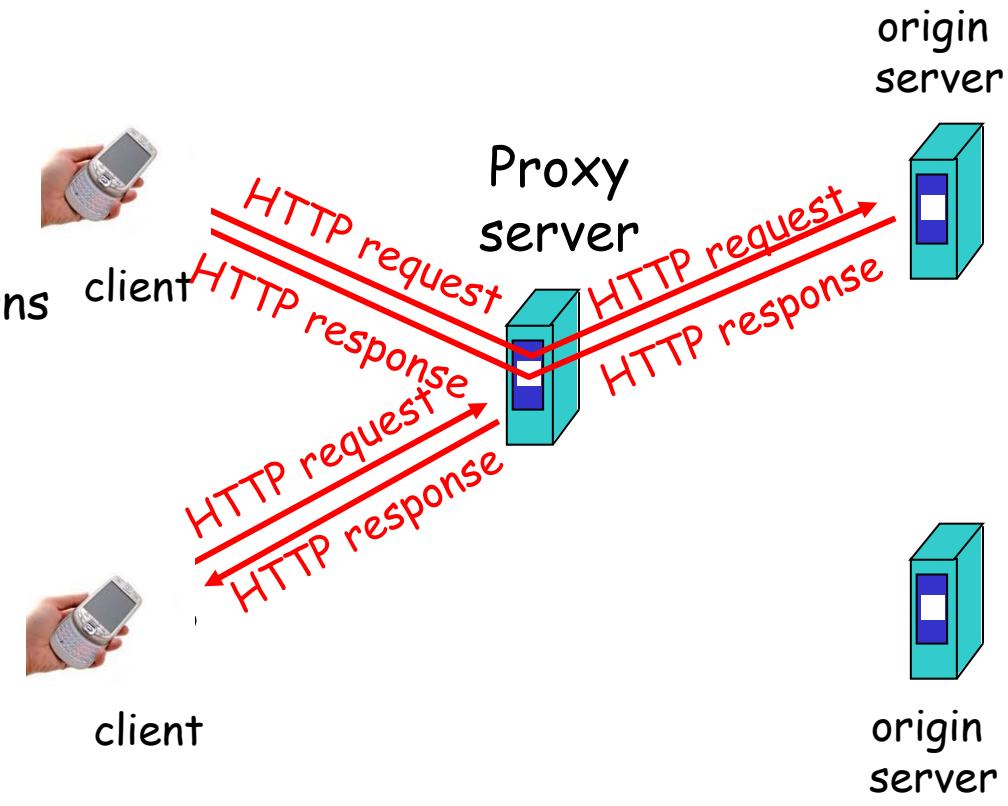
- Caches can **reduce object-retrieval delays**
- Caches can **decrease the amount of Web traffic** sent over the Internet
- Caches introduces a problem -> **a cache may be stale!**

- To ensure that all objects passed to the web browser are up-to-date: **Conditional GET**
- Request message uses **GET method**
- Request message includes an **If-Modified-Since** header line
- Check with the **Last-modified & If-Modified-Since** fields
- The **web server** sends the object ***only if*** the object has been modified since the specified date

Web caches - proxy server

Goal: satisfy client request ***without*** involving origin server

- user sets browser: Web accesses via cache
- browser sends **all HTTP requests to cache**
  - object in cache: cache returns object
  - else cache requests object from origin server, then returns object to client



More about Web caching

- Cache acts as both client and server
- Typically cache is **installed by ISP** (university, company, residential ISP)

Why Web caching?

- **Reduce response time** for client request
- **Reduce traffic** on an institution's access link
- Internet dense with caches enables "poor" content providers to effectively deliver content (but so does P2P file sharing)

Caching example

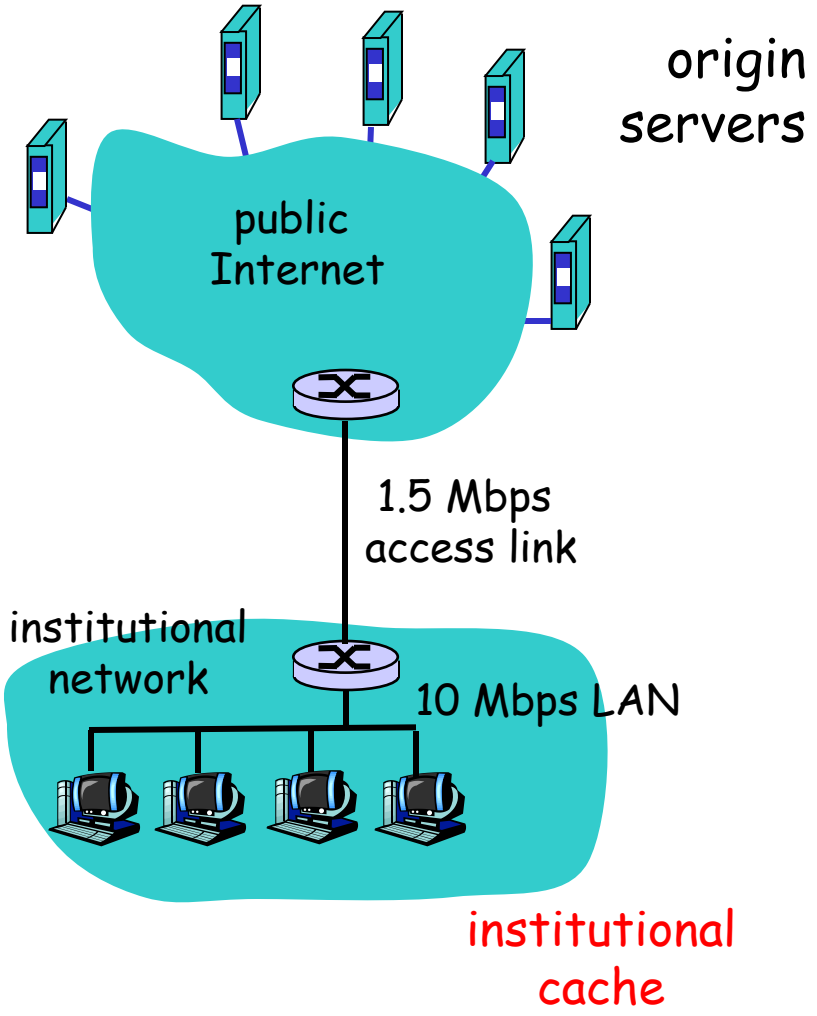
Assumptions

- average object size = 100,000 bits
- avg. request rate from institution's browsers to origin servers = 15/sec
- delay from institutional router to any origin server and back to router = 2 sec

