
# SPDY & QUIC

29/11/2014

# Chi sono?

Carmelo San Giovanni

Presidente di IGLU (Gruppo Linux Udine)

Membro di:

- Associazione Cultura Informatica (AsCI)
- Italian Linux Society (ILS)
- LibreItalia

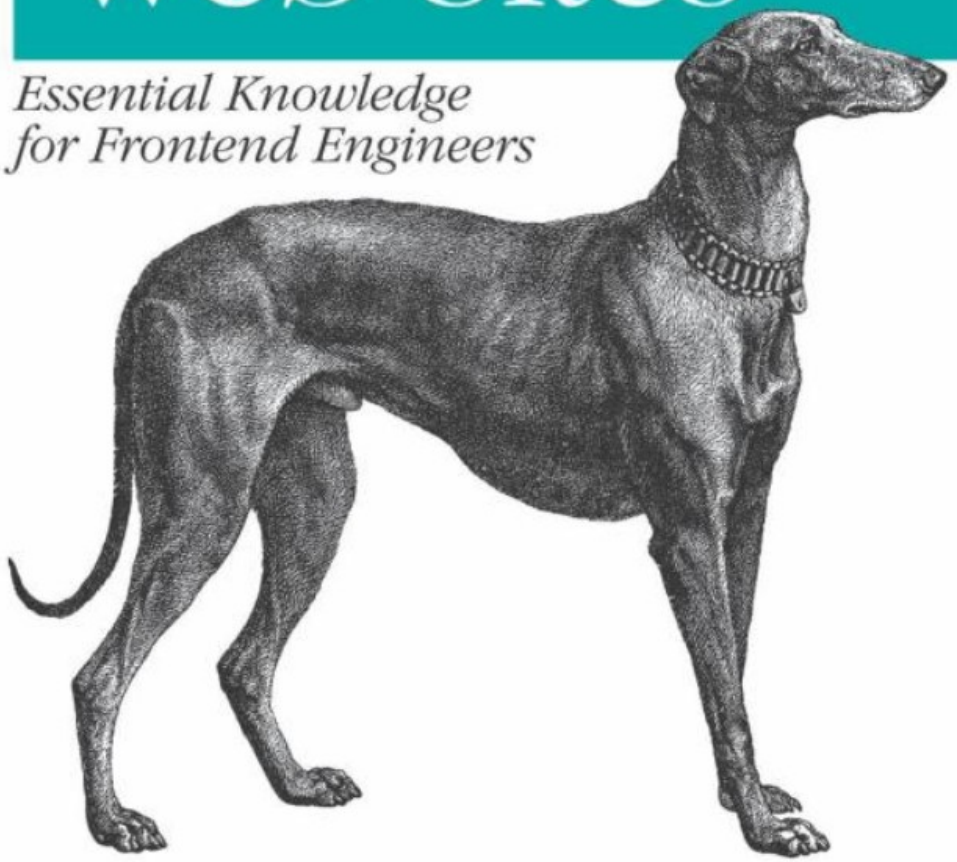


Life's too short to wait page loading

14 Steps to Faster Loading Web Sites

# High Performance Web Sites

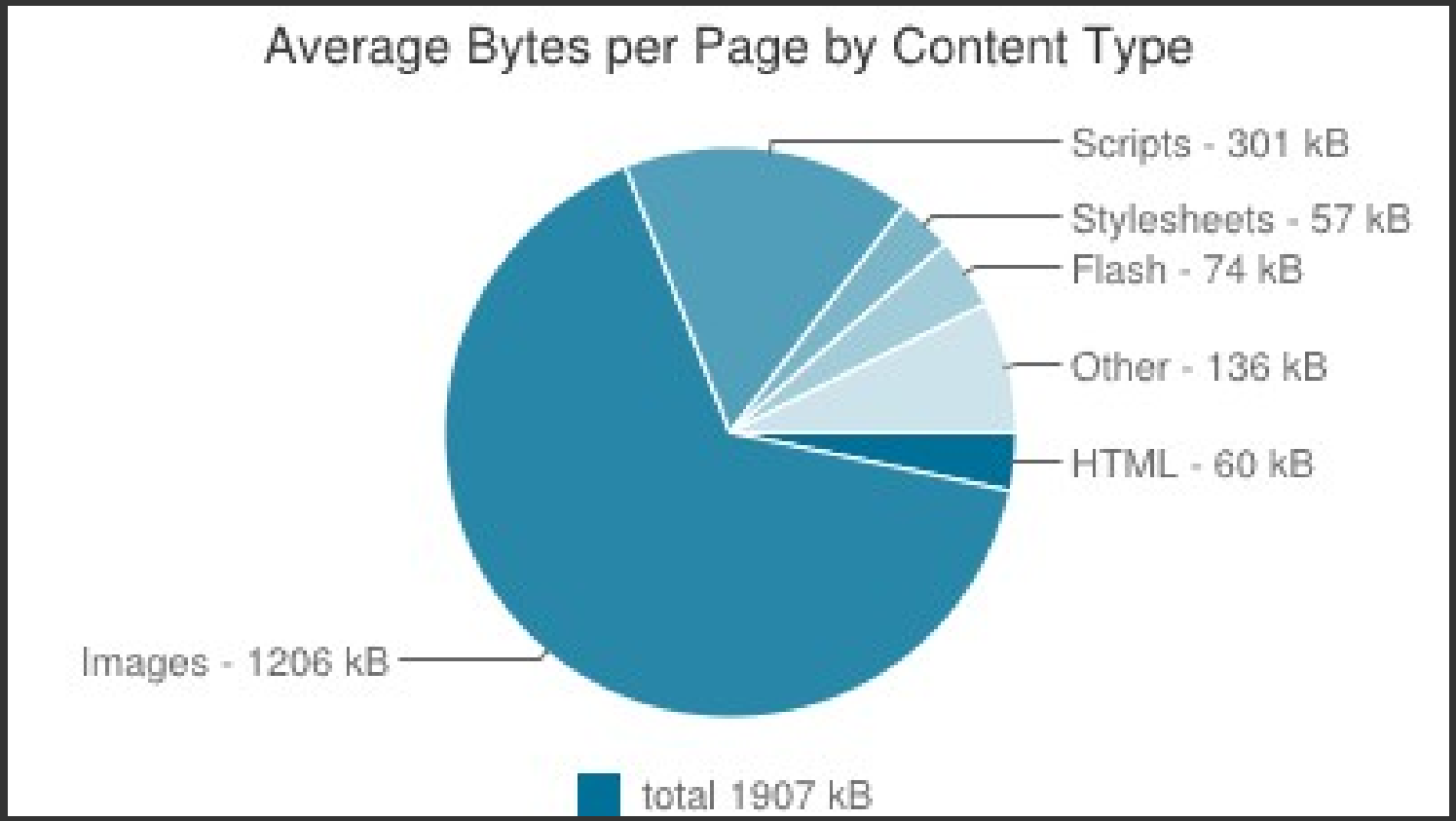
*Essential Knowledge for Frontend Engineers*



