PAS 2013/14
SISTEMI E RETI DI CALCOLATORI PER L'INSEGNAMENTO
Lezione n.2
IL LIVELLO DELLE APPLICAZIONI

28/05/2014

Laura Ricci

INTERNET: LE APPLICAZIONI

- Applicazioni tradizionali:
  - Email
  - News
  - Remote Login (SSH)
  - File Transfer
- L'applicazione 'killer':
  - World-Wide Web (WWW)
- Nuove applicazioni:
  - Videoconferenza
  - Telefonia (VOIP)
  - P2P applications

APPLICAZIONI DI RETE

Applicazione	Protocollo a livello applicazioni	Protocollo a livello trasporto
Posta Elettronica	SMTP	TCP
Login Remota	SSH	TCP
Web	HTTP	TCP
Trasferimento File	FTP	TCP
File Server Remoto	NFS	UDP o TCP
Multimedia	Proprietario	UDP o TCP
Telefonia	Proprietario	UDP
DNS (Domain Name Server)	….	…..

IL WORLD WIDE WEB

- World Wide Web(WWW): applicazione sviluppata per l'accesso ad informazioni memorizzate sugli host di una WAN

- WEB: applicazione client/server

- WEB Client = Web Browser programma (Chrome, Explorer, Mozilla, Opera,…) dotato di una interfaccia grafica che consente di reperire e visualizzare informazioni memorizzate su un server web

- informazioni reperite(pagine web) sono documenti HTML

- documenti HTML = file di testo che
  - contengono alcuni tag che descrivono come il testo deve essere visualizzato all'interno di un web browser
  - possono contenere degli iperlink = riferimenti ad altri documenti

IL WORLD WIDE WEB

- sul web, ogni documento è localizzato mediante una URL (Uniform Resource Locator)

- una URL in genere indica un file HTML, ma può indicare anche qualsiasi altro documento accedibile mediante un web browser

- esempio: URL **`http://www.cs.princeton.edu/index.html`** specifica che un documento HTML di nome **`index.htlm`** può essere acceduto mediante HTTP su un host di nome **`www.cs.princeton.edu`**

- una URL è composta delle seguenti parti:
  - nome del protocollo utilizzato per accedere l'oggetto (http)
  - nome dell'host (**`www.cs.princeton.edu`**)
  - pathname che individua l'oggetto all'interno dell'host

URL (UNIFORM RESOURCE LOCATOR)

Formato generale di una URL

protocol://hostname :port/path/filename?query#fragment

protocol : file,ftp,http,https,news,telnet,wais,...

hostname: nome simbolico o indirizzo IP del server

port: numero della porta, può essere omesso se il server è in esecuzione sulla porta di default (es:la porta 80 per i server HTTP)

path: punta alla directory che contiene la risorsa. Il path è relativo alla document root del server

filename: indica un file specifico nella directory specificata.

query: è una stringa utilizzata per fornire argmenti di forms in esecuzione sul server

fragment: si riferisce ad una parte particolare della risorsa (es: un anchor in un documento HTML

URL (UNIFORM RESOURCE LOCATOR)

- Il document root riferito nel campo path non punta necessariamente alla radice del filesystem del server (infatti il server non necessariamente pubblica il suo intero file system ai clients)
- Esempio: su un server Unix, tutti i file che possono essere acceduti dai clients possono trovarsi nella directory /var/public/html/. Questa directory può apparire ai clients come la directory root del server
- Molto spesso viene omesso il filename (ed i campi successivi). In questo caso il server può:
  - inviare un file di default (definito al momento della configurazione del server). Molto spesso il file di default è `index.html`
  - inviare una lista dei files e delle sottodirectory presenti nella document root
  - inviare un messaggio di errore `403 Forbidden error message`

RELATIVE URL

- Quando un client accede ad un documento D individuato da URL_D e pubblicato da un server, molto spesso nel documento sono contenute altre URL che puntano ad altri documenti nello stesso server.

- Le URL contenute in un documento spesso condividono gran parte dell'informazione contenuta in URL_D (protocollo, nome dell'host, path,..).

- relative URL: URL che ereditano parte dell'informazione contenuta nella URL del documento in cui sono inserite

- Uso di relative URL, vantaggi
  - risparmio di scrittura
  - possibilità di utilizzare più protocolli per l'accesso allo stesso documento (esempio: FTP, HTTP,...)

LABORATORIO:ACCEDERE AD UNA URL

- Scrivere un programma JAVA che consenta di accedere ad un file HTML specificando la URL che lo individua

- Scaricare il file HTML, salvarlo in un file e rileggere il file mediante un browser

- Confrontare il file aperto dal browser con la pagina acceduta in remoto

LABORATORIO:ACCEDERE AD UNA URL

```
import java.io.*; import java.net.*;
public class URLuse {
    public static void main (String[] args) {
      URL u; InputStream is = null; DataInputStream dis;
      String s;
        try { u = new URL("http://www.di.unipi.it");
            is = u.openStream();
            dis = new DataInputStream(new BufferedInputStream(is));
            while ((s = dis.readLine()) != null)
              {System.out.println(s); }
           }
      catch (MalformedURLException mue) {
               System.out.println("Ouch - a MalformedURLException happened.");
               System.exit(1); }
      catch (IOException ioe) {
               System.out.println("Oops- an IOException happened.");
                ioe.printStackTrace(); System.exit(1);}}}
```

WORLD WIDE WEB: CARATTERISTICHE

Accesso ad un oggetto individuato da una URL, ad esempio

`http://www.cs.princeton.edu/index.html`

- apertura di una connessione TCP dal browser web al server web di nome `http://www.cs.princeton.edu` (porta 80 di default)
- invio di un messsaggio di richiesta sulla connessione aperta
- il server web ricerca l'oggetto individuato dal path `index.html` ed invia il contenuto del file al browser
- Il server chiude la connessione con il client (connessione non persistente)

WORLD WIDE WEB: CARATTERISTICHE

- Hypertext Transfer Protocol (HTTP): protocollo di livello applicazione di tipo richiesta/risposta utilizzato per l'invio di contenuto web

- Ogni pagina web visitata, ad esempio una pagina Fecaebook, è scaricata utilizzando HHTP

- HTTP è stato esteso rispetto all'originale progettato solo per il trasferimento di pagine web ed è attualmente utilizzato come base per altri protocolli, esempio il protocollo SIP, utilizzato per VOIP

WORLD WIDE WEB: PROTOCOLLO HTTP

Protocollo di Rete: definisce le regole e le convenzioni per lo scambio di messaggi tra due entità di una rete

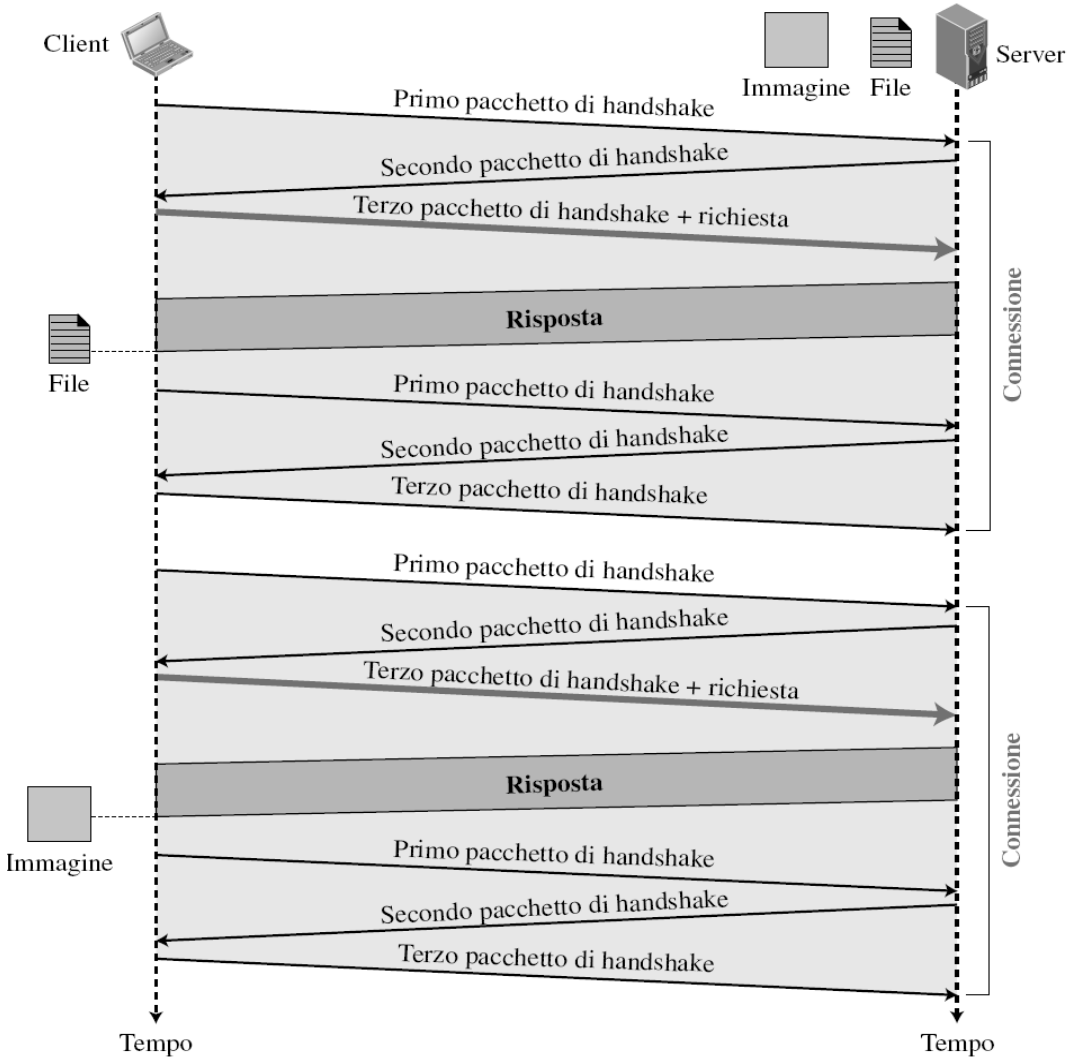
Protocollo HTTP: insieme di regole che definiscono formati e ordine dei messaggi scambiati tra un browser ed un server web.

- protocollo a livello applicazione
- utilizza il livello di trasporto TCP
- protocollo privo di stato:
  - non memorizza informazioni relative alle richieste ricevute dagli utenti (eccezione: vedere meccanismo cookies)
  - ogni richiesta da parte di un utente viene trattata in modo indipendente, non si mantiene memoria delle richieste fatte in precedenza dal solito utente
  - non esiste una nozione di sessione, come nel protocollo Telnet o FTP
  - ma....diversi meccanismi introdotti successivamente per introdurre una qualche forma di stato

WORLD WIDE WEB: PROTOCOLLO HTTP

- HTTP 1.0 connessioni non persistenti: viene aperta una nuova connessione per ogni richiesta effettuata dal client al server. La connessione si chiude con l'invio della risposta da parte del server
- Esempio:
  - Un client richiede una pagina web ad un web server. La pagina contiene un file di base HTML e 10 immagini JPEG. Tutti gli 11 oggetti risiedono sullo stesso server
  - Il server invia il file HTML di base al client e subito dopo chiude la connessione TCP
  - Il client(browser) riceve la pagina HTML e vede che in essa sono contenuti 10 oggetti JPEG. Ogni oggetto è identificato da una URL
  - Vengono stabilite, in sequenza, 10 connessioni con il server per il reperimento delle 10 immagini
- Nei browser moderni esiste la possibilità di gestire più connessioni TCP contemporaneamente

