INTRODUZIONE ALLE APPLICAZIONI WEB

Roberto Basili
Croce Danilo

Corso di Basi di Dati
a.a. 2017/18

Introduzione

- Internet e WWW: Concetti di base
- Formati di dati per il Web
  - HTML, XML, DTD
- Architetture a tre livelli nel Web
- Il livello di presentazione
  - Moduli HTML: GET e POST HTTP, codifica di URL; Javascript;
- Il livello intermedio
  - CGI, application server, servlets, JavaServerPages, passaggio di argomenti, Gestione dello stato (cookie)

Internet & WWW

- Definizioni di base
- Architettura ed Organizzazione generale
- Protocollo di rete: TCP/IP
- L'identificazione delle risorse
- I servizi di rete
  - FTP
  - SMTP
  - HTTP

Reti di Calcolatori

- Una rete di calcolatori e' un insieme di sistemi di elaborazione collegati tra loro mediante una rete di comunicazione

  - Reti di calcolatori
  - Sistemi Distribuiti

Reti di Calcolatori

- Obbiettivi
  - Condivisione delle risorse
  - Comunicazione tra utenti degli elaboratori
  - Maggiore Affidabilità
  - Abbattimento dei costi di manutenzione, aggiornamento delle strutture di calcolo
  - Maggiore scalabilità

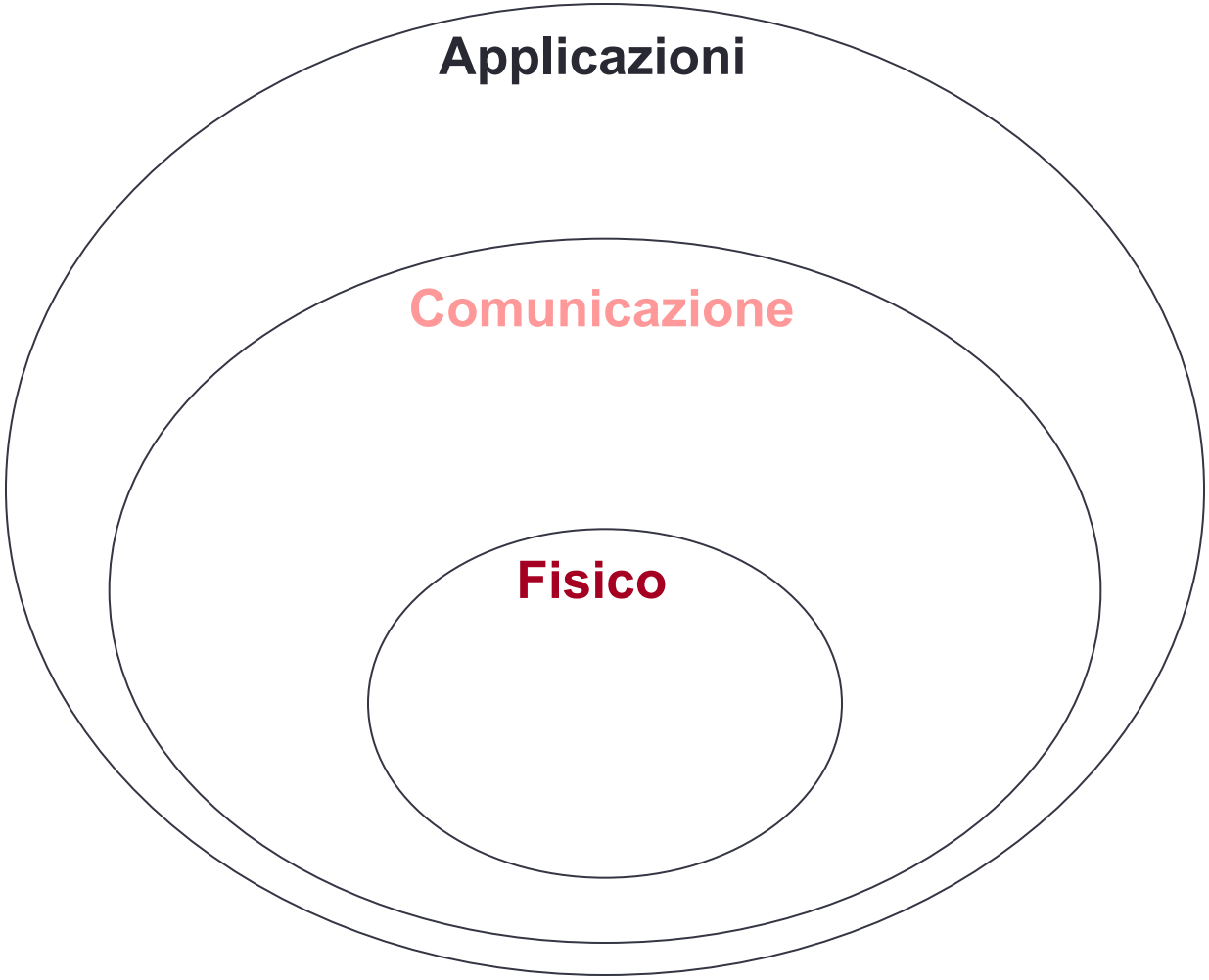
Reti di Calcolatori

- Una rete di calcolatori richiede un insieme di strumenti hardware e software necessari al suo funzionamento (Requisiti),

- Requisiti HW
  - la infrastruttura fisica di collegamento (rete)
  - l'insieme dei dispositivi locali ad ogni elaboratore che ne rendano visibile la rete
- Requisiti SW
  - protocolli e software di comunicazione
  - software applicativo ("sopra" al sw di comunicazione)

Architettura dei Servizi di Rete

- Livelli di gestione della comunicazione

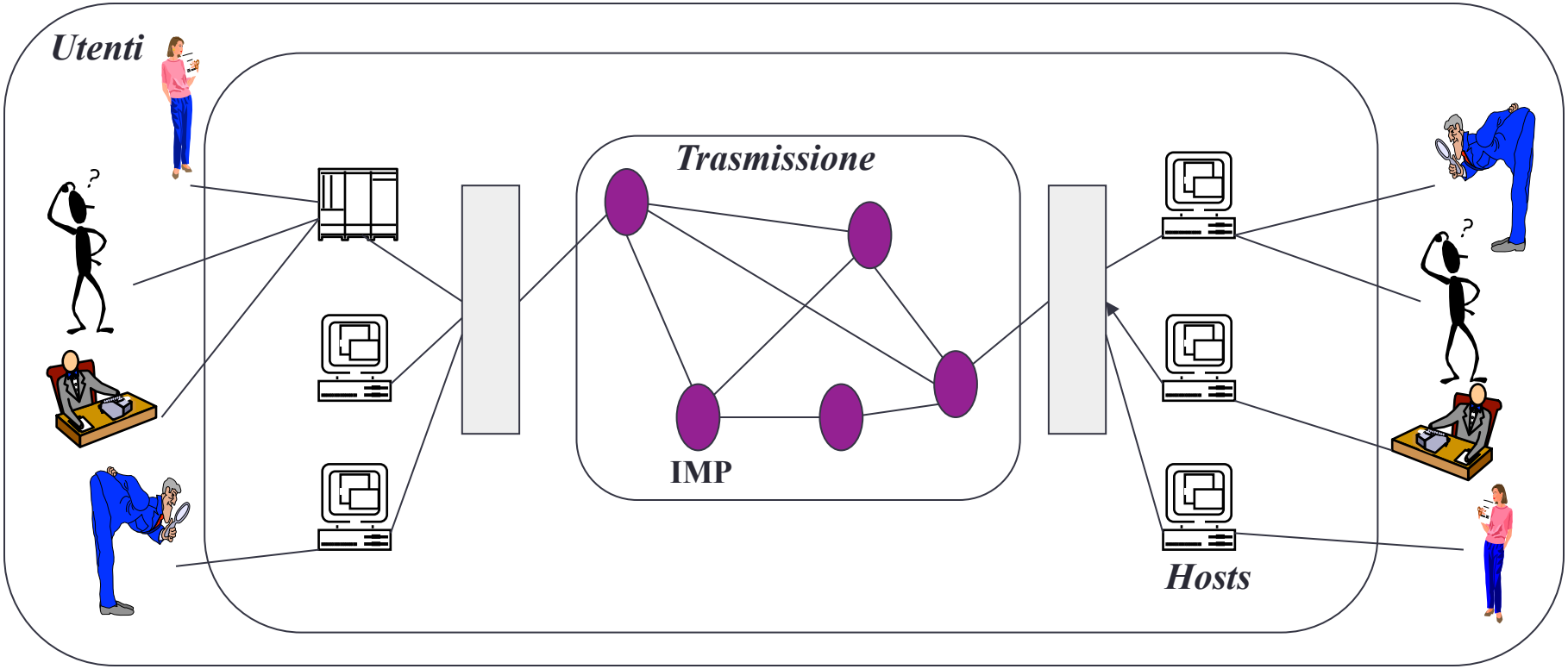


Infrastrutture

- Il Livello Fisico
  - Componenti di Connessione
    - doppino telefonico, cavo coassiale, fibre ottiche
  - Tecnologie di Trasmissione
    - Sincrone vs. Asincrone
    - Half-Duplex, Full-Duplex,
    - Multiplexing (TDM, FDM, STD)
  - Tecnologie di Rete
    - …

Infrastrutture

- Il Livello Fisico
  - Tecnologie di Rete

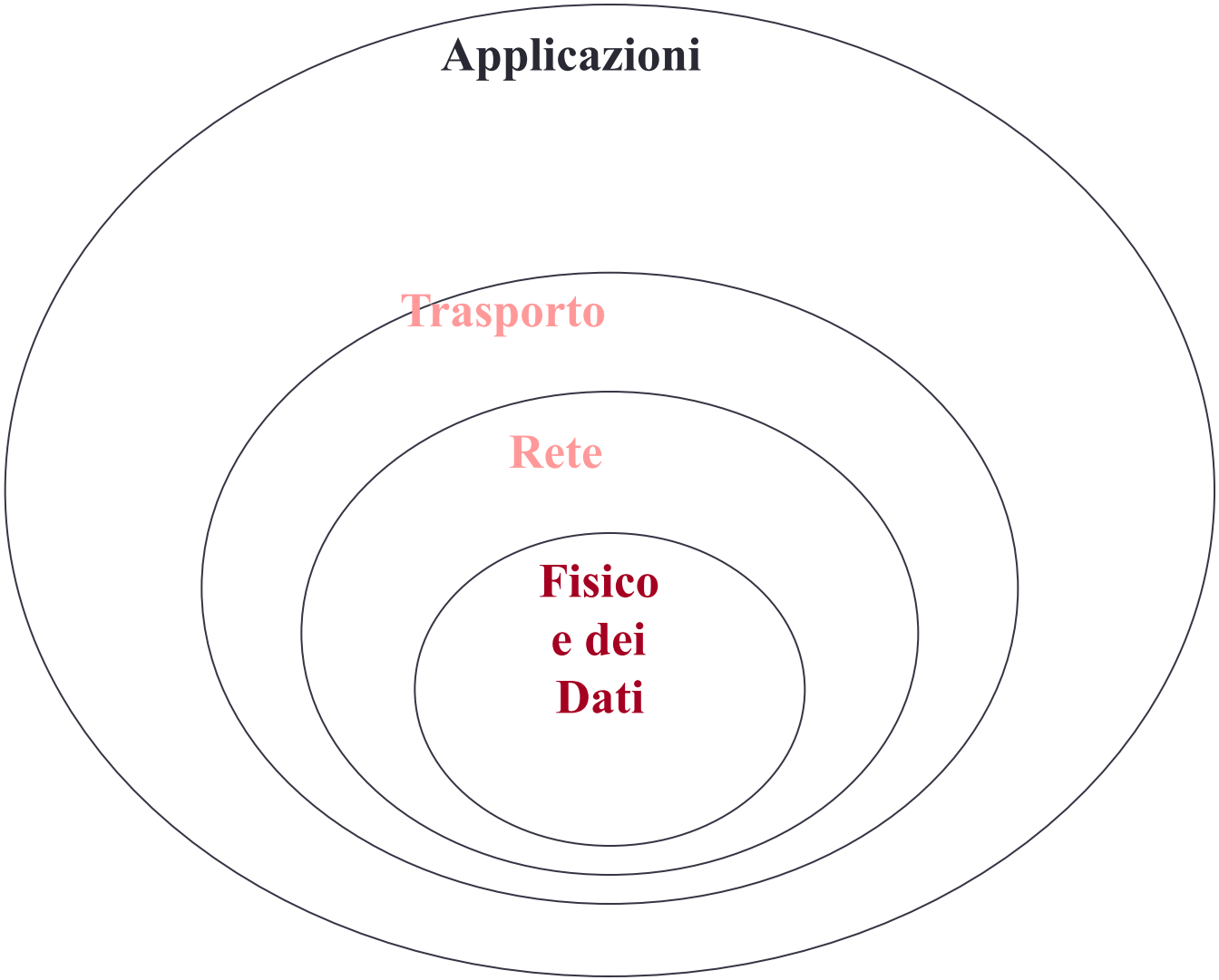


Protocolli di Comunicazione

- Costituiscono le *convenzioni condivise* dai sistemi appartenenti ad una rete per lo scambio di informazioni
  - ISO/OSI
  - TCP/IP

