1 Il Livello Applicazione

Il livello Applicazione si colloca sulla cima del modello ISO/OSI. Tale scelta é dovuta alla natura del livello stesso, ovvero di descrivere i protocolli ad alto livello che dialogano direttamente con i programmi software.

Per permettere lo scambio di messaggi tra due programmi che si trovano sulla stessa rete, é stata creata una semplice ma efficace interfaccia di programmazione al quale tutti i programmi fanno riferimento: i **socket** *(dalla terminologia Unix).*

I socket sono un'interfaccia software messa a disposizione dal sistema operativo che permette di utilizzare in modo trasparente tutta l'architettura protocollare di rete. É analogo ad un punto di accesso/uscita SAP nel modello ISO/OSI.

Ad esempio: se un programma sulla macchina *A* vorrá spedire un messaggio al programma destinazione sulla macchina *B*, non dovrá fare altro che chiamare una utility di sistema che, in gergo, aprirá una socket verso la destinazione e rimarrá in attesa che A "scriva" un messaggio in essa, quindi si occuperá lei della consegna e gestione delle eventuali ritrasmissioni, riconnessioni, frammentazione dei pacchetti ecc.

Come si nota dall'esempio precedente, il software ricopre il ruolo di utilizzatore della rete, delegando ai socket tutta la complessitá del trasferimento; in altri termini presuppone l'esistenza di un'infrastruttura esterna che trasporterá il messaggio. In questo modo, nel caso di un aggiornamento del software di rete, non sará necessario riscrivere ogni applicazione, ma basterá continuare a garantirle sempre la stessa interfaccia.

Una volta che il messaggio é arrivato a destinazione, é necessario un meccanismo che permetta al calcolatore di capire a quale processo inoltrare il messaggio. Ad ogni processo che vuole utilizzare la rete, viene quindi assegnato un numero a 16 bit chiamato **port** dal sistema operativo.

Funny Story In italiano sarebbe piú corretto dire numero di *porto* invece che di *porta*, in quando é la sua traduzione, ma nel gergo comune non é purtroppo utilizzato.

Esistono peró delle restrizioni: le porte da 0-1023 vengono chiamate **Well-known ports** *(porte ben conosciute)* che sono definite per protocolli standard e quindi sono protette dal sistema operativo attraverso particolari autorizzazioni. Le porte invece da 1024-65535 sono invece lasciate per tutti gli altri applicativi che ne faranno richiesta.

1.1 Riassumendo

Il livello applicazione permette a processi diversi di utilizzare la rete sottostante nascondendone tutti i dettagli implementativi; inoltre, permette di identificare ogni processo che utilizza la rete tramite un numero di porta.

2 Architetture

Attraverso il livello applicazione, descritto in precedenza, é possibile categorizzare le applicazioni nel modo in cui comunicano tra di loro sulla rete.

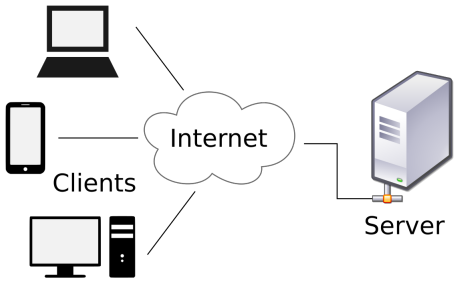
É bene notare che si tratta solo di modelli di comunicazione, il vero protocollo é descritto tramite il software che lo implementa, che spesso é proprietario, e quindi non noto al pubblico. Il vantaggio di questo é la semplicitá di sviluppo a livello applicazione di funzioni che dovrebbero essere svolte dai livelli sottostanti nella pila ISO/OSI, ad esempio: é piú facile creare il proprio protocollo su HTTP in quanto non bloccato dalla maggioranza dei firewall presenti in internet, invece di crearlo parallelamente.

I protocolli pubblici, invece, sono descritti dagli appositi **RFC** *(Request For Comments)*.

2.1 Client-Server

Il primo modo di scambiare messaggi attraverso la rete é quello di inviarli ad un host centrale, il quale si occuperá di elaborarlo e di rispondere adeguatamente. In questo modello si identifica come **Client** *(cliente)*, il processo software che invia i messaggi e **Server** *(servente)* come il processo software che rimane in ascolto delle richieste del client per l'elaborazione.

Il server deve essere individuabile dai client in modo univoco attraverso uno o piú indirizzi IP statici (che non cambiano nel tempo), solitamente ricavabili attraverso il sistema DNS. Il client invece non ha bisogno di un indirizzo IP pubblico, perché il server gestirá l'inoltro della risposta attraverso l'indirizzo sorgente preso della richiesta.



Ovviamente due client non comunicano mai tra di loro ma, ad esempio, il server potrá agire da tramite tra di loro.

In questo caso la complessitá del software risiede tutta nel server, che deve rimanere in attesa delle richieste di tutti i client, elaborale se ben formate, ed inviare la risposta al client. Per ogni richiesta, il server deve spendere del tempo macchina e delle risorse per elaborare la risposta, quindi in quel momento non potrá accettare altre richieste. Per aumentare il numero di client servibili nello stesso istante l'architettura server é solitamente un centro di calcolo molto potente con il software replicato su piú macchine fisiche.

Invece il client é un software nettamente piú semplice in quanto deve solo comunicare al server che vuole iniziare una comunicazione, confezionare ed inviare le richieste e rimanere in attesa della risposta.

Un ulteriore modo di vedere questa architettura é paragonarlo al modello classico del commercio: un cliente *(client)* che richiede un servizio al commerciante *(server)*, e non il contrario.

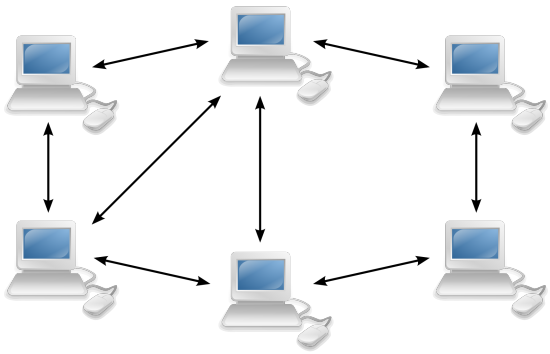
2.2 Peer-to-Peer o P2P

L'architettura **Peer-to-Peer**, o *paritaria* o *paritetica*, riprende il modello client-server ma sposta le resposabilitá del server su ogni client. Ogni host collegato nella rete, chiamato **peer** *(pari)*, svolge la funzione sia da client che da server.

In questo modello le risorse possono essere **distribuite** su piú macchine e non esiste la necessitá che almeno una parte abbia un indirizzo pubblico, in quanto nel momento della connessione tra due o piú host vengono scambiati i rispettivi indirizzi IP.

Il vero problema riguarda la ricerca e l'indicizzazione delle risorse disponibili: un peer deve avere una visuale completa delle risorse presenti negli altri peer per poterli richiedere. A questo scopo si utilizzano strutture dati specifiche, come **DHT** *(Distributed Hash Table)*.

