DIPARTIMENTO
DI INFORMATICA
Via Salaria 113 - 00198, Roma

SAPIENZA
UNIVERSITÀ DI ROMA

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* , 7th edition.

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

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

My Links - Netscape

File Edit View Go Communicator Help

Back Forward Reload Home Search Netscape Print Security Shop Stop

Bookmarks Location: http://www.elet.polimi.it/Users/DEI/Sections/Telecom/Antonio.Capone/mylinks.html What's Related

Instant Message WebMail Radio People Yellow Pages Download Calendar Channels RealPlayer

My Links

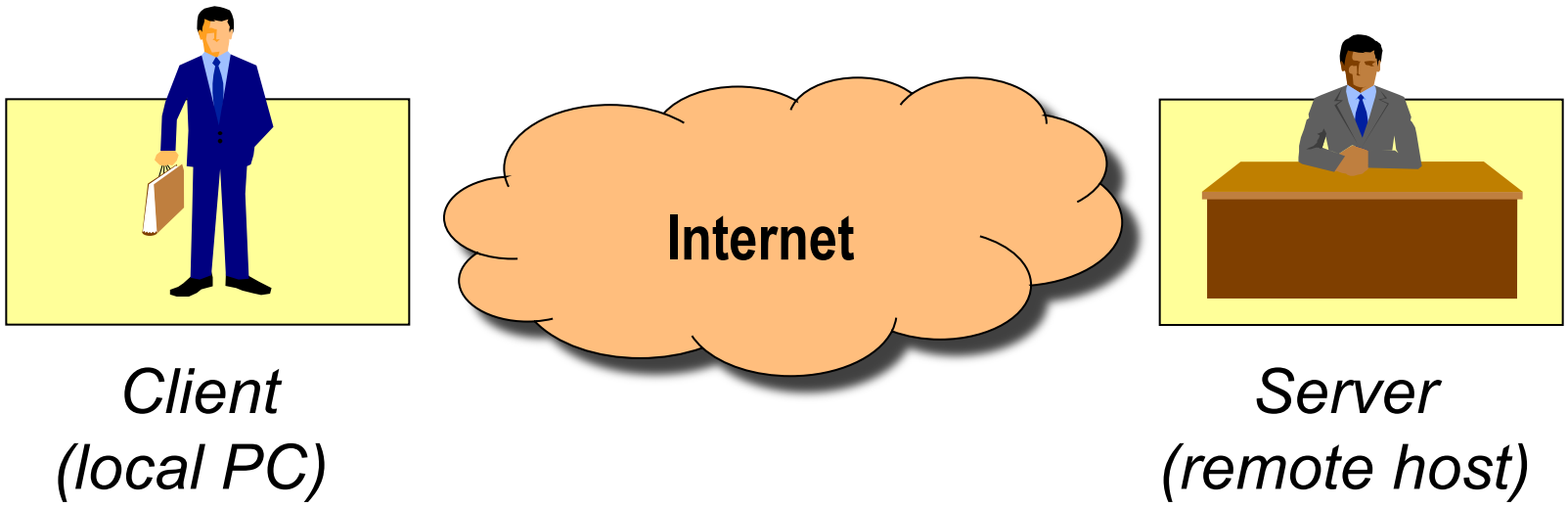
My Home Page
My mail box

- Search Engines
- Scientific and Technical Papers search engines
- Conferences, Journals and Technical societies
  - Conferences
  - Journals and Societies
  - On-line magazines
- Research Centers and Organizations
  - Italian Universities
  - Communication Networks
  - Wireless
  - IP

- Research Topics
  - Wireless ATM
  - UMTS
  - IP
  - Simulation
- Telecom Companies & info sources
- Education Centers
- Telecommunication Networks Documents
- Software resources
- Public institutions
- Art and Culture
- News Papers
- Travels
- Computer Stores
- Friends Home Pages
- Weather
- Italian Internet Access Points

You are offline. Choose "Go Online..." to connect

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. personal web pages:

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

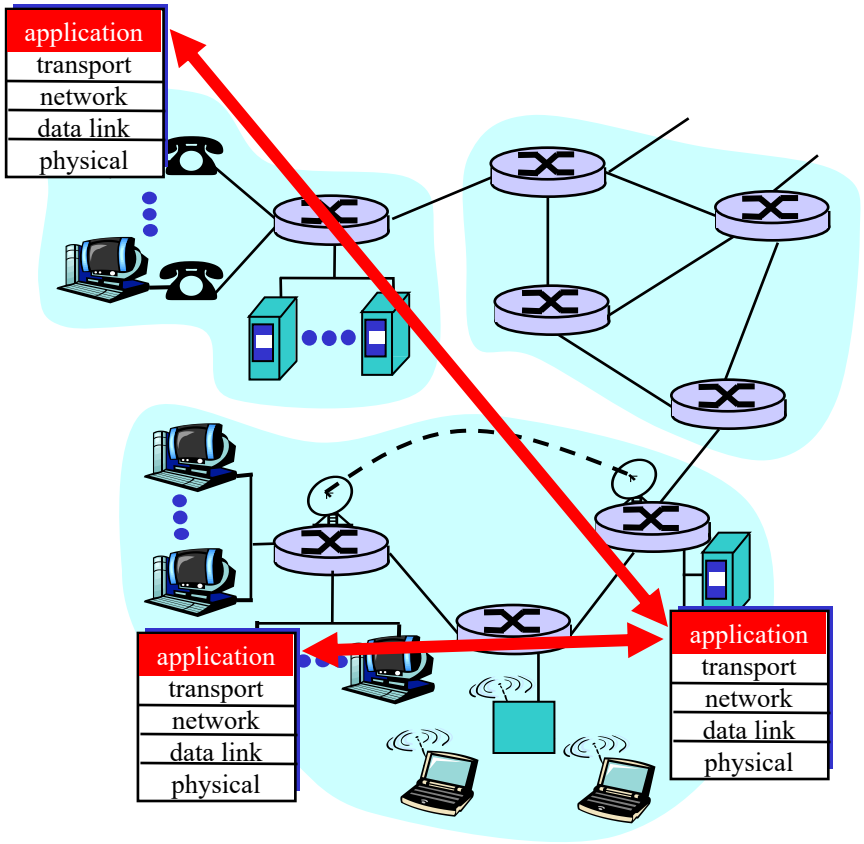
Applications and application-layer protocols

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

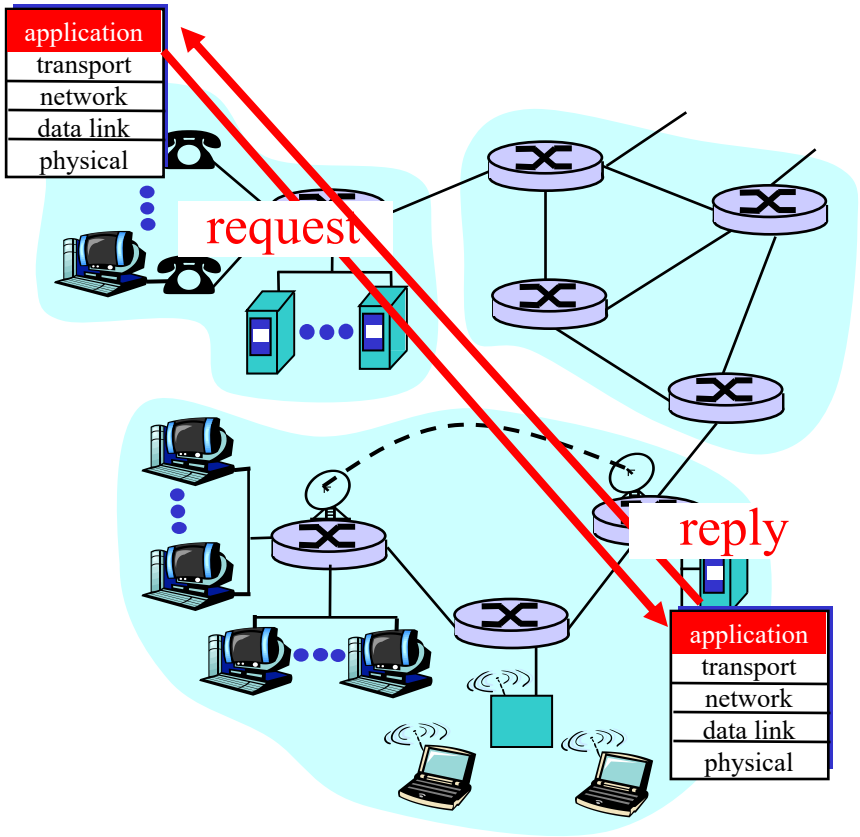
Proprietary protocols:

- eg, KaZaA

Client-server paradigm

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

(NEXT SLIDE)



Client:

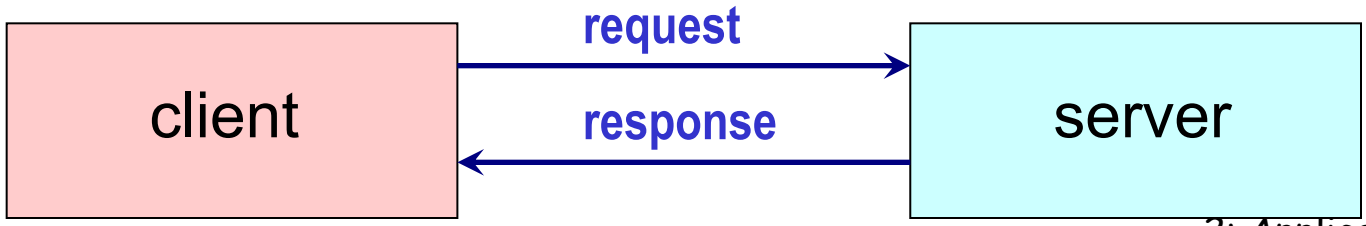
- initiates contact with server ("speaks first")
- typically requests service from server,
- Web: client implemented in browser; e-mail: in mail reader

