Appunti del corso di
Sistemi di elaborazione: Reti I

PROF. G. BONGIOVANNI

7) Il livello cinque (Application)

Nel nostro modello di architettura (e anche nell'architettura TCP/IP) sopra il livello transport c'è il livello application, nel quale viene effettivamente svolto il lavoro utile per l'utente.

In questo livello si trovano diverse tipologie di oggetti:
- protocolli di supporto a tutte le applicazioni, come per esempio il ***DNS*** (***Domain Name System***, RFC 1034 e 1035);
- protocolli di supporto ad applicazioni di tipo standardizzato, come ad esempio:
  - ***SNMP*** (***Simple Network Management Protocol***, RFC 1157) per la gestione della rete;
  - ***FTP*** (***File Transfer Protocol***, RFC 959) per il trasferimento di file;
  - ***SMTP*** e ***POP3*** (***Simple Mail Transfer Protocol***, RFC 821, e ***Post Office Protocol***, RFC 1225) per la posta elettronica;
  - ***HTTP*** (***HyperText Transfer Protocol***, RFC 1945) alla base del ***World Wide Web*** (***WWW***);
- applicazioni scritte in conformità ai protocolli di cui sopra;
- applicazioni proprietarie, basate su regole di dialogo private (ad esempio, un'applicazione di tipo client/server per la gestione remota di un magazzino).

7.1) Il DNS

Poiché riferirsi a una risorsa (sia essa un host oppure l'indirizzo di posta elettronica di un utente) utilizzando un indirizzo IP numerico (della forma x.y.z.w) è estremamente scomodo, si è creato un meccanismo tramite il quale tali risorse possono essere identificate tramite un ***nome logico***, cioé una stringa di caratteri (molto pù comprensibile per un essere umano) quale ad esempio:
- `sparc1.unimi.it` (riferimento ad un host);
- `john@cern.ch` (indirizzo di posta elettronica).

La corrispondenza fra gli indirizzi IP numerici ed i nomi logici si effettua mediante l'uso del DNS.

Esso consiste di:
1. uno ***schema gerarchico di nominazione***, basato sul concetto di dominio (***domain***);
2. un ***database distribuito*** che implementa lo schema di nominazione;
3. un ***protocollo*** per il mantenimento e la distribuzione delle informazioni sulle corrispondenze.

Il funzionamento, in breve, è il seguente:

- quando un'applicazione deve collegarsi ad una risorsa di cui conosce il nome logico (ad es. sparc1.unimi.it), invia una richiesta al ***DNS server*** locale (l'applicazione chiama per questo una apposita procedura di libreria detta ***resolver***);
- il DNS server locale, se conosce la risposta, la invia direttamente al richiedente. Altrimenti interroga a sua volta un DNS server di livello superiore, e così via. Quando finalmente arriva la risposta, il DNS server locale la passa al richiedente;
- quando l'applicazione riceve la risposta (costituita del numero IP della risorsa in questione) crea una connessione TCP con la (o spedisce segmenti UDP alla) destinazione, usando l'indirizzo IP testé ricevuto.

Lo spazio dei nomi DNS è uno ***spazio gerarchico***, ***organizzato in dominii***, ciascuno dei quali può avere dei sottodominii.

Esiste un insieme di dominii di massimo livello (***top-level domain***), i più alti nella gerarchia.

Nel caso di un host, la forma del nome logico è costituita da un certo numero di sottostringhe separate da punti, come nell'esempio seguente:

host.subdomain3.subdomain2.subdomain1.topleveldomain

dove:

- la prima sottostringa (quella più a sinistra) identifica il nome dell'host;
- le altre sottostringhe (tranne quella più a destra) identificano ciascuna un sottodominio del dominio di cui alla sottostringa seguente;
- l'ultima sottostringa (quella più a destra) identifica il top-level domain di appartenenza.

Per gli USA sono definiti, fra gli altri, i seguenti top-level domain:

com	aziende
edu	università
gov	istituzioni governative
mil	istituzioni militari
net	fornitori d'accesso
org	organizzazioni non-profit

Fuori degli USA, ogni nazione ha un suo top-level domain. Ad esempio:

au	Australia
ch	Svizzera
fr	Francia
it	Italia
jp	Giappone
uk	Inghilterra

Ogni dominio è responsabile della creazione dei suoi sottodominii, che devono essere registrati presso una appositàautorità.

Esempi di sottodominii sono:

cern.ch	il CERN a Ginevra, Svizzera
cnr.it	il Consiglio Nazionale delle Ricerche, Italia
mit.edu	il Massachussetts Institute of Technology, USA
nasa.gov	la NASA, USA
unige.ch	Università di Ginevra, Svizzera
uniroma1.it	Università di Roma "La Sapienza", Italia
dsi.uniroma1.it	Dip. di Scienze dell'Informazione, Università di Roma "La Sapienza"

L'estensione di un dominio è del tutto indipendente da quella delle reti e sottoreti IP.

