Chapter 2
Application Layer

Computer Networking: A Top Down Approach Featuring the Internet, 2nd edition.
Jim Kurose, Keith Ross
Addison-Wesley, July 2002.

Risposta alla domanda: comitati di standardizzazione? Chi paga per tutto questo?

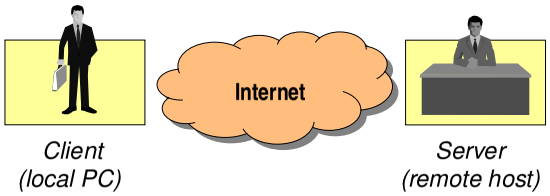
- Personale fisso minimo
- Le attivita' nei comitati di standardizzazione, organizzazione di conferenze, editorial board quasi totalmente su base volontaria
  - -essere a capo di un IETF workpackage e' prestigioso!! Bisogna essere una persona di riferimento nell'area
  - -essere invitati a far parte del comitato organizzativo o TPC di conferenze importanti o negli editorial board di journal idem
  - -Solo accademia? NO!! Per una industria avere propri uomini a capo di workpackage significa influenzare/dirigire il corso della standardizzazione in un settore (importante per carriera nel settore industriale)
  - -Standardizzazione importante: comprereste un cellulare se non funzionasse piu' nulla una volta che uscite dalla copertura del vostro operatore?

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

I servizi di IP: esempio il WEB

Basic scenario



Client wants to retrieve a web page. What happens?

What is a "page" on the web?

a resource (i.e a file), specified by a **URL: Uniform Resource Locator**.

e.g. my home page:

HTTP://cerbero.elet.polimi.it/people/bianchi/index.html

The three components of an URL

1. *Protocol (also called "scheme")*
   - *how can a page be accessed?* **(application protocol used)**
     - **http**://cerbero.elet.polimi.it/people/bianchi/index.html
2. *Host name*
   - *Where is the page located?* **(symbolic or numeric location)**
     - http://**cerbero.elet.polimi.it**/people/bianchi/index.html
3. *File (resource) name*
   - *What is the page called?* **(with full path)**
     - http://cerbero.elet.polimi.it/**people/bianchi/index.html**

Network applications: some jargon

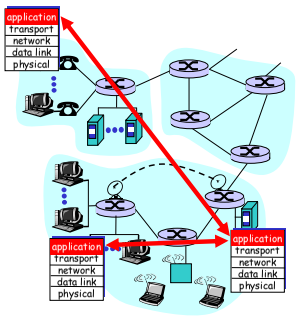
Process: program running within a host.
- within same host, two processes communicate using interprocess communication (defined by OS).
- processes running in different hosts communicate with an application-layer protocol (defining message format, which message to exchange and in which order)

user agent: interfaces with user "above" and network "below".
- implements user interface & application-level protocol
  - Web: browser
  - E-mail: mail reader
  - streaming audio/video: media player

Applications and application-layer protocols

Application: communicating, distributed processes
- e.g., e-mail, Web, P2P file sharing, instant messaging
- running in end systems (hosts)
- exchange messages to implement application

Application-layer protocols
- one "piece" of an app
- define messages exchanged by apps and actions taken
- use communication services provided by lower layer protocols (TCP, UDP)



App-layer protocol defines

- Types of messages exchanged, eg, request & response messages
- Syntax of message types: what fields in messages & how fields are delineated
- Semantics of the fields, ie, meaning of information in fields
- Rules for when and how processes send & respond to messages

Public-domain protocols:
- defined in RFCs
- allows for interoperability
- eg, HTTP, SMTP

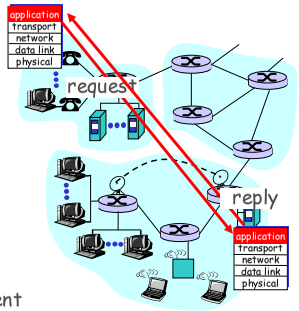
Proprietary protocols:
- eg, KaZaA

Client-server paradigm

Typical network app has two pieces: *client* and *server*

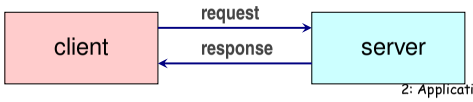
Client:
- initiates contact with server ("speaks first")
- typically requests service from server,
- Web: client implemented in browser; e-mail: in mail reader

Server:
- provides requested service to client
- e.g., Web server sends requested Web page, mail server delivers e-mail



Architettura Client-Server

- Un processo client è solo in grado fare richieste di servizio (informazioni) e di interpretare le risposte
- Un processo server ha solo il compito di interpretare le richieste e fornire le risposte
- Se è necessario nello stesso host sia fare richieste che fornire risposte vengono usati due processi, client e server.
- Un protocollo applicativo per un'architettura client-server rispecchia questa divisione di ruoli e prevede messaggi di richiesta (request) generati dal lato client e messaggi di risposta (response) generati dal server



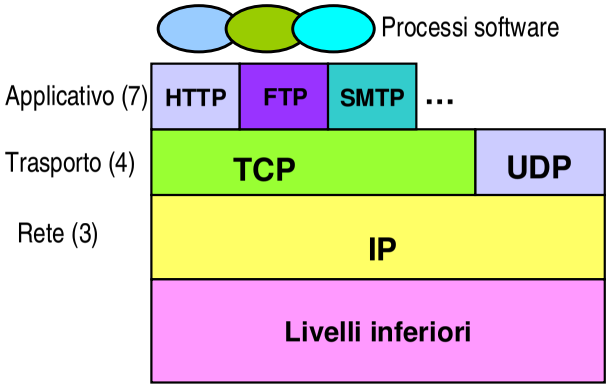
Programmi Client e Server

- Distinguiamo tra programma (software) e processo, istanza del programma in esecuzione su un host
- Un processo server è in esecuzione a tempo illimitato sul proprio host (daemon) e viene attivato mediante una *passive open*
- Un processo client viene attivato solo al momento di fare le richieste e viene attivato mediante una *active open* su richiesta dell'utente o di altro processo applicativo
- la passive open del server fa si che da quel momento il server accetti richieste dai client
- l'active open del client richiede l'indicazione dell'indirizzo e della porta del client

Addressing processes:

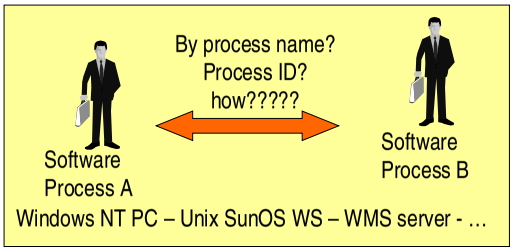
- For a process to receive messages, it must have an identifier
- Every host has a unique 32-bit IP address
- Q: does the IP address of the host on which the process runs suffice for identifying the process?
- Answer: No, many processes can be running on same host
- Identifier includes both the IP address and port numbers associated with the process on the host.
- Example port numbers:
  - HTTP server: 80
  - Mail server: 25

Lo stack di Internet



Addressing software processes

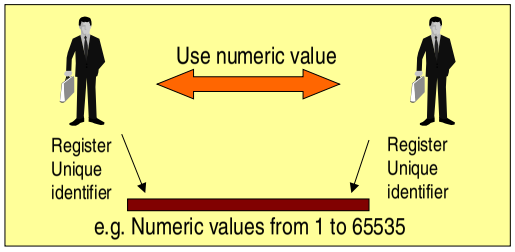
Inside the same machine: an Operating Systems issue



Different OSs = different process naming!

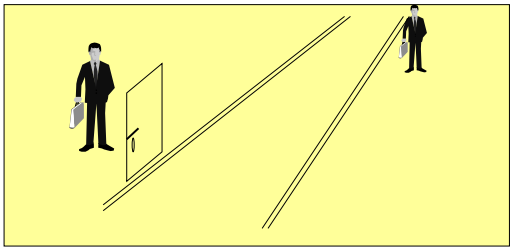
Addressing SW processes

very old solution adopted in most OS



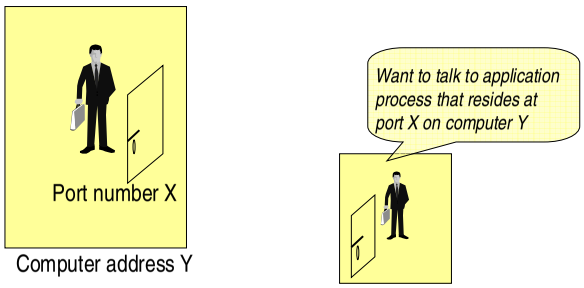
To talk each other, the SW processes need to know their numeric Values.

Port numbers



the "address" of the SW process insite the computer!

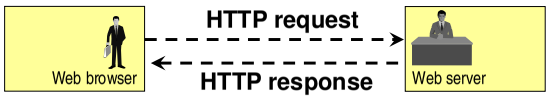
Same addressing scheme works for different machines!!



Re-understanding URLs

HTTP://cerbero.elet.polimi.it/people/bianchi/index.html

protocol — location — Filename



Web browser (SW process) needs to send HTTP Request to Web Server.
Location not enough (it is just the computer address!)

Addressing web servers: (wrong) idea

HTTP://cerbero.elet.polimi.it/….

protocol — location — Filename

Location = computer address!
Protocol = SW process address (HTTP = goto Web Server)
…. No need for port numbers?? …

What if more than one Web Server installed???

