DIPARTIMENTO
DI INFORMATICA
Via Salaria 113 - 00198, Roma

SAPIENZA
UNIVERSITÀ DI ROMA

Chapter 2
Application Layer

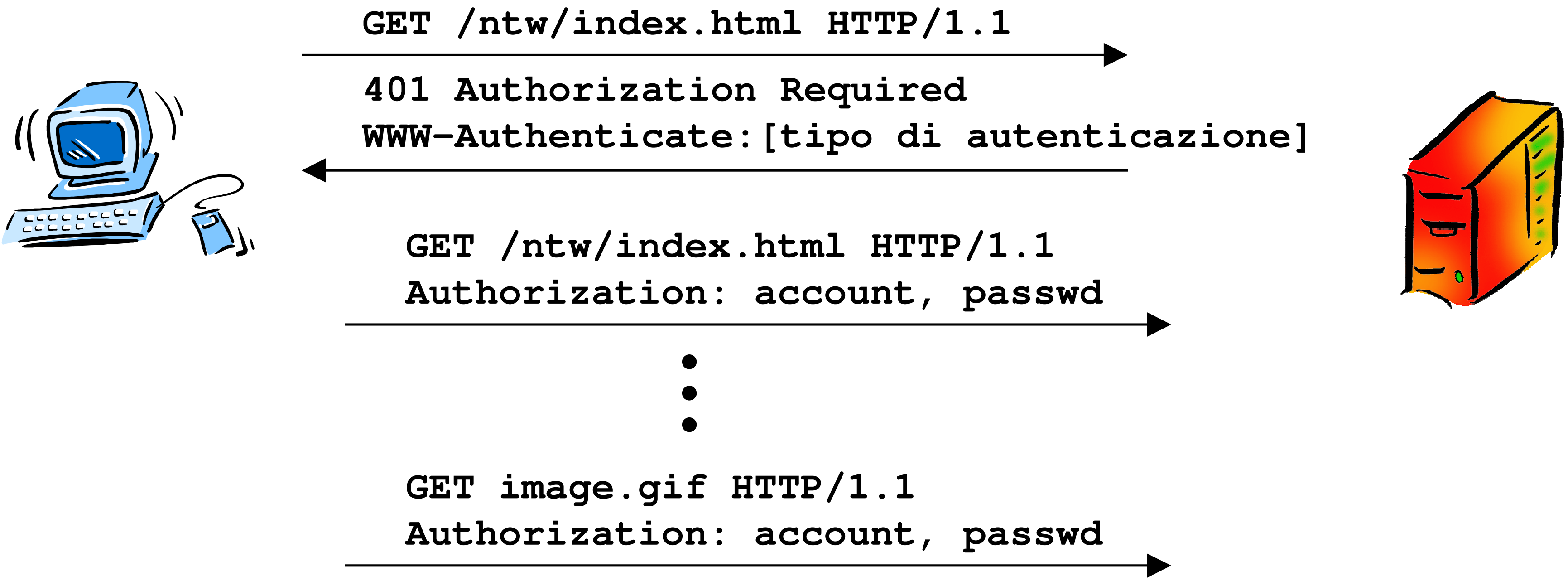
Reti di Elaboratori
Corso di Laurea in Informatica
Università degli Studi di Roma "La Sapienza"
Canale A-L
Prof.ssa Chiara Petrioli

Parte di queste slide sono state prese dal materiale associato al libro *Computer Networking: A Top Down Approach* , 5th edition.

Thanks also to Antonio Capone, Politecnico di Milano, Giuseppe Bianchi and Francesco LoPresti, Un. di Roma Tor Vergata

Autenticazione

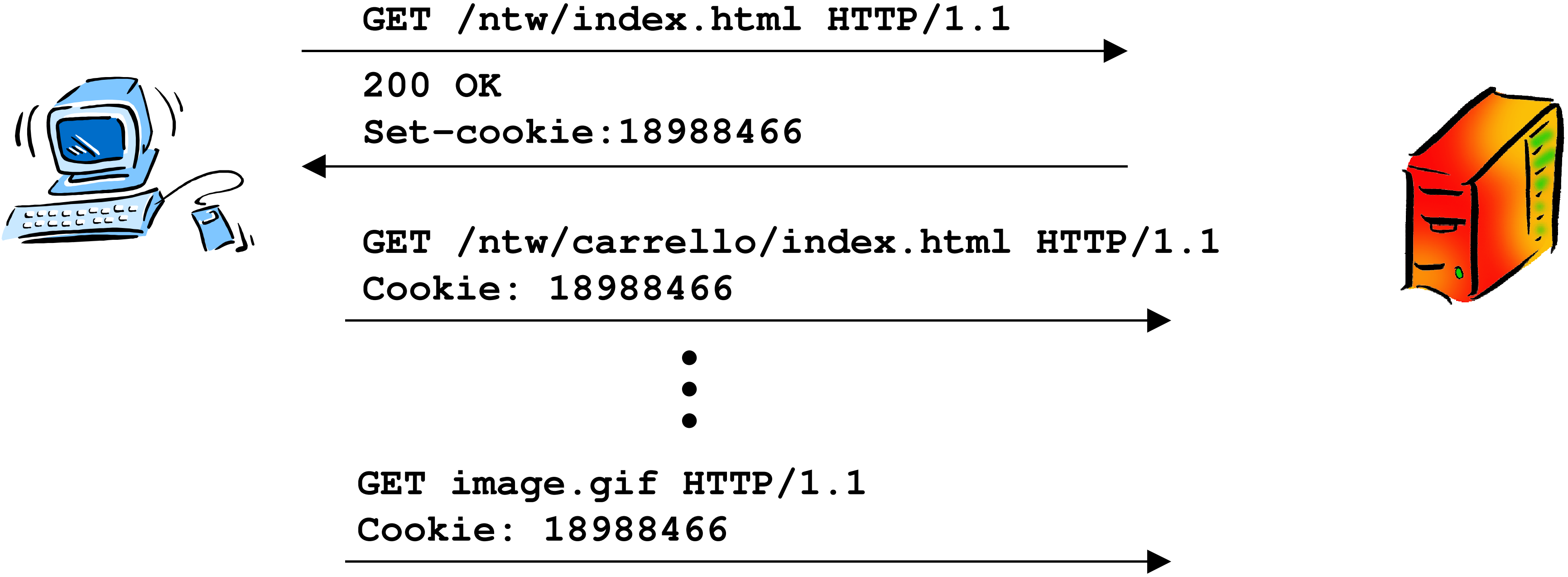
- HTTP è stateless e quindi non si possono riconoscere richieste successive dello stesso utente
- in HTTP esiste un elementare meccanismo di autenticazione (account e password) che serve a riconoscere gli utenti
- Normalmente il browser memorizza passwd e account in modo da non richiedere la digitazione ogni volta



```
GET /ntw/index.html HTTP/1.1

401 Authorization Required
WWW-Authenticate:[tipo di autenticazione]

GET /ntw/index.html HTTP/1.1
Authorization: account, passwd

•
•
•

GET image.gif HTTP/1.1
Authorization: account, passwd
```

Cookie

- Esiste anche un altro modo per riconoscere richieste successive di uno stesso utente che non richiede di ricordare password
- Il numero di cookie inviato dal server viene memorizzato in un opportuno file
- Tramite i cookie si può mantenere uno stato "virtuale" per ciascun utente.



```
GET /ntw/index.html HTTP/1.1

200 OK
Set-cookie:18988466

GET /ntw/carrello/index.html HTTP/1.1
Cookie: 18988466

GET image.gif HTTP/1.1
Cookie: 18988466
```

Cookie

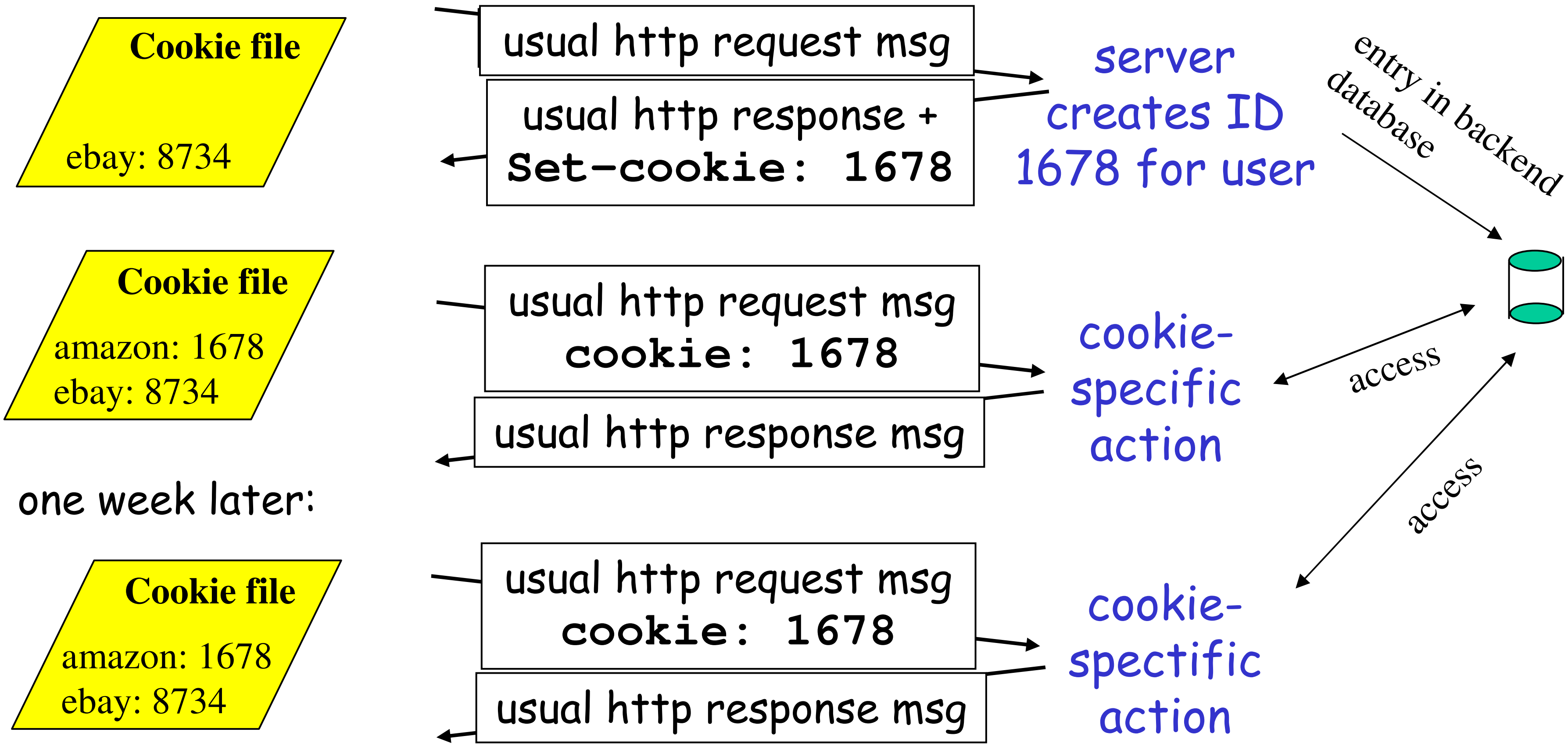
- Permette di identificare gli utenti e mantenere delle info di stato su una transazione che richiede vari scambi di messaggi HTTP
  - Personalizzazione
    - un sito di e-commerce può personalizzare la risposta con il nome dell'utente e suggerimenti di acquisto personalizzati
  - Mantenere informazioni tra sessioni o interazioni
    - esempio: shopping cart

Cookies: keeping "state" (cont.)

