Reti di Calcolatori:
Internet, Intranet e Mobile Computing
a.a. 2007/2008

http://www.di.uniba.it/~lisi/courses/reti/reti0708.htm

dott.ssa Francesca A. Lisi
lisi@di.uniba.it

Orario di ricevimento: mercoledì ore 10-12

dib

Sommario della lezione di oggi: Lo strato di applicazione (1/3)

- Principi dei protocolli dello strato di applicazione
- World Wide Web & HTTP
- Trasferimento di file & il protocollo FTP
- Posta elettronica & SMTP
- DNS: il servizio directory di Internet
- Condivisione di file

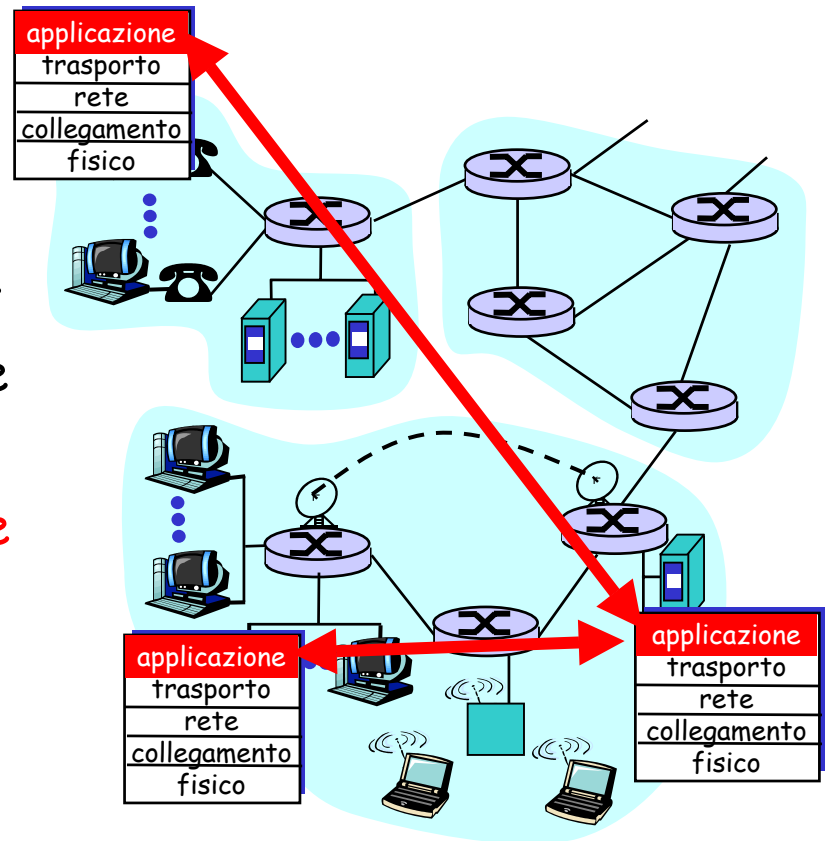
Applicazioni di rete e protocolli dello strato di applicazione

Applicazione di rete = insieme di processi distribuiti e comunicanti

- processi in esecuzione sui terminali nello "spazio utente"
- processi si scambiano messaggi per implementare l'applicazione
- e.g., email, Web

Protocollo dello strato di applicazione

- "pezzo" di un'applicazione di rete
- definisce messaggi scambiati dalle applicazioni e azioni da intraprendere
- usa i servizi forniti da protocolli dello strato sottostante



Applicazioni di rete: un po' di gergo tecnico

- Un processo è un programma in esecuzione su un terminale.
- Nello stesso terminale, due processi comunicano con comunicazione inter-processo definita dal sistema operativo.
- I processi in corso su diversi terminali comunicano tramite un protocollo dello strato di applicazione

- Un agente utente è una interfaccia fra l'utente e l'applicazione di rete.
  - Web: browser
  - E-mail: mail reader
  - streaming audio/video: media player

I protocolli dello strato di applicazione

Application Programming Interface (API):

- definisce l'interfaccia fra strato di applicazione e strato di trasporto
- *socket*: Internet API
  - due processi comunicano inviando dati nel socket, leggendo dati dal socket

D: come fa un processo P1 ad "identificare" il processo P2 con cui vuole comunicare?

- Indirizzo IP del terminale T2 che esegue P2
- numero di porta su cui P2 sta girando - consente al T2 di determinare a quale processo locale il messaggio dovrebbe essere consegnato

Di quale servizi di trasporto ha bisogno un'applicazione di rete?

Trasferimento affidabile dei dati

- alcune applicazioni, p.e. audio, possono tollerare una certa perdita di dati
- altre applicazioni, p.e. file transfer e telnet, richiedono affidabilità al 100%

Tempismo

- alcune applicazioni, p.e. telefonia Internet e giochi interattivi, richiedono un basso ritardo per essere "efficace"

Larghezza di banda

- alcune applicazioni, p.e. multimedia, richiedono un minimo ammontare di larghezza di banda per essere "efficace"
- altre applicazioni ("applicazioni elastiche") fanno uso di qualsiasi larghezza di banda esse dispongano

Applicazioni Internet: requisiti sui servizi di trasporto

Applicazione	Tolleranza alla perdita di dati	Ampiezza di banda	Sensibilità al tempo
Trasferimento file	No	Variabile	No
Posta elettronica	No	Variabile	No
Documenti Web	No	Variabile	No
Audio/video in tempo reale	Sì	Audio: da 5 Kbps a 1 Mbps Video: da 10 Kbps a 5 Mbps	Sì, centinaia di ms
Audio/video memorizzati	Sì	Come sopra	Sì, pochi secondi
Giochi interattivi	Sì	Fino a pochi Kbps	Sì, centinaia di ms
Messaggistica istantanea	No	Variabile	Sì e no