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



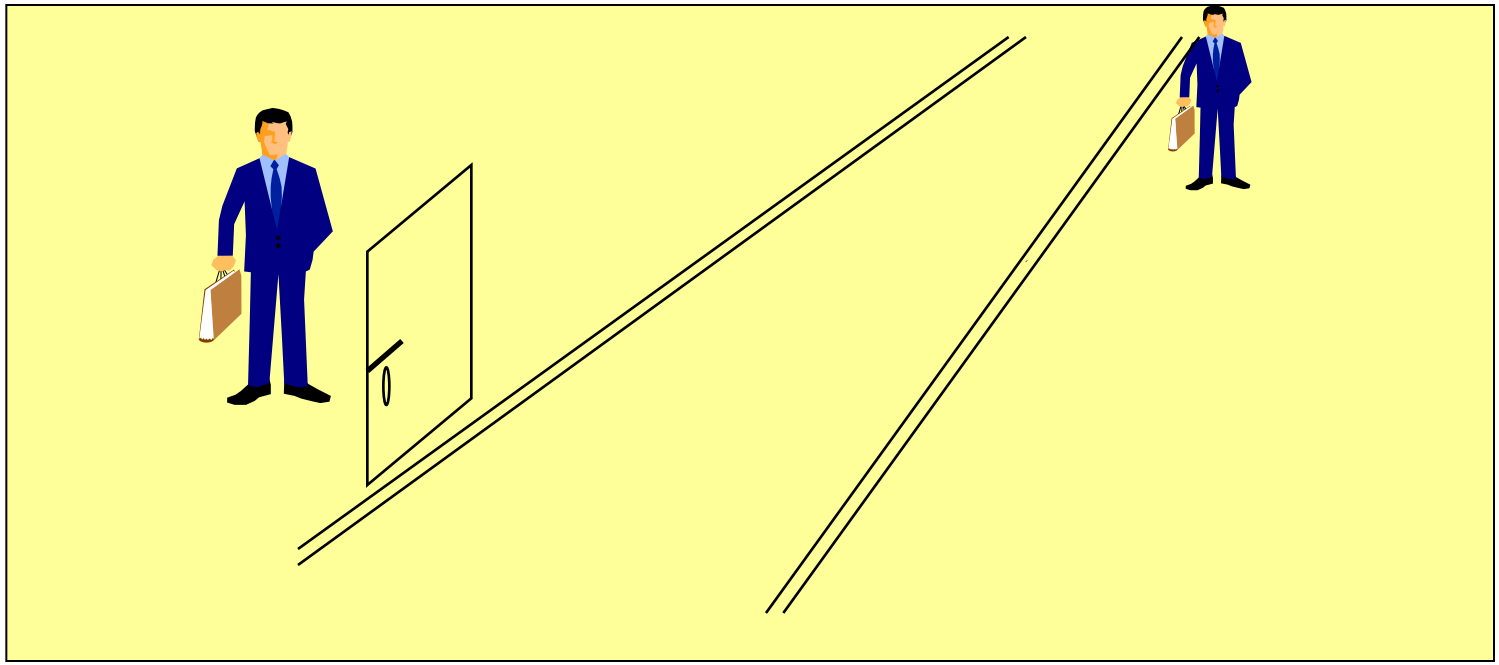
Programmi Client e Server

- 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
- Applicazione Web: l'active open del client richiede l'indicazione dell'indirizzo e della porta del client, ed inizia la connessione TCP per mettere in comunicazione il client e il server

Addressing processes:

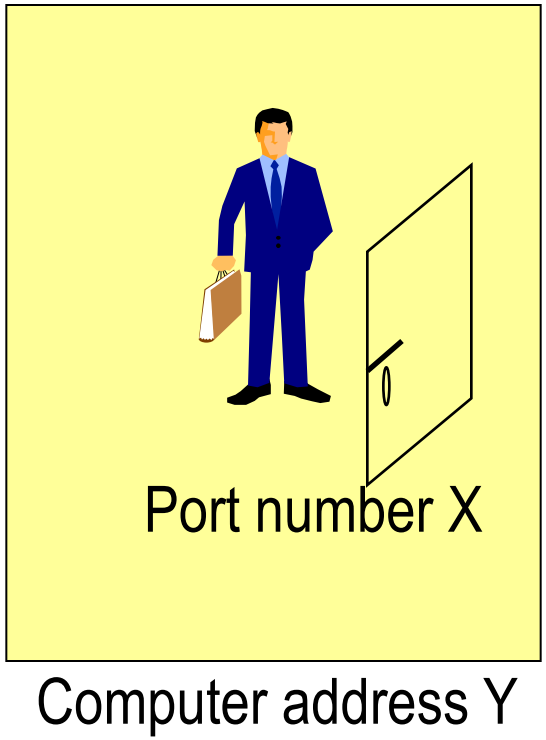
- For a process to receive messages, it must have an identifier
- Every host has a unique 32-bit IP address
- Q: does the IP address of the host on which the process runs suffice for identifying the process?
- Answer: No, many processes can be running on same host
- Identifier includes both the IP address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

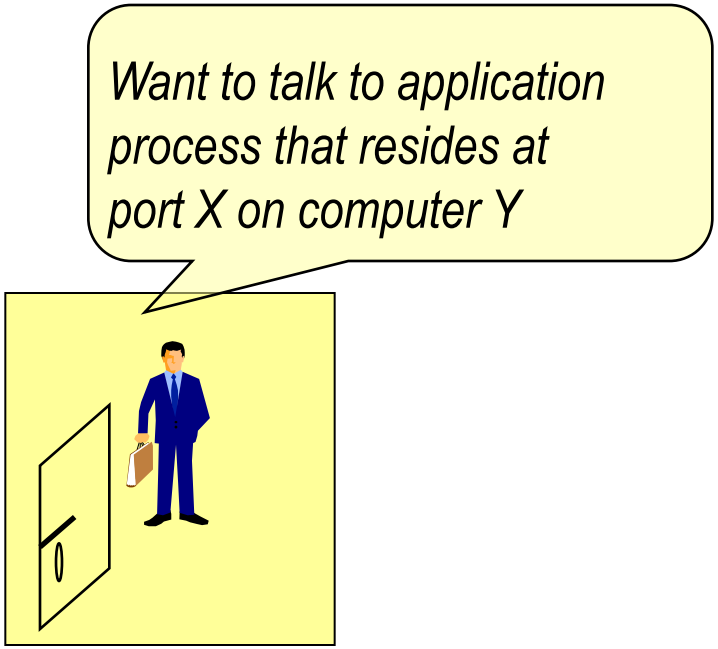
Port numbers



the “address” of the SW process inside the computer!

Same addressing scheme works for different machines!!