Il vantaggio nell'utilizzo di questa architettura é la scalabilitá che si ottiene quando il numero di peer cresce condividono le risorse: non esiste un unico *"centro" (server)* che contiene tutte le risorse, che nel caso di guasti, non sarebbero piú disponibili ai clienti, ma ogni peer puó avere la sua copia della stessa risorsa e condividerla con il resto della rete.



2.3 Ibridi

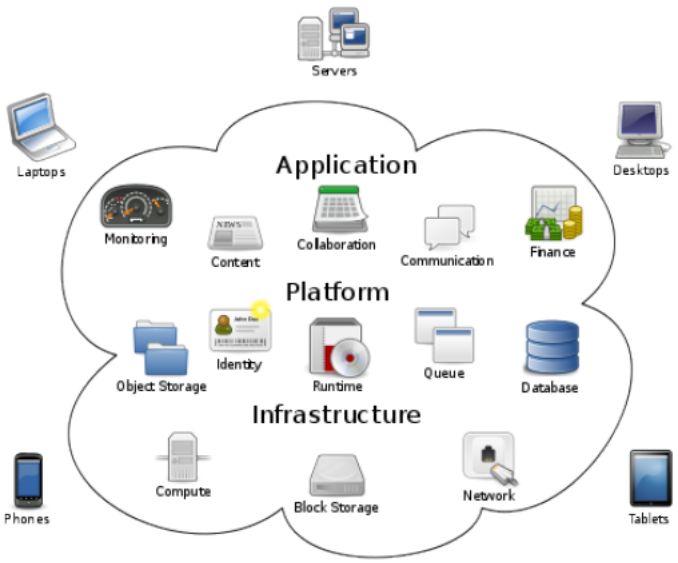
L'architettura **Ibrida** utilizza sia tecniche *client-server* che *peer-to-peer*, nel senso che un nuovo host che si collega ad una rete peer-to-peer puó interrogare un server che ne contiene la locazione, in termini di indirizzi IP degli altri peer. In questo modo andrá a richiedere la risorsa direttamente all'indirizzo suggerito invece di cercarlo in tutta la rete.

Un esempio dell'architettura ibrida é stato **Skype**, un software di *Voice ove IP*: principalmente usava un sistema client-server per l'autenticazione e un'architettura peer-to-peer per permettere la comunicazione vocale tra gli utenti.

2.4 Cloud Computing

L'architettura **Cloud** si puó definire come l'architettura server portata all'estremo, nel senso che non esiste un unico host con installato il software lato server che risponde alle richieste dei client, ma esiste un'infrastruttura composta da molti calcolatori ad alta capacitá di elaborazione che viene vista come un unica macchina ad alto livello. Questo permette di installare il programma server con una certa quantitá di risorse (memoria, processori, dischi ecc) e, in caso di carico eccessivo, aumentarle a richiesta.

É possibile inoltre, se l'infrastruttura cloud lo permette, spostare l'intero software server nelle sedi vicine ai client che richiedono i servizi. Questo é possibile grazie a reti specializzate della consegna di contenuti, chiamate **CDN** *(Content Delivery Network)*. Basti a pensare a **Youtube**, ogni video che viene richiesto da un client, viene spostato, attraverso una CDN, nella sede Google piú vicina al client per ridurre le latenze e i tempi di caricamento.



Un'ulteriore risorsa che l'infrastruttura cloud puó offrire é la mobilitá d'utente: due utenti possono autenticarsi sullo stesso computer collegato al cloud e avere tutte le loro impostazioni e documenti disponibili, in quanto per costruzione dell'architettura tutte le informazioni verranno trasferite su ogni dispositivo a cui gli utenti accedono.

3 DNS

3.1 La Storia

Ai tempi di ARPANET esisteva semplicemente un file, *host.txt*, che manteneva tutte le associazioni tra i nomi degli host e i loro indirizzi IP. Ogni notte tutti i calcolatori lo prelevavano dal sito da cui era gestito. Per una rete composta da poche centinaia di grandi computer questo approccio era abbastanza efficace.

Tuttavia quando alla rete furono connessi milioni di PC, fu chiaro a tutti che questo metodo non poteva funzionare per sempre; da una parte la dimensione del file sarebbe diventata troppo elevata, mentre dall'altra cera il rischio di continui conflitti tra i nomi degli host.

Per risolvere questi problemi nel 1983 fu realizzato **DNS** *(Domain Name System)*. L'essenza del DNS é l'introduzione di uno schema di denominazione gerarchico bastato su domini e di un database distribuito.

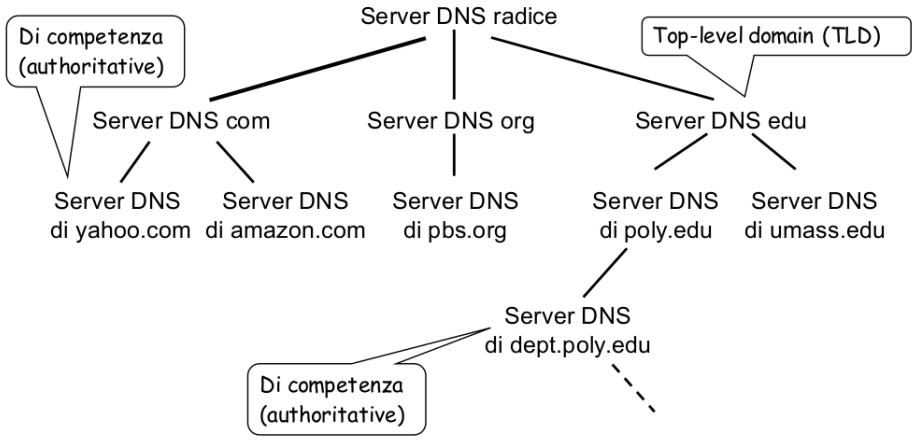
3.2 Funzionamento in Breve

Per associare un nome ad un indirizzo IP un programma applicativo invoca una procedura di libreria chiamata **resolver** *(risolutore)*, passando il nome come parametro. Il resolver invia un pacchetto UDP contenente la richiesta a un server DNS locale, che cerca il nome e risponde con l'indirizzo IP *(processo di risoluzione)*. Equipaggiato con l'indirizzo IP del destinatario cercato, il programma puó quindi stabilire una connessione TCP.

3.2.1 Lo Spazio dei Nomi

In Internet, in cima alla gerarchia di assegnazione dei nome, c'é un'organizzazione chiamata **ICANN** *(Internet Corporation for Assigned Names and Numbers)*, creato nel 1998. Concettualmente Internet é divisa in oltre 250 **domini di primo livello** o **TLD** *(Top Level Domain)*. Ogni dominio é partizionato in **sottodomini**, che sono a loro volta divisi e cosí via.

Tutti questi domini possono essere rappresentati con una struttura ad albero, dove le foglie rappresentano i domini che non hanno sottodomini, ma contengono computer.