URL structure (corrected!)

HTTP://cerbero.elet.polimi.it : *portnumber*/….

protocol — location — Filename

Default value for HTTP: 80

HTTP://cerbero.elet.polimi.it/…

equal to

HTTP://cerbero.elet.polimi.it:80/…

different from

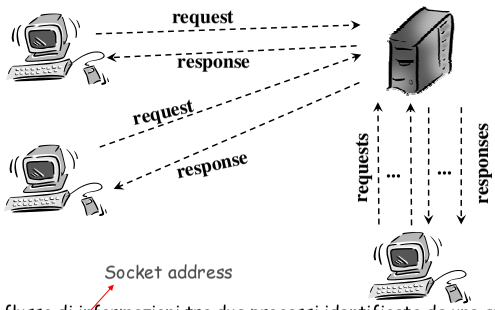
HTTP://cerbero.elet.polimi.it:8080/… (if exists!)

Port numbers

- 16 bit address (0-65535)
- well known port numbers for common servers
  - FTP 20, TELNET 23, SMTP 25, HTTP 80, POP3 110, … (full list: RFC 1700)
- number assignment (by IANA)
  - 0 not used
  - 1-255 reserved for well known processes
  - 256-1023 reserved for other processes
  - 1024-65535 dedicated to user apps

Programmi Client e Server

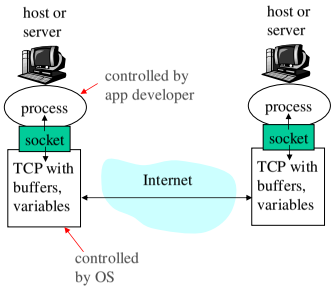
- Normalmente più client possono inviare richieste ad uno stesso server
- Un client può fare più richieste contemporanee



- Un flusso di informazioni tra due processi identificato da una quadrupla
(indirizzo IP sorgente, porta sorgente, indirizzo IP destinazione, porta destinazione)
- Di solito il client NON USA unwell known port #
  - OS assegna un num. di porta disponibile

Processes communicating across network

- process sends/receives messages to/from its socket
- socket analogous to door
  - sending process shoves message out door
  - sending process assumes transport infrastructure on other side of door which brings message to socket at receiving process



- API: (1) choice of transport protocol; (2) ability to fix a few parameters (lots more on this later)

Socket API

- descriptor = socket(protofamily,type,protocol)

crea un socket e ne restituisce un descrittore

Protofamily: PF_INET per TCP/IP; Type: tipo di comunicazione usata dal socket (SOCK_STREAM connection-oriented, SOCK_DGRAM connectionless)

- close(socket)
- bind(socket,localaddr,addrlen)

al server: associa ad un socket creato IP address e porta

- listen(socket,queuesize)

usata dal server per chiedere una passive open

- connect(socket,saddress,saddresslen)

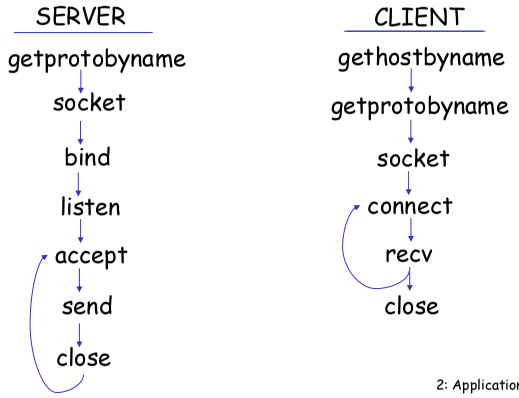
Stabilisce una connessione con un server con address e protocol port number specificati in saddress attraverso il socket del client 'socket'

- newsock=accept(socket,caddress,caddresslen)

per accettazione lato server e creazione del socket per quella connessione

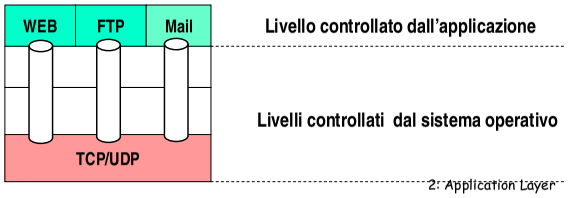
- send(socket,data,length,flags)
- recv(socket,buffer,length,flags)

Sequence of Socket Procedure Calls



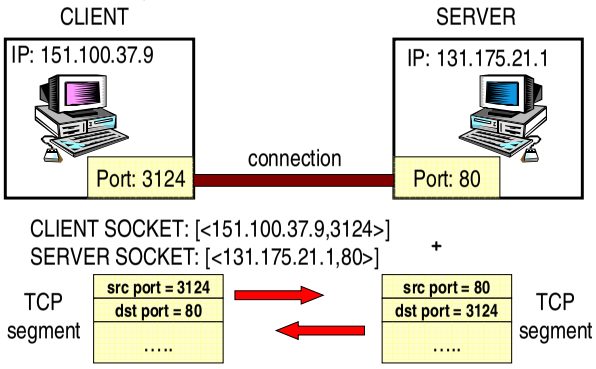
Interazione coi livelli inferiori: architettura TCP/IP

- I protocolli applicativi che si appoggiano sull' Internet Protocol si appoggiano direttamente sul protocollo di trasporto TCP/UDP
- Lo scambio di messaggi fra i processi applicativi avviene utilizzando i servizi dei livelli inferiori attraverso i SAP (Service Access Point)
- I protocolli di trasporto estendono il servizio di trasporto in rete dell'info tra due end systems offerto da IP consentendo il trasporto dell'info tra due PROCESSI APPLICATIVI in esecuzione sui due end systems
- Ogni processo è associato ad un SAP. I SAP tra livello applicativo e di trasporto sono le *porte* e l'insieme porta TCP/UDP e indirizzo IP viene chiamato *socket*
- IP address = NSAP (Network Service Access Point)

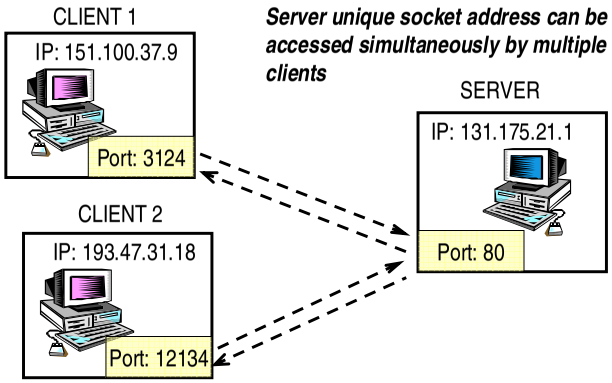


Connections

identified by socket addresses at its ends

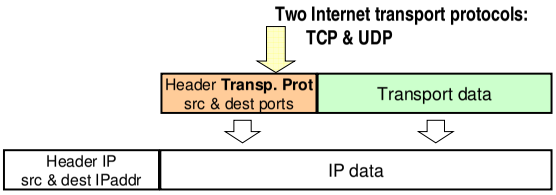


Managing multiple connections



Socket addresses: where?

- Ports: in the Internet Transport protocol header (TCP or UDP)
- IP addresses: In the IP packet header



IP packets travel in the network based on IP address;

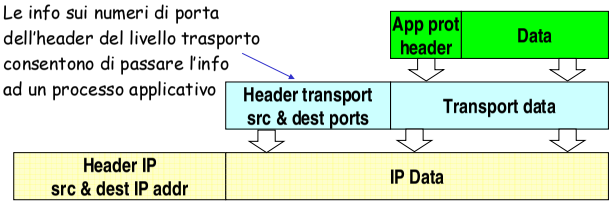
Once they reach destination, they are delivered to app based on port #

Sockets and Internet transport protocols

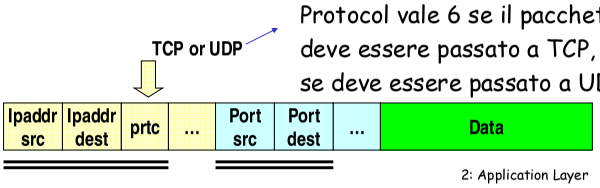
- Two transport protocols in Internet:
  - TCP (reliable, connection-oriented)
  - UDP (unreliable, connectionless)
- When opening socket, needs to specify which transport to use!

