

Tecnologie per la Comunicazione Aziendale

Flavio De Paoli

Livello applicativo

Obiettivi generali:

- Aspetti concettuali/
implementativi dei
protocolli applicativi
 - Paradigma client
server
 - Modelli dei servizi

Obiettivi specifici:

- Protocolli specifici:
 - http
 - ftp
 - smtp
 - pop
 - dns
 - Programmazione di
applicazioni
 - Uso dei socket

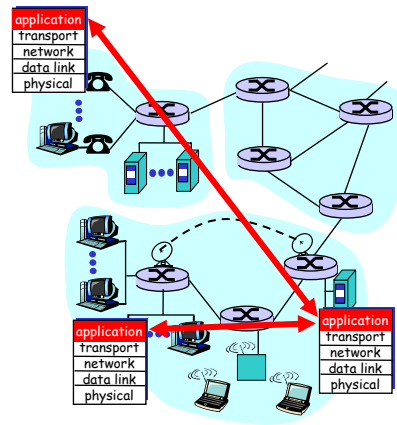
Applicazioni e protocolli applicativi

Applicazione: processi distribuiti in comunicazione

- In esecuzione su host remoti
- Si scambiano messaggi per eseguire l'applicazione
- Es., posta, FTP, WWW

Protocolli applicativi

- Costituiscono una parte di ogni applicazione
- Definiscono il formato dei messaggi scambiati e il loro significato (azioni)
- Usano i servizi degli strati inferiori



2: Application Layer 3

Applicazioni: terminologia essenziale

- Un **processo** è un programma in esecuzione su un host
- Sullo stesso host i processi comunicano mediante meccanismi definiti dal SO.
- Processi in esecuzione su host diversi comunicano mediante meccanismi definiti dal **protocollo dello strato di applicazione (application layer protocol)**
- Un **agente utente (user agent)** è un'interfaccia tra l'utente e l'applicazione di rete.
 - Browser Web
 - E-mail: lettore di posta
 - streaming audio/video: lettore di file audio/video

2: Application Layer 4

Paradigma client-server

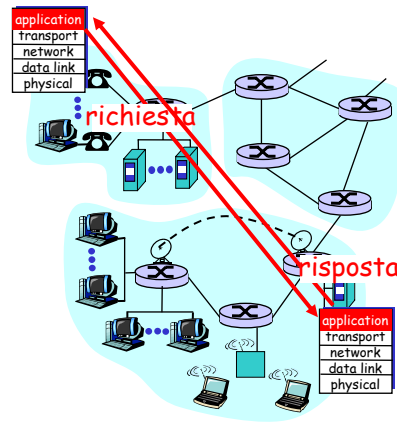
App. di rete tipica consiste di due parti: *client* e *server*

Client:

- Inizia il dialogo col server ("speaks first")
- Di solito richiede un servizio
- Nel caso del Web, il client è integrato nel browser

Server:

- Fornisce il servizio al client, su richiesta
- Es., un Web server invia una pagina Web richiesta, un mail server accede alla casella di posta elettronica



2: Application Layer 5

Protocolli di livello applicativo: servizi dagli strati inferiori e identificazione

API: Application Programming Interface

- Definisce l'interfaccia tra applicazione e strato di trasporto
- Socket: API Internet
 - Due processi (applicazione nel modello client server) comunicano inviando/leggendo dati nel/dal socket

D: come può un processo "identificare" quello con cui intende comunicare?

- **Indirizzo IP** dell'host su cui l'altro processo è in esecuzione
- **Numero di porta (port number)** - permette all'host ricevente di identificare il processo locale destinatario del messaggio

... di più in seguito.

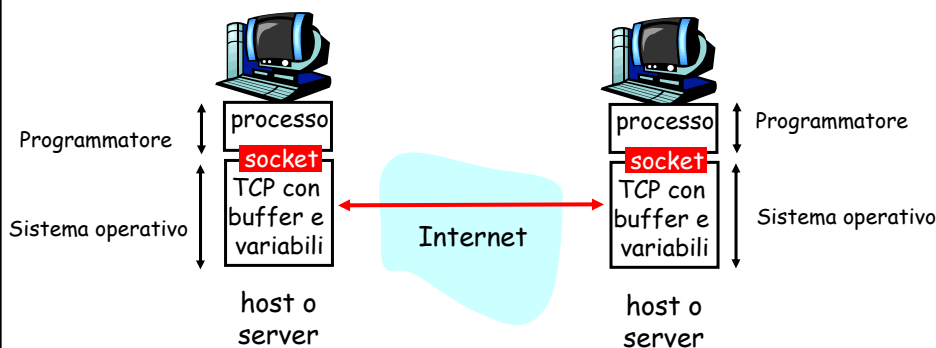
2: Application Layer 6

Processi e programmi

- ❑ I programmi vengono eseguiti dai processi
- ❑ I processi sono entità gestite dal Sistema Operativo
- ❑ Ogni processo comunica attraverso canali
 - Un canale è identificato da un numero intero detto "porta"
- ❑ Le socket sono particolari canali per la comunicazione via rete tra processi che risiedono su macchine diverse