Ogni dominio ha la responsabilità di fornire il servizio DNS per quanto di propria competenza. Ossia, deve poter rispondere a interrogazioni riguardanti tutti gli host contenuti nel dominio stesso.

Ciò si ottiene predisponendo un ***name server*** (o più di uno), che è un processo in grado di gestire le seguenti informazioni:

- informazioni di corrispondenza fra nomi simbolici e indirizzi IP. Per ogni host del dominio esiste un ***resource record*** che contiene tali informazioni; tale record è detto ***authoritative record***, in quanto è gestito dal DNS server responsabile del dominio (che è supposto fornire sempre informazioni corrette e aggiornate);

- l'identità dei name server responsabili dei sottodominii inclusi nel dominio, così da poter inviare loro le richieste che gli pervengono dall'alto della gerarchia;
- l'identità del name server responsabile del dominio di livello immediatamente superiore, così da potergli inviare le richieste che gli pervengono dal basso della gerarchia.

Una richiesta che arriva a un name server può dunque viaggiare verso l'alto nella gerarchia oppure (dal momento in cui perviene a un top-level domain server) verso il basso, a seconda dei casi. Quando una risposta ritorna indietro, essa viene tenuta dal server in una sua cache per un certo periodo; qui costituisce un nuovo record, detto ***cached record*** perché contiene della informazione che potrebbe anche divenire, col passare del tempo, obsoleta e non più corretta.

Un esempio di resource record (relativo a un host) è:

```
spcw.dsi.uniroma1.it 86400 IN A 151.100.17.110
```

dove:

`spcw.dsi.uniroma1.it`	***domain_name***: nome simbolico.
`86400`	***time_to_live***: la quantità di tempo (in secondi) trascorsa la quale il record viene tolto dalla cache.
`IN`	***class***: classe del record (Internet in questo caso).
`A`	***type***: tipo del record (Address in questo caso).
`151.100.17.110`	***value***: indirizzo IP numerico.

7.2) La posta elettronica

La ***posta elettronica*** è uno dei servizi più consolidati ed usati nelle reti. In Internet è in uso da circa 20 anni, e prima del WWW era senza dubbio il servizio più utilizzato.

Un servizio di posta elettronica, nel suo complesso, consente di effettuare le seguenti operazioni:
- comporre un messaggio;
- spedire il messaggio (a uno o più destinatari);
- ricevere messaggi da altri utenti;
- leggere i messaggi ricevuti;

- stampare, memorizzare, eliminare i messaggi spediti o ricevuti.

Di norma, un messaggio ha un formato ben preciso. In Internet un messaggio ha un formato (definito nell'RFC 822) costituito da un ***header*** e da un ***body***, separati da una linea vuota.

Lo header è a sua volta costituito da una serie di linee, ciascuna relativa a una specifica informazione (identificata da una parola chiave che è la prima sulla linea); alcune informazioni sono:

`To`	indirizzo di uno o più destinatari.
`From`	indirizzo del mittente.
`Cc`	indirizzo di uno o più destinatari a cui si invia per conoscenza.
`Bcc`	blind Cc: gli altri destinatari non sanno che anche lui riceve il messaggio.
`Subject`	argomento del messaggio.
`Sender`	chi materialmente effettua l'invio (ad es. nome della segretaria).

Il body contiene il testo del messaggio, in caratteri ASCII. L'ultima riga contiene solo un punto, che identifica la fine del messaggio.

Gli indirizzi di posta elettronica in Internet hanno la forma:

username@hostname

dove username è una stringa di caratteri che identifica il destinatario, e hostname è un nome DNS oppure un indirizzo IP.

Ad esempio, bongiovanni@dsi.uniroma1.it è l'indirizzo di posta elettronica dell'autore di queste dispense.

La posta elettronica viene implementata in Internet attraverso la cooperazione di due tipi di sottosistemi:
- ***Mail User Agent*** (***MUA***);
- ***Mail Transport Agent*** (***MTA***).

Il primo permette all'utente finale di:
- comporre messaggi;
- consegnarli a un MTA per la trasmissione;
- ricevere e leggere messaggi;
- salvarli o eliminarli.

Il secondo si occupa di:
- trasportare i messaggi sulla rete, fino alla consegna a un MTA di destinazione;
- rispondere ai MUA dei vari utenti per consegnare loro la posta arrivata; in questa fase l'MTA richiede ad ogni utente una password per consentire l'accesso ai messaggi.

Corrispondentemente, sono definiti due protocolli principali per la posta elettronica:
- ***SMTP*** (***Simple Mail Transfer Protocol***, RFC 821) per il trasporto dei messaggi:
    - dal MUA di origine ad un MTA;
    - fra vari MTA, da quello di partenza fino a quello di destinazione;
- ***POP3*** (***Post Office Protocol*** versione 3, RFC 1225) per la consegna di un messaggio da parte di un MTA al MUA di destinazione.