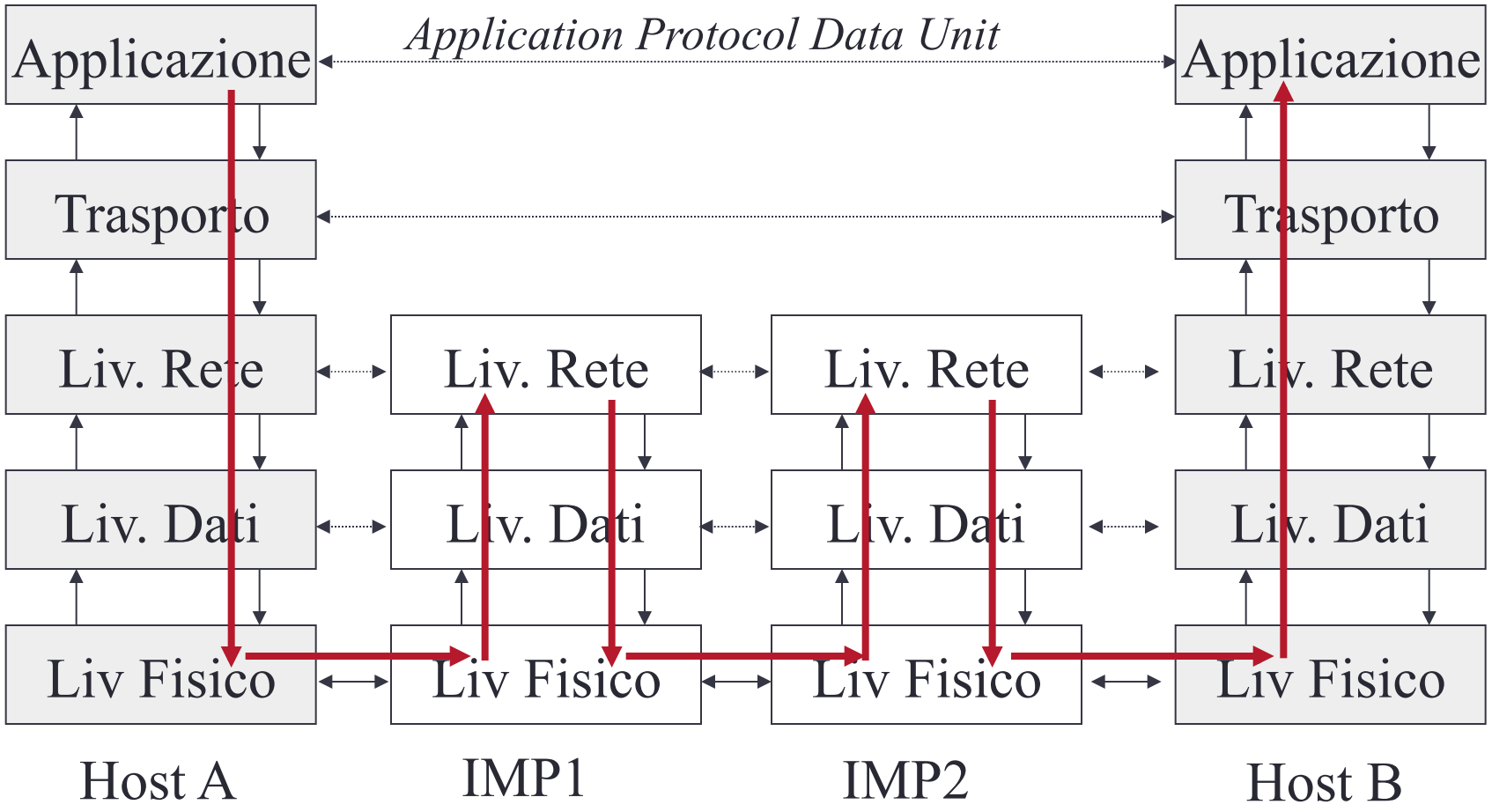
Architettura dei Servizi di Rete

- Livelli protocollo TCP/IP



Architettura dei Servizi di Rete

- Livelli in un protocollo



Architettura dei Servizi di Rete

- Livelli e instradamento
  - L'instradamento avviene mediante la decomposizione dei dati di ogni livello in pacchetti e mediante l'arricchimento dei pacchetti con estensioni dette header
  - Gli IMP che usano
    - un livello sono detti repeater
    - i primi due livelli (Fisico e Dati) sono detti router
    - i primi tre livelli sono detti bridge
    - tutti i livelli sono detti gateway

Le Applicazioni (o Servizi) di Rete

- FTP, File Transfer Protocol
- Telnet
- E-Mail:
  - SMTP
  - Applicazioni (Eudora)
- Il WWW (HTTP)

Il protocollo TCP/IP

- Transmission Control Protocol/Internet Protocol
- E' una versione analoga al protocollo ISO/OSI ma piu' semplice
- La sua fortuna e' soprattutto dovuta al successo della evoluzione di ARPANET in Internet
- Si occupa essenzialmente di controllare:
  - la Localizzazione della informazione
  - la Trasmissione

Il protocollo TCP/IP

- Localizzazione della informazione
  - Ogni host della rete e' caratterizzato da un indirizzo, detto indirizzo IP
  - Indirizzo univoco di 32 bit organizzati in 4 parole (0:255)
    - es. 150.100.20.17
  - Classe A:

Il protocollo TCP/IP

- Localizzazione della informazione
  - L'indirizzo IP

    - se **locale,** viene (in *broadcast*) inviato a tutte le macchine della rete
    - se **esterno** viene inviato ad un IMP (il *router* o *gateway*) che si occupa di instradare i dati

Il protocollo TCP/IP

- Localizzazione della informazione
  - Ad un indirizzo IP in genere corrisponde un nome logico, che determina un indirizzamento simbolico
  - E' un servizio molto generale (usato da molti altri servizi, es FTP o WWW) denominato ***Domain Name Service*** (**DNS**)
  - L'indirizzamento e' regolato da un *host* (il *Domain Name Server*) che si assume la responsabilita' di tradurre gli IP in nomi logici (risoluzione)

Il protocollo TCP/IP

- Il DNS
  - Ogni host della rete (di reti) ammette un nome simbolico (es. **`gaudi.info.uniroma2.it`**)
  - La struttura riproduce la struttura gerarchica della rete
    - *domini* < *sottodominio* < *sottosottodominio* < ...
  - L'ordine riproduce anche la struttura di un IP, ma in modo inverso

    es. **`160.80.65.54`**⇔**`gaudi.info.uniroma2.it`**

Il protocollo TCP/IP

- Il DNS
  - Le reti di primo livello sono definite da autorita' internazionali (Internet Assigned Number Authority, IANA)
  - Le sottoreti sono determinate in genere a livello nazionale (secondo convenzioni)
  - Esempi illustri di reti di primo livello per motivi storici (Arpanet) sono
    - EDU, COM, ORG, MIL,GOV ...

Il protocollo TCP/IP

- Il processo di risoluzione del DNS
  - Il processo di risoluzione dei nomi e' anch'esso *distribuito*

```
1. Se l'IP di un nome logico e' disponibile al
   DNS locale allora usalo subito
2. Altrimenti richiedi l'IP al DNS di livello
   superiore
```

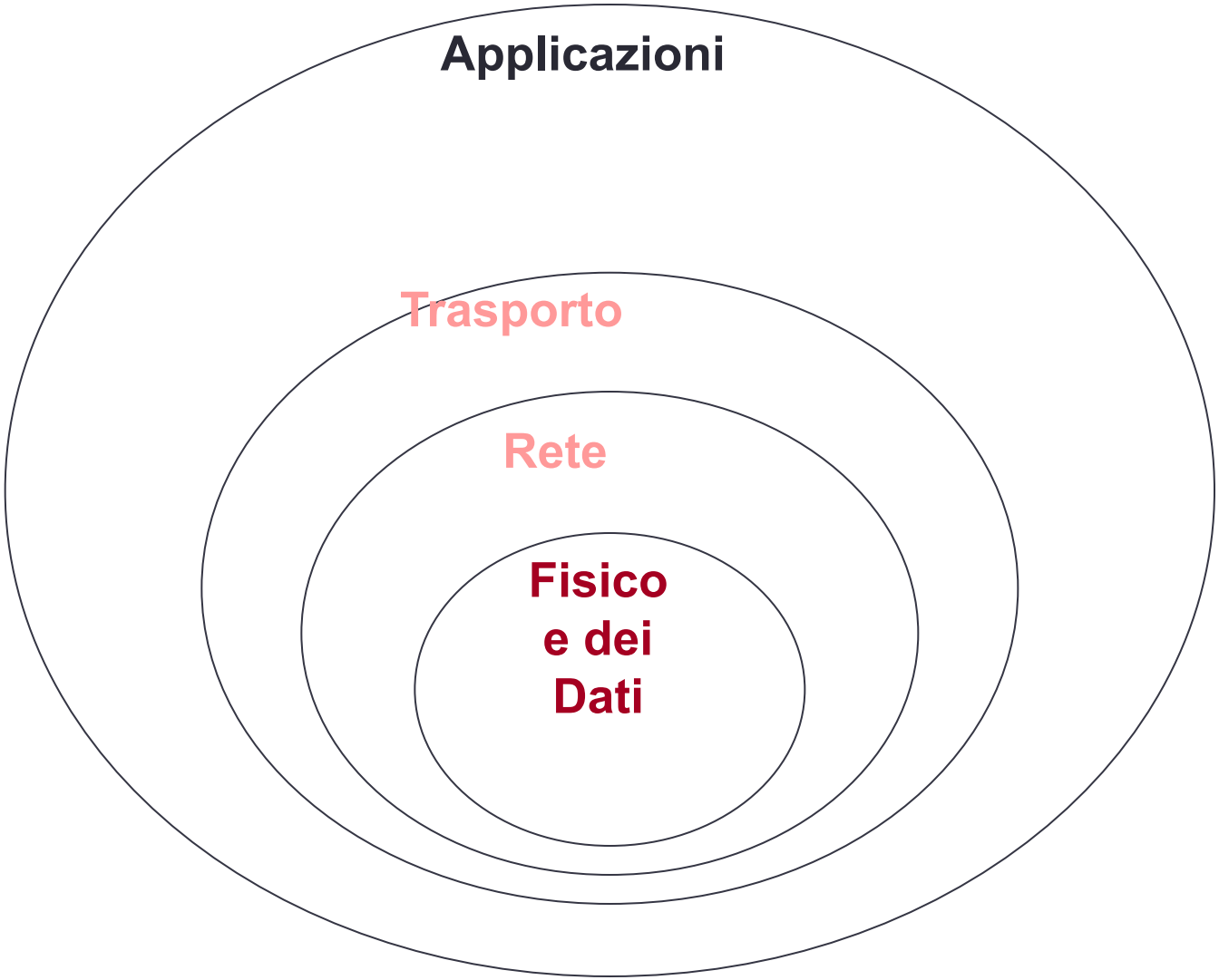
(*oss*. Il DNS contattato al passo 2 fa partire il processo da 1 e quindi tale processo puo' propagarsi quanto necessario attraverso la rete).

Il protocollo TCP/IP

- Trasmissione
  - Internet e' una rete a commutazione di pacchetto
  - Il protocollo di gestione della trasmissione dei dati e' il **TCP**, *Transmission Control Protocol*