Servizi forniti dai protocolli di trasporto di Internet

Servizi TCP:

- *orientamento alla connessione:* setup di richiesta fra *client* e *server*
- *trasferimento affidabile dei dati*
- *controllo di flusso:* il mittente non sommergerà il ricevente
- *controllo della congestione:* rallenta il mittente se la rete è sovraccarica
- *non fornisce*: tempismo, garanzie di ampiezza minima di banda

Servizi UDP:

- *trasferimento dei dati*
- *non fornisce*: orientamento alla connessione, trasferimento affidabile dei dati, controllo di flusso, controllo di congestione, tempismo, o garanzia di ampiezza di banda

D: Perché esiste un modello di servizi UDP?

Applicazioni Internet: protocolli di applicazione e di trasporto

Applicazione	Protocollo a livello applicazione	Protocollo di trasporto sottostante
Posta elettronica	SMTP [RFC 2821]	TCP
Accesso a terminali remoti	Telnet [RFC 854]	TCP
Web	HTTP [RFC 2616]	TCP
Trasferimento file	FTP [RFC 959]	TCP
Multimedia in streaming	Proprietario (ad esempio, RealNetworks)	TCP o UDP
Telefonia Internet	Proprietario (ad esempio, Vonage, Dialpad)	Tipicamente UDP

Architettura delle applicazioni di rete

- Client-server
- Peer-to-peer (P2P)
- Architetture ibride (client-server e P2P)

Il paradigma client-server

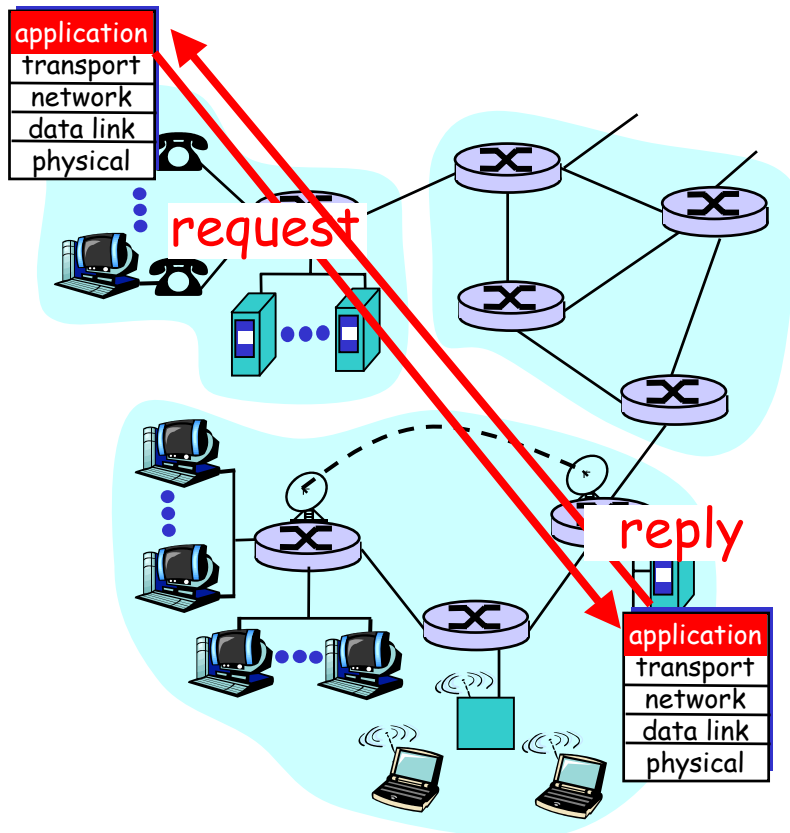
Una tipica appl. di rete ha due pezzi: un *client* ed un *server*

Client:

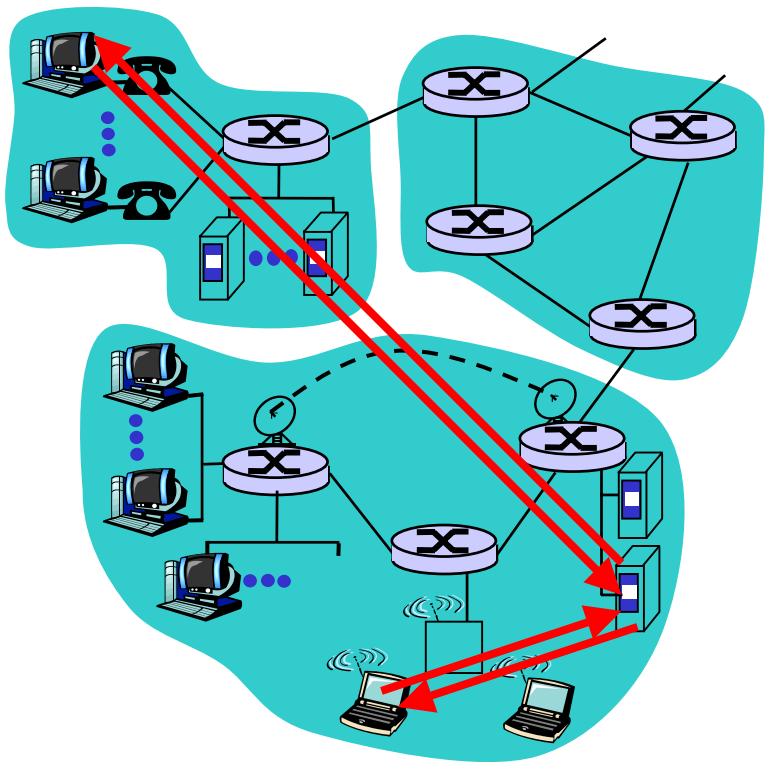
- si mette in contatto con il *server* ("parla per primo")
- tipicamente richiede un servizio al *server*,
- per il Web, il *client* è implementato nel browser; per la e-mail, nel mail reader