Recentemente sono stati introdotti altri protocolli più sofisticati, quali ***IMAP*** (***Interactive Mail Acces Protocol***, RFC 1064) e ***DMSP*** (***Distributed Mail System Protocol***, RFC 1056), il cui supporto però non è ancora molto diffuso nel software disponibile agli utenti.

Come avviene la trasmissione di un messaggio? Supponiamo che l'utente

`pippo@topolinia.wd`

spedisca un messaggio a

`minnie@paperinia.wd`

e immaginiamo che:
- Pippo usi un MUA configurato per consegnare la posta ad un SMTP server in esecuzione sull'host `mailer.topolinia.wd`;
- Minnie abbia un MUA configurato per farsi consegnare la posta da un POP3 server in esecuzione sull'host `mailer.paperinia.wd`.

La sequenza di azioni che hanno luogo è la seguente:
1. Pippo compone il messaggio col suo MUA, che tipicamente è un programma in esecuzione su un PC in rete;
2. appena Pippo preme il pulsante SEND, il suo MUA:
    - interroga il DNS per sapere l'indirizzo IP dell'host `mailer.topolinia.wd`;

    - apre una connessione TCP ed effettua una conversazione SMTP con il server SMTP in esecuzione sull'host `mailer.topolinia.wd`, per mezzo della quale gli consegna il messaggio;
    - chiude la connessione TCP;
3. Pippo se ne va per i fatti suoi;
4. il server SMTP di `mailer.topolinia.wd`:
    - chiede al DNS l'indirizzo IP di `paperinia.wd`;
    - scopre che è quello dell'host `mailer.paperinia.wd`;
    - apre una connessione TCP e poi una conversazione SMTP con il server SMTP in esecuzione su quell'host e gli consegna il messaggio scritto da Pippo;
5. Minnie lancia il suo MUA;
6. appena Minnie preme il pulsante "check mail", il suo MUA:
    - interroga il DNS per avere l'indirizzo IP dell'host `mailer.paperinia.wd`;
    - apre una connessione TCP e poi una conversazione POP3 col server POP in esecuzione su `mailer.paperinia.wd` e preleva il messaggio di Pippo, che viene mostrato a Minnie.

Si noti che `topolinia.wd` e `paperinia.wd` in genere corrispondono a un dominio nel suo complesso e non ad un singolo host, al fine di rendere gli indirizzi di posta elettronica indipendenti da variazioni del numero, dei nomi logici e degli indirizzi IP degli host presenti nel dominio.

Nel DNS ci sono opportuni record detti di tipo ***MX*** (***Mail Exchange***), che si occupano di indicare quale host effetttivamente fa da server SMTP per un dominio.

Nel nostro esempio avremo, nel DNS server di Paperinia, i record:

```
mailer.paperinia.wd A     100.10.10.5
paperinia.wd        MX    mailer.paperinia.wd
```

Il secondo record è un record di tipo MX, e indica che tutta la posta in arrivo per

`chiunque@paperinia.wd`

deve essere consegnata all'host `mailer.paperinia.wd`, e cioé quello che ha l'indirizzo IP

`100.10.10.5`

Inoltre, non è detto che `mailer.topolinia.wd` consegni i messagi direttamente a `mailer.paperinia.wd`. E' possibile che le macchine siano configurate in modo da trasferire i messaggi attraverso un certo numero di server SMTP intermedi.

Infine, vanno citate due significative estensioni di funzionalità della posta elettronica, in via di progressiva diffusione:

- possibilità di inviare messaggi di posta contenenti informazioni di qualunque tipo (per esempio programmi eseguibili, immagini, filmati, suoni, ecc.) attraverso lo standard ***MIME*** (***Multipurpose Internet Mail Extension***, RFC 1341 e 1521);
- possibilità di inviare messaggi corredati di firma digitale o crittografati, attraverso lo standard in via di definizione ***S/MIME*** (***Secure/MIME***, RFC 1847).

7.3) Il World Wide Web

Il ***World Wide Web*** (detto anche ***Web***, ***WWW*** o ***W³***) è nato al Cern nel 1989 per consentire una agevole cooperazione fra i gruppi di ricerca di fisica sparsi nel mondo.

E' un'architettura software volta a fornire l'accesso e la navigazione a un enorme insieme di documenti collegati fra loro e sparsi su milioni di elaboratori.

