

Per la progettazione di reti informatiche vengono utilizzati i seguenti modelli architetturali :

### Client-Server (HTTP, IMAP, FTP)

Questo modello architettonico prevede l'esistenza di due entità :

- Client : un host non necessariamente attivo dotato di un indirizzo **IP dinamico** e che non comunica direttamente con tutti gli altri host client. Si organizzano comunicazioni tra client passando per i server
- Server : host sempre attivi con un indirizzo **IP fisso** locati in dei data-center per garantire una connessione e un hardware ottimali

La comunicazione viene iniziata sempre dal client, esso invia una richiesta al server attraverso la rete e attende la risposta, che viene inviata dal server. In questo modello, il server ha il compito di elaborare le richieste dei client e di fornire loro le informazioni richieste. Il modello client-server è spesso utilizzato in contesti in cui il controllo centrale è importante, come ad esempio nella gestione di database o nella distribuzione di contenuti web. Il server è sempre attivo e controlla le richieste dei client, fornendo risposte appropriate.

Nel modello client-server, il server dispone di un'interfaccia di rete attraverso la quale ascolta le richieste dei client. Questa interfaccia è definita come "ascolto attivo" o "listening socket" ed è una delle entità fondamentali del server. Quando il server riceve una richiesta dal client, crea una "socket di connessione" attraverso la quale il server e il client possono comunicare tra loro.

### Peer-to-Peer (P2P)

Il modello architettonico peer-to-peer (P2P) prevede la presenza di dispositivi che agiscono sia come client che come server. In questo modello, ogni dispositivo è in grado di richiedere informazioni o servizi ad altri dispositivi nella rete e di fornire informazioni o servizi ad altri dispositivi. Non c'è un server centrale che gestisce tutte le richieste, ma ogni dispositivo è in grado di comunicare con gli altri dispositivi della rete. In questo modo, ogni dispositivo può contribuire alla rete, inviando e ricevendo informazioni e servizi.

Il modello architettonico client-server prevede la presenza di un server centrale che fornisce informazioni e servizi ai client, mentre il modello architettonico peer-to-peer prevede la condivisione di informazioni e servizi tra i dispositivi nella rete, senza la presenza di un server centrale.

Due host comunicano attraverso un'interfaccia di comunicazione fornita dal SO. Un client comunica col server tramite messaggi e il loro scambio permette la **sincronizzazione** tra processi.

Il client invia sempre la comunicazione ed il server riceve rimanendo sempre in ascolto.

## Socket

3 processi inviano / ricevono messaggi alti/ dalle loro socket.  
Una socket è una coppia univoca IP, numero di porta che consente la comunicazione tra processi su sistemi operativi diversi.

Sono un modo per entrare/uscire dal SO.

es. IP address : 128.113.245.12  
port number : 80

HTTP server : 80  
mail server : 50

Il server ha porte e IP fatti e finiti mentre il client ha IP variabile e porta assegnata dal SO al momento della comunicazione.

Socket  $\Rightarrow$  è un'interfaccia software per la trasmissione e la ricezione di dati attraverso una rete. Essa c'è il punto in cui il codice applicativo di un processo accede alla comunicazione. Come un processo sull'una altra macchina tramite un porto.

Un socket permette di instaurare le varie comunicazioni all'interno di una rete, sfruttando la pila TCP/IP.

Un socket è composto da:

- indirizzo IP: codice univoco della macchina interessata (grandezza 32 BIT)
- numero di porta: numero usato dal singolo processo per richiedere il protocollo (grandezza 16 BIT)

I messaggi scambiati devono sottostare al protocollo, ed ogni protocollo è documentato dalla RFC.

I protocolli di trasporto godono delle seguenti proprietà:

- Trasferimento dati affidabile: i dati arrivano in modo corretto e completo
- Throughput: quantità di pacchetti spediti nell'unità di tempo (diverso dalla banda che rappresenta invece la capienza massima)
- Timing: avere la certezza che i dati arrivino entro un determinato lasso di tempo
- Sicurezza: fornire riservatezza impedendo che host terzi possano accedere ai messaggi scambiati