Consequences

- utilization on LAN = 15%
- utilization on access link = 100%
- total delay = Internet delay + access delay + LAN delay

= 2 sec + minutes + milliseconds



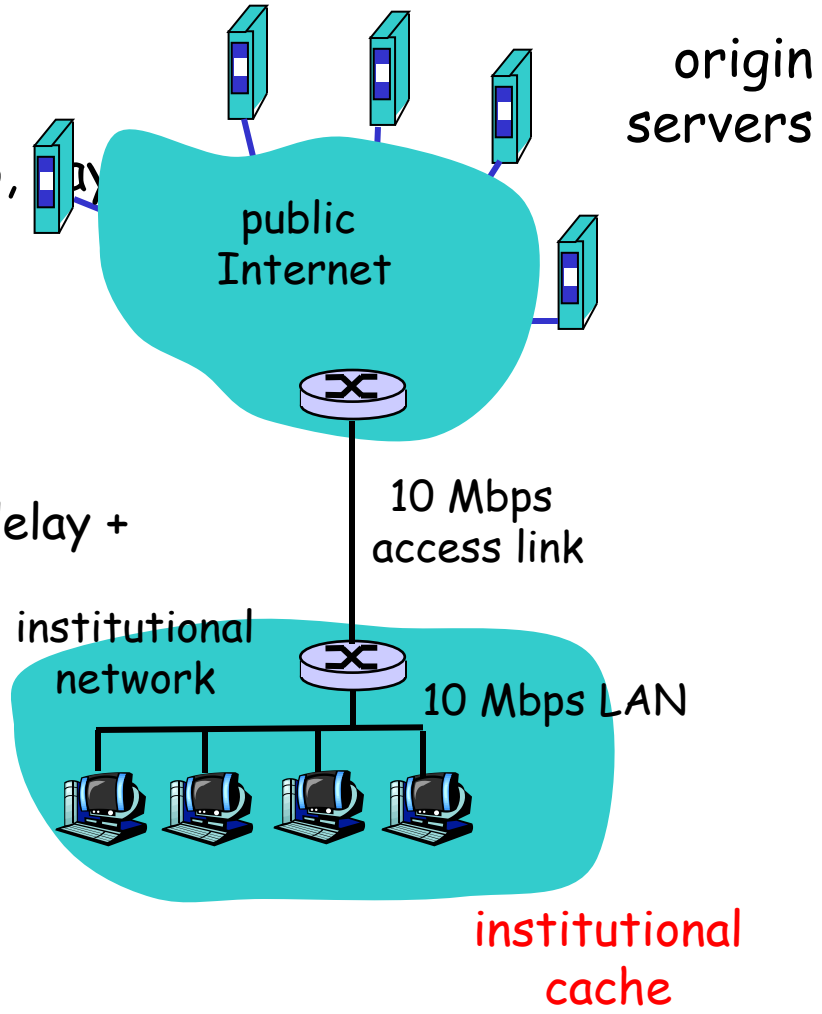
Caching example (cont)

Possible solution

- increase bandwidth of access link to, say, 10 Mbps

Consequences

- utilization on LAN = 15%
- utilization on access link = 15%
- Total delay = Internet delay + access delay + LAN delay
- = 2 sec + msecs + msecs
- often a costly upgrade



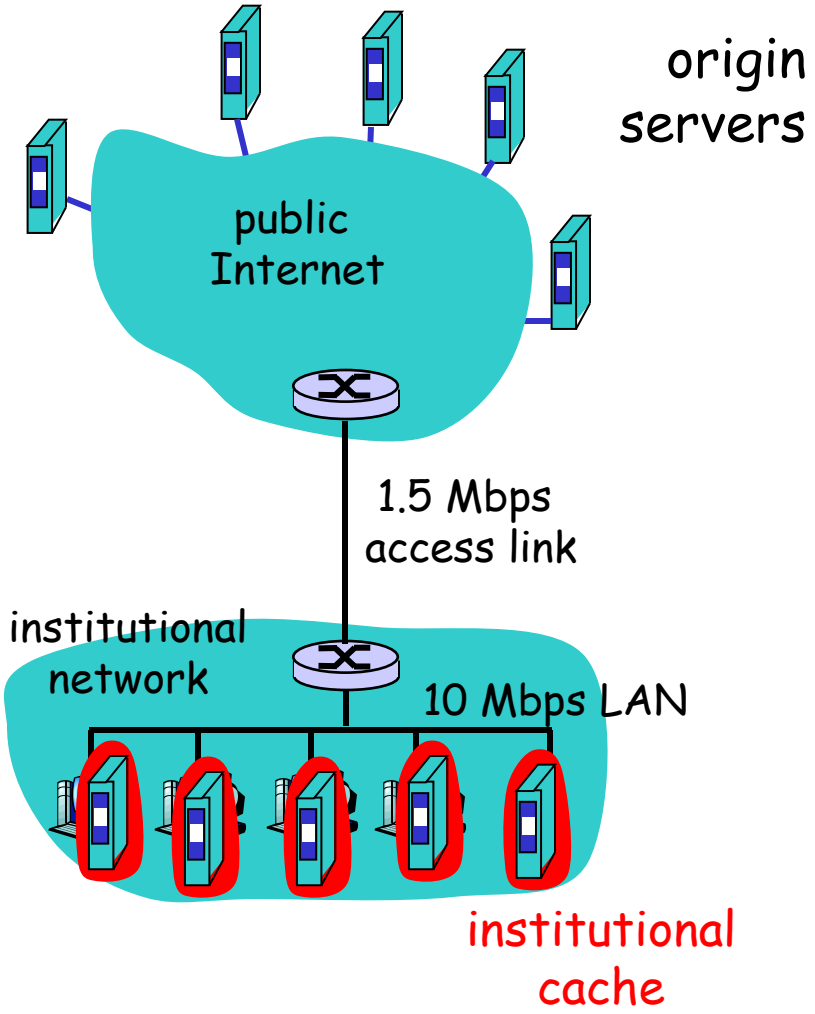
Caching example (cont)

☞ Install cache

suppose hit rate is .4

Consequence

- 40% requests will be satisfied almost immediately
- 60% requests satisfied by origin server
- utilization of access link reduced to 60%, resulting in negligible delays (say 10 msec)
- total avg delay = Internet delay + access delay + LAN delay = .6*(2.01) secs + .4*milliseconds < 1.4 secs



