

Il Livello Applicazione

Principali Obiettivi:

- Aspetti concettuali implementativi dei protocolli di rete del livello applicazione
 - paradigma client server
 - Modelli di servizio
- Impareremo a conoscere i protocolli esaminando alcuni famosi protocolli del livello applicazione

Altri obiettivi

- protocolli specifici :
 - http
 - ftp
 - smtp
 - pop
 - dns
- Programmazione di applicazioni di rete
 - socket programming

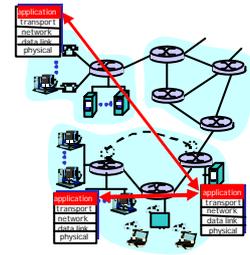
Applicazioni e protocolli application-layer

Applicazioni: processi distribuiti ed intercomunicanti

- che girano sugli host della rete "user space"
- Scambiano messaggi per implementare appl.
- Per es., email, file transfer, il Web

Protocolli Application-layer

- Un "pezzo" di una appl.
- Definiscono i messaggi scambiati dalle appl. e le azioni da intraprendere
- I servizi utente sono forniti dai livelli inferiori



Applicazioni di Rete: "gergo"

- Un **processo** è un programma che sta "girando" su di un host.
- Sullo stesso host, due processi comunicano mediante la comunicazione **interprocesso** definita dal S.O.
- Processi che girano su host differenti comunicano mediante un **protocollo application-layer**
- Uno **user agent** è una interfaccia tra lo user e l'applicazione di rete.
 - Web: browser
 - E-mail: mail reader
 - streaming audio/video: media player

Il paradigma client-server

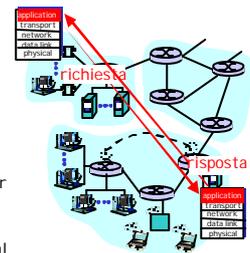
La tipica network app ha due componenti: **client/server**

Client:

- Avvia il contatto con il server ("parla per primo")
- Tipicamente richiede servizio dal server,
- Nel Web, il client è implementato nel browser; per l'e-mail, nel mail reader

Server:

- Fornisce il servizio richiesto al client
- Per es., il Web server invia la pagina Web richiesta, il mail server consegna l'e-mail



Protocolli application-layer (cont).

API: application programming interface

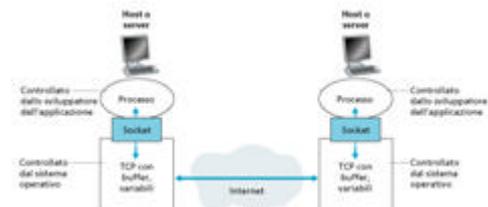
- definisce l'interfaccia tra l'applicazione e il livello transport
- socket: Internet API
 - due processi comunicano inviando dati nel socket, e leggendo dati dal socket (socket=presa di corrente)

D: Come fa un processo ad "identificare" l'altro processo con il quale vuole comunicare?

- **Indirizzo IP** dell'host che fa girare l'altro processo
- Un **"port number"** - consente all'host in ricezione di determinare a quale processo locale va consegnato il messaggio

... ma su questo diremo molto di più in futuro.

Processi dell'applicazione, socket e il sottostante protocollo di trasporto



Che tipo di transport service è necessario per una applicazione?

Perdita di Dati

- Alcune applicazioni (per es., audio) possono tollerare alcune perdite
- Altre applicazioni (per es., ftp, telnet) richiedono un trasferimento dati affidabile al 100%

Tempificazione

- Alcune appl. (per es., la telefonia Internet, I giochi interattivi) richiedono un ritardo piccolo per "funzionare"

Larghezza di Banda (Bandwidth)

- Alcune appl. (per es., quelle multimediali) richiedono un ammontare minimo di banda per "funzionare"
- Altre ("appl.elastiche") fanno uso dell'ammontare di banda disponibile

Il Livello Applicazione 7

Requisiti per il servizio di Transport di alcune applicazioni comuni

Applicazione	Perdita Dati	Bandwidth	Tempificazione
file transfer	nessuna	elastica	no
e-mail	nessuna	elastica	no
Pagine Web	nessuna	elastica	no
real-time audio/video	ammissibile	audio: 5Kb-1Mb video: 10Kb-5Mb	si, 100's msec
stored audio/video	ammissibile	Come sopra	si, pochi sec
Giochi interattivi	ammissibile	pochi Kbps	si, 100's msec
Appl. finanziarie	nessuna	elastica	si e no

Il Livello Applicazione 8

Servizi forniti dai protocolli di Trasporto di Internet

servizio TCP:

- connection-oriented:** setup richiesto tra client e server
- trasporto affidabile:** tra processo mittente e destinatario
- controllo del flusso:** il mittente non sovraccarica il ricevente
- Controllo della congestione:** il mittente viene limitato quando la rete è sovraccarica
- Non fornisce:** tempificazione, garanzie sulla banda minima

servizio UDP:

- Trasferimento dati inaffidabile tra mittente e destinatario
- Non fornisce: setup della connessione, affidabilità, controllo del flusso e della congestione, tempificazione o garanzie sulla banda

D: Ma allora? Perché c'è un UDP?