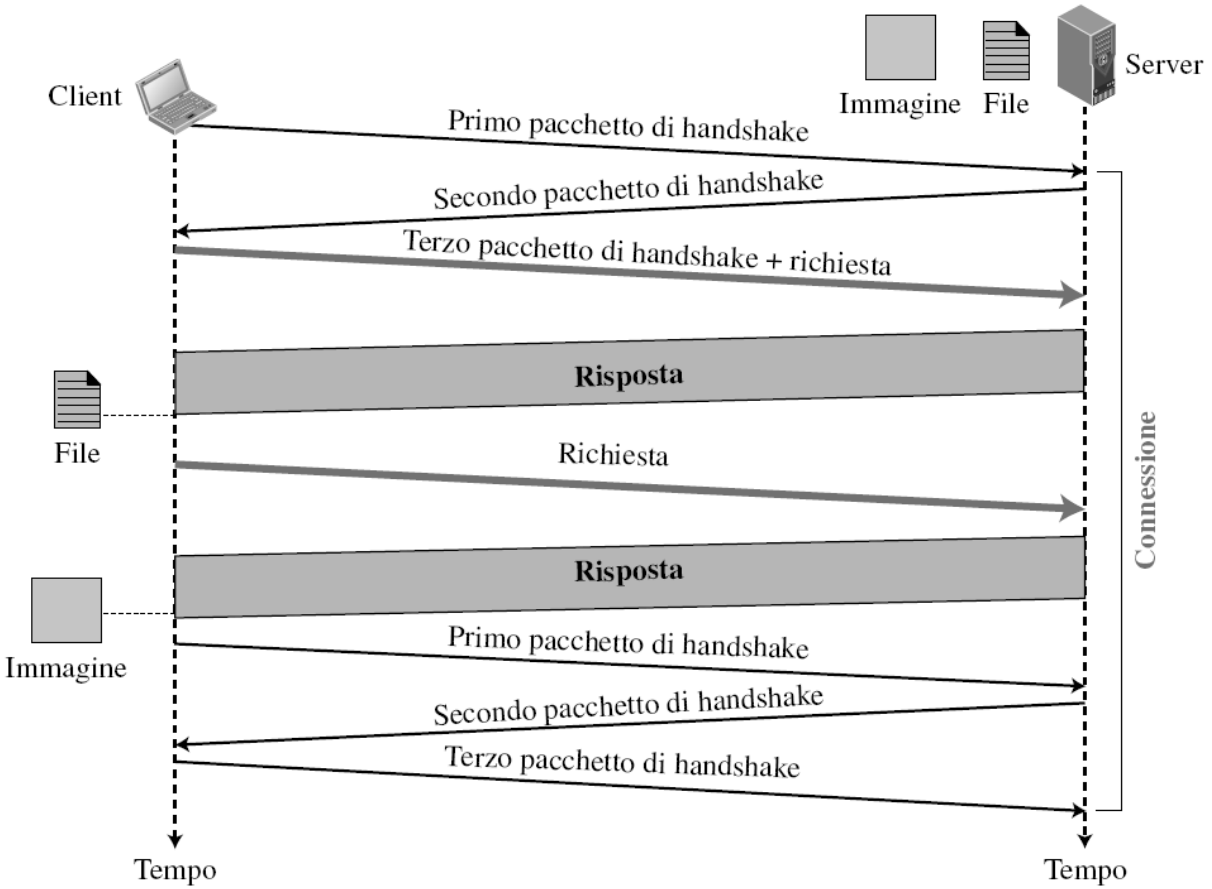
HTTP: CONNESSIONI NON PERSISTENTI



PROTOCOLLO HTTP: CONNESSIONI PERSISTENTI

- svantaggi HTTP 1.0:
  - l'apertura ripetuta di connessioni TCP comporta l'allocazione/deallocazione di buffers per la gestione della connessione
  - 3 way handshake ripetuto per ogni connessione aparta
- HTTP 1.1 utilizza connessioni persistenti:
  - più messaggi di richiesta/risposta sulla stessa connessione TCP
  - la connessione rimane aperta anche dopo che una richiesta del client è stata servita
  - il server può inviare sulla stessa connessione tutte le risorse che compongono una pagina HTML
  - la stessa connessione può essere utilizzata per l'invio di pagine HTML diverse
  - la connessione viene chiusa quando rimane inattiva per un certo intervallo di tempo (uso di timeout)

HTTP: CONNESSIONI PERSISTENTI



HTTP: CONNESSIONI PERSISTENTI

Connessioni persistenti (HTTP 1.1): il server lascia aperta la connessione TCP dopo aver inviato la risposta al client, in attesa di ulteriori richieste

Vantaggi

- minor overhead nella creazione di connessioni
- miglior utilizzazione del meccanismo TCP di controllo di congestione

Problemi

nel server: decidere il momento in cui chiudere una connessione

uso di timeouts = il server chiude una connessione dopo che è trascorso un intervallo di tempo in cui non ha ricevuto richieste dal client

PROTOCOLLO HTTP IN BREVE

Ogni richiesta include tre parti:

- azione (metodo) + risorsa sul server
- una serie di header che contengono metadati
- un messaggio (opzionale)

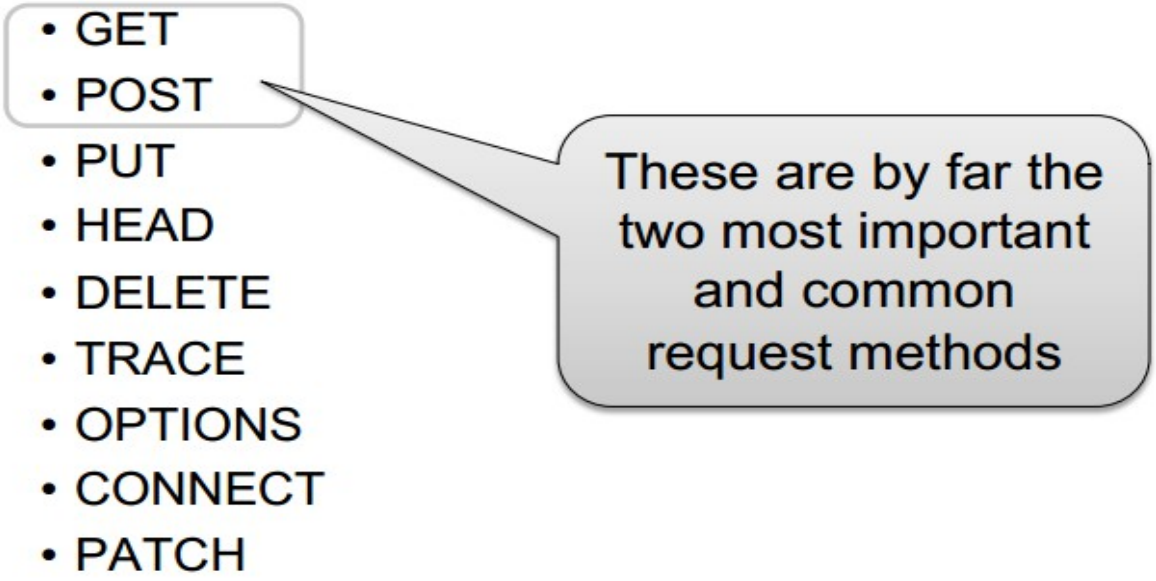
PROTOCOLLO HTTP IN BREVE

In altri termini, si sta indicando al server:

- prego, esegui questa azione (request method)
- su questo dato (risorsa)
- con questi parametri (corpo del messaggio)

PROTOCOLLO HTTP: METODI

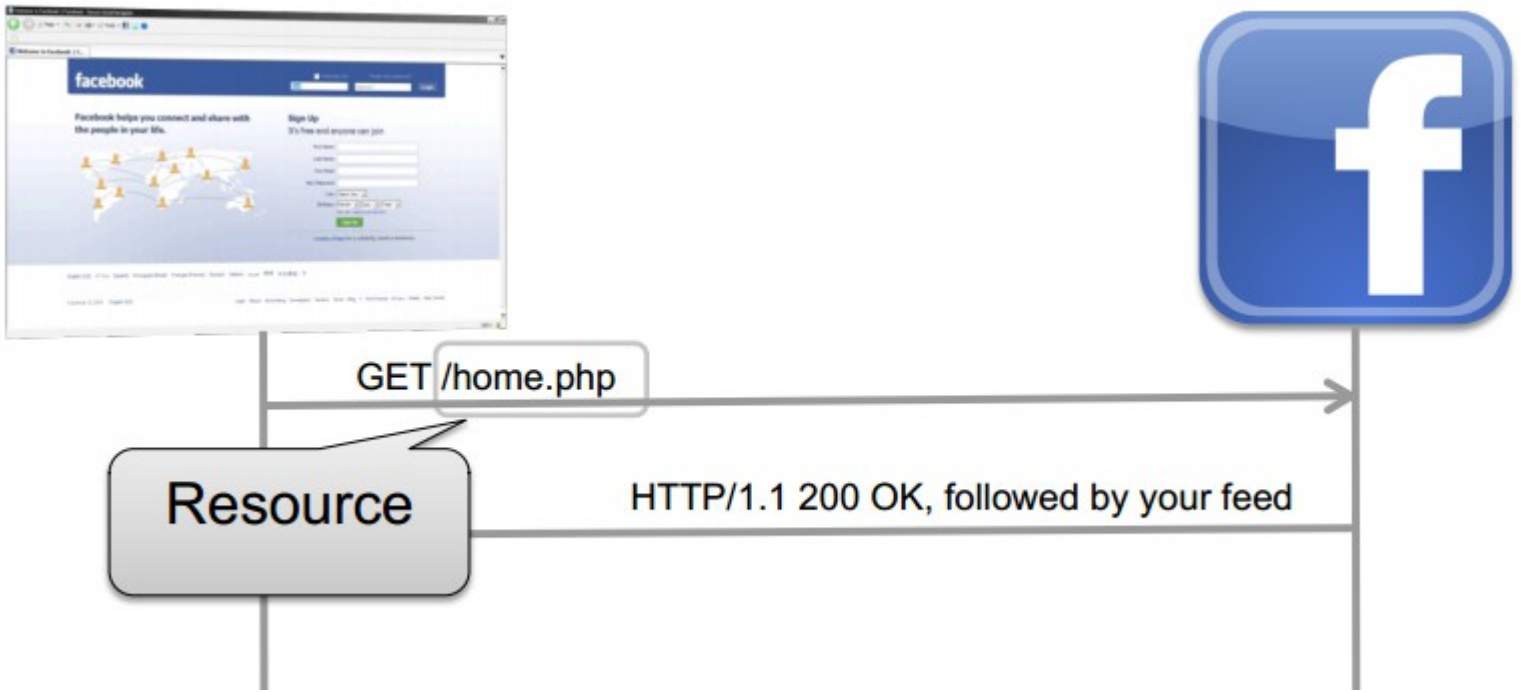
- Metodi utilizzabili in un messaggio di richiesta



GET:per richiedere il contenuto di una risorsa individuata da una URL

POST: utilizzato per inviare i dati necessari per riempire un form (Es: quando si inseriscono i dati da ricercare mediante un motore di ricerca)

PROTOCOLLO HTTP IN BREVE

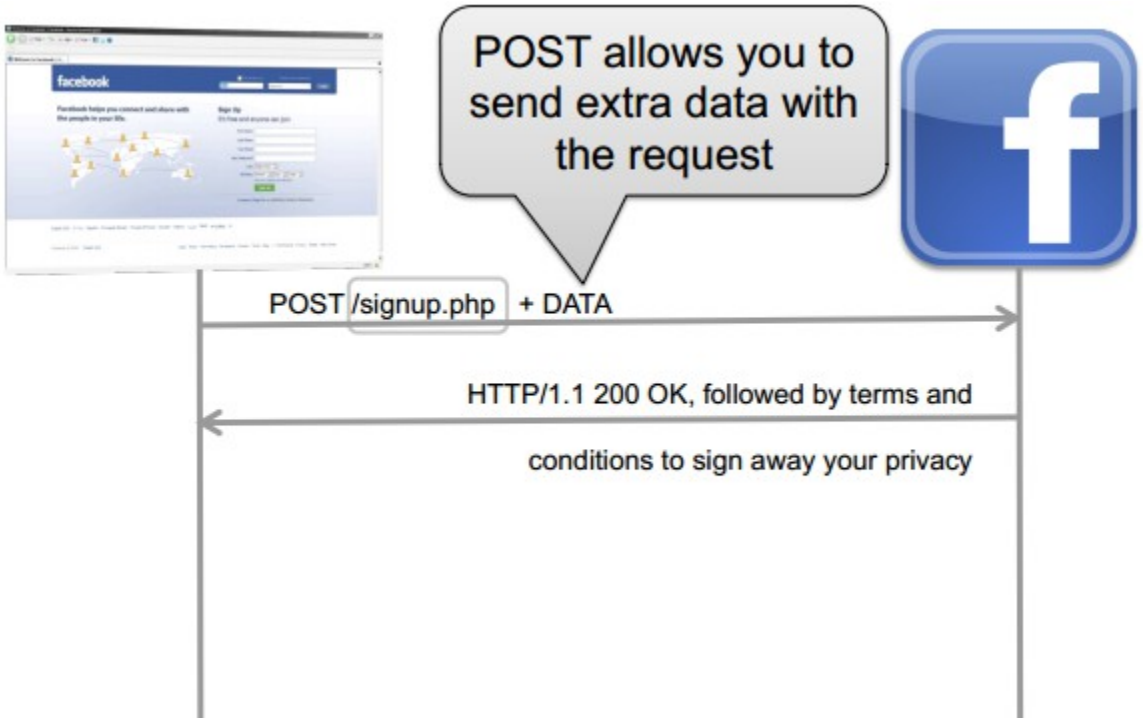


Risorsa: specificata mediante una URL relativa

PROTOCOLLO HTTP IN BREVE

Occorre un metodo che consenta al client di inviare un messaggio al serve

HTTP POST



Metodo POST: una richiesta POST include, oltre alla risorsa da accedere, ulteriori parametri parametri: serie di coppie chiave-valore