Ogni dominio é identificato dal percorso compreso tra esso e la radice. Le componenti sono separati da **punti** *(dot)*. Occorre notare che questa denominazione gerarchica garantisce che, ad esempio, *eng.cisco.com* non sia in conflitto con un potenziale utilizzo di *eng* in *eng.washinton.edu*, che potrebbe essere utilizzato da altri.

I componenti del nome di dominio possono essere lunghi fino a 63 caratteri, mentre il percorso completo non deve superare i 255 caratteri.

Non esistono regole per la registrazione in due domini di primo livello della stessa societá (*sony.com, sony.net ..*).

La denominazione segue i confini dell'organizzazione, non le reti fisiche.

3.2.2 Record delle Risorse di Dominio

A ogni dominio, che sia rappresentato da un singolo host o sia un dominio di primo livello, puó essere associato un insieme di **resource record** *(record delle risorse)* che formano il database DNS.

Quando un resolver sottopone un nome di dominio a un server DNS, ció che ottiene sono i record delle risorse associate a tale nome. Di conseguenza la funzione principale del DNS e associare i nomi di dominio ai record delle risorse. Un record delle risorse é quindi un quintupla:

```
Domain-name Time-to-live Class Type Value
```

- *Domain-name* indica il dominio a cui si riferisce il record
- *Time-to-live* offre un indicatore sulla stabilitá (o durata) del record
- *Class* per le informazioni riguardanti Internet é sempre *IN*
- *Type* specifica il tipo di record tra i molti DNS

Il tipo di record piú importante é il record *A*. Contiene l'indirizzo IPv4 a 32 bit di un'interfaccia di un host. Alcuni host hanno due o piú interfacce di rete e in questo caso possiedono un record delle risorse di tipo A o *AAAA* per ciascuna. Di conseguenza il DNS puó fornire piú indirizzi per un singolo nome.

Un altro record importante é il record *NS* che specifica il name server per dominio o sottodominio. É usato come parte del processo di ricerca dei nomi.

I record *CNAME* consentono la creazione di *alias* ad esempio: se si vuole creare l'alias *cs.mit.edu* per il dominio *csail.mit.edu* il record da creare sará:

```
cs.mit.edu 86400 IN CNAME csail.mit.edu
```

- *Value* che puó essere un numero, un nome di dominio o una stringa ASCII.

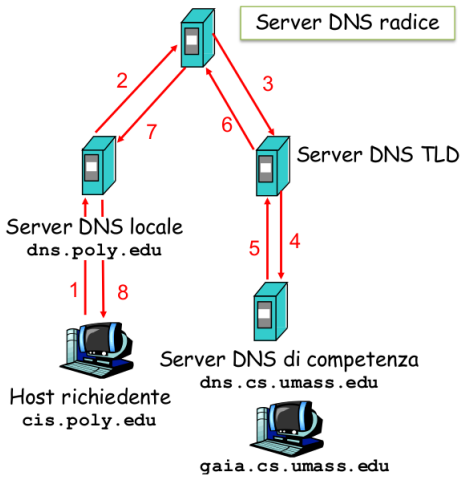
3.2.3 DNS Server

In teoria un singolo name server potrebbe contenere l'intero database DNS e rispondere a tutte le interrogazioni. In pratica, questo server sarebbe troppo sovraccarico per essere utile.

Per evitare problemi associati alla disponibiltá di un a singola fonte di informazioni, lo spazio dei nomi DNS viene diviso in **zone** non sovrapposte. Ogni zona é associata a uno o piú name server. Questi sono host che tengono il database per la zona. Di norma, una zona avrá un name server primario, che contiene le sue informazioni su un file su disco e uno o piú secondari, che ottengono le loro informazioni dal name server primario.

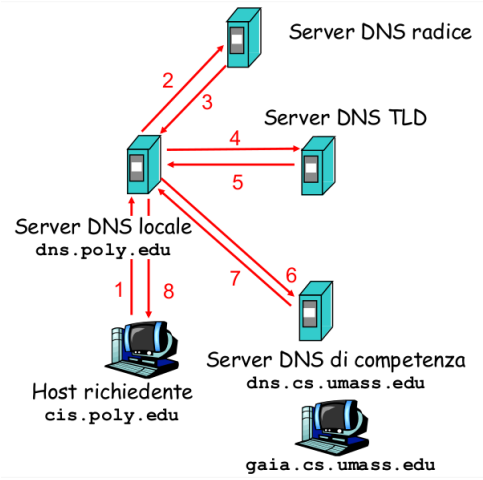
Il processo di ricerca di un nome e di un indirizzo é chiamto **risoluzione del nome**. Un **resolver** passa l'interrogazione a uno dei name server locali. Se il dominio é all'interno della giurisdizione del name server, restituisce i record autoritativi delle risorse. Un **record autoritativo** é un record fornito direttamente dall'autoritá che lo gestisce ed é pertanto sempre corretto.

Interrogazione Ricorsiva l'interrogazione ricorsiva prevede che la risoluzione di ogni parte del nome venga gestita direttamente dai name server. Come si vede dalla figura, l'host richiede la risoluzione del nome *cis.poly.edu* al suo name server locale, il quale non conoscendolo direttamente, inoltra la richiesta di al DNS radice perché viene richiesto un *.edu*, il quale la inoltra al TLD server fino a raggiungere il name server che contiene il record desiderato. A questo punto la risposta completa tornerá all'host che aveva richiesto la risoluzione.



La logica del name server locale nel supportare un'interrogazione ricorsiva é che sta fornendo un servizio agli host nel suo dominio che non devono essere configurati per svolgere essi stessi le funzioni di un name server, ma devono solo raggiungere quello locale. Basti a pensare ad un router casalingo che non svolge funzioni di name server, ma si appoggia sul name server locale.

Interrogazione Iterativa l'interrogazione iterativa si distingue da quella ricorsiva nel nel ruolo del name server durante la risoluzione dell'intero nome. Come si vede nella figura, l'host richiede la risoluzione del nome *cis.poly.edu* al name server locale il quale si preoccupa totalmente della risoluzione: chiede prima al DNS radice la risoluzione di *.edu*, poi al TLD ed infine raggiunge il name server che contiene il dominio richiesto, che passerá all'host.



Il name server locale non restituisce una risposta parziale, perché pur essendo utile, non é quello che é stato richiesto.

In questo caso ricoprono un ruolo molto importante i **cached record** *(record archiviati nella cache)*, memorizzati per una certa quantitá di tempo per permettere la risoluzione di richieste multiple dello stesso dominio in modo piú efficiente.

Tuttavia, le risposte archiviate nella cache non sono autoritative, perché i cambiamenti fatti su un particolare dominio, non si propagheranno alle altre cache nel mondo. Per questa ragione le informazioni non dovrebbero vivere troppo a lungo nella cache; questo é il motivo per cui in ogni record delle risorse viene incluso un campo *Time-to-live*.

