Verso un Web più veloce

Il web così come lo conosciamo noi oggi, risulta radicalmente differente da quando, nel 1990, l'implementazione del protocollo HTTP e dei primi browser testuali, fecero la loro comparsa negli istituti di ricerca prima e su internet poi.

Seppur le basi tecnologiche del web siano rimaste grosso modo invariate, questo si è progressivamente arricchito di funzionalità che lo hanno portato ad essere un ambiente interattivo, altamente multimediale e soprattutto sociale.

Come detto, le tecnologie alla base del web sono rimaste fondamentalmente le stesse: IP, TCP ed HTTP.
A questo punto risulta naturale domandarsi se queste tecnologie progettate decadi fa, siano in grado, nel migliore dei modi, di far fronte alle esigente di un web moderno e dinamico.

Per rispondere a questa domanda sono nate ricerche e progetti con lo scopo di investigare i limiti dell'attuale architettura e di valutarne o svilupparne altre qualora queste possano portare benefici alla rete in se ed ovviamente agli utenti.

Il progetto che sembra essere maggiormente promettente allo stato attuale è targato Google, e porta il nome di SPDY.

Il protocollo SPDY rientra nel progetto Chromium ed ha il dichiarato obiettivo di un web più veloce, funzionale e sicuro.

“”

As part of the “Let's make the web faster” initiative, we are experimenting with alternative protocols to help reduce the latency of web pages. One of these experiments is SPDY (pronounced "SPeeDY"), an application-layer protocol for transporting content over the web, designed specifically for minimal latency.
“” Google

Il TCP è il generico protocollo di trasporto affidabile, che fornisce garanzia di consegna, soppressione dei duplicati, ordine di consegna, controllo di flusso, prevenzione della congestione e altre caratteristiche di trasporto.
L'HTTP è il protocollo di livello applicativo che definisce la semantica delle richieste e delle risposte.

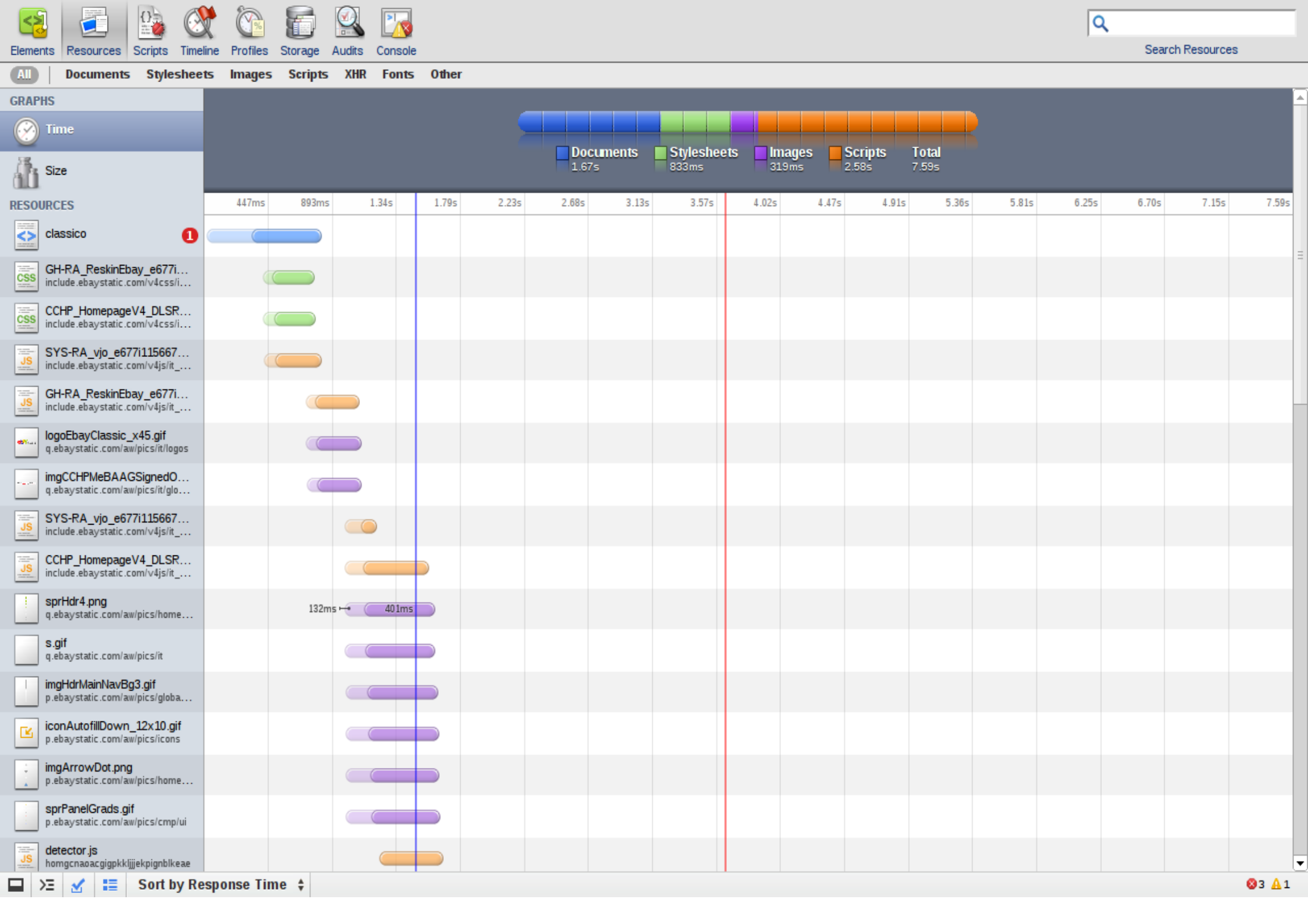
Mentre la Google ritiene che ci possano essere opportunità per migliorare la latenza a livello di trasporto, le prime indagini si sono concentrate sul livello applicativo HTTP.