Server:

- fornisce il servizio richiesto
- p.e., il server Web invia la pagina Web richiesta, il mail server consegna la e-mail



Architettura delle applicazioni di rete: modello client-server



server:

- host sempre attivo
- indirizzo IP fisso
- server farm per creare un potente server virtuale

client:

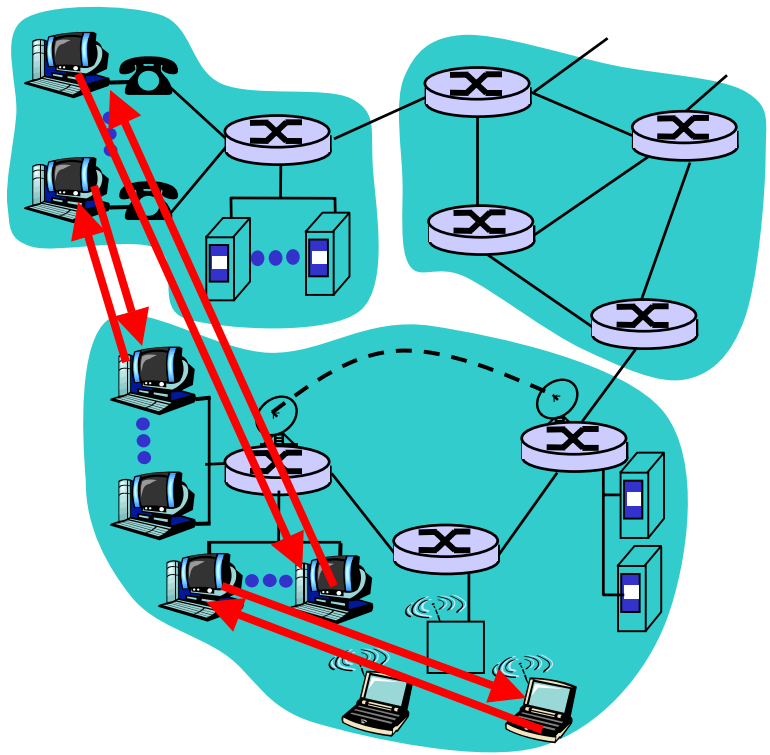
- comunica con il server
- può contattare il server in qualunque momento
- può avere indirizzi IP dinamici
- non comunica direttamente con gli altri client

Architettura delle applicazioni di rete: modello P2P

- non c'è un server sempre attivo
- coppie arbitrarie di host (peer) comunicano direttamente tra loro
- i peer non devono necessariamente essere sempre attivi, e cambiano indirizzo IP
- Un esempio: Gnutella

  Facilmente scalabile

  Difficile da gestire



Architettura delle applicazioni di rete: modello ibrido

Napster

- Scambio di file secondo la logica P2P
- Ricerca di file centralizzata:
  - i peer registrano il loro contenuto presso un server centrale
  - i peer chiedono allo stesso server centrale di localizzare il contenuto

Messaggistica istantanea

- La chat tra due utenti è del tipo P2P
- Individuazione della presenza/location centralizzata:
  - l'utente registra il suo indirizzo IP sul server centrale quando è disponibile online
  - l'utente contatta il server centrale per conoscere gli indirizzi IP dei suoi amici

Il World Wide Web: un po' di gergo tecnico

- Pagina Web:
  - consiste di "oggetti"
  - indirizzata da una URL (Uniform Resource Locator)
- La maggior parte delle pagine Web consiste di:
  - pagina base in HTML, e
  - diversi oggetti referenziati.
- URL ha due componenti: *host name* e *path name*