Il Livello Applicazione 9

Applicazioni Internet: protocolli e protocolli di trasporto

Applicazione	protocollo di applicazione	Protocollo trasporto sottostante
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 (per es., RealNetworks)	TCP or UDP
remote file server	NSF	TCP or UDP
Internet telephony	proprietario (per es., Cisco)	typically UDP

Il Livello Applicazione 10

Il Web: "gergo"

- pagina Web :
 - Consiste di "oggetti"
 - Indirizzata da una URL
- La maggioranza delle pagine Web hanno:
 - una pagina HTML di base e
 - oggetti referenziati nella pagina.
- Una URL ha due componenti: l'host name e il path name:
 - Lo User agent del Web è detto browser:
 - MS Internet Explorer
 - Netscape Communicator
 - Il server del Web è detto Web server:
 - Apache (public domain)
 - MS Internet Information Server

www.someSchool.edu/someDept/pic.gif

Il Livello Applicazione 11

Come sono composti gli URL

<http://www.server.it:80/directory/file.html>

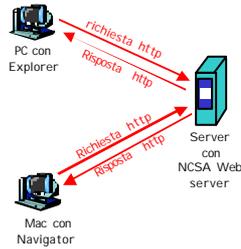
protocollo	Nome del server	Dominio	Numero porta	Directory	File
http:// ftp:// gopher://	www.server	it	: 80 23 60	/ Directory	File.html

Il Livello Applicazione 12

Il Web: protocollo http

http: hypertext transfer protocol

- Il protocollo di applicazione del Web
- Modello client/server
 - **client**: il browser che richiede, riceve e "visualizza" gli oggetti Web
 - **server**: il Web server invia gli oggetti in risposta a richieste
- http1.0: RFC 1945
- http1.1: RFC 2068
- http1.1: RFC 2616



Il Livello Applicazione 13

Ancora sull' http

http: servizio di trasporto TCP :

- Il client avvia una connessione TCP connection (crea un socket) con il server, sulla porta 80
- Il server accetta la connessione TCP dal client
- Vengono scambiati messaggi http (messaggi del protocollo application-layer) tra il browser (http client) ed il Web server (http server)
- La connessione TCP viene chiusa

L'http è "stateless"

- Il server non mantiene informazioni sulle precedenti richieste del client

osservazione

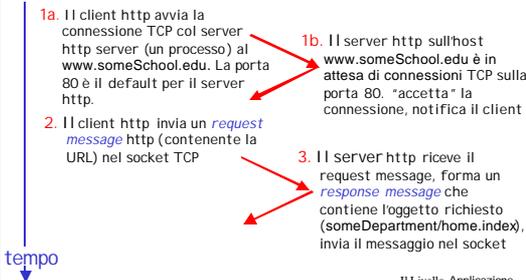
I Protocolli che mantengono uno "stato" sono complessi!

- Occorre conservare la storia passata
- In caso di crash del server/client, gli "stati" possono essere inconsistenti

Il Livello Applicazione 14

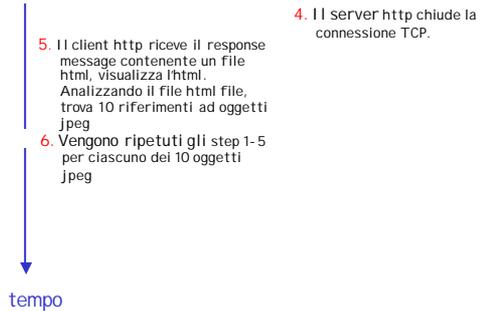
Un esempio http

Supponiamo che lo user digiti una URL (contiene testo, www.someSchool.edu/someDepartment/home.index e riferimenti a 10 immagini jpeg)



Il Livello Applicazione 15

Un esempio http (cont.)