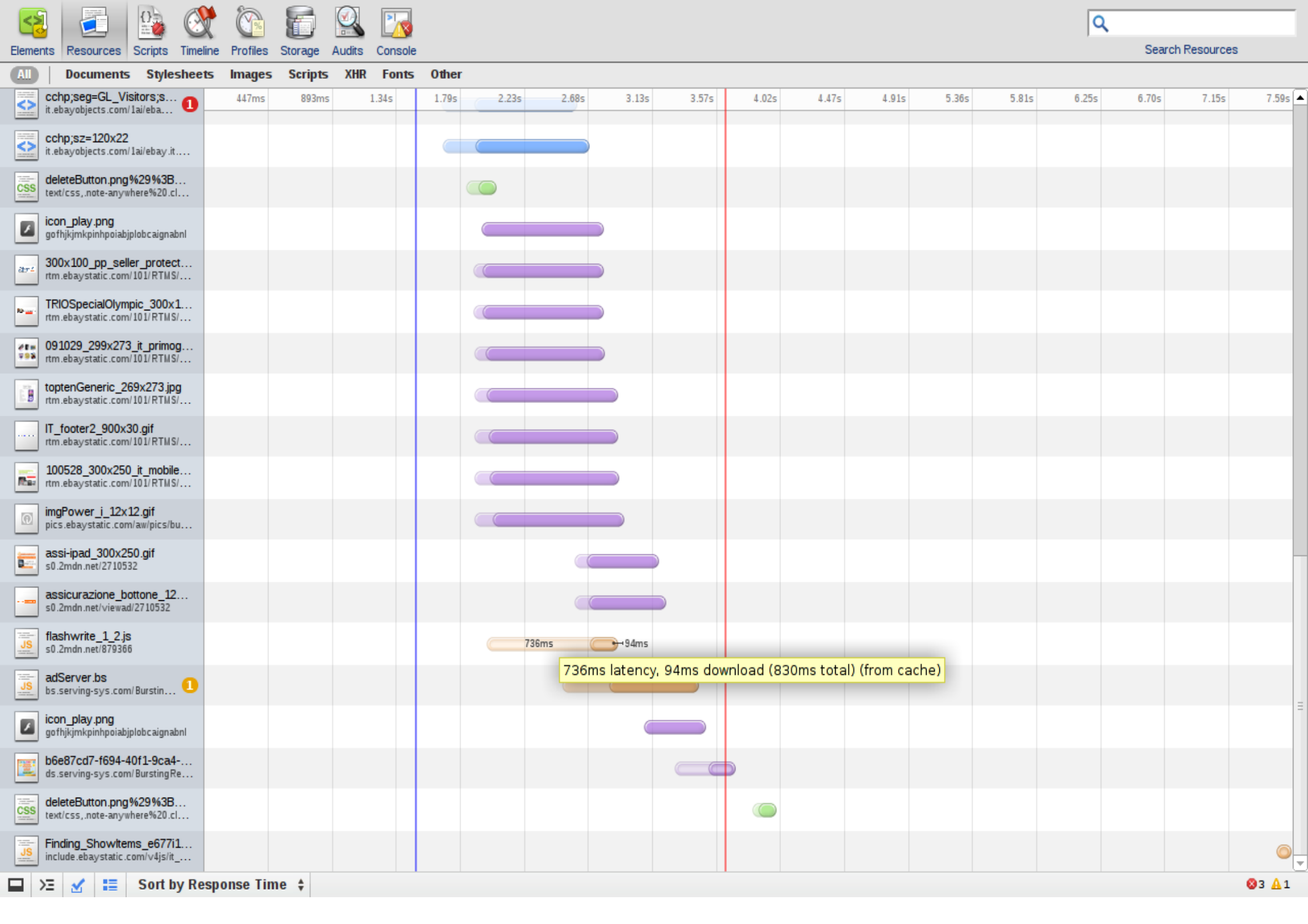
Purtroppo l'HTTP non è stato pensato in particolare per la latenza. Inoltre, le pagine web trasmesse oggi sono significativamente diverse dalle pagine web di 10 anni fa e richiedono miglioramenti dell'HTTP che non si sarebbero potuti immaginare quando l'HTTP fu sviluppato.