Socket addresses (refined)

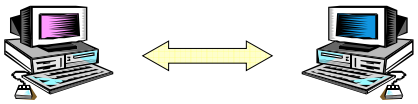
Le info sui numeri di porta dell'header del livello trasporto consentono di passare l'info ad un processo applicativo



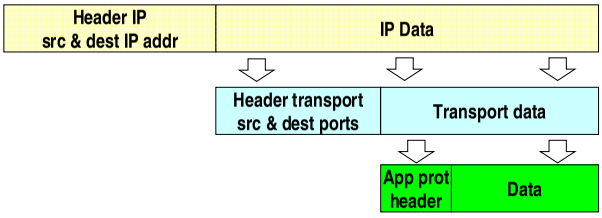
Nell'header IP il campo Protocol vale 6 se il pacchetto deve essere passato a TCP, 17 se deve essere passato a UDP



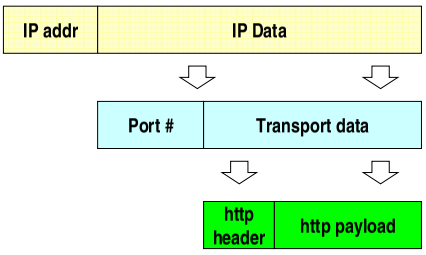
The Internet level view



Information units travelling in the network: IP packets



Demultiplexing at receiver



IP SW: checks IP packet; Sends to transport sw selects whether UDP or TCP
Transport demux

Transport SW: checks segment; Sends to application sw based on Port number
Application demux

Information entering app Software (managed by app Developer)

Programmi Client e Server

- Un client può essere eseguito in modalità parallela o seriale
  - esempio: invia più richieste in parallelo per i file che compongono una pagina web
- Anche un server può essere eseguito in modalità parallela o seriale
- Normalmente i server che usano TCP vengono eseguiti in modalità parallela e sono dunque in grado di rispondere a più richieste contemporaneamente
- Con ognuno dei client viene aperta una connessione TCP che viene mantenuta per il tempo necessario a scambiare richieste e risposte
- La gestione delle procedure per ciascun client collegato avviene mediante la generazione di processi figli (per clonazione del processo o uso di processi multi-thread)

What transport service does an app need?

Data loss
- some apps (e.g., audio) can tolerate some loss
- other apps (e.g., file transfer, telnet) require 100% reliable data transfer

Timing
- some apps (e.g., Internet telephony, interactive games) require low delay to be "effective"

Bandwidth
- some apps (e.g., multimedia) require minimum amount of bandwidth to be "effective"
- other apps ("elastic apps") make use of whatever bandwidth they get

Transport service requirements of common apps

Application	Data loss	Bandwidth	Time Sensitive
file transfer	no loss	elastic	no
e-mail	no loss	elastic	no
Web documents	no loss	elastic	no
real-time audio/video	loss-tolerant	audio: 5kbps-1Mbps video:10kbps-5Mbps	yes, 100's msec
stored audio/video	loss-tolerant	same as above	yes, few secs
interactive games	loss-tolerant	few kbps up	yes, 100's msec
instant messaging	no loss	elastic	yes and no

Internet transport protocols services

TCP service:
- *connection-oriented:* setup required between client and server processes
- *reliable transport* between sending and receiving process
- *flow control:* sender won't overwhelm receiver
- *congestion control:* throttle sender when network overloaded
- *does not providing:* timing, minimum bandwidth guarantees

UDP service:
- unreliable data transfer between sending and receiving process
- does not provide: connection setup, reliability, flow control, congestion control, timing, or bandwidth guarantee

Q: why bother? Why is there a UDP? Which service does it add to IP?

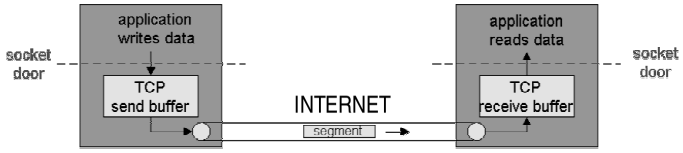
Multiplexing/Demultiplexing

Error Control

Internet apps: application, transport protocols

Application	Application layer protocol	Underlying transport protocol
e-mail	SMTP [RFC 2821]	TCP
remote terminal access	Telnet [RFC 854]	TCP
Web	HTTP [RFC 2616]	TCP
file transfer	FTP [RFC 959]	TCP
streaming multimedia	proprietary (e.g. RealNetworks)	TCP or UDP
Internet telephony	proprietary (e.g., Dialpad)	typically UDP

Why it is trivial (!) to write networking apps?



- Application software duties:
  - open socket (e.g. C, C++, JAVA function call, OS call, external library primitive)
  - Injects message in its own socket
  - being confident message is received on the other side
- TCP software: in charge of managing segments!
  - reliable message transport when TCP used
  - Segmentation performed by TCP transmitter
  - Receive buffer necessary to ensure proper packet's order & reassembly

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

Breve storia del Web

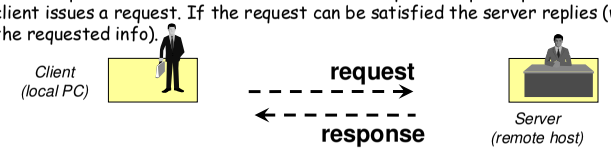
- 1945: Vannevar Bush in 'As we may think' proposes Memex to extend human memory via mechanical means. V. Bush wanted to make the existing store of knowledge (fastly growing) more accessible to mankind. 'A memex is a device in which an individual stores all his books, records, and communications, and which is mechanized so that it maybe consulted with exceeding speed and flexibility. It is an enlarged supplement to his memory. …Wholly new forms of encyclopedias will appear ready made with a mesh of associative trails running through them'
- 1965 Ted Nelson coins the term 'hypertext' to describe nonsequential writing that presents information as a collection of linked nodes. 'Readers can pursue the information in a variety of ways by navigating from one node to another'
- 1970s scientific community communicate through ARPANET (exchanging e-mails, file transfers..)
- 1989 Tim Berners-Lee creates the World Wide Web (networked application that links users and services distributed across computers around the world. Users can view pages, search for information, send and receive e-mails, initiate business transactions etc. Tim Berners Lee proposed linking info presented on various machines at CERN (European Laboratory for Particle Physics, Geneva). Other systems had been created to download files (FTP)from remote computers, search for information (Gopher) etc. Why did the Web succed? 1)SIMPLE INTERFACE 2) ENHANCED FEATURES 3) USE OF HYPERTEXT

Breve storia del Web

- 1991 First browser and server introduced
- 1993 The Web consisted of around 50 servers
- 1993 First release of Mosaic browser for windows (written by Marc Andreesen and Eric Bina). The web accounted for 1% of the traffic of the Internet
- Late 1990s → Web responsible for over 75% of the Internet traffic!! Hundreds millions of users; millions of web sites. Reasons for success: graphical user interface, ease of publishing new content
- Web used for e-business, business to business, interactions between users (chatrooms and games). Web increasingly used also to access private or proprietary data.
  - Web traffic: small number of Web pages accounts for the majority of web accesses
  - from text to images, animations, video files (increased storage needed at the Web servers, increased time for downloading, increased load on the network)
  - Dynamically generated content (e.g. search engines generates pointers based on keywords)
  - Web used for banking, e-business;important personl info (e.g. credit card info transferred)→ security, authentication needed

Basic 'bricks' of the Web

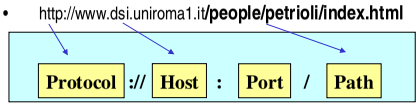
- Uniform Resource Location (URL) –allows to identify a web resource
- Hypertext Markup Language (HTML) –provides a standard representation for hypertext documents in ASCII format. Allows authors to format text, reference images, embed hypertext links. Software tools have been developed for generating HTML files (e.g. Netscape Composer, FrontPage) so that editing a web page is straightforward and doe not require HTML knowledge.
- Hypertext Transfer Protocol (HTTP)—it is the protocol web components use to communicate. Defines the format and meaning of messages exchanged between Web components such as clients and servers. Request-response protocol: the client issues a request. If the request can be satisfied the server replies (with the requested info).



- The Web and the Internet: The Internet provides a global communication infrastructure allowing Web clients to access a wide variety of Web Servers throughout the world

The three components of an URL

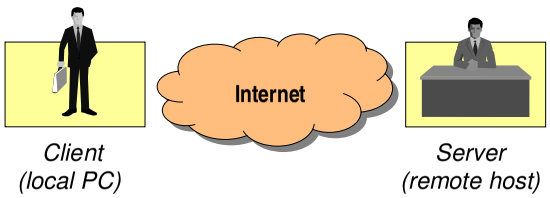
1. *Protocol (also called "scheme")*
   - *how can a page be accessed?* **(application protocol used)**
     - **http**://www.dsi.uniroma1.it/people/petrioli/index.html
2. *Host name*
   - *Where is the page located?* **(symbolic or numeric location)**
     - http://WWW**.dsi.uniroma1.it**/people/petrioli/index.html
3. *File (resource) name*
   - *What is the page called?* **(with full path)**
     - http://www.dsi.uniroma1.it**/people/petrioli/index.html**



Web Content

- The Web is a collection of resources or objects distributed throughout the Internet. Each object is a network- accessible document or service, which maybe available in different formats (e.g. HTML, poscript, PDF). A resource maybe a static file on a machine or maybe dynamically generated upon request.
- A web page consists of a container resource such as HTML file which may include links to one or more embedded resources such as images and animations.
- HTTP 1.0
  - Each HTTP transfer involves a single object
  - Downloading a web page involves separate HTTP transfer (1 for the container resource, one for each of the embedded objects)
- Request-response protocol. A user agent (e.g. a browser) initiates an HTTP request and handles the response. The program that actually sends the HTTP request and receive the response is called client. The origin server is the program that (possibly generates and) provides a Web resource. It receives an HTTP request and sends a response.
- The client may send its HTTP request to an intermediary on the client-server path called proxy (e.g. caching frequently accessed web pages). A proxy acts both as client and server.

Basic scenario



Client wants to retrieve a web page. What happens?

What is a "page" on the web?

a resource (i.e a file), specified by a
URL: Uniform Resource Locator.

