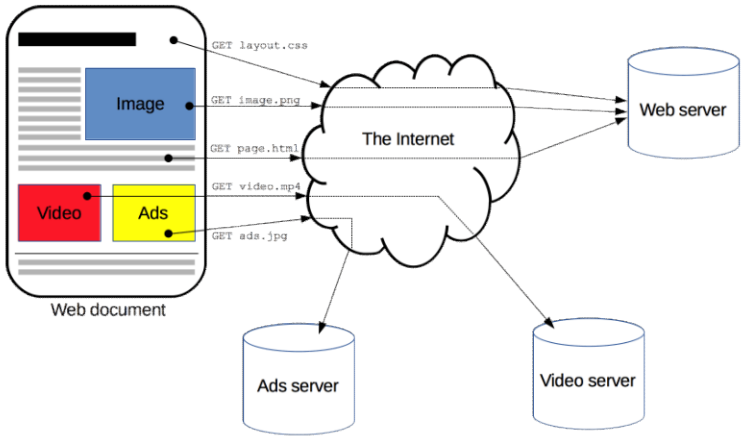
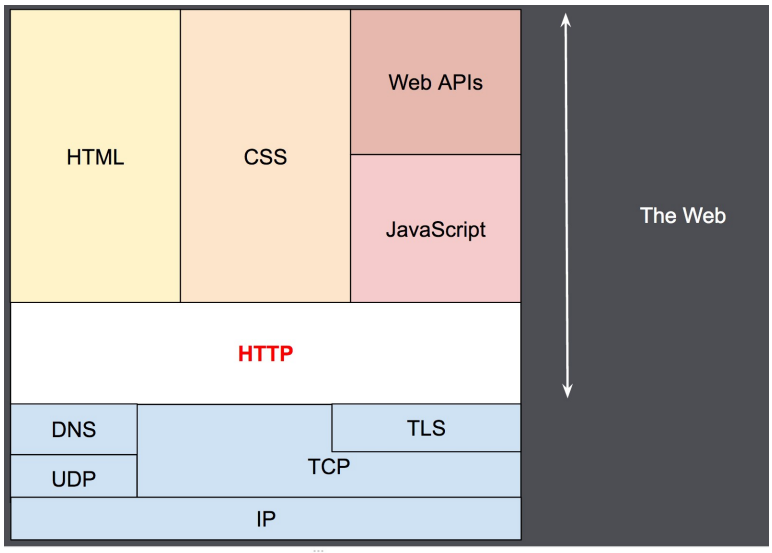
HTTP

martedì 6 febbraio 2024 13:10

L'HTTP è un protocollo utilizzato per andare a prendere risorse come ad esempio documenti HTML. È il fondamento di qualsiasi scambio di dati sul web ed è un protocollo client-server, ovvero la request sono inoltrate dal ricevente, di solito il browser Web.
Il documento completo viene ricostruito dai diversi sotto-documenti presi, come testo, descrizioni, immagini, video, script e altro.



Clients e servers comunicano scambiando messaggi individuali (al contrario di uno stream di dati). I messaggi inviati dal client, di solito il web browser, sono detti requests e i messaggi inviati dal server come risposta sono detti responses.



L'HTTP è un protocollo estendibile che si è evoluto nel tempo. È un protocollo del layer applicativo che viene inviato tramite TCP, anche se teoricamente può essere utilizzato un qualsiasi protocollo di trasmissione affidabile. Grazie alla sua estensibilità, viene utilizzato non solo per andare a prendere documenti iper testuali, ma anche immagini e video o per postare contenuti al server, come con i risultati di un form HTML. HTTP può anche essere utilizzato per prendere parti di documenti per aggiornare pagine web su richiesta.

Componenti dei sistemi HTTP-based

HTTP è un protocollo client-server: le requests vengono inoltrate da un'entità, l' user agent (o da un proxy a suo nome). Speso lo user-agent è un browser, ma può essere qualsiasi cosa.
Ogni request è inviata ad un server, che gestisce e fornisce una response: tra il client e il server vi sono numerose entità, chiamate proxies, che svolgono diverse operazioni e si comportano da gateway o caches, per esempio.
In realtà, ci sono più computer tra un browser e il server che gestisce la request: ci sono router, modem ecc… Grazie al design a strati del Web, questi sono nascosti nella rete e nel livello di trasporto, mentre HTTP sta più in alto, nel livello applicativo. Anche se importanti a livello di diagnostica di problemi di rete, i livelli inferiori sono perlopiù irrilevanti per le funzioni dell' HTTP.