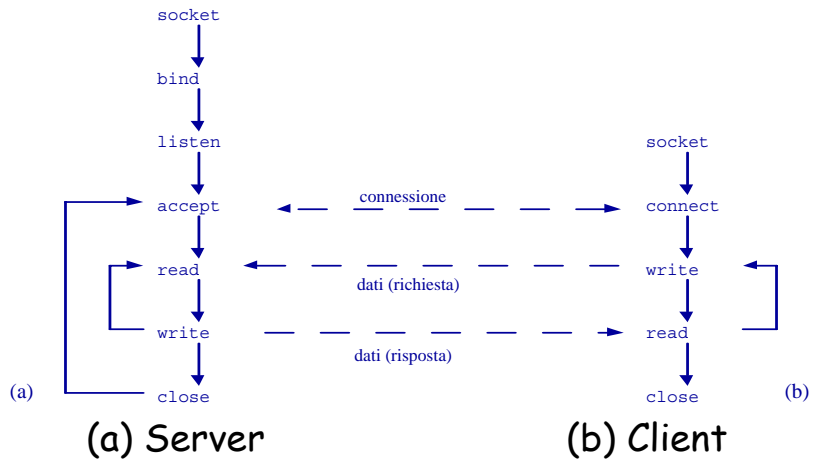
2: Application Layer 7

Socket: funzionamento di base



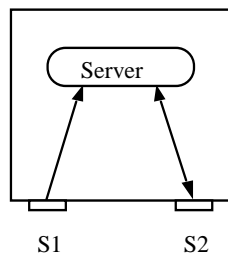
2: Application Layer 8

Client e server TCP/IP



2: Application Layer 9

Processi e Socket

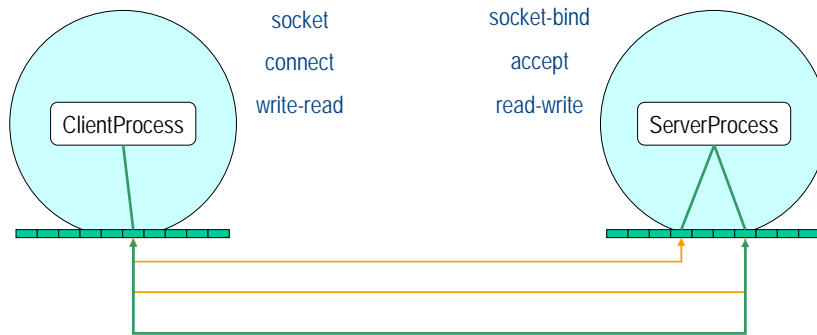


Legenda:

- S1 Socket per accettare richieste di connessione
- S2 Socket per connessioni individuali

2: Application Layer 10

Processi e socket



2: Application Layer 11

URL (Uniform Resource Locator)

www.someSchool.edu/someDept/pic.gif

<http://www.someSchool.edu/someDept/pic.gif>

<http://www.someSchool.edu:80/someDept/pic.gif>

[protocollo://indirizzo_IP\[:porta\]/cammino/file](http://indirizzo_IP[:porta]/cammino/file)

<ftp://www.adobe.com/download/acroread.exe>

2: Application Layer 12

Requisiti delle applicazioni

Perdita (Data loss)

- ❑ Alcune app.ni (es., audio) sono tolleranti (fino a un certo punto)
- ❑ Altre (es., FTP, telnet) richiedono affidabilità totale

Banda

- ❑ Alcune app.ni (soprattutto multimediali) richiedono una banda minima
- ❑ Altre (dette "elastiche") usano la banda a disposizione

Ritardo

- ❑ Alcune applicazioni (es., telefonia Internet, giochi interattivi in rete) richiedono una banda minima per funzionare con qualità sufficiente

Nota: alcuni requisiti sono determinati da esigenze percettive umane (es. ritardo nella telefonia Internet)

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	loss-tolerant	elastic	no
real-time audio/video	loss-tolerant	audio: 5Kb-1Mb video: 10Kb-5Mb	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
financial apps	no loss	elastic	yes and no

Servizi offerti dai protocolli di trasporto Internet

Servizio TCP :

- ❑ *Orientato alla connessione:* richiesto "setup" tra client e server
- ❑ *Trasporto affidabile (reliable transfer)* tra processi mittente e ricevente
- ❑ *Controllo di flusso) flow control:* il mittente non sommerge il ricevente
- ❑ *Controllo della congestione (congestion control):* si limita il mittente quando la rete è sovraccarica
- ❑ *Non offre:* garanzie di banda e ritardo minimi

Servizio UDP :

- ❑ Trasporto non affidabile tra processi mittente e ricevente
 - ❑ Non offre: connessione, affidabilità, controllo di flusso, controllo di congestione, garanzie di ritardo e banda
- D:** perché esiste UDP? Può essere conveniente per le applicazioni (si vedrà più avanti)

2: Application Layer 15

Applicazioni Internet: loro protocolli e protocollo di trasporto usato

<u>Applicazione</u>	<u>Protocollo applicativo</u>	<u>Protocollo di trasporto usato</u>
e-mail	smtp [RFC 821]	TCP
remote terminal access	telnet [RFC 854]	TCP
Web	http [RFC 2068]	TCP
file transfer	ftp [RFC 959]	TCP
streaming multimedia	proprietario (es. RealNetworks)	TCP o UDP
remote file server	NFS	TCP o UDP
Internet telephony	proprietario (e.g., Vocaltec)	tipicamente UDP

2: Application Layer 16

WWW: terminologia essenziale

- Pagina Web:
 - È costituita da "oggetti" (di solito: pagina HTML iniziale+oggetti indirizzati)
 - È indirizzata da una URL
- URL (Uniform Resource Locator)
 - Identifica un oggetto nella rete e specifica il modo per accedere ad esso
 - Ha due componenti: nome dell'host e percorso nell'host:
- Uno user agent per il Web è detto browser:
 - MS Internet Explorer
 - Netscape Communicator
- un server per il Web è detto Web server:
 - Apache (pubblico dominio)
 - MS Internet Information Server

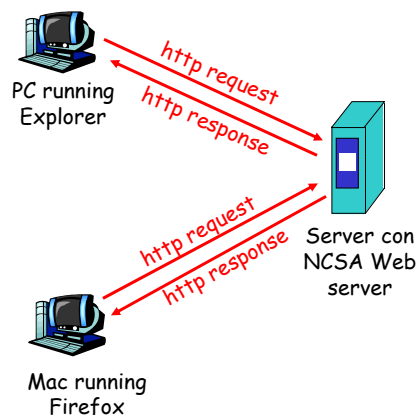
www.someSchool.edu/someDept/pic.gif

2: Application Layer 17

Il Web: protocollo http

http: hypertext transfer protocol

- Protocollo di livello applicativo per il Web
- Usa il modello client/server
 - *client*: browser che richiede, riceve e "mostra" oggetti Web
 - *server*: Web server che invia oggetti in risposta alle richieste
- http1.0: RFC 1945
- http1.1: RFC 2068



2: Application Layer 18

Il protocollo http (cont.)

http: usa TCP:

- ❑ Il client inizia una connessione TCP (crea un socket) verso il server sulla porta 80
- ❑ Il server accetta la connessione TCP dal client
- ❑ Vengono scambiati messaggi http (messaggi del protocollo di livello applicativo) tra il browser (client http) e il Web server (server http)
- ❑ La connessione TCP è chiusa

http è "stateless"