DNS: The Internet's Directory Service

Hostnames are identifiers of hosts

eg,www.yahoo.com,calliope.ics.forth.gr

However

- they do not provide information about the network location of the hos
- Difficult to be processed by routers

⇨ Hosts can be also identified by *IP addresses*

1. Αποτελείται από 4 bytes
2. Έχει αυστηρή ιεραρχική δομή
3. Αποκαλύπτει πληροφορία για τη θέση του host στο Internet

Το DNS μεταφράζει (αντιστοιχεί) το **hostname σε IP address**

Application-layer protocol όπου οι DNS name servers επικοινωνούν γ
να εκτελέσουν τη **«μετάφραση» του hostname σε IP address**

DNS: problems w/ centralized architecture

- **A single point of failure**
- **Μεγαλύτερος φόρτος από DNS queries**

Θα πρέπει να εξυπηρετεί όλα τα DNS queries που δημιουργούνται από εκατοντάδες χιλιάδες συσκευές

- **Μεγαλύτερες καθυστερήσεις**

  Απόσταση από μια κεντρικοποιημένη database που συνεισφέρει σε μεγαλύτερες καθυστερήσεις λόγω της μεγάλης απόστασης της από τα hosts

- **Αυξημένο κόστος συντήρησης**

Θα έχομε μια τεράστια database για ολόκληρο το Internet

Υπάρχουν και authentication & authorization θέματα όταν επιτρέπουμε οποιοδήποτε χρήστη να «εγγράφει» ένα host με την κεντρικοποιημένη database

Αρχιτεκτονική του DNS

- *Κατανεμημένη* database από name servers σε *ιεραρχική δομή*
- **Application-layer** protocol όπου οι name servers επικοινωνούν για να εκτελέσουν τη **«μετάφραση» του hostname σε IP address**
- Το πρωτόκολλο προσδιορίζει τον τρόπο επικοινωνίας
- μεταξύ των hosts που στέλνουν queries και των name servers που απαντούν

☞ Χρησιμοποιεί κυρίως UDP

Αρχιτεκτονική του DNS

Local name servers

- Each ISP (e.g., university, company, residential ISP) has a local name server (default name server)
- When a host issues a DNS query message, the message is first sent to the host's local name server
- The IP address of the local server is typically configured by hand in a host

e.g., Run in the same LAN as the client host

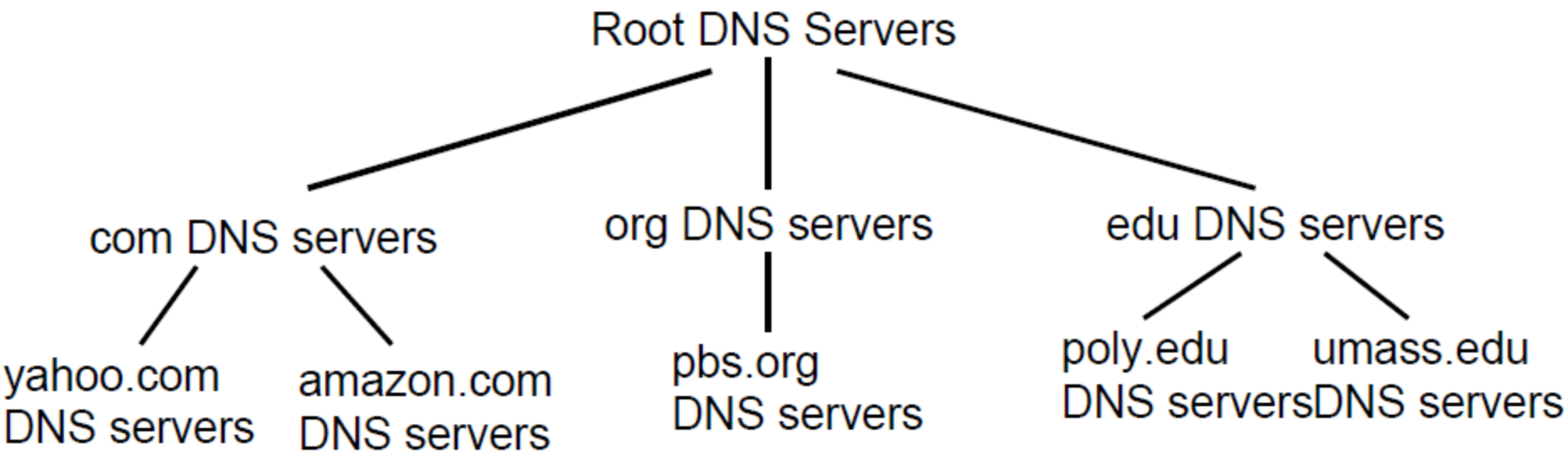
Root name servers

When a local name server cannot immediately satisfy a query, the local name server behaves as a DNS client and queries one of the root name servers

Authoritative name servers

- A name server in the host's local ISP
- (def.) a name server is authoritative if it always has a DNS record that translates the host's hostname to that host's IP address