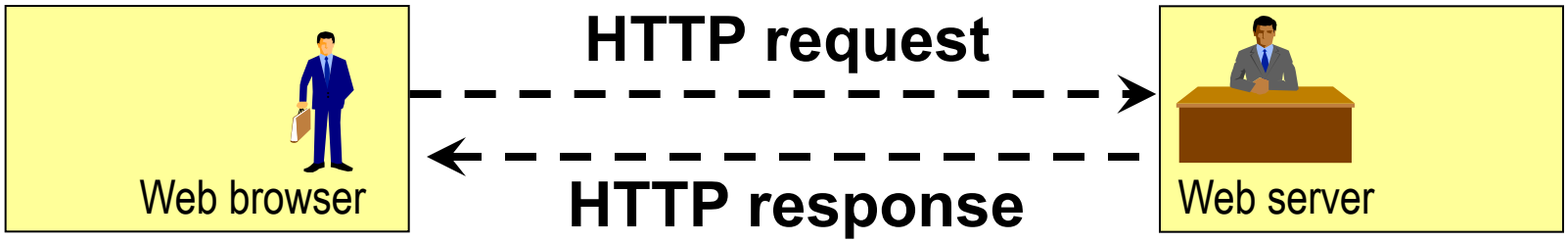




Re-understanding URLs

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

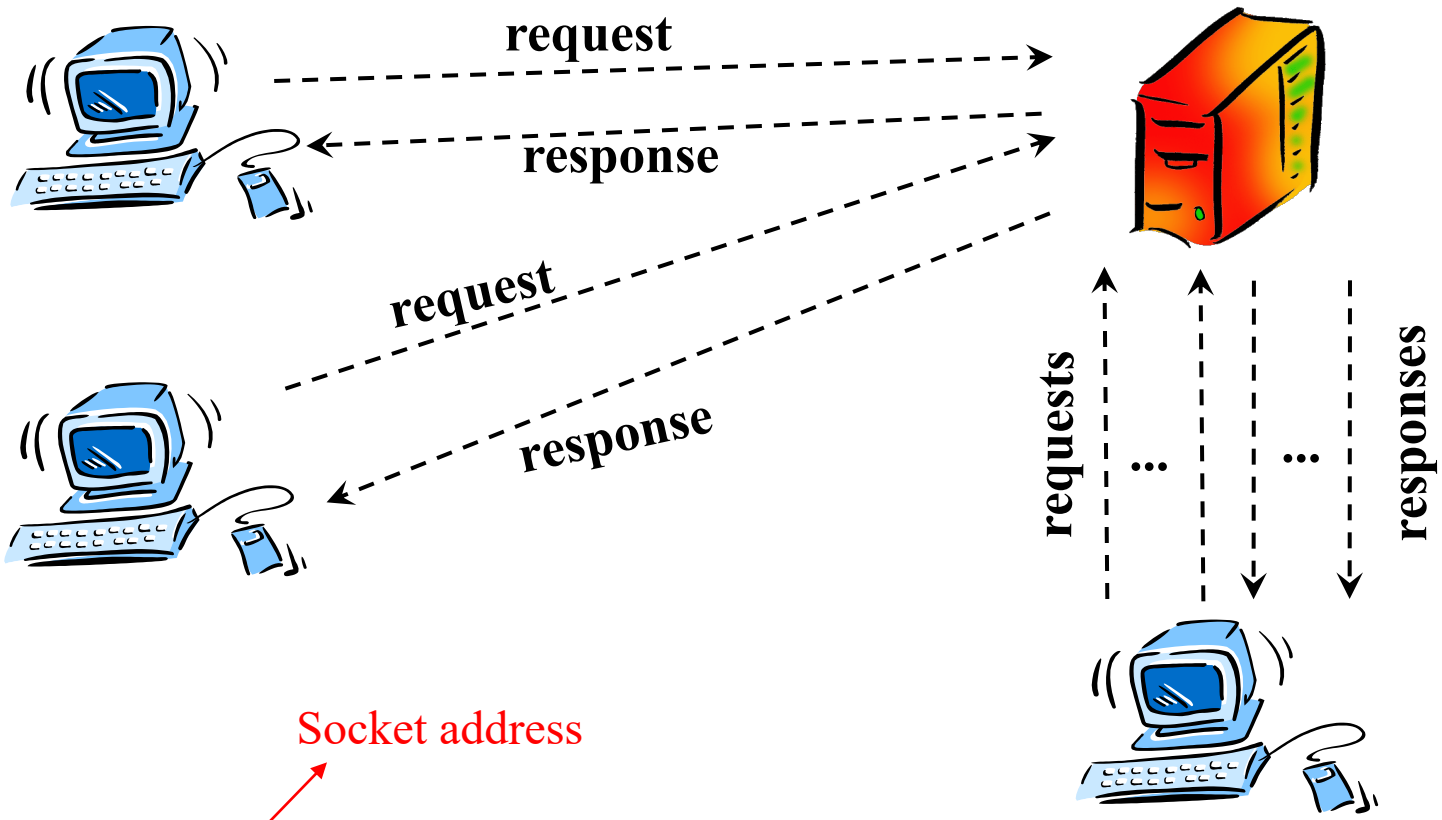
HTTP://cerbero.elet.polimi.it:8080/… (if exists!)

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

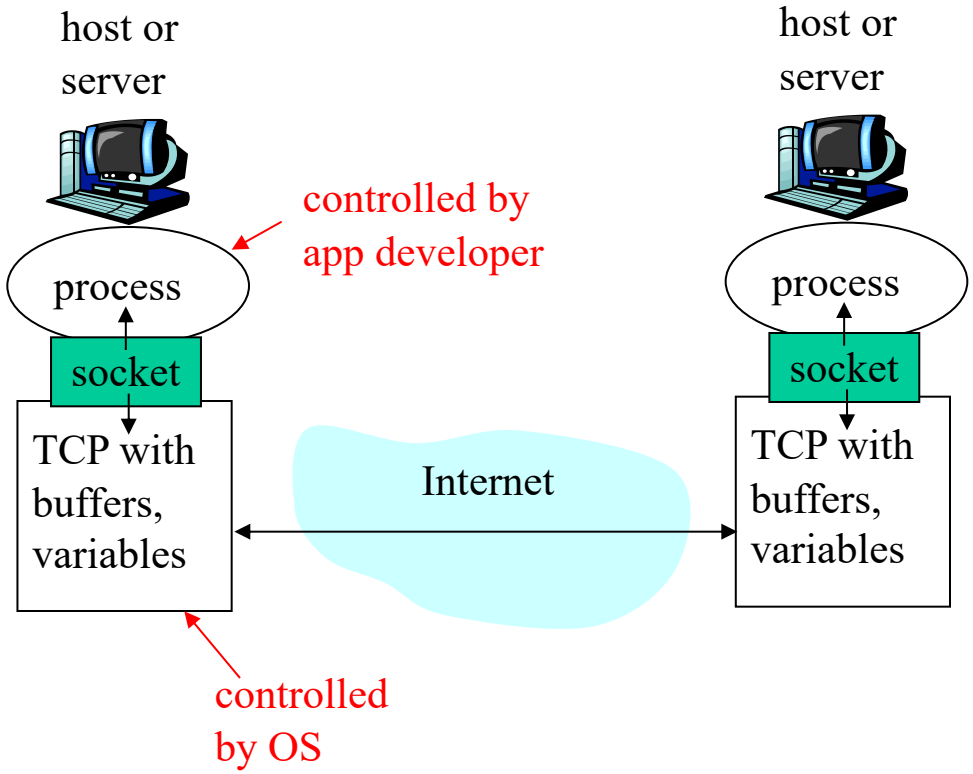


Socket address

- 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 un well known port #
  - OS assegna un numero di porta disponibile

Processes communicating across network

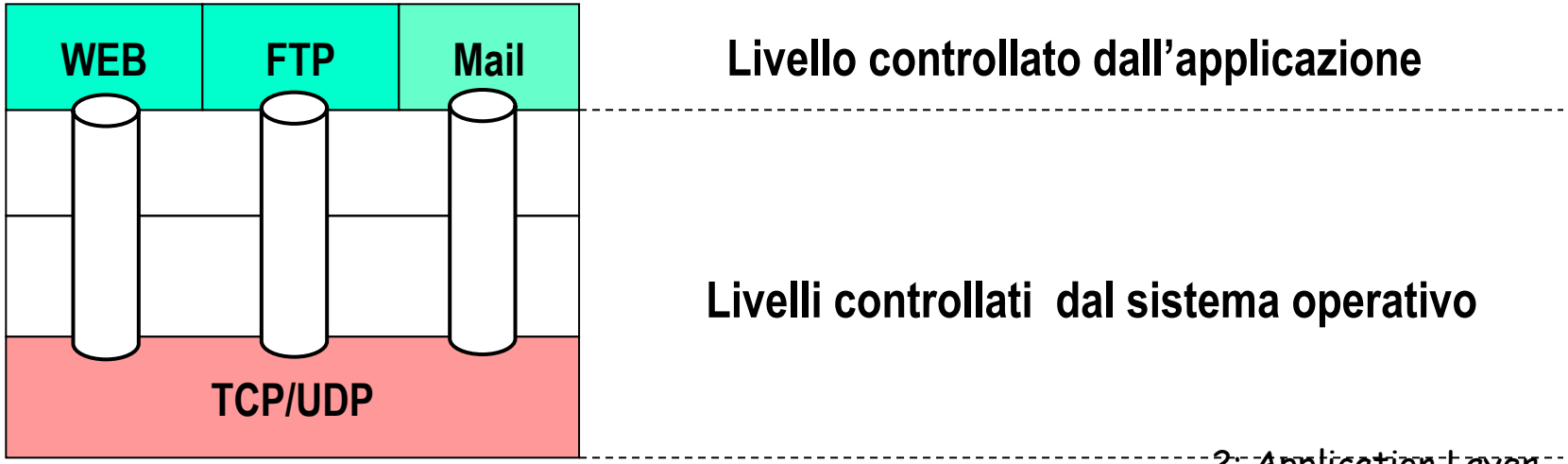
- process sends/receives messages to/from its socket
- socket analogous to door
  - sending process shoves message out door
  - sending process assumes transport infrastructure on other side of door which brings message to socket at receiving process



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

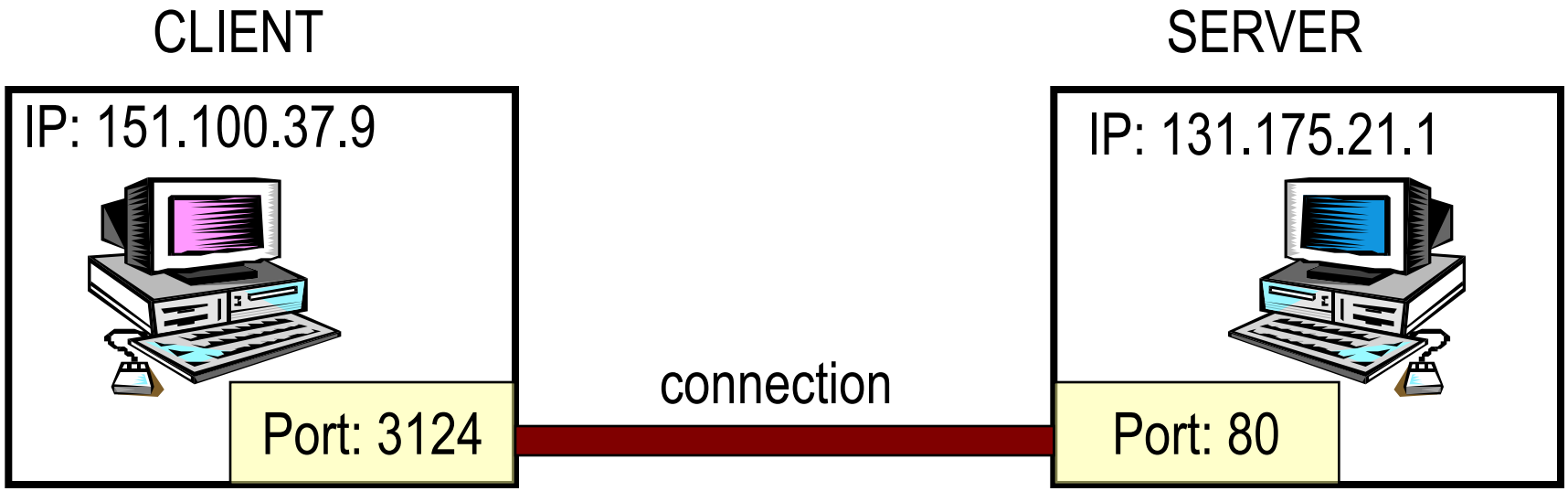
Interazione coi livelli inferiori: architettura TCP/IP