- ❑ Il server non mantiene informazione sulle richieste precedenti del client

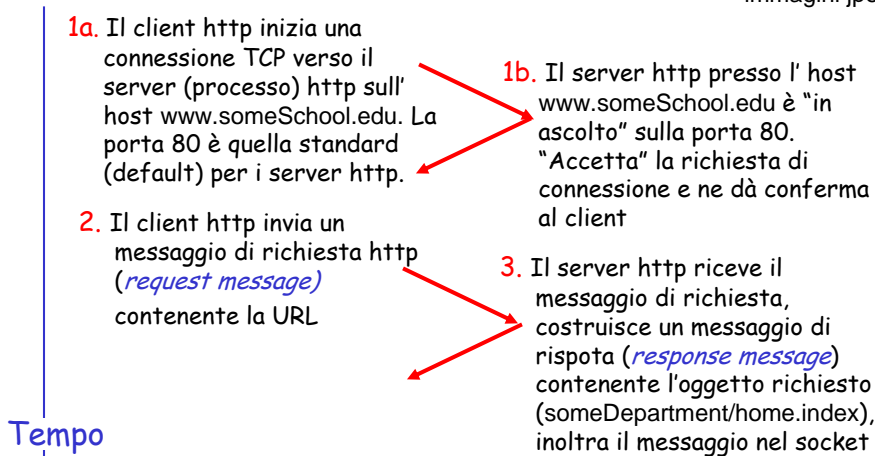
I protocolli che mantengono informazione di stato sono complessi (es. TCP) !

2: Application Layer 19

Esempio http

L' utente accede alla URL

www.someSchool.edu/someDepartment/home.index (contiene testo e i riferimenti a 10 immagini jpeg)



2: Application Layer 20

Esempio http (cont.)

- time ↓
5. Il client http riceve il messaggio di risposta contenente il file html, visualizza la pagina html. Analizzando il file html, il browser trova i riferimenti a 10 oggetti jpeg
 4. Il server http chiude la connessione TCP.
 6. I passi 1-5 sono ripetuti per ciascuno dei 10 oggetti jpeg

2: Application Layer 21

Connessioni persistenti e non-persistenti

Non-persistente

- HTTP/1.0
- Il server analizza la richiesta, risponde e chiude la connessione TCP
- 2 RTTs per ricevere ciascun oggetto
- Ogni oggetto subisce lo "slow start" TCP
- E' possibile parallelizzare le richieste agli oggetti di una pagina

Persistente

- default per HTTP/1.1
- Sulla stessa connessione TCP : il server analizza una richiesta, risponde, analizza la richiesta successiva,...
- Il client invia richieste per tutti gli oggetti appena riceve la pagina HTML iniziale.
- Si hanno meno RTTs e slow start.
- Connessione incanalata:
 - non si attende la risposta alla richiesta precedente prima di inviare la successiva.
 - Richiesta successiva a ridosso della precedente così come le risposte del server.
 - Solo 2 RTT per ottenere un insieme di oggetti

2: Application Layer 22

Formato dei messaggi http

□ Due tipi di messaggi http: *request, response*

□ **Messaggio http request:**

- ASCII (formato testo leggibile)

Chiudi la connessione
al termine della
richiesta

Request line
(GET, POST,
HEAD commands)

header
lines

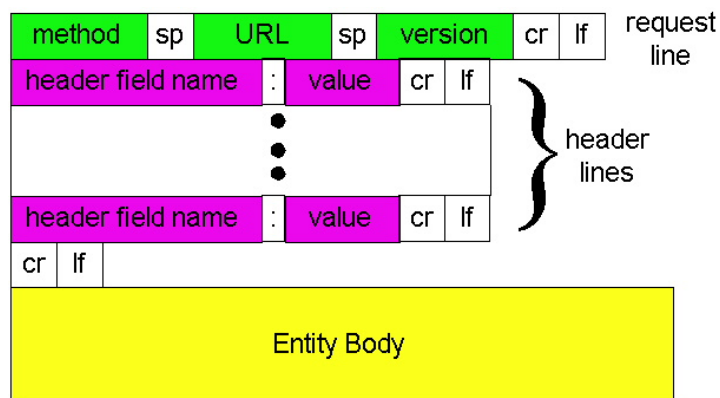
Carriage return,
line feed
indica fine
messaggio

```
GET /somedir/page.html HTTP/1.1
Host: www.someschool.edu
Connection: close
User-agent: Mozilla/4.0
Accept: text/html, image/gif, image/jpeg
Accept-language: fr
```

(extra carriage return, line feed)

2: Application Layer 23

Message http request : formato generale



2: Application Layer 24

Message http request : formato generale

□ Metodo post:

- Usato quando l'utente compila una form. Il contenuto dei campi della forms sono disposti nell'Entity Body
- Il comando richiede una pagina Web il cui contenuto dipende dalle informazioni nel campo body
- Ex: Query inviata ad un motore di ricerca

□ Metodo Head:

- Simile al metodo get ma viene restituito solo l'Head della pagina Web
- Spesso usato in fase di debugging

2: Application Layer 25

Formato del messaggio http response

status line
(protocol
status code
status phrase) → 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

header lines

data, e.g.,
requested
html file → data data data data data ...