Il Livello Applicazione 16

Connessioni non-persistenti e persistenti

Non-persistenti

- HTTP/1.0
- Il server analizza le richieste, risponde e chiude la connessione TCP
- 2 RTT per estrarre ogni oggetto
- Ogni oggetto soffre dello slow start

Persistenti

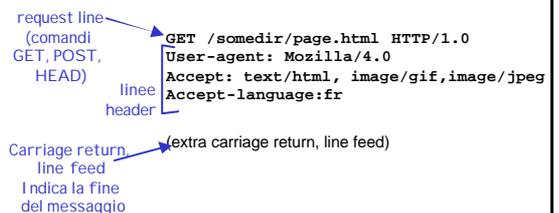
- default per HTTP/1.1
- Sulla stessa connessione TCP il server analizza la richiesta, risponde, analizza la nuova richiesta...
- Il Client invia le richieste per tutti gli oggetti referenziati appena riceve il file base HTML.
- Meno RTT e meno slow start.

Ma la maggior parte dei browser 1.0 usano Più connessioni TCP in parallelo.

Il Livello Applicazione 17

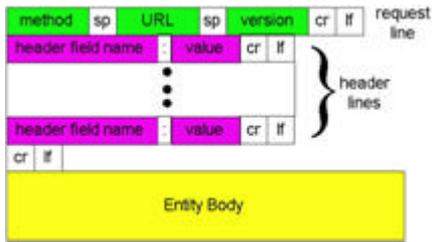
Formato del messaggio http : request

- due tipi di messaggi http: **request**, **response**
- **http request message**:
 - ASCII (formato leggibile)

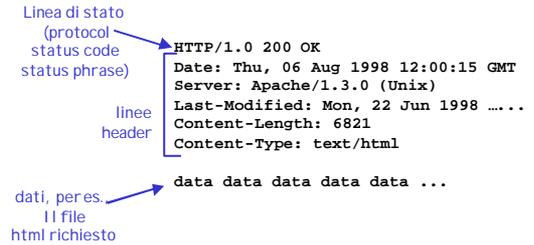


Il Livello Applicazione 18

formato generale del request message:



formato del messaggio http: response



status codes dell'http response

Prima linea nel messaggio di risposta server -> client.
Alcuni codici di esempio:

200 OK

- Richiesta accettata, segue l'oggetto richiesto

301 Moved Permanently

- Oggetto richiesto spostato, segue la nuova posizione (Location:)

400 Bad Request

- Il messaggio di richiesta non è stato compreso dal server

404 Not Found

- Documento richiesto non trovato su questo server

505 HTTP Version Not Supported

Provate l'http (client side) voi stessi

- Fate Telnet al vostro sito Web favorito:

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

Apri la connessione TCP alla porta 80 sul sito www.eurecom.fr. Qualsiasi cosa digitata viene inviata alla porta 80 di www.eurecom.fr

- Digitate una GET:

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

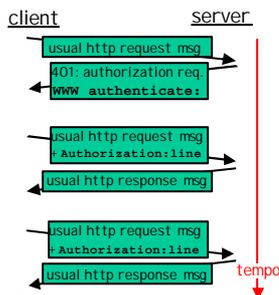
Digitando questo (battete carriage return due volte), inviate una Richiesta GET minima (ma completa) all'http server

- Osservate il response message inviato dal server http!

Interazione user-server: autenticazione

Obiettivo dell'autenticazione:

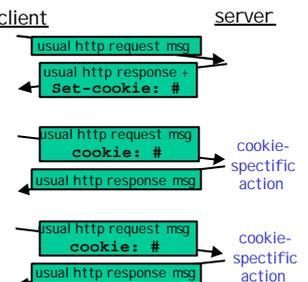
- controllare l'accesso ai documenti del server
- stateless:** il client deve richiedere l'autorizzazione per ciascuna richiesta
- autorizzazione: tipicamente nome, password
 - authorization:** linea di header nella richiesta
 - Se non viene presentata l'autorizzazione, il server rifiuta l'accesso e invia **WWW authenticate:** come header



Il Browser mette nella cache il nome & password così che l'utente non deve digitarle ripetutamente.