Classifichiamo nel seguente modo le porte

- 1) Well-known ports: sono le porte numerate da 0 a 1023. Sono riservate per l'uso da parte di protocolli noti e standardizzati.
  - Porta 20 e 21: utilizzate dal protocollo **FTP** (File Transfer Protocol) per il trasferimento di file tra server e client.
  - Porta 22: utilizzata dal protocollo **SSH** (Secure Shell) per consentire la connessione remota sicura a un server.
  - Porta 23: utilizzata dal protocollo **Telnet** per la connessione remota a un server.
  - Porta 25: utilizzata dal protocollo **SMTP** (Simple Mail Transfer Protocol) per l'invio di e-mail.
  - Porta 53: utilizzata dal protocollo **DNS** (Domain Name System) per la risoluzione dei nomi di dominio.
  - Porta 80: utilizzata dal protocollo **HTTP** (Hypertext Transfer Protocol) per la comunicazione tra client e server web.
  - Porta 110: utilizzata dal protocollo **POP3** (Post Office Protocol 3) per la ricezione di e-mail.
  - Porta 143: utilizzata dal protocollo **IMAP** (Internet Message Access Protocol) per la ricezione di e-mail.
  - Porta 443: utilizzata dal protocollo **HTTPS** (Hypertext Transfer Protocol Secure) per la comunicazione sicura tra client e server web.
- 2) Registered ports: sono le porte numerate da 1024 a 49151. Sono assegnate a protocolli di rete specifici, ma possono essere utilizzate anche da applicazioni utente.
  - Porta 1433: utilizzata dal protocollo SQL Server per la comunicazione con il database.
  - Porta 3306: utilizzata dal protocollo MySQL per la comunicazione con il database.
  - Porta 8080: utilizzata dal protocollo HTTP alternativo per la comunicazione web.
- 3) Dynamic ports: sono le porte numerate da 49152 a 65535. Sono utilizzate da applicazioni che richiedono l'uso di porte effimere, che vengono assegnate dinamicamente dal sistema operativo durante la creazione di una connessione di rete.

#### Telnet → porta 23 TCP

Telnet è un protocollo di livello applicativo che utilizza il modello client-server per consentire agli utenti di accedere ai servizi di un dispositivo remoto, solitamente per la configurazione di applicativi. Un utente (client) si connette a un server Telnet sulla porta 23 e, una volta stabilita la connessione, può interagire con il dispositivo remoto. Viene utilizzato per stabilire una connessione remota con un dispositivo di rete, in particolare, Telnet consente agli utenti di accedere e controllare un dispositivo remoto come se fossero fisicamente presenti di fronte ad esso. Telnet utilizza un formato di messaggio basato su testo, in cui i comandi e le risposte sono trasmessi come stringhe di caratteri ASCII.

Tuttavia, Telnet ha alcuni problemi di sicurezza, in quanto le credenziali di accesso (username e password) e tutti i dati trasmessi sono inviati in chiaro, senza crittografia. Per questo motivo, Telnet è stato sostituito dal protocollo SSH (Secure Shell), che fornisce un livello di sicurezza superiore attraverso la crittografia dei dati trasmessi.

## FTP (File Transfer Protocol) —> porta 20 e 21

FTP è un protocollo di rete di livello applicativo utilizzato per trasferire file da un host a un altro su una rete IP che utilizza il modello client-server. Un server FTP viene eseguito su un host remoto e gli utenti possono connettersi al server utilizzando un client FTP. Il client e il server FTP comunicano tra loro utilizzando una serie di comandi FTP e risposte, che sono trasmessi tramite una connessione TCP (Transmission Control Protocol) sulla porta 21.

Durante la connessione FTP, il client FTP si connette al server FTP sulla porta 21 e invia le credenziali di accesso (username e password). Una volta autenticato, il client può inviare comandi FTP al server per eseguire operazioni di trasferimento file, come ad esempio l'elenco dei file, la creazione di directory, la cancellazione di file e il trasferimento di file da un host a un altro.

Il protocollo FTP prevede anche l'uso di una seconda connessione TCP per il trasferimento dei dati, che viene aperta dal server sulla porta 20. Tuttavia, a causa di alcune problematiche legate alla sicurezza e alla configurazione dei firewall, FTP può essere configurato per utilizzare una singola connessione TCP per il trasferimento dei dati (modo passivo). FTP è stato uno dei primi protocolli di trasferimento file sviluppati per la rete, ma presenta alcune vulnerabilità di sicurezza, in quanto tutte le informazioni, comprese le credenziali di accesso, vengono trasmesse in chiaro. Gli host devono autenticarsi prima di iniziare la connessione.

## HTTP (Hypertext Transfer Protocol) —> porta 80

HTTP è un protocollo di livello applicativo che viene utilizzato per la trasmissione di informazioni su internet, in particolare per la richiesta di pagine web. HTTP è basato sul modello client-server, in cui un client richiede delle informazioni a un server e il server risponde fornendo i dati richiesti. La connessione HTTP è basata su una connessione TCP stabilita sulla porta 80 (o sulla porta 443 se si utilizza HTTPS, la versione crittata di HTTP). Quando un client richiede una risorsa ad un server, invia una richiesta HTTP che contiene un metodo (ad esempio GET, POST, PUT, DELETE) ed un URL che identifica la risorsa richiesta. Il server risponde inviando una risposta HTTP che contiene uno stato (ad esempio 200 OK, 404 Not Found, 500 Internal Server Error) ed i dati richiesti.