Architettura dei Servizi di Rete

- Livelli protocollo TCP/IP



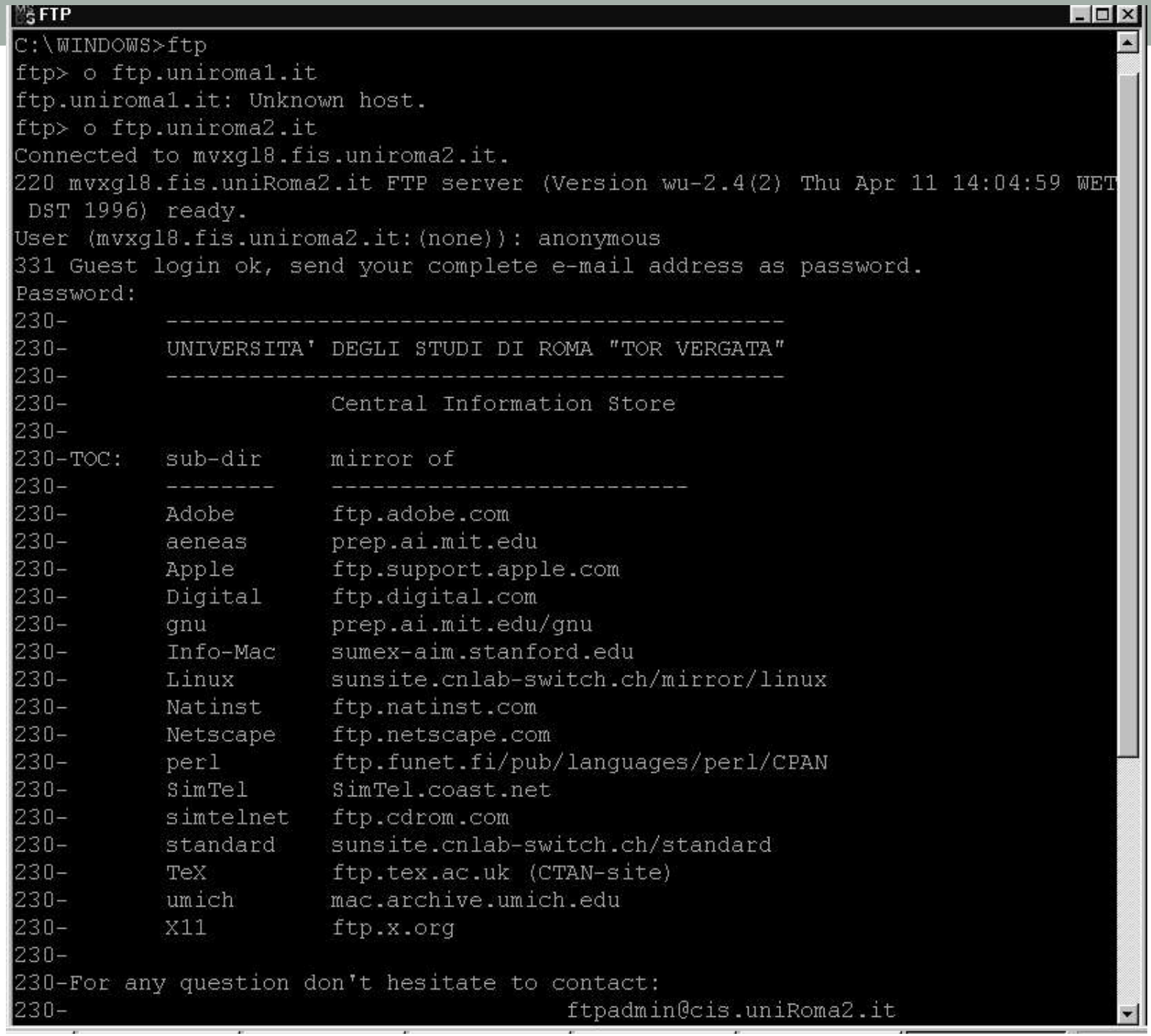
Il protocollo TCP/IP

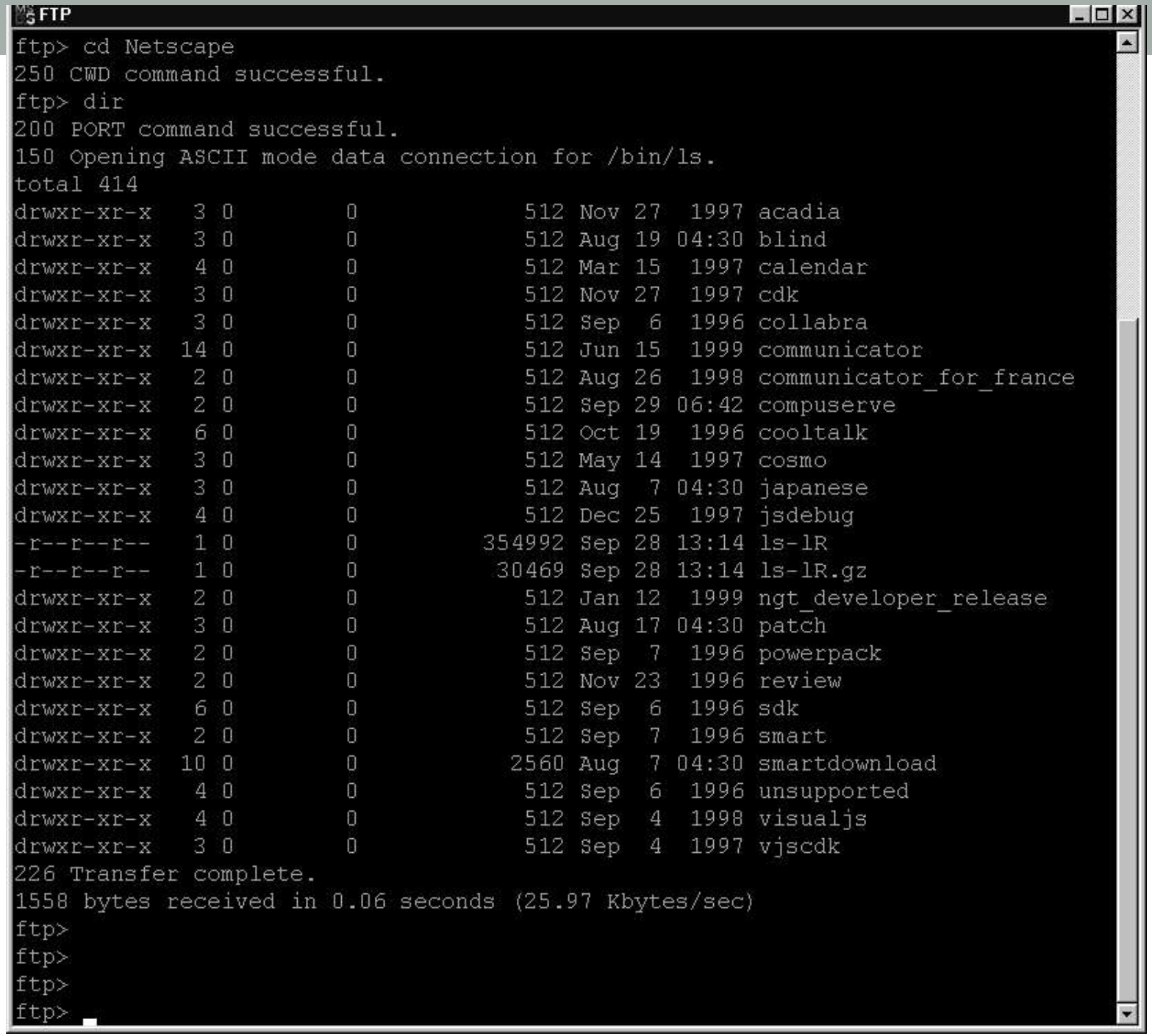
- Il protocollo TCP si occupa di
  - suddividere l'insieme dei dati trasmessi in pacchetti
  - aggiungere ad ogni pacchetto il corrispondente *header,* ed instradarlo
  - ricomporre il dato iniziale alla ricezione
  - mantenere la consistenza della trasmissione

I Servizi di Rete

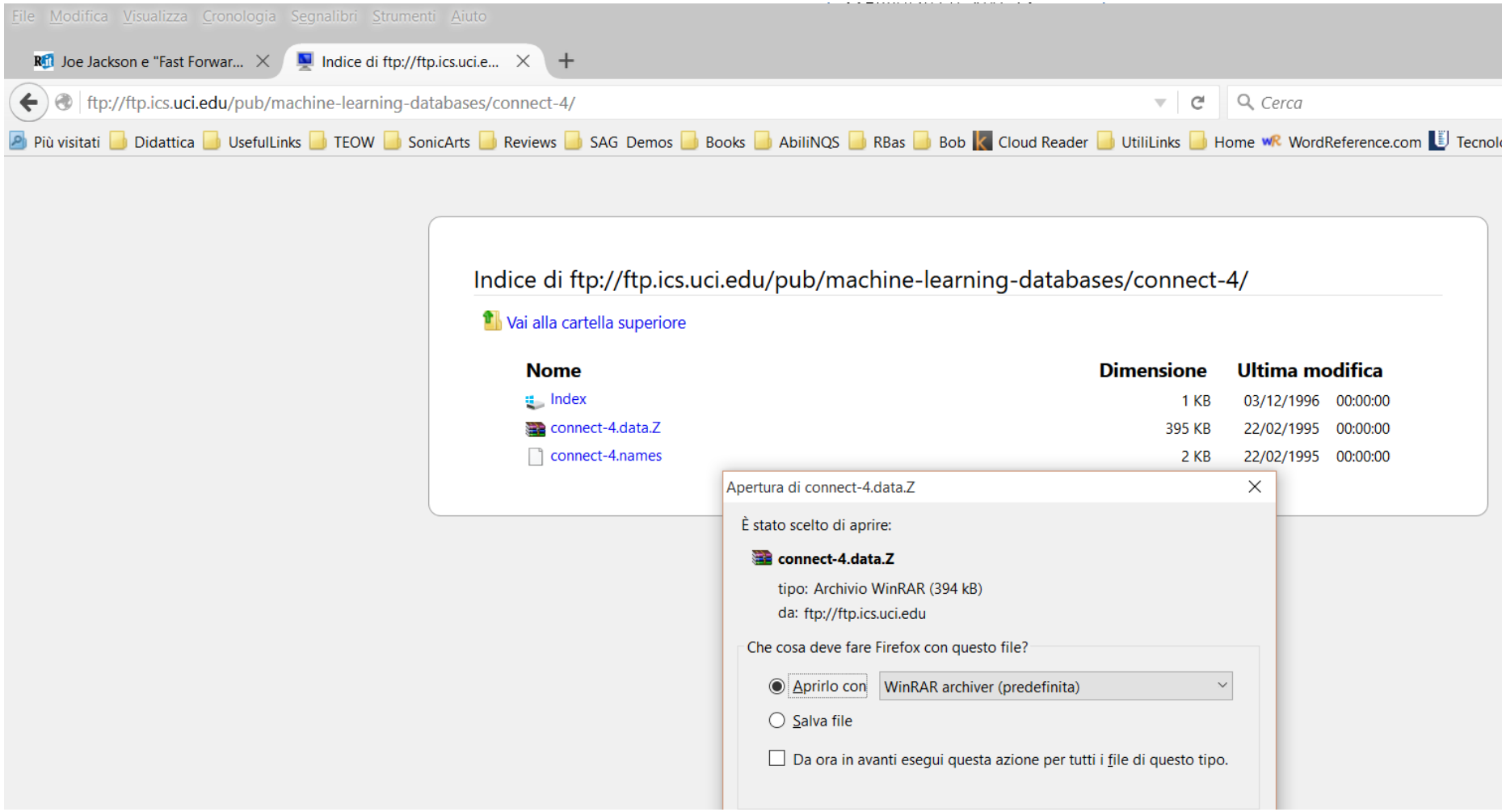
- *File Transfer Protocol* (**FTP**)
- E' il primo servizio introdotto in Arpanet
- Supporta il trasferimento di files tra host eterogenei per hardware e sistema operativo

- Funzionalita' principali
  - `o(open) NOME_HOST`
  - `u(ser) USER_NAME`
  - `set (mode) (I/ascii B/binary)`
  - `(m)get NOME_FILE`


FTP
C:\WINDOWS>ftp
ftp> o ftp.uniroma1.it
ftp.uniroma1.it: Unknown host.
ftp> o ftp.uniroma2.it
Connected to mvxgl8.fis.uniroma2.it.
220 mvxgl8.fis.uniRoma2.it FTP server (Version wu-2.4(2) Thu Apr 11 14:04:59 WET DST 1996) ready.
User (mvxgl8.fis.uniroma2.it:(none)): anonymous
331 Guest login ok, send your complete e-mail address as password.
Password:
230-
230- UNIVERSITA' DEGLI STUDI DI ROMA "TOR VERGATA"
230-
230- Central Information Store
230-
230-TOC: sub-dir mirror of
230- Adobe ftp.adobe.com
230- aeneas prep.ai.mit.edu
230- Apple ftp.support.apple.com
230- Digital ftp.digital.com
230- gnu prep.ai.mit.edu/gnu
230- Info-Mac sumex-aim.stanford.edu
230- Linux sunsite.cnlab-switch.ch/mirror/linux
230- Natinst ftp.natinst.com
230- Netscape ftp.netscape.com
230- perl ftp.funet.fi/pub/languages/perl/CPAN
230- SimTel SimTel.coast.net
230- simtelnet ftp.cdrom.com
230- standard sunsite.cnlab-switch.ch/standard
230- TeX ftp.tex.ac.uk (CTAN-site)
230- umich mac.archive.umich.edu
230- X11 ftp.x.org
230-
230-For any question don't hesitate to contact:
230- ftpadmin@cis.uniRoma2.it


FTP
ftp> cd Netscape
250 CWD command successful.
ftp> dir
200 PORT command successful.
150 Opening ASCII mode data connection for /bin/ls.
total 414
drwxr-xr-x   3 0   0   512 Nov 27 1997 acadia
drwxr-xr-x   3 0   0   512 Aug 19 04:30 blind
drwxr-xr-x   4 0   0   512 Mar 15 1997 calendar
drwxr-xr-x   3 0   0   512 Nov 27 1997 cdk
drwxr-xr-x   3 0   0   512 Sep 6 1996 collabra
drwxr-xr-x 14 0   0   512 Jun 15 1999 communicator
drwxr-xr-x   2 0   0   512 Aug 26 1998 communicator_for_france
drwxr-xr-x   2 0   0   512 Sep 29 06:42 compuserve
drwxr-xr-x   6 0   0   512 Oct 19 1996 cooltalk
drwxr-xr-x   3 0   0   512 May 14 1997 cosmo
drwxr-xr-x   3 0   0   512 Aug 7 04:30 japanese
drwxr-xr-x   4 0   0   512 Dec 25 1997 jsdebug
-r--r--r--   1 0   0   354992 Sep 28 13:14 ls-lR
-r--r--r--   1 0   0   30469 Sep 28 13:14 ls-lR.gz
drwxr-xr-x   2 0   0   512 Jan 12 1999 ngt_developer_release
drwxr-xr-x   3 0   0   512 Aug 17 04:30 patch
drwxr-xr-x   2 0   0   512 Sep 7 1996 powerpack
drwxr-xr-x   2 0   0   512 Nov 23 1996 review
drwxr-xr-x   6 0   0   512 Sep 6 1996 sdk
drwxr-xr-x   2 0   0   512 Sep 7 1996 smart
drwxr-xr-x 10 0   0   2560 Aug 7 04:30 smartdownload
drwxr-xr-x   4 0   0   512 Sep 6 1996 unsupported
drwxr-xr-x   4 0   0   512 Sep 4 1998 visualjs
drwxr-xr-x   3 0   0   512 Sep 4 1997 vjscdk
226 Transfer complete.
1558 bytes received in 0.06 seconds (25.97 Kbytes/sec)
ftp>
ftp>
ftp>
ftp>

