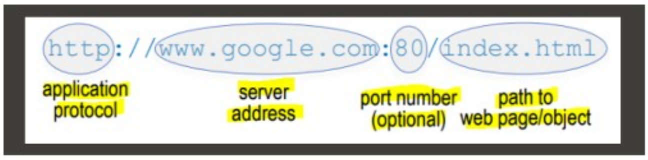
Protocollo HTTP

mercoledì 2 agosto 2023 16:25

HTTP è un protocollo di (a livello) applicativo web standardizzato, definito nel documento pubblico *RFC*.
È implementato in due processi, un client e un server, in esecuzione su sistemi periferici differenti; **HTTP definisce la struttura dei messaggi e la modalità con cui i due processi li scambiano tra loro**.
Una pagina web è composto da un **file HTML base** che **referenzia diversi oggetti** che fanno parte della sua composizione come immagini, file javascript, altri file HTML ecc.
Questi oggetti, quelli per cui viene fatta una richiesta, sono referenziati ed identificati attraverso il loro URL...



HTTP quindi definisce in che modo i client web, ad esempio un browser, richiedono le pagine ai web server, ad esempio apache, e come quest'ultimi trasferiscono le pagine richieste ai client.

Ora vediamo piu' nel dettaglio il meccanismo...

Quando un client fa richiesta, il browser invia al server messaggi di richiesta HTTP per gli oggetti della pagina, il protocollo di trasporto previsto da HTTP è TCP, quindi prima di tutto il client deve stabilire una connessione TCP con il server; una volta stabilita la connessione client e server comunicano attraverso la loro interfaccia socket, quando un client manda un messaggio questo è nelle mani di TCP, il livello inferiore, da quel momento HTTP non se ne dovrà piu' occupare.

***Nota:** HTTP è uno StateLess Protocol, i server HTTP non mantengono informazioni sui client e sulle loro richieste*

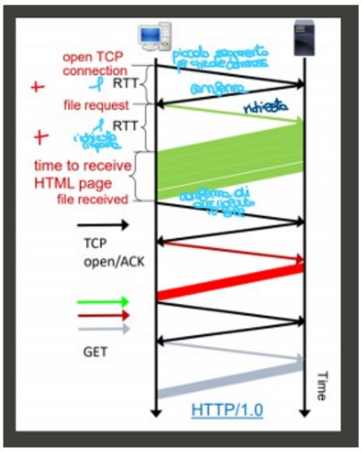
Due connessioni HTTP: persistenza e non persistenza

Il client inoltra una serie di richieste al server, in sequenza o ad intervalli regolari, e il server risponde ad ognuna di esse, la comunicazione come già sappiamo avviene su TCP, quindi ogni volta lo stesso client che richiede allo stesso server dovrà stabilire una connessione TCP diversa, questo è il caso di una **connessione non persistente**.
Una connessione persistente, quindi, è che il client invia richieste tutte sulla stessa connessione TCP, in questo caso il client non dovrà ogni volta prima richiedere di stabilire una connessione.

- **Connessioni non persistenti:** supponiamo ci sia una pagina HTML principale che referenzia 10 oggetti, risiedono tutti sullo stesso server, URL: http://www.someSchool.edu/home.index. Ecco cosa succede:
    1) Il processo client HTTP inizializza una connessione TCP con il server www.someSchool.edu sulla porta 80 (standardizzata);
    2) Il processo server HTTP accetta la connessione notificando il client;
    3) Il client tramite la sua socket invia al server il messaggio di richiesta HTTP che include il path sopra indicato, con questo il client sta dicendo che vuole quell'oggetto;
    4) Il server riceve il messaggio, recupera l'oggetto e lo incapsula in un messaggio di risposta HTTP;
    5) Il server comunica a TCP di chiudere la connessione, tuttavia TCP la chiuderà veramente quando è certo che il messaggio al client sia stato ricevuto ed integro ;
       Ripete fino al passo 5 fino a che sono stati restituiti tutti gli oggetti desiderati dal client
    6) Il client HTTP riceve il messaggio di risposta, la connessione termina e successivamente il client si occuperà di estrarre il file dalla risposta, esaminarlo e trovare tutti gli oggetti.

  L'utente per richiedere la pagina web apre 11 connessioni TCP!

Calcoliamo ora una stima dell'intervallo di tempo che occorre tra una richiesta di un file HTML e una risposta... Per fare questo ci serve **RTT**: il tempo impiegato da un pacchetto per viaggiare dal client al server e poi tornare al client (sono inclusi i ritardi di propagazione, di accodamento nei router e switch, ritardi di elaborazione). Questo è quello che succede quando un utente clicca su un collegamento ipertestuale...