Un ulteriore problema di DNS é il protocollo di trasporto utilizzato per le interrogazioni e le risposte: **UDP**. Non essendo ne connesso ne confermato, se non arriva alcuna risposta entro un limitato periodo di tempo, il client DNS ripeterá l'interrogazione, provando con un altro server del dominio dopo un piccolo numero di tentativi.

4 Web e HTTP

4.1 La Storia

Il web nacque nel 1989 presso il CERN, il centro europeo per la ricerca nucleare. L'idea iniziale era di aiutare grandi gruppi di scienziati provenienti da molte nazioni anche con diversi fusi orari a collaborare utilizzando una raccolta in costante cambiamento di resoconti, documenti, fotografie. La proposta iniziale per una rete di documenti collegati fu opera del fisico Tim Berners-Lee. Il primo prototipo fu operativo 18 mesi dopo. Ne fu eseguita una presentazione pubblica nel '91 alla conferenza Hypertext a San Antonio, Texas. Questa dimostrazione attrasse l'attenzione di altri ricercatori a sviluppare il primo browser grafico, chiamato Mosaic e, successivamente Netscape.

Nel 1994 CERN e MIT firmarono un accordo per creare il **W3C** *(World Wide Web Consortium)*: un'organizzazione che ha come scopo l'ulteriore sviluppo del Web, lo standard dei protocolli e lo sviluppo dell'interoperabilitá tra i siti.

Dal 1990 al 2010 i siti Web, o pagine Web sono cresciuti esponenzialmente fino a raggiungere milioni di siti e miliardi di pagine: **l'era del dot com**.

4.2 Panoramica dell'architettura

Dal punto di vista degli utenti il Web é una vasta raccolta mondiale di documenti o **pagine Web**, spesso definire per brevitá **pagine**. Ogni pagina puó contenere collegamenti *(link)* ad altre pagine situate ovunque nel mondo.L'idea di fare in modo che ogni pagina *"puntasse"* a un'altra viene chiamato **ipertesto** *(hypertext)*.

Un pezzo di testo, icona, immagine o altro, collegati ad un'altra pagina, é chiamato **collegamento ipertestuale** *(hyperlink)*. Seguire il collegamento é un modo per indicare al browser di visualizzare un'altra pagina.

Le pagine sono visualizzate con un programma chiamato **browser** che, preleva la pagina richiesta, interpreta il testo e i comandi di formattazione, quindi visualizza la pagina correttamente sullo schermo. Il contenuto puó essere un misto di testo, immagini oppure video o programmi che producono un'interfaccia grafica con cui l'utente puó interagire. Ogni pagina, quindi, viene scaricata sulla macchina client inviando una o piú richieste a uno o piú server. Le domande e le risposte sono basate su un semplice protocollo testuale che funziona su TCP: viene chiamato **HTTP** *(Hyper Text Transfer Protocol)*.

Se il documento é sempre uguale siamo in presenza di una **pagina statica**, mentre se é generato su richiesta da un programma o contiene del codice eseguibile é chiamato **pagina dinamica**.

Ogni pagina é descritta da un indirizzo univoco chiamato **URL** *(Universal Resource Locator)* della forma:

```
<protocollo>://<DNS/IP>/<percorso-della-risorsa>
```

ad esempio:

```
http://www.someschool.edu/someDept/pic.jpg
```

4.2.1 Il lato client

In pratica, un browser é un programma che puó visualizzare una pagina Web e rilevare i click del mouse sugli elementi in esse visualizzati. Quando un utente fa click su un collegamento ipertestuale il browser esegue una serie di passaggi per recuperare la pagina indicata:

1. determina l'URL dal click dell'utente
2. richiede al DNS l'indirizzo IP e rime in attesa della risposta
3. il DNS risponde con l'indirizzo IP
4. crea una connessione TCP verso l'indirizzo IP recuperato sulla porta 80
5. invia una richiesta HTTP per la risorsa richiesta
6. il server risponde con la risorsa richiesta
7. se la pagina include delle URL necessarie alla visualizzazione, il browser esegue lo stesso procedimento per ognuna di queste
8. una volta terminato l'intero scaricamento della risorsa, la visualizza
9. la connessione TCP viene rilasciata se non vi sono altre richieste agli stessi server per un breve periodo

4.2.2 Il lato server

Ad un server Web viene fornito il nome di un file da cercare e restituire. I passaggi eseguiti dal server nel ciclo principale sono i seguenti:

1. accettare una connessione TCP proveniente da un client
2. recuperare dalla richiesta HTTP il percorso della pagina
3. ottenere il file dal disco
4. inviare il contenuto del file al client
5. rilasciare la connessione TCP

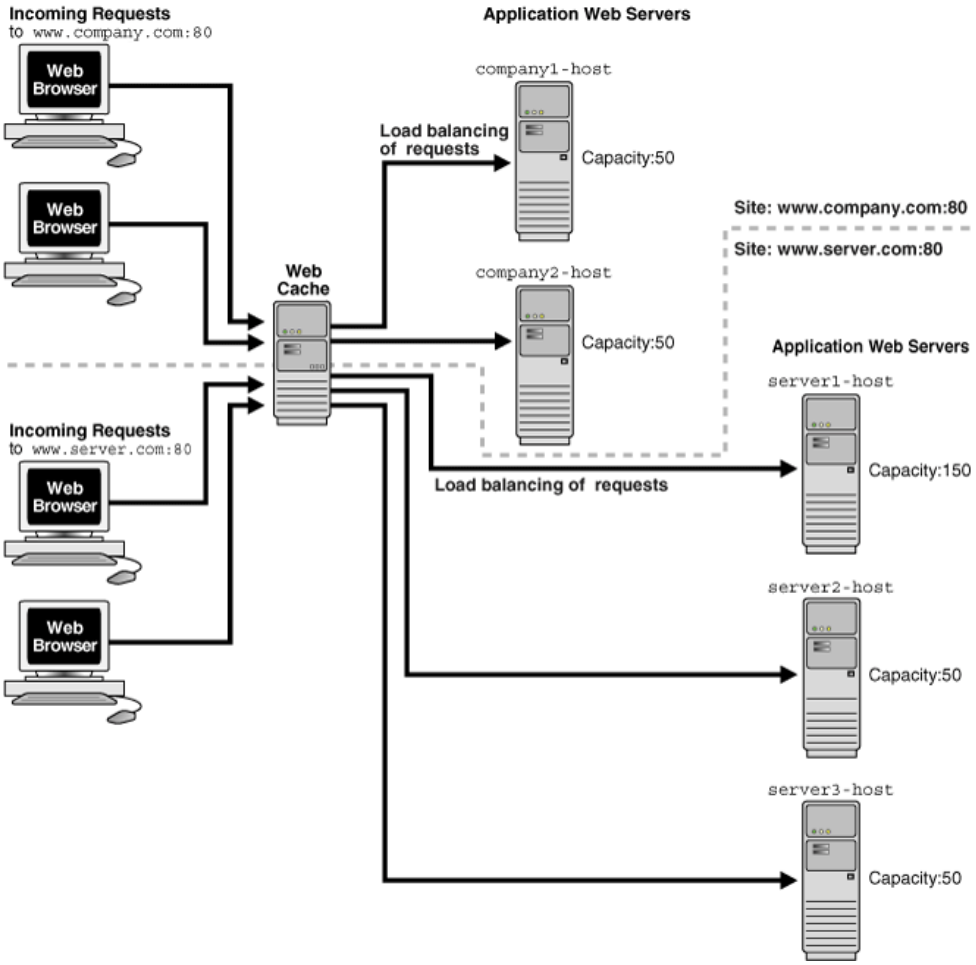
Per contenuti dinamici, il terzo passaggio viene sostituito dall'esecuzione di un programma (determinato dal percorso) che restituisce i contenuti.