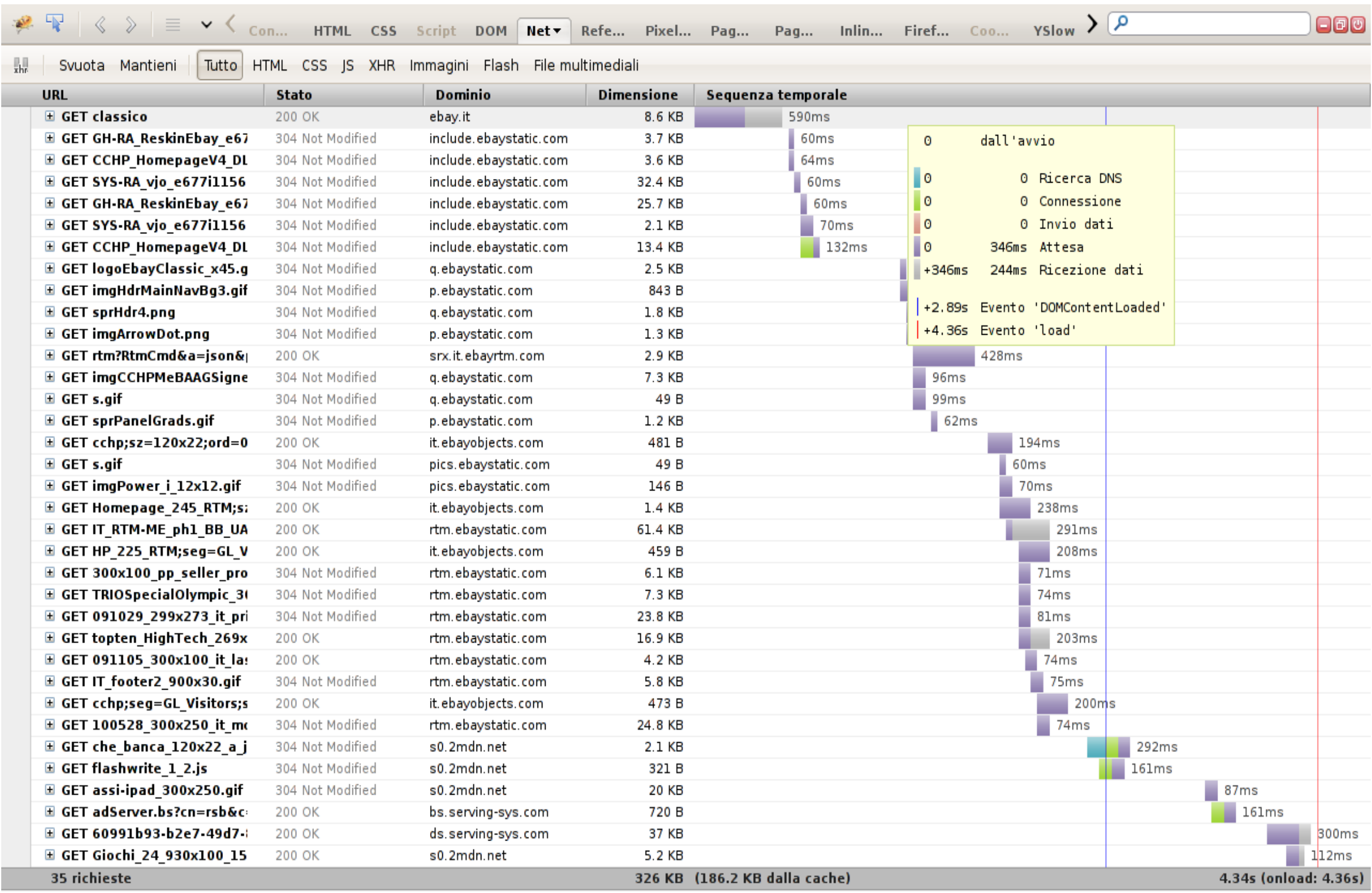
Di seguito sono riportate alcune delle caratteristiche di HTTP che inibiscono prestazioni ottimali:

- Una singola richiesta per connessione.
  Poiché l'HTTP può recuperare solo una risorsa alla volta (il pipelining HTTP aiuta, ma impone l'uso di una sola coda FIFO), un ritardo di 500 ms del server impedisce il riutilizzo del canale TCP per le richieste aggiuntive.
  Il Browser aggira questo problema utilizzando connessioni multiple. Dal 2008, la maggior parte dei browser sono passati da 2 connessioni per dominio a 6.
- Richieste iniziate esclusivamente dal client.
  Nell'HTTP, solo il client può iniziare una richiesta. Anche se il server sa che il cliente ha bisogno di una risorsa, non ha alcun meccanismo per informare il client e deve invece aspettare di ricevere una richiesta per la risorsa da quest'ultimo.
- Headers di richiesta e di risposta non compressi.
  Gli headers della richiesta variano di dimensioni tra circa 200 byte a più di 2 KB. Quando le applicazioni usano più cookie e gli user agents aumentano le funzionalità, le dimensioni tipiche degli headers crescono di 700-800 byte in media.
  Per i modem o le connessioni ADSL, in cui la larghezza di banda di uplink è piuttosto bassa, questa latenza può essere significativa. Ridurre i dati direttamente nell'intestazione potrebbe migliorare la latenza per inviare le richieste.
- Headers ridondanti.
  Inoltre, alcuni headers sono ripetutamente inviati attraverso le richieste sullo stesso canale. Per esempio, le intestazioni come User-Agent, Host e Accept* sono generalmente statici e non hanno bisogno di essere inviati nuovamente.
- Compressione dei dati facoltativa.
  L'HTTP utilizza codifiche di compressione facoltative per i dati. Il contenuto dovrebbe essere sempre inviato in formato compresso.

Quanto affermato risulta particolarmente evidente se prendiamo in esame uno dei siti maggiormente usati sul web: http://www.ebay.it/