Cookie=
string of
characters

client

server



Cookie file

ebay: 8734

usual http request msg

usual http response +
`Set-cookie: 1678`

server creates ID 1678 for user

entry in backend database

Cookie file

amazon: 1678
ebay: 8734

usual http request msg
`cookie: 1678`

usual http response msg

cookie-specific action

access

access

one week later:

Cookie file

amazon: 1678
ebay: 8734

usual http request msg
`cookie: 1678`

usual http response msg

cookie-specific action

Identifier to find user info on the back end database

Cookie

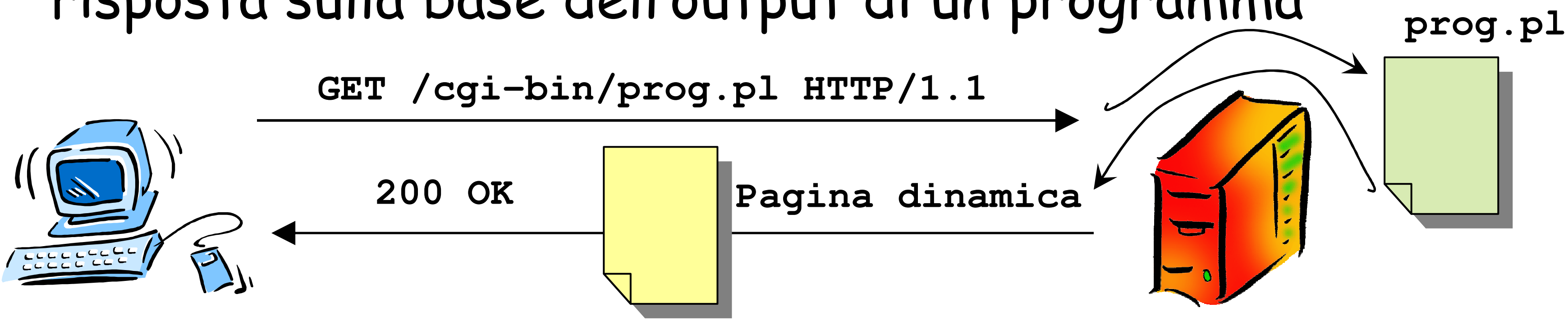
- User control on cookies
  - Whether to accept any cookies at all
  - Set a limit on the size and number of cookies
  - Limit the sites/domains from which a cookie can be accepted
  - Limit to a specific session acceptance of cookies
  - Require that cookies must originate from the same server as the current page being viewed
- Privacy concerns
  - Allow to track user behavior
  - Users typically not aware of when a cookie is sent
    - and on the use of the information he/she is providing through use of cookies
      - user profiling, shared between companies
  - Could also be sent from a Web Server different from the one user is aware to being connected to
    - e.g. due to redirection to download some objects referred to in the Main Server web page

HTML
(HyperText Markup Language)

- HTTP trasferisce file e non si occupa della loro semantica
- Il funzionamento del WWW si basa sull'interpretazione di file e sulla loro visualizzazione
- Pagine di testo formattate sono trasferite come file ASCII mediante dei comandi di formattazione specificate nel linguaggio HTML
- Le pagine HTML possono contenere riferimenti ad altri oggetti che dal browser possono essere interpretate
  - come parte del documento da visualizzare (immagini)
  - Come link ad altre pagine web
- Se una pagina HTML è memorizzata nel server e viene inviata su richieste è una pagina statica

Pagine WEB dinamiche

- Se una pagina viene creata al momento della richiesta (e normalmente in base alle informazioni fornite del client) si parla di pagine dinamiche
- Se una richiesta si riferisce ad una pagina dinamica il server esamina la richiesta, esegue un programma associato a quella richiesta e genera la pagina di risposta sulla base dell'output di un programma

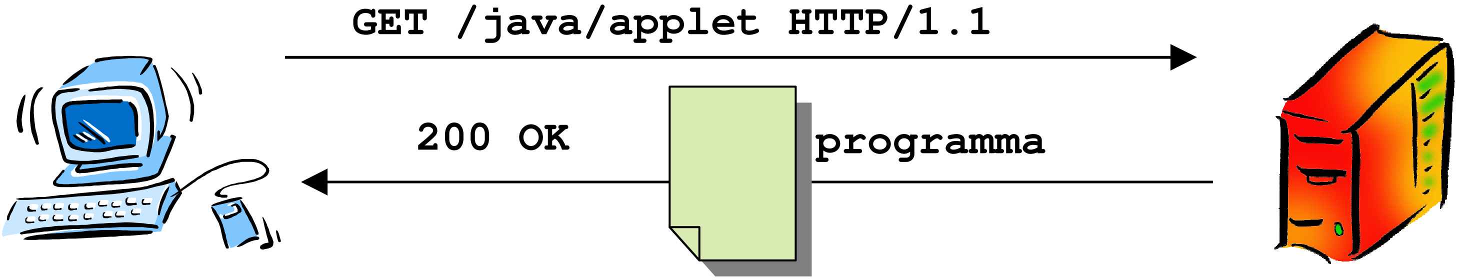


How dynamically generated responses are created

- Server side includes
  - e.g., personalization
    - HTML file can include directives or macros that instruct the server to insert some info (e.g. current time, last modification of the file) at the time of request
- Server scripts
  - URL calls a program
    - to customize response
    - can access info on a data base
    - (SW and raw data used remain private)

Pagine WEB attive

- ❐ Una pagina web può anche contenere un programma che deve essere eseguito dal client
- ❐ Il programma viene scaricato come un oggetto della pagina ed eseguito in locale sulla macchina del client
- ❐ Può essere utile per ottenere delle pagine in grado di interagire con l'utente, per grafici in movimento, ecc.



Searching on the Web

- Where can I find something on XXXX ???
  - Hundreds of thousands of users are searching for strings in tens of millions of documents distributed over millions of machines→tough problem search engines have to solve
- Inverted Index (like at the end of a book) can speed up search: points back to the pages where the indexed term appears
  - Terms NOT included in the index = stops words

How to create central inverted index????

Spiders

- ❒ Programs used to obtain info on some or all the resources stored on a large number of Web sites. Purpose: generating an inverted index
- ❒ Starts from a list of popular sites (start-list) and follows all the URLs within the sites (breadth-wise or depth-wise traversal, up to a given depth in a cycle)—taking care of avoiding cycles

❒Web site administrators can signal thesite (or a part of it) should not be indexed in file robot.txt (

User-Agent: Aracnophobia, BlackWindow LIST OF AGENTS WHICH CANNOT INDEX
Disallow: /stats
Disallow: /cgi-bin/ DIRECTORIES NOT TO BE INDEXED

Robot Exclusion Standard
(for well behaved spiders)

Similar info could be contained in the HTML tag

<META NAME="ROBOTS" CONTENT="NOINDEX, NOFOLLOW">

❒ Search Engine	Agent name
Google	GoogleBot
Yahoo	Yahoo Slurp

Spiders can fetch over a billion web pages

Search Engines

- Perform the search, locates the documents where the string appeared and provide a result set, a subset of the documents filtered to trade-offs between metrics
  - Metrics
    - Recall(how large is the result set as a function of the total list of documents where the search terms have appeared?)
    - Precision (how relevant are the retrieved documents?)
- The results have to be ranked
  - E.g. based on an estimate of the fit to requested keywords and how often they are referred to by authoritative Web pages (Google)

Chapter 2 outline

- 2.1 Principles of app layer protocols
  - clients and servers
  - app requirements
- 2.2 Web and HTTP
- 2.3 FTP
- 2.4 Electronic Mail
  - SMTP, POP3, IMAP
- 2.5 DNS
- 2.9 Content distribution
  - Network Web caching
  - Content distribution networks
  - P2P file sharing

DNS: Domain Name System

- "Domain Names – Concepts and Facilities," RFC 1034, Nov. 1987.
- "Domain Names – Implementation and Specification," RFC 1035, Nov. 1987.

People: many identifiers:

- SSN*, name, passport #

Internet hosts, routers:

- IP address (32 bit) - used for addressing datagrams
- "name", e.g., gaia.cs.umass.edu - used by humans (symbolic names- simple, decouple service and device)

- IP addresses shorter (less overhead / computation to manipulate)
- Using IP addresses at application layer would be inconvenient
  - Symbolic names easy to remember
  - If the IP address of a site changes (e.g., as Web site moves to a new hosting service) URLs would have to change also

Q: map between IP addresses and name ?

Domain Name System:

- ❐ *distributed database* implemented in hierarchy of many *name servers*
- ❐ *application-layer protocol* host, routers, name servers to communicate to *resolve* names (address/name translation)

Application: IP addr?
(client) → name server
←

- note: core Internet function, implemented as application-layer protocol
- complexity at network's "edge"

*in our case we don't have the SSN but: codice fiscale

DNS: Domain Name System

- "Domain Names - Concepts and Facilities," RFC 1034, Nov. 1987.
- "Domain Names - Implementation and Specification," RFC 1035, Nov. 1987.

People: many identifiers:

- SSN*, name, passport #

Internet hosts, routers:

- IP address (32 bit) - used for addressing datagrams
- "name", e.g., gaia.cs.umass.edu - used by humans (symbolic names- simple, decouple service and device)

- IP addresses shorter (less overhead / computation to manipulate)
- Using IP addresses at application layer would be inconvenient
  - Symbolic names easy to remember
  - If the IP address of a site changes (e.g., as Web site moves to a new hosting service) URLs would have to change also

Domain Name System:

- *distributed database* implemented in hierarchy of many *name servers*
- *application-layer protocol* host, routers, name servers to communicate to *resolve* names (address/name translation)

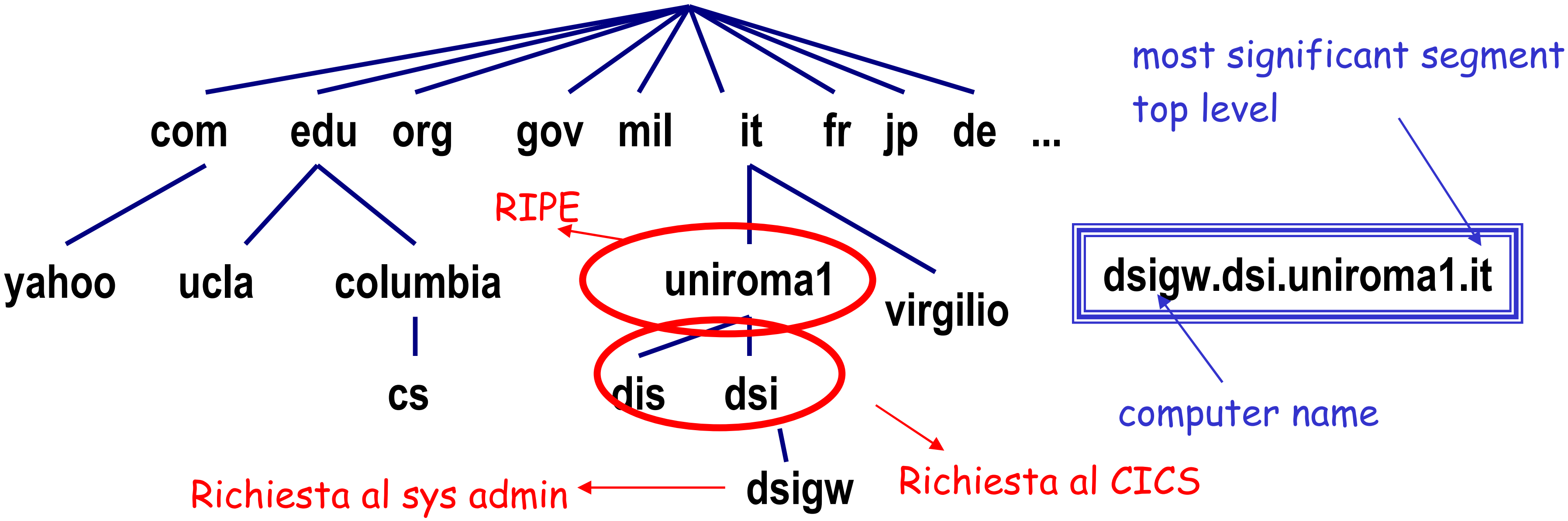
Application: IP addr?
(client) → name server
←