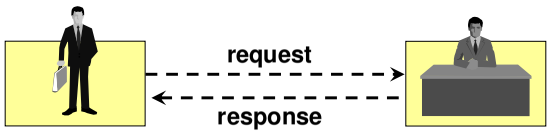
HTTP://cerbero.elet.polimi.it/people/bianchi/index.html

1. Protocol

- HTTP: the protocol of the WWW
  - version 1.0, RFC 1945, may 1996
  - version 1.1, RFC 2068 (jan97), RFC 2616 (jun99)

(but also **FTP**: file transfer protocol, **TELNET**: opens a telnet window, **FILE**: access local file etc.)

- HTTP: Request-response protocol. A request message is sent from the client to a recepient server. The recipient server send a response message back.



- Different requests for the same URL can result in different responses due to the time of request, changes in the resource, the request header fields.

2. Location – host name

Specifies where is the page located:

- **on which host**
  - Humans understand names;
  - Machines prefer numbers!
  - Domain Name System (DNS) protocol:
    - translates names in numbers

www.elet.polimi.it
DNS
131.175.21.1

3. File names

(several shortcuts handled server side)

page name non mandatory

http://cerbero.elet.polimi.it/people/bianchi/
http://www.yahoo.com
http://www.sun.com/products

Unix style user id shortcuts

http://www.cs.columbia.edu/~hgs

Il server e' in grado di estrapolare il path completo verso la risorsa

Various page extensions (with server side meaning: no client side interpretation!)

*.htm *.html *.asp *.jsp *.php ……..

Pagine dinamiche

Case sensitivity

- Protocol: non case-sensitive
- Location: non case-sensitive
- File name: case-sensitive

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

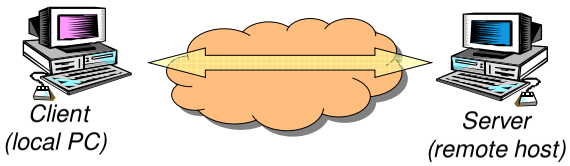
equal to

Http://Cerbero.Elet.PoliMi.it/people/bianchi/index.html

different from

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

Refined scenario

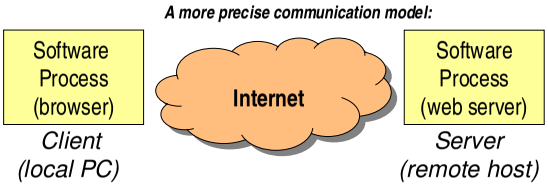


Client wants to retrieve a remote resource:
/people/bianchi/index.html
On the server
cerbero.elet.polimi.it
By using the application layer protocol
http
What happens?

Being more precise

- Who is the "client"?
  - The web browser
- Who is the server?
  - The web server
- What is the Networking application?
  - The WWW
- What is the application layer protocol?
  - HTTP

A more precise communication model:



Networking Application vs Application Layer Protocol

- Networking application
  - software processes
  - on possibly different end-systems
  - that communicate each other by exchanging messages
- Application Layer Protocol
  - define format of messages
  - define order of messages exchanged
  - define actions taken on receipt of a message

An application-layer protocol: *is only one piece (although a big piece!!) of a network application*

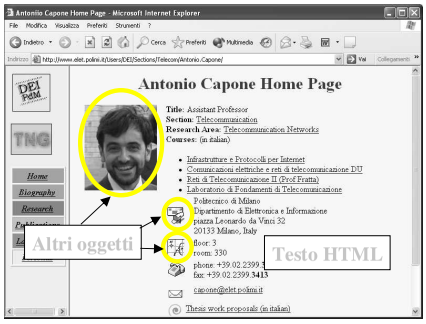
Example: the WWW application

many components, including:

- standard for document formats
  - HTML & HTML interpreters
- Web browsers
  - Netscape Navigator, Internet Explorer,...
- Web servers
  - Apache, Microsoft and Netscape servers, ...
- Back-end Data Base DB connectivity, programming/scripting languages
  - Public domain (e.g. MYSQL) or commercial (Oracle, ...)
  - ASP (active Server Pages), JSP (Java Server Pages), PHP (Perl Helper Pages) scripting embedded in html; e.g. CGI to connect to external program
- An application-layer protocol
  - the HyperText Transfer Protocol (HTTP)

Trasporto dei messaggi

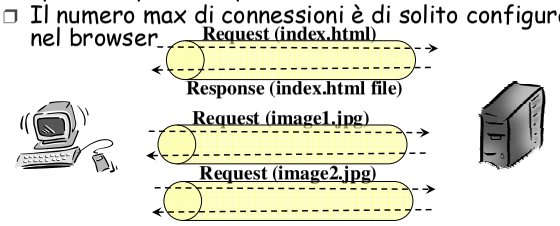
- Supponiamo che un client richieda una pagina HTML di un server al cui interno sono contenuti i riferimenti ad altri oggetti (ad esempio 10 figure che compongono la pagina e che occorre visualizzare insieme al testo HTML).



- **nel trasferimento dell'insieme di oggetti sono possibili 2 modalità**
  - **Non-persistent connection (default mode di HTTP 1.0)**
  - **Persistent connection (default mode di HTTP 1.1)**

Non persistent

- Viene aperta una connessione per una sola request-response: inviato l'oggetto, il server chiude la connessione
- La procedura viene ripetuta per tutti i file collegati al documento HTML base
- Le connessioni TCP per più oggetti possono essere aperte in parallelo per minimizzare il ritardo
- Il numero max di connessioni è di solito configurabile nel browser



Nonpersistent HTTP (HyperText Transfer Protocol)

Suppose user enters URL
`www.someSchool.edu/someDepartment/home.index`

(contains text, references to 10 jpeg images)

1a. HTTP client initiates TCP connection to HTTP server (process) at www.someSchool.edu on port 80

1b. HTTP server at host www.someSchool.edu waiting for TCP connection at port 80. "accepts" connection, notifying client

2. HTTP client sends HTTP *request message* (containing URL) into TCP connection socket. Message indicates that client wants object someDepartment/home.index

3. HTTP server receives request message, forms *response message* containing requested object, and sends message into its socket

time

Nonpersistent HTTP (cont.)

time

4. HTTP server closes TCP connection.

5. HTTP client receives response message containing html file, displays html. Parsing html file, finds 10 referenced jpeg objects

6. Steps 1-5 repeated for each of 10 jpeg objects

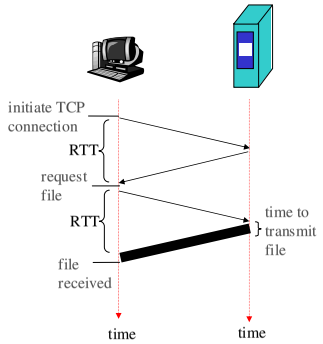
Response time modeling

Definition of RRT: time to send a small packet to travel from client to server and back.

Response time:

- one RTT to initiate TCP connection
- one RTT for HTTP request and first few bytes of HTTP response to return
- file transmission time

total = 2RTT+transmit time

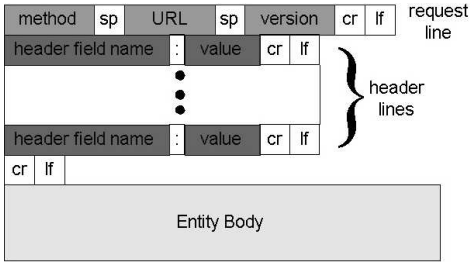


HTTP messages

- Sequence of octets sent over a transport connection; basic unit of communication in HTTP
- Can be a request message from a client to a server or
- a response sent back from a sender to a client

HTTP Messages

Request



HTTP Message

Methods:

GET	E' usato quando il client vuole scaricare un documento dal server. Il documento richiesto è specificato nell'URL. Il server normalmente risponde con il documento richiesto nel corpo del messaggio di risposta.
HEAD	E' usato quando il client non vuole scaricare il documento ma solo alcune informazioni sul documento (come ad esempio la data dell'ultima modifica). Nella risposta il server non inserisce il documento ma solo degli header informativi.
POST	E' usato per fornire degli input al server da utilizzare per un particolare oggetto (di solito un applicativo) identificato nell'URL.
PUT	E' utilizzato per memorizzare un documento nel server. Il documento viene fornito nel corpo del messaggio e la posizione di memorizzazione nell'URL.

