Strato di Applicazione

Slide adattate da:

J. Kurose, K. Ross: "*Reti di calcolatori e Internet* (4a edizione)". Pearson Addison Wesley

Strato di applicazione

- **Fornire i concetti base dei protocolli delle applicazioni di rete**
  - modelli di servizio
  - paradigma client-server
  - paradigma peer-to-peer
- **Esame dei più diffusi protocolli di applicazione**
  - HTTP
  - FTP
  - SMTP / POP3 / IMAP
  - DNS

Strato di applicazione

- Principi delle applicazioni di rete
- Servizio Web e protocollo HTTP
- Servizio di file transfer e protocollo FTP
- Servizo di Posta elettronica (e-mail)
  - Protocolli: SMTP, POP3, IMAP
- Servizio Domain Name System (DNS)
- Applicazioni Peer-to-Peer (P2P)

Applicazioni di rete

- Posta elettronica
- Web
- Messaggistica istantanea
- Autenticazione in un calcolatore remoto
- Condivisione di file P2P
- Giochi multiutente via rete
- Streaming di video-clip memorizzati
- Telefonia via Internet
- Videoconferenza in tempo reale
- Grid computing
- …….

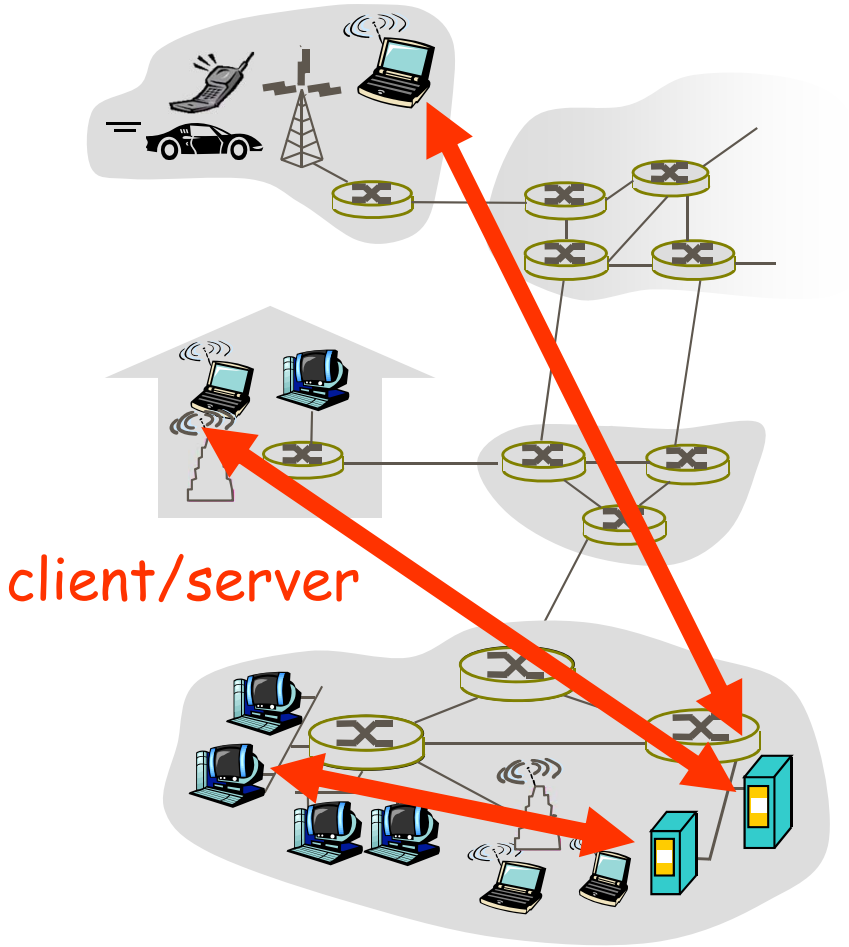
Strato di applicazione

- Principi delle applicazioni di rete
- Web e HTTP
- FTP
- Posta elettronica
  - SMTP, POP3, IMAP
- DNS
- Applicazioni P2P

Architetture delle applicazioni di rete

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

Architettura client-server



server:

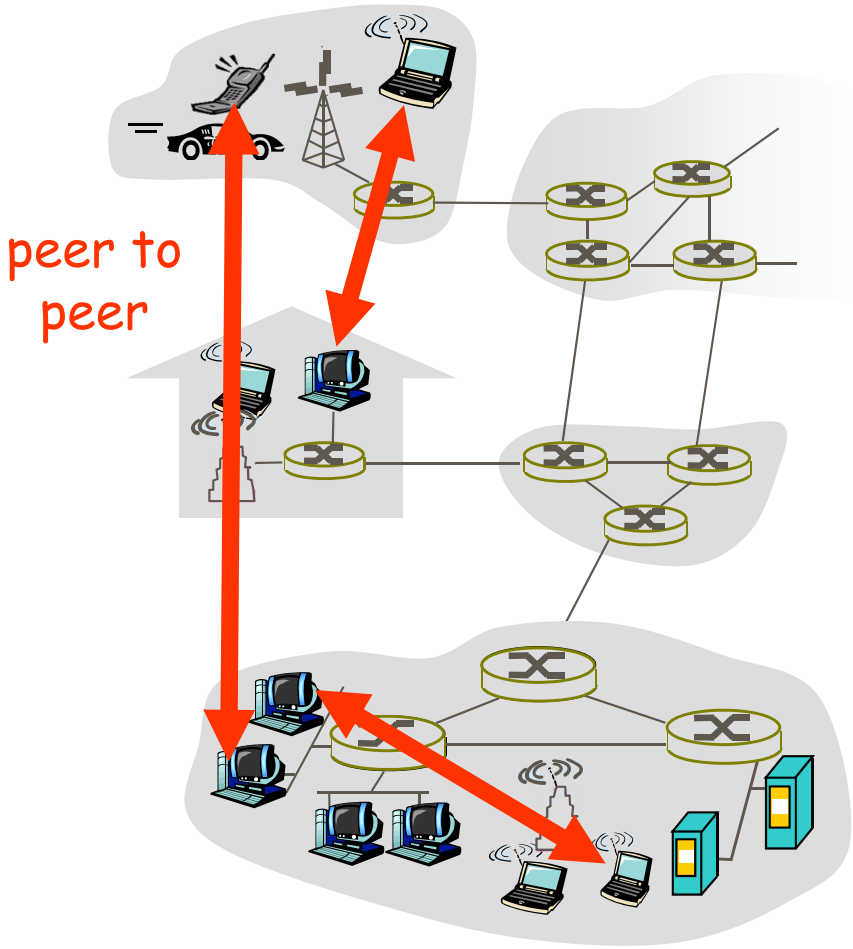
- host (server) sempre attivo
- Indirizzo IP fisso
- Data Center (server virtuale)

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 P2P pura

- Non esiste un server sempre attivo
- Coppie di host (peer) comunicano direttamente tra loro
- I peer non devono necessariamente essere sempre attivi, e possono cambiare indirizzo IP
- Facilmente scalabile
- Difficile da gestire



Archietture ibride (client-server + P2P)

Skype

- Applicazione P2P di Voice over IP
- Server centralizzato: ricerca e memorizza gli indirizzi correnti della utenti
- Connessione client-client: diretta (non attraverso il server)

Messaggistica

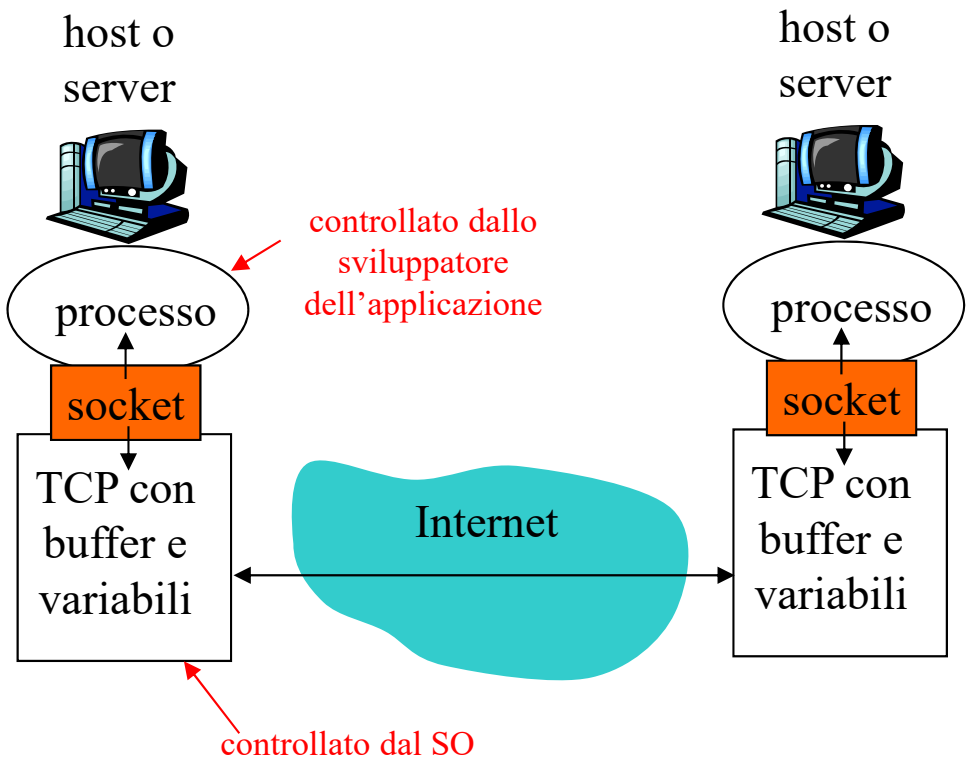
- La chat tra due utenti è del tipo P2P
- Individuazione della presenza/location centralizzata:
  - l'utente registra il suo indirizzo IP sul server centrale quando è disponibile online
  - l'utente contatta il server centrale per conoscere gli indirizzi IP dei suoi amici

Processi comunicanti

- **Processo**
  - programma in esecuzione su di un host.
  - All'interno dello stesso host, due processi comunicano utilizzando schemi interprocesso definiti dal Sistema Operativo
  - processi su host differenti comunicano attraverso la rete mediante lo scambio di messaggi
- **Processo client**
  - processo che dà inizio alla comunicazione
- **Processo server**
  - processo che attende di essere contattato
- **le applicazioni con architetture P2P hanno processi client e processi server**

Socket

- un processo invia/riceve messaggi a/da la sua socket
- una socket è analoga a una porta
  - un processo che vuole inviare un messaggio, lo emette attraverso la propria "porta" (socket)
  - il processo presuppone l'esistenza di un'infrastruttura esterna che trasporterà il messaggio attraverso la rete fino alla "porta" del processo di destinazione



Processi di indirizzamento

- **L'identificatore comprende sia l'indirizzo IP che i numeri di porta associati al processo in esecuzione su un host**
- **Esempi di numeri di porta:**
  - HTTP server: 80
  - Mail server: 25
- **Per inviare un messaggio HTTP al server es. gaia.cs.umass.edu:**
  - Indirizzo IP: 128.119.245.12
  - Numero di porta: 80
- **Affinché un processo su un host invii un messaggio a un processo su un altro host, il mittente deve identificare il processo destinatario**
- **Un host ha un indirizzo IP univoco a 32 bit**
  - D: È sufficiente conoscere l'indirizzo IP dell'host su cui è in esecuzione il processo per identificare il processo stesso?
  - Risposta: No, sullo stesso host possono essere in esecuzione molti processi, quindi è necessario anche il numero di porta

Protocollo a livello di applicazione

- **Tipi di messaggi scambiati**
  - Es. messaggi di richiesta e di risposta
- **Sintassi dei tipi di messaggio**
  - quali sono i campi nel messaggio e come sono descritti
- **Semantica dei campi**
  - significato delle informazioni nei campi
- **Regole per determinare quando e come un processo invia e risponde ai messaggi**

Protocolli di pubblico dominio

- **Definiti nelle RFC**
- **Consentono l'interoperabilità**
- **Es: HTTP, SMTP**

Protocolli proprietari

- **Es: Skype**

Servizio di trasporto di un'applicazione

- **Perdita di dati**
  - alcune applicazioni (ad esempio, audio) possono tollerare qualche perdita
  - altre applicazioni (ad esempio, trasferimento di file, telnet) richiedono un trasferimento dati affidabile al 100%
- **Ritardo**
  - alcune applicazioni (ad esempio, telefonia Internet, giochi interattivi) per essere "realistiche" richiedono piccoli ritardi
- **Throughput**
  - alcune applicazioni (ad esempio, quelle multimediali) per essere "efficaci" richiedono un'ampiezza di banda minima (bit rate) garantita
  - altre applicazioni ("applicazioni elastiche") utilizzano l'ampiezza di banda che si rende disponibile
- **Sicurezza**
  - Cifratura, integrità dei dati, ...

Requisiti del servizio di trasporto

Applicazione	Tolleranza alla perdita di dati	Throughput	Sensibilità al ritardo
Trasferimento file	No	Variabile	No
Posta elettronica	No	Variabile	No
Documenti Web	No	Variabile	No
Real time audio/video	Sì	Audio: da 5 Kbps a 1 Mbps Video: da 10 Kbps a 5 Mbps	Sì (~ 150 ms)
Audio/video memorizzati	Sì	Audio: da 5 Kbps a 1 Mbps Video: da 10 Kbps a 5 Mbps	Sì (~ 1 s)
Giochi interattivi	Sì	Pochi Kbps	Sì (~ 100 ms)
Messaggistica istantanea	No	Variabile	Si/No

Servizi dei protocolli di trasporto Internet

- **Servizio TCP**
- **orientato alla connessione**
  - è richiesto un setup fra i processi client e server
- **affidabilità**
  - trasporto fra i processi d'invio e di ricezione
- **controllo di flusso**
  - evita il sovraccarico del destinatario
- **controllo di congestione**
  - Regola il processo d'invio se la rete è sovraccarica
- **elementi mancanti**
  - ritardo non garantito
  - throughput non garantito
  - sicurezza

- **Servizio di UDP**
- **servizio connectionless**
- **trasferimento dati inaffidabile fra i processi d'invio e di ricezione**
- **elementi mancanti**
  - setup della connessione
  - affidabilità
  - controllo di flusso
  - controllo della congestione
  - controllo del ritardo
  - throughput non garantito
  - sicurezza

Requisiti del servizio di trasporto

Applicazione	Protocollo a livello applicazione	Protocollo di trasporto sottostante
Trasferimento file	FTP [RFC 959]	TCP
Posta elettronica	SMTP [RFC 2821]	TCP
Documenti Web	HTTP [RFC 2616]	TCP
Audio/video	HTTP (es. YouTube) RTP [RFC 1889]	TCP o UDP
Telefonia Internet	SIP, RTP, Prot. proprietari (es. Skype)	UDP
Accesso a terminali remoti	Telnet [RFC 854]	TCP

Strato di applicazione

- Principi delle applicazioni di rete
- Web e HTTP
- File Transfer Protocol (FTP)
- Posta elettronica
  - SMTP, POP3, IMAP
- DNS
- Applicazioni P2P

Web e HTTP

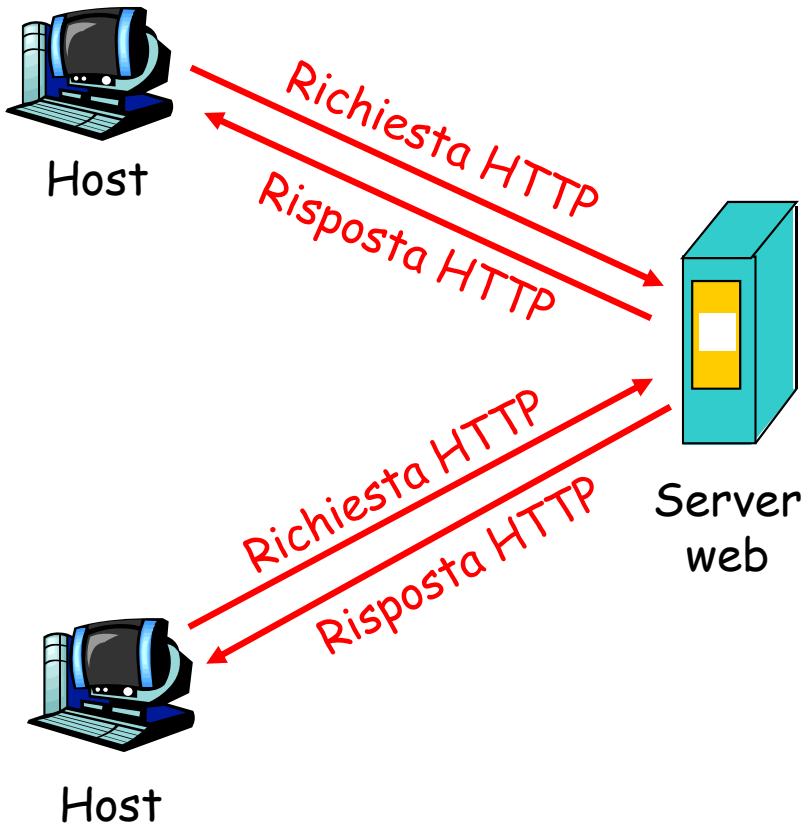
- **Terminologia**
  - Una pagina web è costituita da oggetti (file)
  - Un oggetto può essere un file HTML, un'immagine JPEG, un'applet Java, un file audio/video, …
  - Una pagina web è formata da un file base HTML che include diversi oggetti referenziati
  - Ogni oggetto è referenziato da un Uniform Resource Locator (URL)
    - Un URL ha due componenti: nome del server; percorso dell'oggetto
  - Esempio di URL