- note: core Internet function, implemented as application-layer protocol
- complexity at network's "edge"

Applications access DNS through a resolver (SW library linked to the application)

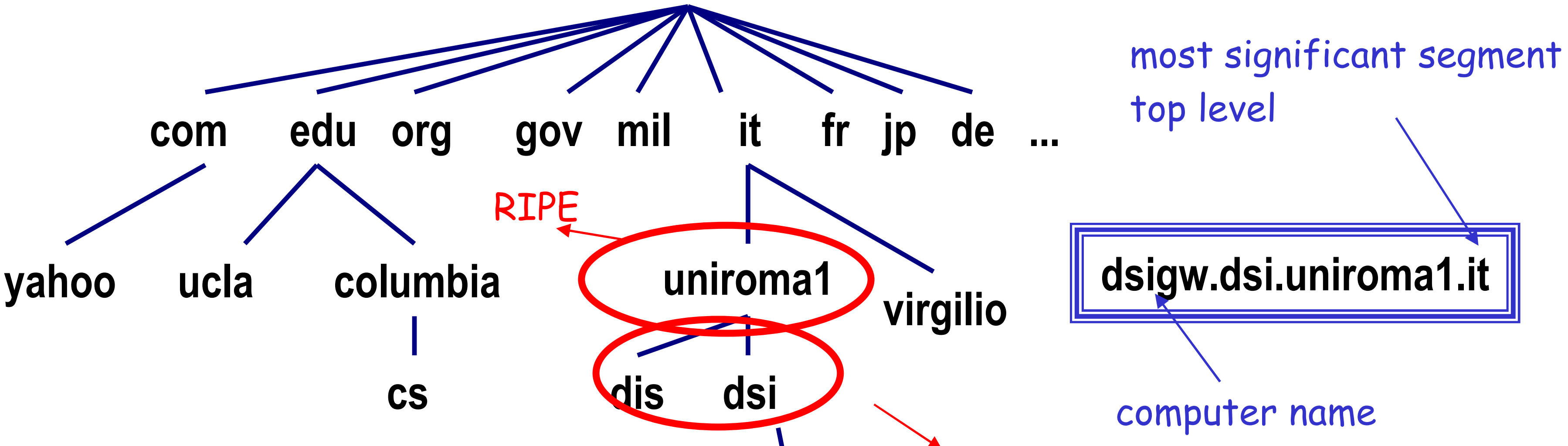
To bootstrap the process the resolver must know at least one DSN server (local DNS server)

Indirizzamento simbolico



- L'indirizzamento è di tipo gerarchico
- Ogni ramo è sotto il controllo di un'autorità
- Per ottenere un nuovo indirizzo occorre chiedere il permesso all'autorità competente
- Una volta che una organizzazione ha avuto un nome sotto un dominio top level tale suffisso (e.g. uniroma1.it) e' a lei riservato; gestisce in maniera autonoma come strutturare i nomi della sua gerarchia (dsi.uniroma1.it e dis.uniroma1.it ma anche inf.dis.uniroma1.it e sys.dis.uniroma1.it –i nomi associati ai vari dipartimenti potrebbero venir ulteriormente strutturati solo in alcuni casi)

Indirizzamento simbolico



ICANN (Internet Corporation for Assigned Names and Numbers) allocates portions of the IP address space to regional Internet registries

-which in turn allocate IP addresses to organizations in their region

- Allocating a fixed set od addresses to each region helps promoting fair and efficient allocation of remaining IP addresses AND favors summarization.

tà

competente

RIPE=

Reseaux IP Europeens Network Coordination Center

RIPE NCC is one of five Regional Internet Registries (RIRs) providing Internet resource allocations, registration services and coordination activities that support the operation of the Internet globally.

rto un dominio top
estisce in maniera
dsi.uniroma1.it e
iroma1.it –i nomi
mente strutturati

DNS name servers

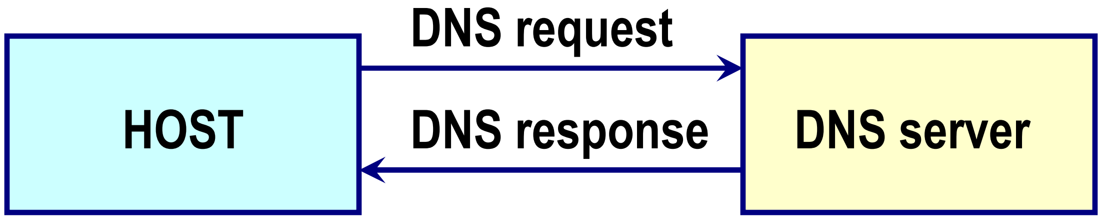
Why not centralize DNS?

- single point of failure
- traffic volume
- distant centralized database
- maintenance

doesn't *scale!*

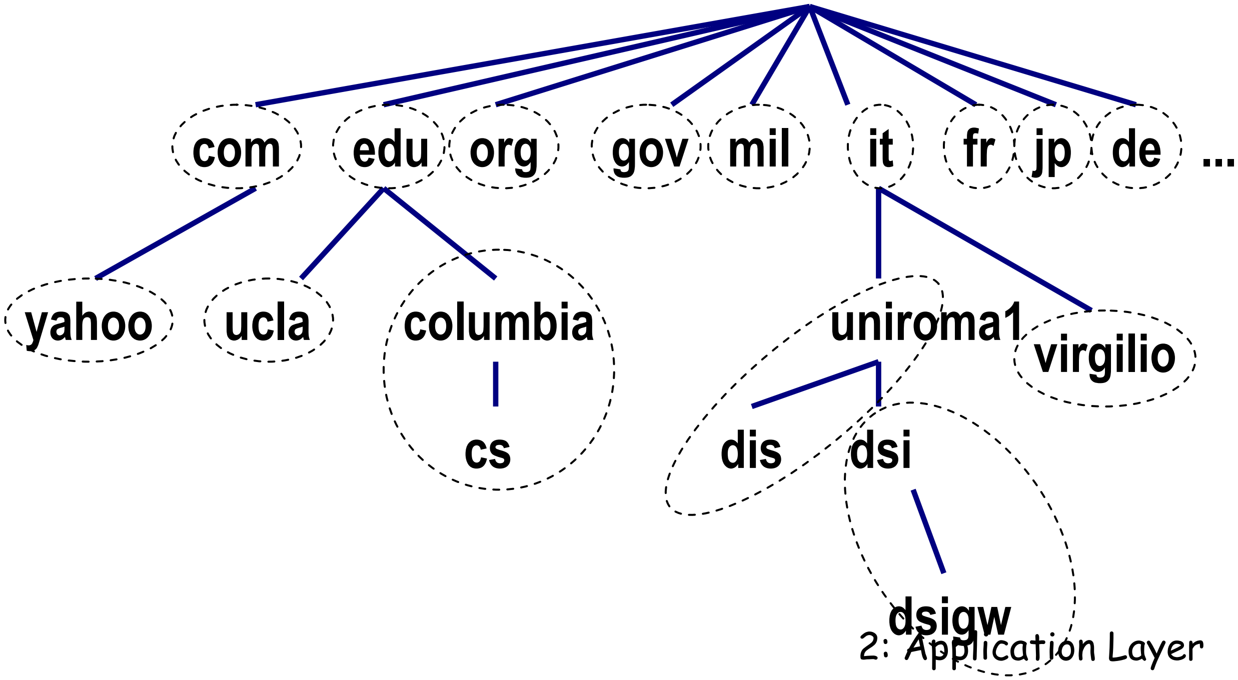
Come ottenere un mappaggio

- Ogni host ha configurato l'indirizzo del server DNS a cui rivolgersi (manualmente, vedi ad es. Pannello di controllo di WIN)
- Le applicazioni che richiedono un mappaggio (browser, ftp, etc.) usano le funzioni del DNS
- Una richiesta viene inviata al server DNS usando UDP come trasporto
- Il server reperisce l'informazione e restituisce la risposta



Organizzazione del database

- I record in ARPANET erano contenuti in un name server centrale
- Per Internet la struttura del database è distribuita
- I rami sono partizionati in zone e un DNS server viene associato ad ogni zona
- Il server di una zona è responsabile per le informazioni di quella zona (authoritative)
- I root server sono autorevoli per i top-level domains (il che non significa che contengano tutte le info ma che sappiano come/da chi recuperarle)



DNS name servers

Why not centralize DNS?

- single point of failure
- traffic volume
- distant centralized database
- maintenance

Motiva anche il fatto di avere server DNS multipli all'interno di una organizzazione

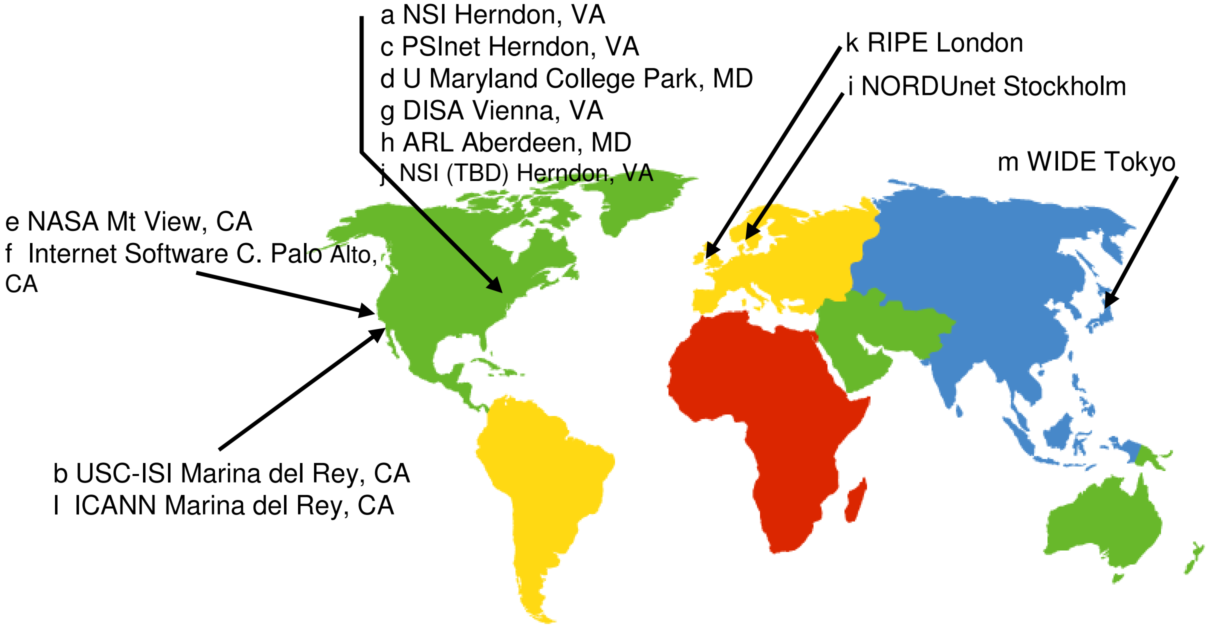
- no server has all name-to-IP address mappings

local name servers:

- each ISP, company has *local (default) name server*
- host DNS query first goes to local name server
  - unless info has been cached the local DNS server will contact a root server to know the DNS server to contact for that top level domain (e.g., .com OR .edu)

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 known
  - gets mapping
  - returns mapping to local name server



13 root name servers worldwide

DNS name servers

Why not centralize DNS?