Client: l' user-agent

Lo user-agent è lo strumento che agisce per conto dell'utente (es. Browser): esso corrisponde sempre all'entità che inoltra la request, non è mai il server a farlo.
Per mostrare una pagina web, il browser invia una richiesta di fetch del documento HTML che rappresenta la pagina. Successivamente analizza il file, facendo request addizionali che corrispondono a tutti i sotto-documenti della pagina (immagini, video, script, layout). A questo punto, il browser combina queste risorse per presentare il documento completo, la pagina web.
Una pagina web è, quindi, un documento iper testuale. Ciò significa che alcuni contenuti esposti in essa sono link, che possono essere attivati per aprire una nuova pagina web, permettendo all'utente di navigare.
Il browser traduce queste direttive dell'utente in request HTTP ed interpreta tutte le responses HTTP das pese tare all'utente.

Web server

Dall'altra parte della comunicazione c'è il server, che fornisce i documenti richiesti dal client. Un server appare virtualmente come una singola macchina, ma può in realtà essere costituito da un insieme di server che condividono il carico, o altri software (caches, database server, e-commerce servers) che generano il documento su richiesta.
Un server non corrisponde necessariamente ad una singola macchina, ma diverse istanze software server possono essere ospitate sulla stessa macchina. Con HTTP/1.1 e l'host header, potrebbero anche condividere lo stesso indirizzo IP.

Proxies

Tra il browser e il server, diversi computer e macchine dipendono dai messaggi HTTP. La maggior parte di questi risiede nel layer di trasporto, nel layer di rete o nel layer fisico, diventando invisibili agli occhi dell'HTTP.
Quelli che invece operano al livello applicativo vengono detti proxies: possono essere trasparenti, inoltrando la request che ricevono senza venire alterati in nessun modo, oppure non-trasparenti, cambiando la request in qualche modo prima di mandarla al server. I proxies possono avere diverse funzioni:

- Funzioni di caching (pubblica o privata)
- Filtraggio (scan antivirus o parental control)
- Bilanciamento del carico (per permettere a molteplici server di gestire diverse

requests)
- Autenticazione (per controllare l'accesso a diverse risorse)
- Logging (permettere di conservare informazioni storiche)

Aspetti base dell'HTTP

- ***HTTP è semplice***: è progettato per essere semplice e leggibile, anche con la complessità aggiunta in HTTP/2 dall'incapsulamento dei messaggi in frame. I messaggi HTTP possono essere letti e compresi dall'uomo.
- ***HTTP è estensibile***: gli header HTTP rendono questo protocollo facile da estendere e sperimentare. Le nuove funzionalità possono essere introdotte da un semplice accordo tra un client e un server sulla semantica di un nuovo header.
- ***HTTP è stateless ma non session less***: non c'è un link tra due request inoltrate successivamente sulla stessa connessione. Per mantenere l'interazione tra diverse pagine coerente, gli HTTP cookies permettono di utilizzare sessioni stateful: utilizzando l'estensibilità degli header, i cookies vengono aggiunti al workflow, garantendo la creazione di una sessione su ogni request per condividere lo stesso contenuto o lo stesso stato.
- ***HTTP e le connessioni***: HTTP si affida allo standard TCP, che è connection-based. Prima che un client e un server possano scambiarsi request/response, devono prima stabilire una connessione TCP. Di default, HTTP/1.0 apre una connessione TCP diversa per ogni coppia request/response, che è però meno efficiente dell' avere una singola connessione quando ci sono molteplici request in successione. Per ovviare a questo difetto, HTTP/1.1 introduce il pipelining a la connessione persistente: la connessione TCP può essere parzialmente controllata usando il connection header. HTTP/2 invece multiplexa messaggi su una singola connessione, mantenendo la connessione efficiente.

Features controllabili con HTTP

- ***Caching***: il server può istruire i proxies e i client su cosa mettere in cache e per quanto. Il client può istruire le cache dei proxies intermedi ad ignorare il documento salvato.
- ***Allenta il vincolo originale***: per prevenire invasioni di privacy, i web browser impongono una stretta separazione tra siti web. Solo le pagine con la stessa origine possono accedere alle informazioni di una pagina web. Essendo questo vincolo un problema per il server, HTTP può allentarlo dal lato server, permettendo ad un documento di diventare un patchwork di informazioni provenienti da diversi domini.
- ***Autenticazione***: alcune pagine potrebbero essere protette in modo da permettere solo ad alcuni specifici utenti di accedervi. HTTP può fornire un'autenticazione di base con degli header o settando una sessione con i cookies
- ***Proxy e tunneling***: server e client sono spesso situati nell' intranet e nascondono il loro vero indirizzo IP da altri computer. Le requests HTTP passano attraverso i proxy per attraversare questa barriera di rete. Non tutti i proxy sono proxy HTTP.
- ***Sessione***: usare i cookie HTTP permette di collegare le request allo stato del

server. Ciò crea una sessione, nonostante HTTP sia di base stateless.