Client HTTP 1.0: Server chiude connessione al termine della richiesta
Client HTTP 1.1: mantiene aperta la connessione oppure chiude se Connection: close

2: Application Layer 26

Risposta: codici di stato

Prima riga del messaggio di risposta server->client.
Alcuni esempi:

200 OK

- Successo, oggetto richiesto più avanti nel messaggio

301 Moved Permanently

- L'oggetto richiesto è stato spostato. Il nuovo indirizzo è specificato più avanti (Location:)

400 Bad Request

- Richiesta incomprensibile al server

404 Not Found

- Il documento non è stato trovato sul server

505 HTTP Version Not Supported

2: Application Layer 27

Prova (client)

1. Telnet verso un Web server:

```
telnet www.dis.uniroma1 80
```

Apre connessione TCP verso la porta 80 (default) presso www.dis.uniroma1.it. Tutto quanto viene digitato è inviato alla porta 80 di www.dis.uniroma1.it

2. Si digita una richiesta http GET:

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

Digitando ciò (carriage return due volte), si invia una richiesta GET al server http

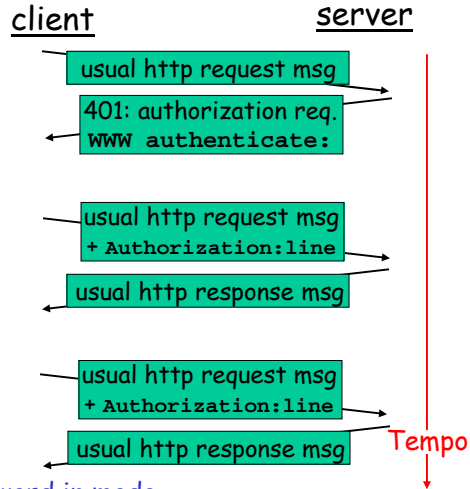
3. Si osservi la risposta!

2: Application Layer 28

Autenticazione

Obiettivo: controllare l' accesso ai documenti sul server

- **stateless:** il client deve autenticare ogni richiesta
- autenticazione: tipicamente log e password
 - authorization: riga nell'header del messaggio di richiesta
 - Senza autenticazione il server rifiuta la connessione
 WWW authenticate:
 Nell' header

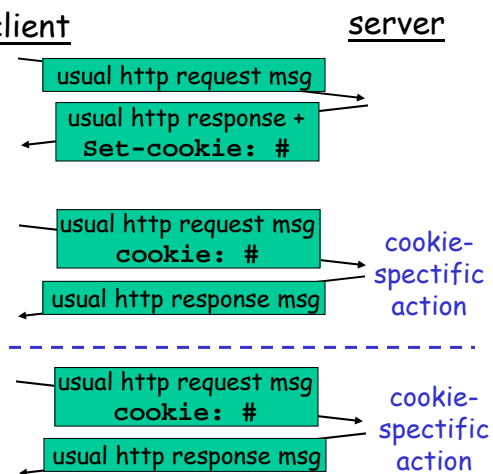


Il browser memorizza nome & password in modo che l'utente non debba digitarli ogni volta.

2: Application Layer 29

Cookie

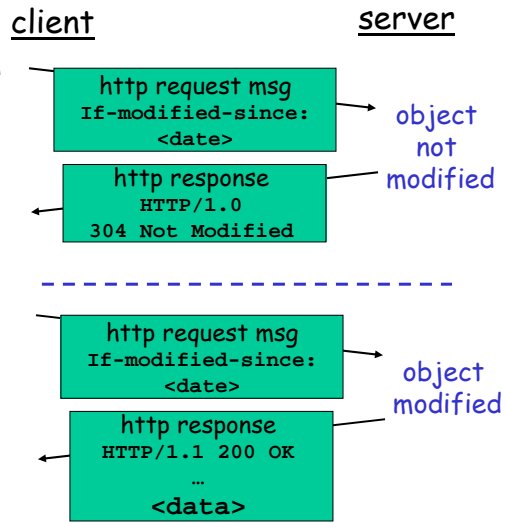
- Il server invia un "cookie" al client con la risposta
 set-cookie: 1678453
- Il client presenta il cookie in accessi successivi
 cookie: 1678453
- Il server controlla il cookie presentato
 - Autenticazione
 - Traccia delle preferenze dell'utente



2: Application Layer 30

GET condizionale (conditional GET)

- ❑ **Obiettivo:** non inviare oggetti che il client ha già in cache
- ❑ **client:** data dell'oggetto memorizzato in cache
If-modified-since: <date>
- ❑ **server:** la risposta è vuota se l'oggetto in cache è aggiornato:
HTTP/1.0 304 Not Modified

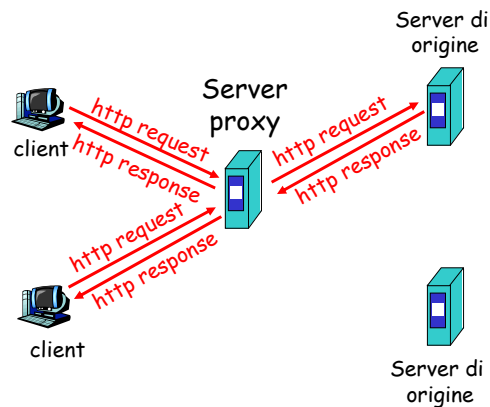


2: Application Layer 31

Web Cache (proxy server)

Obiettivo: rispondere alle richieste evitando di accedere al server remoto

- ❑ L'utente configura il browser: accesso attraverso web cache
- ❑ Il client invia tutte le richieste al proxy
- ❑ La cache restituisce l'oggetto se presente
 - Altrimenti l'oggetto è richiesto prima al server e poi è restituito al client

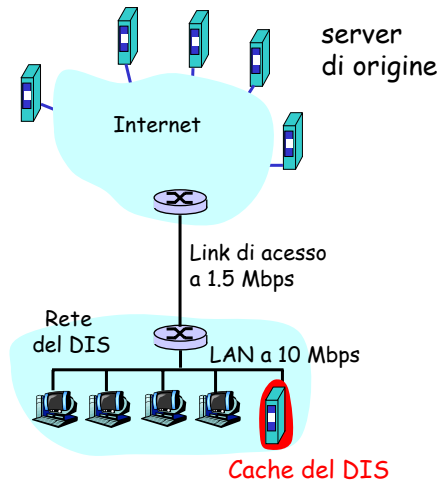


2: Application Layer 32

Perché il Web Caching?