Tuttavia, i Web server possono soddisfare molte richieste al secondo. Il problema é che l'accesso al disco é spesso il collo di bottiglia: gli stessi file possono essere letti ripetutamente dal disco usando chiamate al sistema operativo. Ulteriormente, nel momento in cui il server legge un file dal disco, potenzialmente di grandi dimensioni, non puó accettare nessun'altra connessione.

Un miglioramento ovvio consiste nel mantenere in cache gli n file utilizzati piú di recente o un certo numero di gigabyte di contenuti. Prima di accedere al disco per recuperare il

file, il server controlla la cache. Se il file é presente, puó essere inviato direttamente dalla memoria.

Per ovviare al problema di servire una singola richiesta alla volta, una strategia é implementare il server con un approccio **multithread**: un thread, detto di **front-end** rimane in ascolto delle richieste dei client ed esegue operazioni di **load-balancing**, ovvero ridirige la richiesta su un'altra macchina, la quale gestirá in pieno la richiesta.



Ovviamente questo schema é di piú complessa realizzazione, ma necessario quando cresce il numero di utenti. La macchina incaricata deve quindi:

1. risolve il nome della pagina Web richiesta
2. eseguire il controllo di accesso alla pagina
3. controllare la cache
4. prelevare la pagina richiesta dal disco o eseguire un programma che la costruisca
5. determinare il resto della risposta
6. restituire la risposta al client
7. aggiungere un elemento al registro delle operazioni svolte dal server

4.3 HTTP

Il protocollo usato per trasportare informazioni tra server Web e client é **HTTP** *(Hyper Text Transfer Protocol)*. HTTP é un protocollo basato sul modello di domanda e risposta *(request-response)*, normalmente implementato su TCP. Specifica quali messaggi i client possono inviare ai server e quali risposte vengono restituite. Le intestazioni delle domande e delle risposte sono formulate in ASCII.

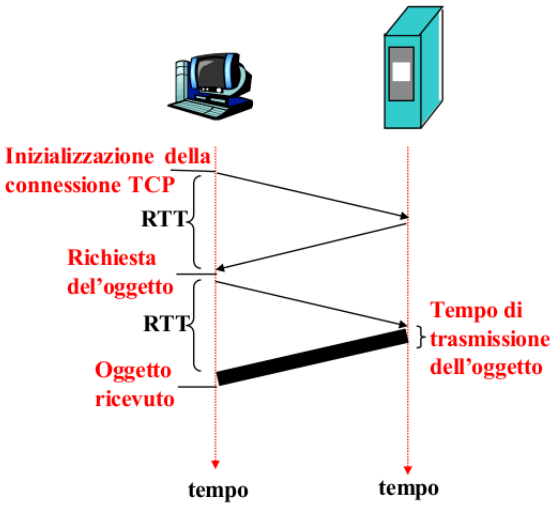
HTTP sta diventando sempre piú un protocollo di trasporto, ovvero che rende possibile ai processi lo scambio di contenuti attraverso reti differenti. Questi processi potrebbero anche non essere un browser Web o un server Web; ad esempio gli sviluppatori possono impiegare HTTP per il recupero di file di progetto.

4.3.1 Connessioni

Il modo solitamente utilizzato dal browser per contattare un server prevede di stabilire una connessione TCP sulla porta 80 al suo indirizzo. Il vantaggio di TCP sta nel fatto che né i browser ne i server devono preoccuparsi dei messaggi lunghi, affidabilitá o controllo di congestione, in quanto sono gestiti dall'implementazione di TCP.

Con HTTP 1.0 le pagine erano generalmente composte da un unico file html quindi, dopo aver stabilito la connessione, veniva inviata una singola richiesta e restituita una singola risposta; a questo punto la connessione TCP veniva rilasciata. Nel giro di pochi anni la pagina Web media inizió a contenere sempre piú collegamenti a contenuti, quali icone e altri elementi per migliorarne l'aspetto, ma stabilire una connessione TCP per trasportare una singola icona diventava un modo molto costoso di operare.

Questa osservazione ha portato ad HTTP 1.1, che supporta **connessioni persistenti**. Con questa tecnologia é possibile stabilire una connessione TCP, inviare una richiesta, ottenere una risposta e poi inviare altre richieste e ottenere nuove risposte. Questa strategia é detta **riutilizzo della connessione**.[1]



[1]RTT *(Round Trip Time)*: tempo impiegato dall'unitá dati per arrivare alla destinazione e ritorno

4.3.2 Messaggi e Metodi

Anche se HTTP fu progettato per l'utilizzo sul Web é stato reso intenzionalmente piú generico del necessario per permettere alle applicazioni future di svilupparsi. Per questo motivo sono supportate varie operazioni chiamate **metodi**.

Ogni richiesta consiste di una o piú righe di testo ASCII, con la prima parola della prima riga che rappresenta il nome del metodo richiesto. Esistono diversi metodi, i piú comuni sono:

- `GET` richiede al server l'invio di un file attraverso, anche dei parametri specificati nell'URL dopo il carattere di escape `?`. Ad esempio:
  `http://www.someurl.edu/res.txt?a=1&b=2`
  richiede al server il file `res.txt` con i parametri `a=1` e `b=2`.

- `POST` stessa funzione di `GET` con la differenza che accoda dei possibili parametri ai dati del messaggio invece che nella URL.

- `PUT` é il contrario di `GET`: invece di leggere la pagina, la scrive.

- `HEAD` richiede solamente l'intestazione del messaggio, senza la pagina effettiva. Questo metodo puó essere utilizzato per ottenere la data dell'ultima modifica di una pagina, per raccogliere informazioni a scopo di indicizzare o per verificare la validitá di un URL.

- `DELETE` rimuove la pagina o almeno indica che il server web é d'accordo a rimuovere la pagina. Autenticazione e autorizzazione giocano un ruolo fondamentale.

Ogni richiesta ottiene una risposta costituita da una riga di stato e magari da altre informazioni. La riga di stato contiene un codice di tre cifre che indica se la richiesta é stata soddisfatta e, in caso contrario, la causa del fallimento.