FTP & Web browsers



Uniform Resource Identifiers

- Uniformano lo schema dei nomi per identificare le *risorse* su Internet
- Una risorsa può essere qualunque cosa:
  - Index.html
  - Canzone.mp3
  - Immagine.jpg
- Esempi di URI:

  http://www.cs.wisc.edu/~dbbokk/index.html

  mailto:webmaster@bookstore.com

Struttura degli URI

http://www.cs.wisc.edu/~dbbokk/index.html

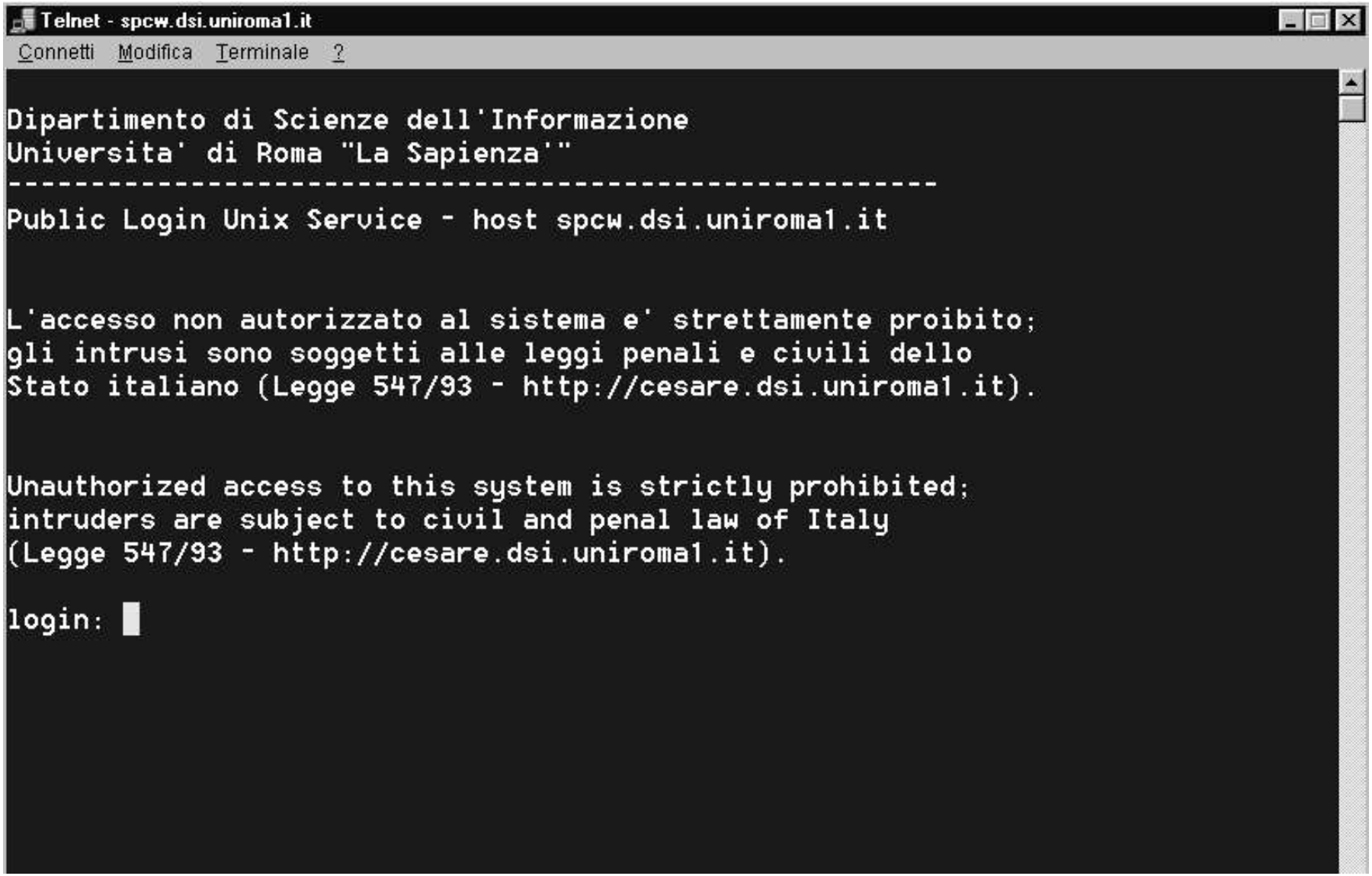
Un URI ha tre parti:

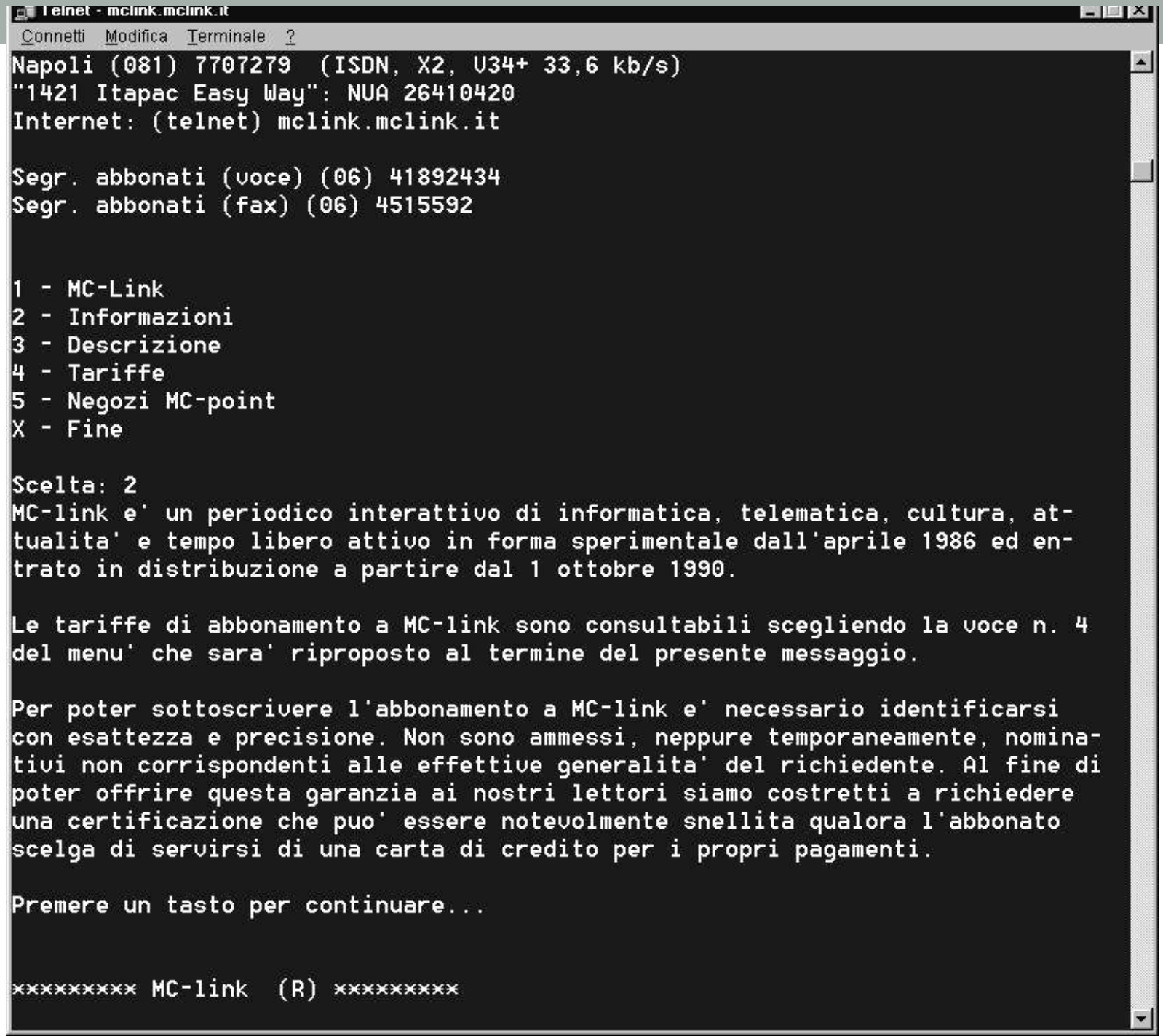
- Schema del nome (**http**)
- Nome del computer host (**www.cs.wisc.edu**)
- Nome della risorsa (**~dbbokk/index.html**)

- Le URL sono un sottoinsieme degli URI

I Servizi di Rete (2)

- Telnet
- Simula sulla macchina “client” un terminale della macchina “server”
- Rende disponibili tutte le funzionalita’ dello shell dei comandi del SO della macchina “server”


Telnet - spcw.dsi.uniroma1.it
Connetti Modifica Terminale ?
Dipartimento di Scienze dell'Informazione
Universita' di Roma "La Sapienza"
Public Login Unix Service - host spcw.dsi.uniroma1.it
L'accesso non autorizzato al sistema e' strettamente proibito;
gli intrusi sono soggetti alle leggi penali e civili dello
Stato italiano (Legge 547/93 - http://cesare.dsi.uniroma1.it).
Unauthorized access to this system is strictly prohibited;
intruders are subject to civil and penal law of Italy
(Legge 547/93 - http://cesare.dsi.uniroma1.it).
login:


Telnet - mclink.mclink.it
Connetti Modifica Terminale ?
Napoli (081) 7707279 (ISDN, X2, V34+ 33,6 kb/s)
"1421 Itapac Easy Way": NUA 26410420
Internet: (telnet) mclink.mclink.it
Segr. abbonati (voce) (06) 41892434
Segr. abbonati (fax) (06) 4515592
1 - MC-Link
2 - Informazioni
3 - Descrizione
4 - Tariffe
5 - Negozi MC-point
X - Fine
Scelta: 2
MC-link e' un periodico interattivo di informatica, telematica, cultura, attualita' e tempo libero attivo in forma sperimentale dall'aprile 1986 ed entrato in distribuzione a partire dal 1 ottobre 1990.
Le tariffe di abbonamento a MC-link sono consultabili scegliendo la voce n. 4 del menu' che sara' riproposto al termine del presente messaggio.
Per poter sottoscrivere l'abbonamento a MC-link e' necessario identificarsi con esattezza e precisione. Non sono ammessi, neppure temporaneamente, nominativi non corrispondenti alle effettive generalita' del richiedente. Al fine di poter offrire questa garanzia ai nostri lettori siamo costretti a richiedere una certificazione che puo' essere notevolmente snellita qualora l'abbonato scelga di servirsi di una carta di credito per i propri pagamenti.
Premere un tasto per continuare...
********** MC-link (R) **********

I Servizi di Rete

- Posta Elettronica (E-mail)
- Nasce come servizio per scambiare messaggi di testo tra utenti di host della rete
- E’ basato sul protocollo specifico SMTP (Simple Mail transfer Protocol)
- Per ogni messaggio un file testuale viene generato contenente il testo e dei meta descrittori

Hypertext Transfer Protocol

- Cos'è un protocollo di comunicazione?
  - Insieme di standard che definisce la struttura dei messaggi
  - Esempi: TCP, IP, HTTP
- Che succede se fate click su http://www.cs.wisc.edu/~dbbokk/index.html?
  - Il client (browser web) manda una richiesta HTTP al server
  - Il server riceve la richiesta e risponde
  - Il client riceve la risposta; invia altre richieste

HTTP (segue)

dal client al server :

GET ~/index.html HTTP/1.1

User-agent: Mozilla/4.0

Accept: text/html, image/gif, image/jpeg

il server risponde :

HTTP/1.1 200 OK

Date: Mon, 04 Mar 2002 12:00:00 GMT

Server: Apache/1.3.0 (Linux)

Last-Modified: Mon, 01 Mar 2002 09:23:24 GMT

Content-Length: 1024

Content-Type: text/html

<HTML> <HEAD></HEAD>

<BODY>

<h1>Libreria Internet di Barns e Nobble </h1>

Il nostro catalogo :

<h3>Scienza</h3>

Natura della legge fisica

...