`www.someschool.edu/someDept/pic.gif`

nome del server | nome del percorso

HTTP: generalità

- **HTTP**
  - HyperText Transfer Protocol
- **Protocollo a livello di applicazione del Web (RFC 1945, 2616)**
- **Modello client/server**
  - *client:* il browser che richiede, riceve e "visualizza" gli oggetti del Web
  - *server:* il server web invia oggetti in risposta a una richiesta di un client



HTTP: generalità

- **Usa TCP**
  - Il client inizializza la connessione TCP (crea una socket) con il server (porta 80)
  - Il server accetta la connessione TCP dal client
  - Scambio di messaggi HTTP fra browser (client HTTP) e server web (server HTTP)
  - Chiusura della connessione TCP chiusa

- **HTTP è un protocollo stateless**
  - Il server non mantiene informazioni sulle richieste fatte dal client
    - due richieste consecutive dello stesso oggetto danno luogo a trasmissioni distinte

nota

- I protocolli che mantengono lo "stato" sono complessi
  - La storia passata (stato) deve essere memorizzata
  - Se il server e/o il client si bloccano, le loro viste dello "stato" potrebbero essere contrastanti e dovrebbero essere riconciliate

Connessioni HTTP

Connessioni non persistenti

- Gli oggetti sono trasmessi su connessione TCP distinte tra client e server (connessioni TCP dedicate)

Connessioni persistenti

- Più oggetti possono essere trasmessi su una singola connessione TCP tra client e server (connessione TCP condivisa)
  - La connessione TCP rimane attiva dopo il termine dell'invio di un oggetto fino alla scadere di un timeout

Connessioni non persistenti (1/2)

Supponiamo che l'utente immetta l'URL
`www.someSchool.edu/someDepartment/home.index`

(contiene testo, riferimenti a 10 immagini jpeg)

tempo

1a. Il client HTTP inizializza una connessione TCP sulla porta 80 con il server HTTP (processo) il cui nome è *www.someSchool.edu*

1b. Il server HTTP *www.someSchool.edu* in attesa di una connessione TCP alla porta 80 "accetta" la connessione e avvisa il client

2. Il client HTTP trasmette un *messaggio di richiesta* verso la socket della connessione TCP. Il messaggio indica che il client vuole l'oggetto *someDepartment/home.index*

3. Il server HTTP riceve il messaggio di richiesta, forma il *messaggio di risposta* che contiene l'oggetto richiesto e invia il messaggio nella sua socket

Connessioni non persistenti (2/2)

tempo

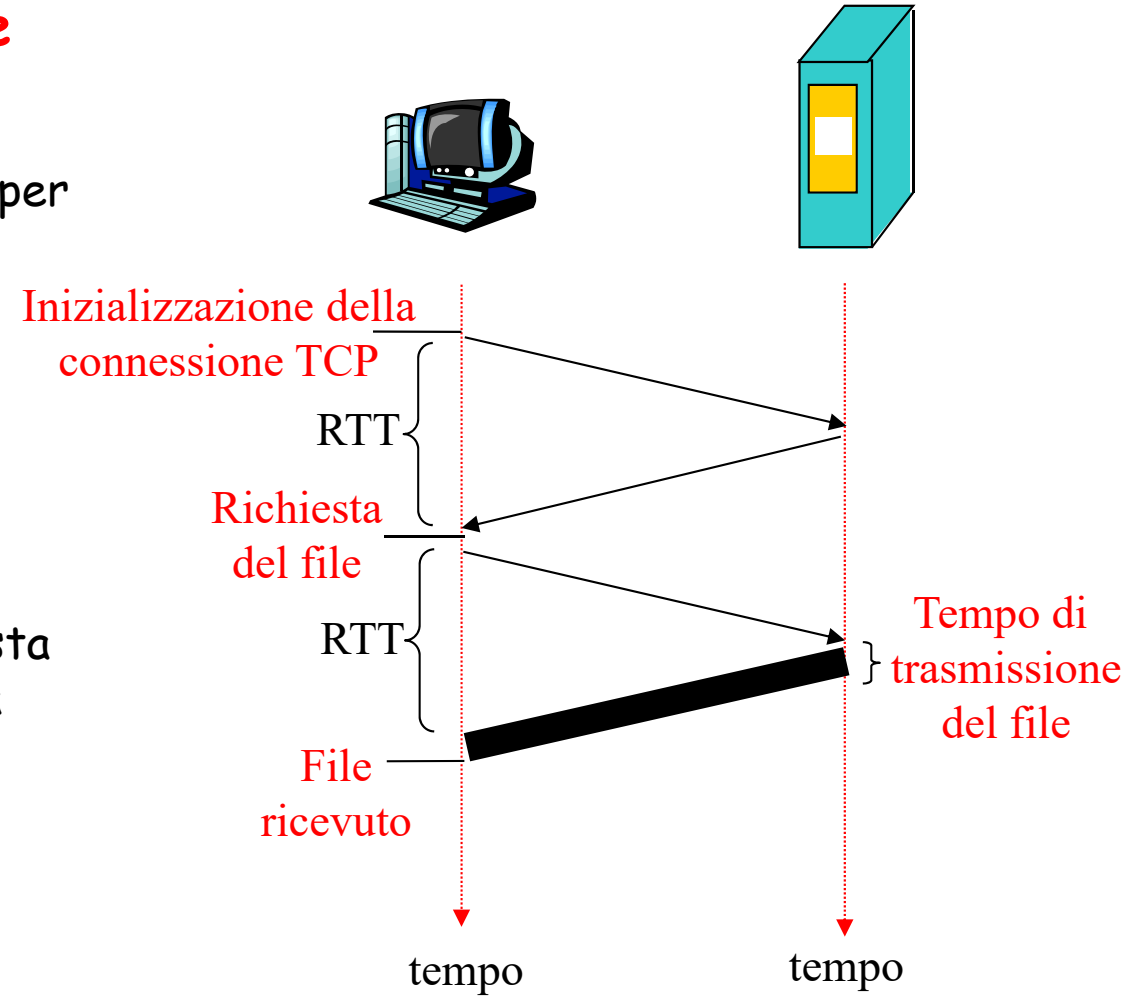
4. Il server HTTP chiude la connessione TCP

5. Il client HTTP riceve il messaggio di risposta che contiene il file html e visualizza il documento html. Esamina il file html, trova i riferimenti a 10 oggetti jpeg

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

Schema del tempo di risposta

- **Definizione di Round Trip Time (RTT)**
  - tempo impiegato da un pacchetto per andare dal client al server e ritornare al client.
- **Tempo di risposta**
  - un RTT per inizializzare la connessione TCP
  - un RTT perché ritornino la richiesta HTTP e i primi byte della risposta HTTP
  - tempo di trasmissione del file



Ritardo totale =

= 2RTT + tempo di trasmissione del file

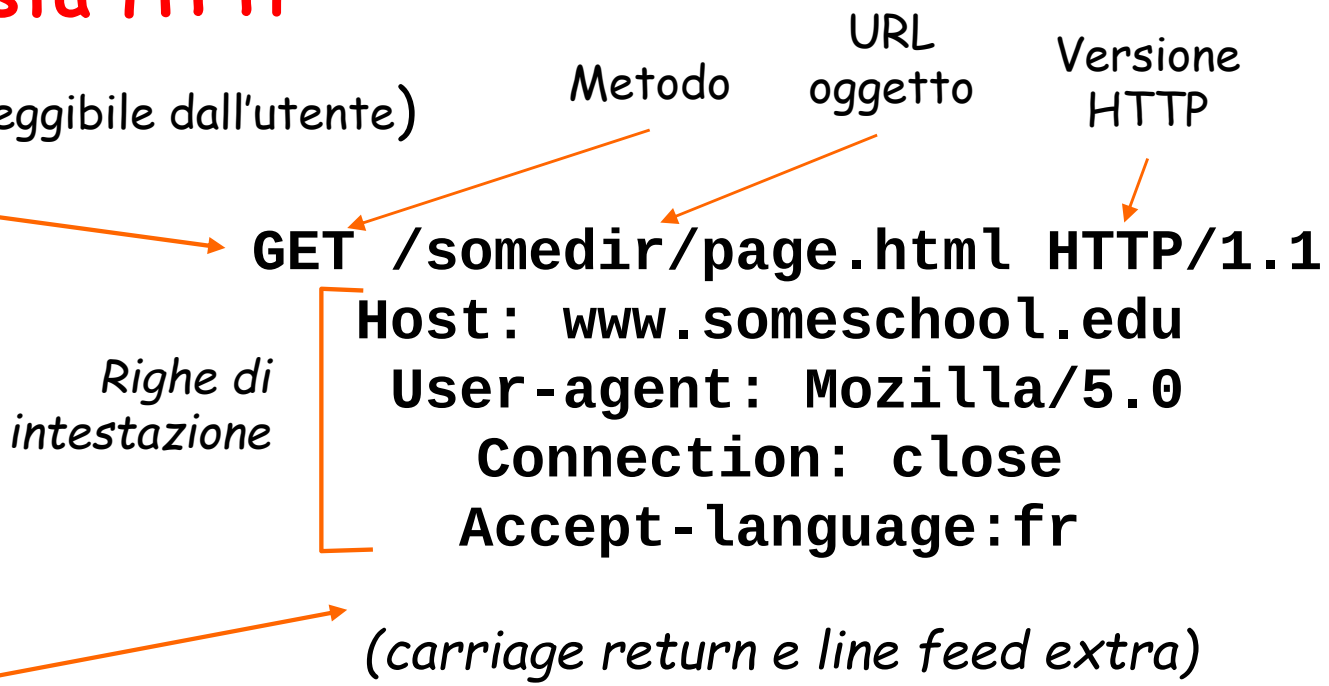
Non persistent vs Persistent connections

- **Connessioni non persistenti**
- **Svantaggi**
- richiedono un tempo pari a 2 RTT per ogni oggetto
- overhead del sistema operativo per *ogni* connessione TCP
- i browser spesso aprono connessioni TCP parallele per trasmettere gli oggetti referenziati
  - In HTTP 1.1 un browser apre da 5 a 10 connessioni TCP (parametro configurabile)

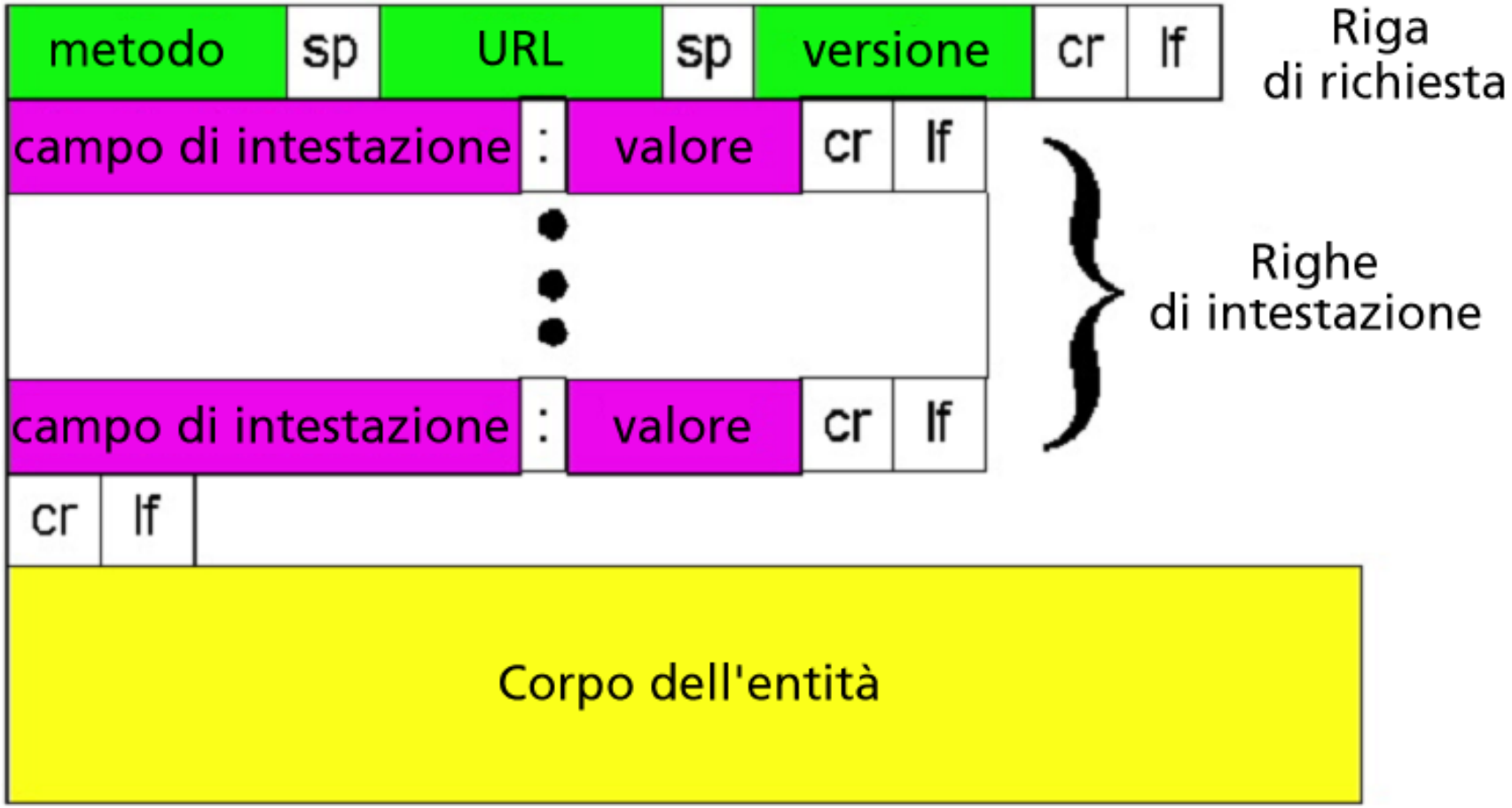
- **Connessioni persistenti**
- il server lascia la connessione TCP aperta dopo l'invio di una risposta
- i successivi messaggi tra gli stessi client/server vengono trasmessi sulla connessione TCP aperta
- il client invia le richieste non appena incontra un oggetto referenziato
- un solo RTT per tutti gli oggetti referenziati

Messaggi HTTP

- due tipi di messaggi HTTP: *richiesta, risposta*
- **Messaggio di richiesta HTTP**
  - Testo ASCII (formato leggibile dall'utente)
    - Riga di richiesta
    - (comandi GET, POST, HEAD)
    - Un carriage return e un line feed indicano la fine del messaggio



Messaggio di richiesta HTTP: formato generale



Upload dell'input di un form

Metodo Post

- La pagina web spesso include un form per l'input dell'utente
  - es. Query per un motore di ricerca)
- L'input arriva al server nel corpo dell'entità

Metodo URL

- Usa il metodo GET
- L'input arriva al server nel campo URL della riga di richiesta:

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

Tipi di metodi

HTTP/1.0

- GET
- POST
- HEAD
  - chiede al server di escludere l'oggetto richiesto dalla rispost
  - usato per diagnostica

HTTP/1.1

- GET, POST, HEAD
- PUT
  - include il file nel corpo dell'entità e lo invia al percorso specificato nel campo URL
- DELETE
  - cancella il file specificato nel campo URL

Messaggio di risposta HTTP

Riga di stato (protocollo codice di stato espressione di stato)

Righe di intestazione

dati, ad esempio il file HTML richiesto

```
HTTP/1.1 200 OK
Connection close
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 ...
```

Codici della risposta HTTP

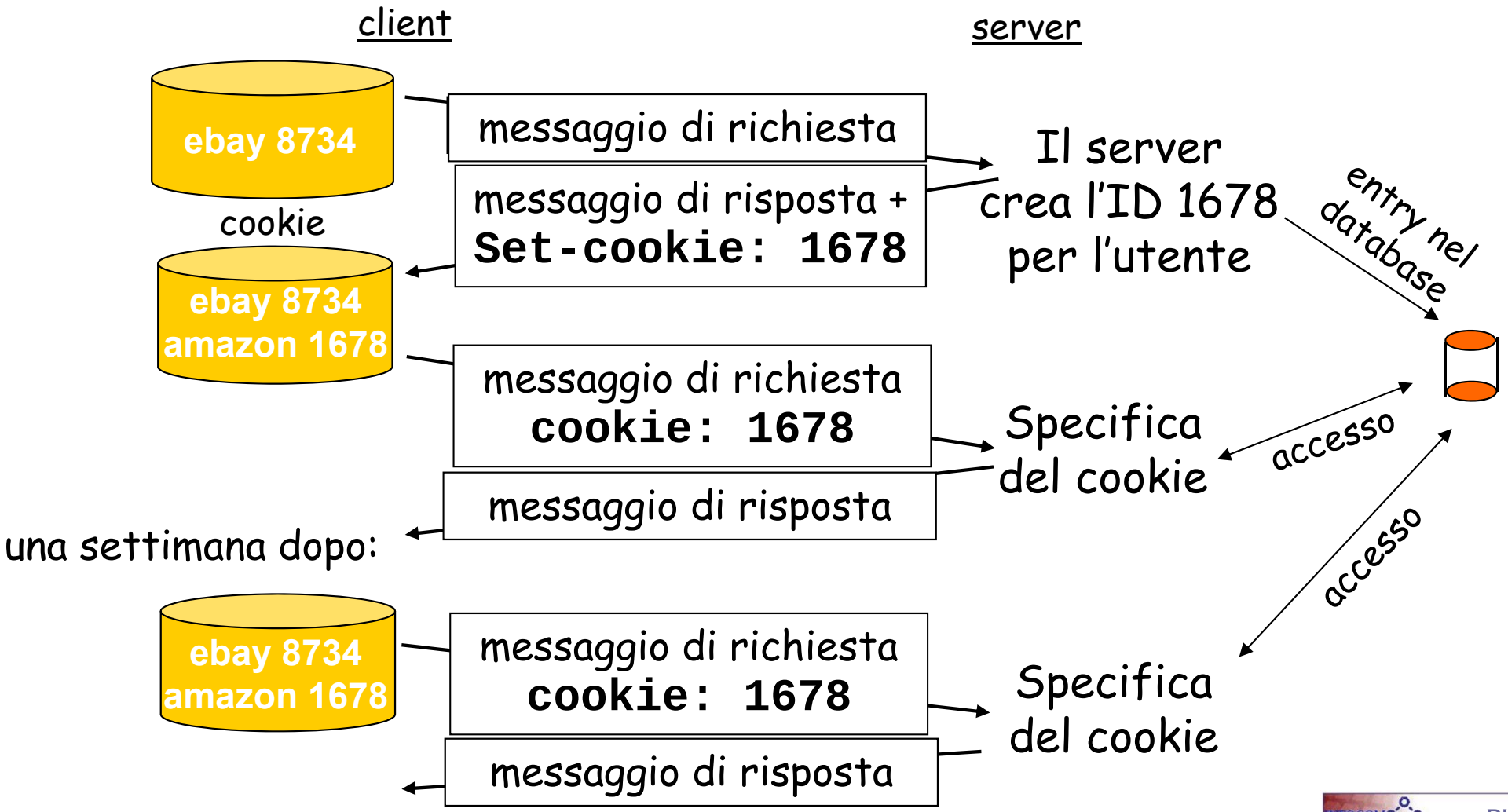
- Nella prima riga nel messaggio di risposta server->client
  - **200 OK**
    - La richiesta ha avuto successo; l'oggetto richiesto viene inviato nella risposta
  - **301 Moved Permanently**
    - L'oggetto richiesto è stato trasferito; la nuova posizione è specificata nell'intestazione **Location** della risposta
  - **400 Bad Request**
    - Il messaggio di richiesta non è stato compreso dal server
  - **404 Not Found**
    - Il documento richiesto non si trova su questo server
  - **505 HTTP Version Not Supported**
    - Il server non supporta la versione richiesta del protocollo HTTP