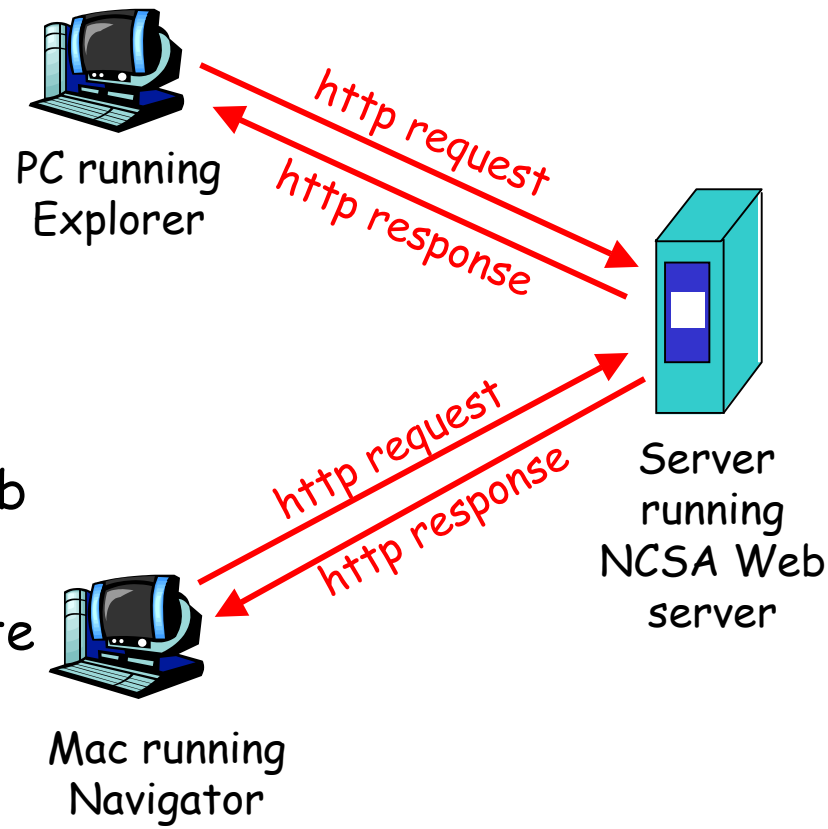
- L'agente utente per il WWW è chiamato *browser*:
  - MS Internet Explorer
  - Netscape Communicator
- Server per il WWW è detto *Web server*:
  - Apache (public domain)
  - MS Internet Information Server

www.someSchool.edu/someDept/pic.gif

Il World Wide Web: il protocollo HTTP

HTTP: hypertext transfer protocol

- protocollo dello strato di applicazione per il WWW
- modello client/server
  - *client:* browser che richiede, riceve, "visualizza" oggetti Web
  - *server:* Web server invia oggetti in risposta alle richieste
- HTTP/1.0: RFC 1945
- HTTP/1.1: RFC 2068



Il World Wide Web: il protocollo HTTP (cont.)

Servizio di trasporto TCP:

- il client avvia una connessione TCP (crea un socket) col server, porta 80
- il server accetta la connessione TCP dal client
- messaggi HTTP (messaggi di protocollo dello strato di applicazione) scambiati fra browser (HTTP client) e Web server (HTTP server)
- chiusura della connessione TCP

HTTP è "stateless"

- il server non conserva alcuna info sulle passate richieste del client

a latere

I protocolli che conservano lo "stato" sono complessi!

- La storia passata (stato) deve essere conservata
- se server/client crolla, le loro viste di "stato" potrebbero essere inconsistenti, vanno quindi riconciliate

Esempio di sessione HTTP

Supponi che un utente visiti la pagina con URL
www.someSchool.edu/someDepartment/home.index

(contiene testo, riferimenti a 10 immagini jpeg)

1a. Il client http inizia una connessione TCP al server HTTP (processo) del terminale www.someSchool.edu. La porta 80 è di default per il server HTTP.

1b. Il server HTTP del terminale www.someSchool.edu in attesa di connessione TCP alla porta 80, "accetta" la connessione, inviandone notifica al client

2. Il client HTTP invia *messaggio di richiesta http* (contenente la URL) nel socket della connessione TCP

3. Il server HTTP riceve il messaggio di richiesta, forma il *messaggio di risposta* contenente l'oggetto richiesto (someDepartment/home.index), e lo invia nel socket

tempo

Esempio di sessione HTTP (cont.)

4. Il server HTTP chiude la connessione TCP.

5. Il client HTTP riceve il messaggio di risposta contenente il file HTML che viene visualizzato. Analizzando il file HTML, trova 10 oggetti jpeg referenziati

6. I passi 1-5 ripetuti per ciascuno dei 10 oggetti .jpeg

tempo

Connessioni non-persistenti e persistenti

Non-persistenti

- HTTP/1.0
- il server parserizza la richiesta, risponde, e chiude la connessione TCP
- 2 RTTs per prelevare ciascun oggetto
- Ogni trasferimento di oggetto soffre di un avvio lento

Ma la maggior parte dei browser 1.0 usano connessioni TCP parallele.

Persistenti

- default per HTTP/1.1
- durante la stessa connessione TCP: il server parserizza la richiesta, risponde, parserizza una nuova richiesta,..
- Il client invia richieste per tutti gli oggetti referenziati non appena riceve l'HTML di base.
- Meno RTTs ed avvio meno lento.

Formato di un messaggio HTTP: richiesta

- Due tipi di messaggi HTTP: *richiesta, risposta*
- messaggio di richiesta HTTP:
  - ASCII (human-readable format)

linea di richiesta (comandi GET, POST, HEAD)