Il browser inizializza una connessione TCP con il server questo comporta handshaking
(le scritte blu sono handshaking nell'immagine)

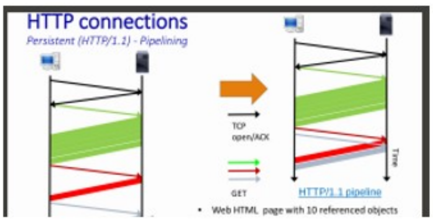
Il tempo totale di risposta approssimativamente è di 2 * RTT + il tempo di trasmissione da parte del server del file HTML, ricaviamo ora la formula...

non-persistent HTTP response time =

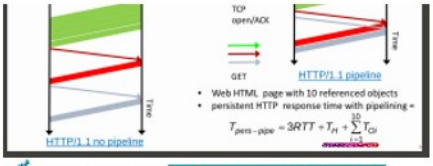
$$T_{non-pers} = 2RTT + T_H + \sum_{i=1}^{10}(2RTT + T_{Oi})$$

Sono evidenti gli svantaggi delle connessioni non persistenti, ovvero nel dover ogni volta richiedere una connessione ad ogni richiesta, dobbiamo allocare ad ogni richiesta delle risorse diverse causando un grave onere sul web server quando deve servire tanti client e infine questo passaggio richiede anche 1 RTT in piu' per stabilire la connessione ogni volta.

- **Connessioni persistenti:** ci agganciamo all'esempio visto precedentemente, in questo caso il server lascia la connessione TCP aperta dopo l'invio della risposta della prima richiesta nella loro storia di comunicazione, le successive richieste verranno trasmesse nella stessa connessione, possono essere sequenziali svolte una dopo l'altra senza aspettare la risposta della richiesta precedente si crea così una "coda di richieste" pendenti, **pipeling** modalità di default di HTTP. Il server chiuderà la connessione quando essa rimane inattiva per un dato lasso di tempo.



la connessione TCP aperta dopo l'invio della risposta della prima richiesta nella loro storia di comunicazione, le successive richieste verranno trasmesse nella stessa connessione, possono essere sequenziali svolte una dopo l'altra senza aspettare la risposta della richiesta precedente si crea così una "coda di richieste" pendenti, **pipeling** modalità di default di HTTP. Il server chiuderà la connessione quando essa rimane inattiva per un dato lasso di tempo.
Abbiamo un solo RTT per tutti gli oggetti referenziati, dimezzando così i tempi di risposta, applichiamo la formula sempre sull'esempio...



$$T_{pers} = 2RTT + T_H + \sum_{i=1}^{10}(RTT + T_{Oi})$$

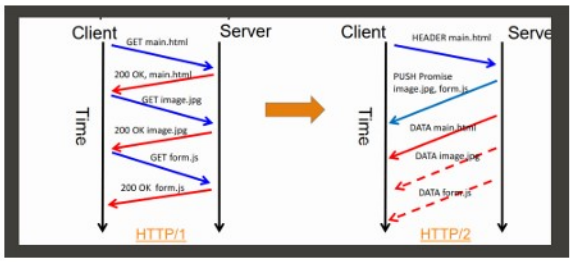
con pipeling RTT si taglie completamente

per inviare ip richiesta

Pipeling in HTTP/1: *gli oggetti vengono inviati nello stesso ordine con cui vengono richiesti, tuttavia può capitare che quando faccio le richieste non conosco le dimensioni degli oggetti, potrebbero esserci oggetti piccoli e oggetti grandi e quelli piccoli dovranno aspettare che prima sia inviato l'oggetto grande, questo fenomeno di chiama **HOL blocking**. È inevitabile che gli oggetti piccoli subiranno ritardi dovuti alla trasmissione dell'oggetto grande...*

I Browser HTTP/1 utilizzano piu' **connessioni TCP parallele** avendo così oggetti della stessa pagina inviati parallelamente; se ci sono *n* connessioni TCP ogni connessione ottiene circa *1/n larghezza di banda*, quindi piu' il browser apre connessioni parallele piu' larghezza di banda ha! Questo però comporta piu' socket che il server deve mantenere...

HTTP/2 permette di suddividere gli oggetti (grandi) in **"frame"** (piccoli) da trasmettere "mescolati" ed interfogliati con altri file interi di piccole dimensioni *(Esempio: con questa suddivisione degli oggetti possiamo ricondurci alle immagini ad alta risoluzione, inizialmente appare sfocata e mano a mano aumenta di risoluzione)*