Tale insieme di documenti forma un ***ipertesto*** (***hypertext***), cioè un testo che viene percorso in modo non lineare. Il concetto di ipertesto risale alla fine degli anni '40, e si deve a vari scienziati:

- Vannevar Bush (sistema Memex, basato su microfilm);
- Douglas Engelbart (sistema NLS/Augment, basato su elaboratori interconnessi);
- Ted Nelson (sistema Xanadu, con enfasi sulla tutela dei diritti d'autore: un documento poteva contenere un riferimento ad altri documenti, che venivano inclusi "al volo" in quello referente e mantenevano così la loro unicità e originalità).

Il Web ha diverse caratteristiche che hanno contribuito al suo enorme successo:

- architettura di tipo client-server:
  - ampia scalabilità;
  - adatta ad ambienti di rete;
- architettura distribuita:
  - perfettamente in linea con le esigenze di gestione di un ipertesto;
- architettura basata su standard di pubblico dominio:
  - possibilità per chiunque di proporre una implementazione;
  - uguali possibilità di accesso per tutte le piattaforme di calcolo;
- capacità di gestire informazioni di diverso tipo (testo, immagini, suoni, filmati, realtà virtuale, ecc.):
  - grande interesse da parte di tutti gli utenti.

I documenti che costituiscono l'ipertesto gestito dal Web sono detti ***pagine web***, e possono contenere, oltre a normale testo formattato, anche:

- rimandi (detti ***link*** o ***hyperlink***) ad altre pagine web;

- immagini fisse o in movimento;
- suoni;
- scenari tridimensionali interattivi;
- codice eseguibile localmente.

L'utilizzo del Web è semplicissimo:
- un utente legge il testo della pagina, vede le immagini, ascolta la musica, ecc.;
- se seleziona col mouse un link (che di solito appare come una parola sottolineata e di diverso colore) la pagina di partenza viene sostituita sullo schermo da quella relativa al link selezionato.

Si noti che la nuova pagina può provenire da qualunque parte del pianeta.

7.3.1) Standard utilizzati nel Web

Ci sono tre standard principali che, nel loro insieme, costituiscono l'architettura software del Web:
- ***sistema di indirizzamento*** basato su ***Uniform Resource Locator*** (***URL***): è un meccanismo standard per fare riferimento alle entità indirizzabili (risorse) nel Web, che possono essere:
  - documenti (testo, immagini, suoni, ecc.);
  - programmi eseguibili (vedremo poi);
- ***linguaggio HTML*** (***HyperText Markup Language***): è il linguaggio per la definizione delle pagine Web;
- ***protocollo HTTP*** (***HyperText Transfer Protocol***): è il protocollo che i client e i server utilizzano per comunicare.

7.3.2) URL

Una URL costituisce un riferimento a una qualunque risorsa accessibile nel Web.

Tale risorsa ovviamente risiede da qualche parte, ed è in generale possibile accedervi in vari modi.

Dunque, una URL deve essere in grado di indicare:
- come si vuole accedere alla risorsa;
- dove è fisicamente localizzata la risorsa;
- come è identificata la risorsa.

Per queste ragioni, una URL è fatta di 3 parti, che specificano:

- il ***metodo di accesso***;
- l'***host*** che detiene la risorsa;
- l'***identità*** della risorsa.

Un tipico esempio di una URL è:

```
http://somewhere.net/products/index.html
```

nella quale:

`http://`	è il metodo di accesso
`somewhere.net`	è il nome dell'host
`/products/index.html`	è l'identità della risorsa

Metodo di accesso
Indica il modo di accedere alla risorsa, cioè che tipo di protocollo bisogna usare per colloquiare col server che controlla la risorsa.

I metodi di accesso più comuni sono:

http	protocollo nativo del Web
ftp	file transfer protocol
news	protocollo per l'accesso ai gruppi di discussione
gopher	vecchio protocollo per il reperimento di informazioni; concettualmente simile al Web, gestisce solo testo
mailto	usato per spedire posta
telnet	protocollo di terminale virtuale, per effettuare login remoti
file	accesso a documenti locali

Il Web nasce con l'idea di inglobare gli altri protocolli di accesso alle informazioni, per costituire un ambiente unificato che soddisfa tutte le esigenze.

Quando il client effettua la richiesta di una risorsa, usa nel dialogo col server il protocollo specificato dal metodo d'accesso. Se non è in grado di farlo, affida il compito a una ***applicazione helper*** esterna (questo è tipicamente il caso del protocollo telnet: il client lancia un emulatore di terminale passandogli il nome dell'host).

Dall'altra parte risponde il server di competenza, che può essere:
- un server Web in grado di gestire anche altri protocolli;
- un server preesistente per lo specifico protocollo (ftp, gopher, ecc.).

Nome dell'host
Può essere l'***indirizzo IP*** numerico o, più comunemente, il ***nome DNS*** dell'host a cui si vuole chiedere la risorsa.

Dopo il nome dell'host può essere incluso anche un ***numero di port***. Se non c'è, si intende il port 80 (che è il default). Ad esempio:

```
http://somewhere.net:8000/products/index.html
```

In questo modo si possono avere, sullo stesso host, diversi server Web in ascolto su diverse porte.