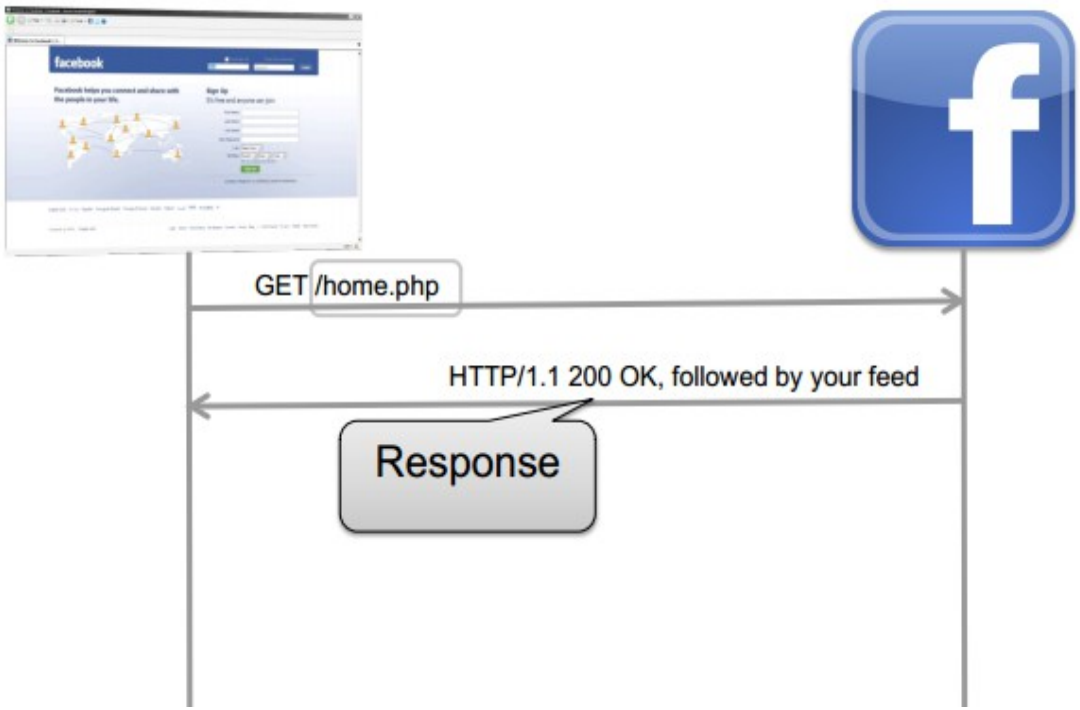
Ad esempio:

- FirstName=John
- LastName=Doe
- SignAwayPrivacy=True

HTTP POST: UN ESEMPIO

```
POST /login.php?login_attempt=1 HTTP/1.1
Accept:application/xml,application/xhtml+xml,text/html;q=0.9,text/plain;q=0.8,image/png,*/*;q=0.5
Content-Type:application/x-www-form-urlencoded
Origin:http://www.facebook.com
Referer:http://www.facebook.com/
User-Agent:Mozilla/5.0 (Macintosh; U; Intel Mac OS X 10_6_3; en-US) AppleWebKit/533.4 (KHTML, like Gecko)
Chrome/5.0.375.53 Safari/533.4
charset_test:€,´,€,´,ж,Д,€
locale:en_US
email:john.doe@gmail.com
pass:somepassword
```

HTTP RESPONSE



Codici di stato comuni:

- 200 OK : è andato tutto bene
- 404 Not Found : la risorsa non è stata trovata
- 500 Server Error : errore in uno script sul server
- 301 Moved Permanently : la risorsa è stata spostata altrove

PROTOCOLLO HTTP: MAGGIORI DETTAGLI

- Esempio di interazione HTTP: supponiamo che un utente abbia inserito la URL **http://www.di.unipi.it/~ricci/papers.html**
- Dopo che la URL è stata inserita, il web browser attiva un client HTTP che stabilisce una connessione sulla porta 80 del server HTTP dell'host www.di.unipi.it
- Formato del messaggio HTTP di richiesta inviato dal client al server

```
GET ~ricci/papers.html HTTP/1.1 <CR LF>
Accept Language:it <CRLF>
User Agent: Mozilla/4.0 <CRLF>
Host:  131.114.3.18 <CRLF>
Connection: keep alive <CRLF>
<CRLF><CRLF>
```

PROTOCOLLO HTTP: MAGGIORI DETTAGLI

- Messaggio codificato in ASCII
- Ogni messaggio è composto da alcune righe separate da un CR (carriage return=invio), LF(Line Feed=nuova linea)
- La prima riga viene detta riga di richiesta, le successive righe di intestazione (headers), segue (eventualmente) il corpo del messaggio
- Riga di richiesta include
  - campo metodo (`GET`, `POST`, `HEAD`)
  - campo URL
  - campo versione
- Ogni riga di intestazione include
  - nome del campo di intestazione (es: connection)
  - valore corrsipondente (es: close/keep-alive)

PROTOCOLLO HTTP: MAGGIORI DETTAGLI

- HTTP headers: specificano metadati inviati dal client al server per "aiutare" il server ad elaborare correttamente la richiesta
- coppie: (Nome, Valore)
- alcuni esempi di header importanti: specificano il tipo di risposte che un client è disposta ad accettare da un server
  - `Accept`: I tipi di contenuti che il client può accettare
  - `Accept-Charset`: il tipo di caratteri che il client può gestire
  - `Accept-Language`: non tutti accettano l'inglese

PROTOCOLLO HTTP: MAGGIORI DETTAGLI

- HTTP headers: specificano metadati inviati dal client al server per "aiutare" il server ad elaborare correttamente la richiesta

- un altro header: User Agent header specifica il tipo del client che ha fatto la richiesta (e.g. Firefox + version, Android + version, etc.)

- Perchè il server è interessato a questa informazione?
  - Il server modifica la presentazione del contenuto a seconda dell'user agent
  - Esempio: presentazione diversa dei contenuti per device mobili

WORLD WIDE WEB: PROTOCOLLO HTTP

Risposta del Server:

```
HTTP/1.1 202 OK <CRLF>Date:…. <CRLF>
    Server: Apache/1.3.0 (Unix) <CRLF>
  Connection: close <CRLF>
  Last Modified: Mon 24 Oct 2007<CRLF>
  Content Length: 6821<CRLF>
  Content Type: text/HTML<CRLF><CRLF>
  <HTML>
     <BODY>
    <H1> LAURA RICCI PAPERS </H1>.............
    </BODY>
     </HTML>
  <CRLF>
```

WORLD WIDE WEB: PROTOCOLLO HTTP

- Struttura generale del messaggio di risposta

  Riga di stato

  Righe di intestazione

  Corpo

```
STATUS  LINE <CRLF>
 MESSAGE HEADER <CRLF><CRLF>
 MESSAGE BODY <CRLF>

  CRLF = Carriage Return + Line Feed
```

WORLD WIDE WEB: CARATTERISTICHE

- l'oggetto O inviato dal server S al client (index.html) può contenere collegamenti ipertestuali, cioè URL che puntano ad altri file, allocati su S stesso o su altri servers

- quando l'utente seleziona un collegamento ipertestuale, il browser apre una nuova connessione TCP per recuperare l'oggetto individuato

- se si usano connessioni non persistenti, viene creata una nuova connessione anche nel caso in cui gli oggetti riferiti siano memorizzati su S stesso

WORLD WIDE WEB

- L'esempio fatto semplifica parecchio le cose
- Nella realtà occorre tenere in considerazione
  - Web caching
  - Pagine dinamiche
  - Esecuzione remota di programmi sul server
  - Applets
  - Sicurezza
  - Cookies

LABORATORIO: IL PROTOCOLLO HTTP

```
import java.net.*; import java.io.*;
public class HttpConnection {
public static void main(String [] args)  throws Exception
{       URL obj;String s;
        obj = new URL("http://www.di.unipi.it");
        HttpURLConnection con = (HttpURLConnection) obj.openConnection();
        con.setRequestMethod("GET");
        int responseCode = con.getResponseCode();
        System.out.println(responseCode);
        InputStream is= con.getInputStream();
        DataInputStream dis = new DataInputStream(is);
        while ((s = dis.readLine()) != null) {
            System.out.println(s);
        } }
```

AUTENTICAZIONE DEI CLIENTS

Meccanismo soft di autenticazione mediante username + password

client: invia la richiesta

server: risponde con un messaggio di richiesta autenticazione

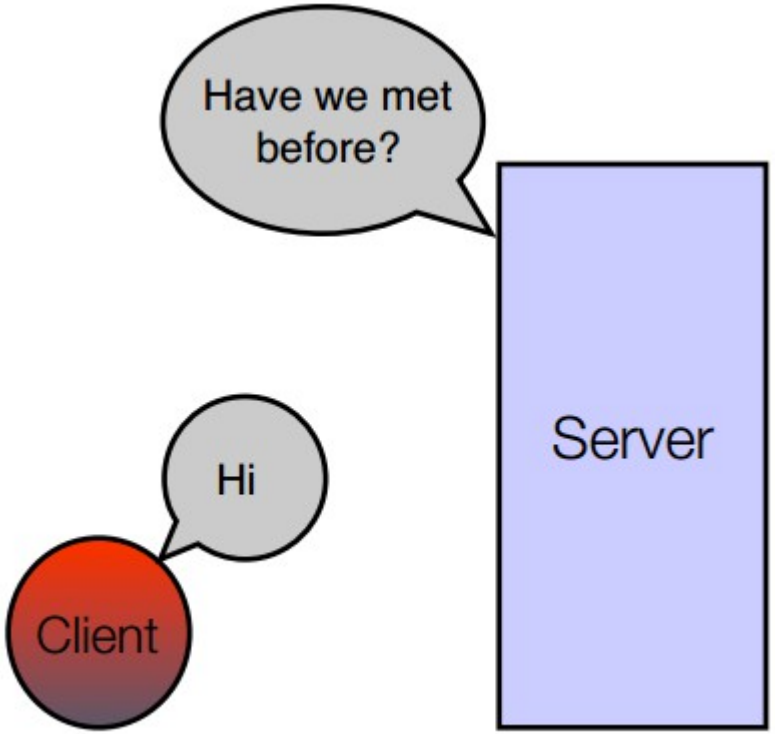
client: visualizza la richiesta di autenticazione ed attende i dati dall'utente. Rispedisce al server username+password e le memorizza (caching)

server: autentica l'utente

client: rispedisce username+password in tutte le richieste successive utilizzando quelle memorizzate nella cache

IL MECCANISMO DEI COOKIES

- HTTP cookies (web cookies, cookies) : uno dei modi per introdurre il concetto di "stato" nel protocollo HTTP
- motivazione per i cookie
- come funzionano in HTTP
- generazione di identificatori ur clients o sessioni



COOKIES: MOTIVAZIONI

- HTTP è un protocollo stateless
- invocazioni successive di funzioni HTTP costituiscono eventi indipendenti, e non vi è alcun meccanismo di interazione diretta
- senza il concetto di stato, è difficile per il server
  - tenere traccia delle richieste successive di un client
  - creare applicazioni web che "ricordino" cosa è accaduto nelle interazioni precedenti
  - associare ad ogni client una identità unica.