Struttura del protocollo HTTP

- Richieste HTTP
- Linea di richiesta: GET ~/index.html HTTP/1.1
  - GET: campo del metodo HTTP (valori possibili sono GET e POST, più avanti)
  - ~/index.html: campo URI
  - HTTP/1.1: campo della versione HTML
- Tipo di client: User-agent: Mozilla/4.0
- Che tipi di documenti verranno accettati dal client:

  Accept: text/html, image/gif, image/jpeg

Struttura del protocollo HTTP (segue)

Risposte HTTP

- Linea di stato: HTTP/1.1 200 OK
  - Versione HTTP: HTTP/1.1
  - Codice di stato: 200
  - Messaggio del server: OK
  - Combinazioni comuni di codice di stato/messaggio del server:
    - 200 OK: la richiesta ha avuto successo
    - 400 Bad Request: il server non ha potuto soddisfare la richiesta
    - 404 Not Found: l'oggetto richiesto non esiste sul server
    - 505 HTTP Version not Supported
- Data di creazione dell'oggetto:
- Last-Modified: Mon, 01 Mar 2002 09:23:24 GMT
- Numero di bytes spediti: Content-Length: 1024
- Tipo di oggetto che viene spedito: Content-Type: text/html
- Altre informazioni quali il tipo di server, l'ora del server, etc.

Formati di dati per Web

- HTML
  - Il linguaggio di **presentazione** per Internet
- XML
  - Un **modello di dati** gerarchico auto-descrittivo
- DTD
  - Schemi standardizzati per XML
- XSLT (non trattato nel corso)

HTML: un esempio

```
<HTML>
  <HEAD></HEAD>
  <BODY>
  <h1>Libreria Internet Barns & Nobble </h1>
  Il nostro inventario :

  <h3>Scienza</h3>
  <b>Natura della legge fisica </b>
  <UL>
     <LI>Autore: Richard Feynman</LI>
  <LI>Pubblicato nel 1980</LI>
  <LI>Copertina dura</LI>
  </UL>

  <h3>Fiction</h3>
  <b>Aspettando il Mahatma</b>
  <UL>
  <LI>Autore: R.K. Narayan</LI>
  <LI>Pubblicato nel 1981</LI>
  </UL>
  <b>L'insegnante di Inglese</b>
  <UL>
  <LI>Autore:  R.K. Narayan</LI>
  <LI>Pubblicato nel 1980</LI>
  <LI>Tascabile</LI>
  </UL>

  </BODY>
</HTML>
```

HTML: breve introduzione

- L'HTML è un linguaggio di marcatura
- I comandi sono tag
  - Tag di inizio e di fine
  - Esempi
    - <HTML>...</HTML>
    - <UL>...</UL>
- Molti editor generano automaticamente l'HTML direttamente dal documento (ad esempio Microsoft Word ha una funzione "Salva come HTML")

HTML: esempio di comandi

- <HTML>
- <UL>: lista non ordinata
- <LI>: elemento di una lista
- <h1>: intestazione più grande
- <h2>: intestazione di secondo livello, analogamente <h3>, <h4>
- <B>Title</B>: grassetto

XML: un esempio

```
<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
<LISTALIBRI>
<LIBRO GENERE="Scienza" FORMATO="Copertina dura">
   <AUTORE>
            <NOME>Richard</NOME>
            <COGNOME>Feynman</COGNOME>
   </AUTORE>
   <TITOLO>Natura della legge fisica</TITOLO>
   <PUBBLICATO>1980</PUBBLICATO>
</LIBRO>
<LIBRO GENERE="Fiction">
   <AUTORE>
            <NOME>R.K.</NOME>
            <COGNOME>Narayan</COGNOME>
   </AUTORE>
   <TITOLO>Aspettando il Mahatma</TITOLO>
   <PUBBLICATO>1981</PUBBLICATO>
</LIBRO>
<LIBRO GENERE="Fiction">
   <AUTORE>
            <NOME>R.K.</NOME>
            <COGNOME>Narayan</COGNOME>
   </AUTORE>
   <TITOLO>L'insegnante di inglese</TITOLO>
   <PUBBLICATO>1980</PUBBLICATO>
</LIBRO>
</LISTALIBRI>
```

XML: eXtensible Markup Language

- Language
  - Un modo di comunicare informazione
- Markup
  - Note o meta-dati che descrivono i dati o il linguaggio
- Extensible
  - Capacità illimitata di definire nuovi linguaggi o insiemi di dati

XML – Qual è il punto?

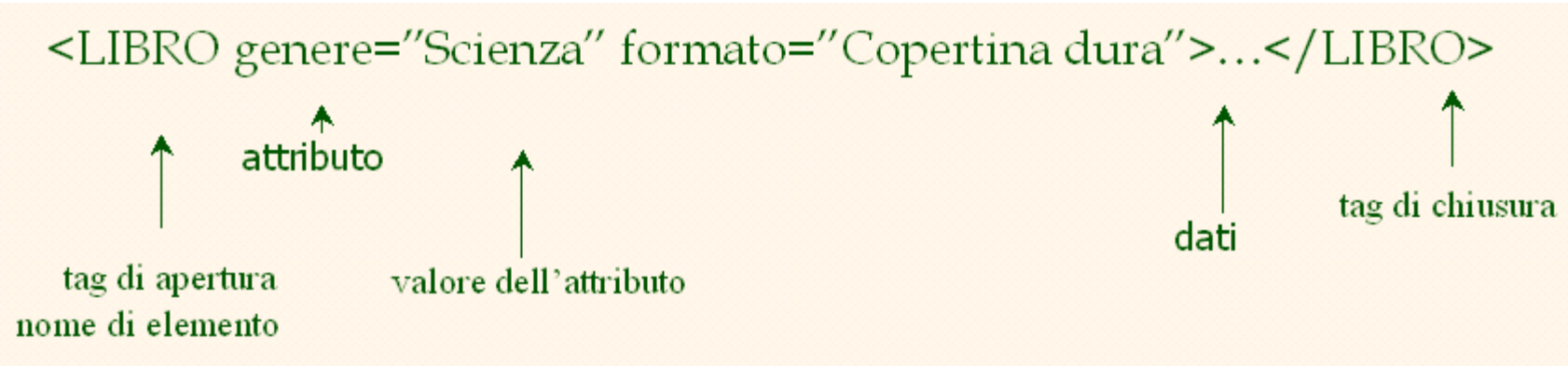
- Si possono includere i propri dati e una descrizione di ciò che tali dati rappresentano
  - Utile per definire il proprio linguaggio o protocollo personale
- Esempio: Chemical Markup Language

```
<molecola>
  <peso>234.5</peso>
  <Spettro>...</Spettro>
  <Numeri>...</Numeri>
</molecola>
```

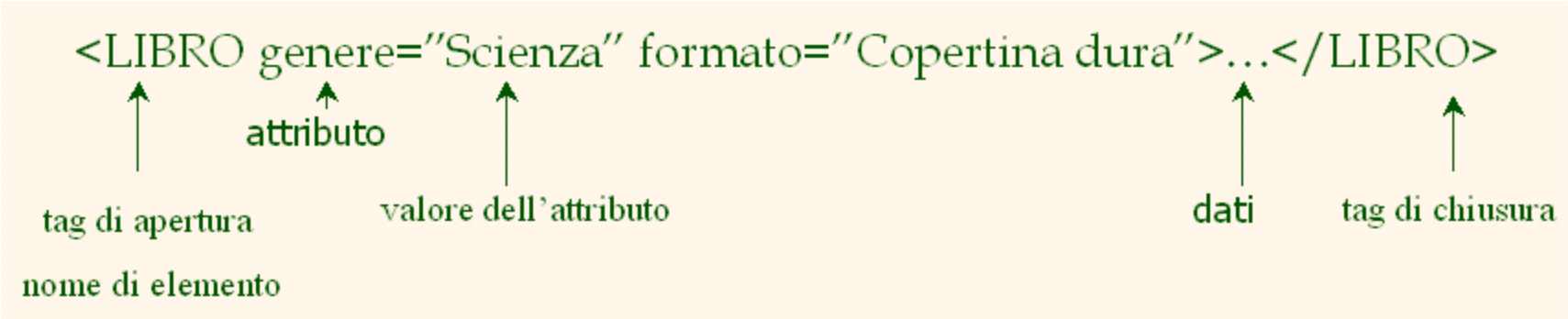
- Obiettivi del progetto XML:
  - L'XML dovrebbe essere compatible con SGML
  - La scrittura di programmi che elaborano documenti XML dovrebbe essere un compito semplice
  - Il progetto dovrebbe essere formale e preciso

XML – Struttura

- XML: punto di incontro di SGML e HTML
- L'XML somiglia all'HTML
- L'XML è una gerarchia di tag definiti dall'utente chiamati elementi con attributi e dati
- I dati sono descritti dagli elementi, gli elementi sono descritti dagli attributi

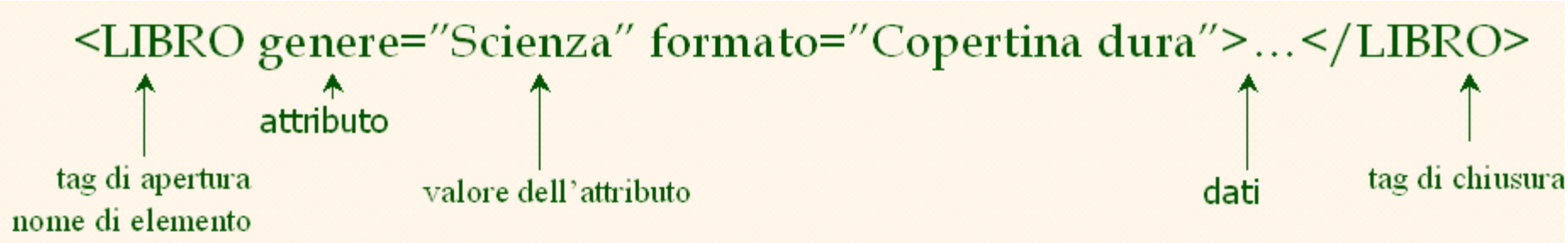


XML – Attributi



- L'XML è sensibile alle maiuscole e agli spazi
- I nomi dei tag di apertura e chiusura devono essere identici
- Tag di apertura: "<" + nome elemento + ">"
- Tag di chiusura: "</" + nome elemento + ">"
- Elementi vuoti non hanno dati e non hanno tag di chiusura:
  - cominciano con un "<" e finiscono con un "/>"
- <LIBRO/>

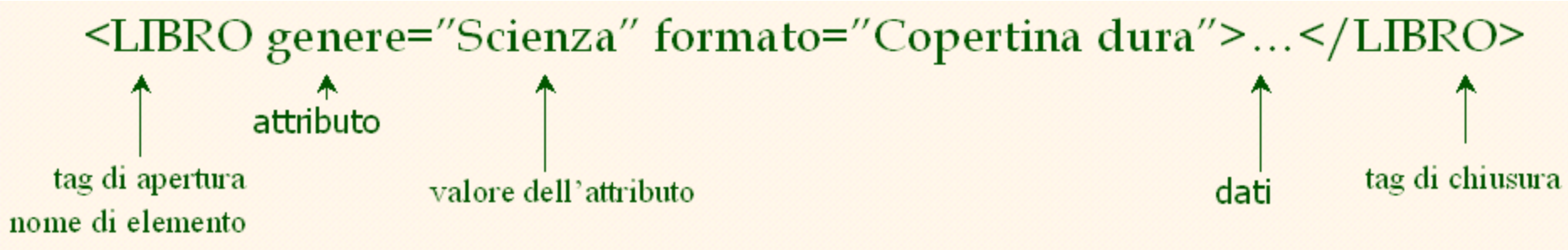
XML – Attributi



elementi

- Ci possono essere zero o più attributi in ogni elemento; ciascuno ha la forma
  - Nome_attributo='valore_attributo'
    - Non ci sono spazi tra il nome e "="
    - I valori degli attributi devono essere racchiusi dai caratteri ' oppure "
- Attributi multipli sono separati da spazi bianchi (uno o più spazi o tabulazioni)

Dati e commenti



- I dati XML sono qualunque informazione tra un tag di apertura e un tag di chiusura
- I dati XML non devono contenere i caratteri '<' oppure '>'
- Commenti:

  <!- commento ->

Annidamento e gerarchia

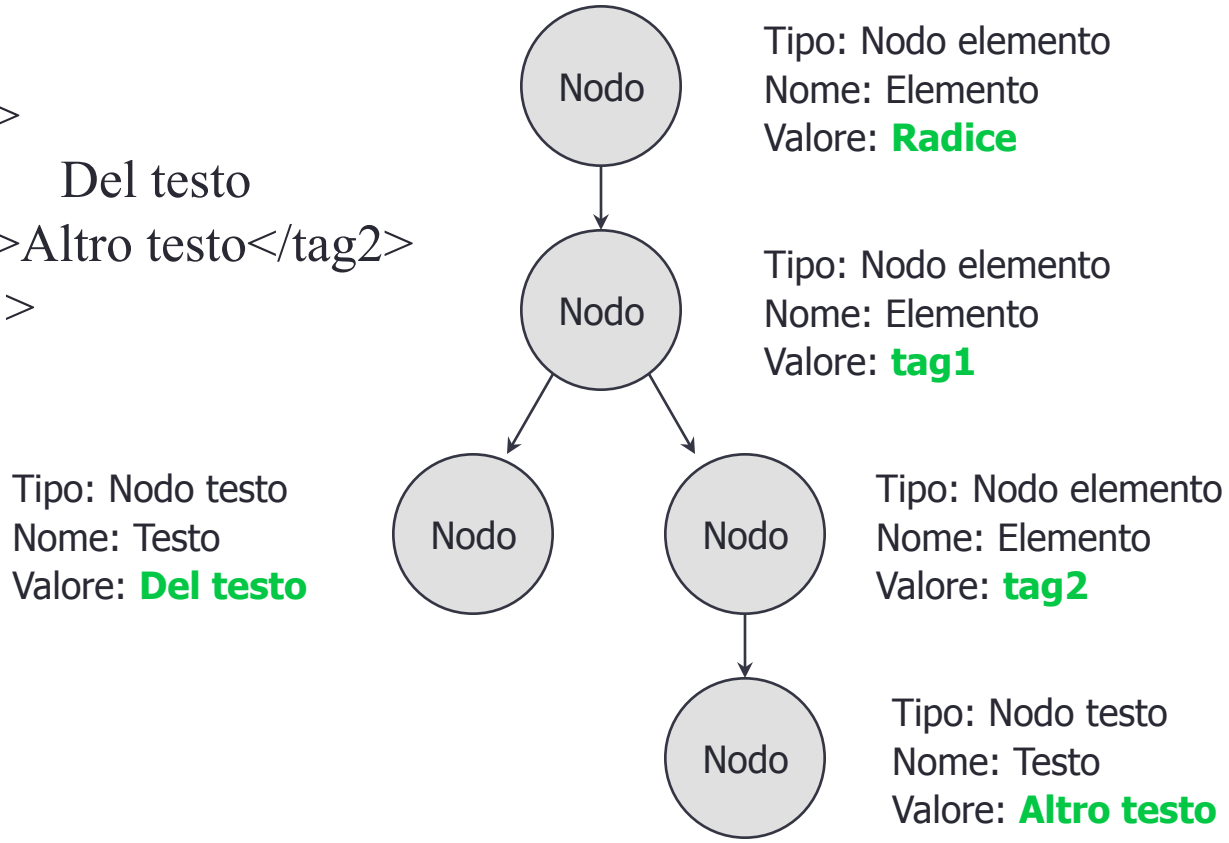
- I tag XML possono essere annidati in una gerarchia ad albero
- I documenti XML possono avere un solo tag radice
- Tra un tag di apertura e un tag di chiusura si possono inserire:
  1. dati
  2. altri elementi
  3. una combinazione di dati ed elementi

```
<radice>
    <tag1>
            Del testo
    <tag2>Dell'altro</tag2>
    </tag1>
</radice>
```

Memorizzazione

- La memorizzazione viene effettuata proprio come in un albero n-ario (DOM)

```
<radice>
    <tag1>
        Del testo
    <tag2>Altro testo</tag2>
    </tag1>
</radice>
```



DTD - Document Type Definition

- Un DTD è uno schema per i dati XML
- I protocolli e i linguaggi XMP possono essere standardizzati con file DTD
- Un DTD dice quali elementi e attributi sono obbligatori e quali opzionali
  - Definisce la struttura formale del linguaggio

DTD – Un esempio

```
<?xml version=’1.0’?>
<!ELEMENT Cesto (Ciliegia+, (Mela | Arancia)*)>
<!ELEMENT Ciliegia EMPTY>
   <!ATTLIST Ciliegia sapore CDATA #REQUIRED>
<!ELEMENT Mela EMPTY>
   <!ATTLIST Mela colore CDATA #REQUIRED>
<!ELEMENT Arancia EMPTY>
   <!ATTLIST Arancia provenienza ‘Florida’>
```

```
<Cesto>
        <Ciliegia sapore=’buono’/>
        <Mela colore=’rosso’/>
        <Mela colore=’verde’/>
</Cesto>
```

```
<Cesto>
        <Mela/>
        <Ciliegia sapore=’buono’/>
        <Arancia/>
</Cesto>
```

DTD - !ELEMENT

<!ELEMENT Cesto (Ciliegia+, (Mela | Arancia)*)>

Nome Figli

- !ELEMENT dichiara il nome di un elemento, e quali elementi figli dovrebbe avere
- Tipi di contenuto:
  - Altri elementi
  - #PCDATA (parsed character data)
  - EMPTY (nessun contenuto)
  - ANY (nessun controllo all'interno di questa struttura)
- Una espressione regolare

DTD - !ELEMENT (segue)

- Una espressione regolare ha la seguente struttura:
  - exp1, exp2, exp3, ..., expk: una lista di espressioni regolari
  - exp*: una espressione opzionale con zero o più occorrenze
  - exp+: una espressione opzionale con una o più occorrenze
  - exp1 | exp2 | ... | exp3: una disgiunzione di espressioni

DTD - !ATTLIST

<!ATTLIST Arancia provenienza CDATA #REQUIRED colore ‘arancione’>

!ATTLIST definisce una lista di attributi per un elemento

- Gli attributi possono essere di tipi diversi, possono essere obbligatori o opzionali, e possono avere valori predefiniti

DTD – Ben formato e valido

```
<?xml version='1.0'?>
<!ELEMENT Cesto (Ciliegia+)>
  <!ELEMENT Ciliegia EMPTY>
        <!ATTLIST Ciliegia sapore CDATA #REQUIRED>
```

Non ben formato

```
<Cesto>
        <Ciliegia sapore=buono>
</Cesto>
```

Ben formato ma non valido

```
<Lavoro>
        <Luogo>Casa</Luogo>
</Lavoro>
```

Ben formato e valido

```
<Cesto>
        <Ciliegia sapore='buono'/>
</Cesto>
```

XML e DTD

- Un numero sempre maggiore di DTD verrà sviluppato
  - MathML
  - Chemical Markup Language
- Permette rapidi scambi di dati con la stessa semantica
- Sono disponibili sofisticati linguaggi di interrogazione:
  - Xquery
  - Xpath

Argomenti del capitolo

- Internet: Concetti di base
- Formati di dati per il Web
  - HTML, XML, DTD
- Architetture a tre livelli e Web
- Il livello di presentazione
  - Moduli HTML: GET e POST HTTP, codifica di URL; Javascript; fogli di stile. XSLT
- Il livello intermedio
  - CGI, application server, servlets, JavaServerPages, passaggio di argomenti, mantenimento dello stato (cookie)

Componenti dei sistemi *"data-intensive"*

Tre tipi separati di funzionalità:

- gestione dei dati
- logica di applicazione
- presentazione

- L'architettura del sistema determina se queste tre componenti risiedono su un singolo sistema (tier) oppure se sono distribuite su diversi tier

Architettura a livello singolo

Tutte le funzionalità sono combinate in un singolo tier, generalmente un mainframe

- Accesso utente tramite terminali non intelligenti

Vantaggi :

- facilità di manutenzione e amministrazione

Svantaggi :

- Oggi gli utenti si aspettano interfacce utente di tipo grafico
- Il calcolo centralizzato di tutte le interfacce grafiche è troppo costoso per un singolo sistema

Architetture client-server

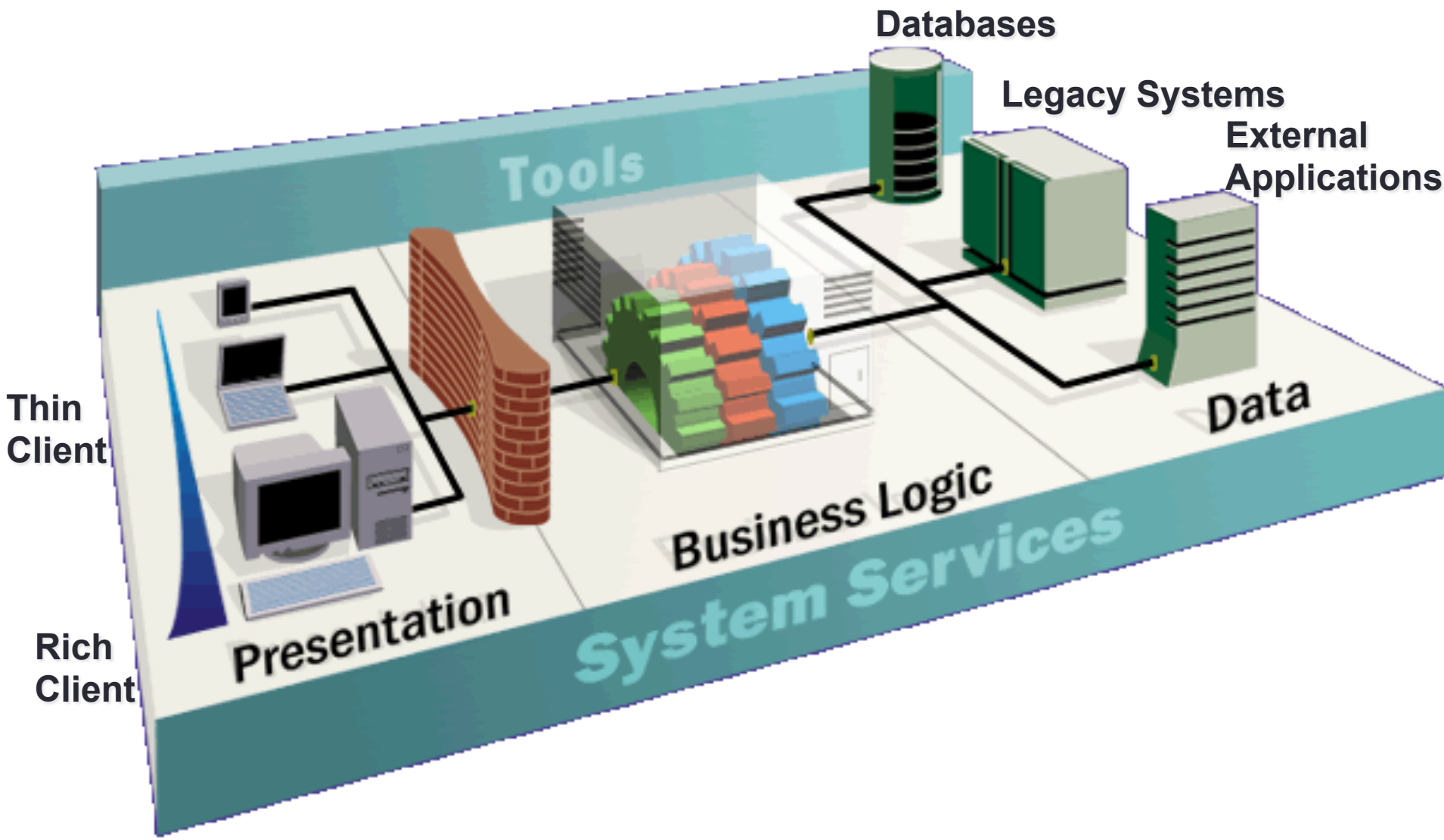
- Divisione del lavoro: thin client
  - Il client implementa solo l'interfaccia utente grafica
  - Il server implementa la logica dell'applicazione e la gestione dei dati
- Divisione del lavoro: thick client
  - Il client implementa sia l'interfaccia grafica che la logica dell'applicazione
  - Il server implementa la gestione dei dati

Architetture client-server (segue)

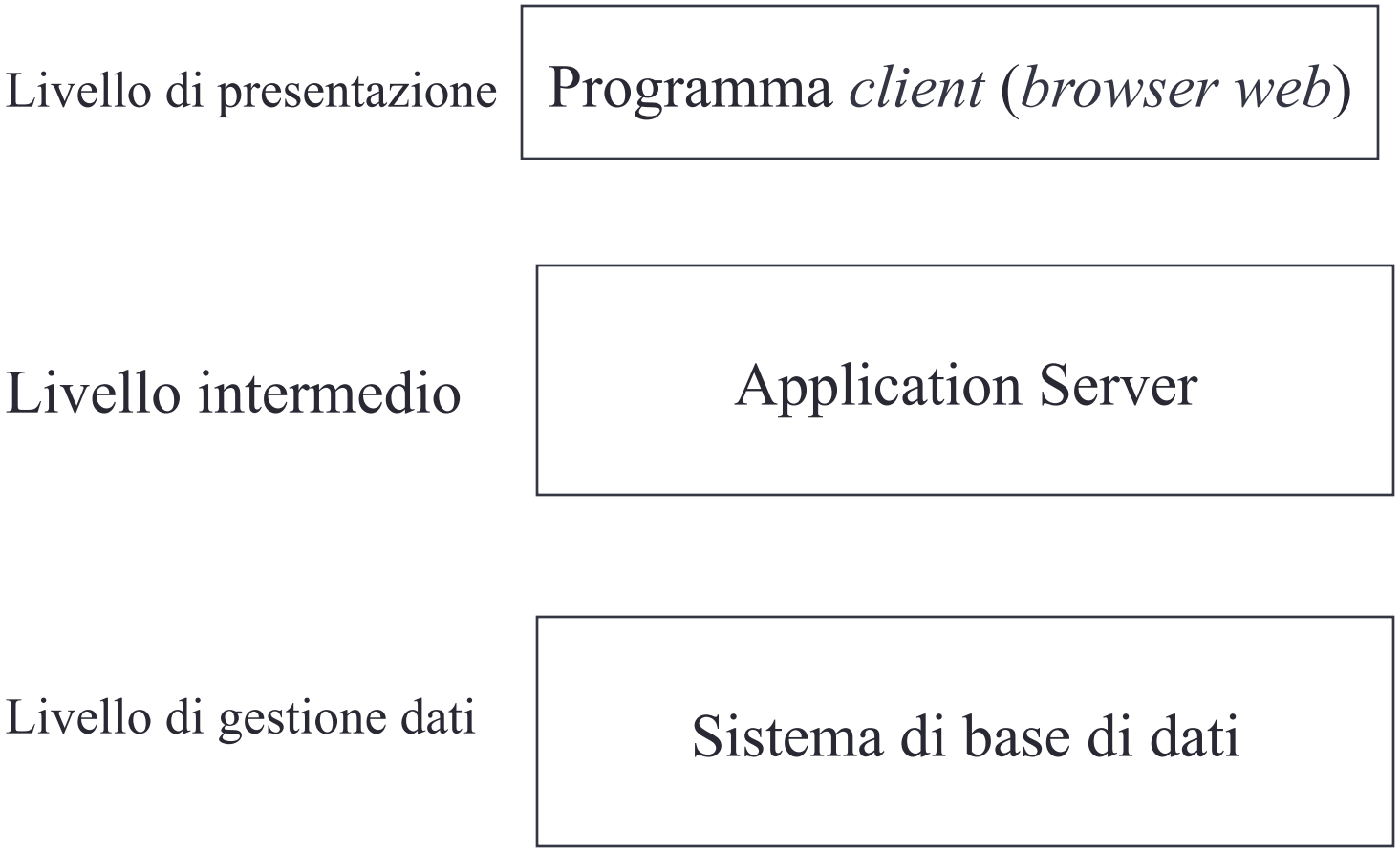
Svantaggi dei thick client

- Nessun luogo centralizzato per aggiornare la logica dell'applicazione
- Problemi di sicurezza: il server deve fidarsi dei client
  - Il controllo di accesso e l'autenticazione devono essere gestiti dal server
  - I client devono lasciare la base di dati del server in uno stato consistente
  - Una possibilità: incapsulare tutti gli accessi alla base di dati in stored procedure
- Non scalabile a più di un centinaio di client
  - Grossi trasferimenti di dati tra server e client
  - Più di un server crea un problema: x client, y server: x*y connessioni