Interazione utente-server: i cookie

- **Molti dei più importanti siti web usano i cookie [RFC 6265]**
- **Quattro componenti**
  1) Una riga di intestazione nel messaggio di *risposta* HTTP
  2) Una riga di intestazione nel messaggio di *richiesta* HTTP
  3) Un file cookie mantenuto sul sistema terminale dell'utente e gestito dal browser dell'utente
  4) Un database gestito dal server

- **Esempio**
  - Susan accede sempre a Internet dallo stesso PC
  - Visita per la prima volta un particolare sito di commercio elettronico
  - Quando la richiesta HTTP iniziale giunge al sito, il sito crea un identificativo unico (ID) e una *entry* nel database per ID

Cookie (continua)



Cookie (continua)

- **Cosa possono contenere i cookie**
  - autorizzazione
  - carta per acquisti
  - raccomandazioni
  - stato della sessione dell'utente (e-mail)
- **Lo "stato":**
  - Mantengono lo stato del mittente e del ricevente per più transazioni
  - I messaggi http trasportano lo stato

nota

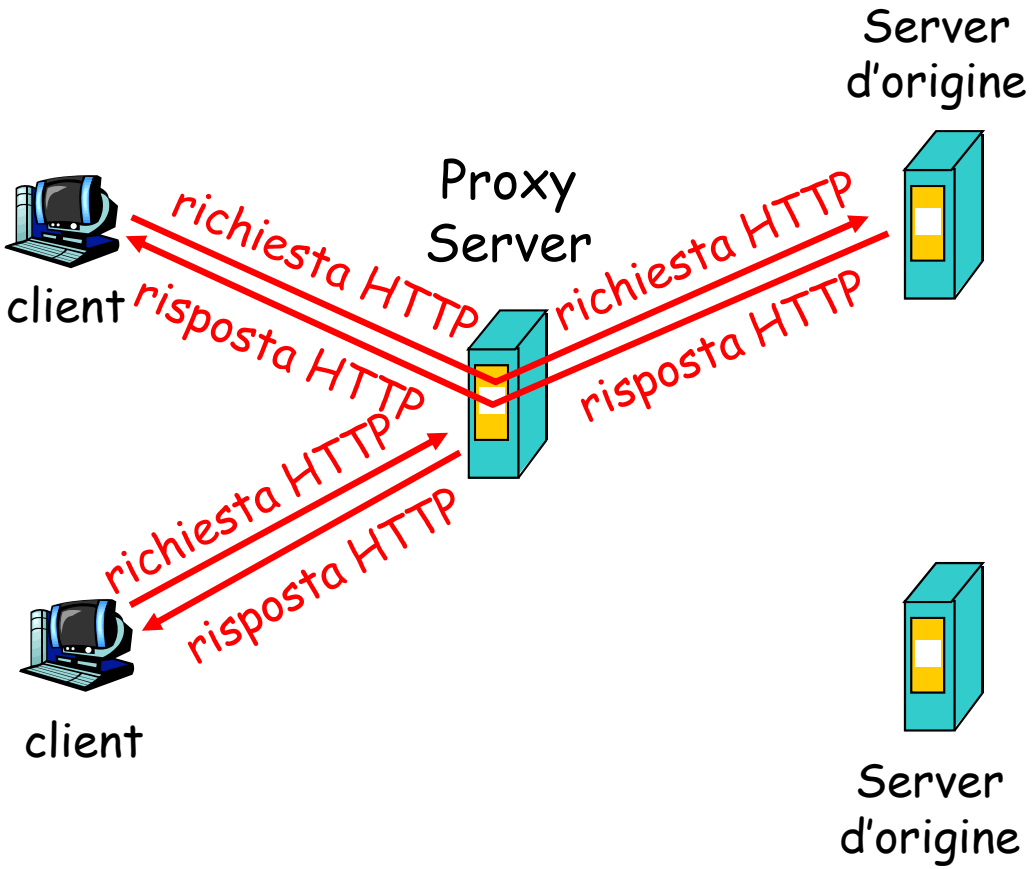
Cookie e privacy

- i cookie permettono ai siti di imparare molte cose sugli utenti
- l'utente può fornire al sito il nome e l'indirizzo e-mail

Web cache (proxy server)

- Obiettivo
  - soddisfare la richiesta del client senza coinvolgere il server d'origine

- **L'utente configura il browser: accesso al Web tramite la cache**
- **Il browser trasmette tutte le richieste HTTP alla cache**
  - oggetto nella cache: la cache fornisce l'oggetto
  - altrimenti la cache richiede l'oggetto al server d'origine e poi lo inoltra al client



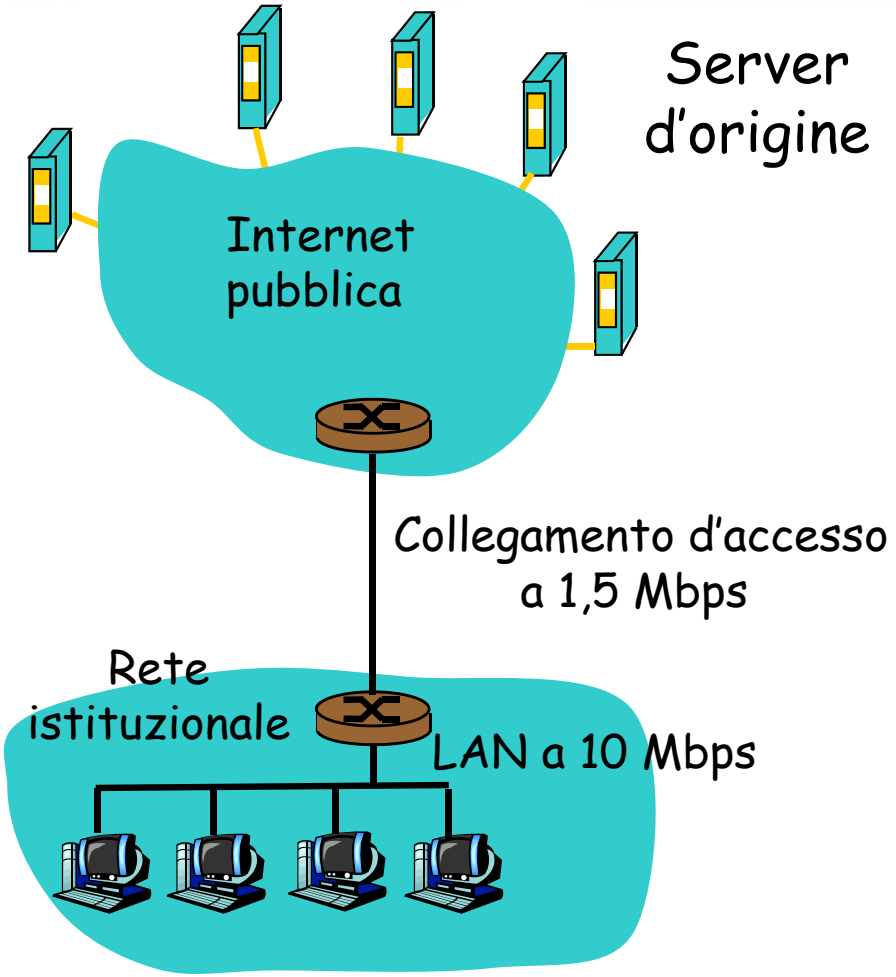
Cache web (continua)

- La cache opera sia come client che come server
- Tipicamente la cache è installata da un ISP (università, aziende o ISP residenziali)

- **Perché il caching web?**
  - Riduce i tempi di risposta alle richieste dei client.
  - Riduce il traffico sul collegamento di accesso a Internet
  - Internet arricchita di cache consente ai provider "scadenti" di fornire dati con efficacia

Esempio di caching (1)

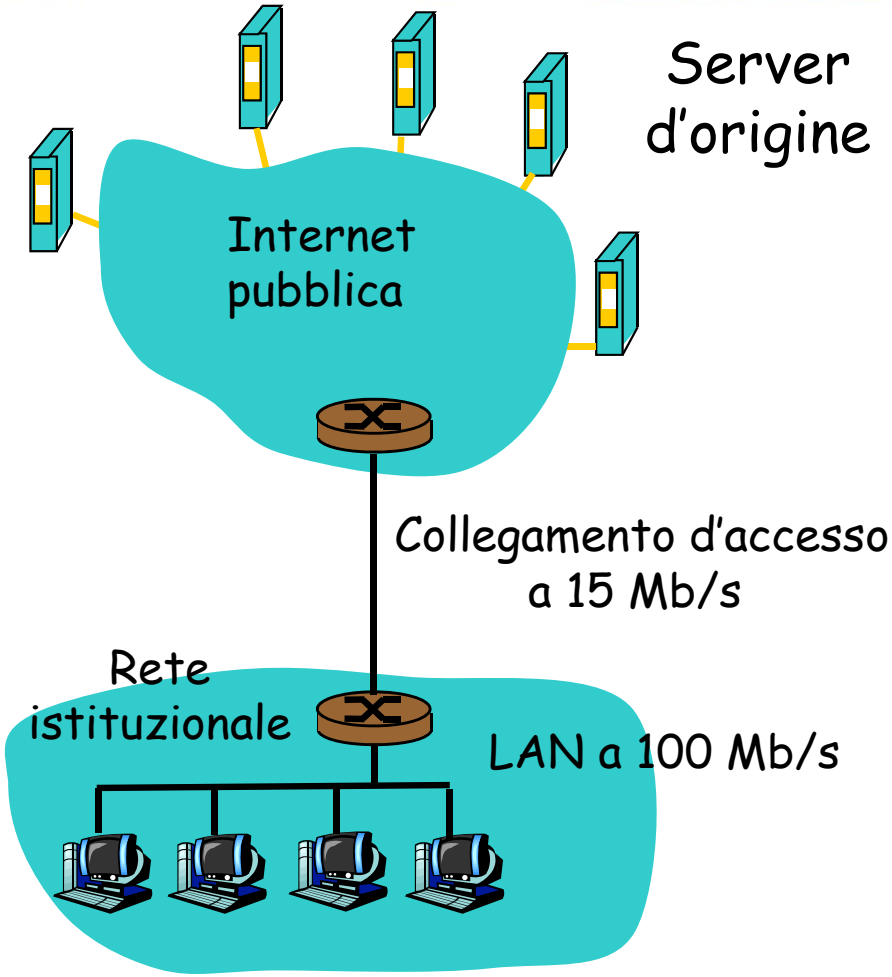
- **Ipotesi**
  - Dimensione media di un oggetto = = 1 Mbit
  - Frequenza media di richieste dai browser istituzionali ai server d'origine = 15 richieste/sec
  - Ritardo dal router istituzionale a qualsiasi server d'origine e ritorno al router = 2 sec (ritardo di Internet)
- **Ritardo totale di risposta**
  - Ritardo LAN + Ritardo accesso + Ritardo internet



Cache istituzionale

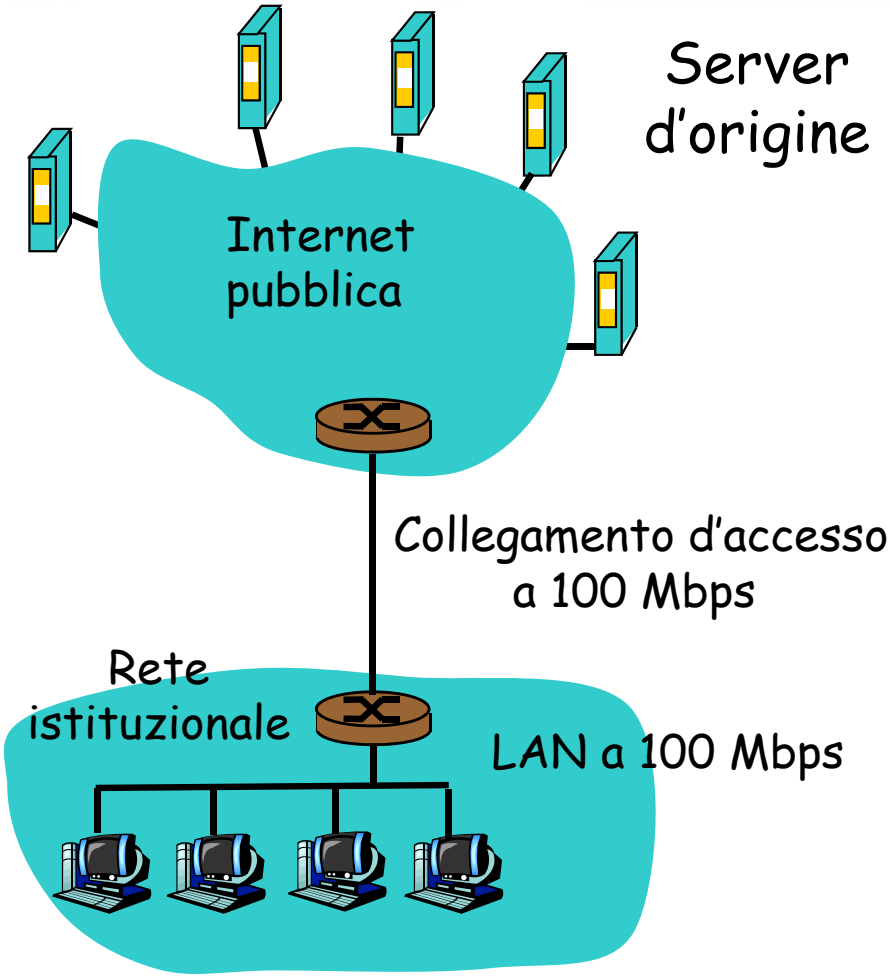
Esempio di caching (2)

- **Intensità del traffico nella rete locale (A_{LAN})**
  - A_{LAN} = 15 r/s x 1 Mbit = 15 Mbit/s
- **Grado di utilizzazione della LAN (ρ_{LAN})**
  - $\rho_{LAN} = A_{LAN} / 100$ Mbit/s = 0.15
  - Ritardo limitato
- **Intensità del traffico nella rete di accesso (A_{acc})**
  - A_{acc} = 15 r/s x 1 Mbit = 15 Mbit/s
- **Grado di utilizzazione della rete di accesso (ρ_{acc})**
  - $\rho_{acc} = A_{LAN} / 15$ Mbit/s = 1
  - Ritardo molto elevato
- **Conseguenze**
  - ritardo totale = 2 sec + x minuti + y millisecondi
  - Il collo di bottiglia (bottleneck) è il segmento di accesso



Esempio di caching (3)

- **Soluzione possibile**
  - aumentare l'ampiezza di banda del collegamento d'accesso a 100 Mbps, per esempio
- **Conseguenze**
  - utilizzo sulla LAN = 15%
  - utilizzo sul collegamento d'accesso = 15%
  - ritardo totale = ritardo di Internet + ritardo di accesso + ritardo della LAN
  - = 2 sec + msecs + msecs
  - Purtroppo l'aggiornamento della rete di accesso è spesso molto costoso
  - Allora ?