IL MECCANISMO DEI COOKIES

- HTTP cookies (web cookies, cookies) : uno dei modi per introdurre il concetto di "stato" nel protocollo HTTP

- cookie: una sequenza di dati in formato testuale (un piccolo file) spediti dal server al client e, successivamente, rispediti dal client al server
  - spesso (ma non sempre) il dato è una stringa che rappresenta un identificatore unico per il client
- sono dati (identificatori) generati dal server, non sono delle porzione di codice

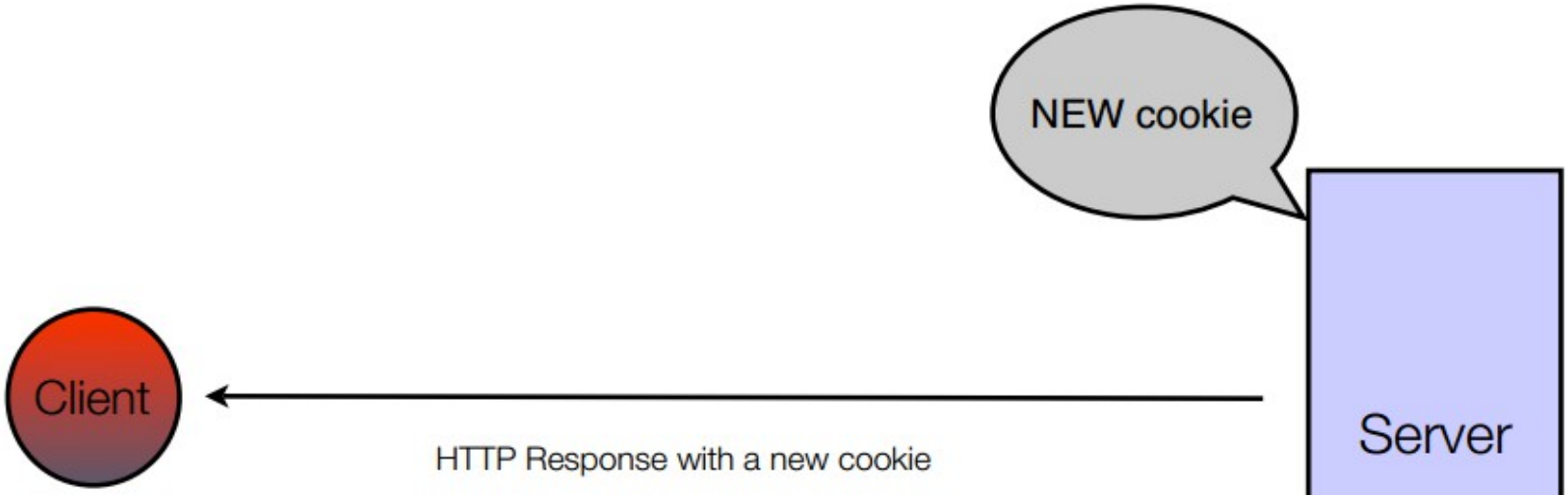
un cookie viene:

- generato dal server quando un client, tramite un browser, effettua una richiesta HTTP
- inviate da un web server al browser (client)
- rispedite senza modifiche dal browser al web server ogni volta che il browser accede nuovamente al sito

UN SEMPLICE SCENARIO: PASSO 1

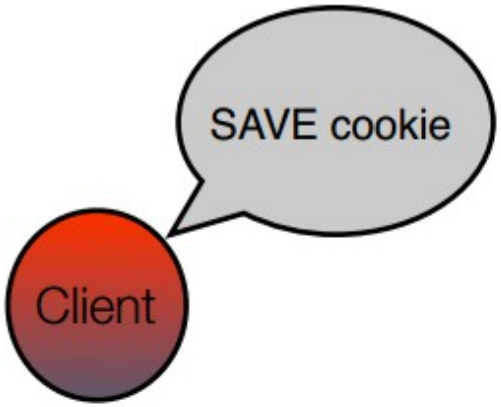
The client (typically a web browser) makes a request to the server for the first time.

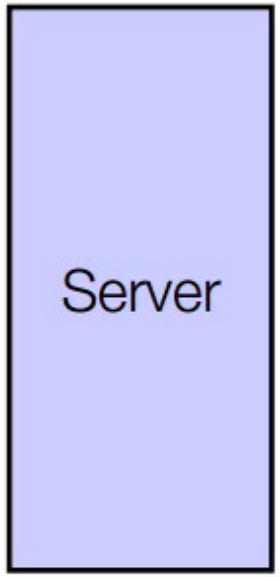
UN SEMPLICE SCENARIO: PASSO 2



The server observes that the client has not sent a cookie. So the server generates a new unique cookie and sends it back to the client as part of its response.

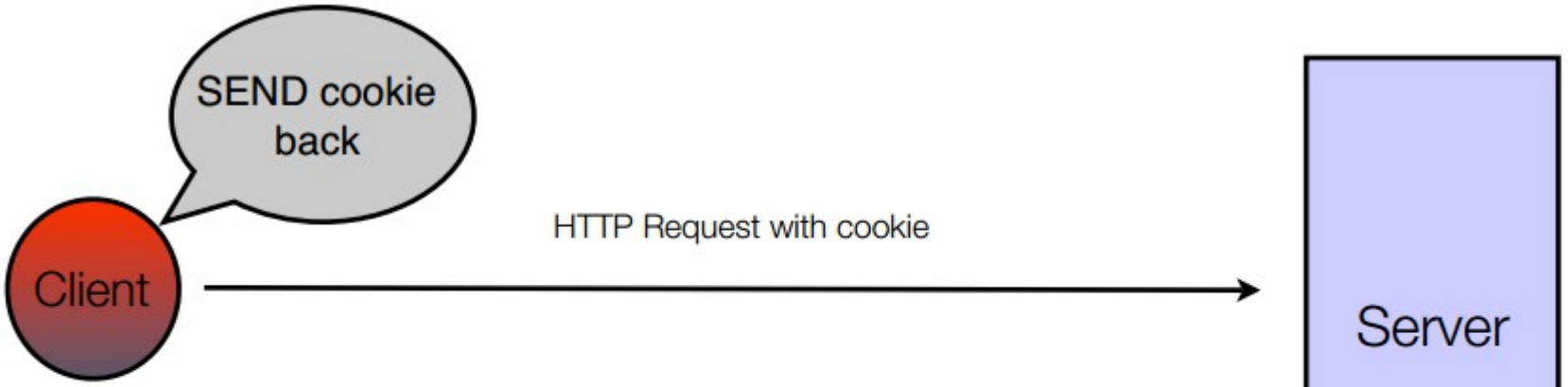
UN SEMPLICE SCENARIO: PASSO 3





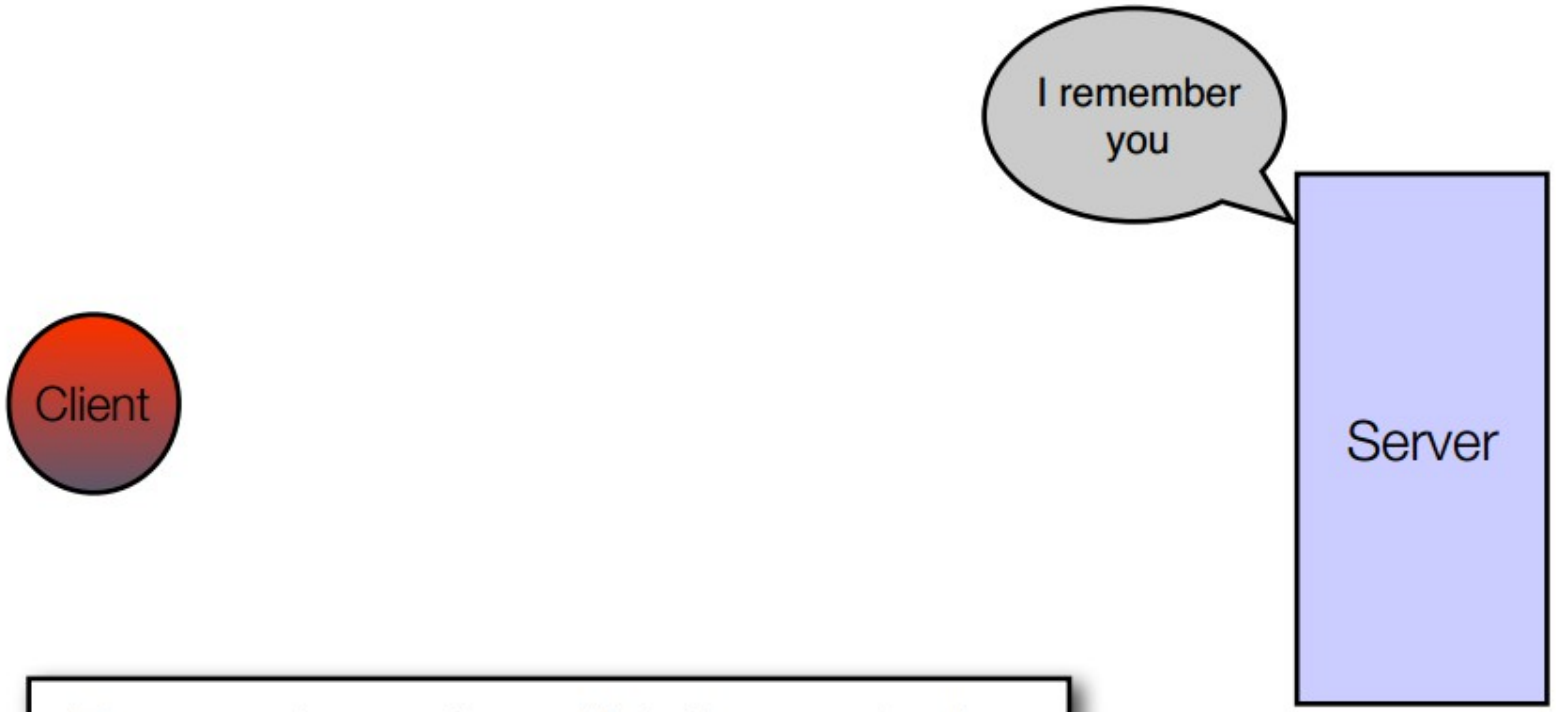
The client saves the new cookie from the server in its private store of cookies. It associates the cookie with the domain name of the server.

UN SEMPLICE SCENARIO: PASSO 4



Later, the client sends another request to the server. This time it observes that is has a cookie saved for the domain name of the server, so it sends the cookie as part of its request.

UN SEMPLICE SCENARIO: PASSO 5



The server observes the cookie in the request and remembers that it sent the cookie in a previous response. The cookie allows the server to identify the client.

IL MECCANISMO DEI COOKIES

- La maggior parte dei browser moderni supportano il meccanismo dei cookies, ma danno all'utente la possibilità di scegliere se accettare i cookie o meno

- In particolare, il browser può

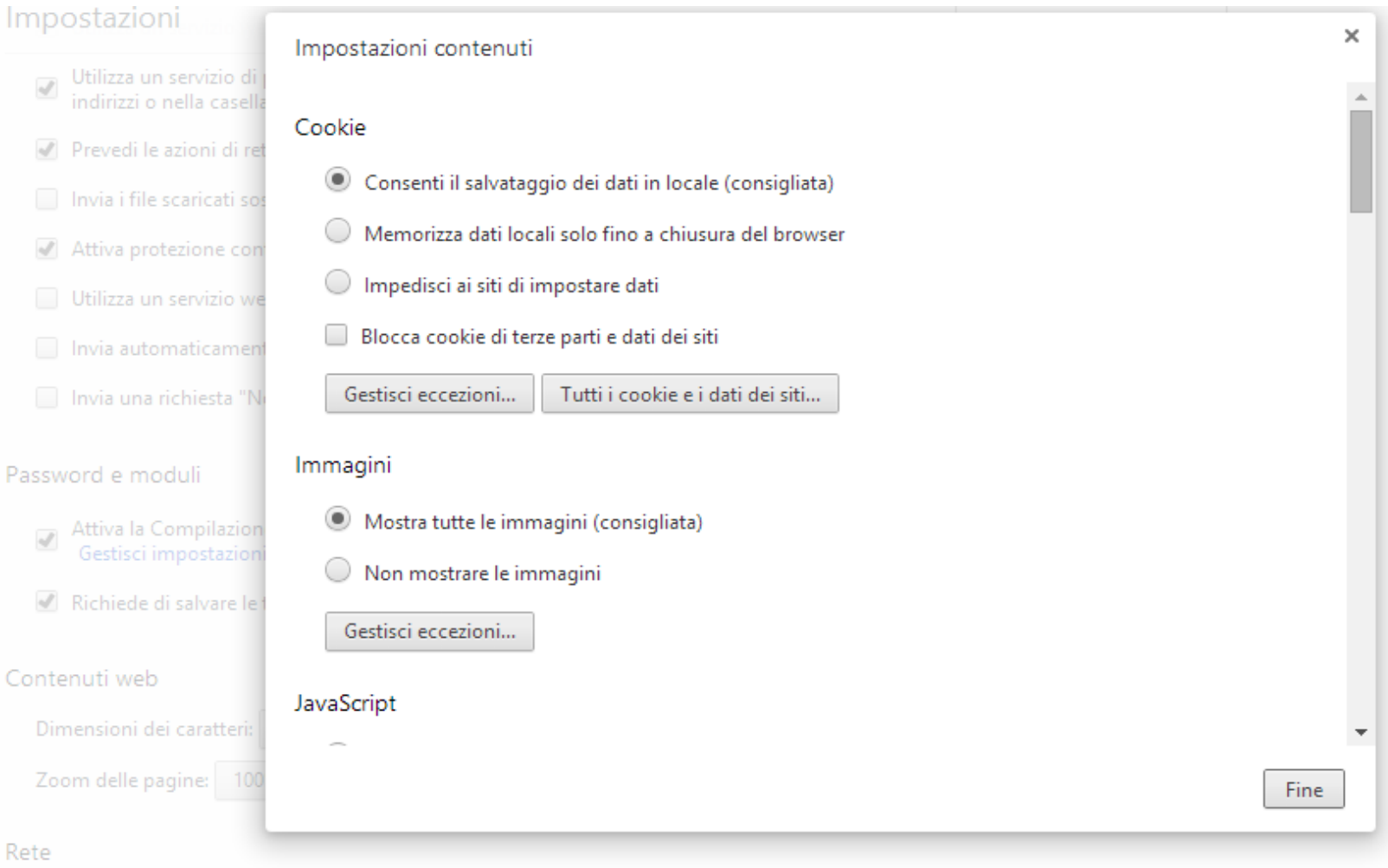
  non accettare mai i cookie

  chiedere all'utente se accettare o meno il cookie

  accettare sempre i cookie

  accettare un cookie solo se contenuto in una lista fornita dall'utente

IL MECCANISMO DEI COOKIES: CHROME



COOKIES: UTILIZZO

- senza i cookie ogni accesso ad una pagina web è un evento isolato (stateless HTTP transaction)