- **Altri methods:**
  - **DELETE, LINK, UNLINK, ...**

Per inviare info al server
E' possibile anche usare GET
GET /sercah.cgi?string=greek-architects HTTP/1.0

Header

Header name : **Header value**

- Gli header servono per scambiare informazione di servizio aggiuntiva
- E' possibile inserire più linee di header per messaggio
- Esempi

Cache-control	Informazione sulla cache
Accept	Formati accettati
Accept-language	Linguaggio accettato
Authorization	Mostra i permessi del client
If-modified-since	Invia il doc. solo se modificato
User-agent	Tipo di user agent

Header Types

- General: used in request and response messages
- Request header (e.g. to express preferences on the nature of the response, include additional info with the request, to specify a constraint on the server in handling the request
- Response header: to provide additional info about the response or to request additional info from the user
- Entity header (both in request and response messages) to provide info on the entity such as the last time it was modified

Header Types-Request

- General: used in request and response messages

Date	Indicates the date and time of the message origination, e.g. Date: Tue 16 May 2000 11:29:32 GMT
Pragma	Permits to send directives to the recipient, requesting it to behave in a particular way while handling a request or response. Pragma: no-cache informs proxies on the path not to return a cached copy

- Request header (e.g. to express preferences on the nature of the response, include additional info with the request, to specify a constraint on the server in handling the request

Authorization	Includes credentials required to access a resource (e.g. Authorization: Basic YXZpYXRpS29IDizDizM1NA== where Basic is the authentication scheme and YXZpYXRpS29IDizDizM1NA== an encoding of user id and password)
From	The user may include e-mail address for identification (From: entry@mostrax.com)
If-Modified-Since	Indicates not to return a copy of the resource if it has not been modified after teh specified date. Used for caching at a proxy or browser (saves the time needed for downloading of the resource). GET /foo.html HTTP/1.0 If-Modified-Since: Sun, 21 May 200 07:00:25 GMT
Referer	Includes the URL from which the requested URL was obtained. Can be used for locating obsolete links. Referer: http://www.cnn.com
User-Agent	User browser used, client machine OS, etc User-Agent: Mozilla/4.04 [en]C-WorldNet (win95;I)

Header Types-Request

- Entity header (both in request and response messages) to provide info on the entity such as the last time it was modified

Allow	Indicates the list of valid methods that can be applied to a resource PUT /foo.html HTTP/1.0 Allow: HEAD, GET, PUT
Content-Type	Media type of the entity body (e.g. image/gif, text/hmtl etc.)
Content-Encoding	Indicates how the resource epresentation can be decoded into the format indicated in the Content-Type field (e.g. if the file has been compressed with gzip Content-Encoding:g-zip
Content-Length	Length of the entity body (bytes). Allow to check whether all the body was received
Last-Modified	Specify the time at which the resource was modified last

Un esempio…

- Esempio: di richiesta oggetto

HTTP è testuale (ASCII)

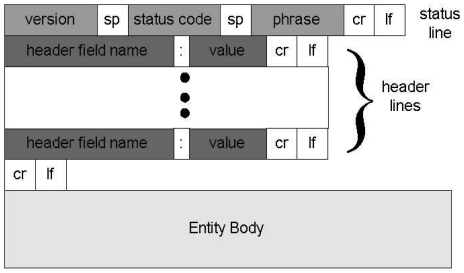
```
GET /ntw/index.html HTTP/1.0
Date: Wed, 22 Mar 2000 09:09:01 GMT
Pragma: No-cache
From: gorby@moskvax.com
User-agent: Mozilla/4.3
```

- Esempio: risposta

```
HTTP/1.0 200 OK
Date: Wed, 22 Mar 2000 09:10:01 GMT
Server: Apache/1.3.0 (Unix)
Content-Length: 6821
Last-Modified: Mon, 22 Jun 1998 09:23:24 GMT
Content-Type: text/html
data data data data data ...
```


Message

Response



Header Types-Response

- Response header: to provide additional info about the response or to request additional info from the user

Location	Used to redirect the request to where the resource can be found Location: http://www.foo.com/level1/twosdown/Location.html
Server	Origin server SW version number and configuration related info Server: Apache/1.2.6 Red Hat
WWW-Authenticate	Request tor etransmit the request with appropriate credentials according to a given scheme

Messaggi

Status codes:

- **1xx Informational**
- **2xx Success**

200 OK:	La richiesta ha avuto successo; l'informazione è inclusa
201 Created:	La risorsa e' stata creata con successo in seguito ad UN POST (se non puo' essere creata subito 202 Aaccepted)

- **3xx Redirection**

301 Moved Permanently:	L'oggetto è stato spostato nell'URL indicato
302 Moved Temporalily:	L'oggetto è stato spostato nell'URL indicato
304 Not Modified:	L'oggetto non modificato dal tempo incluso nella richiesta

- **4xx Client error**

400 Bad Request:	errore generico (sintassi errata(irriconoscibile)
401 Unauthorized:	Accesso senza necessari account e passwd
404 Not Found:	l'oggetto non esiste sul server

- **5xx Server error**

500 Internal server error	Generico. Errore o guasto nel server
501 Not implemented	Funzione non implementata
503 Service unavailable	Servizio non disponibile

Un esempio

HTTP è testuale (ASCII)

- Esempio: di richiesta oggetto

```
PUT /motd HTTP/1.0
Date: Wed, 22 Mar 2000 08: 10:07 GMT
From: gorby@moskvax.com
User-agent: Mozilla/4.0
Content-length:23
Allow:GET,HEAD,PUT
data data data data data ...
```

- Esempio: risposta

```
HTTP/1.1 200 OK
Date: Wed, 22 Mar 2000 09: 10:07 GMT
Server: Apache/1.3.0 (Unix)
...
```


Cache locale: get condizionato

- E' possibile evitare di scaricare oggetti memorizzati nella memoria locale se non sono stati modificati

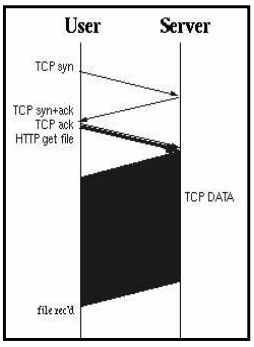
```
Client:
GET /fruit/kiwi.gif HTTP/1.0
User-agent: Mozilla/4.0
Accept: text/html, image/gif, image/jpeg
If-modified-since: Mon, 22 Jun 1998 09:23:24
```

```
Server:
HTTP/1.0 304 Not Modified
Date: Wed, 19 Aug 1998 15:39:29
Server: Apache/1.3.0 (Unix)
(empty entity body)
```

- E' possibile anche usare il metodo HEAD

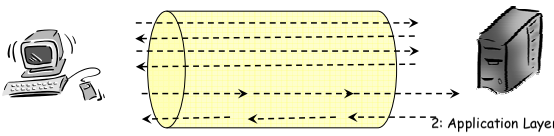
Performance drawbacks

- Mandatory roundtrips
  - TCP three-way handshake
  - get request, data return
  - new connections for each inlined image (parallelize)



Persistent connection (HTTP v1.1)

- Nel caso *persistent* il server non chiude la connessione dopo l'invio dell'oggetto
- La connessione rimane aperta e può essere usata per trasferire altri oggetti della stessa pagina web o anche più pagine
- I server chiudono di solito la connessione sulla base di un *time-out*
- *without pipelining*:il client invia una nuova richiesta solo dopo aver ricevuto la risposta per la precedente
  - *with pipelining:* più richieste vengono inviate consecutivamente dal client



Persistent HTTP

Nonpersistent HTTP issues:
- requires 2 RTTs per object
- OS must work and allocate host resources for each TCP connection
- but browsers often open parallel TCP connections to fetch referenced objects

Persistent HTTP
- server leaves connection open after sending response
- subsequent HTTP messages between same client/server are sent over connection

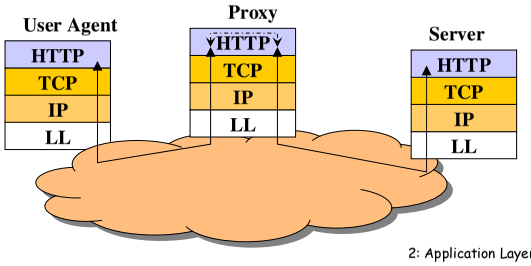
Persistent without pipelining:
- client issues new request only when previous response has been received
- one RTT for each referenced object

Persistent with pipelining:
- default in HTTP/1.1
- client sends requests as soon as it encounters a referenced object
- as little as one RTT for all the referenced objects

Proxy

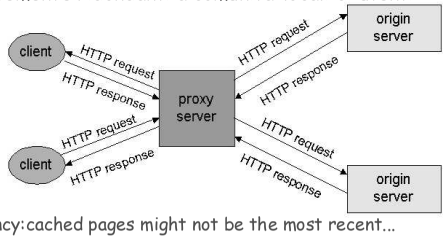
An intermediary program which acts as both a server and a client for the purpose of making requests on behalf of other clients.

- I proxy sono degli instradatori di messaggi di livello applicativo
- Devono essere sia client che server
- Il server vede arrivare tutte le richieste dal proxy (mascheramento degli utenti del proxy)



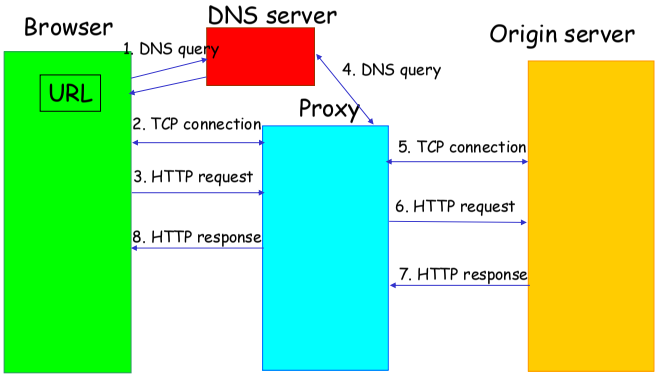
Cache di rete: uso dei proxy

- Compito principale dei proxy è fornire una grande memoria di cache
- Se un documento è contenuto nella cache viene scaricato più velocemente sul client
- Razionale: i server forniscono le stesse info o info fortemente ridondanti a comunita' locali di utenti



- Consistency:cached pages might not be the most recent...
- Ad clickthrough counts:how does Yahoo know how many times you accessed their pages, or ***more importantly***, their ads?

Download atraverso un proxy...