Interazione user-server: cookies

- Il server invia "cookie" al client nel msg di risposta
Set-cookie: 1678453
- Il client utilizza i cookie nelle richieste successive
cookie: 1678453
- Il server confronta i cookie presentati con le info memorizzate
 - autenticazione
 - Preferenze dello user, scelte precedenti

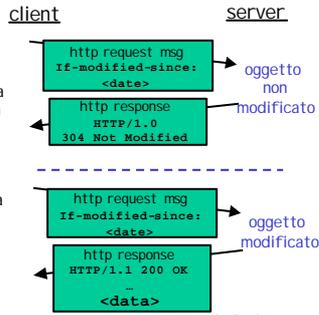


Interaz. user-server: GET condizionale

- Obiettivo: non inviare oggetto se il client già possiede una versione aggiornata nella cache
- client: specifica la data della cached copy nella richiesta http


```
If-modified-since: <date>
```
- server: la risposta non contiene oggetti se la copia nella cache è aggiornata:


```
HTTP/1.0 304 Not Modified
```

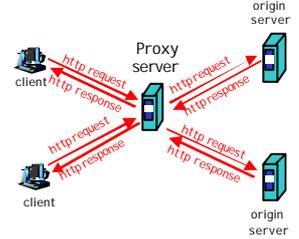


Il Livello Applicazione 25

Web Caches (proxy server)

Obiettivo: soddisfare le richieste del client senza coinvolgere il server originale

- Lo user configura il browser: accesso Web via web cache
- Il client invia tutte le richieste http alla web cache
 - Se l'oggetto è nella web cache, la web cache lo ritorna immediatamente in una http response
 - Altrimenti richiede l'oggetto dal server originale e poi ritorna l'http response al client

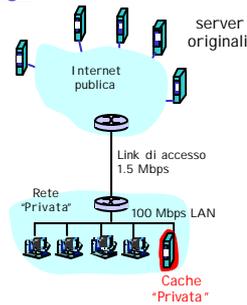


Il Livello Applicazione 26

Perchè Web Caching?

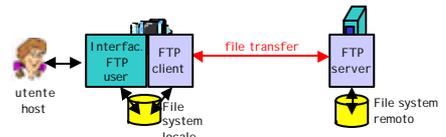
Ipotesi: la cache è "vicina" al client (per es., nella stessa rete)

- Tempo di risposta inferiore
- Diminuzione del traffico verso server distanti
 - i link al di fuori della rete privata/I SP locale sono spesso dei colli di bottiglia



Il Livello Applicazione 27

ftp: protocollo trasferimento file



- trasferisce file da/per host remoti
- modello client/server
 - client: lato che avvia il trasferimento (sia da che per il remoto)
 - server: host remoto
- ftp: RFC 959
- ftp server: porta 21

Il Livello Applicazione 28

ftp: separa connessioni dati e controllo

- L'ftp client contatta l'ftp server alla porta 21, specificando il TCP come protocollo di trasporto
- Apertura di due connessioni TCP parallele:
 - controllo: scambio di comandi, risposte tra client, server.
 - dati: I dati del file da/per il server
- L'ftp server mantiene uno "stato": la dir corrente, precedenti autenticazioni



Il Livello Applicazione 29

ftp: comandi e risposte

Esempi di comandi:

- Inviati come ASCII text sul control channel
- USER** *username*
- PASS** *password*
- LIST** ritorna la lista dei file nella dir corrente
- RETR filename** ritorna (preleva) file
- STOR filename** memorizza (invia) file sull'host remoto

Esempi di codici di ritorno

- status code 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**

Il Livello Applicazione 30

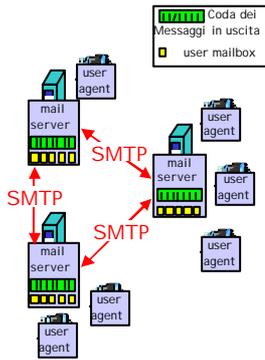
Posta Elettronica

Tre componenti principali:

- user agents
- mail servers
- simple mail transfer protocol: smtp

User Agent

- il "mail reader"
- Composizione, editing, lettura dei messaggi
- Per es., Eudora, Outlook, elm, Netscape Messenger
- I messaggi in uscita ed in ingresso sono memorizzati sul server

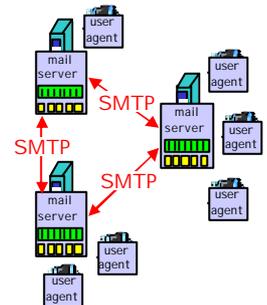


Il Livello Applicazione 31

Posta Elettronica: I mail servers

Mail Servers

- Mailbox: contiene i messaggi in arrivo (ancora da leggere) per lo user
- Coda di messaggi in uscita: contiene i messaggi da inviare
- Smtip: protocollo tra mail servers per inviare i messaggi di email
 - client: server di invio della mail
 - server: server di ricezione della mail