- Restituendo un cookie al server, il browser fornisce al server un modo di collegare la visita corrente alle visite precedenti dello stesso client allo stesso sito

- Possibili utilizzi:
  - ricordare le pagine visitate precedentemente e proporre all'utente dei contenuti personalizzati
  - autenticare un utente senza richiedere username+password ogni volta che l'utente si collega al server
  - ricordare le preferenze dell'utente (es: pagine visitate,…) per riproporle successivamente

COOKIES: UTILIZZO

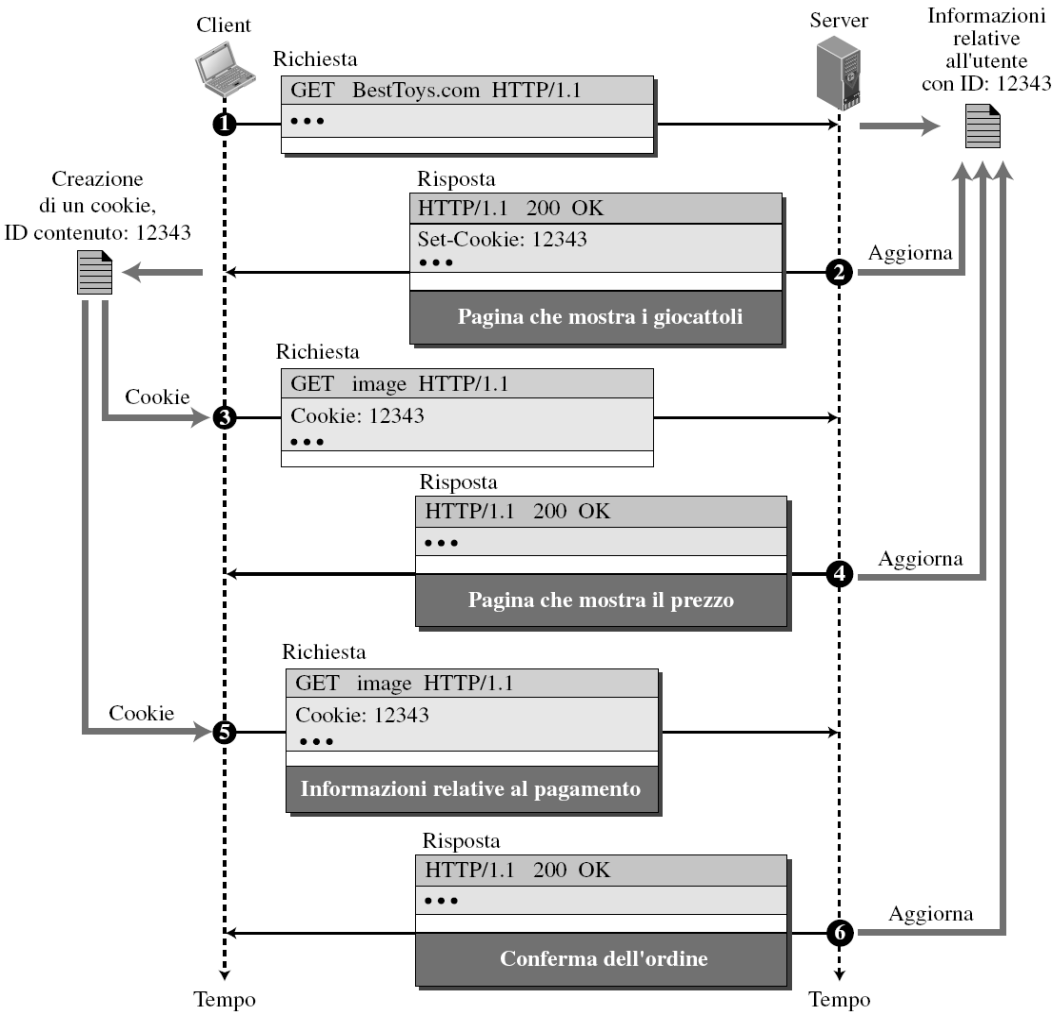
- ricordare le preferenze dell'utente (es: pagine visitate,…) per riproporle successivamente.

L'utente seleziona le sue pagine preferite ed il server crea un cookie che individua le sue preferenze. Al momento dell'accesso successivo, si mostra una pagina personalizzata.

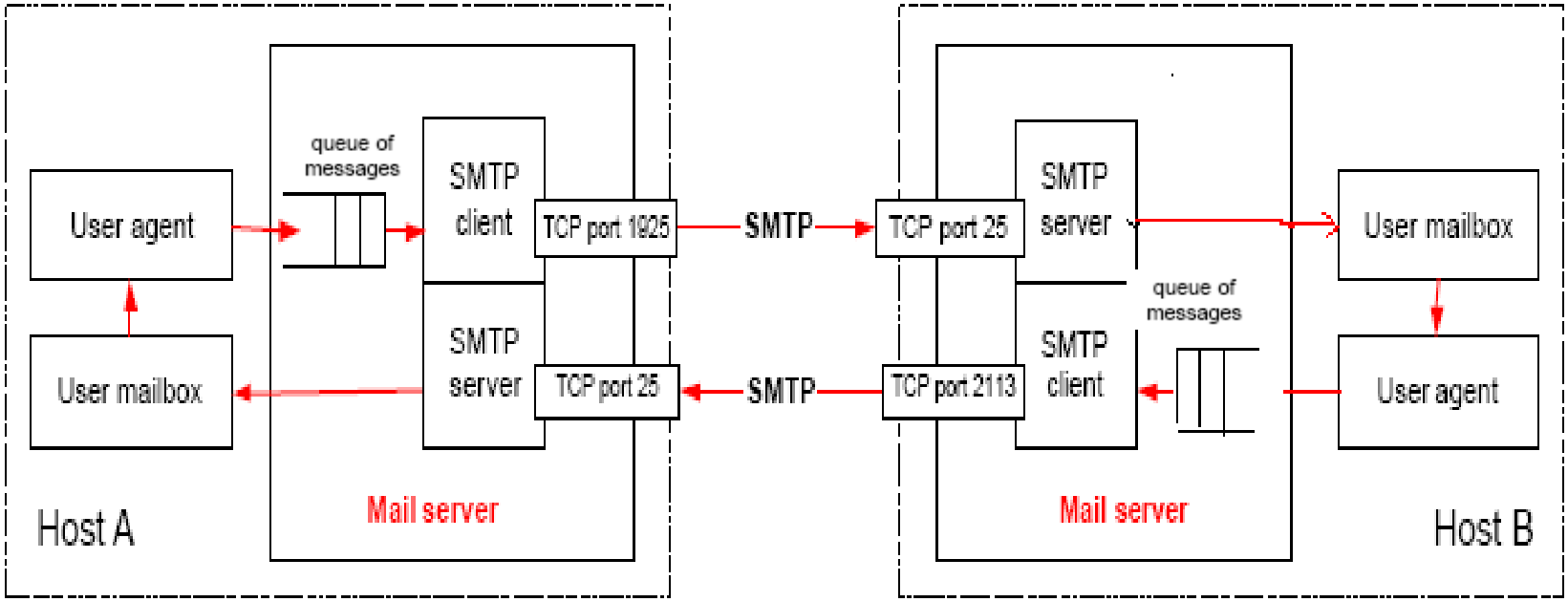
- autenticare un utente senza richiedere username+password ogni volta che l'utente si collega al server. Accesso aperto agli utenti precedentemente registrati

- gestione di un carrello elettronico per un sito di commercio elettronico: memorizzazione nel cookie dei prodotto acquistati
  - il server può calcolare il costo totale in base al cookie

COOKIES: UTILIZZO



LA POSTA ELETTRONICA



LA POSTA ELETTRONICA

- Trasmissione asincrona: la trasmissione e la ricezione dei messaggi avvengono in momenti diversi
- User agent:programma per la gestione dei messaggi di posta elettronica, eudora, elm ,pine, ...
- L'user agent passa la mail ad un mail server
- User agent e mail server possono risiedere su host diversi
- Il mail server utilizza il protocollo SMTP (Simple Mail Transfer Protocol) per trasmettere il messaggio al mail server che gestisce il destinatario della mail
- SMTP utilizza TCP. Il protocollo garantisce che i messaggi vengano effettivamente consegnati
- Per stabilire una connessione con il mailserver del destinatario occorre conoscere l'indirizzo del suo mail server

POSTA ELETTRONICA

```
From: student@local-lab.net
To: instructor@remote-lab.net
Subject: Internet Lab
The dog ate my homework.
```

- Il mail server del mittente interroga il DNS per individuare l'indirizzo IP del destinatario

- Nell'esempio precedente il mail server sull'host mittente interroga il DNS per conoscere ill mail server che e' responsabile del dominio remote-lab.net

- Dopo aver ottenuto l'indirizzo IP, il meil server apre una connessione TCP con il mail server destinatario sulla 'well known port' 25

- Tutti i messaggi vengono trasmessi come stringhe di caratteri

POSTA ELETTRONICA

```
Server:  220 mailserver.remote-lab.net  Mail Server
Client:  HELO local-lab.net
Server:  250 Hello local-lab.net, pleased to meet you
Client:  MAIL FROM: student@local-lab.net
Server:  50 <student@local-lab.net>... Sender ok
Client:  RCPT TO: instructor@remote-lab.net
Server:  250 <instructor@remote-lab.net>... Recipient ok
Client:  DATA
Server:  354 Enter mail, end with "." on a line by itself
Client:  Received: (from student@local-lab.net)
               by mailserver@local-lab.net (8.11.2/8.11.2) id g4OK3nu19221
           for instructor@remote-lab.net; Fri, 24 May 2002 23:03:49 -0400
         Date: Fri, 24 May 2002 23:03:49 -0400
         From: student <student@local-lab.net >
            Message-Id: <200205242003.g4OK3nu19221@mailserver@local-lab.net>
         To: instructor@remote-lab.net
         Subject: Internet Lab

         The dog ate my homework.
Client:  .
Server:  250 QAA01884 Message accepted for delivery
Client:  QUIT
Server:  221 mailserver@remote-lab.net closing connection
```

IL DOMAIN NAME SYSTEM (DNS)

- gli indirizzi IP sono poco adatti per essere memorizzati da utenti umani. possibiltà di associare nomi simbolici agli hosts della rete, ad esempio **fujim3.cli.di.unipi.it**

- indirizzi IP hanno lunghezza fissa e possono essere gestiti in modo semplice dai routers. Inoltre hanno struttura gerarchica in modo da favorire l'instradamento nei routers.