- single point of failure
- traffic volume
- distant centralized database
- maintenance

Motiva anche il fatto di avere server DNS multipli all'interno di una organizzazione

- no server has all name-to-IP address mappings

local name servers:

- each ISP, company has *local (default) name server*
- host DNS query first goes to local name server

authoritative name server:

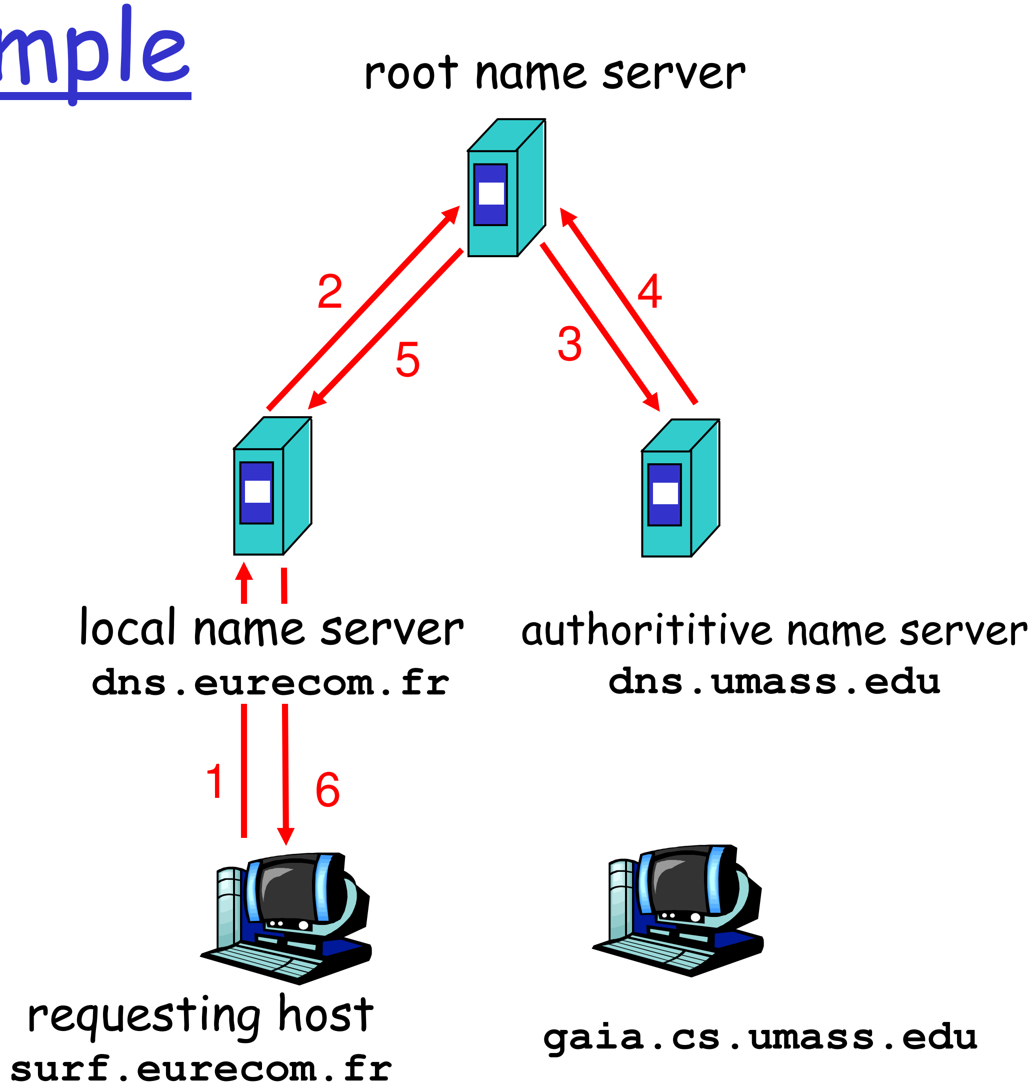
- for a host: stores that host's IP address, name
- can perform name/address translation for that host's name

Locality of reference: user tend to look up the names of local computers often; in any case tends to look up the same set of domain sets repeatedly → caching (LATER!!)

Simple DNS example

host **surf.eurecom.fr** wants IP address of **gaia.cs.umass.edu**

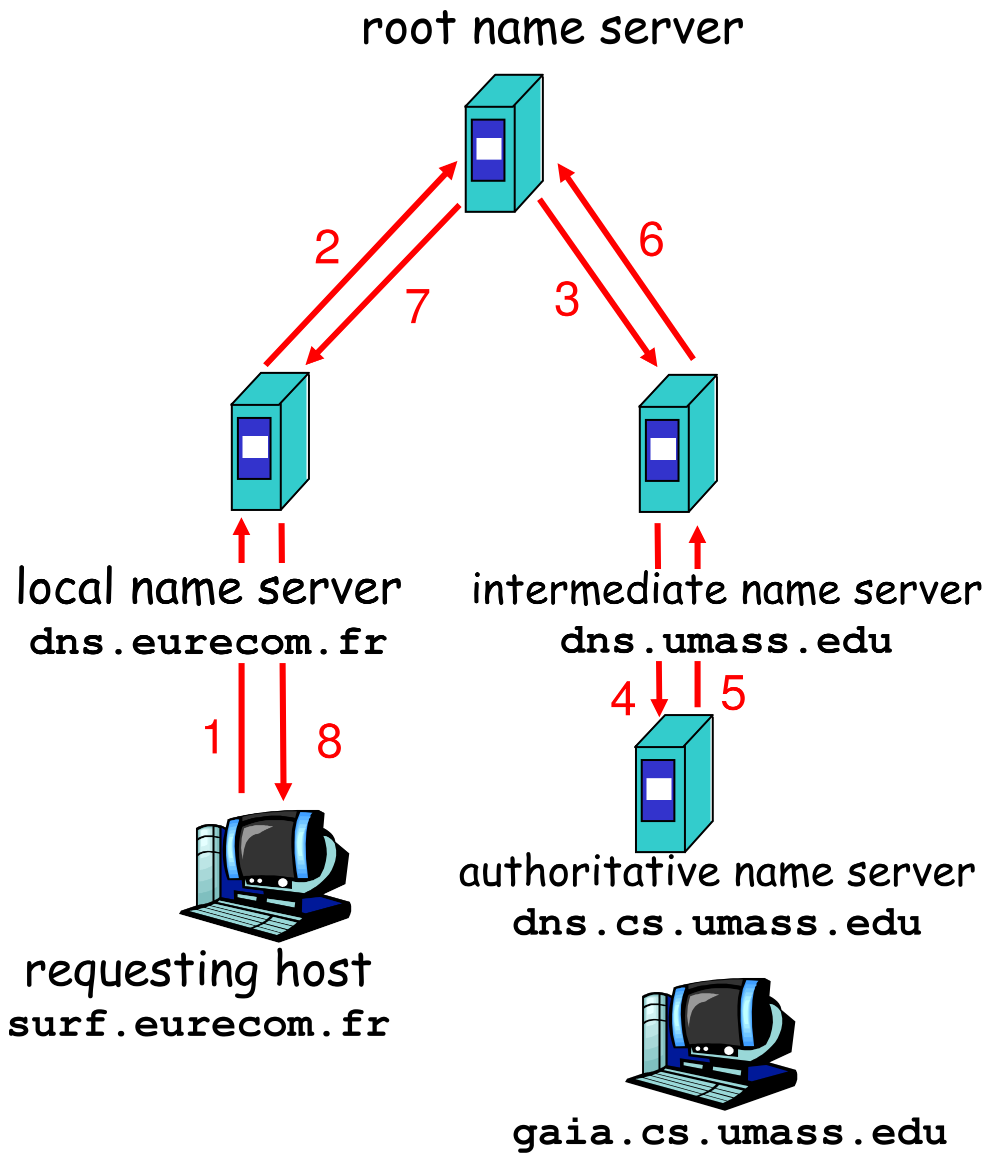
1. contacts its local DNS server, **dns.eurecom.fr**
2. **dns.eurecom.fr** contacts root name server, if necessary
3. root name server contacts authoritative name server, **dns.umass.edu**, if necessary



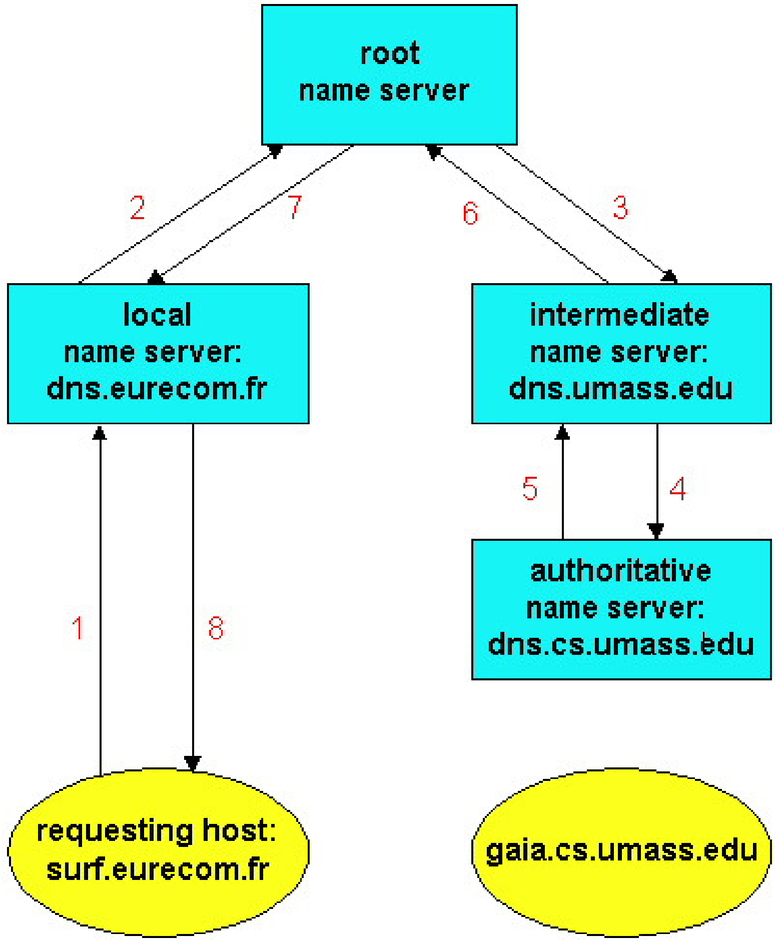
DNS example

Root name server:

- may not know authoritative name server
- may know *intermediate name server:* who to contact to find authoritative name server

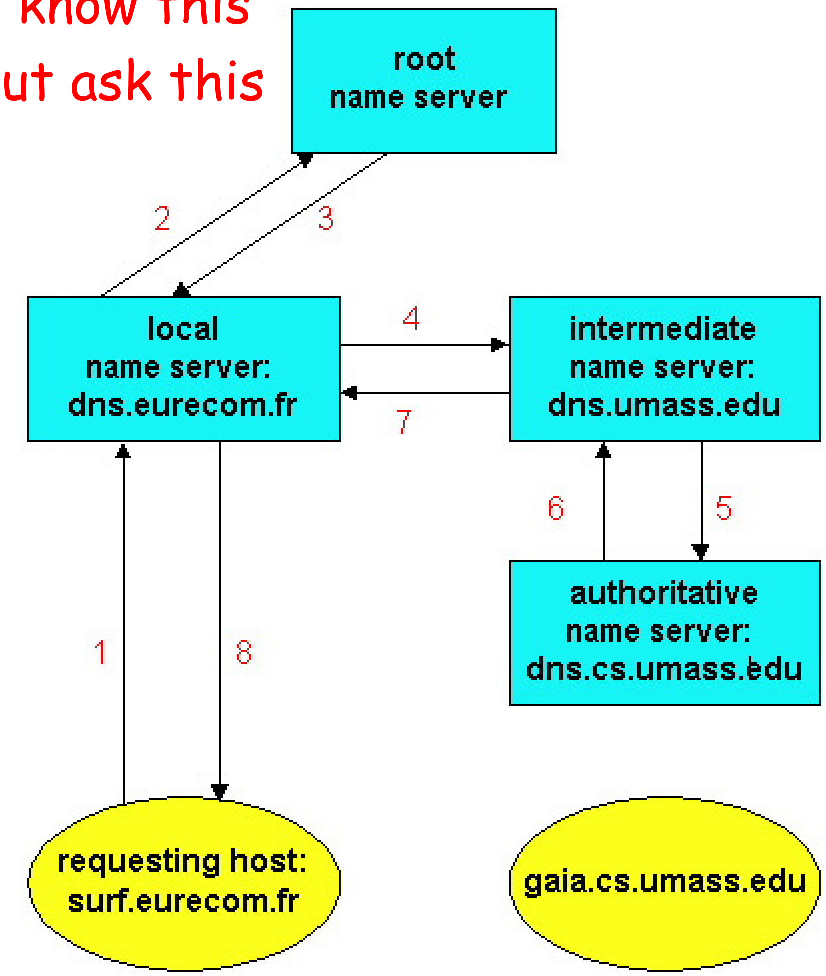


Reperire informazioni: modo ricorsivo



- in modalità puramente recursive
- la richiesta viene inoltrata seguendo la gerarchia
- la risposta segue la strada inversa