- 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).
- Qualsiasi protocollo di livello applicativo accede al servizio di traporto mediante socket.
- 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
- Come si distingue a quale protocollo di trasporto passare il segmento? Protocol number



Connections

identified by socket addresses at its ends



CLIENT SOCKET: [<151.100.37.9,**3124**>]

SERVER SOCKET: [<131.175.21.1,**80**>]

+

TCP segment

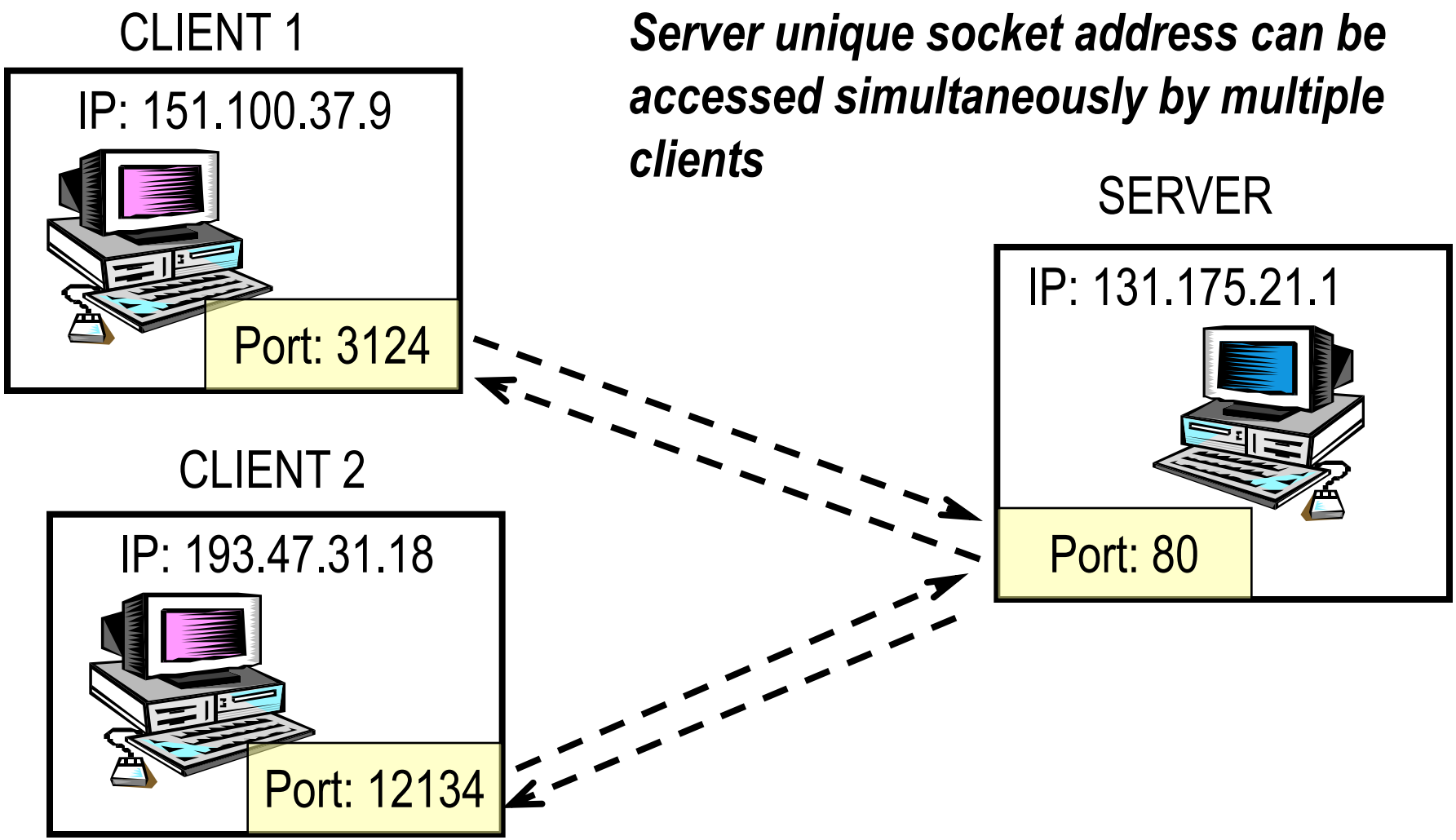
src port = 3124
dst port = 80
…..

src port = 80
dst port = 3124
…..

TCP segment

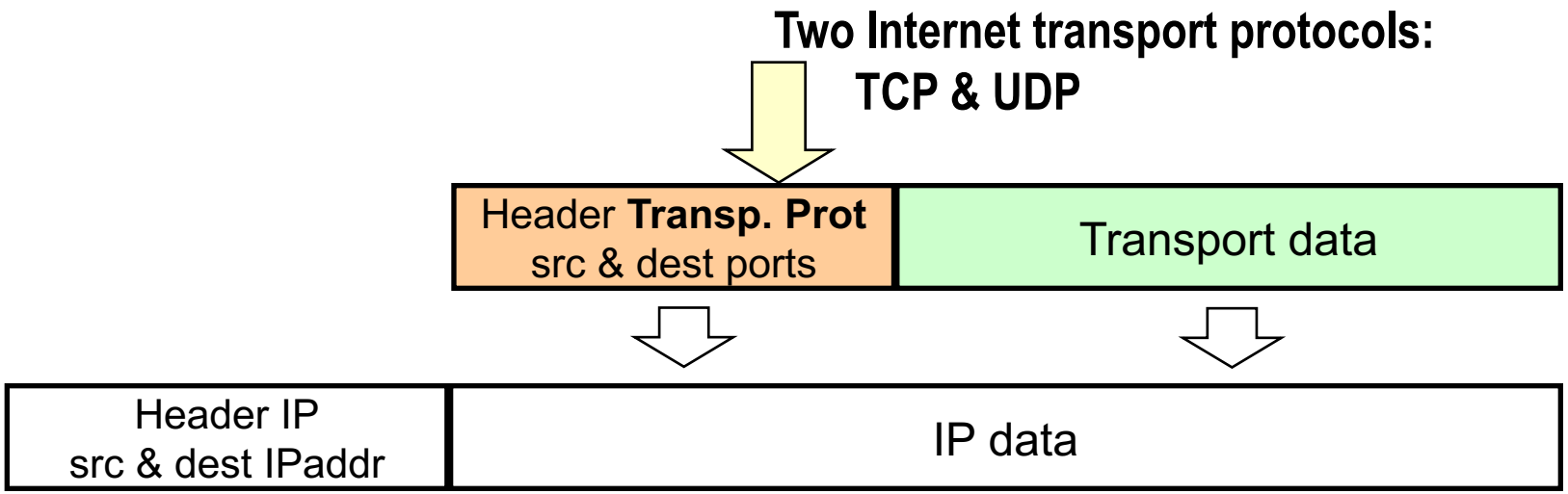
Managing multiple connections

Server unique socket address can be accessed simultaneously by multiple clients



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;

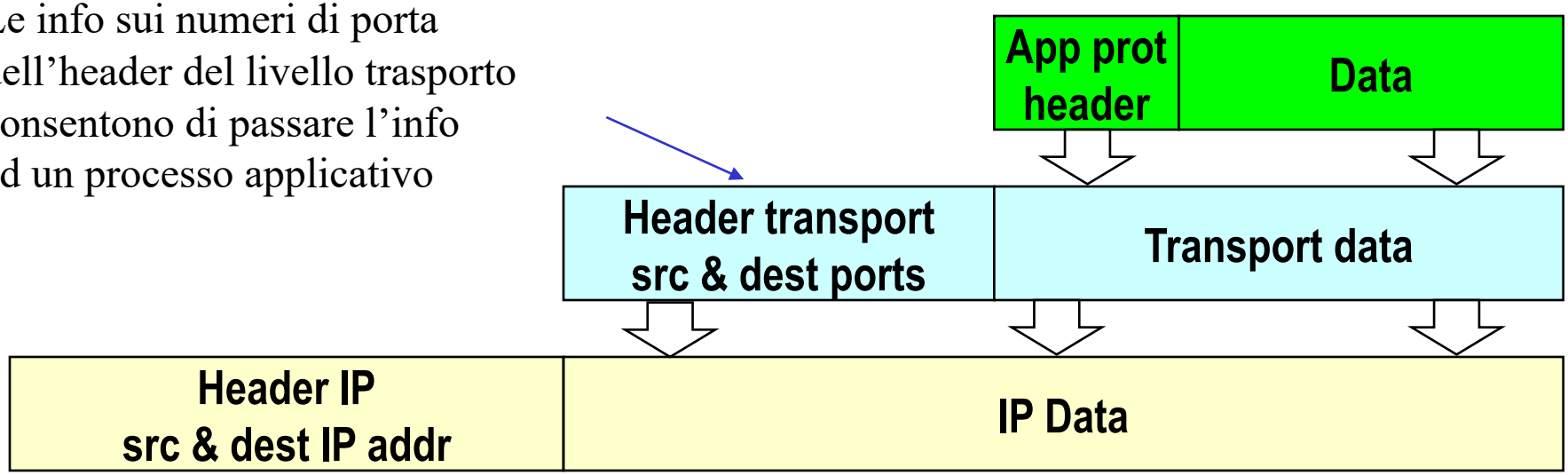
Once they reach destination, they are delivered to app based on port #