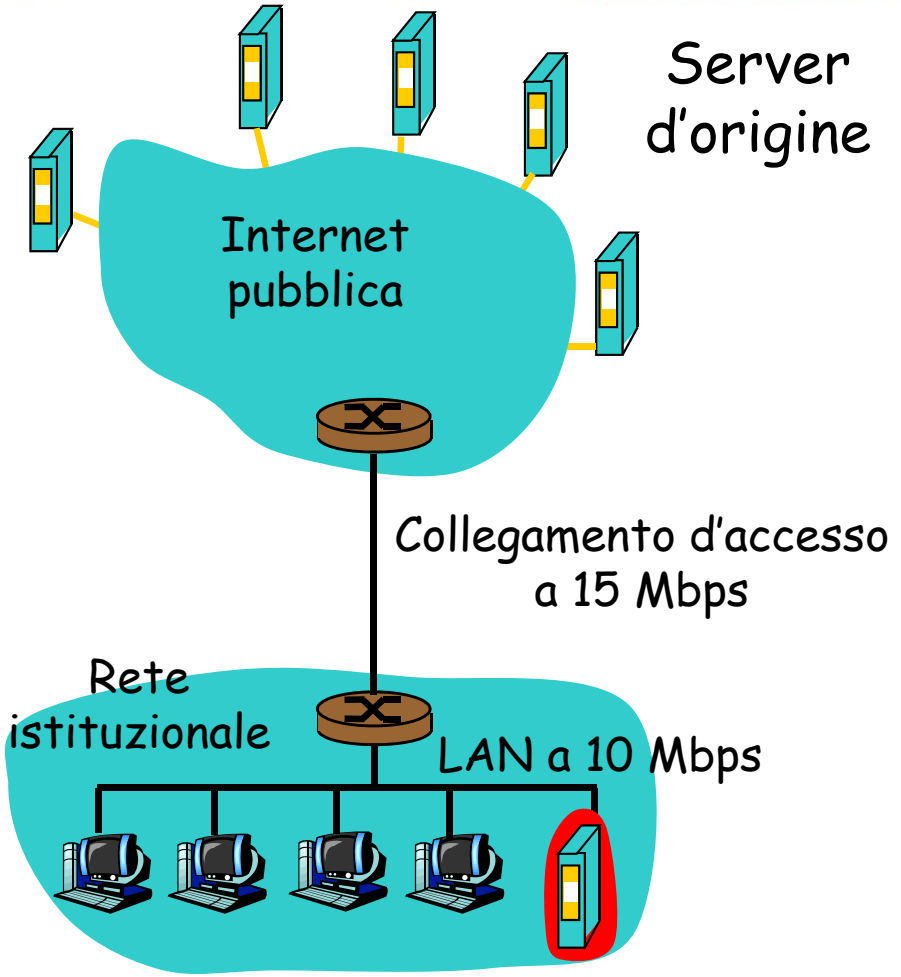


Cache istituzionale

Esempio di caching (4)

- **Soluzione possibile: installare una cache**
  - supponiamo una percentuale di successo (*hit rate*) pari a 0,4
- **Conseguenze**
  - il 40% delle richieste sarà soddisfatto quasi immediatamente
  - il 60% delle richieste sarà soddisfatto dal server d'origine
  - l'utilizzazione del collegamento d'accesso si è ridotta al 60%, determinando ritardi trascurabili (circa 10 msec)
  - ritardo totale medio = ritardo di Internet + ritardo di accesso + ritardo della LAN =

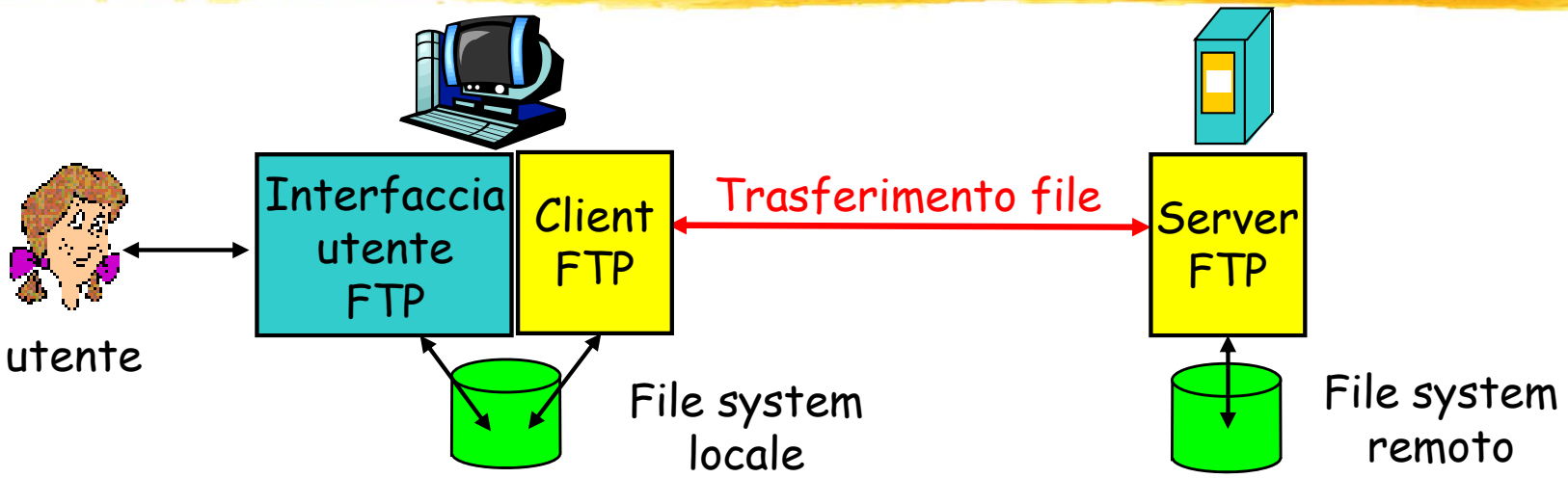
0,6*(2,01) sec + millisecondi < 1,4 sec



Strato di applicazione

- Principi delle applicazioni di rete
- Web e HTTP
- File Transfer Protocol (FTP)
- Posta elettronica
  - SMTP, POP3, IMAP
- DNS
- Applicazioni P2P

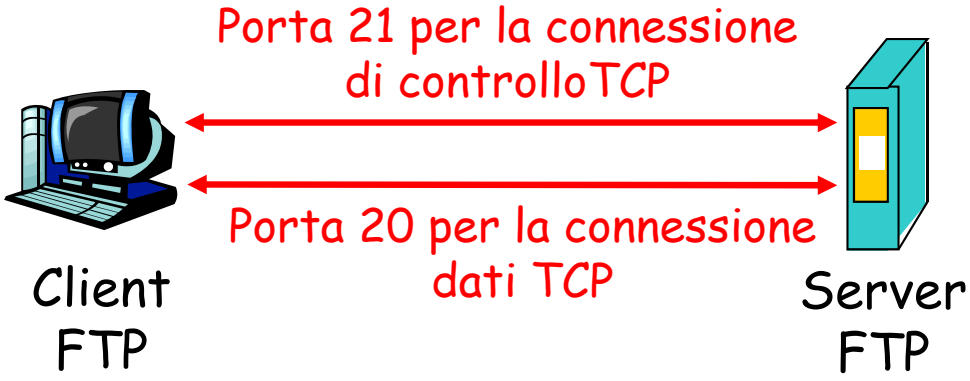
FTP: file transfer protocol



- **Trasferimento file a/da un host remoto**
- **Modello client/server**
  - *client:* il lato che inizia il trasferimento (a/da un host remoto)
  - *server:* host remoto
- **ftp: RFC 959**
- **server ftp: porta 21**

FTP: connessione di controllo, connessione dati

- Il client FTP contatta il server FTP alla porta 21, specificando TCP come protocollo di trasporto
- Il client ottiene l'autorizzazione sulla connessione di controllo
- Il client cambia la directory remota inviando i comandi sulla connessione di controllo
- Quando il server riceve un comando per trasferire un file, apre una connessione dati TCP con il client
- Dopo il trasferimento di un file, il server chiude la connessione



- Il server apre una seconda connessione dati TCP per trasferire un altro file.
- Connessione di controllo: "fuori banda" (*out of band*)
- Il server FTP mantiene lo "stato": directory corrente, autenticazione precedente

Comandi e risposte FTP

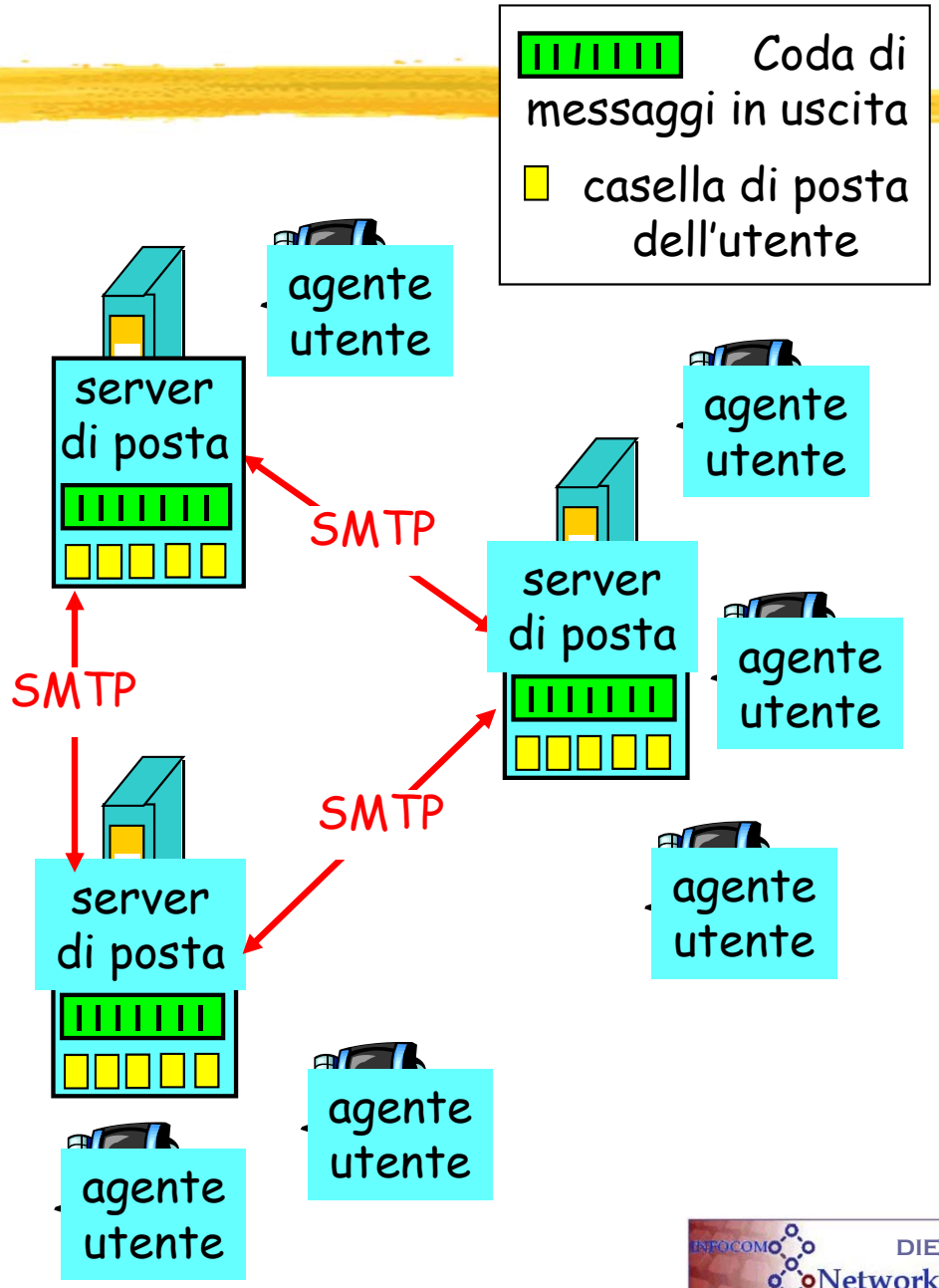
- **Comandi comuni**
  - Inviati come testo ASCII sulla connessione di controllo
  - USER username
  - PASS password
  - LIST elenca i file della directory corrente
  - RETR filename recupera (get) un file dalla directory corrente
  - STOR filename memorizza (put) un file nell'host remoto
- **Codici di ritorno comuni**
  - Codice di stato ed espressione (come in HTTP)
  - 331 Username OK, password required
  - 125 data connection already open; transfer starting
  - 425 Can't open data connection
  - 452 Error writing file

Strato di applicazione

- Principi delle applicazioni di rete
- Web e HTTP
- File Transfer Protocol (FTP)
- Posta elettronica
  - SMTP, POP3, IMAP
- DNS
- Applicazioni P2P

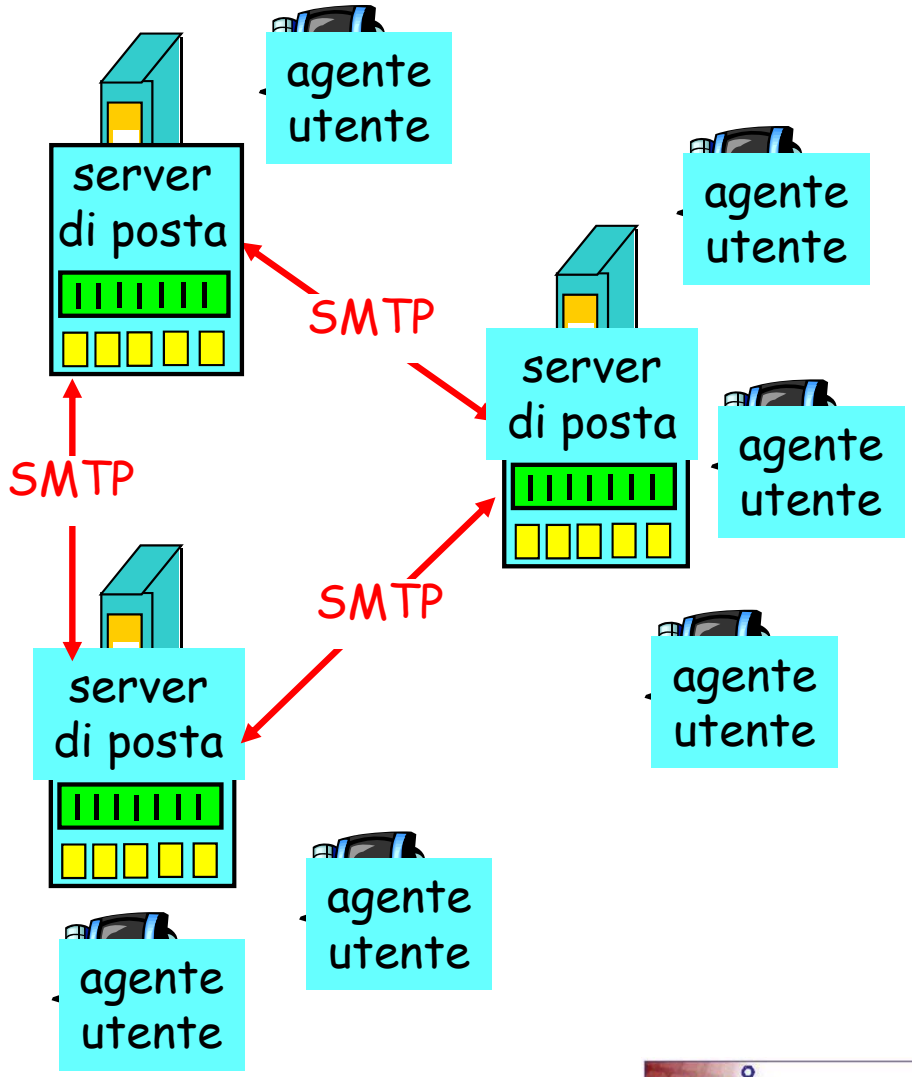
Posta elettronica

- **Tre componenti principali**
  - agente utente
  - server di posta
  - simple mail transfer protocol: SMTP
- **Agente utente**
  - detto anche "mail reader"
  - composizione, editing, lettura dei messaggi di posta elettronica
  - esempi: Outlook, Thunderbird
  - i messaggi in uscita o in arrivo sono memorizzati sul server



Posta elettronica: server di posta

- **Server di posta**
  - Casella di posta (*mailbox*) contiene i messaggi in arrivo per l'utente
  - Coda di messaggi da trasmettere
  - Protocollo SMTP tra i server di posta per inviare messaggi di posta elettronica tra server
    - client: server di posta trasmittente
    - "server": server di posta ricevente

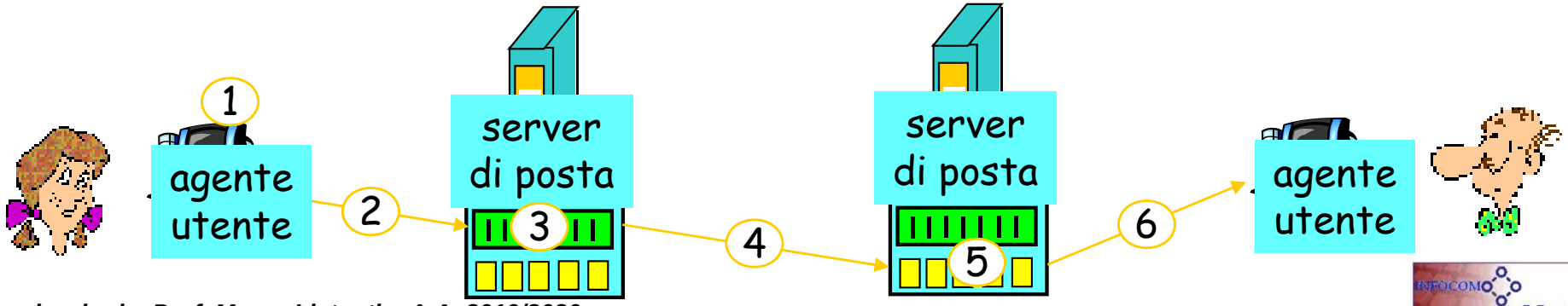


Posta elettronica: SMTP [RFC 5321]

- usa TCP per trasferire in modo affidabile i messaggi di posta elettronica dal client al server, porta 25
- trasferimento diretto: il server trasmittente al server ricevente
- tre fasi per il trasferimento
  - handshaking
  - trasferimento di messaggi
  - Chiusura connessione TCP
- interazione comando/risposta
  - comandi: testo ASCII
  - risposta: codice di stato ed espressione
- i messaggi devono essere nel formato ASCII a 7 bit