Uso dei Proxy(1)

- Caching: '*storage of a respose obtained earlier for later use, when clients request the same resource*'. Se la pagina richiesta e' nella cache il proxy puo' inviarla immediatamente al client purche' sia fresh (ovvero non sia stata modificata rispetto alla copia in cache)
  - -latency minore MA cache consistency importante. Strong consistency/weak consistency
- Agisce come front-end per un gruppo di utenti che condividono l'accesso al web (caching ma anche suddivisione delle risorse—pro: se piu' utenti richiedono la stessa risorsa ad un server un'unica connessione deve essere stabilita; con—possibili ritardi dovuti al fatto che piu' utenti condividono le stesse risorse)

Uso dei Proxy(2)

- Mascheramento degli utenti del proxy: Il server vede arrivare tutte le richieste dal proxy e quindi non e' in grado di riconoscere l'utente. Il proxy conosce le informazioni di accesso relative agli utenti ma e' considerata un'entita' trusted.
- Filtraggio delle richieste e delle risposte
  - per limitare l'accesso a ceri siti
  - filtrare richieste a siti non adatte ai minori
  - verificare la presenza di virus prima di inviare al client le risposte

Tipi di proxy

- Caching proxy: indica la capacita' di effettuare caching
- Manipolazione dei messaggi
  - Transparent proxy (non modifica le richieste/risposte se non superficialmente, ad esempio fornendo info che permettono di identificare il proxy prima di riinviare il messaggio; info che potrebbero anche solo essere rchieste dal protocollo HTTP)
  - Non-transparent proxy (modifica pesantemente il messaggio, e.g. garantendo l'anonimita' del client, modificando il formato di un messaggio ad esempio con una codifica dell'immagine meno pesante prima di trasferirla al client su un link di bottleneck, cambiando il contenuto del messaggio che sara' espresso in un diverso linguaggio coem richiesto dal client

Entrambi i transparent/non-transparent proxy possono avere associata una cache

HTTP operation with intermediaries

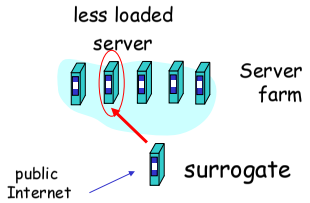
- Three forms of intermediaries:
  - Tunnel: intermediary program acting as a blind relay between two connections. Once active, a tunnel is not considered a party to the HTTP communication. No caching. E.g. in case of encripted communication
  - Gateway: server acting as intermediary for some other server (typically used for 'speaking' to a non HTTP server). Unlike a proxy, a gateway receives requests *as if it were the origin server for the requested resource*; the requesting client may not be aware that it is communicating with a gateway.
  - HTTP Proxy (speaks HTTP, cache usually associated)

Some rules proxies must obey

- Semantic neutrality
  - If you can handle the request locally do it, otherwise (also if the proxy cannot udnerstand teh request) forward it to teh origin server
- Identification (in HTTP 1.1 all proxies must identify themselves, e.g. to avoid potential loops in the path)
- Cannot lie on the HTTP version it supports (it can happen the client speaks HTTP 1.0, the proxy HTTP 1.1 or viceversa)
- Info must be added when returning a resource (e.g. the time since the resource was validated from the origin server).
- Connections must be maintained operational while the client is ON
- Must handle the high number of connections with clients and origin servers
- Proxies passes cookies to the client. They are forbidden to add their cookie-related headers …

Other types of proxies

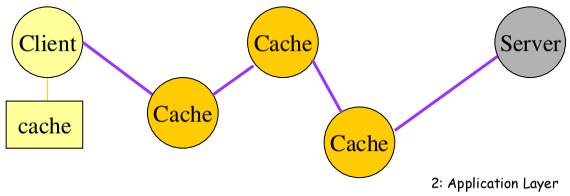
- Reverse proxy (surrogates)
  - Proxy placed close to the origin server to reduce the load on them. Load balancing among servers of a server farm



- Interception proxy. Are transparent (invisible) to the user. All messages pass through the interception proxy that can inyercept them, examine the request, generate a response locally or redirect the request to another place (e.g. to the 'best replica')

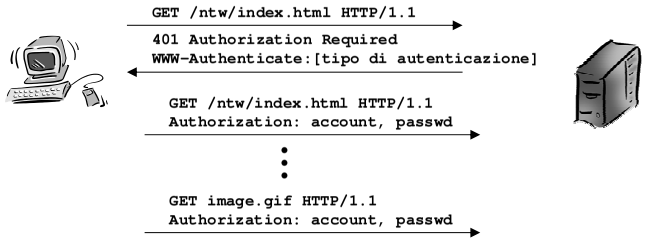
Hierarchical Caching

- The message being transported can cross several caches (e.g. at the department, ISP side, in fronto of the origin server –reverse proxies)
- Decreases communication costs
- Regional/national cache→ high hit ratio



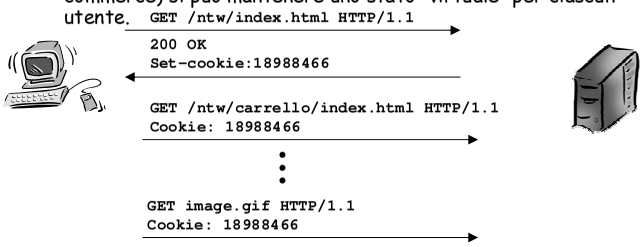
Autenticazione

- 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

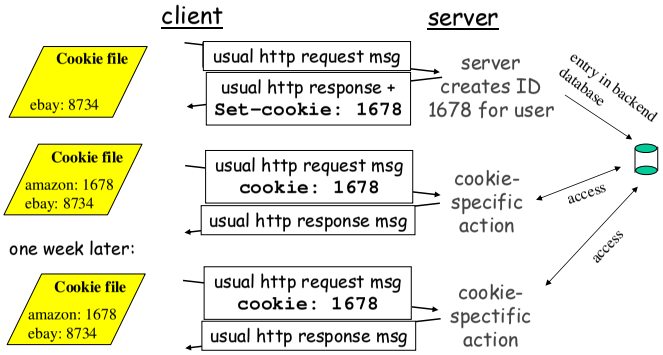


Cookie

- 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
- Cambiando il numero di cookie ad ogni operazioni (ad es. e-commerce) si può mantenere uno stato "virtuale" per ciascun utente.



Cookies: keeping "state" (cont.)

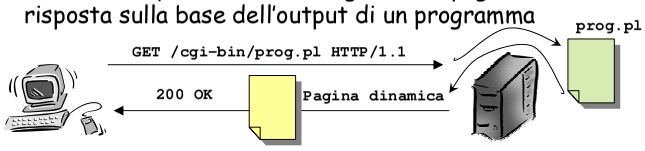


HTML (HyperText Markup Language)

- 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

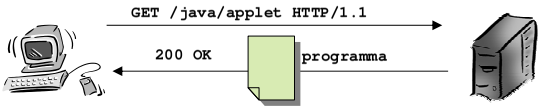
Pagine WEB dinamiche

- 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



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 worlds

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
- Administrators can signal a given site should not be indexed in file robot.txt

User-Agent: Aracnophobia, BlackWindow
Disallow: /*
Similar info could be contained in the HTML tag

- Search Engine

	Agent name
Lycos	Lycos_Spider
AltaVista	Scooter
Google	BackRub
Excite	ArchitextSpider

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 teh serach terms hve appeared?)
    - Precision (how relevant are the retrieved documents?)
- The results have to be ranked

→ More on these topics: Algoritmi III (Algoritmi per il Web)

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)

Symbolic names easy to remember BUT IP addresses shorter (less overhead/computation to manipulate)

Q: map between IP addresses and name ?

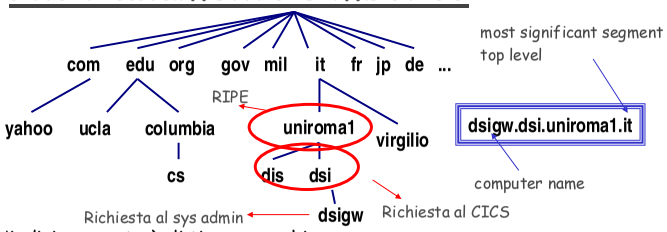
Domain Name System:
- *distributed database* implemented in hierarchy of many *name servers*
- *application-layer protocol* host, routers, name servers to communicate to *resolve* names (address/name translation)

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

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)

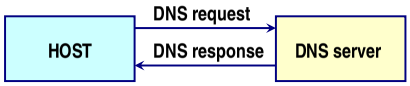
DNS name servers

Why not centralize DNS?
- single point of failure
- traffic volume
- distant centralized database
- maintenance

doesn't *scale!*

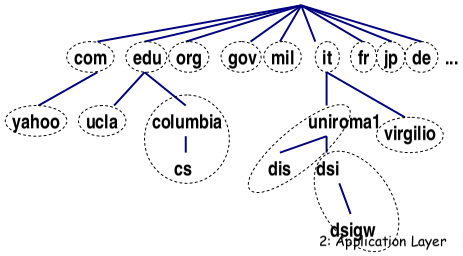
Come ottenere un mappaggio

- 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

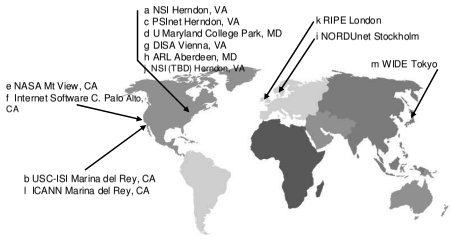
local name servers:
- each ISP, company has *local (default) name server*
- host DNS query first goes to local name server

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!!)