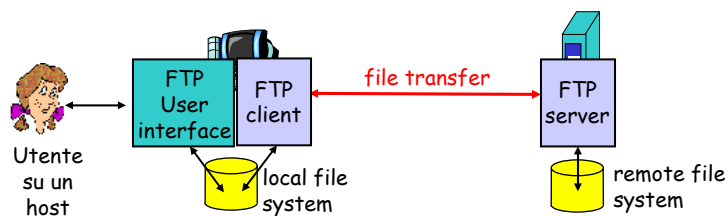
Assunzione: la cache è "vicina" al client (es., stessa rete locale)

- Tempo di risposta minore: la cache è "più vicina" al client
- Diminuisce il traffico verso server lontani
 - Il link di uscita della rete di un ISP istituzionale/locale è spesso un collo di bottiglia



2: Application Layer 33

ftp: File transfer protocol



- Trasferimento file da/verso un host remoto
- Usa il modello client/server
 - *client*: parte che richiede il trasferimento (da/verso l'host remoto)
 - *server*: host remoto
- ftp: RFC 959
- ftp server: porta 21

2: Application Layer 34

ftp: connessioni controllo e dati separate

- Il client contatta il server sulla porta 21, specificando TCP come protocollo di trasporto
- due connessioni TCP parallele:
 - **controllo**: scambio di messaggi di controllo tra client e server.
"controllo fuori banda"
 - **dati**: trasferimento dati da/verso il server
- Entrambi le connessioni aperte dal client
- Il server ftp mantiene info di "stato": directory corrente, autenticazione
- Una nuova connessione per ogni file trasferito



2: Application Layer 35

ftp comandi, risposte

Esempi di comandi:

- Inviati come testo ASCII mediante il canale di controllo
- **USER** *username*
- **PASS** *password*
- **LIST** richiede la lista dei file nella directory corrente (*ls*)
- **RETR** <file> richiede (get) un file
- **STOR** <file> scarica (put) un file sull' host remoto

Esempi di codici

- Codice di stato e frase (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

2: Application Layer 36

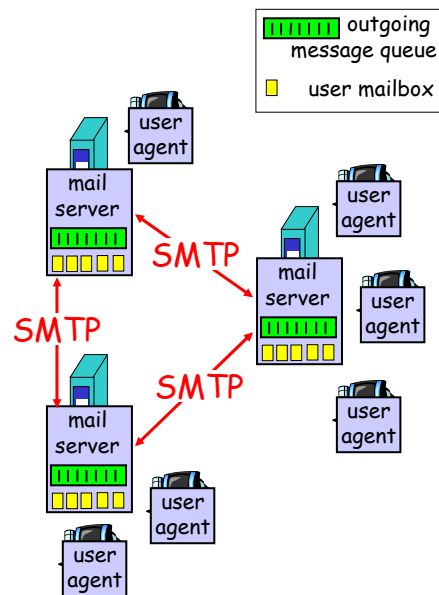
Posta elettronica

Tre componenti principali :

- User agent
- Server di posta
- Simple Mail Transfer Protocol: SMTP

User Agent

- "Lettore di posta"
- Composizione e lettura di messaggi di posta
- Es., Eudora, Outlook, elm, Netscape Messenger
- I messaggi in ingresso/uscita memorizzati sul server



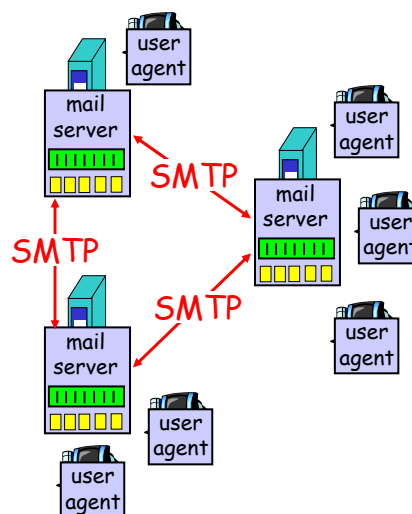
2: Application Layer 37

Posta elettronica: mail server

Mail Server

- Mailbox contenente messaggi (non ancora letti) per l'utente
- Coda di messaggi in uscita (non ancora spediti)
- Protocol smtp tra i mail server per il recapito dei messaggi
 - client: il server che invia il messaggio
 - "server": server che riceve il messaggio

Nota: solo una distinzione di ruoli



2: Application Layer 38

Recapito di un messaggio di posta elettronica

- ❑ Alice compone un messaggio e lo inoltra al suo Mail Server
- ❑ Mail Server dispone il messaggio nella coda di messaggi in uscita
- ❑ Mail Server di Alice apre una connessione smtp con il Mail Server di Bob ed inoltra il messaggio
- ❑ Se il contatto fallisce, l'invio è ripetuto ogni trenta minuti
- ❑ Se l'invio fallisce per diversi giorni, mail di notifica inviato ad Alice
- ❑ Mail Server di Bob riceve il messaggio dal Mail Server di Alice e lo salva nella Mailbox di Bob
- ❑ Bob accede la propria Mailbox specificando Username e Password
- ❑ Messaggi possono essere trasferiti dalla Mailbox all'host da cui Bob ha acceduto la Mailbox e/o lasciati sul server
- ❑ Bob legge il messaggio di Alice

2: Application Layer 39

Posta elettronica: smtp [RFC 821] (1982!)

- ❑ Usa tcp per il trasferimento affidabile dei messaggi da client a server, porta 25
- ❑ Trasferimento diretto: da server a server, non si usano server intermedi di posta
- ❑ Tre fasi
 - Handshaking (saluto)
 - Trasferimento di uno o più messaggi (connessione permanente)
 - Chiusura
- ❑ Interazione mediante comandi/risposte
 - **Comando:** testo ASCII
 - **Risposta:** codice di stato e frase
- ❑ **Attenzione:** I messaggi devono essere comunque riportati in formato ASCII a 7 bit, **anche dati multimediali**

2: Application Layer 40

Esempio di interazione SMTP

```
S: 220 hamburger.edu
C: HELO crepes.fr
S: 250 Hello crepes.fr, pleased to meet you
C: MAIL FROM: <alice@crepes.fr>
S: 250 alice@crepes.fr... Sender ok
C: RCPT TO: <bob@hamburger.edu>
S: 250 bob@hamburger.edu ... Recipient ok
C: DATA
S: 354 Enter mail, end with "." on a line by itself
C: Do you like ketchup?
C: How about pickles?
C: .
S: 250 Message accepted for delivery
C: QUIT
S: 221 hamburger.edu closing connection
```

2: Application Layer 41

Prova

- telnet servername 25
- Attendi 220 risposta dal server
- Inserisci HELO, MAIL FROM, RCPT TO, DATA, QUIT
- Comandi permettono di inviare email senza usare un User Agent