Ad ogni richiesta la sessione viene chiusa, quindi viene generata una nuova sessione per ogni richiesta. Se in una pagina sono presenti degli oggetti viene generata una sessione per ogni oggetto (con HTTP 1.0). HTTP 1.1 invece genera una sessione per ogni pagina, quindi chiude la sessione solo dopo aver caricato tutti gli oggetti che conteneva, in un'unica sessione.

HTTP 1.0 è un perfetto esempio di comunicazione stateless perché non tengono conto dello stato della connessione e dello storico degli host, infatti ogni richiesta HTTP è totalmente indipendente dalle precedenti e dalle successive. Supponiamo che si voglia contattare una pagina web che richieda di impostare la lingua per migliorare la visualizzazione del contenuto. Se non si tiene conto dello stato dell'utente ogni volta si va a richiedere la lingua per ogni sessione generata. HTTP 1.1 sopperisce a questa mancanza tramite l'utilizzo dei cookie.

## Cookie

I cookie nel protocollo HTTP sono dei piccoli file di testo che vengono memorizzati sul dispositivo del client (ad esempio il browser) dal server quando si effettua una richiesta HTTP. Questi file di testo contengono informazioni sulle preferenze dell'utente e sullo stato della sessione tra il client e il server. In questo modo, il server può utilizzare le informazioni contenute nel cookie per mantenere lo stato della sessione tra le varie richieste HTTP inviate dal client.

I cookie possono contenere informazioni come le preferenze dell'utente, le informazioni sul login, le informazioni sul carrello degli acquisti, ecc. Inoltre, i cookie possono essere utilizzati anche per tracciare l'attività dell'utente sul sito web, ad esempio per fornire pubblicità mirate.

## HTTP Request e HTTP Response

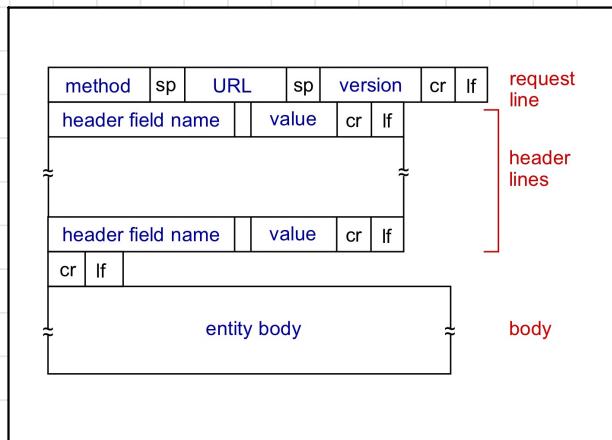
Una richiesta HTTP è costituita da una riga di richiesta (request line) seguita da una serie di header HTTP (HTTP headers) e, optionalmente, dal body della richiesta (request body). La riga di richiesta specifica il metodo HTTP utilizzato per la richiesta (ad esempio GET, POST, PUT, DELETE), l'URL della risorsa richiesta e la versione di HTTP utilizzata. Gli header HTTP contengono informazioni aggiuntive sulla richiesta, come ad esempio il tipo di contenuto richiesto (Content-Type), l'indirizzo IP del client (X-Forwarded-For), il tipo di compressione supportato (Accept-Encoding), ecc. Il body della richiesta, presente solo in alcuni metodi HTTP come POST o PUT, contiene i dati trasmessi dal client al server, come ad esempio un modulo di registrazione o un file da caricare.

Una risposta HTTP è costituita da una riga di stato (status line) seguita da una serie di header HTTP (HTTP headers), optionalmente, dal body della risposta (response body). La riga di stato specifica lo stato della richiesta, rappresentato da un codice numerico a tre cifre, come ad esempio 200 OK per una richiesta eseguita con successo, 404 Not Found per una risorsa non trovata, ecc. Gli header HTTP contengono informazioni aggiuntive sulla risposta, come ad esempio il tipo di contenuto restituito (Content-Type), la data e l'ora di generazione della risposta (Date), la dimensione del contenuto (Content-Length), ecc. Il body della risposta contiene i dati restituiti dal server al client, come ad esempio il contenuto di una pagina web o un file richiesto.