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

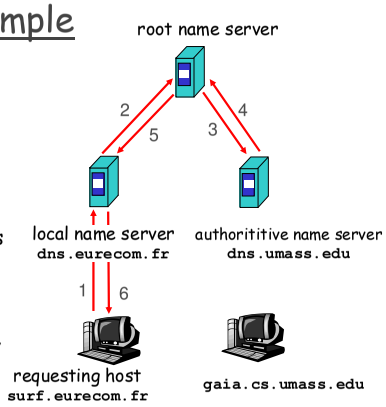


13 root name servers worldwide

Simple DNS example

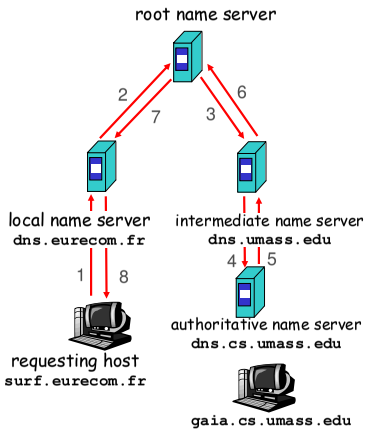
host **surf.eurecom.fr** wants IP address of **gaia.cs.umass.edu**

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

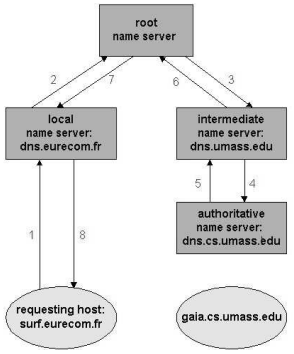


DNS example

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



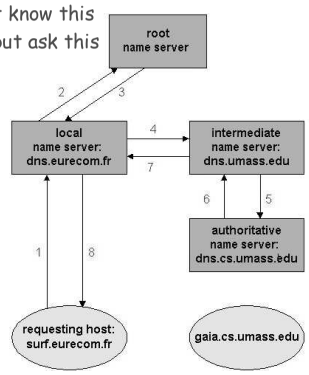
Reperire informazioni: modo ricorsivo



- 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

DNS optimizations

- Replication:
  - E.g si usa il root server piu' vicino
- Caching:
  - Un 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
- I server non-authoritative usano il TTL per settare un time-out per le risposte.

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 IP address 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**

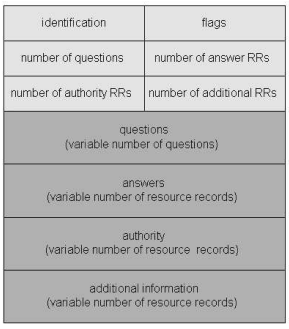
  (elet.polimi.it, mailserver.elet.polimi.it, MX,TTL)

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



question for NS

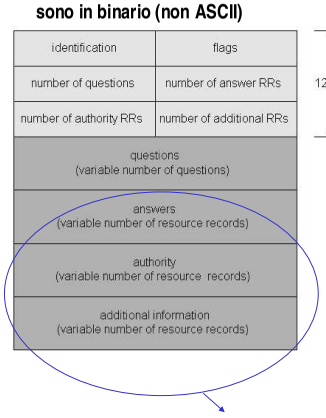
RR answering question

RR pointing to an authority

RR with add. info

Messaggi DNS

sono in binario (non ASCII)



Resource Records

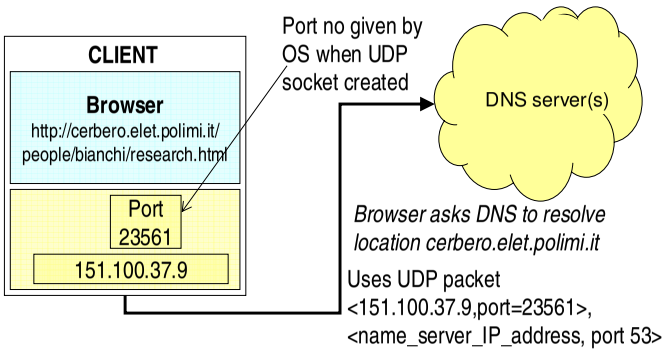
- identification: identificativo coppia richiesta/risposta
- flags: richiesta/risposta, authoritative/non auth., iterative/recursive (richiesta dal client, supporto comunicato dal server),flag che indicano il verificarsi di errori(e.g. dall'authoritative server 'il dominio non esite')
- number of: relativo 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?

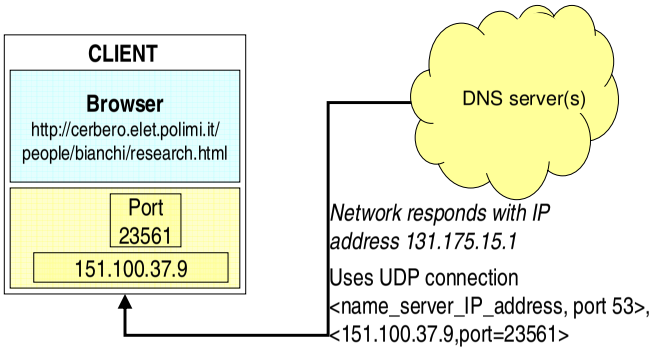
- Less overhead
  - Messaggi corti
  - Tempo per set-up connessione di TCP lungo
  - Un unico messaggio deve essere scambiato tra una coppia di server (nella risoluzione contattati diversi server—se si usasse TCP ogni volta dovremmo Mettere su la connessione!!)
- Se un messaggio non ha risposta entro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

Porta usata per il DNS: 53!!

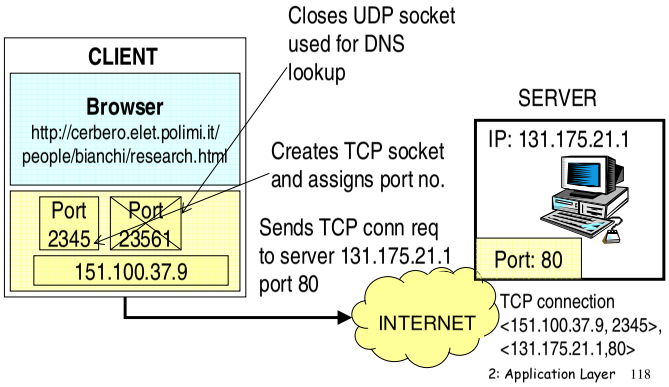
Un esempio: uso di DNS da parte di un client web



opening transport session: client side, step b



opening transport session: client side, step c



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

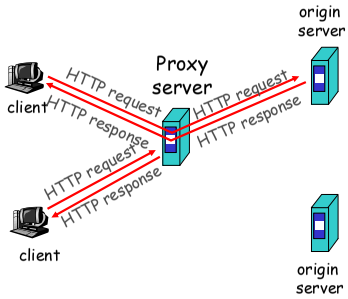
Chapter 2 outline

- 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.
- Reduce the load on the origin server.
- Internet dense with caches enables "poor" content providers to effectively deliver content

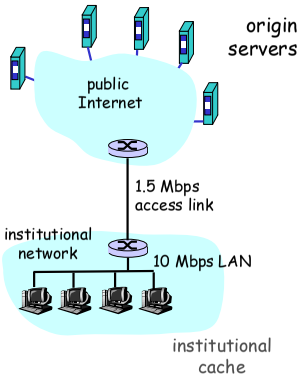
Caching example (1)

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



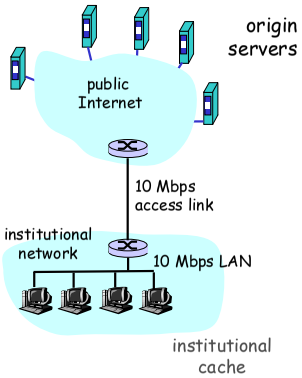
Caching example (2)

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



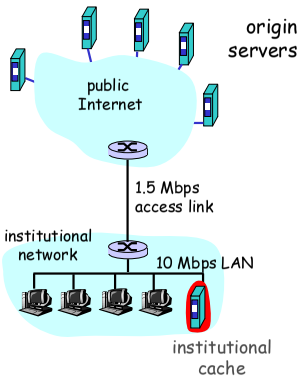
Caching example (3)

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

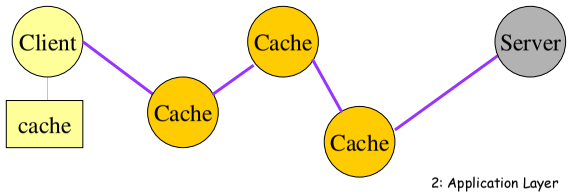
- Different metrics: cost of fetching again the resource, cost of storing the resource (size), number of access to the resource in the past, probability of accessing the resource in the near future, time since the last modification of the resource, expiration time
- Algorithms used to decide which cache 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

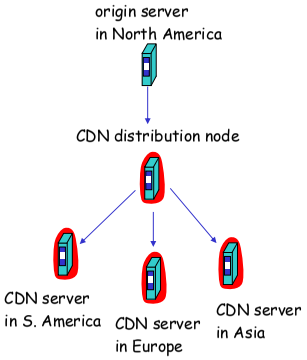
- Caching has some problems
  - Impossibility to count the access to servers (requested for advertisement)
  - possible staleness of content with weak consistency
- Servers have performed
  - cache busting: any technique preventing a cacheable resource from being cached at prowsers or broxies
  - Es. 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)

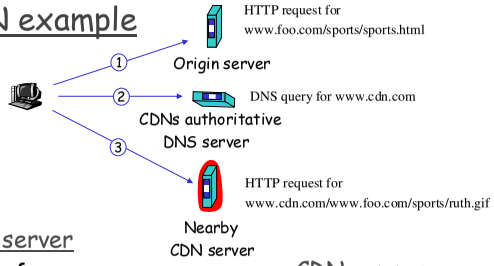
- The content providers are the CDN customers.

