Verifica di TPSIT, classe 5^A ROB.

1. Descrivi quelle che, a tuo motivato giudizio, sono le caratteristiche essenziali di HTTP, che lo hanno reso un protocollo così diffuso e utilizzabile anche in applicazioni più generiche della semplice navigazione web.

Penso che si sia diffuso molto perchè, al momento della sua uscita era davvero comodo in quanto consente di allegare facilmente qualsiasi file al messaggio http. Oltre a questo ha portato la grande comodità degli header personalizzati che consentono di scambiarsi variabili. Una così una semplice gestione da parte di client e server e una grande ottimizzazione del lavoro.

1. Il sito web avente nome host “www.mymountain.it” ospita un sito web accessibile tramite la pagina index.html riportata sotto:

**<!DOCTYPE *html*>**

**<html>**

**<head>**

**<meta *charset*="utf-8">**

**<title>My mountains</title>**

**</head>**

**<body>**

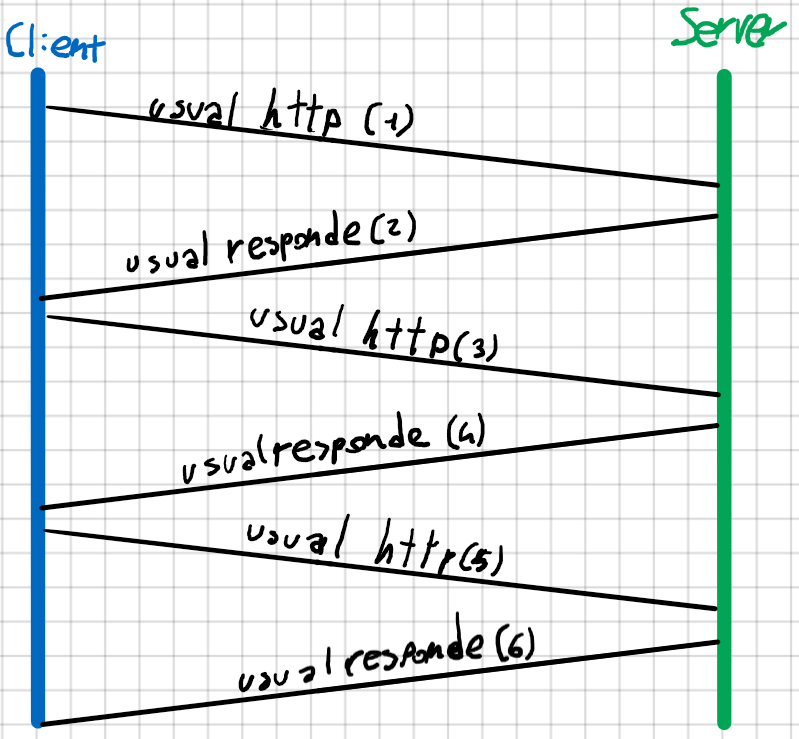
**<h2>Italian Mountain Trip</h2>**

**<img border="0" *src*="**[**http://my.cdn.com/matterhorn.jpg**](http://my.cdn.com/matterhorn.jpg)**">**

**</body>**

**</html>**

Disegnare un sequence diagram per rappresentare lo scambio di messaggi HTTP completo che avviene tra browser e server, a partire dal momento in cui l’utente digita la URL: “www.mymountain.it/index.html” nella barra degli indirizzi, sino alla fine della comunicazione.



1:

GET [www.mymountain.it/index.html](http://www.mymountain.it/index.html) HTTP/1.1

Host: …

2.

HTTP/1.1 200 OK (potrebbe rispondere numeri relativi ad errori)

Date: ….

Server: …

Last-Modified: …

Cookie: … (Se il sito li supporta)

Content-Lenght: len(data)

Content-Type: text/html

Data (file html)

Possibili risposte:

HTTP/1.1 3xx(300) Redirection

HTTP/1.1 4xx(400) Error del Client

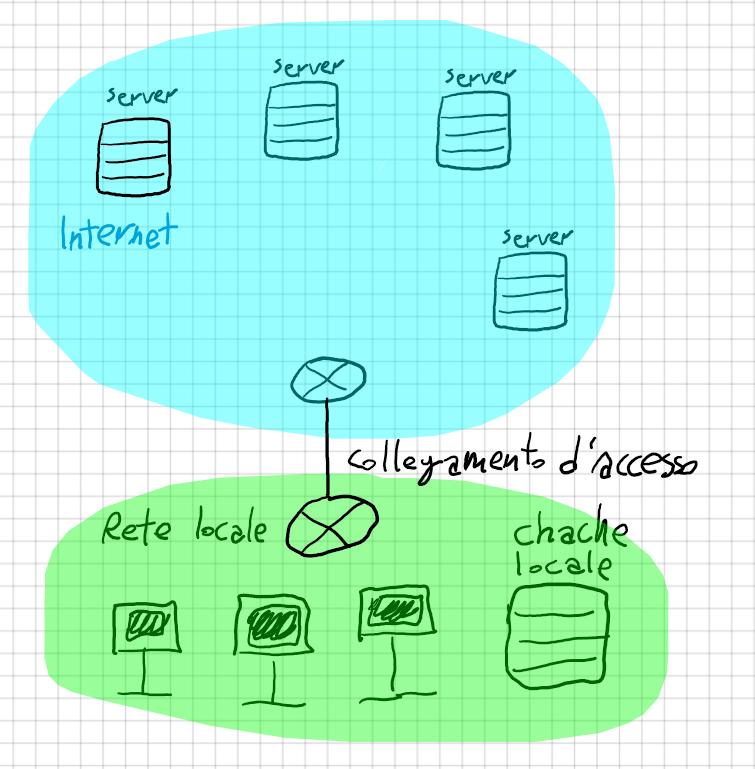
HTTP/1.1 5xx(500) Error del server

Anche i campi di Header possono cambiare, ne esistono molti già preimpostati.

3.-4. (ecc…)

Il client ora può effettuare qualsiasi operazione di GET / HEAD / POST / PUT e riceverà in base alla richiesta la risposta dal Server. (in questo caso ogni volta il client deve reinviare il valore del cookie)

1. Immagina un caso reale di navigazione web, con cache sul browser attiva: rappresenta lo scambio di messaggi HTTP, tramite sequence diagram, nel caso in cui l’oggetto richiesto dal browser sia già presente nella cache locale e inoltre esso non abbia subito modifiche sul server. Indica tutti i campi di header necessari!

****

Il Caching è un'operazione che consente di ottimizzare le prestazioni di internet. Salva le pagine richieste per riutilizzarle in seguito senza doverle nuovamente richiederle al server. è efficiente se visitiamo spesso una pagina. Il Caching può essere fatto dal proxy o dal browser.

Il browser può mantenere una cache delle pagine visitate ed è personalizzabile dall’utente. Ci sono vari meccanismi per mantenere la pagina in modo che non ne venga mantenuta una obsoleta. (Mantenuta per tot tempo o con un campo “Expires” che indica per quanto tempo è valida.

Il proxy ha una memoria interna o esterna (come nel disegno) per mantenere le copie delle pagine visitate. Il browser può essere configurato per richiedere alla cache del proxy e così tutte le richieste di esso passano di lì.

Se la cache contiene l’oggetto lo restituisce altrimenti va a prenderlo su internet.

Il Caching velocizza la rete perchè solitamente la connessione più lenta è quella con l’esterno. il Ritardo totale +è uguale al Ritardo di internet (del server, Lunghissimo -secondi) + Ritardo all’accesso (punto di accesso a internet, Breve -millisecondi) + Ritardo della LAN (Brevissimo -millisecondi). Quindi sostituendo internet con la cache le prestazioni aumentano notevolmente.

ESEMPIO:

Il client chiede la pagina alla cache.

GET page.html HTTP/1.1

Host: [www.google.it](http://www.google.it)

**CASO 1: La cache ha l’oggetto:**

(verifica che non sia scaduto: spiegazione sotto)

HTTP/1.1 200 OK

Date: ….

Server: …

Last-Modified: …

Expires: ...

Cookie: … (Se il sito li supporta)

Content-Lenght: len(data)

Content-Type: text/html

**CASO 2: La cache non ha la pagina salvata:**

La cache gira la richiesta HTTP al server che risponde alla cache.

Rientriamo così nel CASO 1 perchè ora la cache ha la risposta

**verifica che non sia scaduto:**

Richede al server solo se l’ultima modifica corrisponde a quella salvata nella sua cache (meno passaggio di dati e quindi più rapido) se risponde affermativamente allora la cache invia al client la sua pagina altrimenti la richiede al server come nel CASO 2.

**NOTA: per ognuno dei punti 2 e 3 è richiesta precisione nella scrittura di richieste/risposte HTTP.**