Sockets and Internet transport protocols

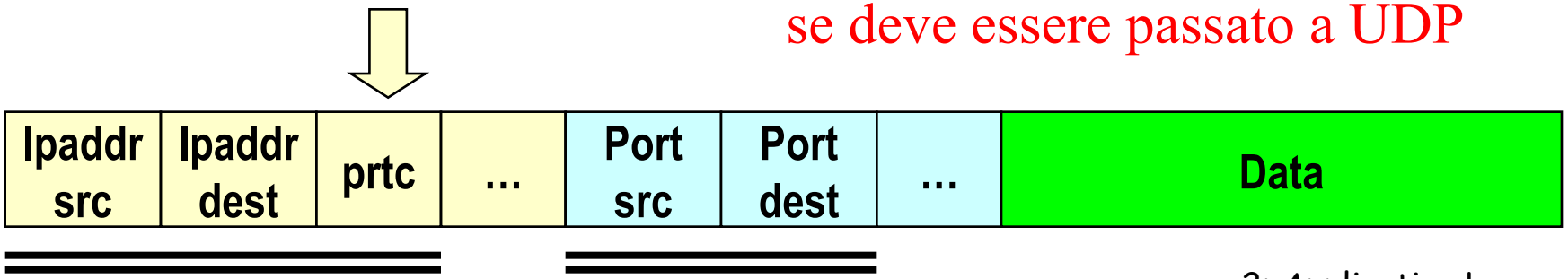
- Two transport protocols in Internet:
  - TCP (reliable, connection-oriented)
  - UDP (unreliable, connectionless)
- When opening socket, needs to specify which transport to use!

Socket addresses (refined)

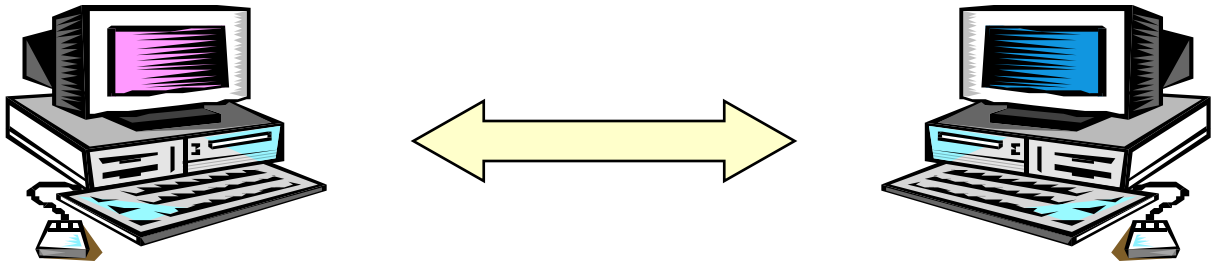
Le info sui numeri di porta dell'header del livello trasporto consentono di passare l'info ad un processo applicativo



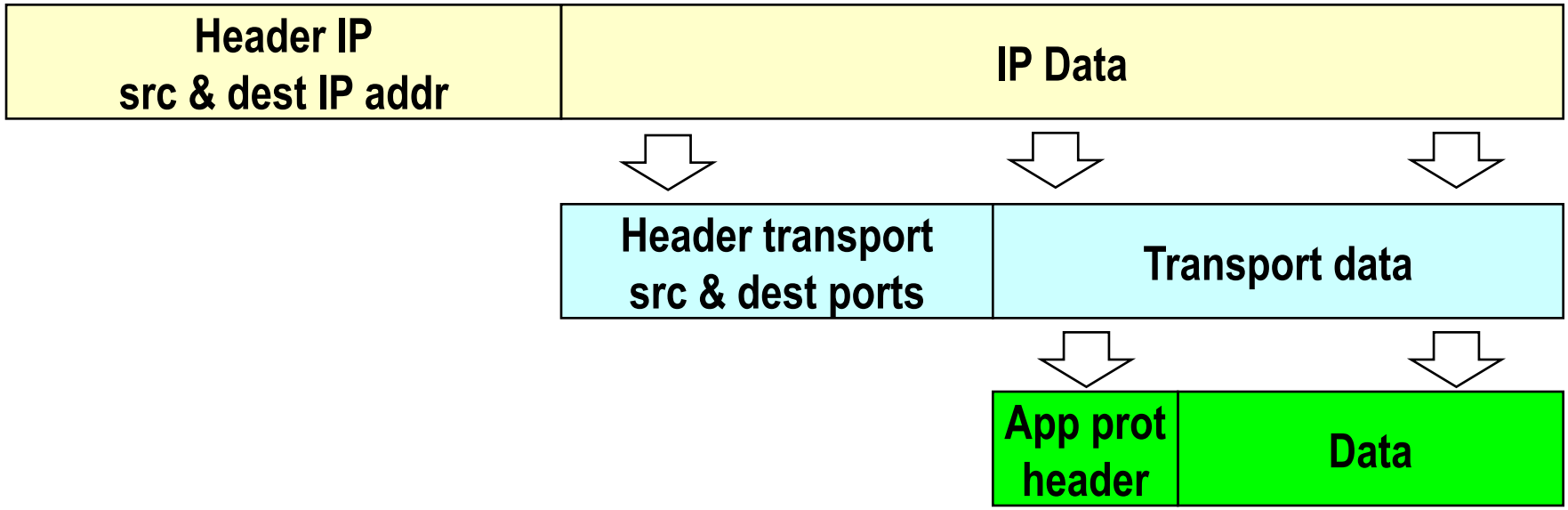
Nell'header IP il campo Protocol vale 6 se il pacchetto deve essere passato a TCP, 17 se deve essere passato a UDP



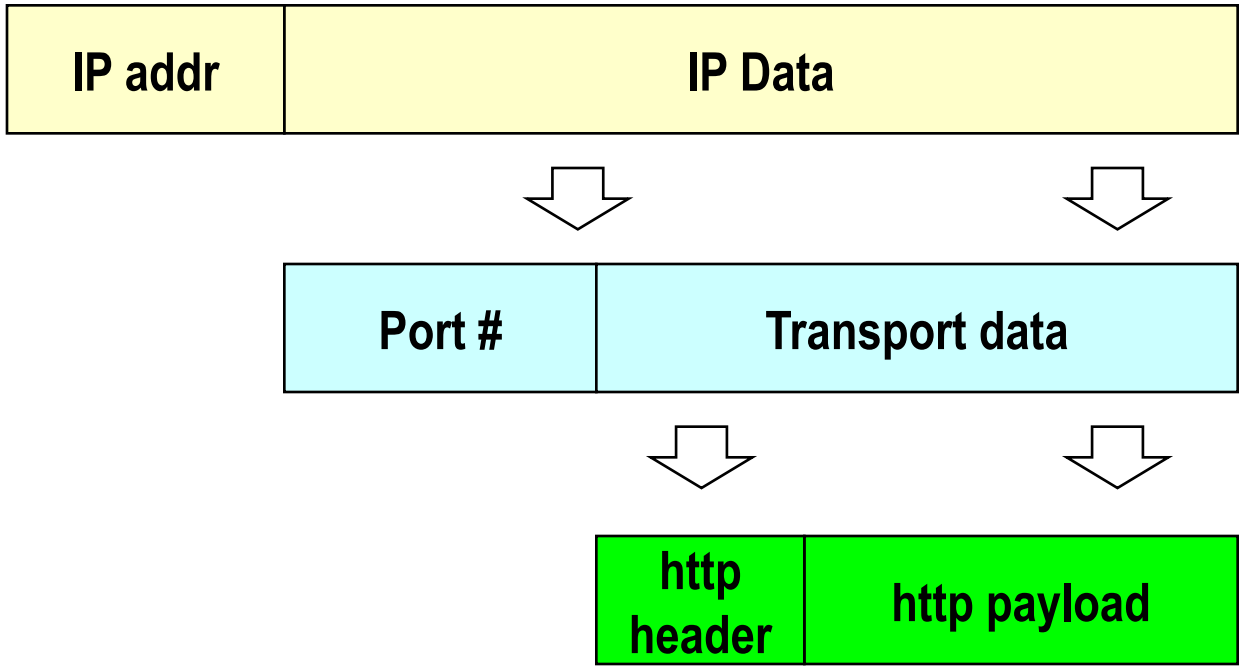
The Internet level view



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

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?

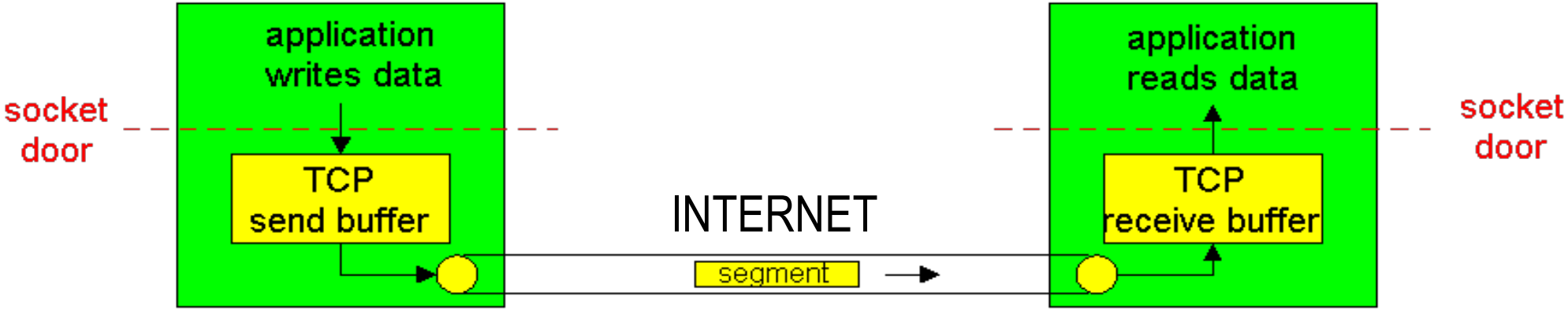
Multiplexing/Demultiplexing

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?