Reperire informazioni: modo iterativo



"I don't know this name, but ask this server"

- con la modalità iterativa
- un server può rispondere ad una richiesta con il nome di un altro server dove reperire l'informazione

Root servers handle only iterative queries

DNS optimizations

- Replication:
  - E.g si usa il root server piu' vicino
- Caching:
  - Un local DNS server, dopo aver reperito un'informazione su cui NON è authoritative può memorizzarla temporaneamente (caching)
  - All'arrivo di una nuova richiesta può fornire l'informazione senza risalire sino al server authoritative
  - Cache entries hanno associato un TTL (scompaiono dopo un po' di tempo)
  - Il TTL è settato dal server authoritative ed è un indice di quanto stabile nel tempo sia l'informazione relativa
  - Caching anche dell' IP address del server autoritativo E di risposte negative (failed queries)

DNS records

Domain name

Record type

DNS: distributed db storing resource records (RR)

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

- Type=A

domain name→address

  - **name** is hostname
  - **value** is IP address

(morgana.elet.polimi.it, 131.175.21.1, A, TTL)

- 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 cannonical name

- Type=MX (Mail eXchanger)
  - **value** is name of mailserver associated with **name**

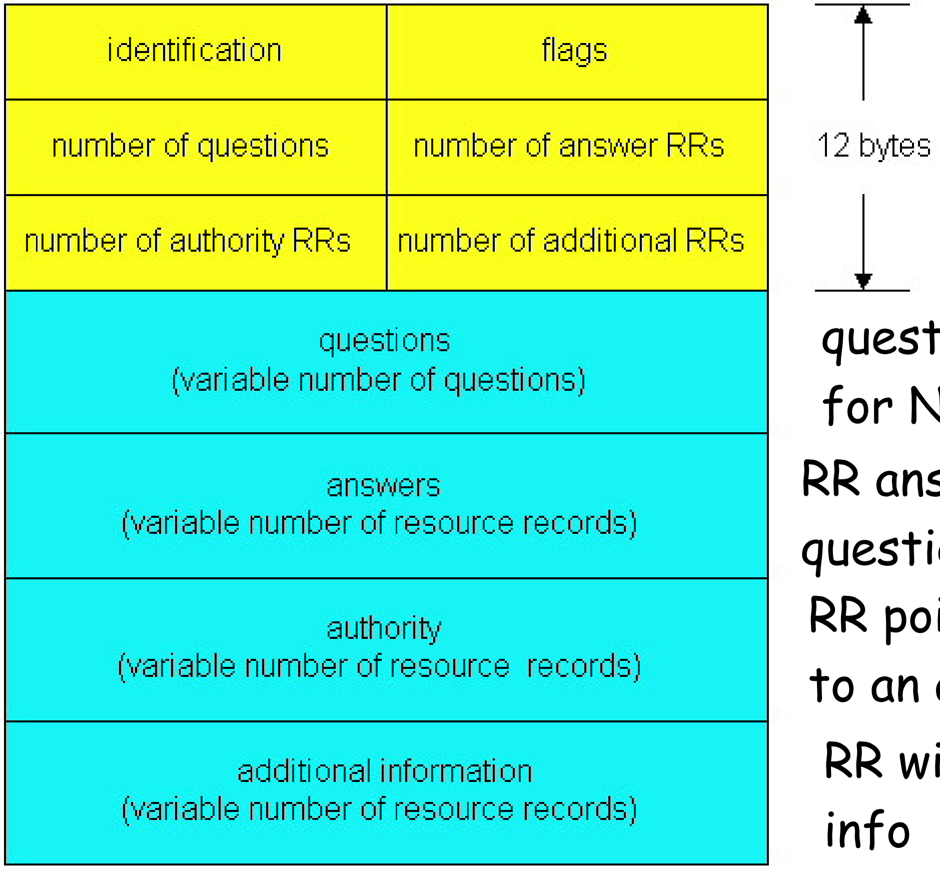
(elet.polimi.it, mailserver.elet.polimi.it, MX,TTL)

DNS protocol, messages

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



identification	flags
number of questions	number of answer RRs
number of authority RRs	number of additional RRs
questions (variable number of questions)	
answers (variable number of resource records)	
authority (variable number of resource records)	
additional information (variable number of resource records)	

question for NS

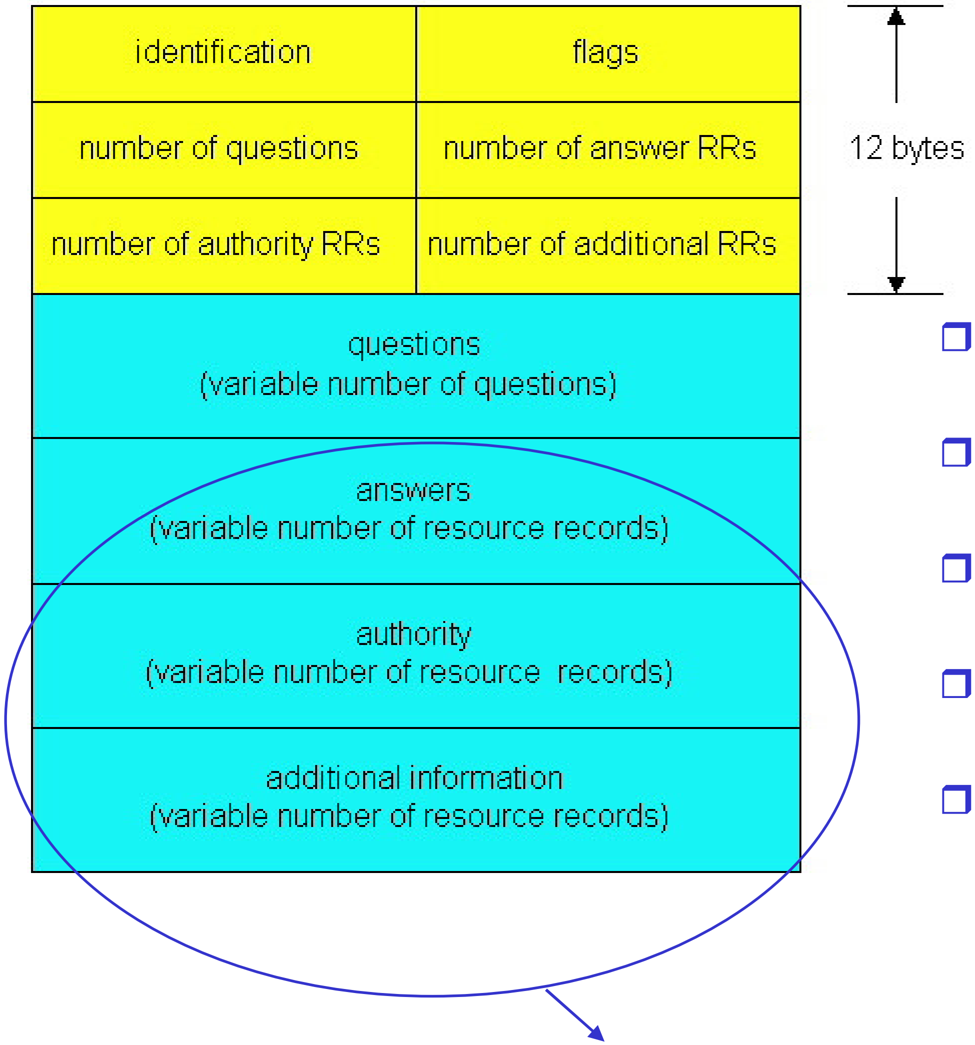
RR answering question

RR pointing to an authority

RR with add. info

Messaggi DNS

sono in binario (non ASCII)



identification	flags
number of questions	number of answer RRs
number of authority RRs	number of additional RRs

questions
(variable number of questions)

answers
(variable number of resource records)

authority
(variable number of resource records)

additional information
(variable number of resource records)

Resource Records

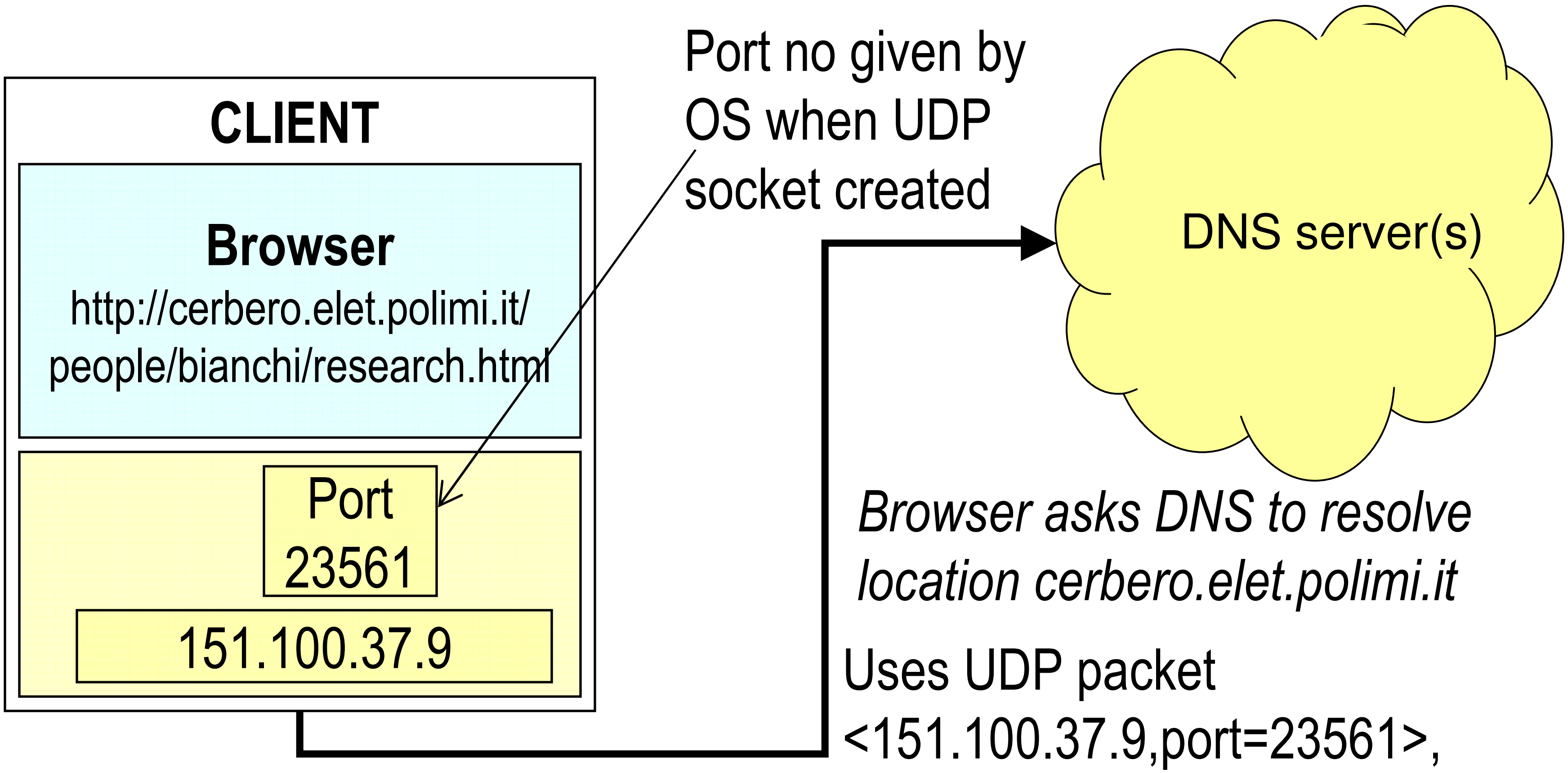
- identification: identificativo coppia richiesta/risposta
- flags: richiesta/risposta, authoritative/non auth., iterative/recursive (richiesta dal client, supporto comunicato dal server),flag che indicano il verificarsi di errori(e.g. dall'authoritative server 'il dominio non esite')
- number of: relativo al numero di campi nelle sez. successive
- questions: nome richiesto e tipo (di solito A o MX)
- answers: resourse records completi forniti in risposta
- authority: contiene altri record forniti da altri server
- additional infor.: informazione addizionale, ad es. il record con l'IP ADDR. per il MX fornito in answers

Perche' UDP?