Le famiglie di codici si dividono in:

- `1xx` Informazione
- `2xx` Successo
  - `200 OK` richiesta andata a buon fine
  - `204 No Content` nessun contenuto trovato
- `3xx` Redirezione
  - `301 Moved Permanently` pagina spostata
- `4xx` Errore del Client
  - `400 Bad Request` richiesta malformata
  - `404 Not Found` contenuto non trovato
- `5xx` Errore del Server
  - `500 Internal Server Error` errore interno al server
  - `503 Service Unavailable` servizio non disponibile

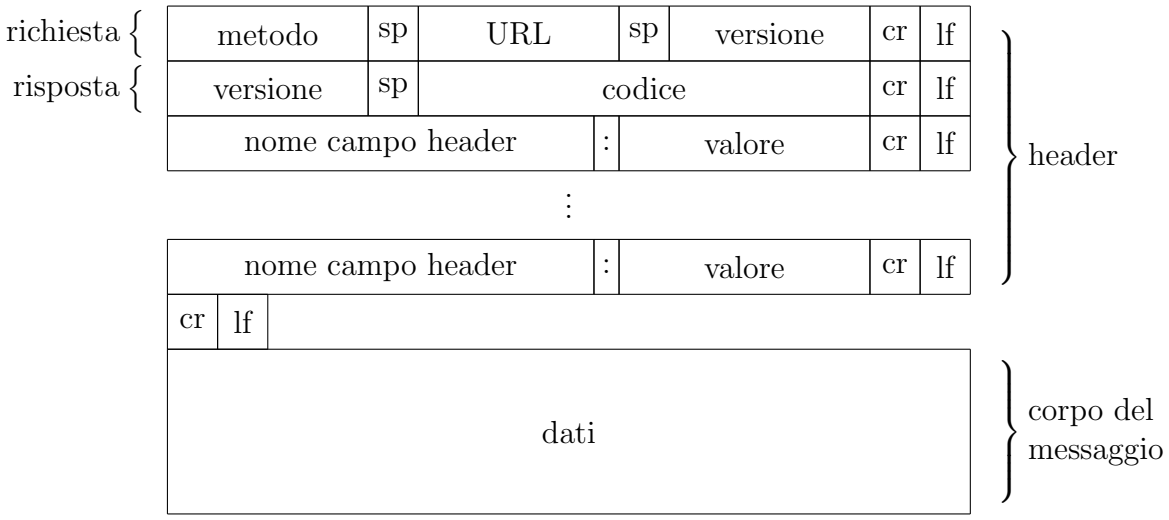
Di seguito un esempio di richiesta HTTP:

```
GET /somedir/page.html HTTP/1.1
Host:  www.someschool.edu
User-agent:  Mozilla/4.0
Connection:  close
Accept-language:  it
```

Di seguito un esempio di risposta HTTP:

```
HTTP/1.1 200 OK
Connection:  close
Date:  Thu, 06 Aug 1998 12:00:15 GMT
Server:  Apache/1.3.0 (Unix)
Last-Modified:  Mon, 22 Jun 1998
Content-Length:  6821
Content-Type:  text/html
...dati utente ...
```

Si puó generalizzare la richiesta o risposta HTTP come segue:



Da Notare: Ogni parte del messaggio viene separata dai caratteri **carriage return (cr)** *(cursore ad inizio riga)* e **line field (lf)** *(cursore spostato nella riga sottostante)*. Per separare le parti della prima riga si utilizza un **separatore (sp)**, solitamente uno spazio.

4.3.3 I Cookie

Navigare per il Web comporta il recupero di una serie di pagine indipendenti: il browser invia una richiesta a un server, ottiene il file, dopo di che il server dimentica di aver mai visto quel particolare client. Questo comportamento viene chiamato comportamento **stateless** *(senza stato)*.

Tuttavia esso non é adatto per fornire pagine differenti a utenti diversi a seconda dell'interazione avuta in precedenza col server. Questo comportamento é necessario per molti sistemi che prevedano una sequenza logia di scambi di informazioni con i siti Web, per esempio: alcuni siti Web richiedono agli utenti di registrarsi oppure per rendere l'esperienza sul sito piú gradevole.

Questo fa nascere il problema di come possono i server distinguere tra le richieste dei vari utenti. Il problema fu risolto col meccanismo, spesso criticato, dei **cookie** *(biscotti)*, inventati da Netscape nel 1994.

Quando un client richiede una pagina Web il server puó fornire informazioni aggiuntive sotto forma di cookie insieme alla pagina richiesta. Un cookie é un piccolo file (circa 4KB) che il server puó associare a un browser. I browser archiviano i cookie in un'apposita directory sul disco rigido del client, a meno che l'utente non li abbia disattivati. I cookie potrebbero teoricamente contenere un virus, ma visto che sono trattati come dati testuali, non esiste un modo ufficiale per mandare in esecuzione l'ipotetico virus.

Un cookie é composto dai seguenti campi:

`Dominio-Percorso-Contenuto-Scadenza-Sicuro`

dove:

- `Dominio` indica il nome del dominio da dove proviene
- `Percorso` é un percorso nella struttura delle directory del server che identifica quale parte della struttura di file del server puó utilizzare il cookie
- `Contenuto` campo sotto forma di *chiave = valore*
- `Scadenza` specifica quando scade il cookie. Se il campo é assente, il browser scarta il cookie alla sua chiusura: un cookie di questo tipo é definito **cookie non persistente**. Se vengono fornite una data e un'ora, il cookie é **persistente** ed é conservato fino alla scadenza.
- `Sicuro` puó essere impostato per indicare che il browser puó restituire il cookie a un server solo usando un sistema di trasporto sicuro.

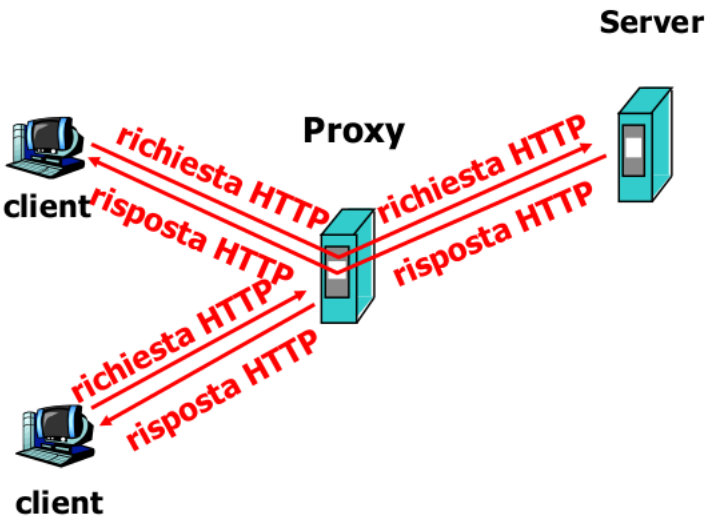
Il browser, prima di inviare una richiesta di una pagina Web, cerca nell'apposita directory tutti i cookie inseriti con dominio uguale al dominio richiesto. Se la ricerca porta dei risultati, vengono inclusi nel messaggio di richiesta. Quando il server li riceve, puó interpretarli come desidera.