Identità della risorsa
Consiste, nella sua forma più completa, della specifica del nome di un file e del cammino che porta al direttorio in cui si trova.

Ad esempio, la URL:

```
http://somewhere.net/products/toasters/index.html
```

specifica il file `index.html` contenuto nel direttorio `toasters`, a sua volta contenuto nel direttorio `products` il quale si trova nel direttorio radice dell'host `somewhere.net`.

Si noti che:
- la sintassi è quella di Unix;
- il direttorio radice è relativo all'albero dei documenti Web, e non è necessariamente la radice dell'intero file system dell'elaboratore;
- ciò fa sì che sia di fatto impossibile accedere per mezzo del Web al di fuori di tale parte del file system: il server, di norma, non consente di accedere a nulla che non sia nell'albero dei documenti Web.

Esistono alcune regole per il completamento di URL non interamente specificate:
- se manca il nome del direttorio, si assume quello della pagina precedente;
- se manca il nome del file (ma c'è quello del direttorio), a seconda del server:
    - si restituisce un file prefissato del direttorio specificato (index.html, default.html oppure welcome.html);
    - se tale file non esiste, talvolta si restituisce un elenco dei file nel direttorio.

Infine, una convenzione usata spesso è la seguente. A fronte di una URL del tipo:

```
http://somewhere.net/~username/
```

il server restituisce il file `welcome.html` situato nel direttorio `public_html` situato nel direttorio principale (***home directory***) dell'utente `username`.

Questo meccanismo consente agli utenti, che di norma hanno libero accesso al proprio home directory, di mantenere facilmente proprie pagine Web.

7.3.3) Linguaggio HTML

Il linguaggio per la formattazione di testo HTML è una specializzazione del linguaggio ***SGML*** (***Standard Generalized Markup Language***) definito nello standard ISO 8879.

HTML è specializzato nel senso che è stato progettato appositamente per un utilizzo nell'ambito del Web.

Un ***markup language*** si chiama così perché i comandi (***tag***) per la formattazione sono inseriti in modo esplicito nel testo, a differenza di quanto avviene in un word processor ***WYSIWYG*** (***What You See Is What You Get***), nel quale il testo appare visivamente dotato dei suoi formati, come fosse stampato. TROFF e TeX sono altri markup language, mentre ad esempio Microsoft Word è WYSIWYG.

Per esempio in HTML il testo:

```
...questo è <B>grassetto</B> e questo no...
```

indica che la parola `grassetto` deve essere visualizzata in ***grassetto*** (***bold***). Quindi il testo in questione dovrà apparire come segue:

...questo è **grassetto** e questo no...

Il ruolo di HTML è quindi quello di definire il modo in cui deve essere visualizzata una pagina Web (detta anche ***pagina HTML***), che tipicamente è un documento di tipo testuale contenente opportuni tag di HTML.

Il client, quando riceve una pagina compie le seguenti operazioni:

- interpreta i tag presenti nella pagina;

- formatta la pagina di conseguenza, provvedendo automaticamente ad adattarla alle condizioni locali (risoluzione dello schermo, dimensione della finestra, profondità di colore, ecc.);
- mostra la pagina formattata sullo schermo.

Nella formattazione si ignorano:
- sequenze multiple di spazi;
- caratteri di fine riga, tabulazioni, ecc.

I tag HTML possono essere divisi in due categorie:
- tag per la formattazione di testo;
- tag per altre finalità (inclusione di immagini, interazione con l'utente, elaborazione locale, ecc.).

Il linguaggio HTML è in costante evoluzione, si è passati dalla versione 1.0 alla 2.0 (rfc 1866), poi alla 3.0 e ora alla 3.2.

E' in corso una attività di standardizzazione della versione 3, che cerca di mediare le proposte, spesso incompatibili, che sono portate avanti da diverse organizzazioni (quali Netscape e Microsoft) le quali spingono perché proprie estensioni (ad esempio i ***frame*** di Netscape e gli ***style sheet*** di Microsoft) divengano parte dello standard.

In genere i tag hanno la forma:

```
<direttiva> ... </direttiva>
```

e possono contenere parametri:

```
<direttiva parametro1="valore"...> ... </direttiva>
```

Struttura di un documento HTML
Una pagina HTML ha questa struttura:

```
<HTML>
<HEAD>
...
<TITLE>...</TITLE>
...
</HEAD >
<BODY>
...
</BODY>
</HTML>
```

Il ruolo di questi marcatori è il seguente:

HTML	Inizio e fine del documento
HEAD	Questa parte non viene mostrata e contiene metainformazioni sul documento (creatore, data di "scadenza", e se c'è, il titolo)
TITLE	Il titolo del documento: appare come titolo della finestra che lo contiene
BODY	Il suo contenuto viene visualizzato nella finestra

Tag per la formattazione
Alcuni dei tag esistenti per la formattazione del testo sono i seguenti:

`<B>...</B>`	Grassetto (***bold***)
`<I>...</I>`	Corsivo (***italic***)
`<Hx>...</Hx>`	Intestazione (***heading***) di livello x (da 1 a 6)
`<PRE>...</PRE>`	Testo visualizzato esattamente come è scritto (***preformatted***), con spazi multipli, caratteri di fine linea, ecc.