Distributed, Hierarchical Database

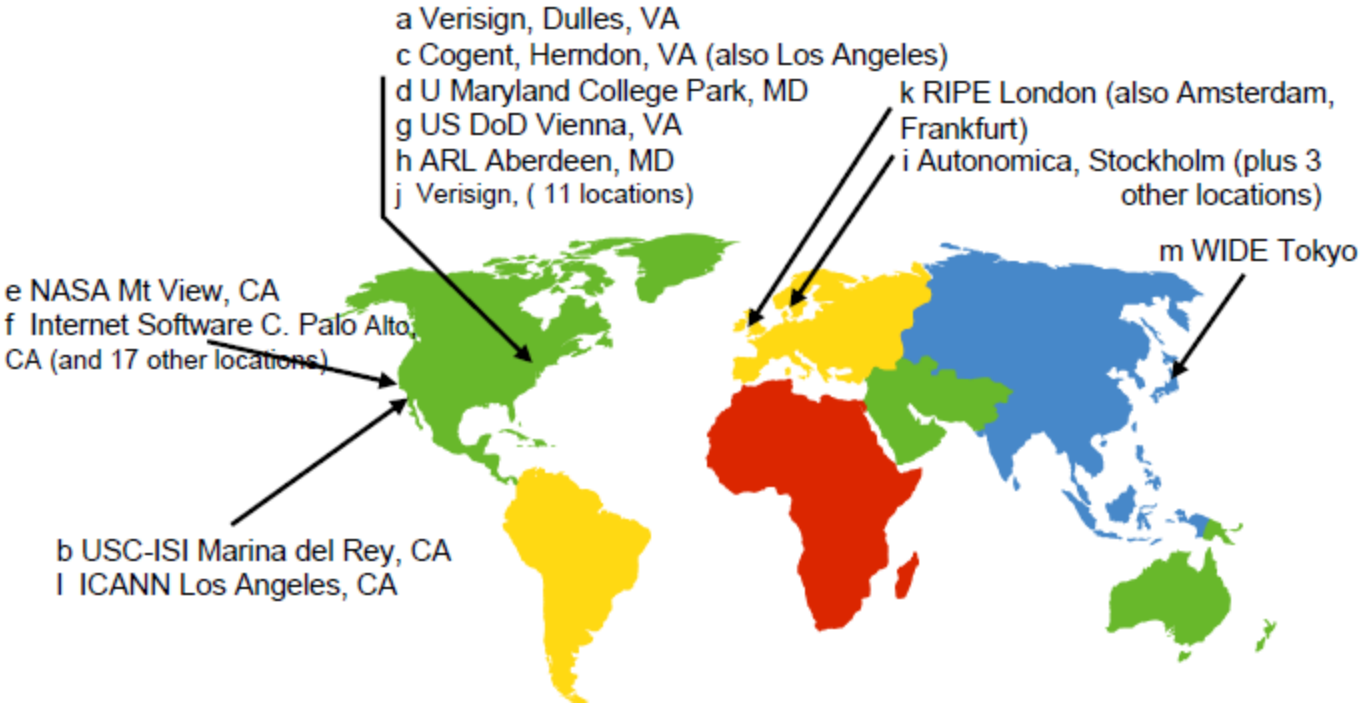


Client wants IP for www.amazon.com; 1st approx:

- Client queries a root server to find com DNS server
- Client queries com DNS server to get amazon.com DNS server
- Client queries amazon.com DNS server to get IP address for www.amazon.com

DNS: Root name servers

- contacted by local name server that can not resolve name
- root name server:
  - contacts authoritative name server if name mapping not know
  - gets mapping
  - returns mapping to local name server



13 root name servers worldwi

TLD and Authoritative Servers

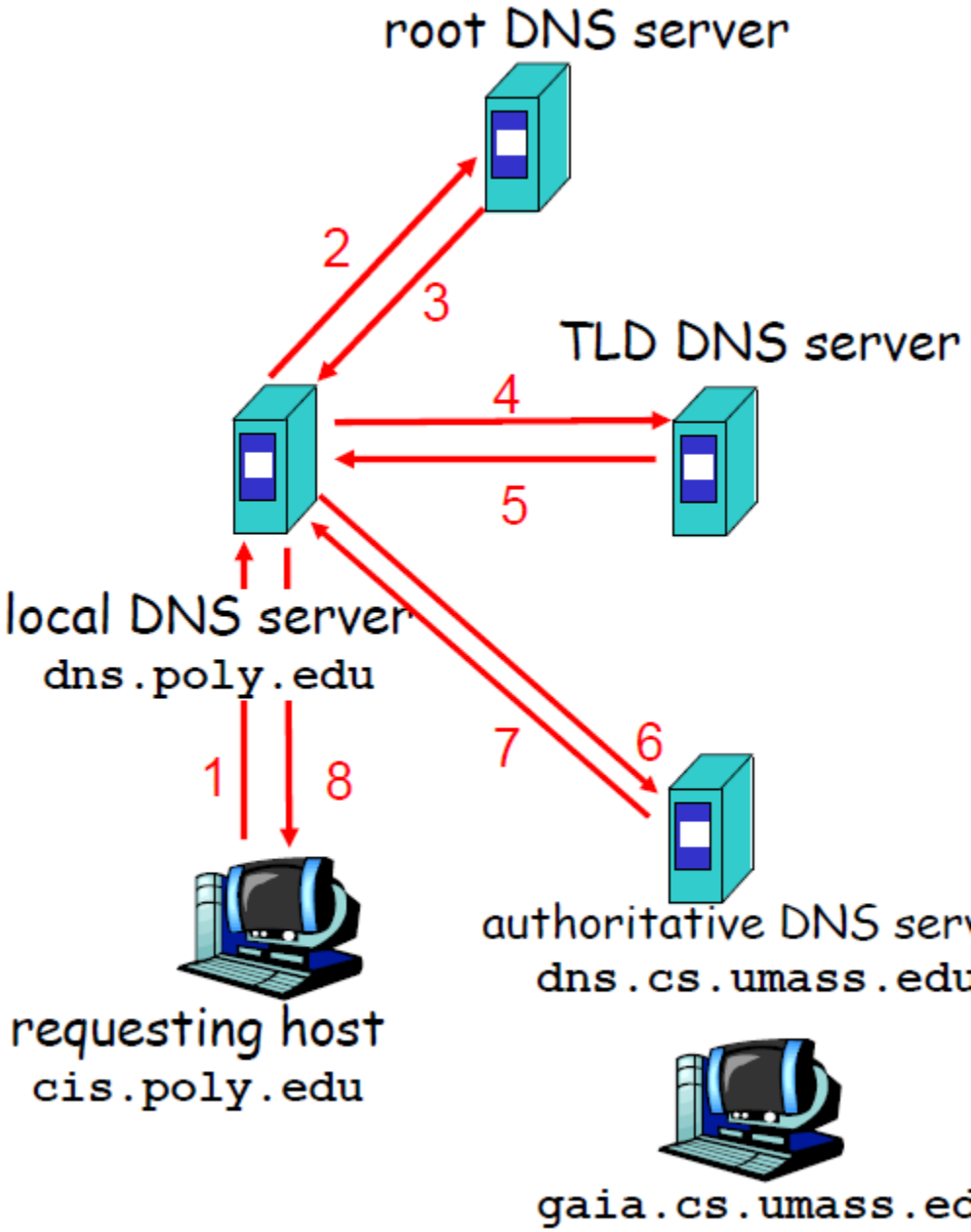
- Top-level domain (TLD) servers: responsible for com, org, net, edu, etc, and all top-level country domains uk, fr, ca, jp.
  - Network solutions maintains servers for com TLD
  - Educause for edu TLD
- Authoritative DNS servers: organization's DNS servers, providing authoritative hostname to IP mappings for organization's servers (e.g., Web and mail).
  - Can be maintained by organization or service provider

Local Name Server

- Does not strictly belong to hierarchy
- Each ISP (residential ISP, company, university) has one.
  - Also called "default name server"
- When a host makes a DNS query, query is sent to its local DNS server
  - Acts as a proxy, forwards query into hierarchy.