Elements
Resources
Scripts
Timeline
Profiles
Storage
Audits
Console
Search Resources
All
Documents
Stylesheets
Images
Scripts
XHR
Fonts
Other
GRAPHS
Time
Size
RESOURCES
Documents 1.67s
Stylesheets 833ms
Images 319ms
Scripts 2.58s
Total 7.59s
447ms
893ms
1.34s
1.79s
2.23s
2.68s
3.13s
3.57s
4.02s
4.47s
4.91s
5.36s
5.81s
6.25s
6.70s
7.15s
7.59s
classico
GH-RA_ReskinEbay_e677i...
include.ebaystatic.com/v4css/i...
CCHP_HomepageV4_DLSR...
include.ebaystatic.com/v4css/i...
SYS-RA_vjo_e677i115667...
include.ebaystatic.com/v4js/it_...
GH-RA_ReskinEbay_e677i...
include.ebaystatic.com/v4js/it_...
logoEbayClassic_x45.gif
q.ebaystatic.com/aw/pics/it/logos
imgCCHPMeBAAGSignedO...
q.ebaystatic.com/aw/pics/it/glo...
SYS-RA_vjo_e677i115667...
include.ebaystatic.com/v4js/it_...
CCHP_HomepageV4_DLSR...
include.ebaystatic.com/v4js/it_...
sprHdr4.png
q.ebaystatic.com/aw/pics/home...
132ms
401ms
s.gif
q.ebaystatic.com/aw/pics/it
imgHdrMainNavBg3.gif
p.ebaystatic.com/aw/pics/globa...
iconAutofillDown_12x10.gif
p.ebaystatic.com/aw/pics/icons
imgArrowDot.png
p.ebaystatic.com/aw/pics/home...
sprPanelGrads.gif
p.ebaystatic.com/aw/pics/cmp/ui
detector.js
homgcnaoacgigpkkljjjekpignblkeae
Sort by Response Time
3
1


Elements
Resources
Scripts
Timeline
Profiles
Storage
Audits
Console
Search Resources
All
Documents
Stylesheets
Images
Scripts
XHR
Fonts
Other
447ms
893ms
1.34s
1.79s
2.23s
2.68s
3.13s
3.57s
4.02s
4.47s
4.91s
5.36s
5.81s
6.25s
6.70s
7.15s
7.59s
cchp;seg=GL_Visitors;s...
it.ebayobjects.com/1ai/eba...
cchp;sz=120x22
it.ebayobjects.com/1ai/ebay.it....
deleteButton.png%29%3B...
text/css,.note-anywhere%20.cl...
icon_play.png
gofhjkjmkpinhpoiabjplobcaignabnl
300x100_pp_seller_protect...
rtm.ebaystatic.com/101/RTMS/...
TRIOSpecialOlympic_300x1...
rtm.ebaystatic.com/101/RTMS/...
091029_299x273_it_primog...
rtm.ebaystatic.com/101/RTMS/...
toptenGeneric_269x273.jpg
rtm.ebaystatic.com/101/RTMS/...
IT_footer2_900x30.gif
rtm.ebaystatic.com/101/RTMS/...
100528_300x250_it_mobile...
rtm.ebaystatic.com/101/RTMS/...
imgPower_i_12x12.gif
pics.ebaystatic.com/aw/pics/bu...
assi-ipad_300x250.gif
s0.2mdn.net/2710532
assicurazione_bottone_12...
s0.2mdn.net/viewad/2710532
flashwrite_1_2.js
s0.2mdn.net/879366
736ms
94ms
736ms latency, 94ms download (830ms total) (from cache)
adServer.bs
bs.serving-sys.com/Burstin...
icon_play.png
gofhjkjmkpinhpoiabjplobcaignabnl
b6e87cd7-f694-40f1-9ca4-...
ds.serving-sys.com/BurstingRe...
deleteButton.png%29%3B...
text/css,.note-anywhere%20.cl...
Finding_ShowItems_e677i1...
include.ebaystatic.com/v4js/it_...
Sort by Response Time
3
1