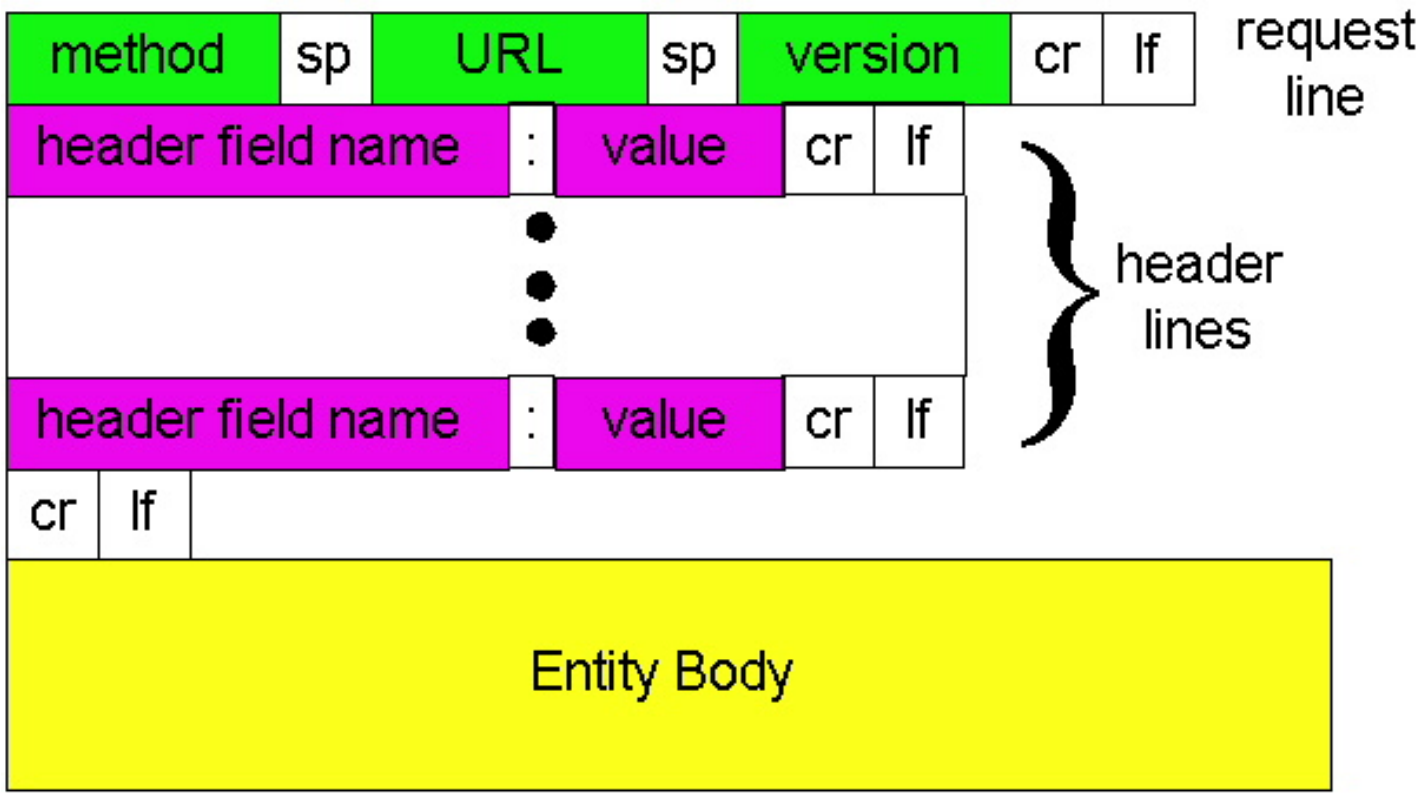
linee di intestazione

```
GET /somedir/page.html HTTP/1.0
User-agent: Mozilla/4.0
Accept: text/html, image/gif,image/jpeg
Accept-language:fr
```

Ritorno a capo, line feed indica la fine del messaggio

(extra carriage return, line feed)

Formato di un messaggio HTTP: richiesta (cont.)



Formato di un messaggio HTTP: risposta

Linea di status (protocollo, codice di stato, frase di stato)

linee di intestazione

dati, p.e., file HTML richiesto

```
HTTP/1.0 200 OK
Date: Thu, 06 Aug 1998 12:00:15 GMT
Server: Apache/1.3.0 (Unix)
Last-Modified: Mon, 22 Jun 1998 …...
Content-Length: 6821
Content-Type: text/html

dati dati dati dati dati ...
```

Formato di un messaggio HTTP: risposta (cont.)

Esempi di codice di stato:

200 OK

- richiesta di successo, oggetto richiesto più tardi in questo messaggio

301 Moved Permanently

- oggetto richiesto spostato, nuova locazione specificata più tardi in questo messaggio (Location:)

400 Bad Request

- messaggio di richiesta non compreso dal server

404 Not Found

- documento richiesto non trovato su questo server

505 HTTP Version Not Supported

Vuoi provare HTTP (lato client)?

1. Telnet al tuo sito Web preferito:

```
telnet www.eurecom.fr 80
```

Apri una connessione TCP alla porta 80 (porta di default per il server HTTP) del terminale www.eurecom.fr. Qualsiasi cosa digitata viene inviata alla porta 80 di www.eurecom.fr

2. Formula una richiesta HTTP di tipo GET:

```
GET /~ross/index.html HTTP/1.0
```

Digitando questo (batti due volte il ritorno a capo), tu invii questa richiesta GET minimale (ma completa) al server HTTP