Example

- Host at cis.poly.edu wants IP address for gaia.cs.umass.edu



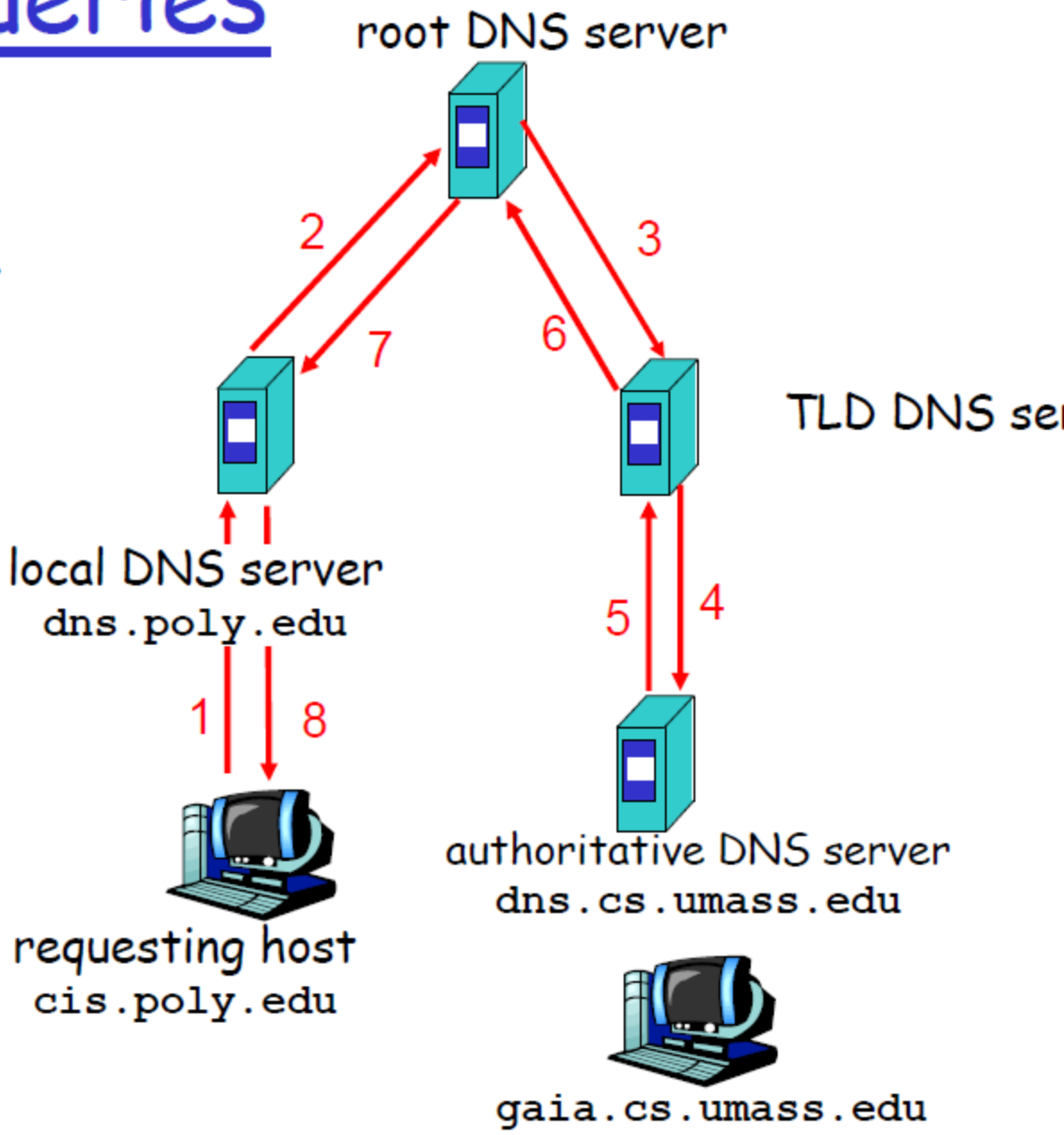
Recursive queries

recursive query:

- puts burden of name resolution on contacted name server
- heavy load?

iterated query:

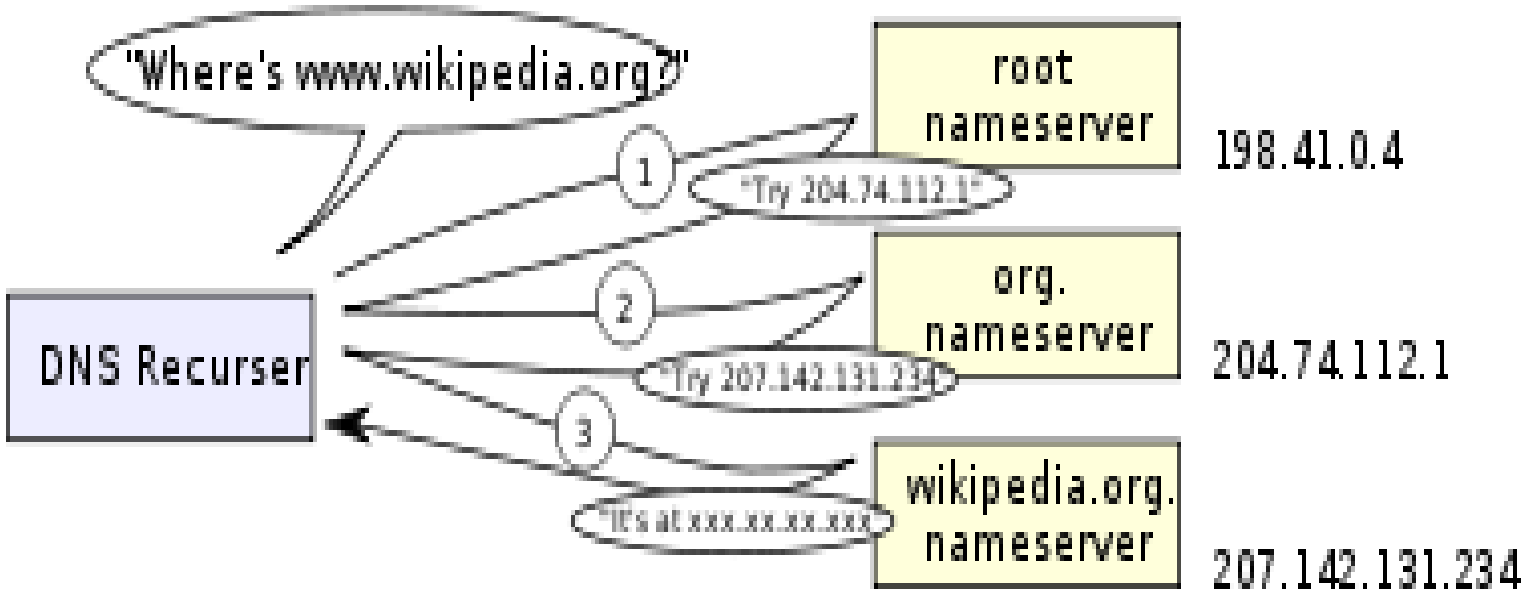
- contacted server replies with name of server to contact
- "I don't know this name, but ask this server"