Con... HTML CSS Script DOM Net Refe... Pixel... Pag... Pag... Inlin... Firef... Coo... YSlow
Svuota Mantieni Tutto HTML CSS JS XHR Immagini Flash File multimediali
URL Stato Dominio Dimensione Sequenza temporale
GET classico 200 OK ebay.it 8.6 KB 590ms
GET GH-RA_ReskinEbay_e67 304 Not Modified include.ebaystatic.com 3.7 KB 60ms
GET CCHP_HomepageV4_DL 304 Not Modified include.ebaystatic.com 3.6 KB 64ms
GET SYS-RA_vjo_e677i1156 304 Not Modified include.ebaystatic.com 32.4 KB 60ms
GET GH-RA_ReskinEbay_e67 304 Not Modified include.ebaystatic.com 25.7 KB 60ms
GET SYS-RA_vjo_e677i1156 304 Not Modified include.ebaystatic.com 2.1 KB 70ms
GET CCHP_HomepageV4_DL 304 Not Modified include.ebaystatic.com 13.4 KB 132ms
GET logoEbayClassic_x45.g 304 Not Modified q.ebaystatic.com 2.5 KB
GET imgHdrMainNavBg3.gif 304 Not Modified p.ebaystatic.com 843 B
GET sprHdr4.png 304 Not Modified q.ebaystatic.com 1.8 KB
GET imgArrowDot.png 304 Not Modified p.ebaystatic.com 1.3 KB
GET rtm?RtmCmd&a=json&| 200 OK srx.it.ebayrtm.com 2.9 KB 428ms
GET imgCCHPMeBAAGSigne 304 Not Modified q.ebaystatic.com 7.3 KB 96ms
GET s.gif 304 Not Modified q.ebaystatic.com 49 B 99ms
GET sprPanelGrads.gif 304 Not Modified p.ebaystatic.com 1.2 KB 62ms
GET cchp;sz=120x22;ord=0 200 OK it.ebayobjects.com 481 B 194ms
GET s.gif 304 Not Modified pics.ebaystatic.com 49 B 60ms
GET imgPower_i_12x12.gif 304 Not Modified pics.ebaystatic.com 146 B 70ms
GET Homepage_245_RTM;s: 200 OK it.ebayobjects.com 1.4 KB 238ms
GET IT_RTM-ME_ph1_BB_UA 200 OK rtm.ebaystatic.com 61.4 KB 291ms
GET HP_225_RTM;seg=GL_V 200 OK it.ebayobjects.com 459 B 208ms
GET 300x100_pp_seller_pro 304 Not Modified rtm.ebaystatic.com 6.1 KB 71ms
GET TRIOSpecialOlympic_3(304 Not Modified rtm.ebaystatic.com 7.3 KB 74ms
GET 091029_299x273_it_pri 304 Not Modified rtm.ebaystatic.com 23.8 KB 81ms
GET topten_HighTech_269x 200 OK rtm.ebaystatic.com 16.9 KB 203ms
GET 091105_300x100_it_la: 200 OK rtm.ebaystatic.com 4.2 KB 74ms
GET IT_footer2_900x30.gif 304 Not Modified rtm.ebaystatic.com 5.8 KB 75ms
GET cchp;seg=GL_Visitors;s 200 OK it.ebayobjects.com 473 B 200ms
GET 100528_300x250_it_m(304 Not Modified rtm.ebaystatic.com 24.8 KB 74ms
GET che_banca_120x22_a_j 304 Not Modified s0.2mdn.net 2.1 KB 292ms
GET flashwrite_1_2.js 304 Not Modified s0.2mdn.net 321 B 161ms
GET assi-ipad_300x250.gif 304 Not Modified s0.2mdn.net 20 KB 87ms
GET adServer.bs?cn=rsb&c: 200 OK bs.serving-sys.com 720 B 161ms
GET 60991b93-b2e7-49d7-: 200 OK ds.serving-sys.com 37 KB 300ms
GET Giochi_24_930x100_15 200 OK s0.2mdn.net 5.2 KB 112ms
0 dall'avvio
0 0 Ricerca DNS
0 0 Connessione
0 0 Invio dati
0 346ms Attesa
+346ms 244ms Ricezione dati
+2.89s Evento 'DOMContentLoaded'
+4.36s Evento 'load'
35 richieste 326 KB (186.2 KB dalla cache) 4.34s (onload: 4.36s)

Il test è stato effettuato su di una connessione ADSL domestica tipo.

Il sito impiega ben 4 secondi e mezzo per caricare, e come si evince dalle immagini, la latenza nell'instaurazione di una nuova connessione ed il conseguente invio della richiesta e della risposta, in alcuni casi supera di gran lunga il tempo richiesto allo scaricamento dei dati stessi.

Nel caso peggiore si è arrivati ad una latenza 7 volte superiore al tempo di scaricamento dei dati su una risorsa JavaScript che per natura è blocking e che quindi non permette il download di altre risorse contemporaneamente.

Senza contare che la latenza media sembra incidere per un fattore del 20% sul tempo di caricamento dell'intero sito.

Alla luce di questo semplice test, risulta evidente come il tentativo da parte di Google di ridurre la latenza delle connessioni HTTP sia tutt'altro che vano.