2: Application Layer 42

SMTP: conclusioni

- ❑ Connessioni TCP persistenti
 - ❑ Richiede che il messaggio (header & corpo) sia in formato ascii 7-bit
 - ❑ Alcune sequenze di caratteri non consentite nel messaggio (es., CRLF.CRLF).
Conseguenza: il messaggio deve essere codificato
 - ❑ Il server smtp usa CRLF.CRLF per determinare la fine del messaggio
- Confronto con http**
- ❑ http: pull
 - ❑ email: push
 - ❑ Entrambi usano un'interazione mediante comandi/risposta in testo ASCII e codici di stato
 - ❑ http: ogni oggetto incapsulato nel messaggio di risposta
 - ❑ smtp: un messaggio con più oggetti è inviato mediante un messaggio in più parti

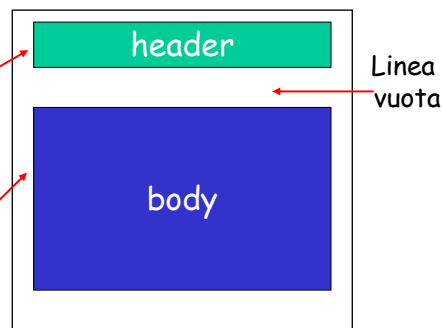
2: Application Layer 43

Formato dei messaggi

smtp: protocollo per lo scambio di messaggi di posta

RFC 822: standard per il formato dei messaggi inviati:

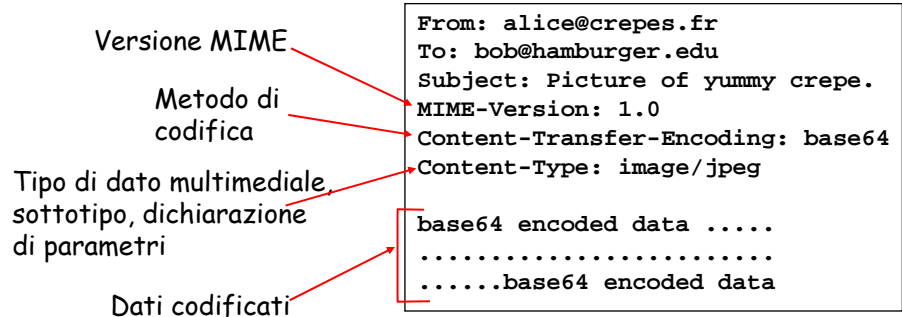
- ❑ **header**, es.,
 - To:
 - From:
 - Subject:*Diversi dai comandi smtp*
- ❑ **body**
 - Il "messaggio" vero e proprio, solo caratteri ASCII



2: Application Layer 44

Formato: estensioni multimediali

- MIME: multipurpose internet mail extension, RFC 2045, 2056. Dati Multimediali e di specifiche applicazioni
- Righe aggiuntive dell'header specificano il tipo del contenuto MIME



2: Application Layer 45

Tipi MIME

Content-Type: type/subtype; parameters

Text

- Esempi di sottotipi: plain, html

Video

- Esempi di sottotipi: mpeg, quicktime

Image

- Esempi di sottotipi: jpeg, gif

Application

- Dati che devono essere processati da un'applicazione prima di essere "visibili"
- Esempi di sottotipi: msword, octet-stream

Audio

- Esempi di sottotipi: basic (8-bit mu-law encoded), 32kadpcm (32 kbps coding)

2: Application Layer 46

Tipo Multipart

```
From: alice@crepes.fr
To: bob@hamburger.edu
Subject: Picture of yummy crepe.
MIME-Version: 1.0
Content-Type: multipart/mixed; boundary=98766789
```

```
--98766789
Content-Transfer-Encoding: quoted-printable
Content-Type: text/plain
```

```
Dear Bob,
Please find a picture of a crepe.
```

```
--98766789
Content-Transfer-Encoding: base64
Content-Type: image/jpeg
```

```
base64 encoded data .....
.....base64 encoded data
--98766789--
```

- E-mail contenenti più oggetti.
- Boundary character: delimitano i messaggi.
- Content-Transfer-Encoding e Content-Type per ogni oggetto

2: Application Layer 47

Messaggio ricevuto

```
Received: from crepes.fr by hamburger.edu; 6 Oct
2003
```

```
From: alice@crepes.fr
To: bob@hamburger.edu
Subject: Picture of yummy crepe.
MIME-Version: 1.0
Content-Transfer-Encoding: base64
Content-Type: image/jpeg
```

- Received indica i Mail Server che hanno recapitato il messaggio
- Più linee "Received" se il messaggio è stato inoltrato da più server SMTP lungo il percorso da mittente a destinatario

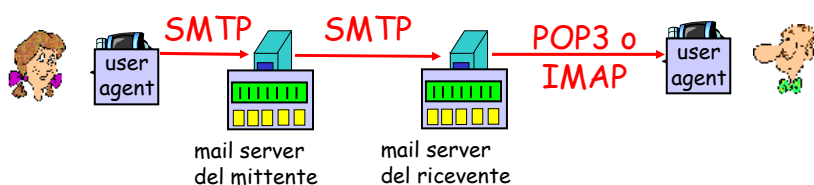
2: Application Layer 48

Protocolli di accesso alla posta

- ❑ Soluzione tradizionale: utente legge direttamente la posta sul Mail Server
- ❑ L'host su cui è disposto il Mail Server deve essere sempre attivo
- ❑ Agenti di posta permettono di trasferire la posta dal Mail Server all'host locale al ricevente
- ❑ Possibile visualizzare file multimediali e di specifiche applicazioni
- ❑ Occorre un protocollo "Pull" per accedere alla Mailbox collocata sul Mail Server

2: Application Layer 49

Protocolli di accesso alla posta



- ❑ SMTP: consegna al/memorizzazione nel server di posta del ricevente
- ❑ Protocollo di accesso: recupero della posta dal server locale
 - POP: Post Office Protocol [RFC 1939]
 - Autenticazione (agent <-->server) e scaricamento
 - IMAP: Internet Mail Access Protocol [RFC 1730]
 - Più possibilità (più complesso)
 - Manipolazione dei messaggi memorizzati sul server
 - HTTP: Hotmail , Yahoo! Mail, ecc.