Un impiego piú controverso é quello di tracciare il comportamento on-line degli utenti. Questo permette agli operatori di siti Web di capire come gli utenti navigano nel loro sito; le aziende pubblicitarie costruiscono profili di pubblicitá o siti che un particolare utente ha visitato. Purtroppo molti utenti sono completamente inconsapevoli di questa collezione di informazioni, a volte anche riguardante l'indirizzo mac o il browser utilizzato, oppure la lingua e il sistema operativo.

4.3.4 Proxy Web

Una strategia per aumentare l'efficacia delle cache consiste nel condividerle tra piú utenti. In questo modo una pagina giá richiesta da un utente puó essere restituita ad un altro senza che la richiesta raggiunga effettivamente il web server. Naturalmente questa condivisione non puó essere fatta per: traffico codificato, pagine che richiedono l'autenticazione e per pagine che sono restituite da programmi.

Un **Proxy Web** é usato per fare condivisione di cache tra utenti. Un proxy, generalmente, é un agente che agisce in vece di qualcuno, per esempio l'utente, occupandosi delle richieste Web al posto dei suoi utenti.



Per usare il proxy ogni browser deve essere configurato in modo da fare le richieste delle pagine al proxy invece che al vero server. Se il proxy ha la pagina, la gestisce immediatamente; altrimenti, prende la pagina dal server, la memorizza nella cache per usarla ancora in futuro e la restituisce al client che l'ha richiesta. Il proxy viene consultato solo dopo che il browser abbia tentato di soddisfare la richiesta usano la propria cache.
Quindi il proxy determina:

- un miglioramento del tempo di risposta delle risorse Web
- riduce il traffico verso internet, cruciale quando la tariffazione della linea é svolta tramite conteggio della banda utilizzata dagli utenti.
- puó filtrare i contenuti
- garantisce l'anonimato dell'utente dal server

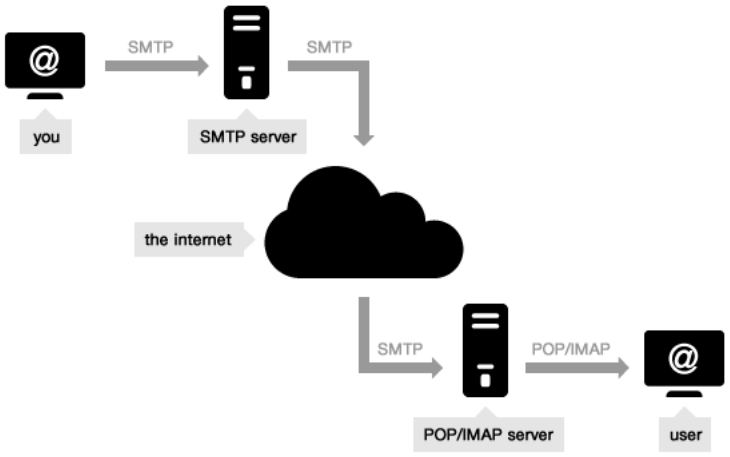
Proxy $\neq$ NAT La differenza tra Web proxy e NAT é che il primo possiede la nozione semantica dell'applicazione, ovvero esiste un proxy che esegue le richieste per ogni tipo di applicazione, invece il NAT avviene a livello di rete, ovvero per ogni comunicazione che avviene nei livelli superiori, viene eseguito la traduzione di indirizzo.

5 Posta Elettronica

La **Posta Elettronica** *(e-mail)* é disponibile da piú di tre decenni. Piú veloce ed economica della posta tradizionale, l'e-mail é stata un'applicazione popolare fin dai primi esordi di Internet. Prima del 1990 era utilizzata principalmente nella universitá. Successivamente é divenuta nota al grande pubblico ed é cresciuta in modo esponenziale. Sfortunatamente, come per le lettere cartacee, la maggior parte delle e-mail, 9 su 10, é costituita da **spam** *(messaggi spazzatura)*.

I protocolli e-mail si sono evoluti nel corso degli anni. I primi sistemi di posta elettronica erano semplicemente protocolli di trasferimento per file testuali, con la convenzione che la prima riga di ogni messaggio (o file) conteneva l'indirizzo del destinatario. Con il trascorrere del tempo l'e-mail si é diversificata dal trasferimento file e sono state aggiunte molte proprietá, quali la capacitá di inviare un messaggio a una lista di destinatari. Negli anni '90 divennero importati le funzionalitá multimediali per inviare messaggi contenenti immagini o altro materiale non testuale. Anche i programmi di lettura delle e-mail divennero piú sofisticati, spostandosi da interfacce basate sul testo a interfacce grafiche e aggiungendo la possibilitá di accedere alla proprie e-mail da qualsiasi computer ovunque ci si trovi.

5.1 Architettura e Servizi



L'architettura del sistema é composta da due sottosistemi: gli **user agent**, che consentono alle persone di leggere e inviare la posta elettronica, e i **message transfer agent**, che spostano i messaggi dalla sorgente alla destinazione. Questi ultimi sono informalmente chiamati **mail server**.

Alcune elaborazioni dello user agent possono essere fatte automaticamente: le e-mail in ingresso possono essere filtrate o attribuire una minore prioritá ai messaggi che potrebbero essere spam.

I message transfer agent sono tipicamente processi di sistema. Vengono eseguiti come servizi su appositi server e sono pensati per essere sempre disponibili. Il loro compito é di spostare automaticamente le e-mail attraverso il sistema della sorgente al destinatario facendo uso del protocollo SMTP.

SMTP invia le e-mail con un approccio orientato alla connessione e notifica lo stato della consegna e degli errori.

I message transfer agent implementano anche le **mailing list**, in cui copie identiche di un messaggio vengono consegnate a ogni iscritto alla lista. Altre funzionalitá avanzate sono le copie per conoscenza *(carbon copy, cc)*, le copie per conoscenza nascosta *(blind carbon copy, bcc)*, i messaggi di posta elettronica ad alta prioritá.

I concetti di **mailbox** *(caselle di posta)* e di formato standard dei messaggi e-mail sono il collegamento tra user agent e message transfer agent. Le mailbox memorizzano le e-mail ricevute dall'utente e sono mantenute dai mail server, mentre gli user agent semplicemente presentano agli utenti una visione dei contenuti delle loro mailbox. Per fare ció, gli user agent inviano ai mail server i comandi per manipolare le mailbox, ispezionando i loro contenuti, cancellando messaggi e cosí via. Con questa architettura per accedere a una mailbox un utente puó usare differenti user agent su diversi computer.

Un'idea basilare nei sistemi di posta elettronica é la distinzione tra l'*involucro* e il suo contenuto. L'involucro incapsula il messaggio e contiene tutte le informazioni necessarie per il suo trasporto, come l'indirizzo di destinazione, la prioritá e il livello di protezione. I message transfer agent utilizzano l'involucro per l'instradamento.