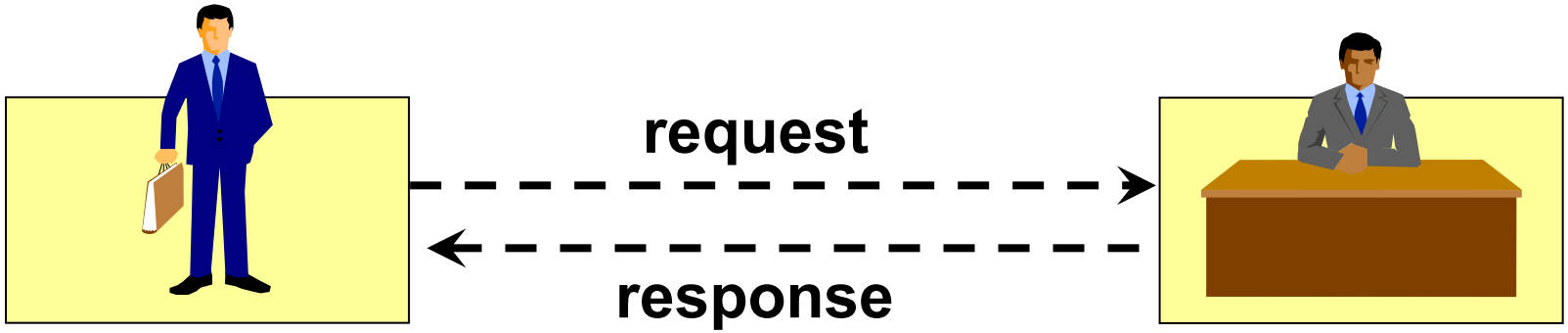
- Application software duties:
  - open socket (e.g. C, C++, JAVA function call, OS call, external library primitive)
  - Injects message in its own socket
  - being confident message is received on the other side
- TCP software: in charge of managing segments!
  - reliable message transport when TCP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

1. Web Application: HTTP 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.)

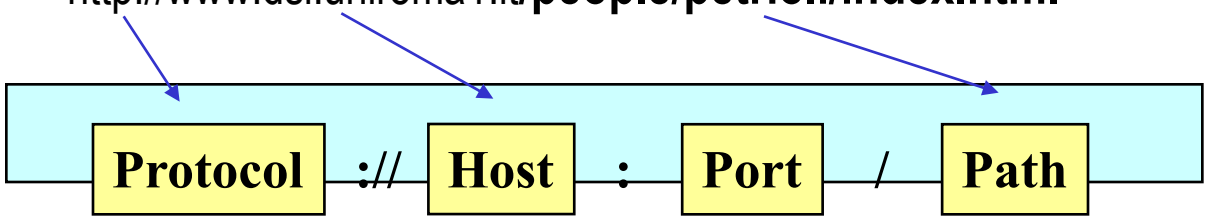
- HTTP: Request-response protocol. A request message is sent from the client to a recepient server. The recipient server sends a response message back.



- 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.

The three components of an URL

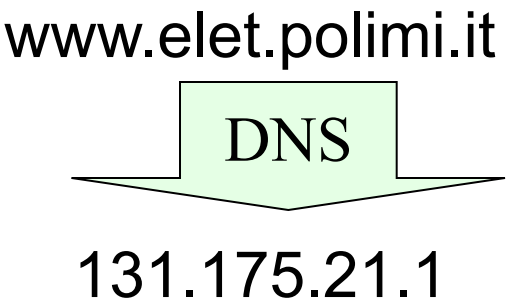
1. *Protocol (also called "scheme")*
   - *how can a page be accessed?* (application protocol used)
     - **http**://www.dsi.uniroma1.it/people/petrioli/index.html
2. *Host name*
   - *Where is the page located?* (symbolic or numeric location)
     - http://www.**dsi.uniroma1.it**/people/petrioli/index.html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



2. Location – host name

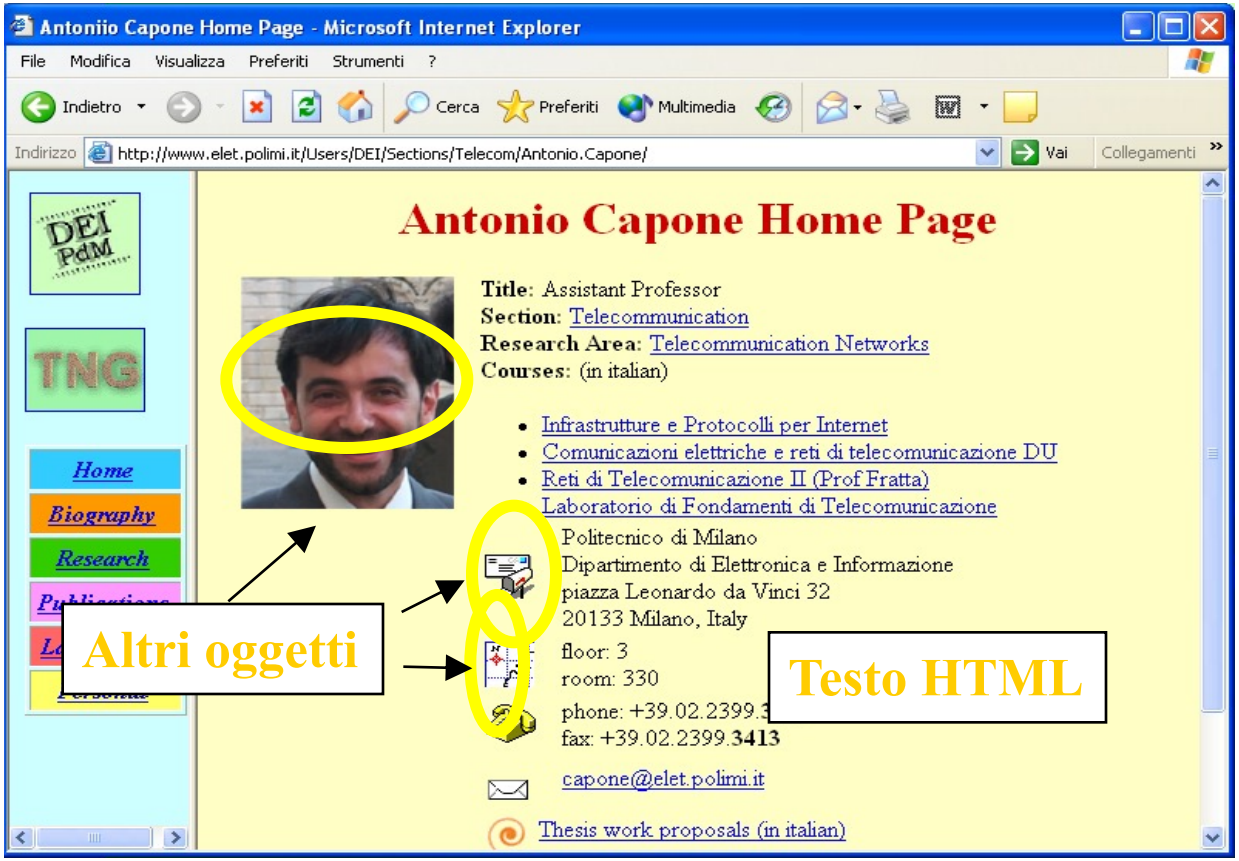
Specifies where is the page located:

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



Trasporto dei messaggi

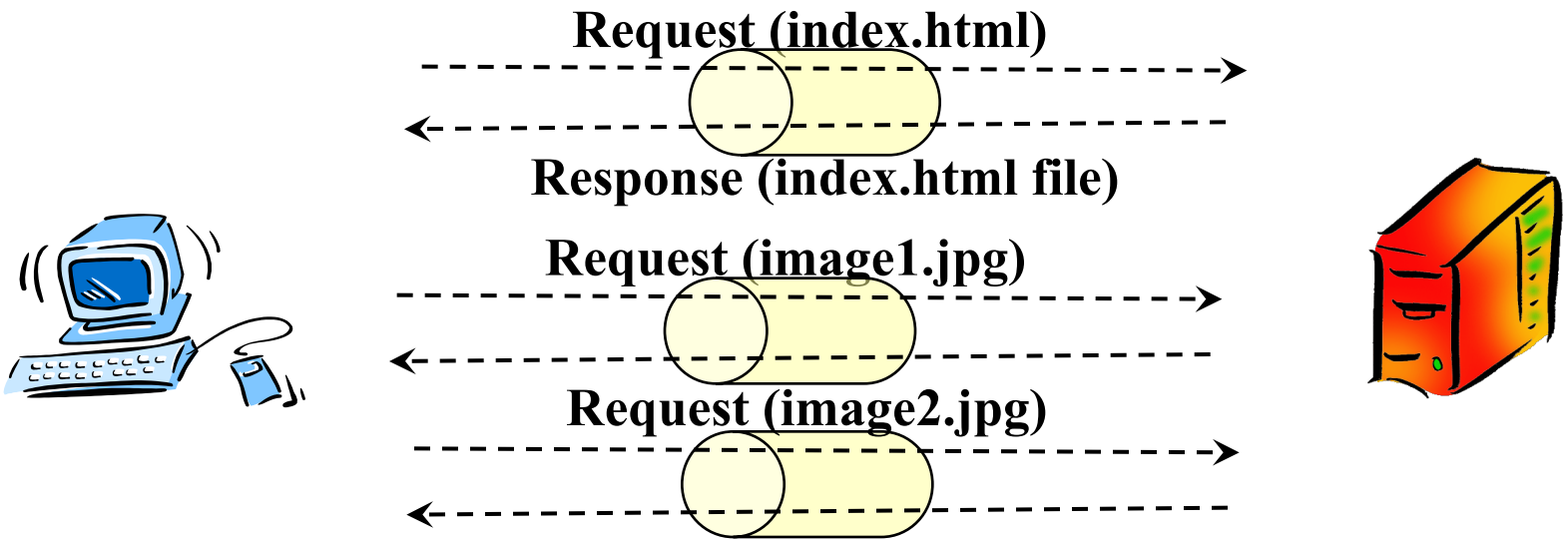
- Supponiamo che un client richieda una pagina HTML di un server al cui interno sono contenuti i riferimenti ad altri oggetti (ad esempio 10 figure che compongono la pagina e che occorre visualizzare insieme al testo HTML).