Da rilevare come l'utilizzo di 6 connessioni concorrenti diminuisca, e anche di molto, il tempo di caricamento del sito. Questa comunque resta una soluzione paliativa e tutt'altro che ottimale, considerato che i siti moderni raggiungono anche il numero di 50 risorse esterne servite mediante richieste GET.

Obiettivi di SPDY

Il progetto SPDY definisce ed implementa uno protocollo nello strato applicativo per il web che riduce grandemente la latenza.

Gli altri obiettivi primari sono:

- Minimizzare la complessità di utilizzo. SPDY usa TCP come strato di trasporto sottostante, il che si traduce in nessun cambio alla infrastruttura di rete esistente.
- Evitare la necessità di alcun cambiamento ai contenuti da parte dell'autore del sito. Gli unici cambiamenti richiesti per supportare SPDY sono collocati nell'user agent del client e nell'application server.
- Sviluppare il protocollo in partnership con la community open-source e gli specialisti del settore.

Alcuni degli obiettivi tecnici sono:

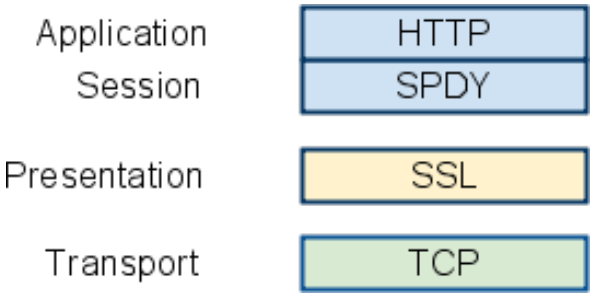
- Permettere molteplici richieste HTTP concorrenti su una singola sessione TCP.
- Ridurre la banda attualmente utilizzata dall'HTTP comprimendo gli headers ed eliminando quelli non necessari.
- Definire un protocollo che sia facile da implementare ed efficiente lato server. La speranza è di ridurre la complessità dell'HTTP eliminando i casi limite e definendo formati facilmente parsabili.
- Rendere SSL il sottostante protocollo di trasporto, per una migliore sicurezza e per garantire la compatibilità con le esistenti infrastrutture di rete. Anche se l'SSL

introduce una penalità in termini di latenza, il futuro a lungo termine del web dipende da una connessione di rete sicura.
- Permettere al server di iniziare le comunicazioni col client e inviare i dati al client non appena possibile.

SPDY design e funzionalità

SPDY aggiunge uno strato di sessione al di sopra del'SSL che permette richieste multiple e concorrenti e gli interleaved streams su una singola connessione TCP.

Il formato dei messaggi HTTP GET e POST rimane lo stesso; tuttavia, SPDY specifica un nuovo formato per codificare e trasmettere i dati sulla rete.



Gli streams sono bi-direzionali e possono essere iniziati sia dal cliente che dal server.

SPDY specifica alcune base (sempre abilitate) e altre avanzate (opzionalmente abilitate) funzionalità.

Funzionalità base

- Multiplexed streams

  SPDY permette un numero illimitato di streams concorrenti su di una singola connessione TCP. Poiché le richieste sono interleaved su di un singolo canale, l'efficienza del TCP è di molto superiore: minori richieste di rete devono essere effettuate, e minori pacchetti, ma più densi, sono generati.
- Richieste con priorità

  Anche se gli streams paralleli illimitati risolvono il problema della serializzazione, ne introducono uno nuovo:

  se la banda sul canale è limitata, il client potrebbe bloccare le richieste per paura di saturare il canale. Per superare il problema, SPDY implementa le richieste con priorità:

  il client può richiedere quante risorse vuole al server e assegnare ad ognuna di loro una priorità. Questo modo di operare previene la congestione della rete con risorse non critiche, quando invece richieste ad alta priorità sono in attesa.

- Compressione degli headers HTTP

  SPDY comprime gli headers delle richieste e delle risposte HTTP, risultando in un minor numero di byte, e di conseguenza di pacchetti inviati.

Funzionalità avanzate

In aggiunta SPDY fornisce una funzionalità avanzata: *server-initiated streams*.

I server-initiated streams possono essere utlizzati per consegnare contenuti al client senza la necessità da parte del client di richiederli. Questa opzione può essere configurata dallo sviluppatore web in due modi:

- Server push

  Mediante l'header X-Associated-Content. Questo header informa il client che il server sta inviando una risorsa al client prima che quest'ultimo l'abbia richiesta.