- Less overhead
  - Messaggi corti
  - Tempo per set-up connessione di TCP lungo
  - Un unico messaggio deve essere scambiato tra una coppia di server (nella risoluzione contattati diversi server—se si usasse TCP ogni volta dovremmo mettere su la connessione!!)
- Se un messaggio non ha risposta entro un timeout?
  - Semplicemente viene riinviato dal resolver (problema Risolto dallo strato applicativo)

Porta usata per il DNS: 53!!

Un esempio: uso di DNS da parte di un client web



CLIENT

Browser

http://cerbero.elet.polimi.it/people/bianchi/research.html

Port 23561

151.100.37.9

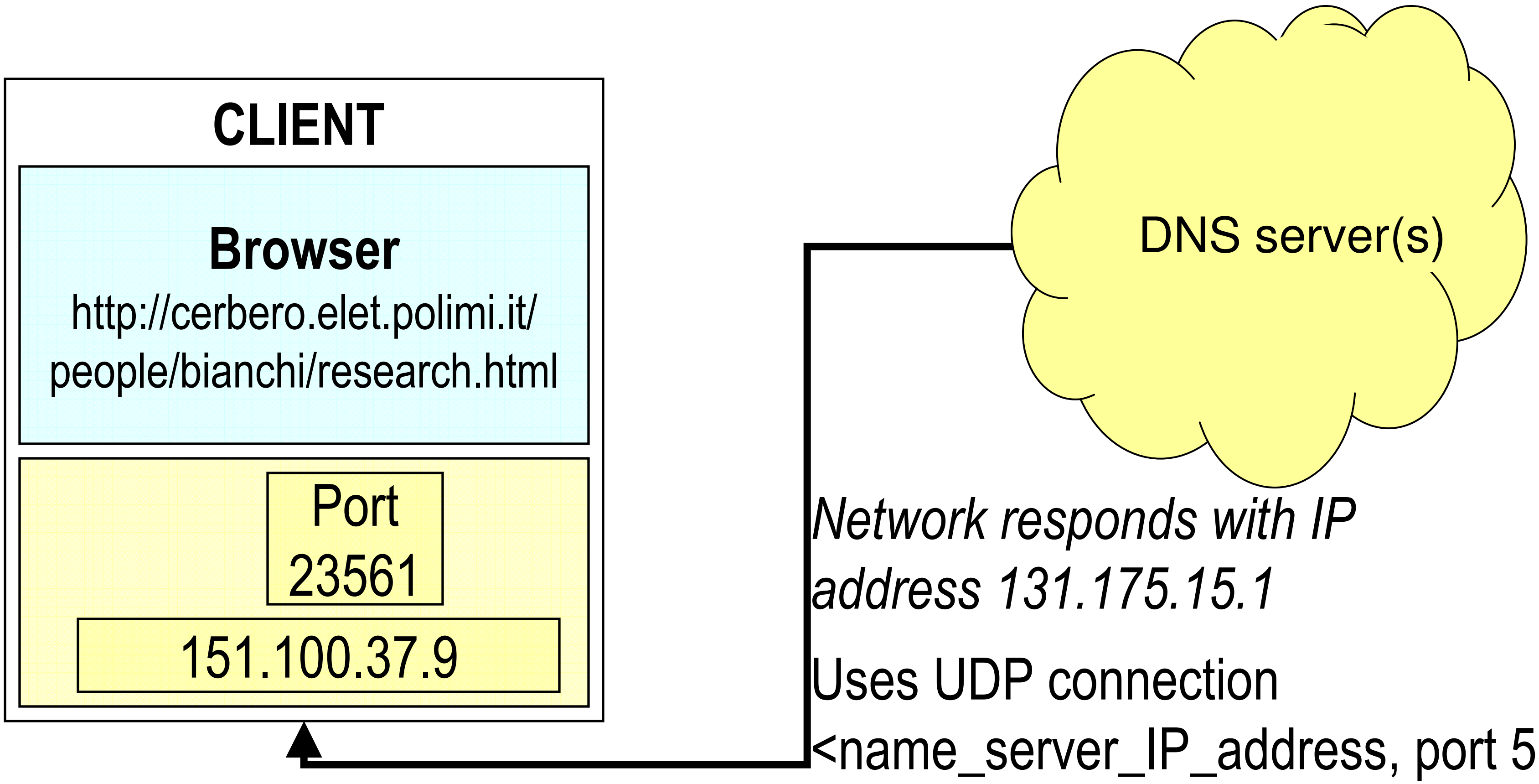
Port no given by OS when UDP socket created

DNS server(s)

Browser asks DNS to resolve location cerbero.elet.polimi.it

Uses UDP packet
<151.100.37.9,port=23561>,
<name_server_IP_address, port 53>

opening transport session: client side, step b



CLIENT

Browser

http://cerbero.elet.polimi.it/people/bianchi/research.html

Port 23561

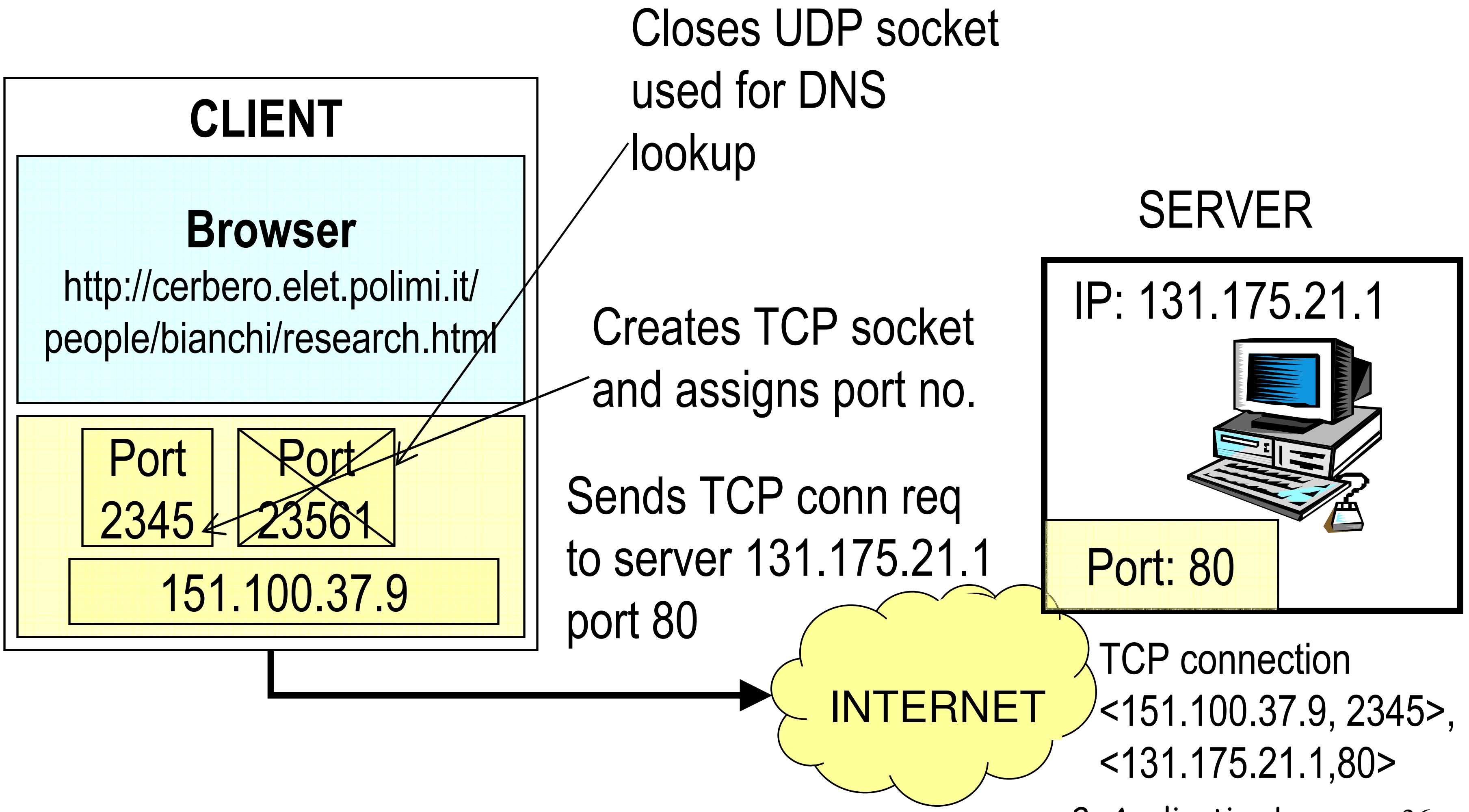
151.100.37.9

DNS server(s)

Network responds with IP address 131.175.15.1

Uses UDP connection <name_server_IP_address, port 53>, <151.100.37.9,port=23561>

opening transport session: client side, step c



CLIENT

Browser

http://cerbero.elet.polimi.it/people/bianchi/research.html

Port 2345

~~Port 23561~~

151.100.37.9

Closes UDP socket used for DNS lookup

Creates TCP socket and assigns port no.