- **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



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

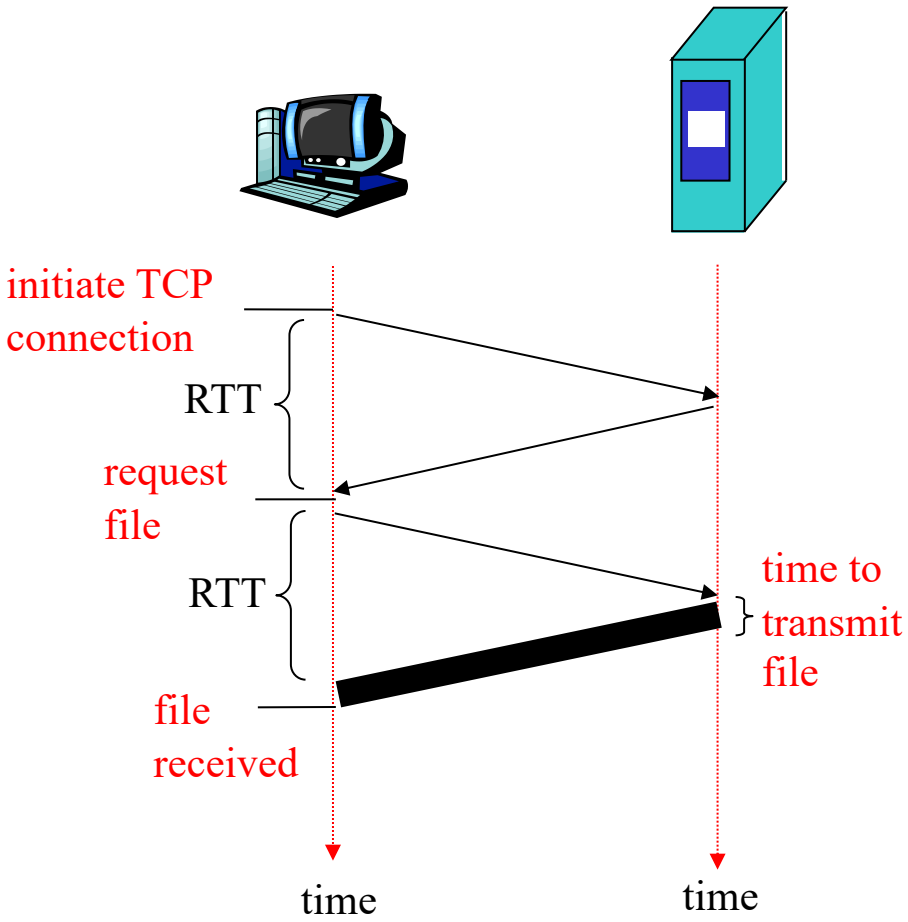
Response time modeling

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

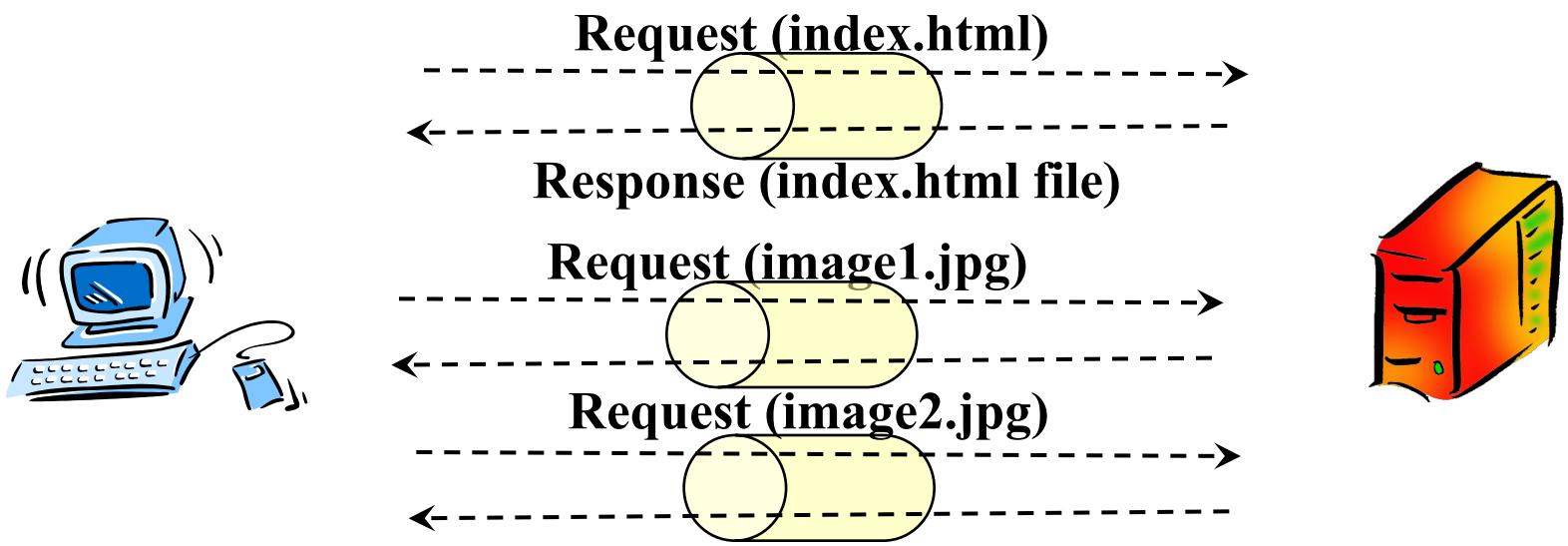
total = 2RTT+transmit time



Problema: OGNI VOLTA CHE SI APRE UNA CONNESSIONE SI PAGA 2RTT PRIMA DI SCAMBIARE DATI

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



Un esempio…

- Esempio: di richiesta oggetto

HTTP è testuale (ASCII)

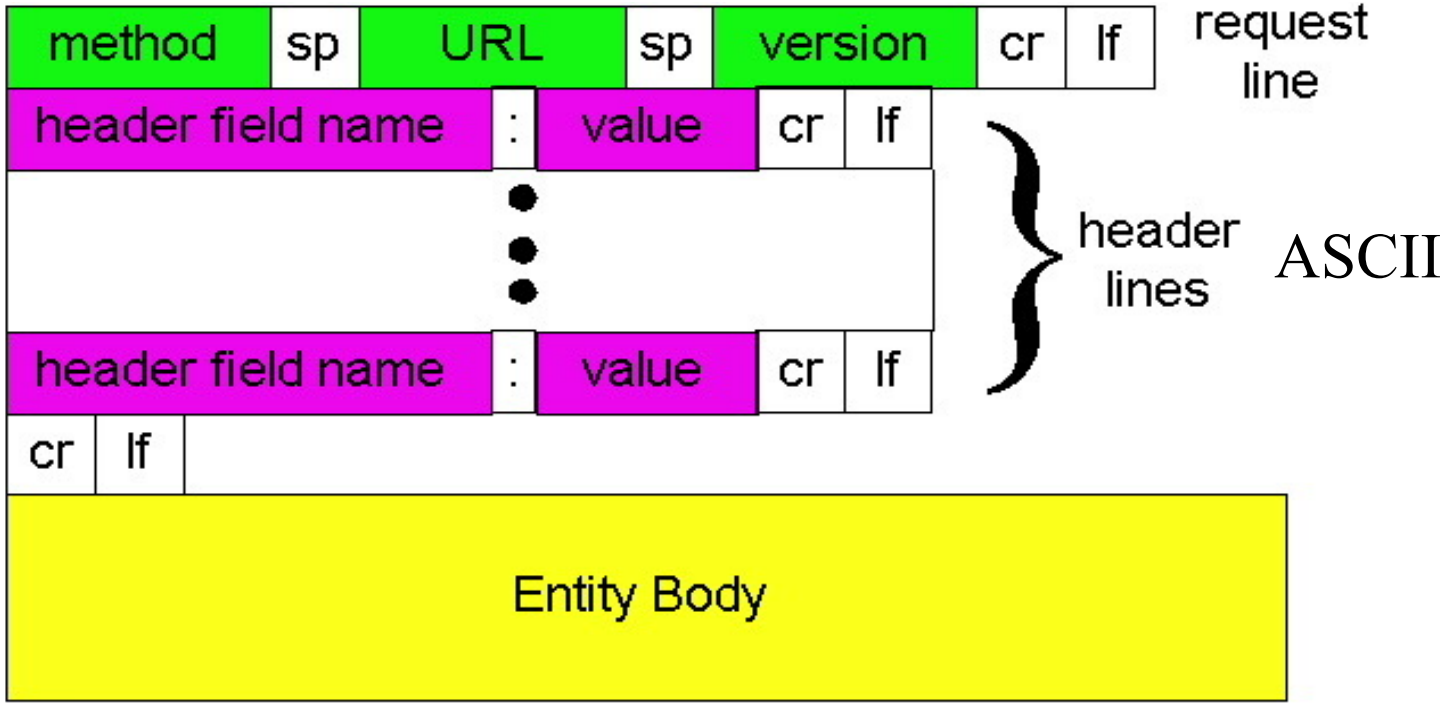
```
GET /ntw/index.html HTTP/1.0
Date: Wed, 22 Mar 2000 09:09:01 GMT
Pragma: No-cache
From: gorby@moskvax.com
User-agent: Mozilla/4.3
```