Architetture 3-tier



L'architettura a tre livelli



I tre livelli

Livello di presentazione

- Interfaccia primaria con l'utente
- Deve adattarsi a diversi dispositivi di visualizzazione (PC, PDA, telefoni cellulari, accesso vocale?)

Livello intermedio

- Implementa la logica dell'applicazione (implementa azioni complesse, mantiene lo stato tra diversi passi di un flusso di lavoro)
- Accede a diversi sistemi di gestione dei dati

Livello di gestione dei dati

- Uno o più sistemi standard per la gestione di basi di dati

Esempio 1: prenotazioni aeree

- Costruire un sistema per prenotazioni aeree
- Cosa viene fatto dai vari livelli?
- Sistema di basi di dati
  - Informazioni sulle aerolinee, posti disponibili, informazioni sui clienti, etc.
- Application server
  - Logica per fare le prenotazioni, cancellare le prenotazioni, aggiungere nuove aerolinee, etc.
- Programma client
  - Log in dei vari utenti, visualizzazione di moduli e output in forma leggibile

Esempio 2: iscrizione a corsi

- Costruire un sistema usando il quale degli studenti possono iscriversi a dei corsi
- Sistema di base di dati
  - Informazioni sugli studenti, informazioni sui corsi, informazioni sui docenti, disponibilità dei corsi, pre-requisiti, etc.
- Application server
  - Logica per modificare un corso, cancellare un corso, creare un nuovo corso, etc.
- Programma client
  - Login dei vari utenti (studenti, personale, professori), visualizzazione di moduli e output in forma leggibile

Tecnologie

Programma client *(Browser web)*	*HTML* *Javascript* *XSLT*
Application Server *(Tomcat, Apache)*	*JSP, Servlets,* *PHP* *CGI, Cookies*
DBMS *(MySQL, Oracle, DB2)*	*SQL,* *Stored Procedures,* *XML*

Vantaggi dell'architettura a tre livelli

- Sistemi eterogenei
- Thin client
- Accesso integrato ai dati
- Scalabilità
- Sviluppo software

Argomenti del capitolo

- Internet: Concetti di base
- Formati di dati per il Web
  - HTML, XML, DTD
- Architetture a tre livelli e Web
- Il livello di presentazione
  - Moduli HTML: GET e POST HTTP, codifica di URL; Javascript; fogli di stile. XSLT
- Il livello intermedio
  - CGI, application server, servlets, JavaServerPages, passaggio di argomenti, mantenimento dello stato (cookie)

Introduzione al livello di presentazione

- Richiamo: funzionalità del livello di presentazione
  - Interfaccia primaria per l'utente
  - Deve adattarsi ai diversi dispositivi di visualizzazione
  - Funzionalità semplice, come il controllo della validità dei campi
- Tecnologie:
  - moduli HTML: come passare dati al livello intermedio
  - JavaScript: funzionalità semplice al livello di presentazione
  - Fogli di stile: separare i dati dalla visualizzazione

Moduli HTML

- Modo diffuso per comunicare dati **dal client al livello intermedio**
- Formato generale di un modulo :

```
<FORM ACTION=”pagina.jps” METHOD=”GET”
NAME=”ModuloLogin”>
…
</FORM>
```

- Componenti di un tag FORM HTML:
  - ACTION: specifica l’URI che gestisce il contenuto
  - METHOD: specifica il metodo HTML GET o POST
  - NAME: nome del modulo; può essere usato in script sul lato *client* per far riferimento al modulo

Dentro i moduli HTML

- Tag INPUT:
  - Attributi:
    - TYPE: text (campo per l'inserimento di testo), password (campo per l'inserimento di testo dove il testo immesso è visualizzato in maniera protetta, reset (ripristina tutti i campi del modulo)
    - NAME: nome simbolico, usato per identificare il valore del campo al livello intermedio
    - VALUE: valore predefinito
  - Esempio: <INPUT TYPE="text" Name="titolo">

- Modulo di esempio:

```
<form method="POST"   action="Sommario.jsp">
   <input type="text"          name="userid">
   <input type="password" name="password">
   <input type="submit"    value="Login"    name="Invia">
   <input type="reset"       value="Reimposta">
</form>
```

Passaggio di argomenti

Due metodi: GET e POST

- GET
  - I contenuti del modulo vanno nell'URI specificato
  - Struttura:
    - azione?nome1=valore1&nome2=valore2&nome3=valore3
      - azione: nome dell'URI specificato nel modulo
      - le coppie (nome, valore) provengono dai campi INPUT del modulo; campi vuoti hanno valori vuoti ("nome=”))
- esempio dal precedente modulo per l'immissione di una password:
  - Sommario.jps?userid=john&password=johnpw
- Notate che la pagina chiamata azione deve essere un programma, uno script o una pagina che dovrà elaborare i dati inseriti dall'utente

Codifica dei campi del modulo

- I campi del modulo possono contenere caratteri ASCII generici che possono non apparire in un URI
- Una speciale convenzione di codifica converte tali valori in caratteri “compatibili con gli URI”:
  - Converte tutti i caratteri “speciali” in %xyz, dove xyz è il codice ASCII del carattere. I caratteri speciali includono &, =, +, %, etc.
  - Converte tutti gli spazi nel carattere “+”
  - Incolla le coppie (nome, valore) dai tag INPUT del modulo tramite “&” per formare l’URI

Moduli HTML: un esempio completo

```
<form method="POST" action="Sommario.jsp ">
  <table align = "center" border="0" width="300">
  <tr>
        <td>Userid</td>
        <td><input type="text" name="userid" size="20"></td>
  </tr>
  <tr>
        <td>Password</td>
        <td><input type="password" name="password" size="20"></td>
  </tr>
  <tr>
        <td align = "center"><input type="submit" value="Login“
              name="submit"></td>
  </tr>
  </table>
</form>
```

JavaScript

- Scopo: aggiungere funzionalità al livello di presentazione
- Applicazioni di esempio:
  - rilevare il tipo di browser e caricare una pagina specifica per quel browser
  - validazione di moduli: validare i campi di immissione testo del modulo
  - controllo del browser: aprire nuove finestre, chiudere finestre esistenti (esempio: finestre pop-up di pubblicità)
- Di solito incapsulato direttamente nell'HTML tramite il tag <SCRIPT>...</SCRIPT>
- <SCRIPT> ha diversi attributi:
  - LANGUAGE: specifica il linguaggio dello script (ad esempio javascript)
  - SRC: file esterno con il codice di script
  - Esempio:
- <SCRIPT LANGUAGE="JavaScript" SRC="validazione.js">
  </SCRIPT>

JavaScript (segue)

- Se il tag <SCRIPT> non ha un attributo SRC, allora il JavaScript è direttamente nel file HTML
- Esempio:
  <SCRIPT LANGUAGE=”JavaScript”>
  <!—alert(“Benvenuto nella nostra libreria”)
  //-->
  </SCRIPT>
- Due diversi stili di commento:
  - <!—commento per HTML, poiché il codice JavaScript che segue dovrebbe essere ignorato dall’elaboratore HTML
  - // commento per JavaScript allo scopo di chiudere il commento HTML

JavaScript (segue)

- JavaScript è un linguaggio di scripting completo
  - Variabili
  - Assegnazioni (=. +=, ...)
  - Operatori di confronto (<, >, ...) operatori booleani (&&, ||, !)
  - Comandi
    - If (condizione) {comandi;} else {comandi;}
    - Cicli for, cicli do-while e cicli while
  - Funzioni con restituzione di valori
    - Si creano funzioni usando la parola chiave `function`
    - `Function F(arg1, ..., argk) {comandi;}`

JavaScript: un esempio completo

Modulo HTML :

```
<form method=”POST"
  action="tmp.html"
  id="LoginForm">
  <input type="text"
  name="userid">
  <input type="password"
  name="password">
  <input type="submit"
  value="Login"    name="Invia"
  onClick="controllaLoginVuoto()">
  <input type="reset"
  value="Reimposta">
</form>
```

JavaScript associato :

```
<script>
function controllaLoginVuoto() {
  loginForm =
   document.getElementById("LoginForm")
  if ((loginForm.name.value == "") ||
     (loginForm.password.value == ""))
  {
    alert("Immettere un valore per userid e
   password") ;
    return false;
  }
  else return true;
}
</script>
```

Fogli di stile

- Idea: separare la visualizzazione dal contenuto e adattarla a differenti formati di presentazione
- Due aspetti:
  - le trasformazioni del documento decidono quali parti del documento visualizzare, e in quale ordine
  - “Spezzettamento” del documento per decidere come ciascuna parte deve essere visualizzata
- Perché usare i fogli di stile?
  - Riutilizzo dello stesso documento per visualizzazioni differenti
  - Adattamento della visualizzazione alle preferenze dell’utente
  - Riutilizzo dello stesso documento in contesti diversi
- Due linguaggi per i fogli di stile
  - Fogli di stile ad albero (CSS): per documenti HTML
- Extensible stylesheet language (XSL): per documenti XML

CSS: fogli di stile ad albero

- Definiscono come devono essere visualizzati i documenti HTML
- Molti documenti HTML possono far riferimento allo stesso CSS
  - Si può cambiare il formato di un sito web cambiando un singolo foglio di stile
  - Esempio

    <LINK rel=”foglio di stile” TYPE=”text/css” HREF=”libri.css”/>

- Ogni riga consiste di tre parti:

  Selettore {proprietà: valore}

- Selettore: tag di formato definito
- Proprietà: attributo del tag il cui valore viene impostato
- Valore: valore dell’attributo

CSS: fogli di stile ad albero (segue)

Esempio di foglio di stile:

```
body {background-color: yellow}
h1 {font-size: 36pt}
h3 {color: blue}
p {margin-left: 50px; colo: red}
```

La prima riga ha lo stesso effetto di

```
<body background-color=”yellow”>
```

XSL

- Linguaggio per esprimere i fogli di stile
  - Maggiori informazioni su http://www.w3.org/Style/XSL
- Tre componenti
  - XSLT: linguaggio di trasformazione XSL (*XSL Transformation Language*)
    - Può trasformare un documento in un altro
    - Maggiori informazioni su http://www.w3.org/TR/xslt
  - XPath: Linguaggio per Percorsi XML (*XML Path Language*)
    - Seleziona parti di un documento XML
    - Maggiori informazioni su http://www.w3.org/TR/xpath
  - Oggetti di formattazione XSL (*XSL Formatting Objects*)
    - Formatta il risultato di una trasformazione XSL
    - Maggiori informazioni su http://www.w3.org/TR/xsl/

Argomenti del capitolo

- Internet: Concetti di base
- Formati di dati per il Web
  - HTML, XML, DTD
- Architetture a tre livelli e Web
- Il livello di presentazione
  - Moduli HTML: GET e POST HTTP, codifica di URL; Javascript; fogli di stile. XSLT
- Il livello intermedio
  - CGI, application server, servlets, JavaServerPages, passaggio di argomenti, mantenimento dello stato (cookie)

Introduzione al livello intermedio

- Richiamo: funzionalità del livello intermedio
  - Codifica la logica dell'applicazione
  - Effettua le connessioni al/ai sistema/i di basi di dati
  - Riceve il testo immesso nei moduli al livello di presentazione
  - Genera i risultati per il livello di presentazione
- Tecnologie
  - **CGI**: protocollo per il passaggio di argomenti ai programmi in esecuzione al livello intermedio
  - **Application server**: ambienti di esecuzione al livello intermedio
  - **Servlet**: programmi Java al livello intermedio
  - **JavaServerPages**: script Java al livello intermedio
  - Mantenimento dello stato: come mantenere lo stato al livello intermedio. Argomento principale: **cookie**