Ci sono moltissimi altri tag per la formattazione, coi quali si possono specificare:
- dimensione, colore, tipo dei caratteri;
- centratura del testo;
- liste di elementi;
- tabelle di testo in forma grafica (`<TABLE>`);
- divisori (`<HR>`,`<BR>`,`<P>`);
- colore di sfondo della pagina;
- suddivisione della finestra fra più pagine (`<FRAMESET>`, `<FRAME>`).

Tag per altre finalità
Questi sono i tag che forniscono al Web la sua grande versatilità. Anch'essi sono in continua evoluzione, permettendo di includere sempre nuove funzionalità.

I tag di questo tipo più usati sono quelli per la inclusione di ***immagini in-line*** (visualizzate direttamente all'interno della pagina) e per la gestione degli hyperlink.

Il tag per la inclusione di immagini ha la seguente forma:

`<IMG SRC="url">` oppure `<IMG SRC="url" ALT="testo...">`

Questo tag fa apparire l'immagine di cui alla URL. L'immagine (se il client è configurato per farlo) viene richiesta automaticamente e quando è disponibile viene mostrata. Altrimenti, al suo posto appare una piccola icona, sulla quale bisogna fare click se si vuole vedere la relativa immagine (che solo allora verrà richiesta), seguita dal testo specificato nel parametro `ALT`.

Altri parametri del tag `<IMG>` servono a:
- specificare le dimensioni dell'immagine (`WIDTH, HEIGHT`);
- specificare l'allineamento dell'immagine e del testo circostante (`ALIGN`);
- specificare le aree dell'immagine sensibili ai click del mouse (`ISMAP`).

Tag per la gestione degli hyperlink
Costituiscono il fondamento funzionale su cui è basato il Web, perché è per mezzo di questi che si realizzano le funzioni ipertestuali.

Il tag è uno solo (con alcune varianti) e viene chiamato ***anchor***:

```
<A> .....</A>
```

La sua forma standard è:

```
...<A HREF="url">testo visibile</A>...
```

Nella pagina la stringa `testo visibile` appare sottolineata e, di norma, di colore blu:

...testo visibile...

Quando l'utente fa click su un'ancora (ossia sul testo visibile della stessa) il client provvede a richiedere il documento di cui alla URL, lo riceve, lo formatta e lo mostra nella finestra al posto di quello precedente.

7.3.4) Il protocollo HTTP

Il protocollo HTTP sovraintende al dialogo fra un client e un server web, ed è il linguaggio nativo del Web.

HTTP non è ancora uno standard ufficiale. Infatti, HTTP 1.0 (rfc 1945) è ***informational***, mentre HTTP 1.1 (rfc 2068) è ancora in fase di proposta; parleremo di quest'ultimo più avanti.

HTTP è un ***protocollo ASCII***, cioè i messaggi scambiati fra client e server sono costituiti da sequenze di caratteri ASCII (e questo, come vedremo, è un problema se è necessaria la riservatezza delle comunicazioni).

In questo contesto per ***messaggio*** si intende la richiesta del cliente oppure la risposta del server, intesa come informazione di controllo; viceversa, i dati della URL richiesta che vengono restituiti dal server non sono necessariamente ASCII (esempi di dati binari: immagini, filmati, suoni, codice eseguibile).

Il protocollo prevede che ogni singola interazione fra client e server si svolga secondo il seguente schema:

- apertura di una connessione di livello transport fra client e server (TCP è lo standard di fatto, ma qualunque altro può essere usato);
- invio di una singola richiesta da parte del client, che specifica la URL desiderata;
- invio di una risposta da parte del server e dei dati di cui alla URL richiesta;
- chiusura della connessione di livello transport.

Dunque, il protocollo è di tipo ***stateless***, cioè non è previsto il concetto di ***sessione*** all'interno della quale ci si ricorda dello stato dell'interazione fra client e server. Ogni singola interazione è storia a se ed è del tutto indipendente dalle altre.

La richiesta del client
Quando un client effettua una richiesta invia diverse informazioni:

- il metodo (cioè il comando) che si chiede al server di eseguire;

- il numero di versione del protocollo HTTP in uso;
- l'indicazione dell'oggetto al quale applicare il comando;
- varie altre informazioni, fra cui:
    - il tipo di client;
    - i tipi di dati che il client può accettare.