- 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 ...
```

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 /search.cgi?string=greek-architects HTTP/1.0

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 ...
```

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: gorby@moskvax.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

- 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
`Expires`	The entity should be considered stale after the time specified in this header (after this time the content could still be stored in cache but should not be returned to users without first validating with the origin server)
`Last-Modified`	Specify the time at which the resource was modified last

Un esempio…

- Esempio: di richiesta oggetto

HTTP è testuale (ASCII)

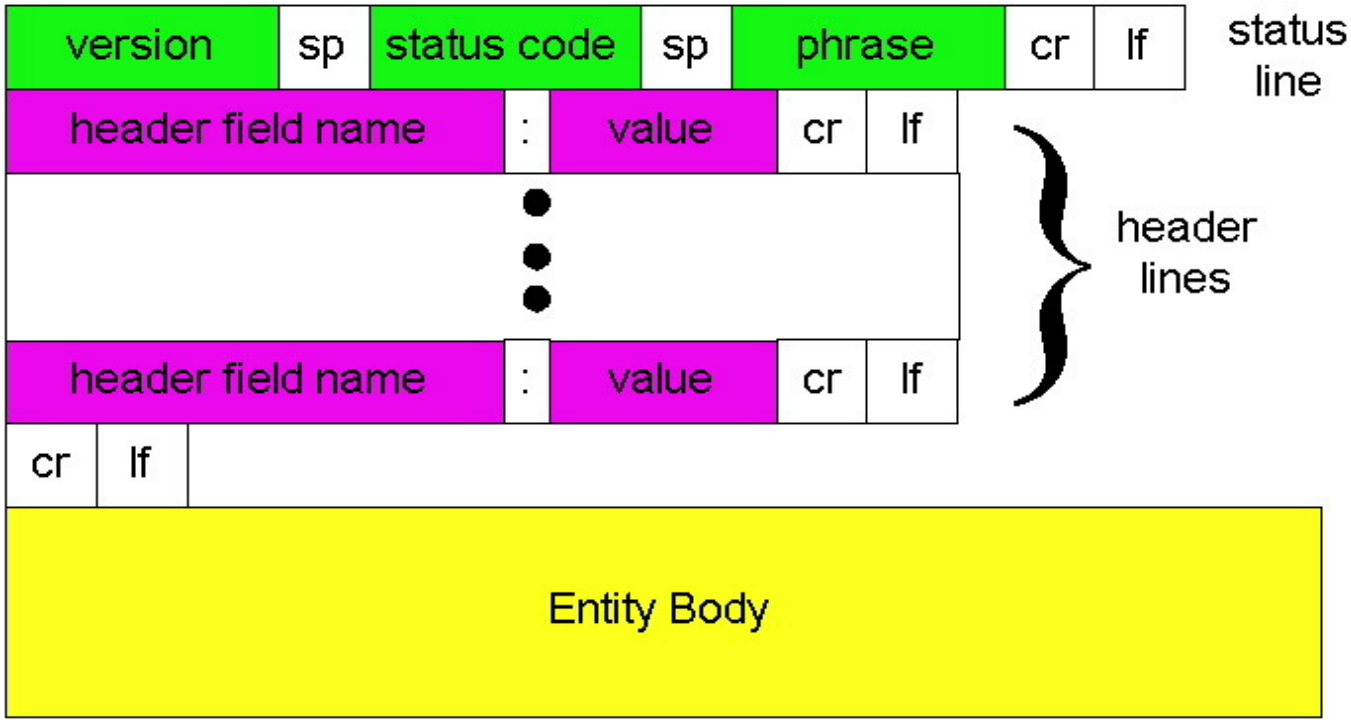
```
GET /ntw/index.html HTTP/1.0
Date: Wed, 22 Mar 2000 09:09:01 GMT
Pragma: No-cache
From: gorby@moskvax.com
User-agent: Mozilla/4.3
```

- 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 retransmit the request with appropriate credentials according to a given scheme

Messaggi

- 1xx Informational

Status codes:

- 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)
202 Accepted:	The request has been received but has not yet been handled in full (e.g., since a program must run which requires a long time→the user agent can continue with its task without waiting for the action to complete at the origin server)
204 No content:	Successfully recived, no change required in what the user actually sees

Messaggi

- **3xx Redirection**

301 Moved Permanently:	L'oggetto è stato spostato nell'URL indicato
302 Moved Temporarily:	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:	Autenticazione fallita (e.g., Accesso senza necessari account e password)
403 Forbidden:	La richiesta e' stata ricevuta e compresa ma il server ha stabilito di non servirla. Le ragioni possono essere indicate nell'entity body.
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

- 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 ...
```

HTTP è testuale (ASCII)

- 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:

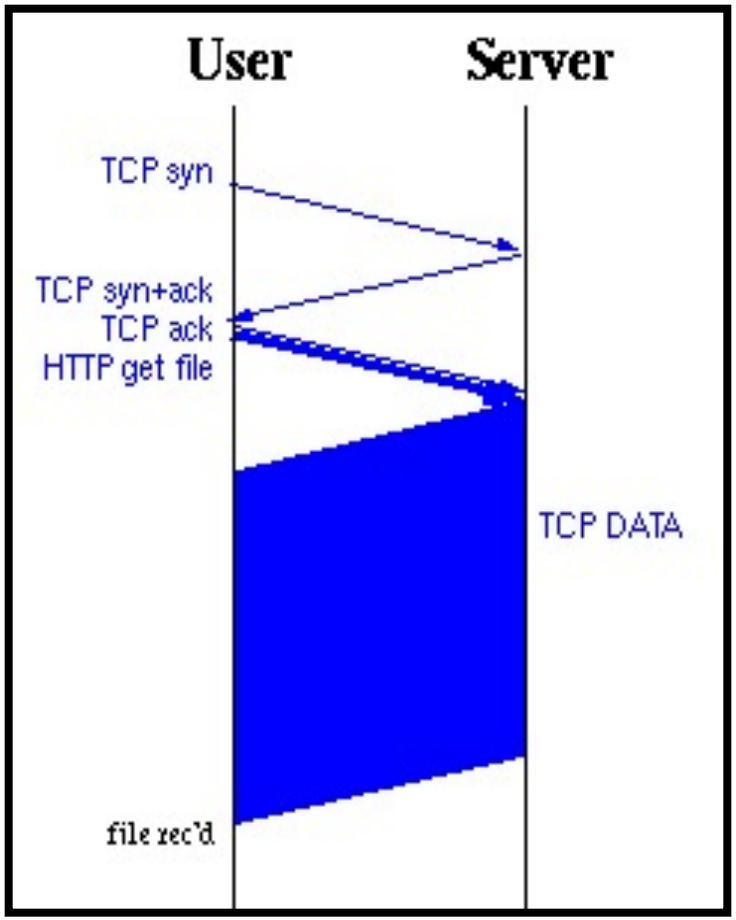
```
HTTP/1.0 304 Not Modified
Date: Wed, 19 Aug 1998 15:39:29
Server: Apache/1.3.0 (Unix)
```

(empty entity body)

- E' possibile anche usare il metodo HEAD

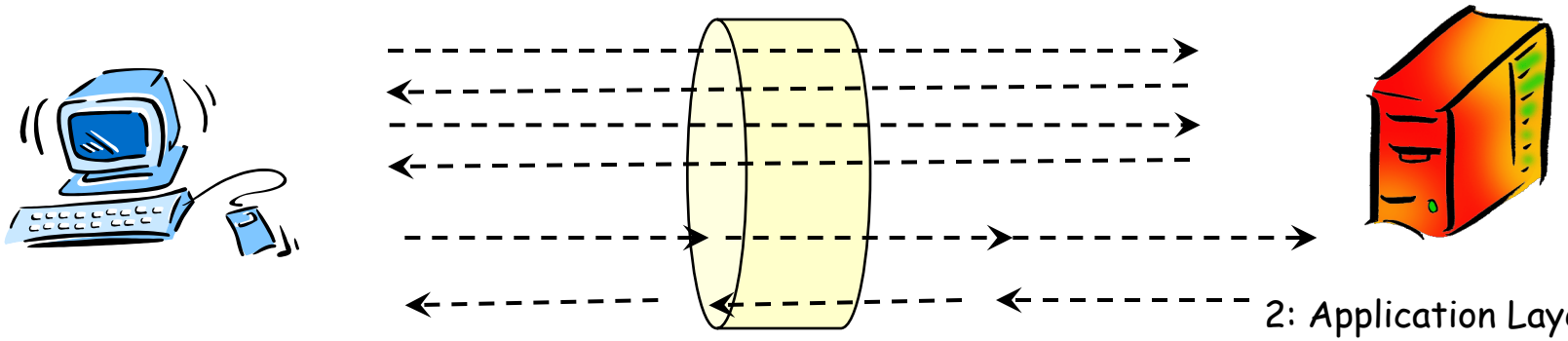
Performance drawbacks

- 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