Il messaggio all'interno dell'involucro consiste in due parti: l'*intestazione* e il **corpo**. L'intestazione contiene informazioni di controllo per gli user agent, mentre il corpo é dedicato interamente al destinatario umano.

5.2 User Agent

Uno user agent *(o e-mail reader o client di posta)* é un programma che accetta una varietá di comandi per comporre, ricevere e rispondere ai messaggi, nonché per organizzare le caselle di posta. Ci sono molti user agent popolari quali Google Gmail, Microsoft Outlook e Mozzilla Thunderbird.

Gli user agent devono anche essere in grado di mostrare i messaggi in arrivo in modo che le persone possano leggere le proprie e-mail e, successivamente, decidere cosa farne: distruggerlo, inviare una risposta, inoltrare il messaggio o conservarlo.

L'archiviazione puó essere fatta automaticamente dallo stesso user agent prima che l'utente legga i messaggi. Un esempio comune si ha quando i campi e i contenuti dei messaggi sono ispezionati e usati, insieme a quanto detto dall'utente riguardo ai messaggi precedenti, per determinare se un messaggio puó essere spam. Molti ISP e aziende fanno uso di software che etichetta le e-mail come importanti o spam in modo che lo user agent possa archiviarle nella casella di posta corrispondente.

Gli utenti possono costruirsi delle regole di archiviazione. Ogni regola specifica una condizione e un'azione. Le cartelle piú importanti sono la *Inbox* e la *Junk e-mail*.

Una proprietá di base degli user agent é come comporre le e-mail. Questa include la creazione di messaggi o risposte a messaggi e l'invio del tutto nel sistema di e-mail per la consegna. I messaggi inviati hanno un formato standard che viene creato a partire dalle informazioni fornite dallo user agent. La parte pií importante del messaggio per il trasferimento é l'involucro, che contiene l'indirizzo della destinazione nella forma *user@indirizzo-dns*.

5.3 Trasferimento dei Messaggi

Il trasferimento delle e-mail é fatto dal protocollo **SMTP** *(Simple Mail Transfer Protocol)*. Il modo piú semplice per svolgere questa operazione é stabilire una connessione di trasporto della macchina sorgente a quella destinazione per poi trasferire semplicemente il messaggio. Cosí faceva originariamente SMTP. Tuttavia nel corso degli anni, si sono differenziati due diversi tipi di uso di SMTP. Il primo é la sottomissione della mail, modo in cui lo user agent invia messaggi nel sistema e-mail per la consegna, il secondo consiste nel trasferire direttamente i messaggi ai message transfer agent del destinatario.

5.3.1 SMPT e le sue estensioni

All'interno di Internet la posta elettronica viene consegnata costituendo una connessione tra la macchina sorgente e la porta 25 della macchina di destinazione. In ascolto su questa porta esiste un server di posta elettronica che utilizza **SMTP**.

SMTP é un semplice protocollo ASCII. Usare testo ASCII facilita lo sviluppo dei protocollo e il relativo debug: possono essere validati inviando comandi composta manualmente de i record dei messaggi sono facili da leggere. La maggior parte dei protocolli Internet di livello applicazione funzionano in questo modo.

SMTP nella sua forma base funziona bene, ma é limitato in alcuni aspetti: non include l'autenticazione, trasferisce i messaggi in ASCII e, soprattutto, in chiaro. Per risolvere questi problemi SMPT é stato rivisto per avere un meccanismo di estensioni. L'uso di SMTP con tali estensioni é chiamato **ESMTP** *(Extended SMTP)*.

5.3.2 Sottomissione delle e-mail

SMTP é normalmente usato per la sottomissione della e-mail con l'estensione *AUTH*, che permette al server di controllare le credenziali di accesso al servizio di posta elettronica del client.

Ci sono ulteriori differenze nel modo in cui SMTP é usato per la sottomissione della posta. Per esempio, la porta 587 é preferita alla porta 25 e il server SMTP puó controllare e correggere il formato dei messaggi inviati dallo user agent.

5.3.3 Message Transfer

Quando un message transfer agent riceve un messaggio da uno user agent, lo consegnerá al message transfer agent del destinatario usando SMTP. Per fare ció la sorgente usa l'indirizzo di destinazione.

Per determinare il corretto server di posta elettronica da contattare viene interrogato il sistema DNS riguardante il record MX **(o mail exchanger)** del dominio, restituendo una lista ordinata degli indirizzi IP di uno o piú server di posta elettronica. Il message transfer agent sorgente stabilisce quindi una connessione TCP sulla porta 25 verso l'indirizzo IP cosí ottenuto e usa SMTP per consegnare il messaggio. Il ricevente quindi metterá l'e-mail nella corretta casella di posta dell'utente.

Con questo processo di consegna la posta viaggia dal message transfer agent iniziale a quello finale con un solo salto. Non ci sono server intermedi nel trasferimento di messaggi.

5.4 Consegna Finale

La e-mail una volta arrivata alla casella di posta non rimane altro che trasferirne una copia allo user agent dell'utente per essere visualizzata. I compiti dello user agent sono quelli di visualizzare i contenuti della casella di posta e permettere che venga gestita da remoto. A questo scopo possono essere usati diversi protocolli differenti, ma non SMTP, che é un protocollo di *push*[2].

La consegna finale non puó essere eseguita in questo modo, sia perché la casella di posta deve continuare ad essere memorizzata sul mail server sia perché lo user agent potrebbe non essere connesso ad Internet nel momento in cui SMTP tenta di inoltrare i messaggi.

5.4.1 IMAP e POP3

Uno dei protocolli principali usati per la consegna finale é **IMAP** *(Internet Message Access Protocol)*. Per fare uso di IMAP il mail server esegue un server IMAP che rimane in ascolto sulla porta 143. Lo user agent esegue la versione client di IMAP che si connette al server e inizia la comunicazione.

Innanzitutto il client instaura una sessione di trasporto sicura e quindi effettua il login o un'altra autenticazione sul server. Una volta autenticato esistono molto comandi per elencare le cartelle e i messaggi, per prendere i messaggi o parte di essi, etichettarli per una successiva cancellazione e organizzarli in cartelle.

IMAP é un aggiornamento di un protocollo di consegna finale precedente, **POP3** *(Post Office Protocol version 3)*. Invece di rimanere sul server i messaggi sono solitamente scaricati dallo user agent del computer. Questo facilita la vita ai server, ma la complica agli utenti.

Tuttavia POP3 viene ancora usato. Possono essere usati anche protocolli proprietari, perché il protocollo puó essere impiegato tra server di posta elettronica e user agent forniti dalla stessa azienda; Microsoft Exchange é un sistema di e-mail con protocollo proprietario

[2]Un protocollo che quindi "spinge" *(push)* i dati verso la destinazione in contrapposizione ai protocolli di *pull* che invece li attirano a sé.