Sends TCP conn req to server 131.175.21.1 port 80

SERVER

IP: 131.175.21.1

Port: 80

INTERNET

TCP connection <151.100.37.9, 2345>, <131.175.21.1,80>

opening transport session: server side

- httpd (http daemon) process listens for arrival of connection requests from port 80.
- Upon connection request arrival, server decides whether to accept it, and send back a TCP connection accept
- This opens a TCP connection, uniquely identified by client address+port and server address+port 80

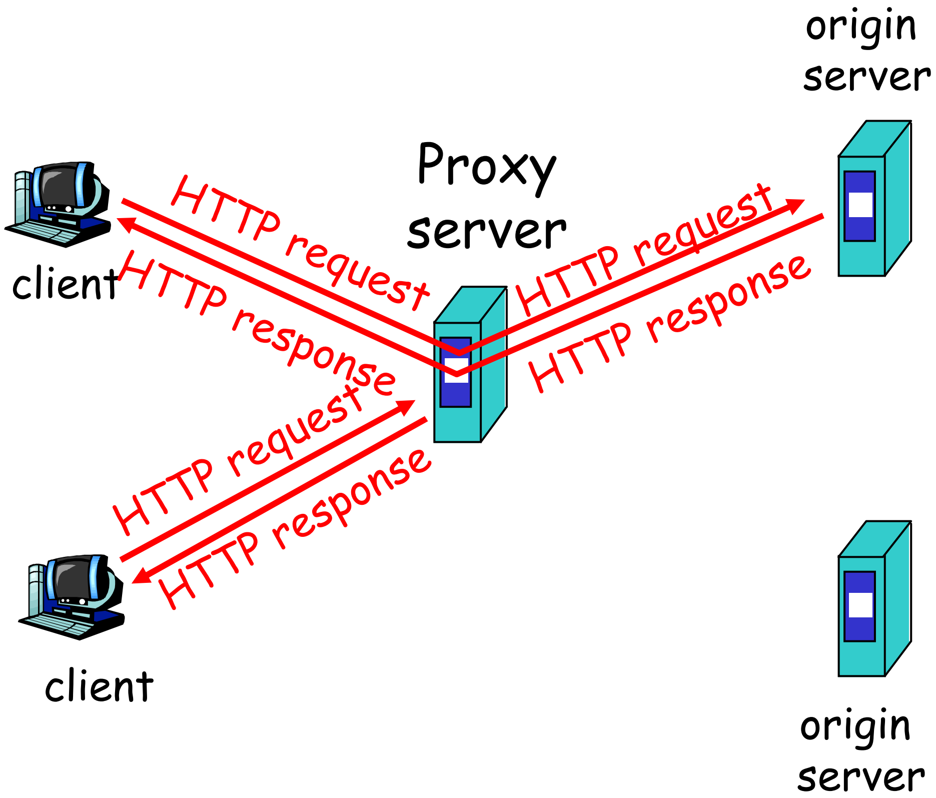
Chapter 2 outline

- 2.1 Principles of app layer protocols
  - clients and servers
  - app requirements
- 2.2 Web and HTTP
- 2.3 FTP
- 2.4 Electronic Mail
  - SMTP, POP3, IMAP
- 2.5 DNS
- 2.9 Content distribution
  - Network Web caching
  - Content distribution networks
  - P2P file sharing

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
- Cache can do up-to-date check using `If-modified-since` HTTP header
  - Issue: should cache take risk and deliver cached object without checking?
  - Heuristics are used.
- 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 (browser).
- Reduce network load
  - Reduce overall congestion improving user perceived performance
- Reduce the load on the origin server.
- Internet dense with caches enables "poor" content providers to effectively deliver content

Caching example (1)

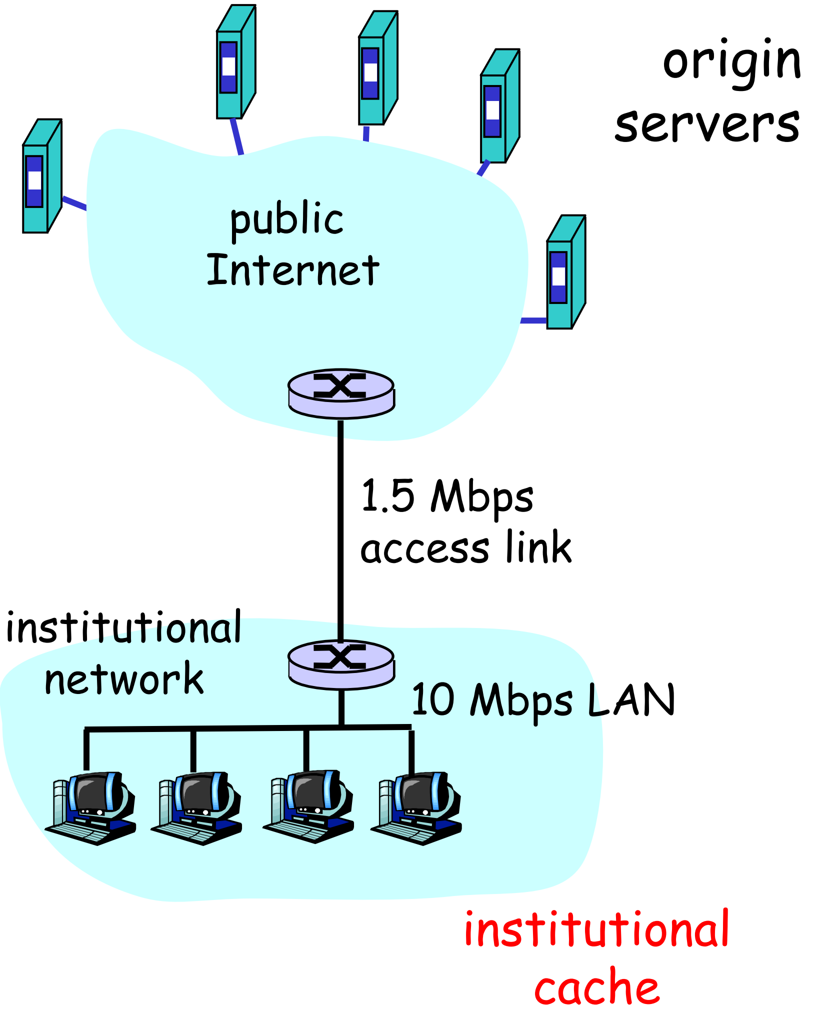
Assumptions

- ❒ average object size = 100,000 bits
- ❒ avg. request rate from institution's browser to origin serves = 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 (2)

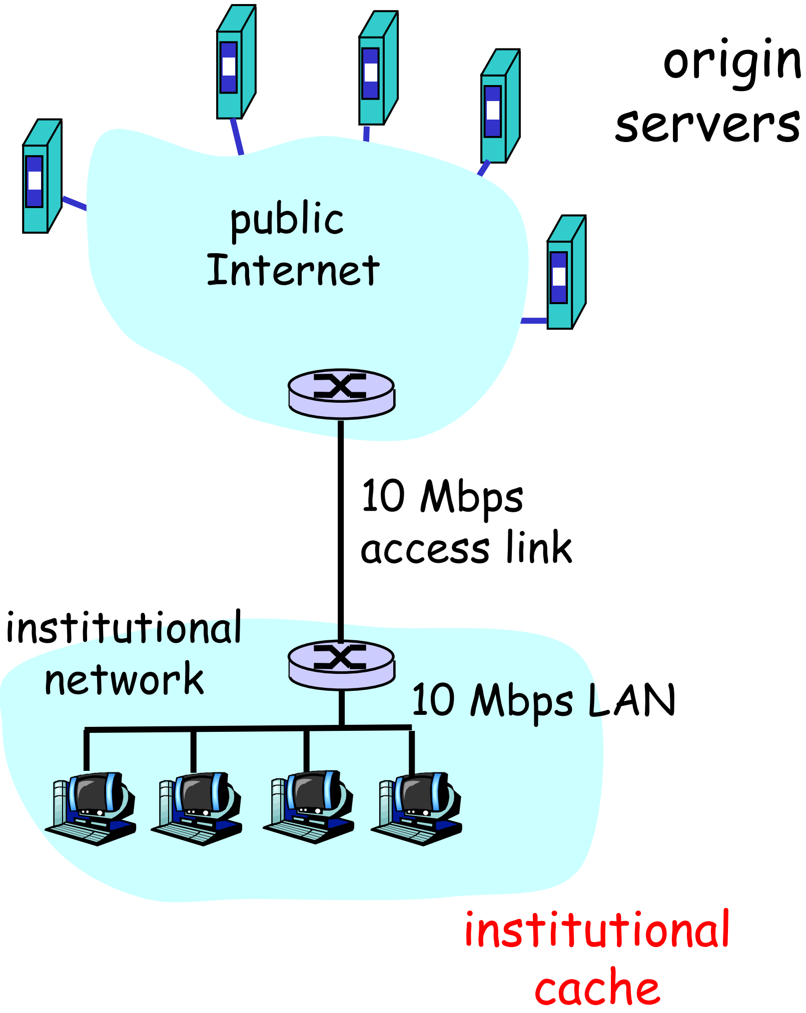
Possible solution

- 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 (3)

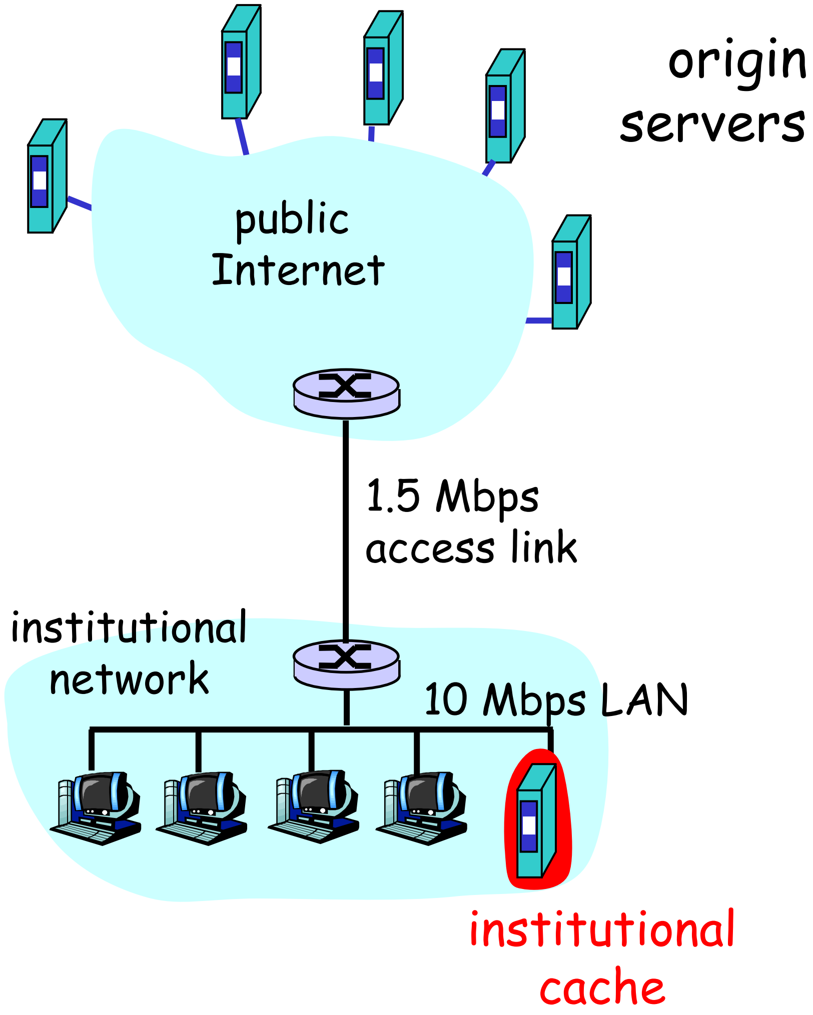
Install cache

- 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 delay = Internet delay + access delay + LAN delay

= .6*2 sec + .6*.01 secs + milliseconds < 1.3 secs



Delays rationale in content access & caching

Delays

Network connectivity

- If link speed low this is an ultimate bottleneck (browser caching)

DNS-related delay

- High: DNS caching helps, especially if TTL high

Network congestion

- Caching→ less packets transmitted in the network has a beneficial effect

Origin server load

- Caching→less requests to handle at the origin server, load distributed among several caches

Time to generate responses

Browser rendering of response

Cache locations

At the Browser

→the fastest!!

At a proxy

→takes advantage of frequently requested resources by other users in the same environment

Cache hit← resource in the cache

Cache miss←not in the cache

Issues related with caching

- Is the content cacheable ? (e.g. dynamic data may not be cacheable; the server may indicate a content as not cacheable)
- Storing in the cache: what if there is no space in the cache (cache replacement-next slide)
- Storing in the cache: for how long? Expiration time inlcluded in the HTTP reply. Otherwise heuristic adopted to compute a reasonable expiration time (should the cached resource be delivered to the client?→problem: cache coherency—is the cache content stale or 'fresh'?).
- Cache maintainance:
  - Eviction of stale objects
  - Popular resources could be proactively prevalidated (e.g. via HTTP HEAD) and prefetched if changes have occurred.