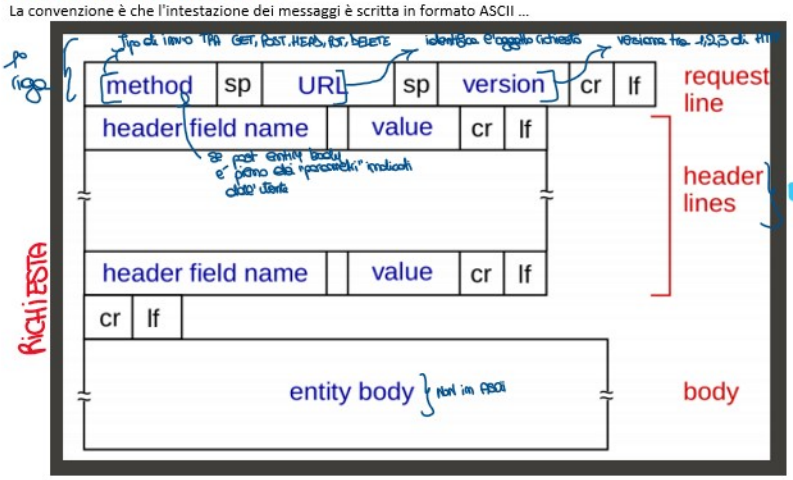


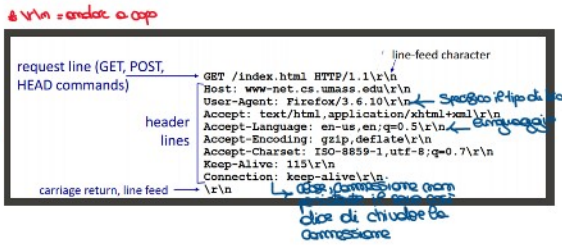
Viene introdotto il **concetto di priorità**: un client quando invia richieste al server può dare una priorità a questa assegnando un peso da 1 a 256 a ogni messaggio, dove il numero piu' alto con priorità maggiore è 256.
Inoltre il server può ora inviare piu' risposte al client per singola richiesta: il server può effettuare un invio **push** al client di oggetti aggiuntivi, serve così viene fatta una sola richiesta per caricare una pagina intera: la pagina di base HTML indica gli oggetti aggiuntivi necessari per eseguire il rendering completo della pagina web, il server identifica questi oggetti necessari e li invia al client prima di ricevere le richieste esplicite per essi.

HTTP/3 il servizio di trasporto previsto è UDP e viene aggiunto un nuovo protocollo **QUIC** interposto tra il livello applicativo (con HTTP/3) e il livello di trasporto (con UDP), il quale scopo è quello di fornire comunque un servizio orientato alla connessione eliminando il blocco HOL del TCP.

Formati dei messaggi HTTP

La convenzione è che l'intestazione dei messaggi è scritta in formato ASCII ...





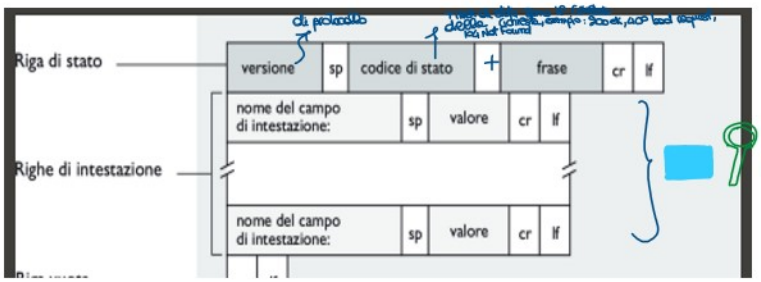
Metodo POST:
Viene usato in genere quando un utente *riempe un form*, ovvero passa dei "parametri" alla richiesta, essa dipenderà quindi dai dati immessi dall'utente nel *form*.

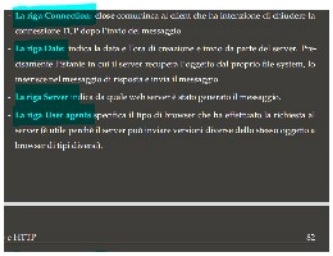
Metodo GET:
Include i dati immessi dall'utente in un *form* nella struttura, nel campo, URL della richiesta HTTP.

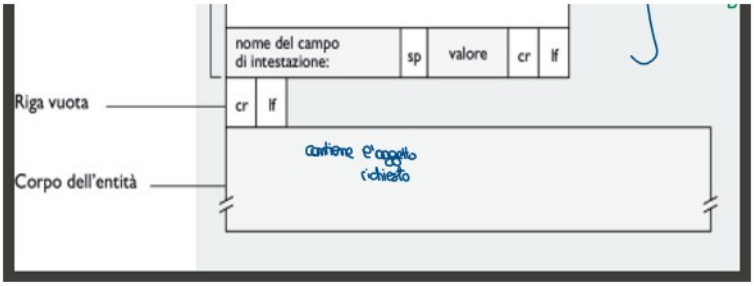
Metodo HEAD:
Come metodo è simile al *get* ma produce output diverso, ovvero *tralascia gli oggetti richiesti*. Viene usato ad esempio dagli sviluppatori per verificare la correttezza del codice prodotto.

Metodo PUT:
Consente agli utenti di *inviare oggetti*, oltre che la richiesta, al web server.

Metodo DELETE:
Consente agli utenti di *cancellare un oggetto* su un web server.







e HTTP 82

- La riga Last-Modified: indica l'istante e la data in cui l'oggetto è stato creato o modificato l'ultima volta.
- La riga content-length: specifica il numero di byte dell'oggetto inviato
- Content-Type: specifica il tipo e il sottotipo dell'oggetto
- Keep Alive: è il tempo della connessione.