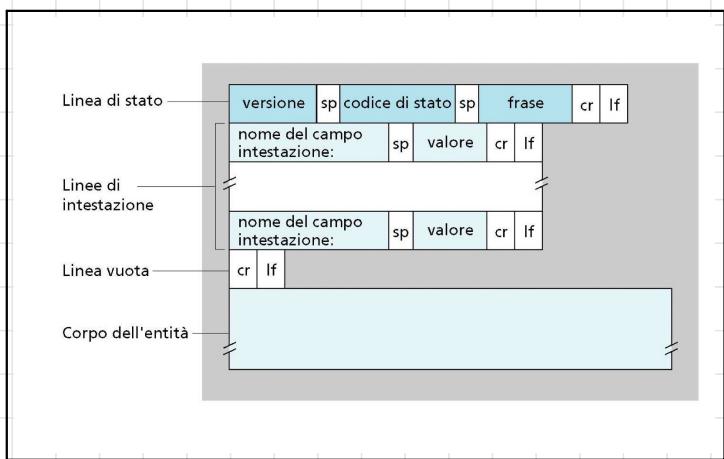
In sintesi, una richiesta HTTP è un messaggio inviato da un client a un server per richiedere una risorsa, mentre una risposta HTTP è un messaggio inviato dal server al client in risposta ad una richiesta, contenente la risorsa richiesta o un messaggio di errore. Entrambe le richieste e le risposte sono costituite da una riga di stato (request line o status line), una serie di header HTTP (HTTP headers) e optionalmente dal body della richiesta o della risposta (request body o response body).

## HTTP - Request

La struttura di una richiesta HTTP



# La struttura di una richiesta HTTP



I metodi presenti nel metodo della richiesta HTTP sono i seguenti:

Method	Description
GET	Request to read a Web page
HEAD	Request to read a Web page's header
PUT	Request to store a Web page
POST	Append to a named resource (e.g., a Web page)
DELETE	Remove the Web page
TRACE	Echo the incoming request
CONNECT	Reserved for future use
OPTIONS	Query certain options

put=Richiesta di memorizzazione di una pagina Web

get=Richiesta di lettura di una pagina web

GET => mette nell' URL i campi richiesti al web server

POST => ha lo stesso scopo del metodo GET ma non scrive nell' URL i campi (es. usato per richiedere i login)

le risposte HTTP forniscano i seguenti codici da ritornare al richiamante (browser)

#### Information Codes

100 Continue

#### Success Codes

200 OK

203 Non-Authoritative Information

204 No Content

#### Redirection Codes

305 Use Proxy

#### Client Error Codes

400 Bad Request

403 Forbidden

404 Not Found

405 Method Not Allowed

#### Server Error Codes

500 Internal Server Error

505 HTTP Version not supported

## HTTP 2.0

I cambiamenti proposti non richiedono nessuna modifica al modo di lavorare delle applicazioni web esistenti, ma le nuove applicazioni possono avvantaggiarsi delle novità introdotte per incrementare la velocità.[7]

HTTP/2 mantiene ad alto livello la maggior parte della sintassi di HTTP 1.1 come metodi, codici di stato, campi degli header, URI. La differenza sta nel modo in cui è strutturato e trasportato il flusso dei dati tra il client e il server.[7] I siti web efficienti minimizzano il numero di richieste necessarie per restituire una pagina con la tecnica del "minifying" o minimizzazione (riducendo le dimensioni del codice e impacchettando piccoli pezzi di codice in unità più grandi, senza intaccarne la funzionalità) applicata su risorse come immagini e script. Ad ogni modo la minimizzazione non è necessariamente conveniente né efficiente e può ancora richiedere connessioni HTTP distinte per ottenere la pagina e le risorse minimizzate. HTTP/2 permette al server di inviare ("push") più dati di quelli richiesti dal client. Questo consente al server di fornire dati che sa essere necessari ad un web browser per completare la pagina, senza attendere che il browser esamini la prima risposta e senza l'overhead di un ciclo di richiesta addizionale.[8]

Altri miglioramenti prestazionali nella prima stesura di HTTP/2 (che era una copia di SPDY) vengono dal multiplexing di richieste e risposte, allo scopo di evitare le problematiche di tipo HOLB (head-of-line blocking) note in HTTP/1.1 (anche quando viene utilizzata la tecnica dell'HTTP pipelining), compressione degli header, gestione delle richieste in base alla priorità delle stesse (prioritazione).

## HTTP 3.0

HTTP/3 è la terza versione del protocollo Hypertext Transfer Protocol usato per il World Wide Web nonché il successore di HTTP/2.[1][2] HTTP/3 è basato su una precedente bozza di RFC intitolata Hypertext Transfer Protocol (HTTP) over QUIC.[3] QUIC è un protocollo di rete di livello di trasporto sviluppato originariamente da Google in cui il controllo della congestione nello spazio utente viene realizzato su protocollo User Datagram Protocol (UDP) e proprio l'appoggiarsi su QUIC/UDP invece che su TCP rappresenta la caratteristica distintiva di HTTP/3. Quindi trova un modo per non by-passare il controllo di congestione.