I metodi definiti in HTTP sono:

GET	Richiesta di ricevere un oggetto dal server
HEAD	Richiesta di ricevere la sola parte head di una pagina html
PUT	Richiesta di mandare un oggetto al server
POST	Richiesta di appendere sul server un oggetto a un altro (vedremo che si usa molto)
DELETE	Richiesta di cancellare sul server un oggetto
LINK e UNLINK	Richieste di stabilire o eliminare collegamenti fra oggetti del server

In proposito, si noti che:
- il metodo che si usa quasi sempre è GET;
- POST ha il suo più significativo utilizzo in relazione all'invio di dati tramite form, come vedremo in seguito;
- HEAD si usa quando il client vuole avere delle informazioni per decidere se richiedere o no la pagina;
- PUT, DELETE, LINK, UNLINK non sono di norma disponibili per un client, tranne che in quei casi in cui l'utente sia abilitato alla configurazione remota (via Web) del server Web.

Ad esempio, supponiamo che nel file HTML visualizzato sul client vi sia un'ancora:

```
<A HREF="http://somewhere.net/products/toasters/index.html"> ..... </A>
```

e che l'utente attivi tale link. A tal punto il client:
- chiede al DNS l'indirizzo IP di somewhere.net;
- apre una connessione TCP con somewhere.net, port 80;
- invia la sua richiesta.

Essa è costituita da un insieme di comandi (uno per ogni linea di testo) terminati con una linea vuota:

`GET /products/toasters/index.html HTTP/1.0`	Metodo, URL e versione protocollo
`User-agent: Mozilla/3.0`	Tipo del client
`Host: 160.10.5.43`	Indirizzo IP del client
`Accept: text/html`	Client accetta pagine HTML
`Accept: image/gif`	Client accetta immagini
`Accept: application/octet-stream`	Client accetta file binari qualunque
`If-modified-since: data e ora`	Inviare il documento solo se è più recente della data specificata

La risposta del server
La risposta del server è articolata in più parti, perché c'è un problema di fondo: come farà il client a sapere in che modo dovrà gestire le informazioni che gli arriveranno?

Ovviamente, non si può mostrare sotto forma di testo un'immagine o un file sonoro! Dunque, si deve informare il client sulla natura dei dati che gli arriveranno prima di iniziare a spedirglieli.

Per questo motivo la risposta consiste di 3 parti:
- una ***riga di stato***, che indica quale esito ha avuto la richiesta (tutto ok, errore, ecc.);
- delle ***metainformazioni*** che descrivono la natura delle informazioni che seguono;
- le ***informazioni*** vere e proprie (ossia, l'oggetto richiesto).

La riga di stato, a sua volta, consiste di tre parti:
- Versione del protocollo http;
- Codice numerico di stato;
- Specifica testuale dello stato.

Tipici codici di stato sono:

Esito	Codice numerico	Specifica testuale
Tutto ok	`200`	`OK`
Documento spostato	`301`	`Moved permanently`
Richiesta di autenticazione	`401`	`Unauthorized`
Richiesta di pagamento	`402`	`Payment required`
Accesso vietato	`403`	`Forbidden`
Documento non esistente	`404`	`Not found`
Errore nel server	`500`	`Server error`

Dunque, ad esempio, si potrà avere

```
HTTP/1.0 200 OK
```

Le metainformazioni dicono al client ciò che deve sapere per poter gestire correttamente i dati che riceverà.

Sono elencate in linee di testo successive alla riga di stato e terminano con una ***linea vuota***.

Tipiche metainformazioni sono:

`Server: ...`	Identifica il tipo di server
`Date: ...`	Data e ora della risposta
`Content-type: ...`	Tipo dell'oggetto inviato
`Content-length: ...`	Numero di byte dell'oggetto inviato
`Content-language: ...`	Linguaggio delle informazioni
`Last-modified: ...`	Data e ora di ultima modifica
`Content-encoding: ...`	Tipo di decodifica per ottenere il content

Il Content-type si specifica usando lo standard ***MIME*** (***Multipurpose Internet Mail Exchange***), nato originariamente per estendere la funzionalità della posta elettronica.

Un tipo MIME è specificato da una coppia

```
MIME type/MIME subtype
```

Vari tipi MIME sono definiti, e molti altri continuano ad aggiungersi. I più comuni sono:

Type/Subtype	**Estensione**	**Tipologia delle informazioni**
`text/plain`	`.txt, .java`	testo
`text/html`	`.html, .htm`	pagine html
`image/gif`	`.gif`	immagini gif
`image/jpeg`	`.jpeg, .jpg`	immagini jpeg
`audio/basic`	`.au`	suoni
`video/mpeg`	`.mpeg`	filmati
`application/octet-stream`	`.class, .cla, .exe`	programmi eseguibili
`application/postscript`	`.ps`	documenti Postscript
`x-world/x-vrml`	`.vrml, .wrl`	scenari 3D

Il server viene configurato associando alle varie estensioni i corrispondenti tipi MIME. Quando gli viene chiesto un file, deduce dall'estensione e dalla propria configurazione il tipo MIME che deve comunicare al client.