Il Livello Applicazione 32

Posta Elettronica : smtp [RFC 821]

- Usa il tcp per trasferire affidabilmente msg di email dal client al server, porta 25
- Trasferimento: dal server di invio a quello di ricezione
- Trasferimento a tre fasi
 - handshaking (saluti)
 - Trasferimento di messaggi
 - chiusura
- Interazione comando/risposta
 - comandi: testo ASCII
 - risposte: codici di stato e frase
- I messaggi dovevano essere in 7-bit ASCII

Il Livello Applicazione 33

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
```

Il Livello Applicazione 34

Provate l'smtp interaction da soli:

- telnet servername 25
- Osservate la reply 220 dal server
- Digitare i comandi HELO, MAIL FROM, RCPT TO, DATA, QUIT

In questo modo è possibile inviare una email senza usare l'email client

Il Livello Applicazione 35

smtp: conclusioni

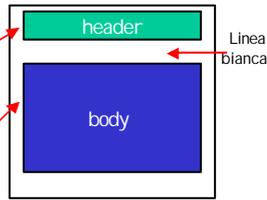
- smtp usa connessioni persistenti
 - smtp richiede che il corpo del messaggio (header & body) sia in 7-bit ascii
 - Certe stringhe di caratteri non sono permesse nel messaggio (per es., CRLF.CRLF). Così il messaggio deve essere codificato (usualmente sia in base-64 o nel quoted printable)
 - L' smtp server usa CRLF.CRLF per determinare la fine del message
- ### Comparazione con l' http
- http: pull (tira)
 - email: push (spingi)
 - Ambedue hanno interazione comando/risposta in ASCII, status codes
 - http: ogni oggetto è incapsulato nel suo proprio response message
 - smtp: oggetto multipli inviati in un messaggio multipart

Il Livello Applicazione 36

Formato del messaggio di mail

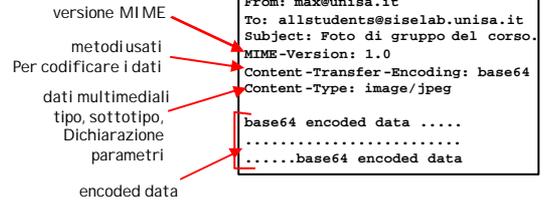
smtp: protocollo per scambiare msg di email
RFC 822: standard per il formato messaggi di testo:

- header lines, per es.,
 - To:
 - From:
 - Subject:
- Diversi dai comandi smtp!*
- corpo
 - Il "messaggio", solo caratteri ASCII



Formato messaggio: estensioni multimediali

- MIME: multimedia mail extension, RFC 2045, 2056
- Linee aggiuntive nel msg header per dichiarare il tipo di contenuto MIME



tipi MIME

Content-Type: type/subtype; parameters

Text

- Esempi di subtypes: **plain, html**

Image

- Esempi di subtypes: **jpeg, gif**

Audio

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

Video

- Esempi di subtypes: **mpeg, quicktime**

Application

- Altri dati che devono essere elaborati dal prima di essere "visualizzabili"
- Esempi di subtypes: **msword, octet-stream**

Multipart Type

```
From: max@unisa.it
To: allstudents@siselab.unisa.it
Subject: Foto di gruppo del corso.
MIME-Version: 1.0
Content-Type: multipart/mixed; boundary=98766789
```

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

```
Cari Ragazzi,
In allegato trovate una foto di gruppo del nostro corso.
--98766789
Content-Transfer-Encoding: base64
Content-Type: image/jpeg
```

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

Protocolli di accesso Mail



- SMTP: consegna/memorizzazione al server del destinatario
- Protocollo di accesso alla Mail: recupero dal server
 - POP: Post Office Protocol [RFC 1939]
 - autorizzazione (agent <->server) e download
 - IMAP: Internet Mail Access Protocol [RFC 1730]
 - Maggiori caratteristiche (e complessità)
 - Manipolazione dei messaggi archiviati sul server
 - HTTP: Hotmail , Yahoo! Mail, etc.

protocollo POP3

fase di autorizzazione

- comandi del client:
 - **user**: declare username
 - **pass**: password
- risposte del server:
 - +OK
 - -ERR

fase di transazione, client:

- **list**: elenca i messaggi
- **retr**: recupera i messaggi dal loro numero
- **dele**: delete
- **quit**

```
S: +OK POP3 server ready
C: user colace
S: +OK
C: pass abbassoilprof
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 2 contents>
S: .
C: dele 2
C: quit
S: +OK POP3 server signing off
```