Scenario
Alice invia un messaggio a Roberto

1) Alice usa il suo agente utente per comporre il messaggio da inviare "a" rob@someschool.edu
2) L'agente utente di Alice invia un messaggio al server di posta di Alice; il messaggio è posto nella coda di messaggi
3) Il lato client di SMTP apre una connessione TCP con il server di posta di Roberto
4) Il client SMTP invia il messaggio di Alice sulla connessione TCP
5) Il server di posta di Roberto pone il messaggio nella casella di posta di Roberto
6) Roberto usa il suo agente utente per leggere il messaggio



SMTP: note finali

- SMTP usa connessioni persistenti
- SMTP richiede che il messaggio (intestazione e corpo) sia nel formato ASCII a 7 bit
- Il server SMTP usa CRLF.CRLF per determinare la fine del messaggio

- Confronto con HTTP:
- HTTP: pull
  - gli utenti accedono al server per effettuare il download dei contenuti
- SMTP: push
  - gli utenti collocano i messaggi sul server
- Entrambi hanno un'interazione comando/risposta, codici di stato
- HTTP: ciascun oggetto è incapsulato nel suo messaggio di risposta
- SMTP: più oggetti vengono trasmessi in un unico messaggio

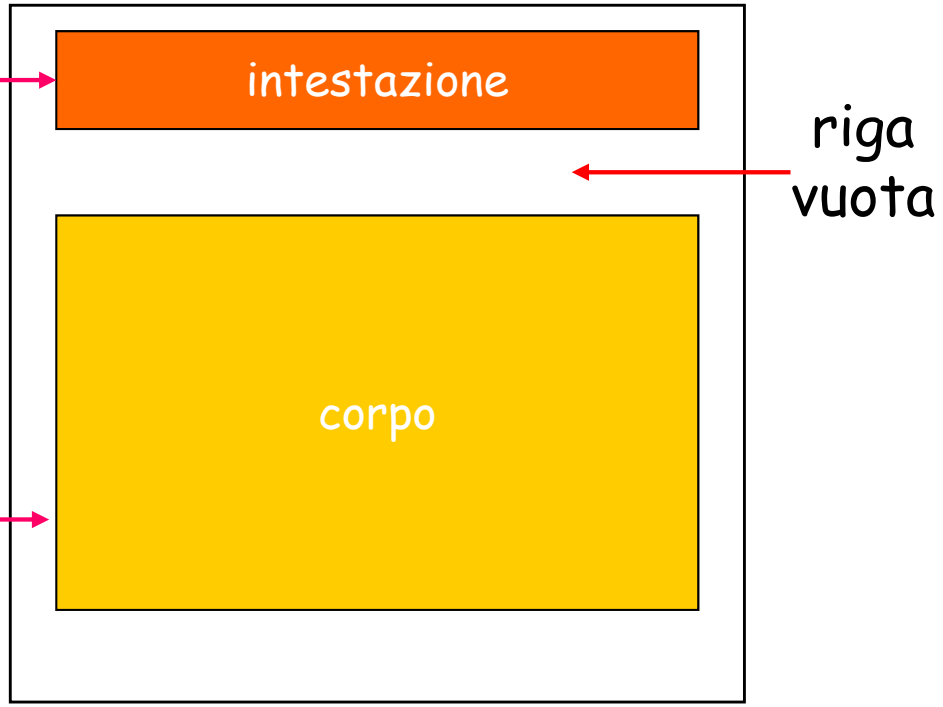
Formato dei messaggi di posta elettronica

SMTP: protocollo per scambiare messaggi di posta elettronica

RFC 822: standard per il formato dei messaggi di testo:

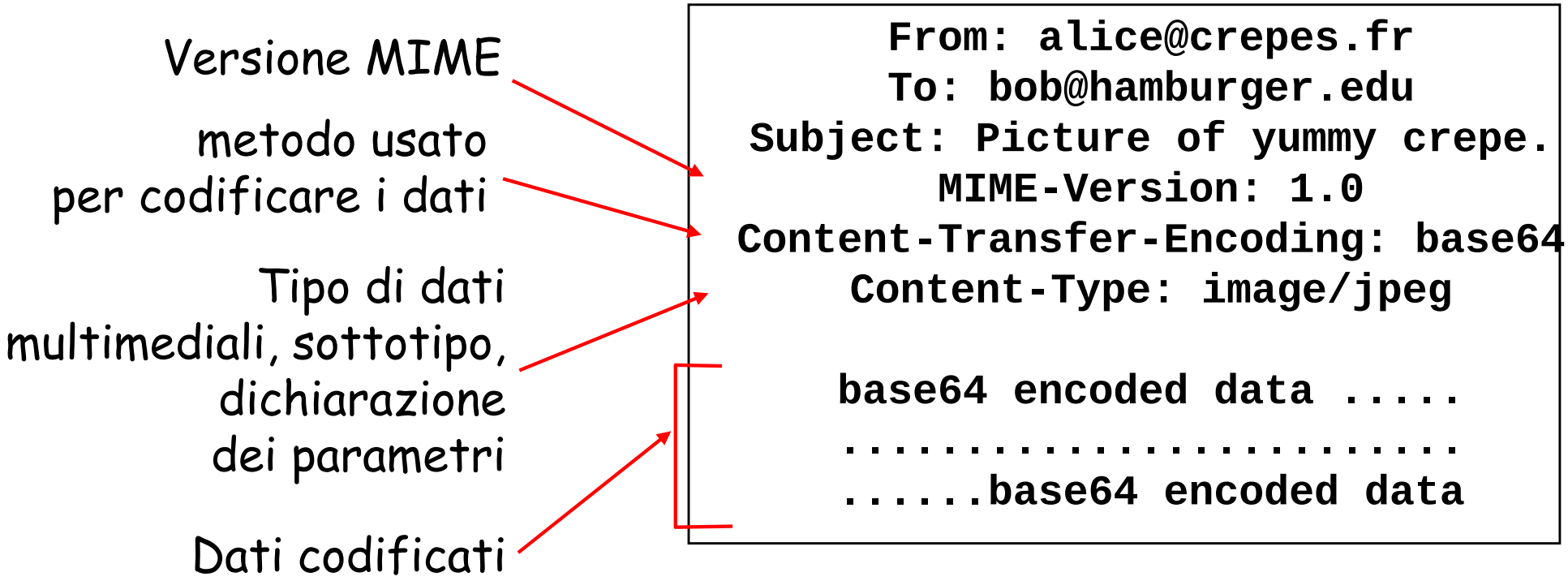
- **Righe di intestazione, per esempio**
  - To/A:
  - From/Da:
  - Subject/Oggetto:

  differenti dai comandi SMTP!
- **corpo**
  - il "messaggio", soltanto caratteri ASCII

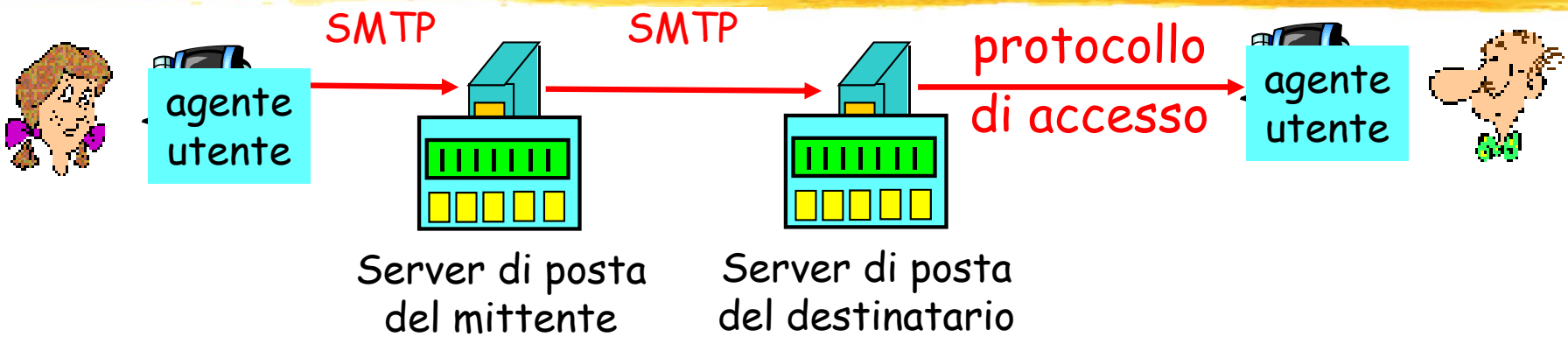


Formato del messaggio: estensioni di messaggi multimediali

- MIME: estensioni di messaggi di posta multimediali, RFC 2045, 2056
- Alcune righe aggiuntive nell'intestazione dei messaggi dichiarano il tipo di contenuto MIME



Protocolli di accesso alla posta



- **SMTP: consegna/memorizzazione sul server del destinatario**
- **Protocollo di accesso alla posta: ottenere i messaggi dal server**
  - POP: Post Office Protocol [RFC 1939]
    - autorizzazione (agente <--> server) e download
  - IMAP: Internet Mail Access Protocol [RFC 3501]
    - più funzioni (più complesse)
    - manipolazione di messaggi memorizzati sul server
  - HTTP: gmail, Hotmail , Yahoo! Mail, ecc.

Protocollo POP3

- **Fase di autorizzazione**
- Comandi del client:
  - `user`: dichiara il nome dell'utente
  - `pass`: password
- Risposte del server
  - `+OK`
  - `-ERR`
- **Fase di transazione**
- client:
- `list`: elenca i numeri dei messaggi
- `retr`: ottiene i messaggi in base al numero
- `dele`: cancella il messaggio
- `quit`

```
S: +OK POP3 server ready
C: user rob
S: +OK
C: pass hungry
S: +OK user successfully logged on
```

```
C: list
S: 1 498
S: 2 912
S: .
C: retr 1
S: <message 1 contents>
S: .
C: dele 1
C: retr 2
S: <message 1 contents>
S: .
C: dele 2
C: quit
S: +OK POP3 server signing off
```

POP3 (altro) e IMAP

Ancora su POP3

- Il precedente esempio usa la modalità "scarica e cancella"
- Roberto non può rileggere le e-mail se cambia client
- Modalità "scarica e mantieni": copia i messaggi su più client
- POP3 è un protocollo senza stato tra le varie sessioni

IMAP [RFC 3501]

- Mantiene tutti i messaggi in un unico posto: il server
- Consente all'utente di organizzare i messaggi in cartelle
- IMAP conserva lo stato dell'utente tra le varie sessioni:
  - I nomi delle cartelle e l'associazione tra identificatori dei messaggi e nomi delle cartelle

Strato di applicazione

- Principi delle applicazioni di rete
- Web e HTTP
- File Transfer Protocol (FTP)
- Posta elettronica
  - SMTP, POP3, IMAP
- DNS
- Applicazioni P2P

DNS: Domain Name System

- **Persone: molti identificatori:**
  - nome, codice fiscale, numero della carta d'identità
- **Host e router di Internet:**
  - indirizzo IP (32 bit) - usato per indirizzare i datagrammi
  - "nome", ad esempio, www.yahoo.com
- **D: Come associare un indirizzo IP a un nome?**

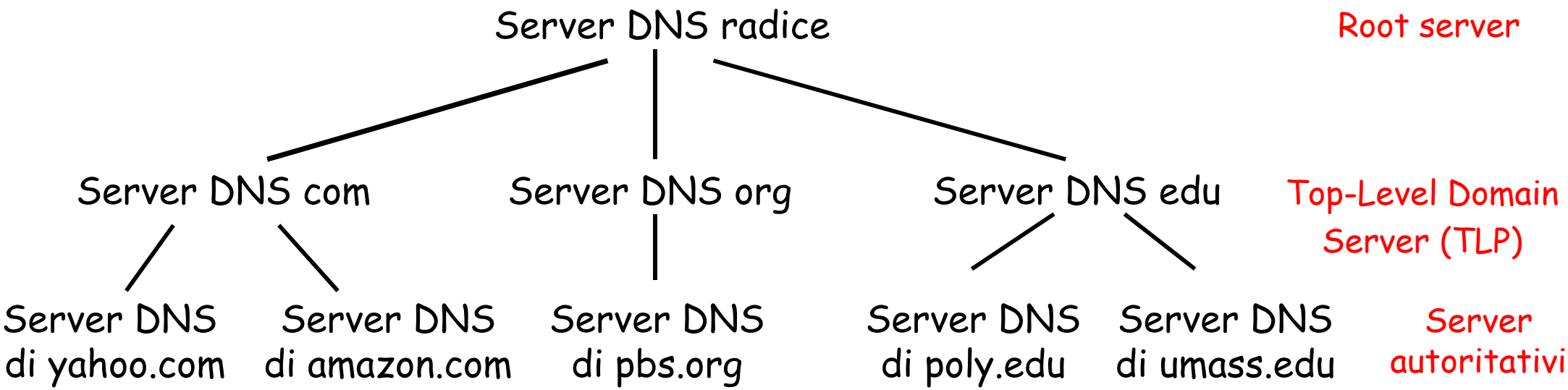
- **Domain Name System**
- ***Database distribuito***
  - implementato in una gerarchia di *server DNS*
- ***Protocollo a livello di applicazione***
  - consente agli host, ai router e ai server DNS di comunicare per *risolvere* i nomi (tradurre nomi/indirizzi)
  - Si noti: funzioni critiche di Internet implementate come protocollo a livello di applicazione
  - complessità nelle parti periferiche della rete
  - Ritardo addizionale

DNS

- **Servizi DNS**
- **Traduzione degli hostname in indirizzi IP**
- **Host aliasing**
  - un host può avere più nomi: nome canonico + alias
  - Il DNS traduce un alias nel nome canonico
  - Mail server aliasing
- **Distribuzione locale**
  - server web replicati: insieme di indirizzi IP per un nome canonico

- **Perché non centralizzare DNS?**
- **singolo punto di guasto**
- **volume di traffico**
- **database centralizzato distante**
- **manutenzione**
- **Un database centralizzato su un singolo server DNS non è *scalabile* !**

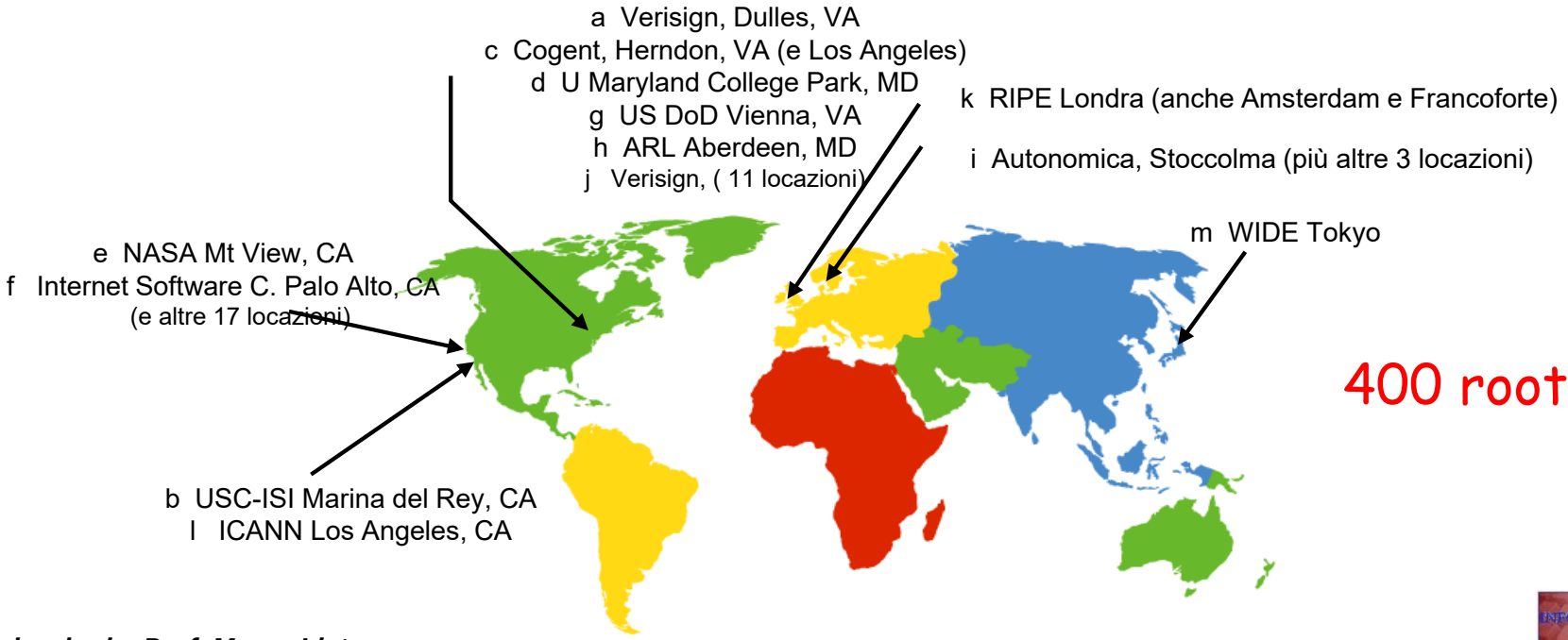
Database distribuiti e gerarchici



- **Il client vuole ottenere l'indirizzo IP del nome www.amazon.com**
  - Il client interroga il root server per trovare il server DNS .com
  - Il client interroga il server DNS .com per ottenere il server DNS amazon.com
  - Il client interroga il server DNS amazon.com per ottenere l'indirizzo IP di www.amazon.com