CGI: Common Gateway Interface

- Scopo: trasmettere argomenti dai moduli HTML ai programmi applicativi che vengono eseguiti al livello intermedio
- I dettagli del reale protocollo CGI non sono importanti -> le librerie implementano le interfacce ad alto livello

- Svantaggi:
  - Il programma dell'applicazione viene eseguito come un nuovo processo ad ogni invocazione (rimedio: FastCGI)
  - Non c'è condivisione di risorse tra i programmi applicativi (ad esempio connessioni a basi di dati)
  - Rimedio: application server

CGI: esempio

- Modulo HTML:

```
<FORM ACTION="trovaLibri.cgi" METHOD="POST">
Inserisci il nome di un autore:
   <INPUT TYPE="text" NAME="nomeAutore">
   <INPUT TYPE="submit" VALUE="Invia">
   <INPUT TYPE="reset" VALUE="Cancella">
</FORM>
```

- Codice Perl:

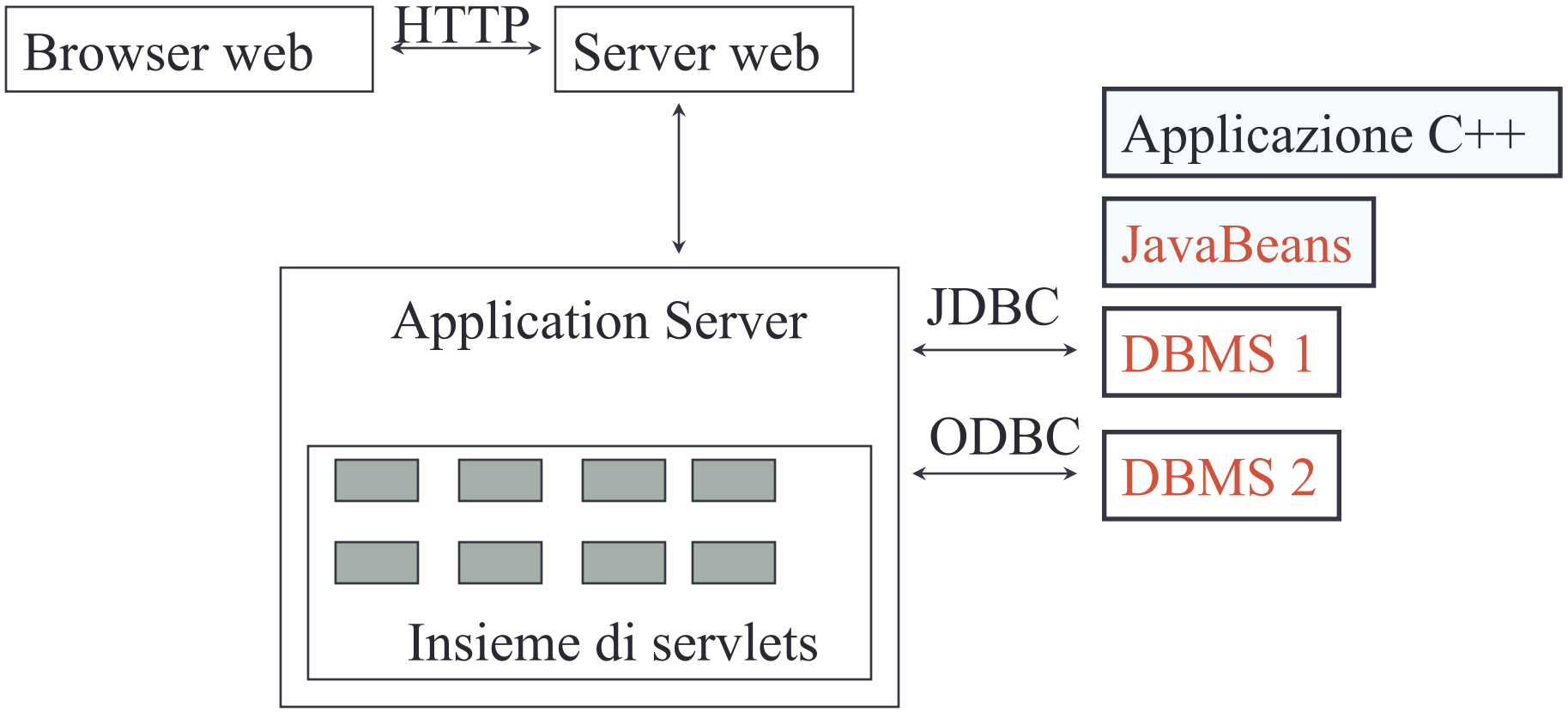
```
use CGI;
$dataIn=new CGI;
$dataIn->header();
$authorName=$dataIn->param('nomeAutore');
print("<HTML><TITLE>Prova di passaggio di argomenti</TITLE>");
print("Il nome dell'autore è " + $authorName);
print("</HTML>");
exit;
```

Application Server

- Idea: evitare il sovraccarico delle CGI
  - Insieme principale dei thread dei processi
  - Gestisce le connessioni
  - Consente l'accesso a sorgenti di dati eterogenee
  - Altre funzionalità quali API per la gestione delle sessioni

Application server: struttura dei processi

Struttura dei processi



Servlet

- Java Servlets: codice Java che viene eseguito al livello intermedio
  - Indipendente dalla piattaforma
  - API Java completamente disponibile, incluso JDBC

Esempio :

```
import java.io.*;
import java.servlet.*;
import java.servlet.http.*;
public class ScheletroDiServlet extends HttpServlet {
   public void doGet(HTTPServletRequest richiesta,
                                                HTTPServletResponse risposta)
   throws ServletException, IOException {

            PrintWriter out=risposta.getWriter();
            out.println("Ciao mondo");
   }
}
```

Servlet (segue)

- Vita di un servlet?
  - Il server web inoltra la richiesta al contenitore del servlet
  - Il contenitore crea una istanza del servlet (chiama il metodo init(); al momento della deallocazione: chiama il metodo destroy())
  - Il contenitore chiama il metodo service()
    - Service() chiama doGet() per il GET HTTP o il doPost() per il POST HTTP
    - Di solito service() non viene sovrascritto, ma vengono sovrascritti doGet() e doPost()

Servlet: un esempio completo

```
public class LeggiNomeUtente extends HttpServlet {
 public void doGet( HttpServletRequest richiesta,
                          HttpServletResponse risposta)
                           throws ServletException, IOException {
      risposta.setContentType("text/html");
     PrintWriter out=risposta.getWriter();
      out.println("<HTML><BODY>\n <UL> \n" +
        "<LI>" + richiesta.getParameter("userid") + "\n" +
         "<LI>" + richiesta.getParameter("password") + "\n" +
          "<UL>\n<BODY></HTML>");
 }
 public void doPost( HttpServletRequest richiesta,
                          HttpServletResponse risposta)
                           throws ServletException, IOException {
             doGet(richiesta, risposta);
 }
}
```

Java Server Pages

- Servlet
  - Generano HTML scrivendolo sull'oggetto "**`PrintWriter`**"
  - Prima il codice, poi la pagina web
- JavaServerPages
  - Codice scritto in HTML, simile al codice dei servlet, incapsulato nell'HTML
  - Prima la pagina web, poi il codice
  - Di solito sono compilate in un servlet

JavaServerPages: esempio

```
<html>
<head><title>Benvenuto alla B&N</title></head>
<body>
 <h1>Bentornato!</h1>
 <% String name="NuovoUtente";
      if (request.getParameter("UserName")!= null) {
            nome=request.getParameter("UserName");
      }
 %>
 Sei connesso come <%=nome%>
 <p>
</body>
</html>
```

Mantenimento dello stato

- L'HTTP è senza memoria
- Vantaggi
  - Facile da usare: non c'è bisogno di nulla
  - Ottimo per applicazioni con informazioni statiche
  - Non richiede spazio extra in memoria
- Svantaggi
  - Niente registrazione delle richieste precedenti significa
    - Niente carrelli per la spesa
    - Niente login degli utenti
    - Nessun contenuto personalizzato o dinamico
- Maggiore difficoltà di implementazione della sicurezza

Stato delle applicazioni

- Stato sul lato server
  - L'informazione è memorizzata in una base di dati, o nella memoria locale dello strato applicativo
- Stato sul lato client
  - L'informazione è memorizzata sul computer client sotto forma di cookie
- Stato nascosto
  - L'informazione è nascosta in pagine web create dinamicamente

Stato delle applicazioni

Stato sul lato server

- Molti tipi di stato sul lato server:
- 1. Mantenimento delle informazioni in una base di dati
  - I dati sono al sicuro nella base di dati
  - MA: richiede un accesso alla base di dati per interrogare o aggiornare le informazioni
- 2. Uso della memoria locale del livello dell'applicazione
  - Possibile mappare l'indirizzo IP dell'utente in qualche stato
  - MA: questa informazione è volatile e impiega parecchia della memoria principale del server

5 milioni di IP = 20 MB

Stato sul lato server (segue)

- Si dovrebbe usare il mantenimento dello stato sul lato server per le informazioni persistenti
  - Vecchi ordini del cliente
  - Memorizzazione dei movimenti di un utente in un sito
  - Scelte permanenti fatte dall'utente

Stato sul lato client: cookie

- Memorizzare sul client del testo che verrà passato all'applicazione con ogni richiesta HTTP.
  - Possono essere disabilitati dal client
  - Sono erroneamente percepiti come "pericolosi", e quindi potenziali visitatori del sito saranno spaventati dalla richiesta di abilitare i cookie!
- Sono una collezione di coppie (Nome, Valore)

[1]http://www.webdevelopersjournal.com/columns/stateful.html

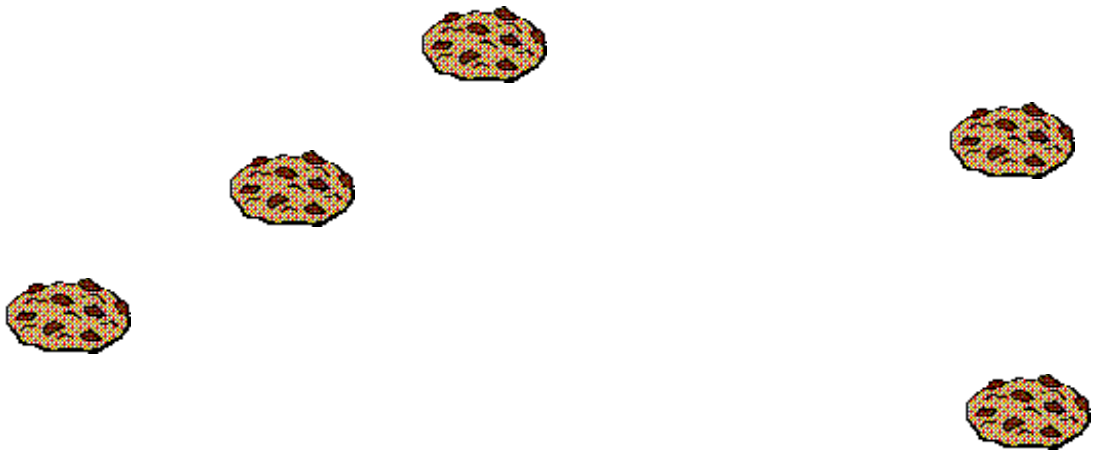
Stato sul lato client: cookie (segue)

- Vantaggi
  - Facilità d'uso in servlet Java / JSP
  - Forniscono un modo semplice per mantenere dati non essenziali sul client anche quando il browser è chiuso
- Svantaggi
  - Limite di 4 KB di informazione
  - Gli utenti possono (e spesso lo fanno) disabilitarli
- Si dovrebbero usare i cookie per memorizzare lo stato interattivo
- Informazioni sul login dell'utente corrente
- Carrello della spesa corrente
- Qualunque scelta non permanente fatta dall'utente

Creare un cookie

```
Cookie mioCookie =
  new Cookie(“nomeutente”, “jeffd”);
response.addCookie(mioCookie);
```

- Si può creare un cookie in qualunque momento



Accedere ad un *cookie*

```
Cookie[] cookies = request.getCookies();
String Utente;
for(int i=0; i<cookies.length; i++) {
  Cookie cookie = cookies[i];
  if(cookie.getName().equals("username"))
      Utente = cookie.getValue();
}
// a questo punto Utente == "username"
```

- Si deve accedere ai cookie PRIMA di impostare l'intestazione della risposta:

```
response.setContentType("text/html");
PrintWriter out = response.getWriter();
```

Caratteristiche dei cookie

- I *cookie* possono avere
  - Una durata (scadono immediatamente oppure persistono anche dopo che il browser è stato chiuso)
  - Filtri per stabilire a quali domini/cartelle il cookie viene spedito
- Maggiori informazioni nei manuali Java Servlet API e Servlet

Sommario

- Concetti Internet
  - Reti, Protocolli, TCP/IP, HTTP, WWW, URIs
- Formati di dati per Web
  - HTML, XML, DTD
- Architetture a tre livelli e Web
- Strato di presentazione
  - Moduli HTML: GET e POST HTTP, codifica di URL; Javascript; fogli di stile. XSLT
- Strato intermedio
  - application server, CGI, Servlets, JavaServerPages, passaggio di argomenti, mantenimento dello stato (*cookies*)