Another Example on DNS

☞ Domain name resolvers determine the appropriate domain name servers responsible for the domain name in question by a sequence of queries starting with the right-most (top-level) domain label.

DNS recursor consults three name servers to resolve the address www.wikipedia.org.



Another Example on DNS (cont'd)

The process entails:

1 > A network host is configured with an initial cache of the known addresses of ***the root names servers***.

Such a *file* is updated periodically by an **administrator from a reliable source.**

2 > A query to one of the root servers to find the **server authoritative for the *top-level* domain**

3 > A query to the obtained TLD server for the address of a DNS **server authoritative for the *second-level* domain.**

4 > Repetition of the previous step to process **each domain name label in sequence,**

until the final step which returns the IP address of the host sought.

DNS: λειτουργία (1/5)

- Χρησιμοποιείται από πολλά πρωτόκολλα στο application layer
- Ο browser "βρίσκει" το hostname (e.g., www.someschool.edu) στο URL και «προωθεί» το hostname στο **DNS client** που τρέχει στο τοπικό μηχάνημα.
- Ο **DNS client** στέλνει ένα **DNS query** που περιέχει το hostname σε ένα DNS server
- Ο DNS client θα λάβει τελικά μια απάντηση που περιέχει το IP address για το hostname
- Ο browser θα ανοίξει μετά μια TCP σύνδεση στη διεργασία του HTTP server που βρίσκεται σε αυτή την IP διεύθυνση
- Το DNS συνεισφέρει με μια καθυστέρηση …
- Η IP address μπορεί να βρίσκεται cached σε γειτονικό DNS name server, το οποίο θα ελαττώσει την συνολική μέση καθυστέρηση λόγω του DNS

DNS: caching and updating records (2/5)

Once (any) name server learns mapping, it *caches* mapping

1. **cache entries timeout** (disappear) *after some time*
2. TLD servers typically cached in local name servers
   - Thus root name servers not often visited

update/notify mechanisms under design by IETF

RFC 2136

http://www.ietf.org/html.charters/dnsind-charter.html

DNS records (3/5)

DNS: distributed db storing resource records (RR)

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

- Type=A
  - **name** is hostname
  - **value** is IP address

- Type=NS
  - **name** is domain (e.g. foo.com)
  - **value** is hostname of authoritative name server for this domain

- Type=CNAME
  - **name** is alias name for some "canonical" (the real) name `www.ibm.com` is really `servereast.backup2.ibm.com`
  - **value** is canonical name

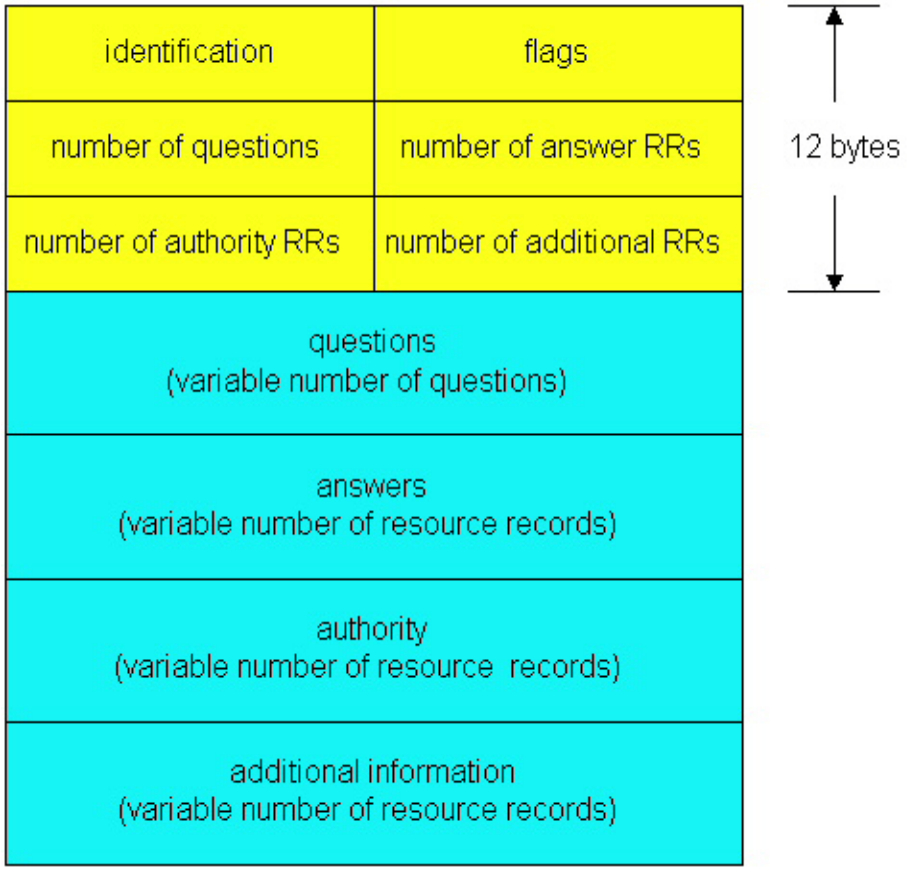
- Type=MX
  - **value** is name of mailserver associated with **name**

DNS protocol, messages (4/5)

DNS protocol : *query* and *reply* messages, both with same *message format*

msg header

- identification: 16 bit # for query, reply to query uses same #
- flags:
  - query or reply
  - recursion desired
  - recursion available
  - reply is authoritative