- Server hint

  Mediante l'header X-Subresources. Piuttosto che inviare autonomamente i dati, il server suggerisce al client una risorsa e rimane in attesa di una richiesta da parte di quest'ultimo per la risorsa suggerita.

Risultati preliminari

Table 1: Average page load times for top 25 websites

	DSL 2 Mbps downlink, 375 kbps uplink		Cable 4 Mbps downlink, 1 Mbps uplink	
	Average ms	**Speedup**	**Average ms**	**Speedup**
HTTP	3111.916		2348.188	
SPDY basic multi-domain* connection / TCP	2242.756	27.93%	1325.46	43.55%
SPDY basic single-domain* connection / TCP	1695.72	45.51%	933.836	60.23%
SPDY single-domain + server push / TCP	1671.28	46.29%	950.764	59.51%
SPDY single-domain + server hint / TCP	1608.928	48.30%	856.356	63.53%
SPDY basic single-domain / SSL	1899.744	38.95%	1099.444	53.18
SPDY single-domain + client prefetch / SSL	1781.864	42.74%	1047.308	55.40%

Table 2: Average page load times for top 25 websites by packet loss rate

	Average ms		Speedup
Packet loss rate	HTTP	SPDY basic (TCP)	
0%	1152	1016	11.81%
0.5%	1638	1105	32.54%
1%	2060	1200	41.75%
1.5%	2372	1394	41.23%
2%	2904	1537	47.7%
2.5%	3028	1707	43.63%

Table 3: Average page load times for top 25 websites by RTT

	Average ms		**Speedup**
RTT in ms	**HTTP**	**SPDY basic (TCP)**	
20	1240	1087	12.34%
40	1571	1279	18.59%
60	1909	1526	20.06%
80	2268	1727	23.85%
120	2927	2240	23.47%
160	3650	2772	24.05%
200	4498	3293	26.79%

Conclusioni

I risultati preliminari condotti su 25 fra i più diffusi siti in circolazione mostrano valori veramente molto incoraggianti e vicini al target annunciato da Google di una riduzione del tempo di caricamento del 50%.

Particolarmente interessanti sono i valori ottenuti con packet lose rate elevati, il che fa pensare a grandi miglioramenti in ambito mobile e ultra mobile su reti WiFi, 3G e successive.

Il protocollo SPDY, pur ancora in pesante fase development, si preannuncia come una delle ennesime innovazioni targate Google, che, in un futuro non troppo lontano, rivoluzionerà il modo di fruire il web.

Nel momento in cui si scrive, sono disponibili implementazioni mod_spdy, Java e Python di Application Server SPDY enabled, ed è disponibile un fork del noto browser Chromium con supporto al protocollo “della velocità”.
http://src.chromium.org/viewvc/chrome/trunk/src/net/spdy/
http://src.chromium.org/viewvc/chrome/trunk/src/net/tools/flip_server/
http://github.com/mnot/nbhttp/tree/spdy
http://svn.apache.org/repos/asf/tomcat/trunk/modules/tomcat-lite
http://code.google.com/p/mod-spdy/

Ovviamente SPDY non è l'unico protocollo esistente che serve il web in maniera più efficiente dell'HTTP over TCP. Sono disponibili numerose risorse online fra cui:

Stream Control Transmission Protocol (SCTP) http://www.sctp.org/
HTTP over SCTP http://tools.ietf.org/html/draft-natarajan-http-over-sctp-00
Structured Stream Transport http://pdos.csail.mit.edu/uia/sst/

Ognuna di queste soluzione esistenti risolve uno o più problemi del protocollo HTTP, ma nessuna li risolve tutti, come fa SPDY invece.

Riferimenti

http://www.chromium.org/spdy/spdy-whitepaper
http://tools.ietf.org/html/draft-natarajan-http-over-sctp-00
http://dev.chromium.org/spdy/spdy-protocol
http://dev.chromium.org/spdy
http://tools.ietf.org/html/rfc2616
http://en.wikipedia.org/wiki/HTTP_persistent_connection
http://en.wikipedia.org/wiki/HTTP_pipelining
http://www.mozilla.org/projects/netlib/http/pipelining-faq.html