Cache replacement

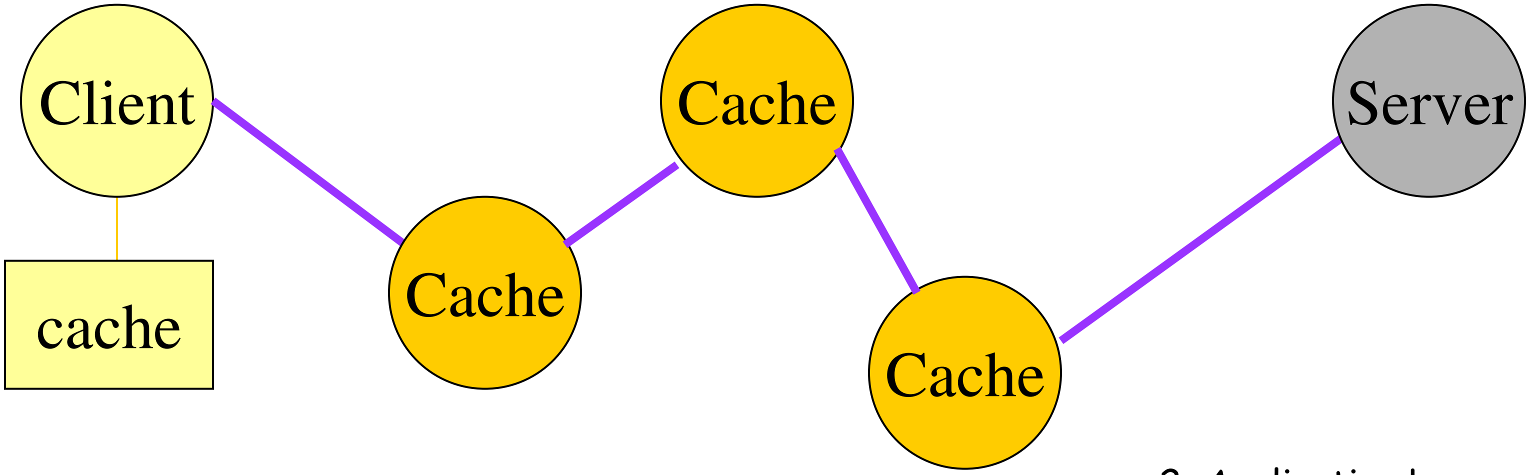
- Different metrics: cost of fetching again the resource, cost of storing the resource (size), number of access to the resource in the past, probability of accessing the resource in the near future, time since the last modification of the resource, expiration time
- Algorithms used to decide which cache item to replace
  - Least Recently Used (LRU)—delete the LRU object
  - Least Frequently Used (LFU)—delete the LFU object
  - Size of object (SIZE)—delete the largest object
  - Hyper-G algorithm: Remove the LFU. If more than one object with the same metric the LRU; if more than one object the largest
  - Greedy Dual Size: replace the resource with the lowest utility (a function of the cost of fecthing the object, size, age)
- Research efforts in the first years of caching. Less critical with 1)falling cost of storage 2)dynamic pages 3)development of acceptably performing algorithms (e.g. Hyper-G, Greedy Dual Size)

Cache coherency

- Strong coherency: a revalidation request is sent every time a cache hit occurs (es. messaggio di HEAD con If-Modified-Since Header in HTTP)
- Weak coherency: An heuristic is adopted *locally* at the cache to decide whether the cached response is still fresh without consulting the origin server
  - Es. responses from the server have a TTL associated with them. During the TTL period the cache does not revalidate the response saving bandwidth at the price of possible staleness.
- In HTTP v1.1 l'utente puo' effettuare una richiesta con Cache-Control: only-if-cached chiedendo che in ogni caso la risposta provenga dalla cache senza contattare l'origin server

Hierarchical Caching: How do cache in a hierarchy communicate

- Advantage: message transmission kept at a regional level, less traffic, low delay
- Delay added with each cache miss
- Light protocols designed for inter-cache communication



An example: Inter Cache Protocol

- ICP (1997) used in the freely available and widely used caching SW Squid
- Hierachy with caches, regional and national caches
- If the original cache does not have teh resource sends a ICP request (UDP message) to all its peers. If someone replies HTTP request directed to such cache. Otherwise after a timeout the parent of the cache will repeat the process. If none of the caches has the resource it will be requested from teh origin server.
- Optimization: when a request comes back teh intermediate caches on the path stores it for future use.
- Improvements: Cache Digest Protocol(98). A digest of a cache contents is exchanged via HTTP. Only the caches with the resource in their digest are contacted. Size of the digest? Staleness of the digest?

Cache busting

- Caching has some problems
  - Impossibility to count the access to servers (requested for advertisement)
  - possible staleness of content with weak consistency
- Servers have performed

  cache busting: any technique preventing a cacheable resource from being cached at prowsers or broxies

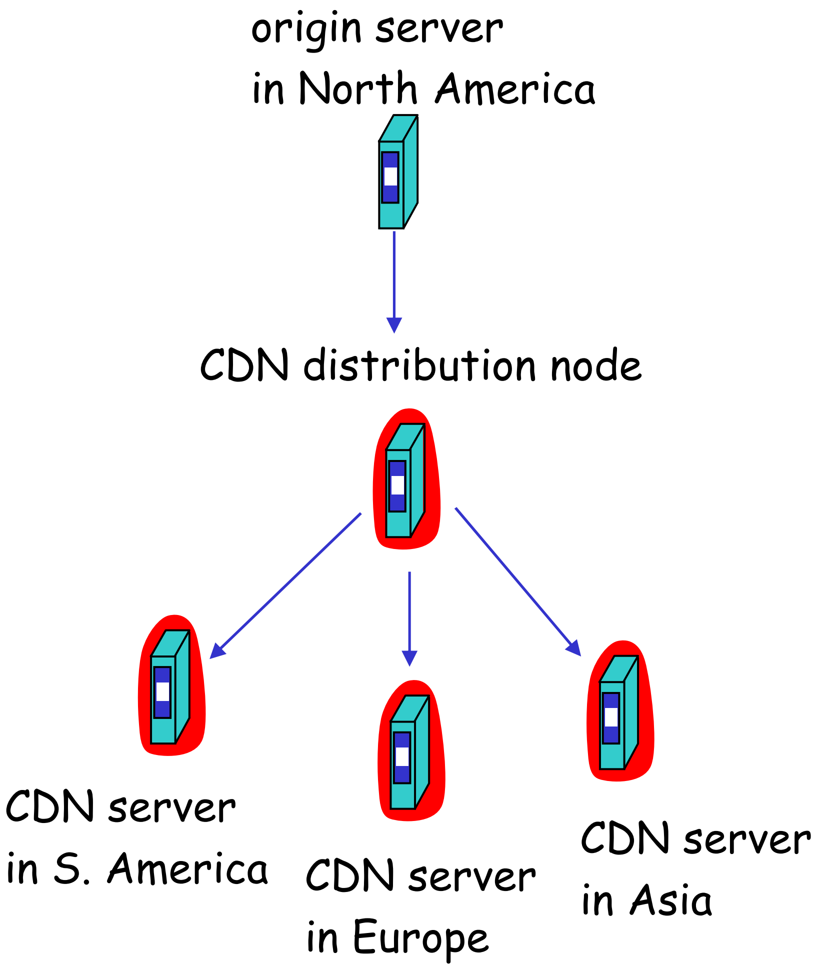
  Es. expiration time in the past; Cache-Control header set to no-cache or no-store
- How to discourage cache busting?
  - Hit-metering(97)—not really successful: introduces a new HTTP header, Meter, containing the number of cache hits on a given resource. Proxy can serve a resource directly a fine number of times before recontacting the origin server. During revalidation number of hits delivered to the origin server. Such number also communicated in case the item is removed from the cache.
  - proxy handle advertisement to the page on behalf of the server

Content distribution networks (CDNs)

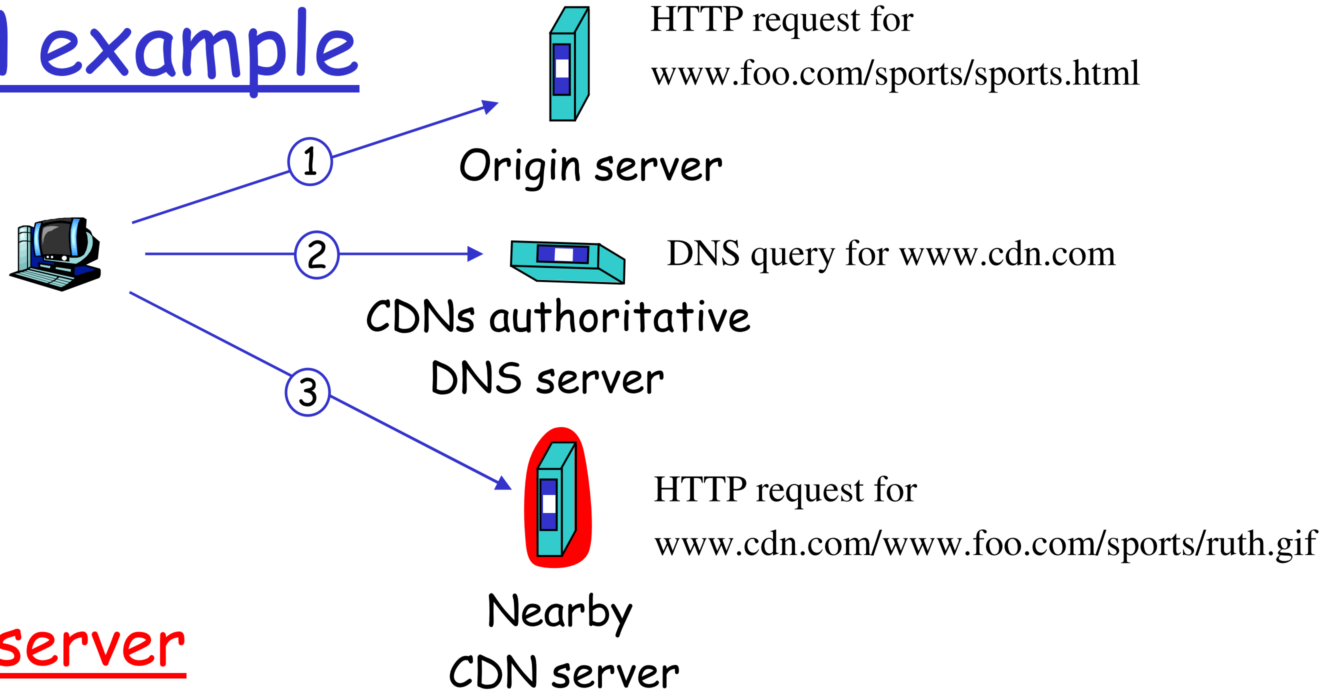
- The content providers are the CDN customers.

Content replication

- CDN company installs hundreds of CDN servers throughout Internet
  - in lower-tier ISPs, close to users
- CDN replicates its customers' content in CDN servers. When provider updates content, CDN updates servers



CDN example



HTTP request for
www.foo.com/sports/sports.html

Origin server

DNS query for www.cdn.com

CDNs authoritative
DNS server

HTTP request for
www.cdn.com/www.foo.com/sports/ruth.gif

Nearby
CDN server

origin server

- www.foo.com
- distributes HTML
- Replaces:

  http://www.foo.com/sports.ruth.gif

  with

  http://www.cdn.com/www.foo.com/sports/ruth.gif

  URL rewriting

CDN company

- cdn.com
- distributes gif files
- uses its authoritative DNS server to route redirect requests

More about CDNs

routing requests

- CDN creates a "map", indicating distances from leaf ISPs and CDN nodes
- when query arrives at authoritative DNS server:
  - server determines ISP from which query originates
  - uses "map" to determine best CDN server

not just Web pages

- streaming stored audio/video
- streaming real-time audio/video
  - CDN nodes create application-layer overlay network

N.B. DNS TTL values must be set so that DNS responses are not cached for long (increase DNS traffic)