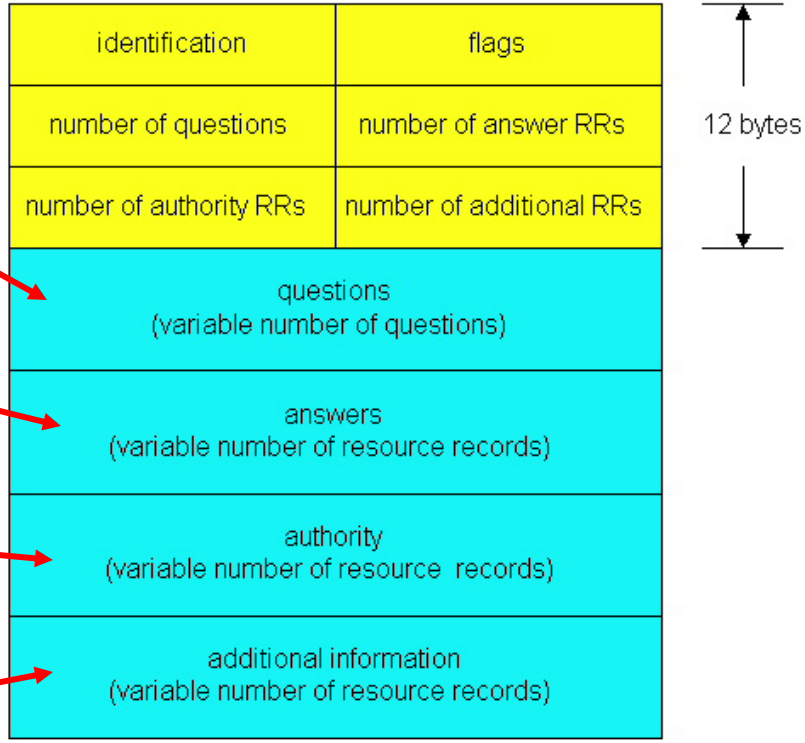
DNS protocol, messages (5/5)

Name, type fields for a query

RRs in response to query

records for authoritative servers

additional "helpful" info that may be used



DNS: load distribution

- Το DNS βοηθά στην **κατανομή φόρτου σε «αντίγραφα» των servers** (replicated servers)
- Ένα δημοφιλές site μπορεί να διατηρεί μια «φάρμα» από servers, το καθένα με διαφορετική IP address
- DNS **γνωρίζει το σύνολο αυτών των IP addresses**
- Όταν ο DNS client στέλνει μια query για ένα όνομα που αντιστοιχεί σε ένα σύνολο διευθύνσεων, ο DNS server απαντά ***στέλνοντας όλες τις IP addresses***, αλλά ***αλλάζοντας τη σειρά τους***
- Ο client συνήθως στέλνει το TCP connection στην πρώτη IP address και με αυτό τον τρόπο κατανέμεται το φορτίο σε όλους τους replicated servers

DNS: recursive & iterative queries

- When a host or name server A makes a recursive query to a name server B, then name server B obtains the requested mapping on behalf of A and then forwards the mapping to A
- The DNS protocol also allows for ***iterative queries*** at any step in the chain between requesting host and authoritative name server
- When a name server A makes an iterative query to name server B, if name server B does not have the requested mapping, it immediately sends a DNS reply to A that contains the IP address of the next name server in the chain, say name server C
- DNS *caching* is used to improve delay performance

Electronic Mail

Three major components:

- user agents
- mail servers
- simple mail transfer protocol: SMTP

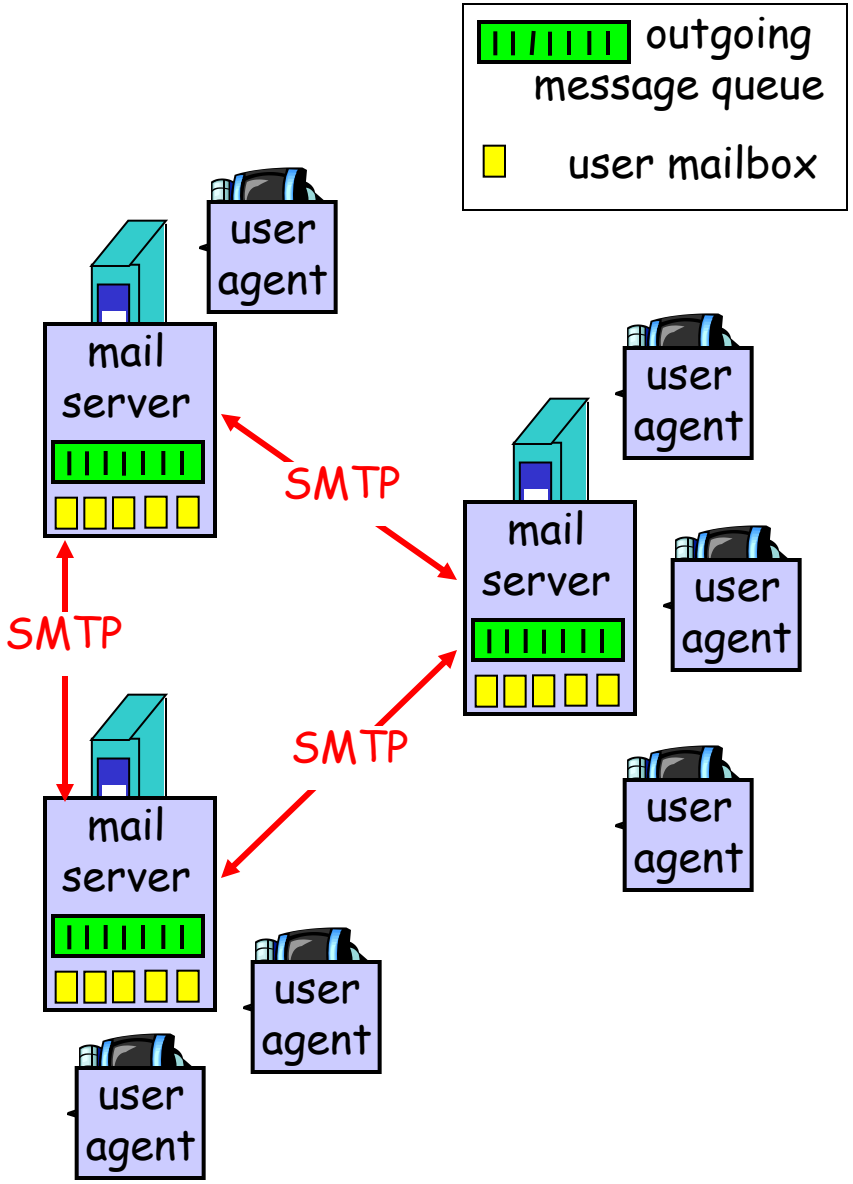
User Agent

a.k.a. "mail reader"