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



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

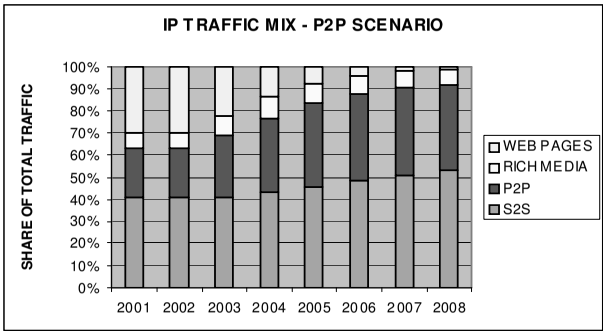
- CDN creates a "map", indicating distances from leaf ISPs and CDN nodes
- when query arrives at authoritative DNS server:
  - server determines ISP from which query originates
  - uses "map" to determine best CDN server

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)

Traffic share - projections



P2P file sharing

Example

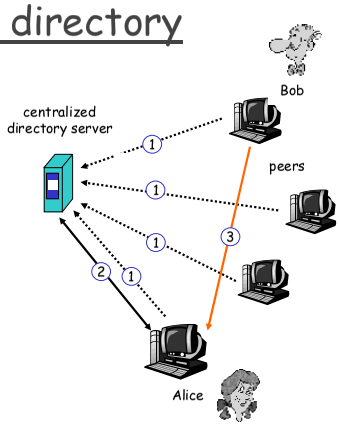
- Alice runs P2P client application on her notebook computer
- Intermittently connects to Internet; gets new IP address for each connection
- Asks for "Hey Jude"
- Application displays other peers that have copy of Hey Jude.
- Alice chooses one of the peers, Bob.
- File is copied from Bob's PC to Alice's notebook: HTTP
- While Alice downloads, other users uploading from Alice.
- Alice's peer is both a Web client and a transient Web server.

All peers are servers = highly scalable!

P2P: centralized directory

original "Napster" design

1) when peer connects, it informs central server:
   - IP address
   - content
2) Alice queries for "Hey Jude"
3) Alice requests file from Bob



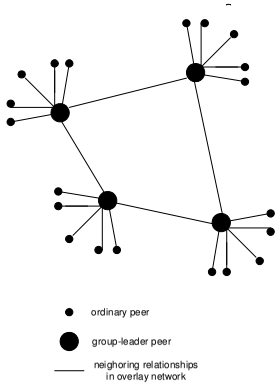
P2P: problems with centralized directory

- Single point of failure
- Performance bottleneck
- Copyright infringement

file transfer is decentralized, but locating content is highly centralized

P2P: decentralized directory

- Each peer is either a group leader or assigned to a group leader.
- Group leader tracks the content in all its children.
- Peer queries group leader; group leader may query other group leaders.



More about decentralized directory

overlay network
- peers are nodes
- edges between peers and their group leaders
- edges between some pairs of group leaders
- virtual neighbors

bootstrap node
- connecting peer is either assigned to a group leader or designated as leader

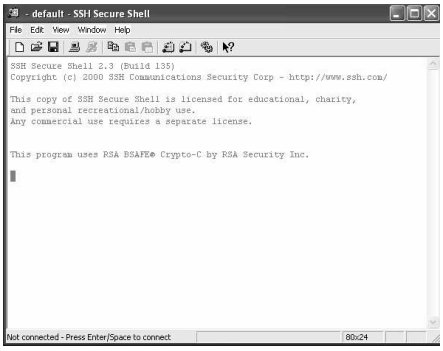
advantages of approach
- no centralized directory server
  - location service distributed over peers
  - more difficult to shut down

disadvantages of approach
- bootstrap node needed
- group leaders can get overloaded

Chapter 2 outline

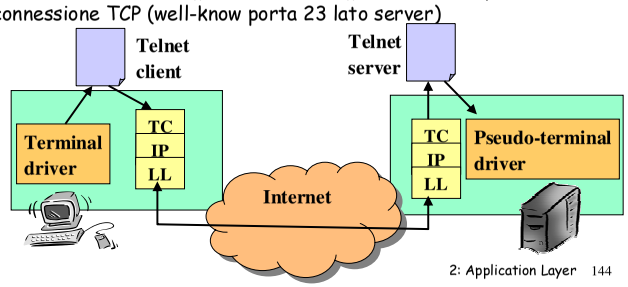
- 2.1 Principles of app layer protocols
  - clients and servers
  - app requirements
- 2.2 Web and HTTP
- 2.3 FTP
- 2.4 Electronic Mail
  - SMTP, POP3, IMAP
- 2.5 DNS
- 2.9 Content distribution
  - Network Web caching
  - Content distribution networks
  - P2P file sharing
- Telnet

TELNET (TErminaL NETwork)



TELNET (TErminaL NETwork)

- E' un semplice applicativo che consente ad un programma su un host (TELNET client) di accedere alle risorse su un host remoto (TELNET server) come se il client fosse un terminale (testuale) attaccato all'host remoto
- Un applicativo TELNET agisce da client o server a seconda che contatti o sia contattato
- Al contrario di un terminale locale i comandi sono trasferiti su una connessione TCP (well-know porta 23 lato server)



TELNET: Basic Ideas

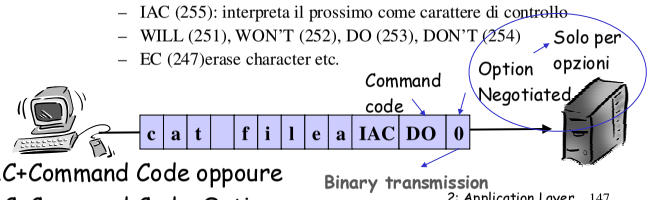
- Network Virtual Terminal
  - Imaginary device having a basic structure common to a wide range of real terminals→guarantees the colloquium follows clear rules and formats independent of the real terminal features
    - Data represented in 7 bit ASCII transmitted in 8-bit bytes
    - NVT half-duplex device operating in line-buffered mode (the characters of a line are buffered and a complete line is transmitted together)
    - Local echo provided
  - NTV made of: a printer (or display) and a keyboard. The keyboard produces the data transmitted over the TCP connection; the printer receives and displays the data.
- Negotiation of terminal options
  - Two hosts can negotiate different options to extend the NVT capabilities to reflect the capabilities of the real HW in use (at the beginning of the colloquium or dynamically during it)
- Symmetric view of terminals and processes
    - Being a client or server may only depend on who is contacing whom

Terminal options

- Terminal options: kind of transmission (e.g. binary), window size, terminal type, lien width, etc.
- How to negotiate them?
  - Usage of DO/DON't/WILL/WON'T followed by option code
  - Es (CLIENT). DO Transmit binary (requests that the other party use binary transmission)
    Reply: WILL Transmit binary (OK),
    WON'T Transmit binary (NO)
    Es2 (CLIENT) WILL Transmit binary (desire to use)
    Reply: DO Transmit binary(OK, I expect you to do so)

TELNET (TErminaL NETwork)

- TELNET trasferisce caratteri
  - Caratteri dati:
    - Sono caratteri ASCII con il primo bit pari a 0
    - Si possono trasferire anche caratteri ASCII con il primo bit pari a 1 se li si fa precedere da un byte di controllo speciale
  - Caratteri di controllo:
    - Sono comandi codificati in sequenze di 8 bit con il primo pari ad 1
    - Tra questi
      - IAC (255): interpreta il prossimo come carattere di controllo
      - WILL (251), WON'T (252), DO (253), DON'T (254)
      - EC (247)erase character etc.



IAC+Command Code oppoure
IAC+Command Code+Option

Come leggere uno standard

- Lo standard RFC di TELNET disponibile sul sito (in blu descrizione protocollo quindi vari livelli di dettaglio crescente: arancione, fucsia, nero)
- Vari livelli di lettura: primo pass per capire il funzionamento (negli standard anche molte info di dettaglio sin dall'inizio che non potete capire/che non vi interessano). NON come un libro: cio' che non capite potrebbe essere spiegato chiaramente pagien dopo. Piu' pass per una comprensione totale.
- Distinguere le info necessarie per una prima comprensione da info di dettaglio che fanno esempi/specificano meglio alcuni aspetti da info di estremo dettaglio (implementazioni in alcuni sistemi, formati etc.) che potrebbero non interessarvi affatto!!
- Dopo aver compreso con una lettura delle parti importanti il funzionamento tornate sugli argomenti di dettaglio e li rileggete con piu' attenzione se e solo se vi serve quel livello di dettaglio (un unico standard risponde a diverse esigenze: studente, implementatore, etc.!!)