Se la corrispondenza non è nota, si usa quella di default (tipicamente text/html), il che può causare errori in fase di visualizzazione.

Anche la configurazione del client (in merito alle applicazioni helper) si fa sulla base dei tipi MIME.

Tornando al nostro esempio, una richiesta del client quale:

```
GET /products/toasters/index.html HTTP/1.0
User-agent: Mozilla/3.0
ecc.
```

riceverà come risposta dal server (supponendo che non ci siano errori) le metainformazioni, poi una riga vuota e quindi il contenuto del documento (in questo caso una pagina HTML costituita di 6528 byte):

```
HTTP/1.0 200 OK
Server: NCSA/1.4
Date: Tue, july 4, 1996 19:17:05 GMT
Content-type: text/html
Content-length: 6528
Content-language: en
Last-modified: Mon, july 3, 1996 15:05:35 GMT
                                                <----- notare la riga vuota
<HTML>
<HEAD>
...
<TITLE>...</TITLE>
...
</HEAD >
<BODY>
...
</BODY>
</HTML>
```

Sulla base di quanto detto finora, si possono fare alcune osservazioni:
- il protocollo HTTP è molto semplice, essendo basato su interazioni che prevedono esclusivamente l'invio di una singola richiesta e la ricezione della relativa risposta;
- questa semplicità è insieme un punto di forza e di debolezza:
    - di forza perché è facilissimo, attraverso la definizione di nuovi tipi MIME e di corrispondenti funzioni sui client, estendere le tipologie di informazioni gestibili (il server non entra nel merito di ciò che contengono i file: si limita a consegnare i dati che gli vengono richiesti, senza preoccuparsi della loro semantica);
    - di debolezza perché queste estensioni di funzionalità talvolta mal si adattano alla concezione originaria (stateless) del protocollo, come ad esempio è il caso delle transazioni commerciali.

7.3.5) I problemi del Web

Il protocollo HTTP è nato con una finalità (lo scambio di informazioni fra gruppi di ricercatori) che si è radicalmente modificata col passare del tempo.

Infatti, attualmente:
- il suo utilizzo si è ampliato enormemente;
- si cerca di sfruttarlo anche per transazioni di tipo commerciale.

Questi due aspetti hanno messo alla frusta il protocollo soprattutto da tre punti di vista:
- efficienza nell'uso delle risorse di rete;
- assenza del concetto di sessione;
- carenza di meccanismi per lo scambio di dati riservati.

Efficienza nell'uso della rete
Il problema di fondo è che il protocollo HTTP prevede che si apra e chiuda una connessione TCP per ogni singola coppia richiesta-risposta.

Si deve tenere presente che:
- l'apertura e la chiusura delle connessioni TCP non è banale (***three way handshake*** per l'apertura, idem più timeout per la chiusura);
- durante l'apertura e la chiusura non c'è controllo di flusso;
- la dimensione tipica dei dati trasmessi è piccola (pochi KByte);
- TCP usa la tecnica dello ***slow start*** per il controllo di flusso.

Di conseguenza, si ha che:

- il protocollo HTTP impone un grave overhead sulla rete (sono frequentissime le fasi di apertura/chiusura di connessioni TCP);
- gli utenti hanno la sensazione che le prestazioni siano scarse (per via dello slow start).

Assenza del concetto di sessione
In moltissime situazioni (ad esempio nelle transazioni di tipo commerciale) è necessario ricordare lo stato dell'interazione fra client e server, che si svolge per fasi successive.

Questo in HTTP è impossibile, per cui si devono trovare soluzioni ad hoc (ad esempio, restituire pagine HTML dinamiche contenenti un campo HIDDEN che identifica la transazione).
In proposito c'è una proposta (***State Management***, rfc 2109) basata sull'uso dei ***cookies***, cioè particolari file che vengono depositati presso il client e che servono a gestire una sorta di sessione.

Carenza di meccanismi per lo scambio di dati riservati
L'unico meccanismo esistente per il controllo degli accessi (***Basic Authentication***) fa viaggiare la password dell'utente in chiaro sulla rete, come vedremo fra breve.

HTTP versione 1.1
E' in corso di completamento la proposta, che diventerà uno standard ufficiale di Internet, della versione 1.1 di HTTP.

Essa propone soluzioni per tutti gli aspetti problematici che abbiamo citato ed introduce ulteriori novità. I suoi aspetti salienti sono:

- il supporto per connessioni persistenti;
- il supporto per l'autenticazione;
- l'introduzione dei metodi estensibili.