3. Osserva il messaggio di risposta inviato dal server HTTP!

Interazione user-server: autenticazione

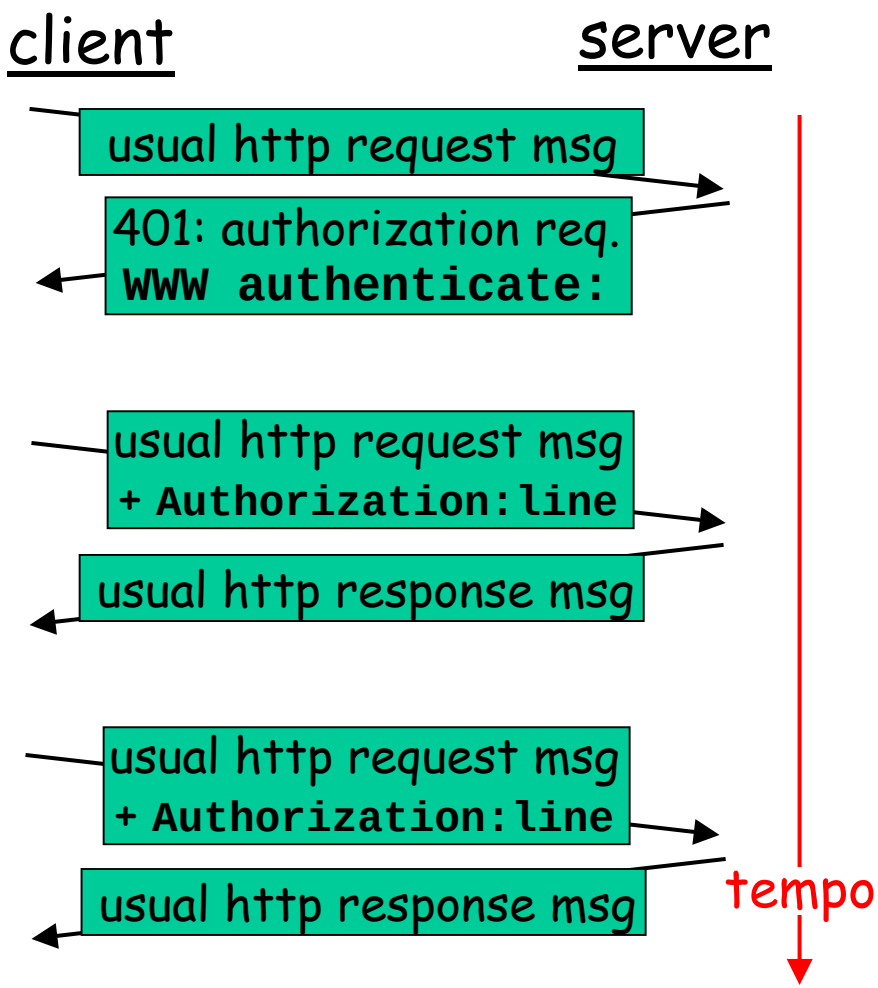
Obiettivo: controllare l'accesso ai documenti residenti sul server

- il client deve presentare autorizzazione in ogni richiesta
  - **`authorization:`** riga di intestazione nella richiesta con user name e password
- se nessuna autorizzazione presentata, il server rifiuta accesso, invia risposta con

  `WWW authenticate:`

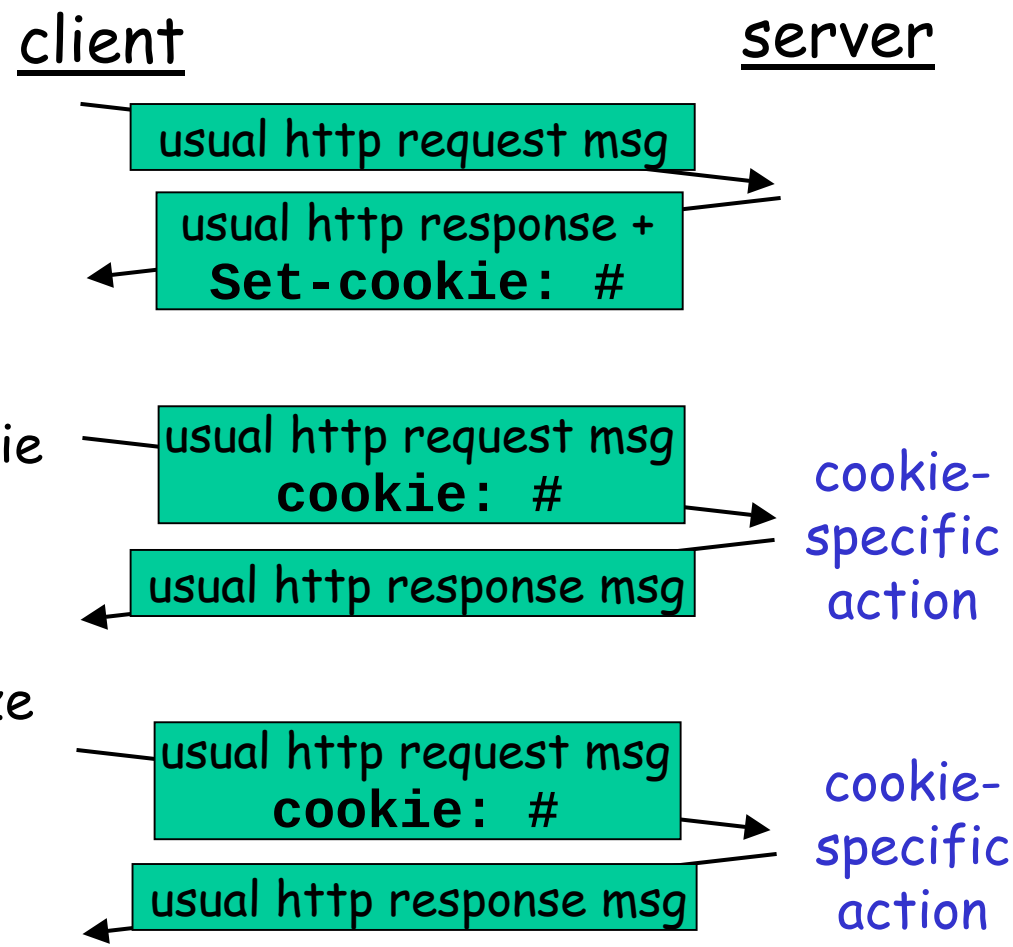
  come riga di intestazione

Il browser pone nome & pwd nella cache così che l'utente non deve reinserirli ripetutamente.