DNS: root server

- **Sono contattati da un server DNS locale che non può tradurre il nome**
  - contatta un server DNS autorizzato se non conosce la mappatura
  - ottiene la mappatura
  - restituisce la mappatura al server DNS locale



400 root server

Server TLD e server autoritativi

- **Server TLD (Top-Level Domain server)**
  - si occupano dei domini com, org, net, edu, ecc. e di tutti i domini locali di alto livello, quali uk, fr, ca e jp.
    - Network Solutions gestisce i server TLD per il dominio com
    - Educause gestisce quelli per il dominio edu

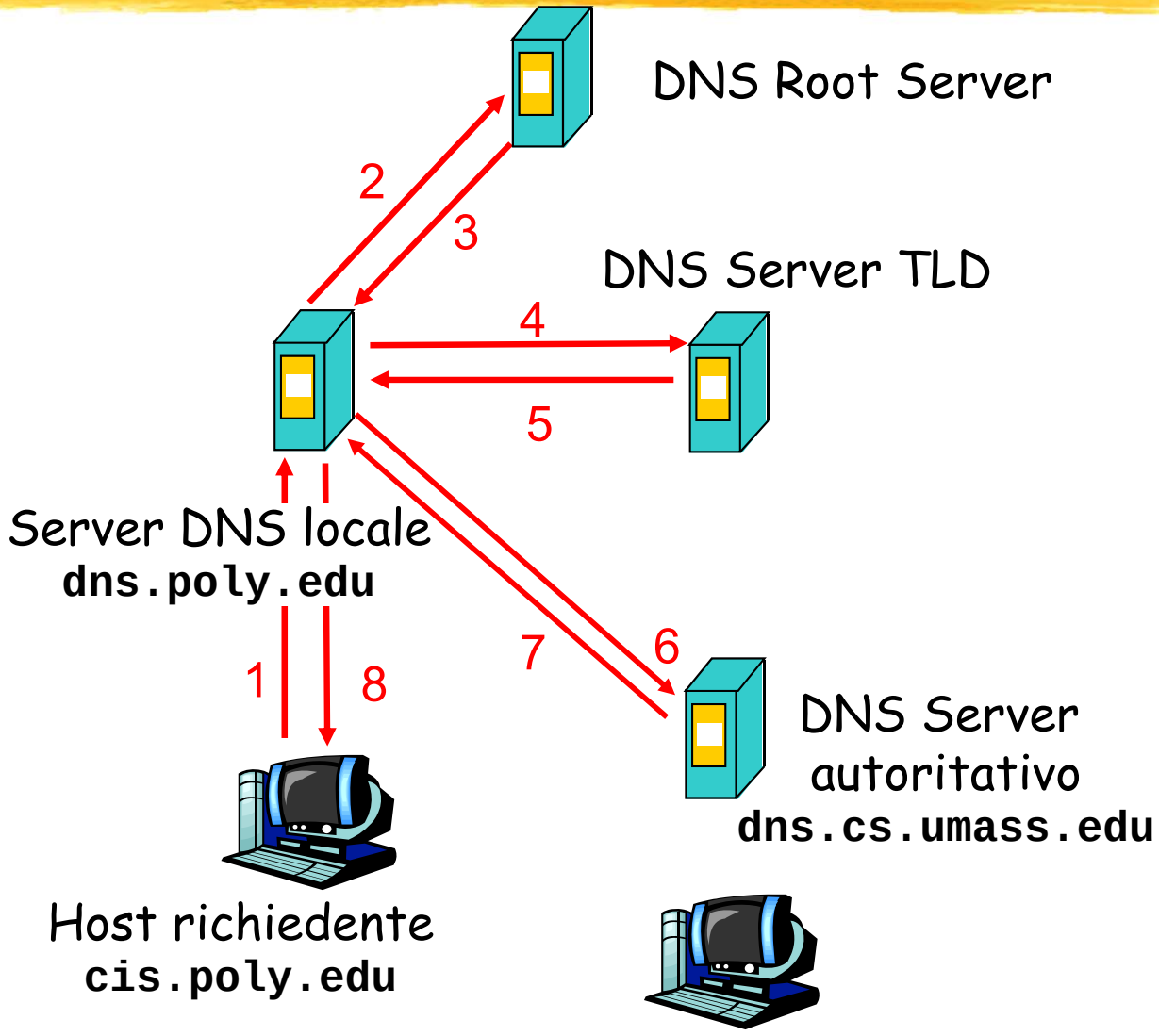
- **Server Autoritativi (*authoritative server*)**
  - ogni organizzazione dotata di host Internet pubblicamente accessibili (quali i server web e i server di posta) deve fornire i record DNS di pubblico dominio che mappano i nomi di tali host in indirizzi IP.
    - possono essere gestiti dall'organizzazione o dal service provider

Server DNS locale

- **Ciascun ISP (università, società, ISP residenziale) ha un server DNS locale**
  - detto anche "default name server"
- **Non appartiene alla gerarchia dei server**
- **Quando un host effettua una richiesta DNS, la query viene inviata al suo server DNS locale**
  - il server DNS locale opera da proxy e inoltra la query in una gerarchia di server DNS

Esempio (1)

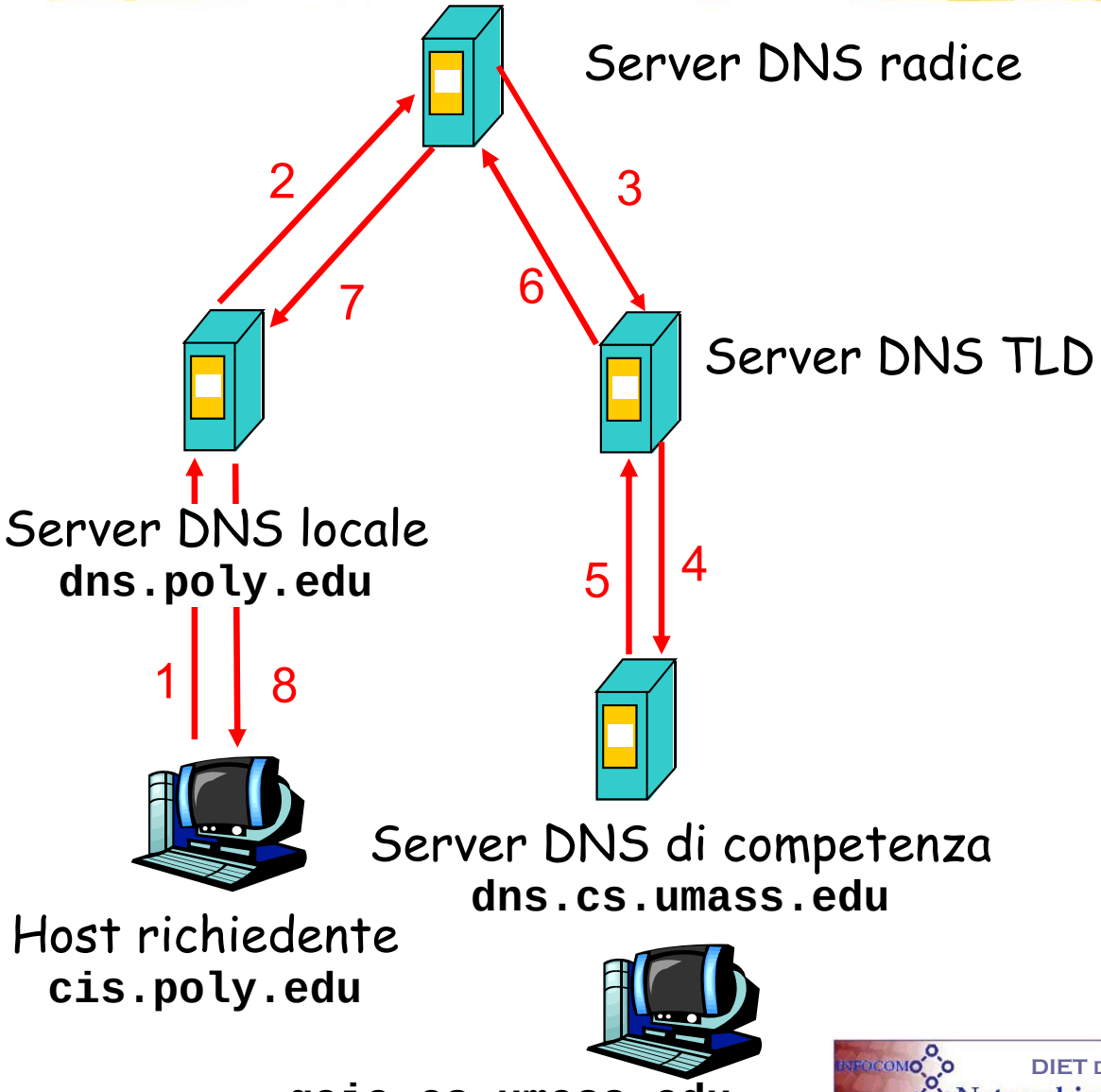
- L'host cis.poly.edu richiede l'indirizzo IP di gaia.cs.umass.edu
- **Query ierative**
  - Il server contattato risponde con il nome del server da contattare
  - "Io non conosco questo nome, ma puoi chiederlo a questo server".



Esempio (2)

- **Query ricorsiva**
  - Affida il compito di tradurre il nome al server DNS contattato

- Il traffico addizionale ed il delay introdotti dalla risoluzione dei nomi può essere diminuito mediante l'uso di cache



DNS Caching

- **Una volta che un server DNS conosce la mappatura «nome, indirizzo», la memorizza nella propria memoria *cache***
  - Le informazioni nella cache vengono cancellate dopo un certo periodo di tempo (tipicamente 48 ore)
  - Normalmente un server DNS locale memorizza nella cache gli indirizzi IP dei server TLD
    - quindi i server DNS radice non vengono visitati spesso
- **I meccanismi di aggiornamento/notifica sono definiti dall'IETF**
  - RFC 2136
  - http://www.ietf.org/html.charters/dnsind-charter.html

DNS Resource Record (RR)

- DNS è un database distribuito che memorizza i Resource Record (RR)

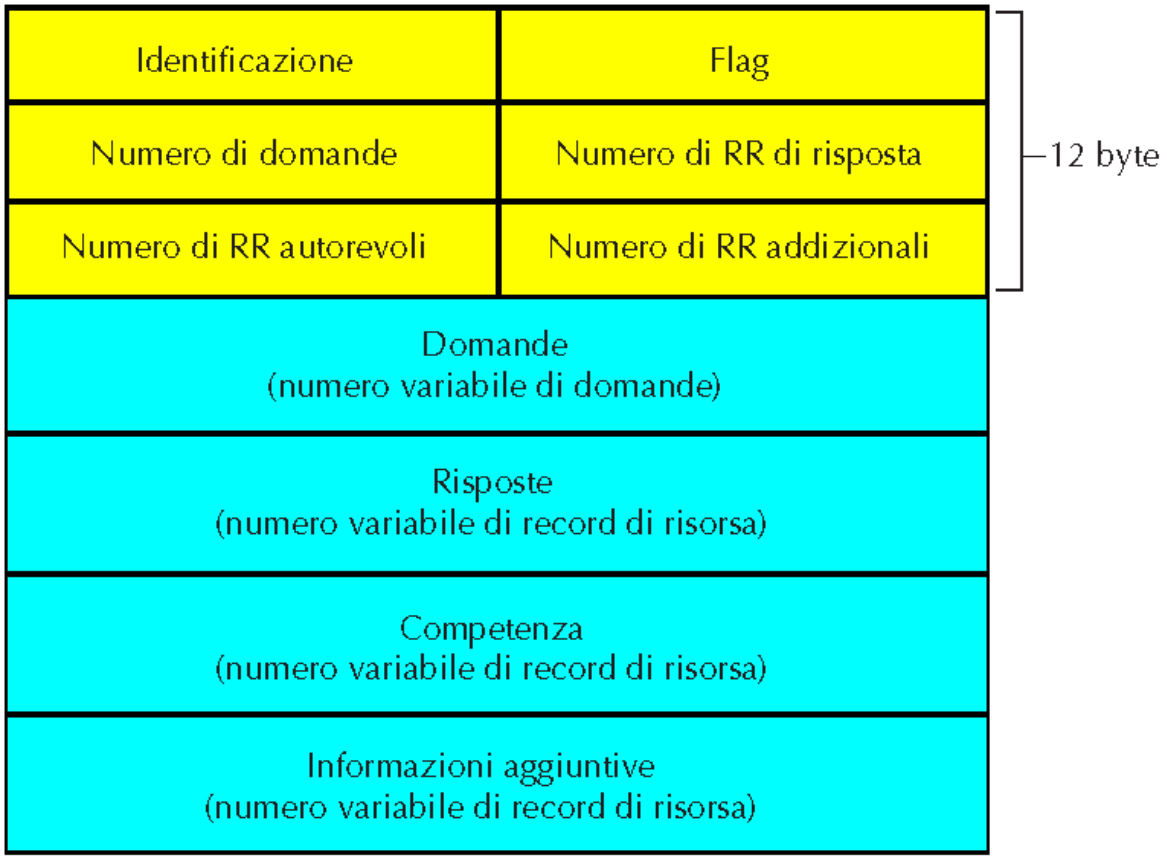
Formato RR: **(name, value, type, ttl)**

- Type = A
  - **name** è il nome dell'host
  - **value** è l'indirizzo IP
- Type=NS
  - **name** è il dominio (ad esempio foo.com)
  - **value** è il nome dell'host del server autoritativo di questo dominio
- Type = CNAME
  - **name** è il nome alias di in server
  - **www.ibm.com** è in realtà servereast.backup2.ibm.com
  - **value** è il nome canonico
- Type = MX
  - **value** è il nome del server di posta associato a **name**

Formato messaggi DNS

- **Protocollo DNS: domande (query) e messaggi di *risposta*, entrambi con lo stesso *formato***

- Intestazione del messaggio
  - Identificazione: numero di 16 bit per la domanda; la risposta alla domanda usa lo stesso numero
  - Flag
    - domanda o risposta
    - richiesta di ricorsione
    - ricorsione disponibile



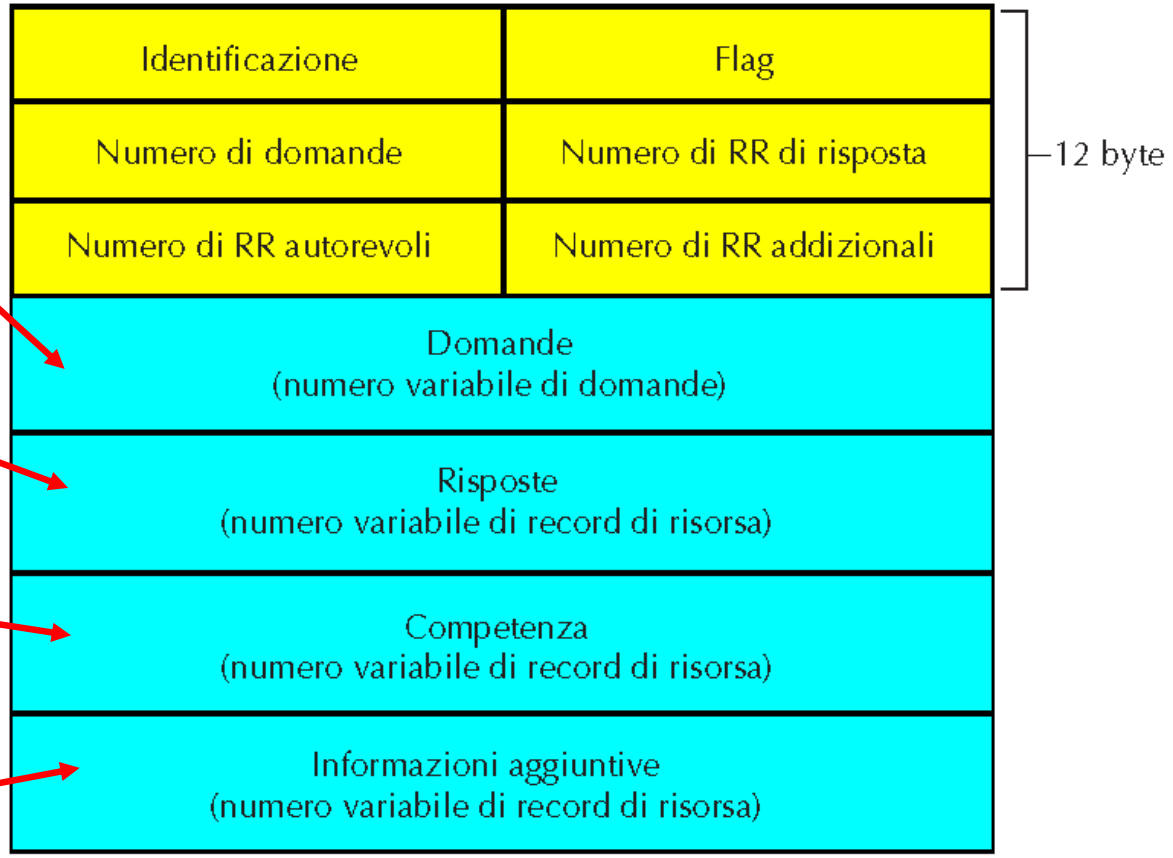
Formato messaggi DNS

Campi per
il nome richiesto
e il tipo di domanda

RR nella risposta
alla domanda

Record per
i server autoritativi

Informazioni extra



Inserire record nel database DNS

- **Esempio**
  - si è avviata una nuova società "Network Utopia"
  - Si registra il nome `networkuptopia.com` presso registrar (ad esempio, Network Solutions)
    - Forniamo a registrar i nomi e gli indirizzi IP dei server DNS di competenza (primario e secondario)
    - Registrar inserisce due RR nel server TLD com:

  - `(networkutopia.com, dns1.networkutopia.com, NS)`
  - `(dns1.networkutopia.com, 212.212.212.1, A)`

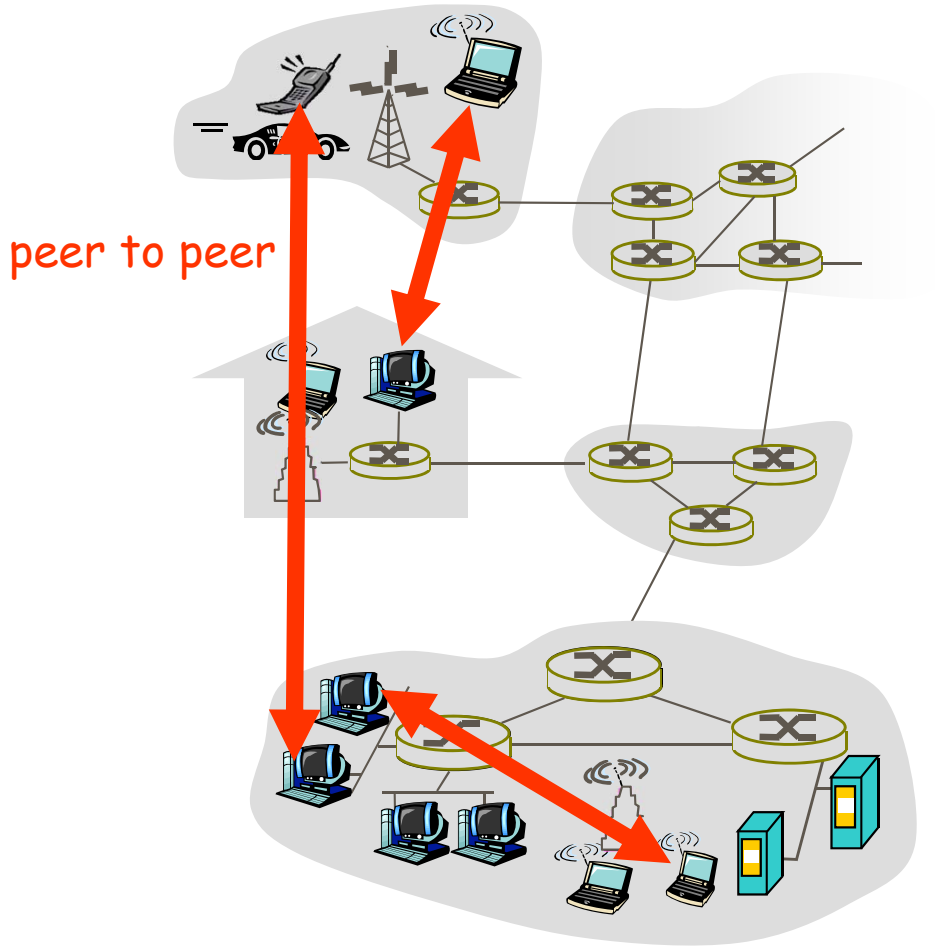
- **Si inseriscono nel server autoritativo un record tipo A per www.networkuptopia.com e un record tipo MX per networkutopia.com**
- **In che modo gli utenti otterranno l'indirizzo IP del nostro sito web?**

Strato di applicazione

- Principi delle applicazioni di rete
- Web e HTTP
- File Transfer Protocol (FTP)
- Posta elettronica
  - SMTP, POP3, IMAP
- DNS
- Applicazioni P2P

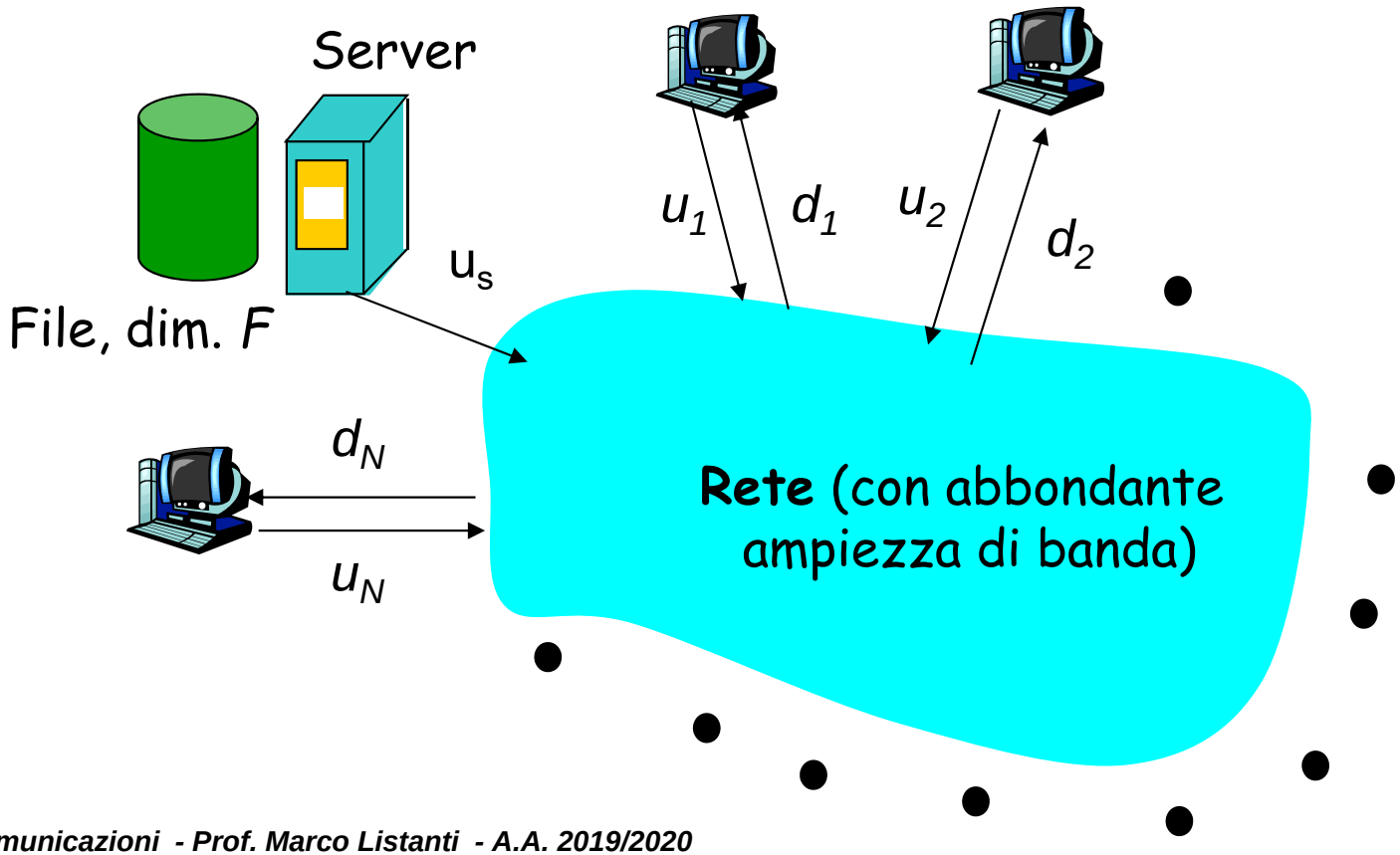
Architettura P2P pura

- non esiste un server dei contenuti sempre attivo
- coppie arbitrarie di host (peer) comunicano direttamente tra loro
- i peer non devono necessariamente essere sempre attivi, e possono cambiare indirizzo IP
- Tre argomenti chiave:
  - Distribuzione di file
  - Ricerca informazioni
  - Caso di studio: Skype



Tempo di distribuzione di un file: Server-Client vs P2P

- Domanda : Quanto tempo ci vuole per distribuire file da un server a *N peer*?



u_s: banda in upload del collegamento di accesso del server (bit/s)

u_i: banda in upload del collegamento di accesso dell'*i*-esimo peer (bit/s)

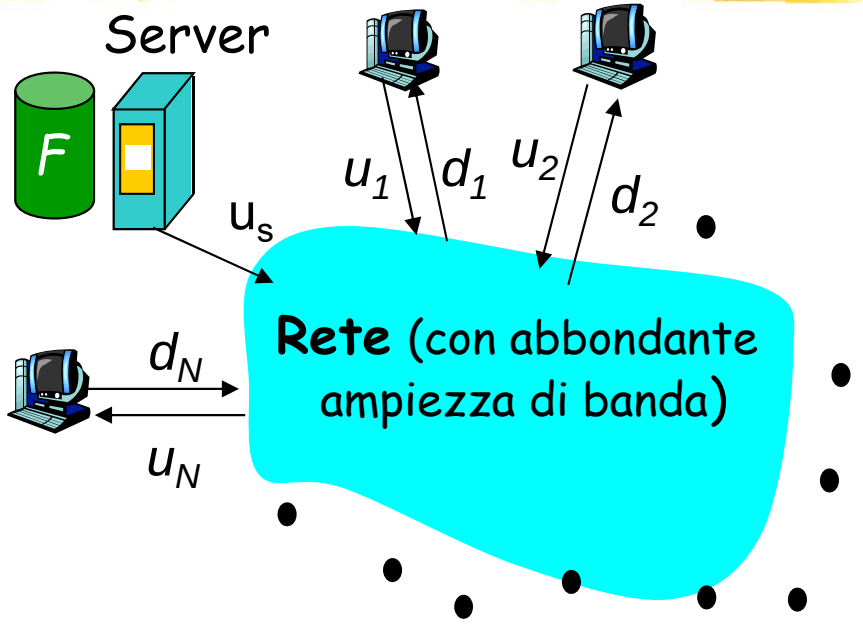
d_i: banda in download del collegamento di accesso dell'*i*-esimo peer (bit/s)

F: dimensione del file (bit)

N: numero dei peer

Tempo di distribuzione di un file: Architettura client-server

- Il server invia in sequenza N copie:
  - $Tempo = NF/u_s$
- Il client i impiega il tempo F/d_i per scaricare



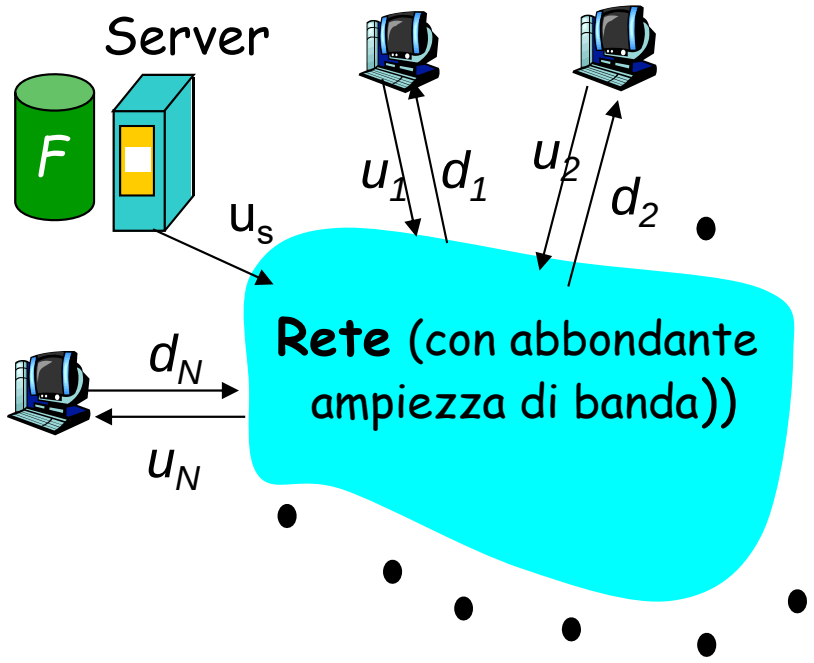
D_{cs} = Tempo di distribuzione di un file di dimensione F a N client usando l'approccio client/server

$$D_{cs} = \max \{ NF/u_s, F/\min(d_i) \}$$

aumenta linearmente con N peer

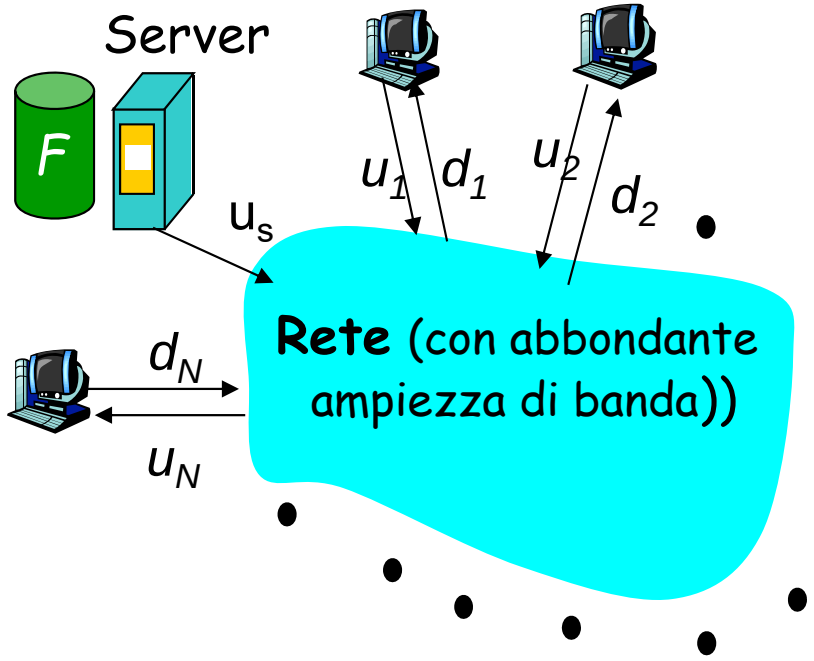
Tempo di distribuzione di un file: Architettura P2P (1)

- Il server invio solo una volta il file
- I peer redistribuiscono porzioni (chunk) del file verso gli altri pee
- I peer utilizzano la propria banda di upload



Tempo di distribuzione di un file: Architettura P2P (2)

- il server deve inviare una copia nel tempo F/u_s
- il client con la banda di download d_{min} più bassa riceve il file nel tempo F/d_{min}
- Devono essere scaricati NF bit
- La banda massima possible di upload dei peer è: $u_s + \Sigma u_i$
- Ipotesi: pien utilizzazione dell banda di uoload
- Il tempo minimo D_{P2P} di distribuzione del file tra gli N peer è dato da



$$D_{P2P} = \max \{ F/u_s, F/d_{min}) , NF/(u_s + \Sigma u_i) \}$$

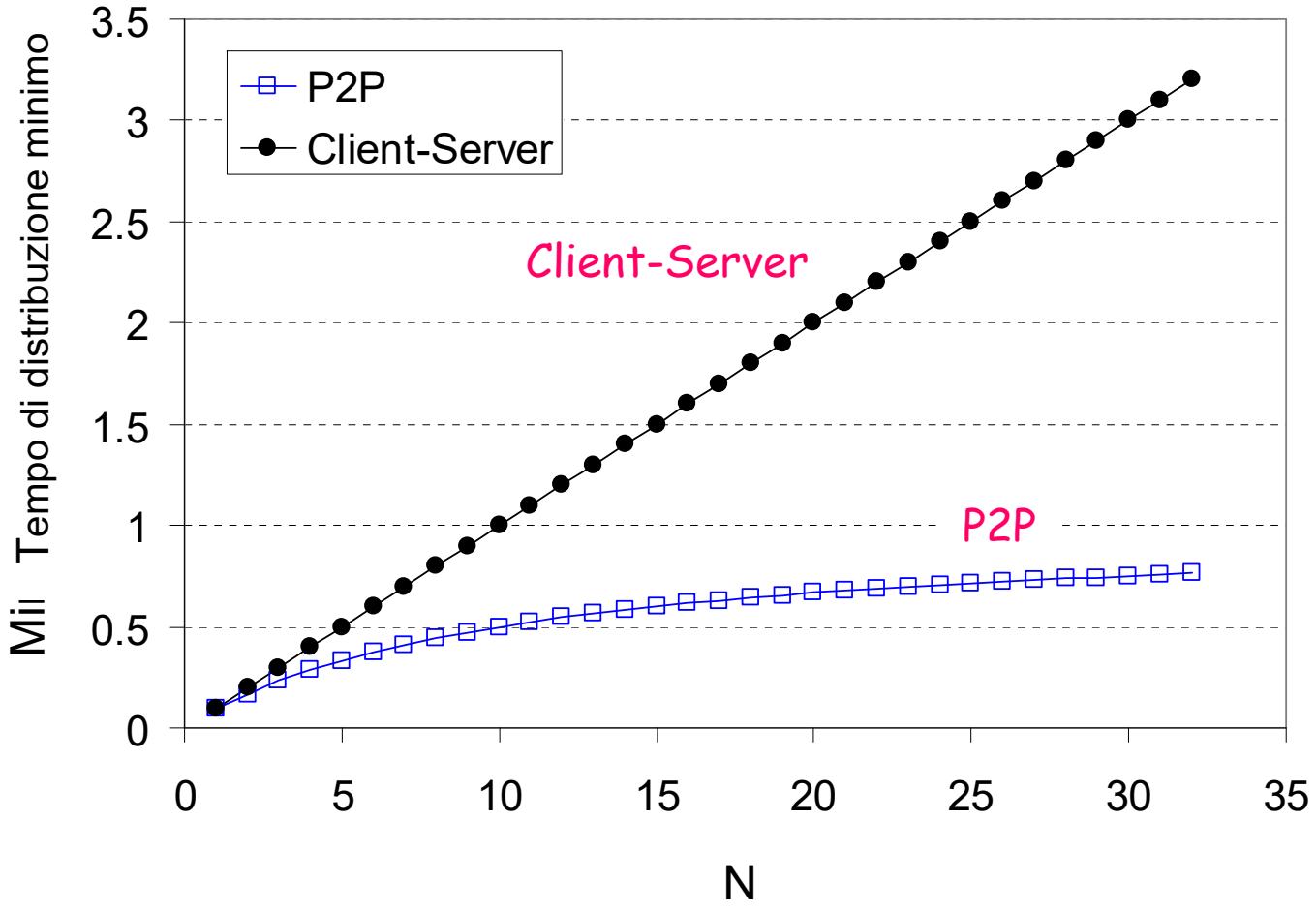
Il bottleneck è il server

Il bottleneck è il peer che riceve il file

Il bottleneck è la distribuzione tra i peer

Tempo di distribuzione di un file: Server-Client vs P2P

Tasso di upload del client F/u = 1 ora, u_s = 10 u, $d_{min} \geq u_s$



P2P: ricerca di informazioni

- Indice nei sistemi P2P
  - corrispondenza tra le informazioni e la loro posizione negli host