- nomi simbolici a lunghezza variabile adatti per gli utenti

Domain Name System: servizio di rete che consente di tradurre nomi simbolici in indirizzi di risorse di rete

qual'è l'indirizzo IP corrispondente a **fujim3.cli.di.unipi.it?**
qual'è l'indirizzo del mail server che gestisce la posta per il dominio **di.unipi.it?**

IL DOMAIN NAME SYSTEM (DNS)

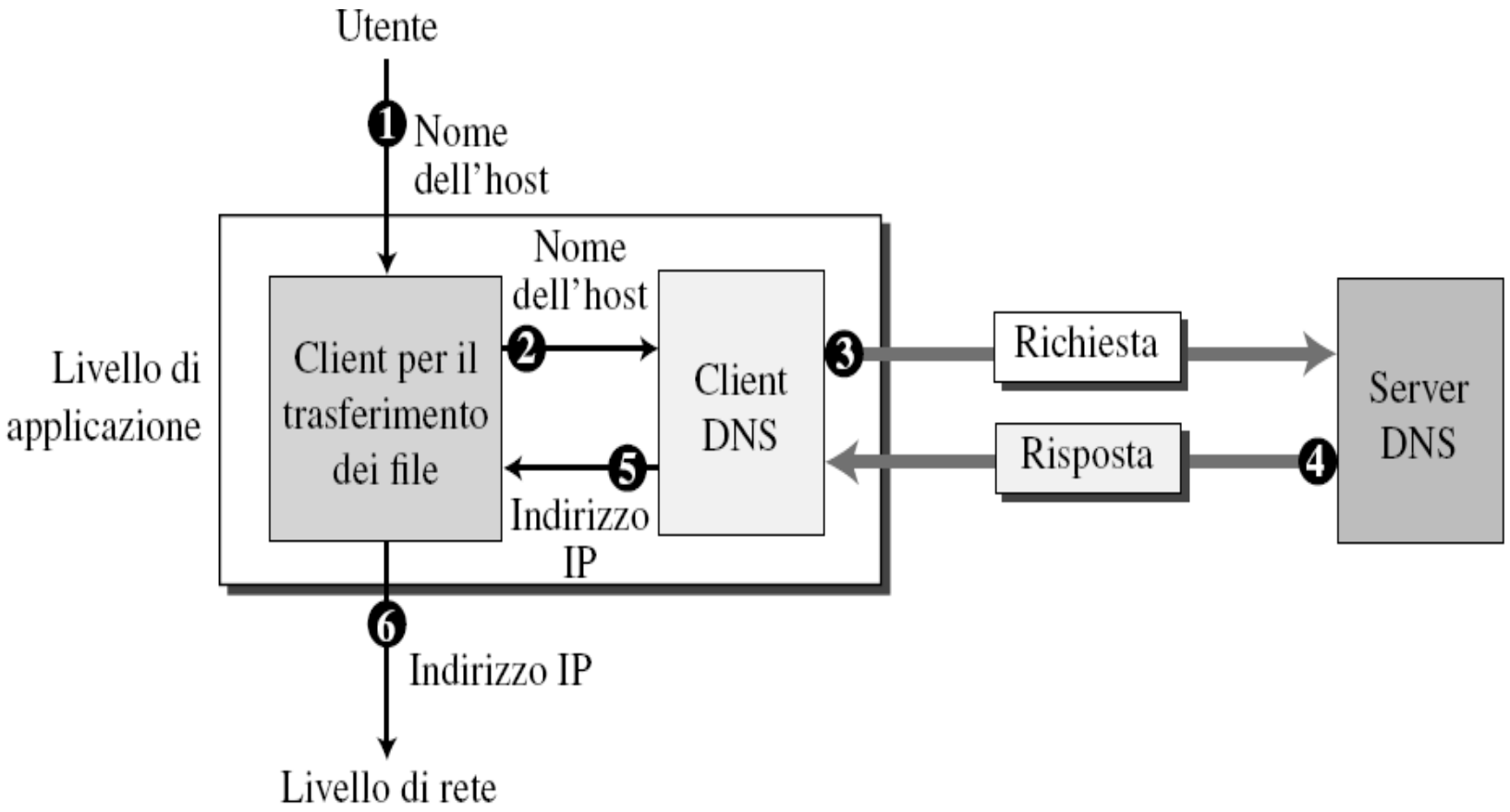
- un data base distribuito che memorizza coppie (nome simbolico - indirizzo IP) su un insieme di nodi della rete (name servers)
- un protocollo a livello applicazione che regola la comunicazione tra hosts e name servers
- basato sul paradigma client/server
- utilizzato da altri protocolli per la risoluzione dei nomi simbolici
  - esempio HTTP estrae dalla URL digitata dall'utente il nome simbolico dell'host e lo traduce in un indirizzo IP

IL DOMAIN NAME SYSTEM

Funzioni principali del DNS

- traduzione nomi indirizzi IP
- gestione degli alias = definire più nomi simbolici per lo stesso host
- bilanciamento del carico
  - un web server (es:`www.cnn.com`) destinato a servire una enorme quantità di richieste può essere replicato (più hosts offrono quel servizio)
  - allo stesso nome simbolico viene associato un insieme di indirizzi IP
  - il DNS restituisce gli indirizzi IP associati allo stesso nome simbolico secondo una disciplina circolare

DNS: ESEMPIO DI UTILIZZO



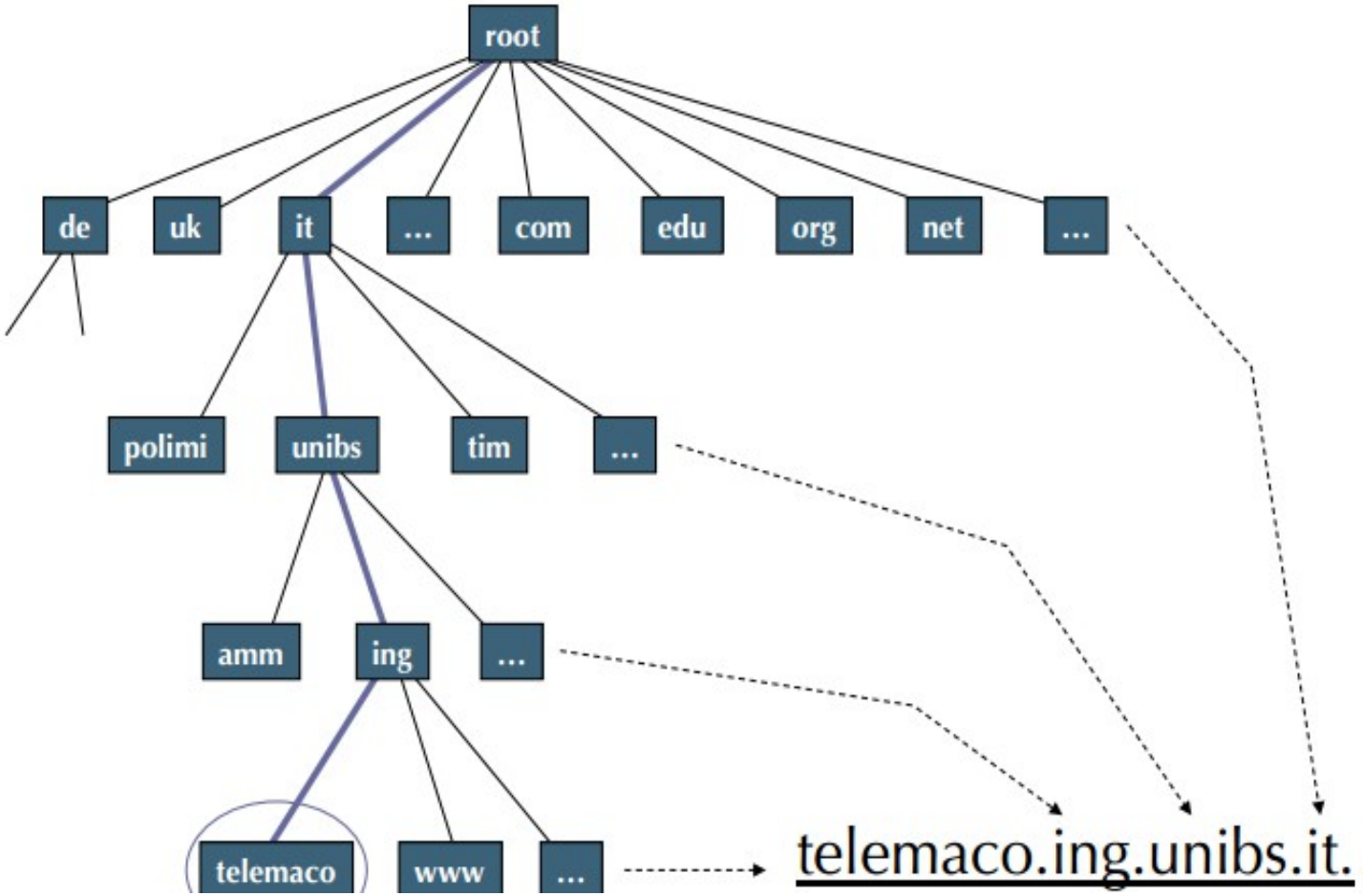
UN PO' DI STORIA: ANNI '80

- lista degli indirizzi IP degli host interconnessi ad ARPANET mantenuta in un singolo file di testo
- il master file veniva quotidianamente scaricato con FTP da un sito centrale, e poi distribuito a tutti gli host di una data sottorete
- oltre al mappaggio nome/indirizzo IP, il file conteneva anche altre informazioni, come indirizzi di rete e relativi nomi simbolici di sottoreti, indirizzi dei gateway responsabili di una certa sottorete, ecc.
- perchè non un singolo server centralizzato?
  - namespace piatto: conflitti sui nomi
  - single point of failure
  - volume di traffico eccessivo
  - mantere un database di queste dimensioni risulta problematico
  - scalabilità limitata !!

DNS: LA STRUTTURA ATTUALE

- specifiche base in [RFC 1034, RFC 1035]
- sistema facilmente estendibile e scalabile:
  - permette mapping di indirizzi IP end-host e "altri"
- componenti del sistema
  - namespace gerarchico, organizzato ad albero
    - ogni livello dell'albero è gestito da un'entità amministrativa potenzialmente indipendente
    - ogni sotto-albero rappresenta un dominio
  - database che contiene il namespace distribuito su diversi name server
  - possibilità di caching locale
  - protocollo client-server
- namespace gerarchico DNS: parallelo con l'indirizzamento usato in altri campi
  - nella rete telefonica (es: +39-050-221.....)
  - nel sistema postale (Mario Rossi, UniPi, via ....., Pisa, Italy)

DNS: LA GERARCHIA DEL NAMESPACE



DNS: LA STRUTTURA ATTUALE

- nell'albero gerarchico del namespace DNS, ogni nodo, con tutti i suoi discendenti, corrisponde a un dominio
  - un dominio rappresenta quindi un sotto-albero nell'albero gerarchico di tutti nodi connessi alla rete, includendo tutti i nomi di risorse e nodi di rete che ne fanno parte.

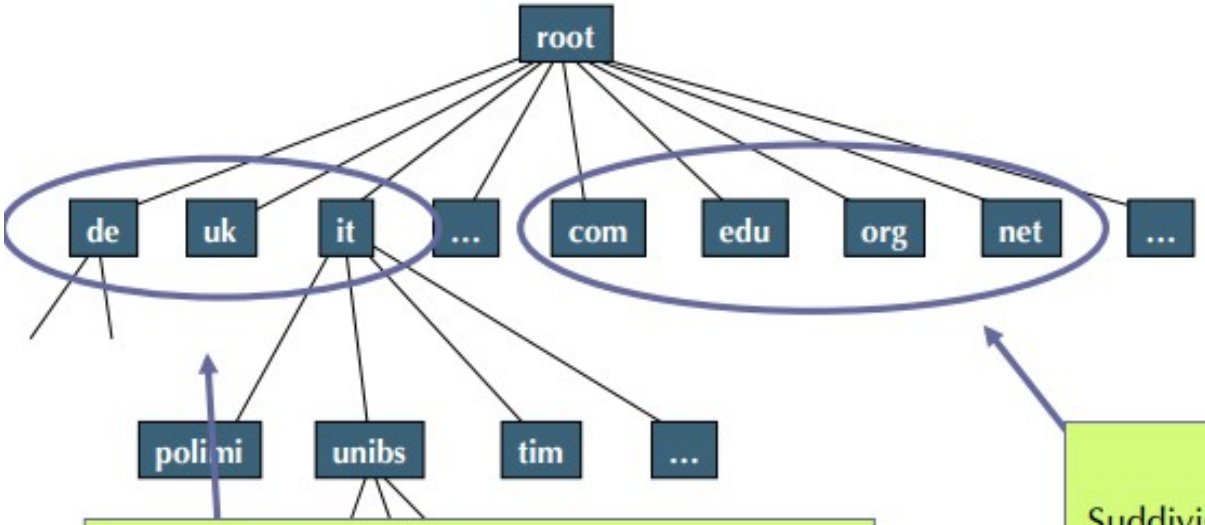
- Esempio
  - il dominio `unipi.it` è un sotto-albero del dominio it, e rappresenta tutti i nomi, reti e nodi di rete che sono in gestione all'Università di Pisa, il cui nome ha la parte più significativa uguale a `unipi.it`
  - il dominio `it`, a sua volta, è un sotto-albero della root

- casi particolari di dominio
  - root: l'intero albero
  - un singolo nodo di rete (`luna.di.unipi.it`)

DNS: NOMI DI DOMINIO

- composto da label, separate da dot ('.'). Il nome di dominio di un nodo nell'albero gerarchico è l'unione ordinata, separata da dot, dei nomi (label) di tutti i nodi che stanno sul percorso dal nodo in questione alla root
- label
  - composta da lettere, numeri e '-'
  - case-insensitive
  - deve iniziare con una lettera
  - massimo 63 caratteri
- massima lunghezza nome di dominio: 255 caratteri
- nome di dominio assoluto (luna.di.unipi.it)
  - termina con un '.'
  - specifica tutte le parti dell'intero percorso, fino alla più significativa
  - ha significato globale e univoco nell'intero albero gerarchico DNS
- nome di dominio relativo
  - ha significato locale, all'interno del dominio di afferenza (luna.di)