2: Application Layer 50

Protocollo POP3

Fase di autorizzazione

- ❑ Comandi del client:
 - user: nome utente
 - pass: password
- ❑ Risposte del server
 - +OK
 - -ERR

Fase di transazione, client:

- ❑ list: lista numeri e dim. msg
- ❑ retr: scarica messaggio in base al numero
- ❑ dele: cancella
- ❑ quit

```
S: +OK POP3 server ready
C: user alice
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
```

2: Application Layer 51

Protocollo POP3

- ❑ Scarica ed elimina:
 1. User Agent elimina la posta dalla Mailbox dopo averla scaricata
 2. Un utente disperde la posta sui diversi host da cui accede la Mailbox
 3. User Agent permette di creare cartelle, spostare messaggi, effettuare ricerche nei messaggi

- ❑ Scarica e conserva
 1. User Agent conserva la posta sulla Mailbox
 2. Utente può leggere i messaggi da macchine diverse
 3. POP3 stateless, non permette di strutturare i messaggi in directory

2: Application Layer 52

Protocollo IMAP

- Permette di gestire cartelle di posta remote come se fossero locali
- IMAP deve mantenere una gerarchia di cartelle per ogni utente
- Permette allo User Agent di scaricare solo parti del messaggio:
 - Intestazione
 - Solo intestazione file MIME Multipart
 - Messaggi di dimensione piccola per utenti a banda limitata
- Stati:
 - Non-authenticated: utente deve fornire username e password per la connessione
 - Authenticated State: utente deve specificare una cartella prima di eseguire comandi che influiscono sul messaggio
 - Selected State: utente può dare comandi che influiscono sul messaggio, e.g. elimina, salva, sposta
 - Logout State: sessione terminata

2: Application Layer 53

DNS: Domain Name System

Persone: molte mezzi di identificazione:

- CF, nome, # Passaporto

Host, router Internet:

- Indirizzi IP (32 bit) - usati per indirizzare i datagrammi IP
- "Nome", es., gaia.cs.umass.edu - usati dagli utenti

Q: corrispondenza tra indirizzo IP e nome?

Domain Name System:

- *Database distribuito* implementato come una gerarchia di molti *name server*
- *Protocollo applicativo* usato da host, router, name server per comunicare allo scopo di *risolvere* (tradurre) i nomi in indirizzi IP
 - Nota: funzione di base di Internet implementata come protocollo applicativo
 - La complessità trasferita al "bordo" della rete

2: Application Layer 54

Funzione del DNS

- ❑ Utilizzato da diverse applicazioni: HTTP, SMTP, FTP
- ❑ Connessione TCP richiede la conoscenza dell'indirizzo IP corrispondente all'hostname
- ❑ Applicazione interroga un sever DNS per ottenere l'indirizzo IP
- ❑ Opportuno utilizzare la cache per ridurre il ritardo
- ❑ Utilizza UDP
- ❑ UNIX: gethostbyname()
- ❑ Fornisce l'hostname corrispondente ad un alias
- ❑ Alias di posta: fornisce l'indirizzo IP o l'hostname del Mail Server di un dominio
- ❑ Distribuzione del carico tra Server replicati. IL DNS fornisce un gruppo di indirizzi alternando l'ordine
- ❑ Permette di dirigere un client al server più vicino

2: Application Layer 55

Name server DNS

Perché non un server DNS centralizzato?

- ❑ Minore tolleranza ai guasti
- ❑ Traffico eccessivo
- ❑ Database centrale troppo distante in molti casi
- ❑ Scarsa scalabilità!
- ❑ Autorizzazione ed accesso per registrare nuovo host
- ❑ Nessun name server contiene tutte le associazioni nome simbolico/indirizzo IP

Name server locali :

- Ogni ISP o compagnia ha un *name server locale (default)*
- La richiesta di traduzione (mapping) di un host è prima rivolta al name server locale

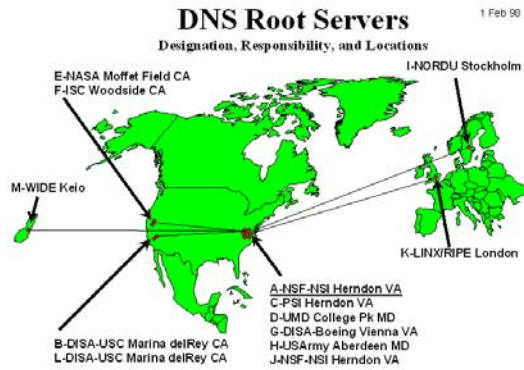
Name server di riferimento :

- Per un host: per definizione è quello che è sempre in grado di eseguire la traduzione (mapping) nome simbolico/indirizzo IP dell'host

2: Application Layer 56

DNS: Root name servers

- Contattato dal name server locale che non riesce a risolvere un nome
- root name server:
 - Contatta il name server di riferimento (authoritative) se la traduzione non è nota
 - Ottiene la traduzione
 - Restituisce la traduzione al name server locale
- ~ una dozzina di root name server nel mondo

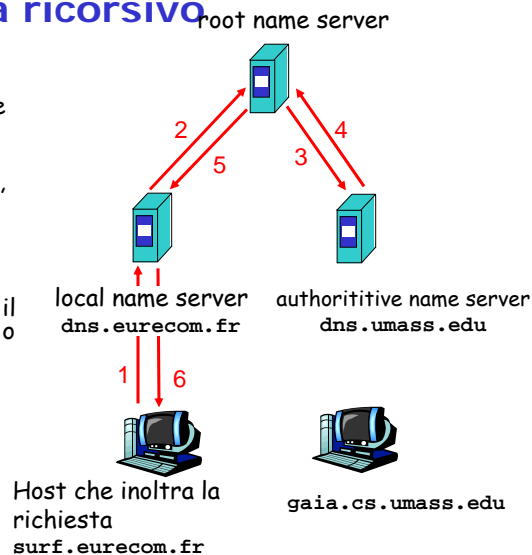


2: Application Layer 57

Esempio: schema ricorsivo

L'host `surf.eurecom.fr` vuole l'indirizzo IP di `gaia.cs.umass.edu`

1. Contatta il server DNS locale, `dns.eurecom.fr`
2. `dns.eurecom.fr` contatta il root name server, se necessario
3. Il root name server contatta il name server di riferimento o assoluto, `dns.umass.edu`, se necessario
4. Ogni host è registrato in almeno due name server assoluti
4. Il name server assoluto può coincidere con il name server locale.