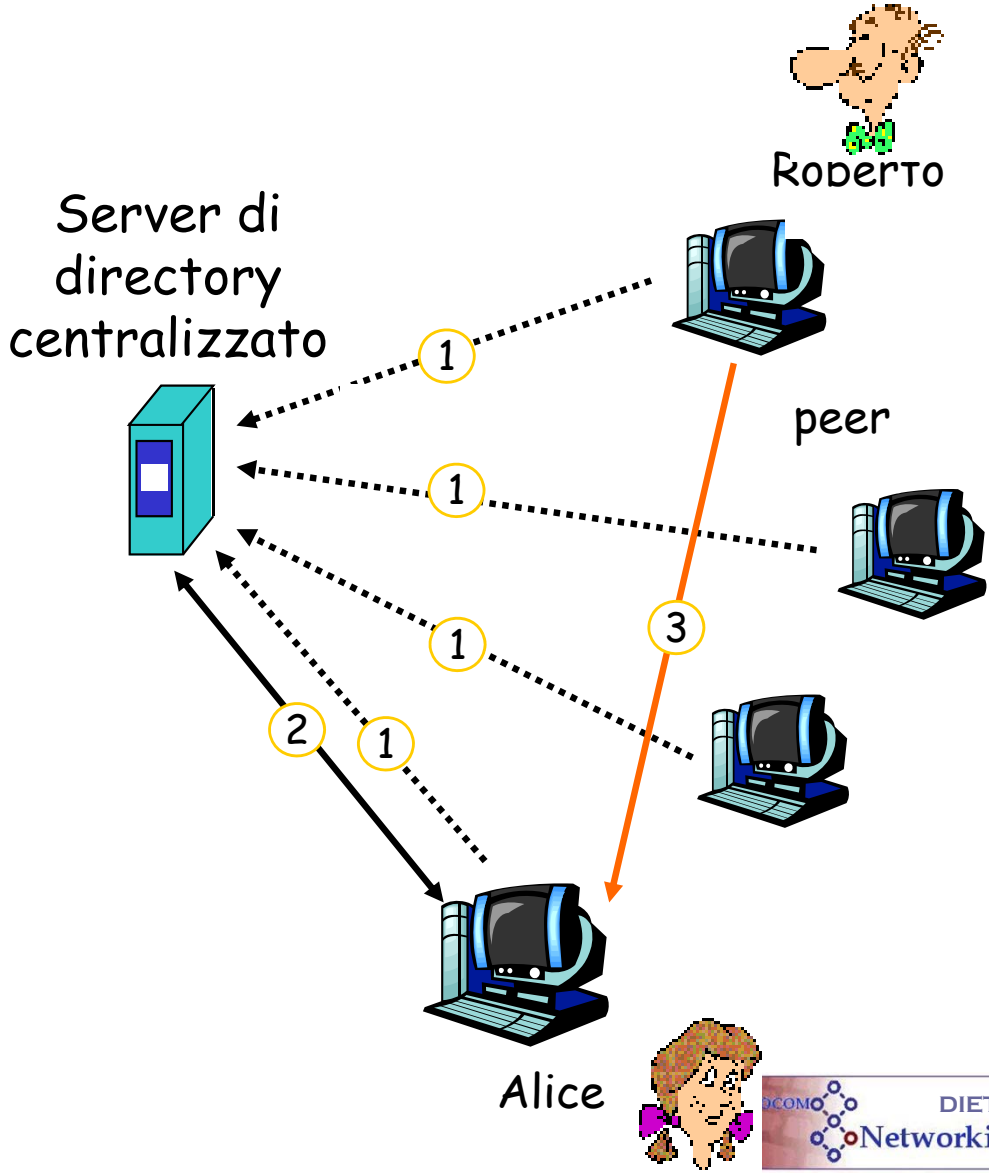
- File sharing
  - L'indice tiene traccia dinamicamente della posizione dei file che i peer condividono.
  - I peer comunicano all'indice ciò che possiedono.
  - I peer consultano l'indice per determinare dove trovare i file.

- Messaggeria istantanea
  - L'indice crea la corrispondenza tra utenti e posizione.
  - Quando l'utente lancia l'applicazione, informa l'indice della sua posizione
  - I peer consultano l'indice per determinare l'indirizzo IP dell'utente.

P2P: directory centralizzata

- Progetto originale di "Napster"

1) quando il peer si collega, fornisce al server centrale:
   - Il proprio indirizzo IP
   - I contenuti posseduti
2) Alice cerca la canzone "Hey Jude" ed il server risponde con la lista dei peer in cui si trova il contenuto richiesto
3) Alice richiede il file a Roberto



P2P: problemi con la directory centralizzata

- Unico punto di guasto
- Collo di bottiglia per le prestazioni
- Violazione del diritto d'autore

Il trasferimento dei file è distribuito, ma il processo di localizzazione è fortemente centralizzato

Bit Torrent (1)

- Il più diffuso protocollo P2P
- Il download di un file è chiamato torrent
- Per ogni torrent esiste un nodo centrale chiamato tracker
  - Il nodo centrale non contiene la lista dei contenuti dei peer ma solo gli indirizzi dei peer apparteneti al torrent
- Al momento in cui un peer si aggiunge ad un torrent deve registrarsi sul nodo tracker e ottiene dal tracker la lista dei peer fanno parte del torrent e che il peer può contattare per il download del file
  - La lista contiene normalmente un sottoinsieme dei peer del torrent
- Un peer può aggiungersi o lascare il torrent in qualsiasi momento, ed aggiorna il suo stato periodicamente
  - Il tracker tiene traccia dei peer che fanno parte del torrent
- I peer scaricano da altri peer chunk del file di uguale dimensione (256 kbyte)

Bit Torrent (2)

- **Il nuovo peer tenta di stabilire delle connessioni TCP con i peer della lista**
  - I peer connessi al nuovo peer sono detti neighboring peer
- **Il nuovo peer chiederà periodicamente ai vicini il download dei chunk del file**
  - Normalmente si usa un algoritmo detto rarest first richiesta dei chunk più rari
- **Un peer invia i chunk ai peer con la velocità di upload più elevata**

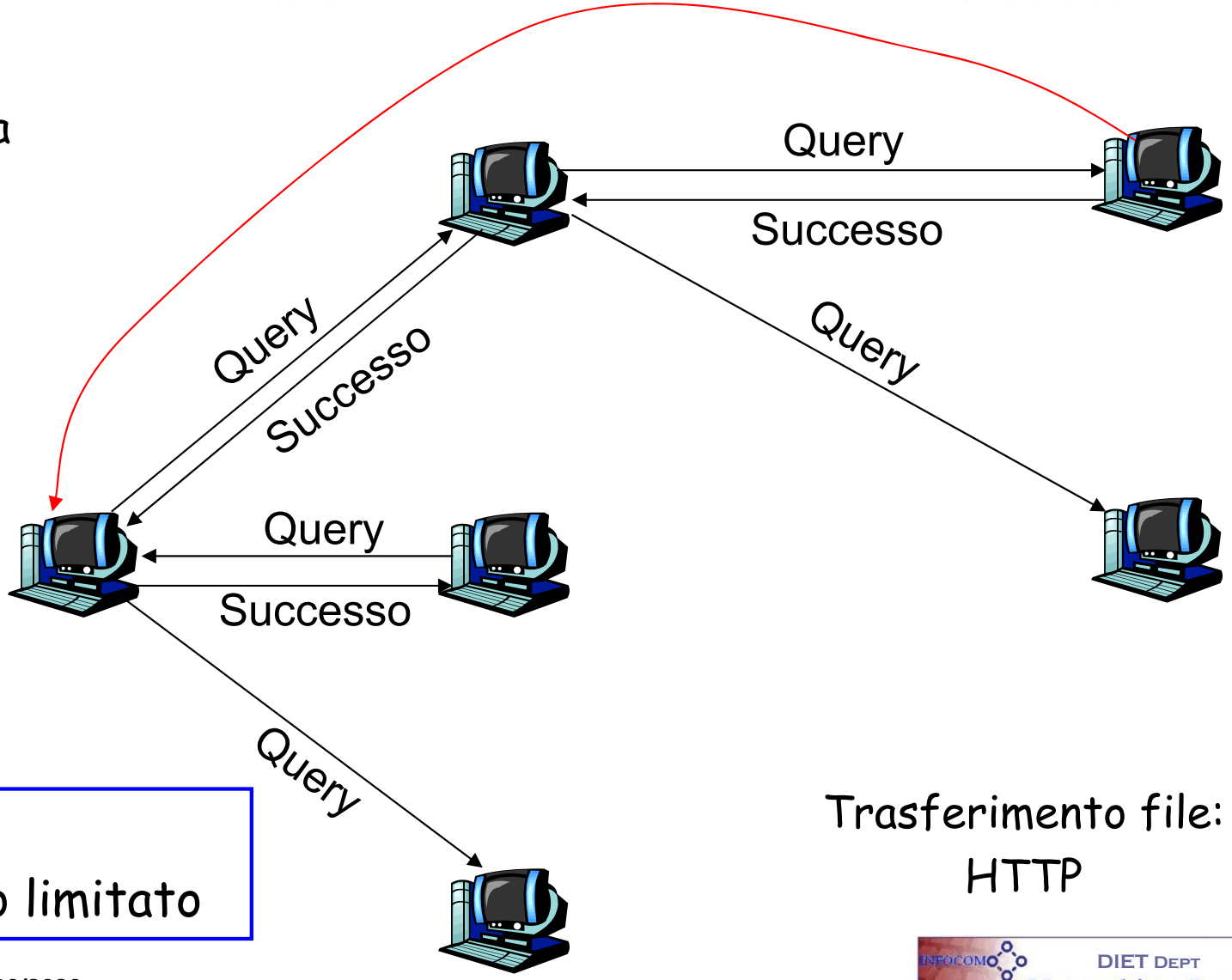
Query flooding

- Completamente distribuito
  - nessun server centrale
- Ciascun peer indicizza i file che rende disponibili per la condivisione (e nessun altro)

- Rete di copertura: grafo
- Arco tra i peer X e Y se c'è una connessione TCP
- Tutti i peer attivi e gli archi formano la rete di copertura
- Un arco è un collegamento virtuale e *non* fisico
- Un dato peer sarà solitamente connesso con meno di 10 peer vicini nella rete di copertura

Query flooding

- Il messaggio di richiesta è trasmesso sulle connessioni TCP esistenti
- Il peer inoltra il messaggio di richiesta
- Il messaggio di successo è trasmesso sul percorso inverso

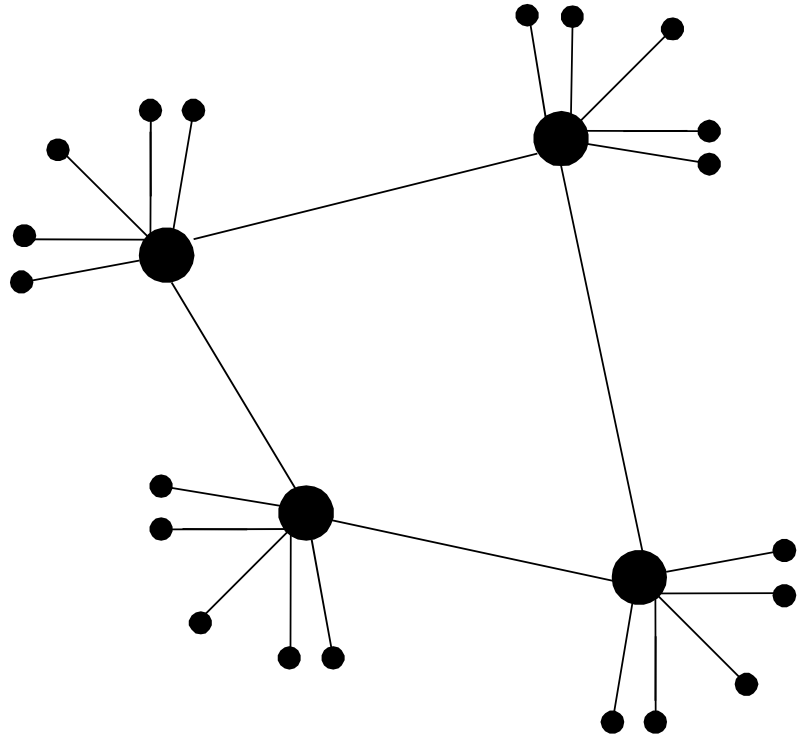


Scalabilità:
query flooding a raggio limitato

Trasferimento file:
HTTP

Copertura gerarchica

- La copertura gerarchica combina le caratteristiche di indice centralizzato e query flooding
- Ogni peer è assegnato a un leader di gruppo
  - Connessione TCP tra peer e il suo leader di gruppo
  - Connessioni TCP tra qualche coppia di leader di gruppo
- Il leader di gruppo tiene traccia del contenuto di tutti i suoi figli.



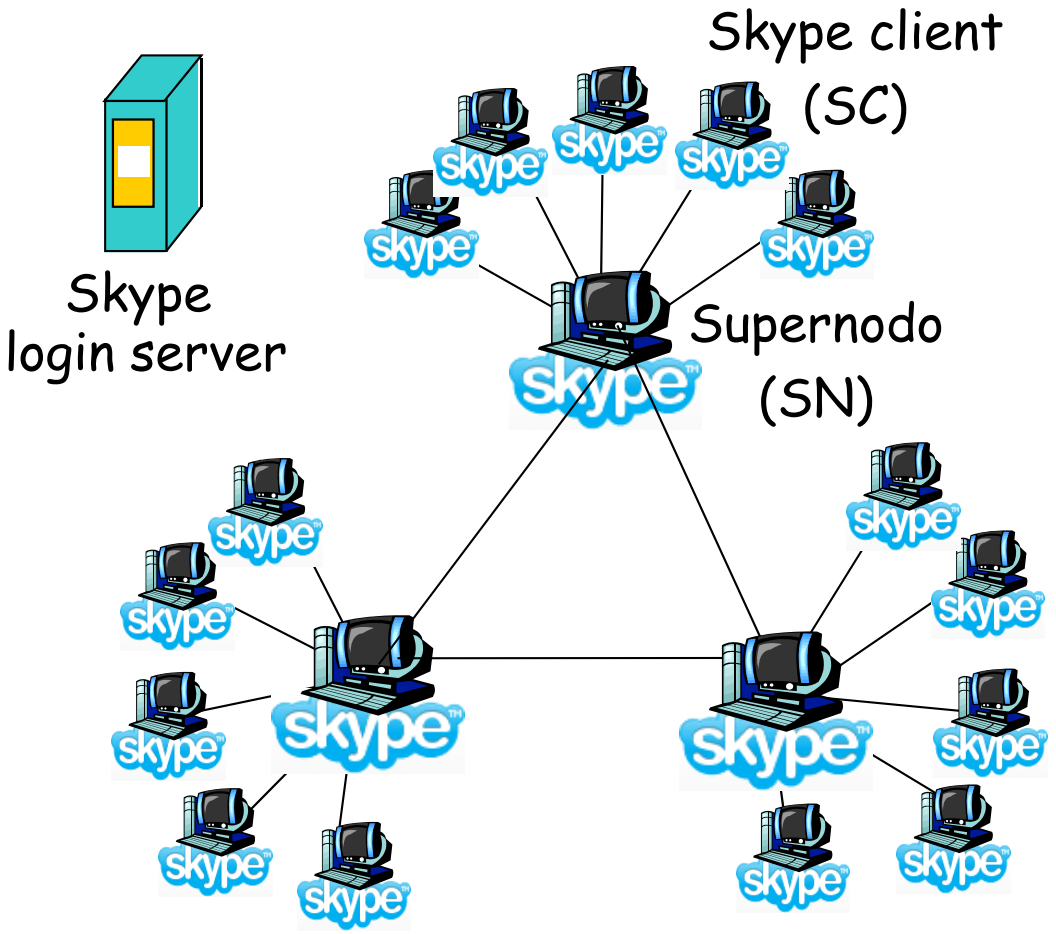
Peer ordinario

Peer leader di gruppo

Relazioni di adiacenza nella rete di copertura

Caso di studio P2P: Skype

- **Intrinsecamente P2P**
  - coppie di utenti comunicano tra loro
- **Protocollo proprietario (dedotto mediante reverse engineering)**
- **Copertura gerarchica con i supernodi**
- **Ogni supernodo memorizza la corrispondenza tra nomi utente e indirizzi IP**



Peer e relay

- **Si pone un problema quando sia Alice che Roberto sono dietro ad un NAT.**
  - NAT evita che un host al di fuori della rete domestica crei una connessione con un host all'interno di questa
- **Soluzione**
  - Usando il supernodo di Alice e Roberto, si sceglie un relay
  - Ciascun peer inizia la sessione con il relay.
  - I peer ora comunicano con NAT attraverso il relay