DNS: NOMI DI DOMINIO



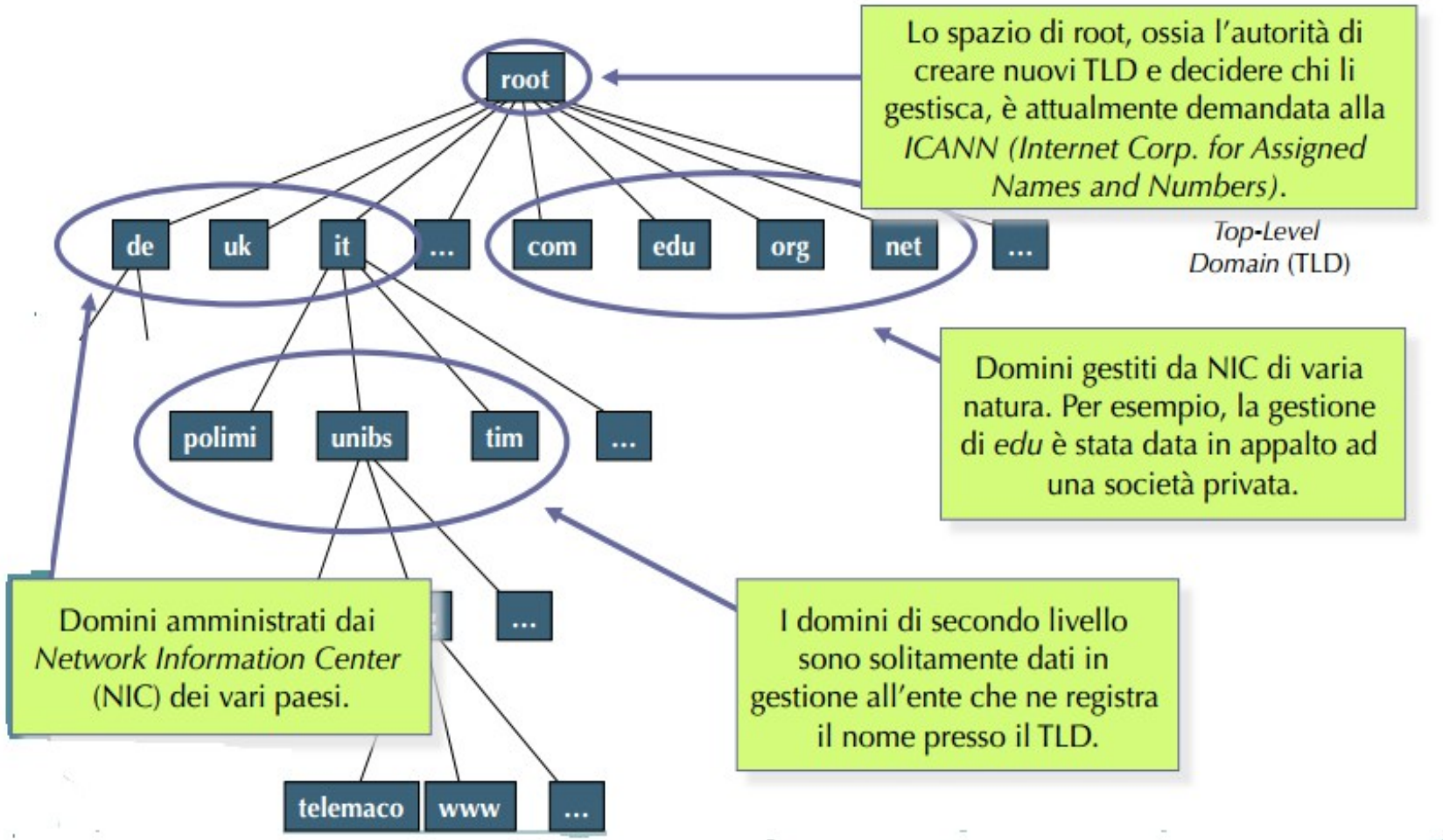
Tutta Internet

Top-Level Domain (TLD)

Suddivisione per **appartenenza geografica**. I domini che fanno capo a questi TLD sono amministrati da enti/società che (*di solito*) geograficamente operano nella specifica area. *NOTA BENE*: questo non significa che gli host appertenenti al dominio risiedano fisicamente in una data area geografica.

Suddivisione per **tipologia di utilizzo** del dominio. Oggi praticamente non più seguita, se non per particolari TLD statunitensi, come *gov, edu, mil*

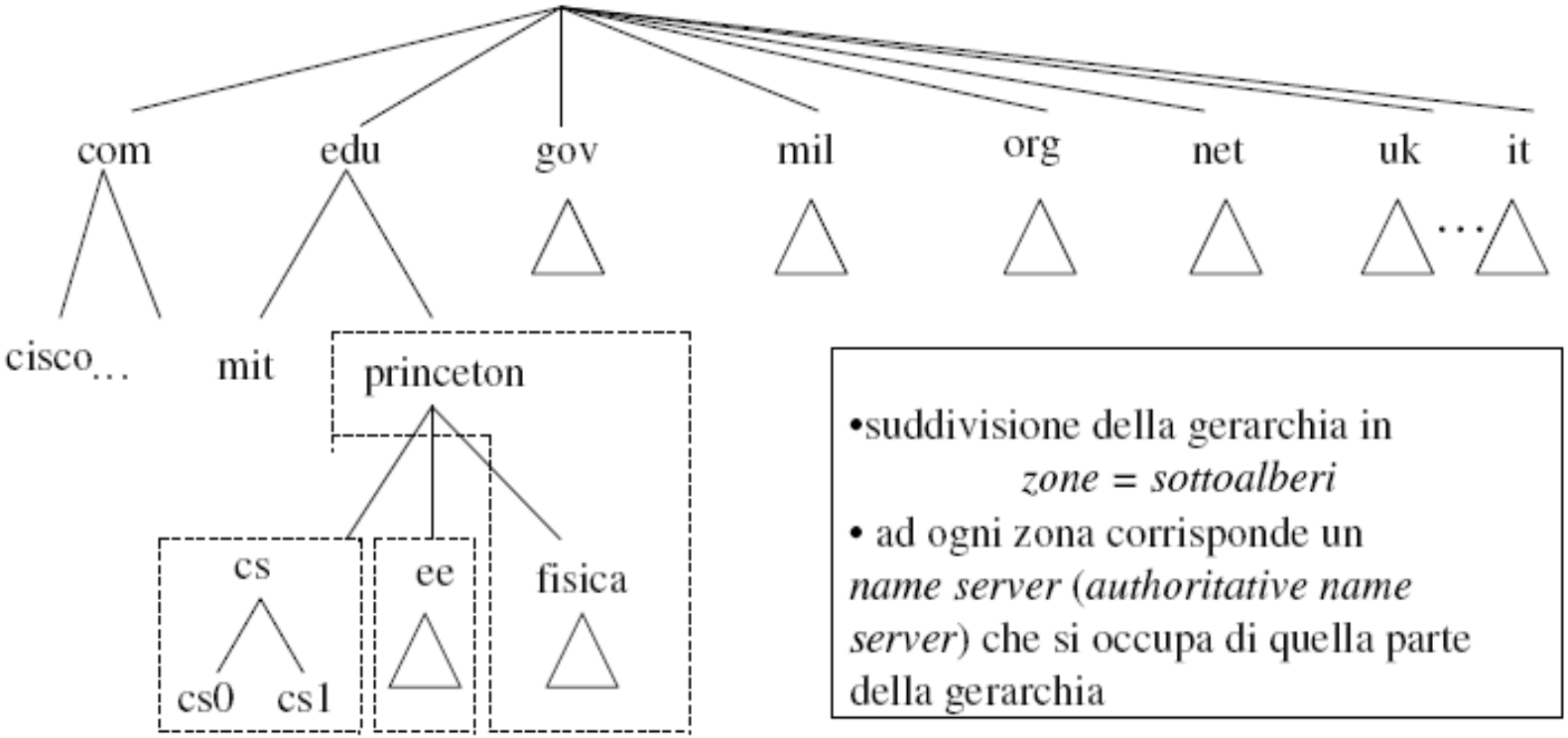
CHI AMMINISTRA IL NAME SPACE DI UN DOMINIO?



DNS: STRUTTURA DEL DATABASE

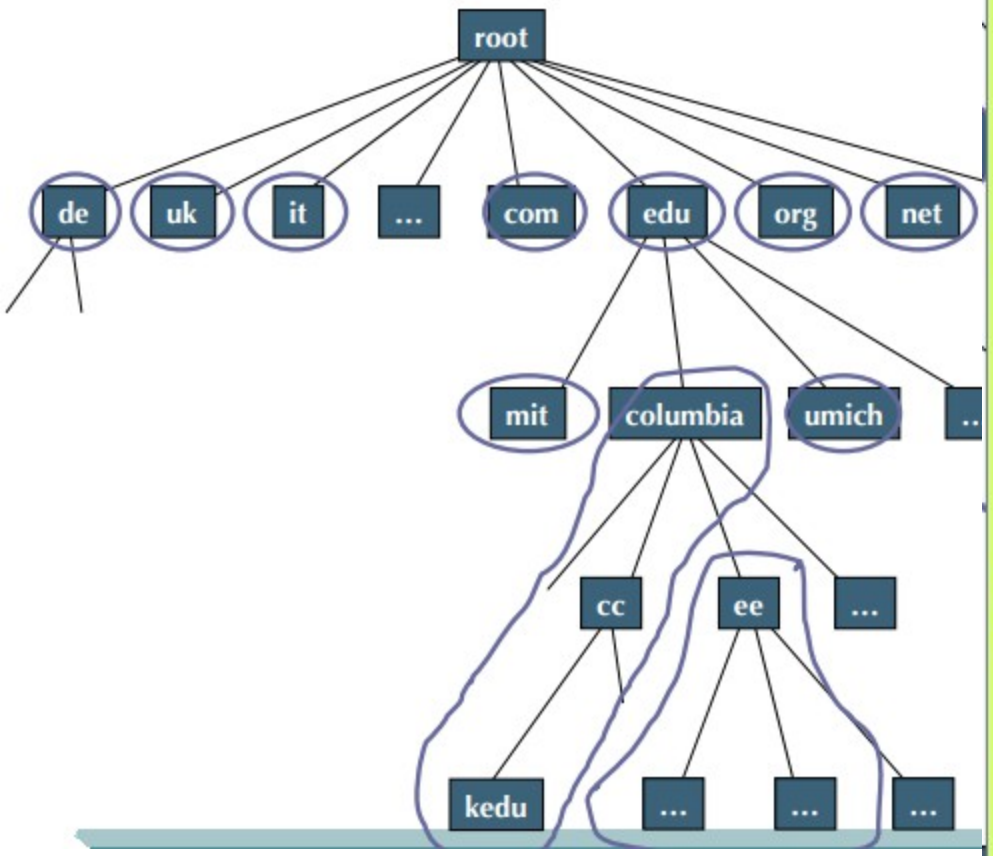
- I database DNS sono composti da resource record (RR)
- Ciascun dominio (= nodo nell'albero del namespace) è rappresentato nel database da un insieme di RR
- Ogni RR specifica le proprietà di una particolare risorsa del dominio a cui si riferisce. Per esempio:
  - Indirizzo IP di un nome appartenente al dominio
  - Indirizzo IP del Mail Server che è responsabile per il dominio
  - Il database DNS contiene la mappa nomi di dominio ⇔ RR
- Campi del RR:
  - nome del dominio
  - TTL (Time to Live)
  - classe
  - tipo
  - valore

DNS: SPAZIO GERARCHICO DEI NOMI



- suddivisione della gerarchia in *zone* = *sottoalberi*
- ad ogni zona corrisponde un *name server* (*authoritative name server*) che si occupa di quella parte della gerarchia

DNS: UN ESEMPIO REALISTICO



Zona: insieme arbitrario di nodi adiacenti nell'albero gerarchico del namespace DNS. Unico requisito: due zone *non possono sovrapporsi*.

Ad ogni zona sono associati due (o più, per backup) ***Name Server***, che gestiscono il database DNS che contiene tutte le RR dei nodi della zona. Uno o più name server di una zona può/possono far parte di un'altra zona, per aggiungere robustezza al sistema. Vi è un NS *primario* più uno o più (ma sempre almeno uno) NS *secondari*.

Authoritative Name Server: è il Name Server che gestisce il database di una zona. Corrispondentemente, i RR contenuti nel database amministrato da quel Name Server sono ***Authoritative RR***. Eccezione: i RR che vengono utilizzati per specificare i NS delle zone "figlie" non sono considerati authoritative.

TIPI DI DNS SERVER

Root Servers.

Ne esistono un numero limitato su tutta la rete (alcune decine). Ciascuno realizzato come un cluster di servers replicati.