composing, editing, reading mail messages

e.g., Eudora, Outlook, elm, Netscape Messenger

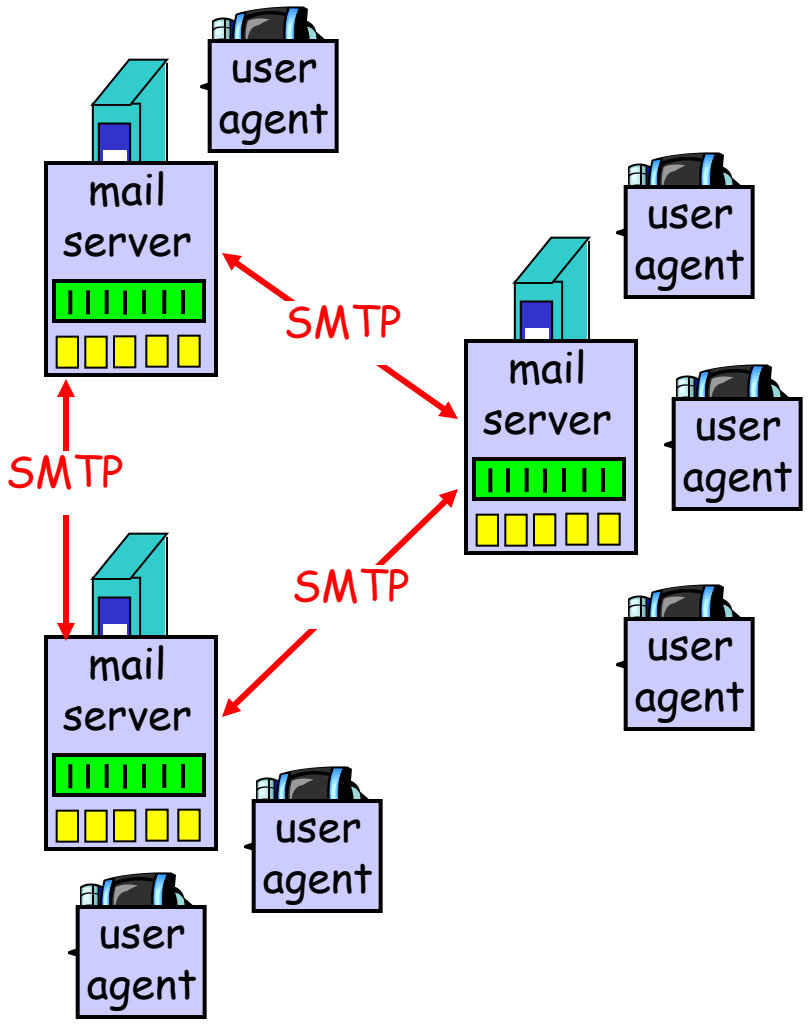
outgoing, incoming messages stored on server



Electronic Mail: mail servers

Mail Servers

- mailbox contains incoming messages for user
- message queue of outgoing (to be sent) mail messages
- SMTP protocol between mail servers to send email messages
  - client: sending mail server
  - "server": receiving mail server



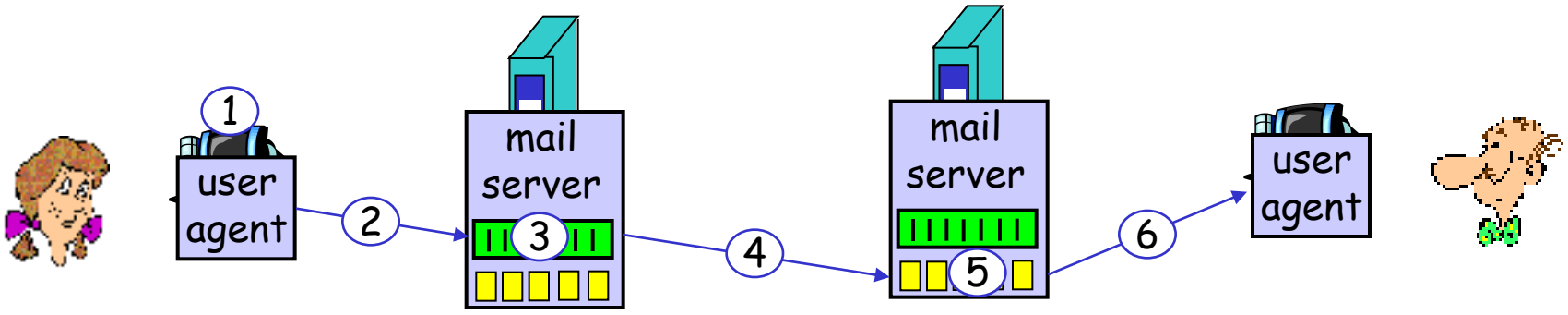
Electronic Mail: SMTP [RFC 2821]

- uses TCP to reliably transfer email message from client to server port 25
- direct transfer: sending server to receiving server
- three phases of transfer
  - handshaking (greeting)
  - transfer of messages
  - closure
- command/response interaction
  - commands: ASCII text
  - response: status code and phrase

messages must be in 7-bit ASCII

Scenario: Alice sends message to Bob

1) Alice uses UA to compose message and "to" bob@someschool.edu
2) Alice's UA sends message to her mail server; message placed in message queue
3) Client side of SMTP opens TCP connection with Bob's mail server
4) SMTP client sends Alice's message over the TCP connection
5) Bob's mail server places the messag in Bob's mailbox
6) Bob invokes his user agent to read message



Sample SMTP interaction

```
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
```

Try SMTP interaction for yourself:

- **`telnet servername 25`**
- see 220 reply from server
- enter HELO, MAIL FROM, RCPT TO, DATA, QUIT commands

above lets you send email without using email client (reader)

SMTP: final words

- SMTP uses persistent connections
- SMTP requires message (header & body) to be in 7-bit ASCII
- SMTP server uses `CRLF.CRLF` to determine end of message

Comparison with HTTP:

- HTTP: pull
- SMTP: push
- both have ASCII command/response interaction, status codes
- HTTP: each object encapsulated in its own response msg
- SMTP: multiple objects sent in multipart msg