Interazione user-server: cookie

- Il server invia "cookie" al client nel msg di risposta
  `Set-cookie: 1678453`
- il client presenta il cookie nelle successive richieste
  `cookie: 1678453`
- il server confronta il cookie presentato con info memorizzata nel server
  - autenticazione
  - ricordare le preferenze dell'utente, scelte precedenti

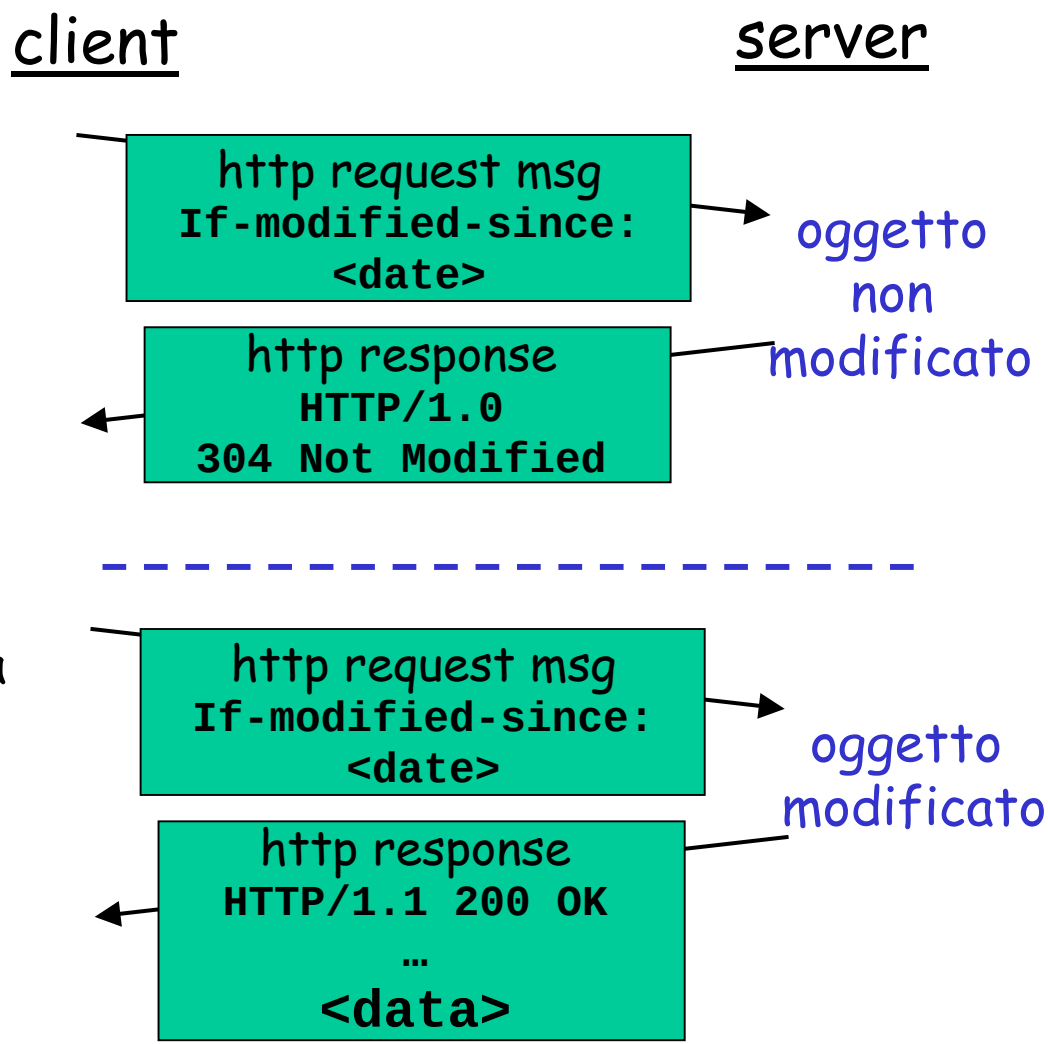


Interazione user-server: GET condizionato

- Obiettivo: non inviare oggetti se il client ha una versione memorizzata aggiornata (in cache)
- client: specificare la data della copia in cache nella richiesta HTTP

  `If-modified-since: <date>`
- server: la risposta non contiene alcun oggetto se la copia in cache è aggiornata:

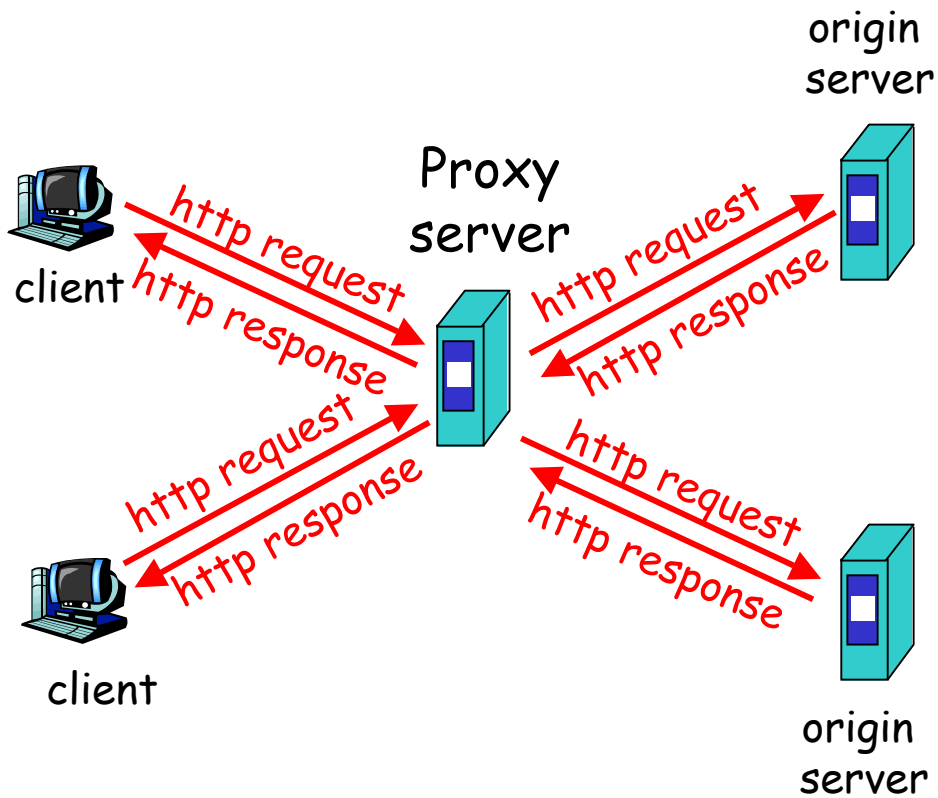
  `HTTP/1.0 304 Not Modified`



Le cache del Web (*proxy server*)

Obiettivo: soddisfare la richiesta del client senza coinvolgere il server di origine

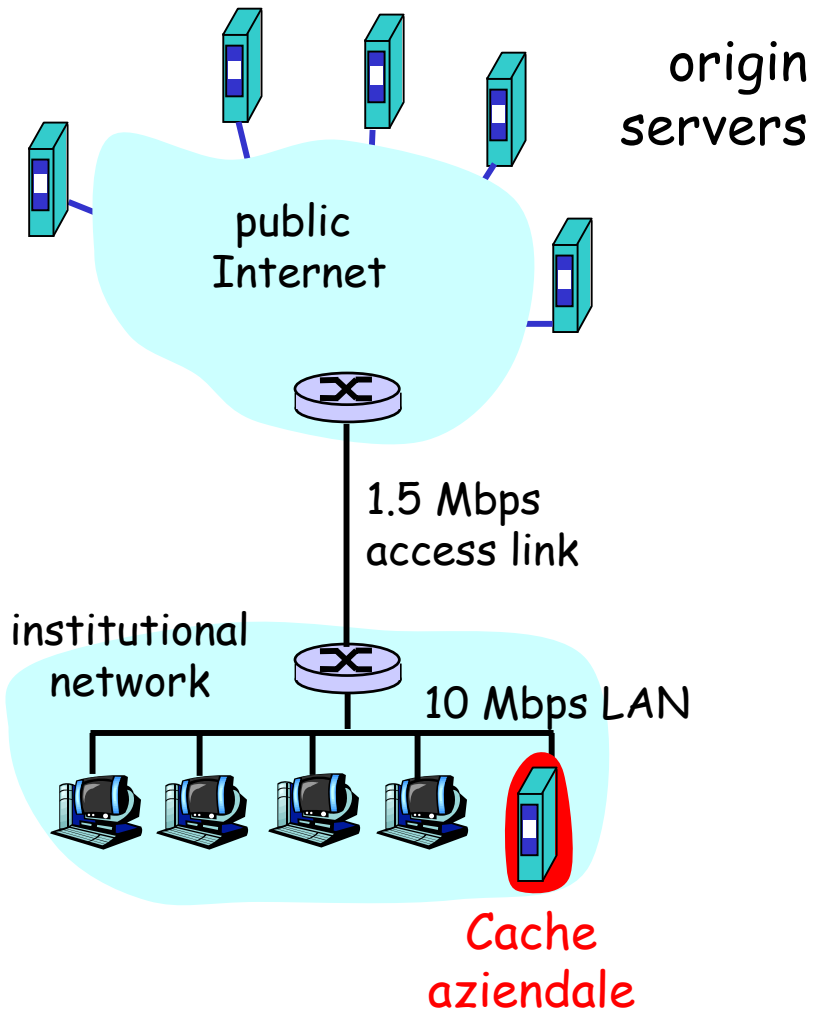
- l'utente imposta il browser: accessi Web via web cache
- il client invia tutte le richieste HTTP alla cache del web
  - se essa contiene l'oggetto, lo invia immediatamente nella risposta HTTP
  - altrimenti richiede l'oggetto dal server di origine, quindi lo acclude alla risposta HTTP



Perché il Web Caching?

Assumiamo che: la cache è "vicina" al client (p.e., nella stessa rete)

- minor tempo di risposta: cache "+ vicina" al client
- decrementa il traffico verso server distanti
  - il collegamento in uscita alla rete di ISP aziendali/locali è spesso un collo di bottiglia



Sommario della prossima lezione: Lo strato di applicazione (2/3)

- Principi dei protocolli dello strato di applicazione
- World Wide Web & HTTP
- Trasferimento di file & il protocollo FTP
- Posta elettronica & SMTP
- DNS: il servizio directory di Internet
- Condivisione di file