Mantengono il database dei RR che contengono gli indirizzi dei NS reponsabili dei TLD

TLD top-level domain server: responsabili dei domini di alto livello, come .com gestiti da enti governativi o privati

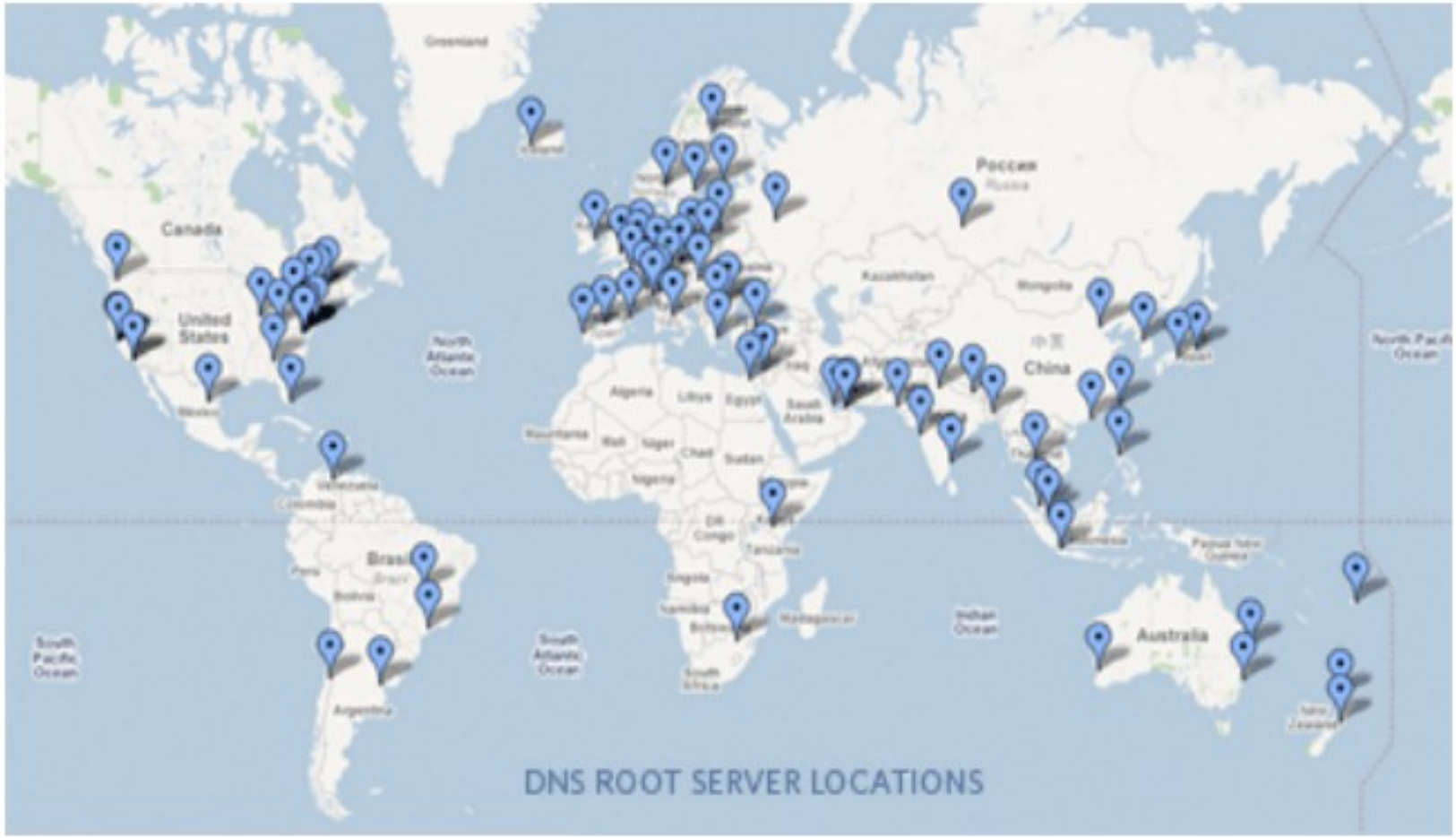
Verisign maintains servers for .com TLD, Educause for .edu TLD

Authoritative Server: Servers associati ad una zona dell'albero dei nomi. Spesso DNS di organizzazioni private

mantenuti dalla organizzazione o dal server privato

devono fornire accesso pubblico per i record RR della organizzazione privata

ROOT NAME SERVER MAP



Oltre 245 servers
Ognuno contiene gli stessi dati

DNS: STRUTTURA DEL DATABASE

- DNS locale: non "strettamente" appartenente alla gerarchia
  - ogni ISP (residential ISP, company, university) ne possiede uno

- Quando un host effettua una query al DNS, la query viene inviata al suo DNS locale

- Possiede una cache locale delle traduzioni recenti nome-indirizzo (che però può essere obsoleta)

- Funziona come un proxy forwarda la query nell gerarchia

- Servizio DNS in esecuzione sulla porta 53, usa UDP.

IL DOMAIN NAME SYSTEM

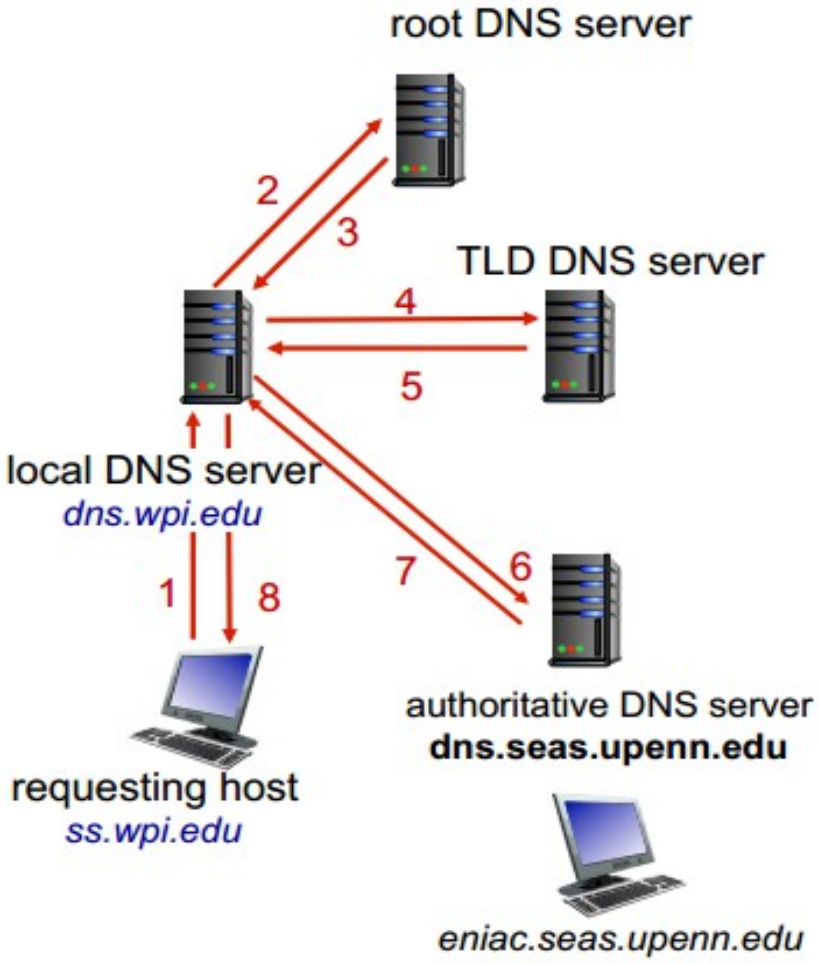
Name Server Locale : Associato ad un ad una organizzazione *O* (università, dipartimento, industria,…).

- contiene le associazioni (nome-indirizzo IP) per tutti gli host di *O*
- la ricerca dell'inizio IP associato ad un nome simbolico inizia sempre dal name server locale.
- l'indirizzo IP del name server locale può essere impostato da ogni hostalmomento della configurazione dell'host
- se il name server locale non riesce a risolvere il nome, inoltra la richiesta ad un root name server. Quest'ultimo
  - se riesce a risolvere il nome,lo inoltra all'host che lo ha richiesto
  - altrimenti lo invia a un altro name server che possiede il mapping ricercato, oppure conosce l'indirizzo IP di un altro DNS in grado di risolvere il nome

DNS: RISOLUZIONE ITERATIVA DI NOMI

- L'host `ss.wpi.edu` richide l'indirizzo IP address per `eniac.seas.upenn.edu`

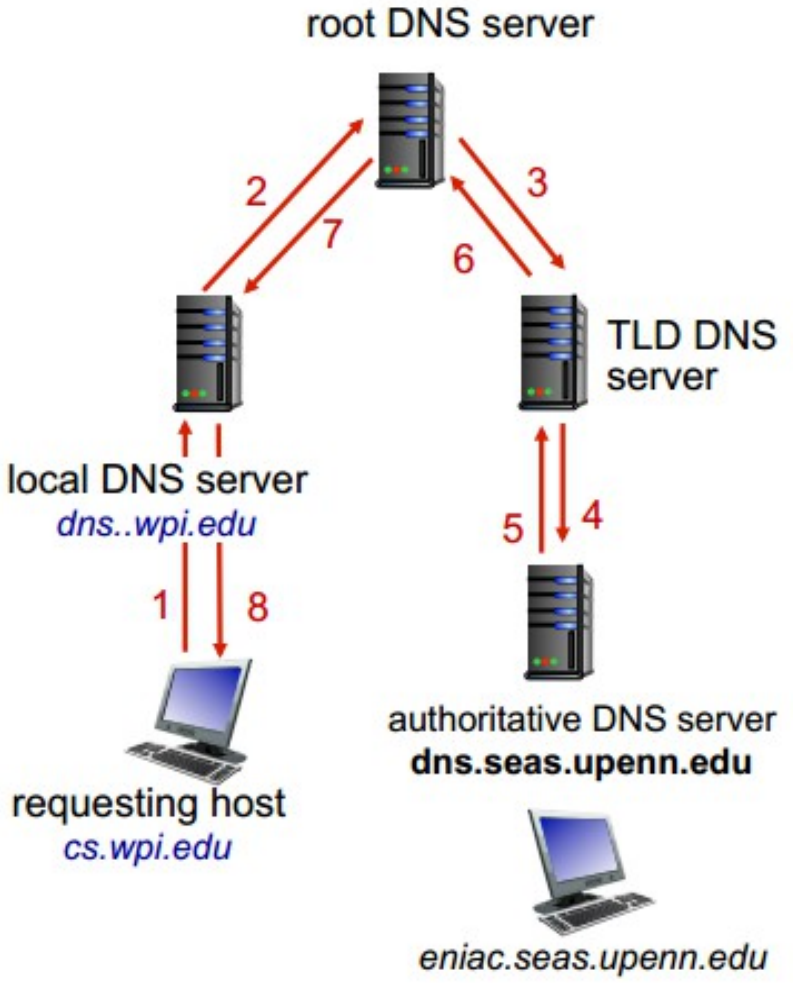
- Risoluzione iterativa della query:
  - ogni server contattato risponde con il nome di un altro server da contattare
  - "non conosco questo nome, ma conosco un altro server che deve essere contattato"



DNS: RISOLUZIONE RICORSIVA DI NOMI

- L'host **ss.wpi.edu richide** l'indirizzo IP address per **eniac.seas.upenn.edu**

- Risoluzione ricorsiva della query:
  - Carico computazionale per la risoluzione sui name server contattati
  - Alto carico ai livelli più alti della gerarchia



DNS: TECNICHE DI CACHING

- quando un name server riceve un mapping nome simbolico-indirizzo IP,lo memorizza nella sua memoria locale (cache)
- se successivamente riceve la richiesta per lo stesso nome simbolico, può risolvere il nome localmente, senza propagare la query
- Esempio: `fujim3.cli.di.unipi.it` richiede a `dns.di.unipi.it` di risolvere il nome simbolico `google.it`. Dns.di.unipi.it risolve il nome in indirizzo IP e memorizza il mapping nella propria cache. E' molto probabile che un altro host del dominio `cli.di.unipi.it` richieda entro un breve lasso di tempo la risoluzione dello stesso nome simbolico
- un mapping viene scartato dopo un certo intervallo di tempo (tipicamente due giorni)
- I meccanismi di caching sono introdotti per migliorare il tempo di risposta ad una richiesta di risluzione di un nome simbolico e per diminuire il numero di messaggi spediti sulla rete

DNS: ESPERIENZA DI LABORATORIO

Obiettivi

- mostrare come operano le tecniche di caching
- poichè è complesso mostrare le tecniche di caching a livello di server DNS, si possono mostrare a livello della JAVA Virtual Machine
- evidenziare che esistono diversi livelli di caching
- utilizzo dell'API JAVA

DNS: ESPERIENZA DI LABORATORIO

```
import java.net.*;
public class CachingDNS {
  public static void main (String[] args)
   {long start_time = System.currentTimeMillis();
   java.security.Security.setProperty("networkaddress.cache.ttl","1000");
   for (int i=0; i<1000; i++)
      try{
          {InetAddress ia =InetAddress.getByName("www.google.it");} }
      catch (Exception e){System.out.println(e);};
   long end_time = System.currentTimeMillis();
   long difference = end_time-start_time;
   System.out.println("la differenza e'"+difference);}}
```

Provare il programma con diversi valori della proprietà networkaddress.cache