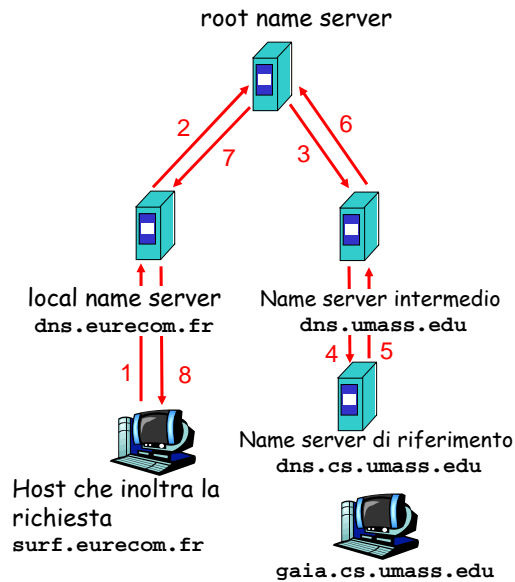


2: Application Layer 58

Esempio (2)

Root name server:

- Può non essere a conoscenza di un name server di riferimento
- Può tuttavia conoscere un *name server intermedio* che contatta per avere raggiungere quello di riferimento



2: Application Layer 59

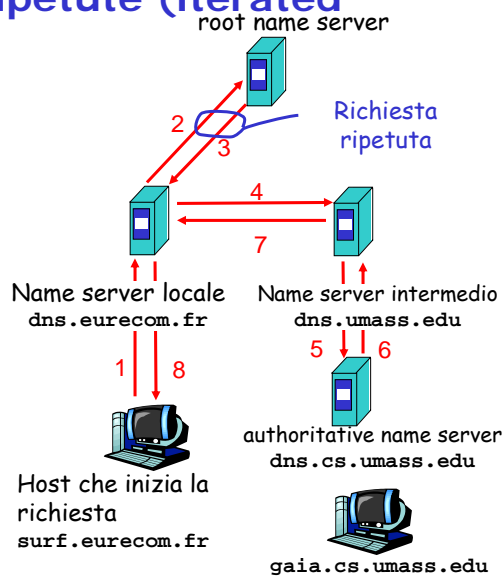
DNS: richieste ripetute (iterated queries)

Richieste ricorsive (recursive query):

- Trasferisce il carico della traduzione al name server contattato
- Carico eccessivo?

Richieste ripetute (iterated query):

- Il name server contattato risponde con l'indirizzo del prossimo name server da contattare
- "Non conosco questo nome, ma prova a rivolgerti a quest'altro server"



2: Application Layer 60

DNS: caching e aggiornamento

- ❑ Quando un qualsiasi name server apprende una traduzione la memorizza localmente (**caching**)
 - Le traduzioni memorizzate nella cache (cache entries) scadono (timeout) dopo un certo tempo (di solito un paio di giorni)
- ❑ Se possibile, richieste successive vengono servite usando la traduzione presente in cache
- ❑ I meccanismi di aggiornamento/modifica in studio da parte dell' IETF
 - RFC 2136
 - <http://www.ietf.org/html.charters/dnsind-charter.html>

2: Application Layer 61

Record DNS

DNS: database distribuito che memorizza Resource Record (**RR**)

Formato RR: (nome, valore, tipo, ttl)

- | | |
|---|--|
| <ul style="list-style-type: none">❑ Tipo=A<ul style="list-style-type: none">○ nome è il nome dell' host○ valore è l' indirizzo IP❑ Tipo=NS<ul style="list-style-type: none">○ nome è il dominio (es. foo.com)○ valore è l'indirizzo IP del name server di riferimento per questo dominio | <ul style="list-style-type: none">❑ Tipo=CNAME<ul style="list-style-type: none">○ nome è un alias di qualche nome reale ("canonico")○ valore è il nome canonico❑ Tipo=MX<ul style="list-style-type: none">○ valore è il nome di un mailserver associato a nome |
|---|--|

2: Application Layer 62

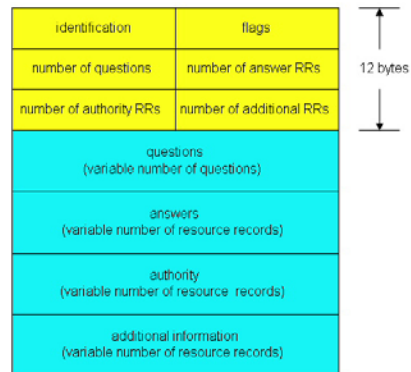
Protocollo DNS, messaggi

Protocollo DNS : messaggi di *richiesta (query)* e *risposta (reply)*,
 → *client server*

Header di messaggio

- **identification**: numero a 16 bit per la richiesta, la risposta usa lo stesso numero
- **flags**:
 - Richiesta o risposta
 - Chiesta la ricorsione (Q)
 - Ricorsione disponibile (R)
 - Il server che risponde è di riferimento per la richiesta (R)

Nota: richiesta e risposta hanno lo stesso formato



2: Application Layer 63

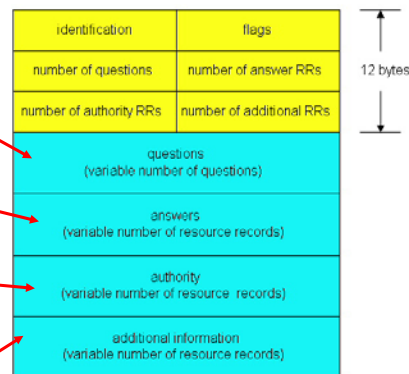
Protocollo DNS, messaggi (2)

Nome, campi tipo per una richiesta

RR in risposta a una richiesta

Record per server di riferimento

Informazioni aggiuntive
 Es.: RR di tipo A contenente indirizzo IP di un mail server il cui nome canonico è contenuto nella answer section



2: Application Layer 64