Flusso HTTP

Quando un client vuole comunicare con un server, sia server finale o in proxy intermedio, esegue i seguenti passaggi:

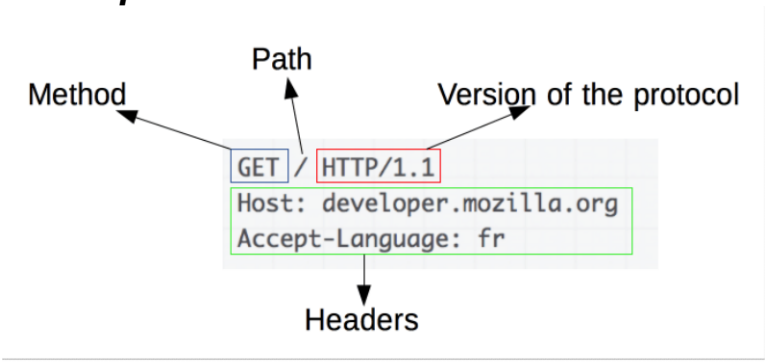
1. ***Apre una connessione TCP***: la connessione viene utilizzata per inviare una o più request e ricevere una response. Il client può aprire una nuova connessione, utilizzarne una già esistente o aprire molteplici connessioni TCP verso il server.
2. ***Invia un messaggio HTTP***: i messaggi HTTP (prima di HTTP/2) sono messaggi leggibili. I messaggi HTTP/2 invece vengono incapsulati in frame, rendendone impossibile la lettura.
3. ***Legge la response*** ricevuta dal server.
4. ***Chiude la connessione***, o la riutilizza per altre request.

Se il pipelining è attivo, possono venire mandate più request senza aspettare che la prima response venga totalmente ricevuta.

Messaggi HTTP

I messaggi HTTP, in HTTP/1.1 e precedenti, sono leggibili dall'uomo. In HTTP/2, questi messaggi sono incorporati in una struttura binaria (frame), permettendo così l' ottimizzazione come la compressione degli header e il multiplexing. Anche se solo una parte del messaggio originale viene inviata in questa versione, la semantica di ogni messaggio rimane invariate e il client ricostituisce (virtualmente) la richiesta HTTP/1.1 originale. È utile quindi comprendere i messaggi TTP/2 in formato HTTP/1.1

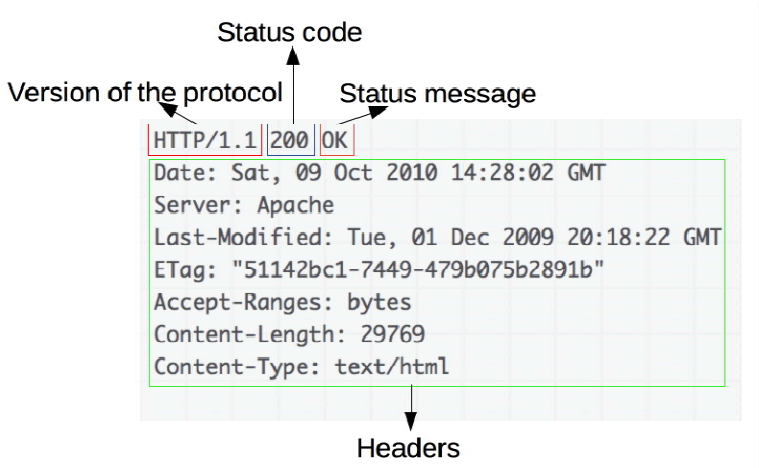
Requests



Una request HTTP si compone di:

- Method: definisce il tipo di operazione che il client vuole eseguire. (GET/POST/HEAD/OPTIONS).
- Path: l'URL della risorsa privato degli elementi di contesto, come il protocollo, il dominio o la porta TCP
- Versione del protocollo
- Headers opzionali: forniscono al server maggiori informazioni.
- Body: per alcuni metodi come POST che contiene le risorse inviate

Responses



Una response si compone di:

- Versione di protocollo HTTP che segue
- Status code: indica se la request è riuscita o no, e perché.
- Status message: piccola descrizione dello status code
- Headers: come nella request.
- Body opzionale: contiene le risorse acquisite.