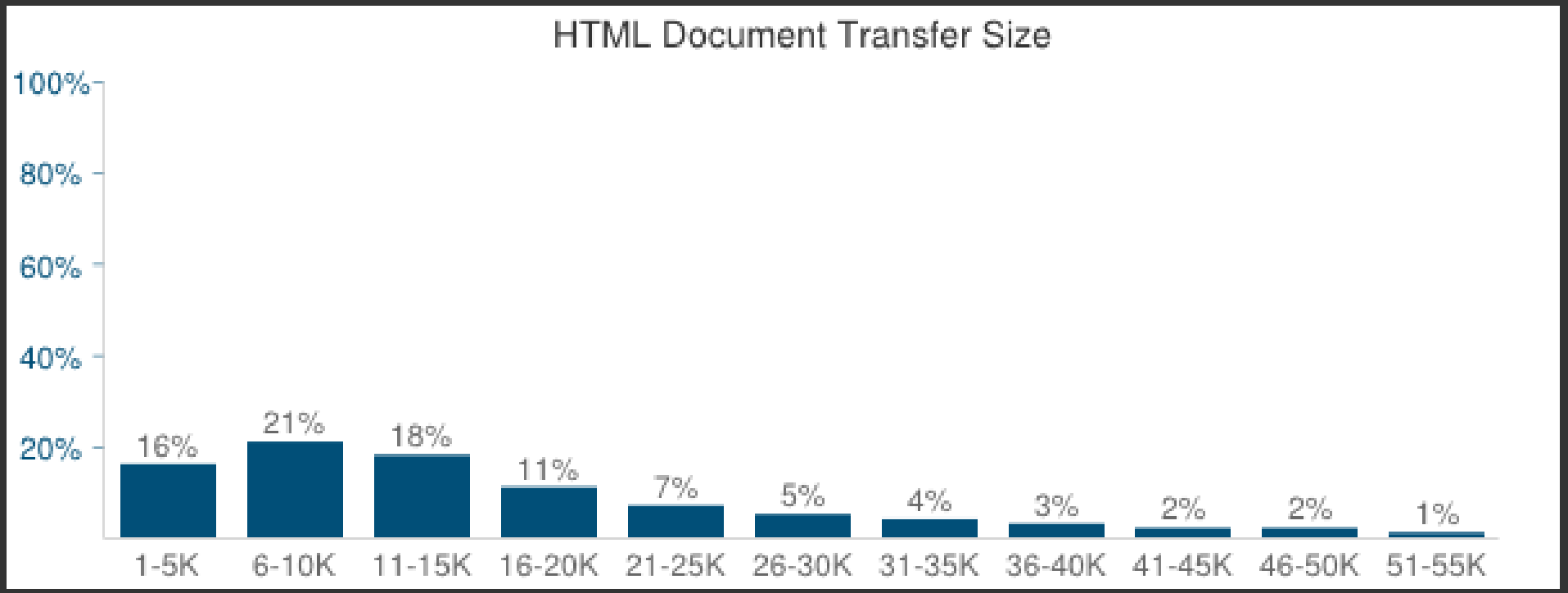
O'REILLY®

*Steve Souders*
*Foreword by Nate Koechley*

Solo circa il 10% del tempo di caricamento di un sito web è dovuto all'html in essa contenuto.




Sorgente: http://httparchive.org

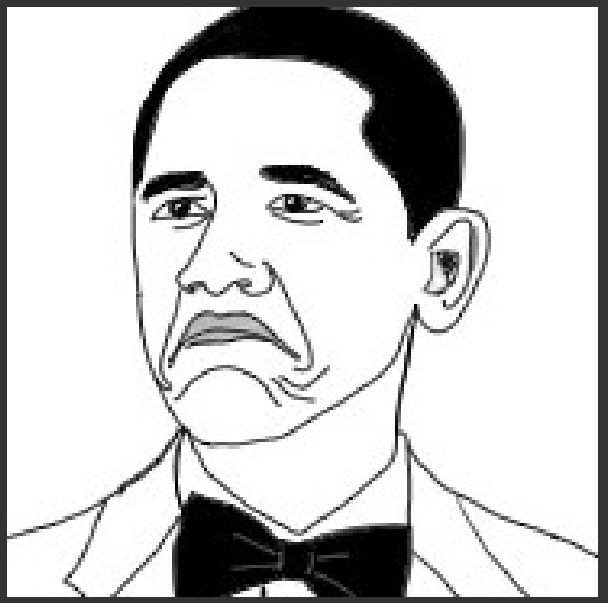



Sorgente: http://httparchive.org

# Ottimizzazioni comuni

- usare mappa immagine;
- usare sprite css;
- combinare (e minimizzare) gli script in un unico file (UglifyJS, YuiCompressor);
- combinare (e minimizzare) i css in un unico file (UglifyCSS, YuiCompressor);
- usare un CDN;
- ottimizzare le impostazioni degli Header Expires;
- attivare la compressione lato server;
- ottimizzare il TTL delle query DNS

Con queste tecniche possiamo risparmiare più del 40% di tempo di caricamento della pagina.



E se non fosse abbastanza?

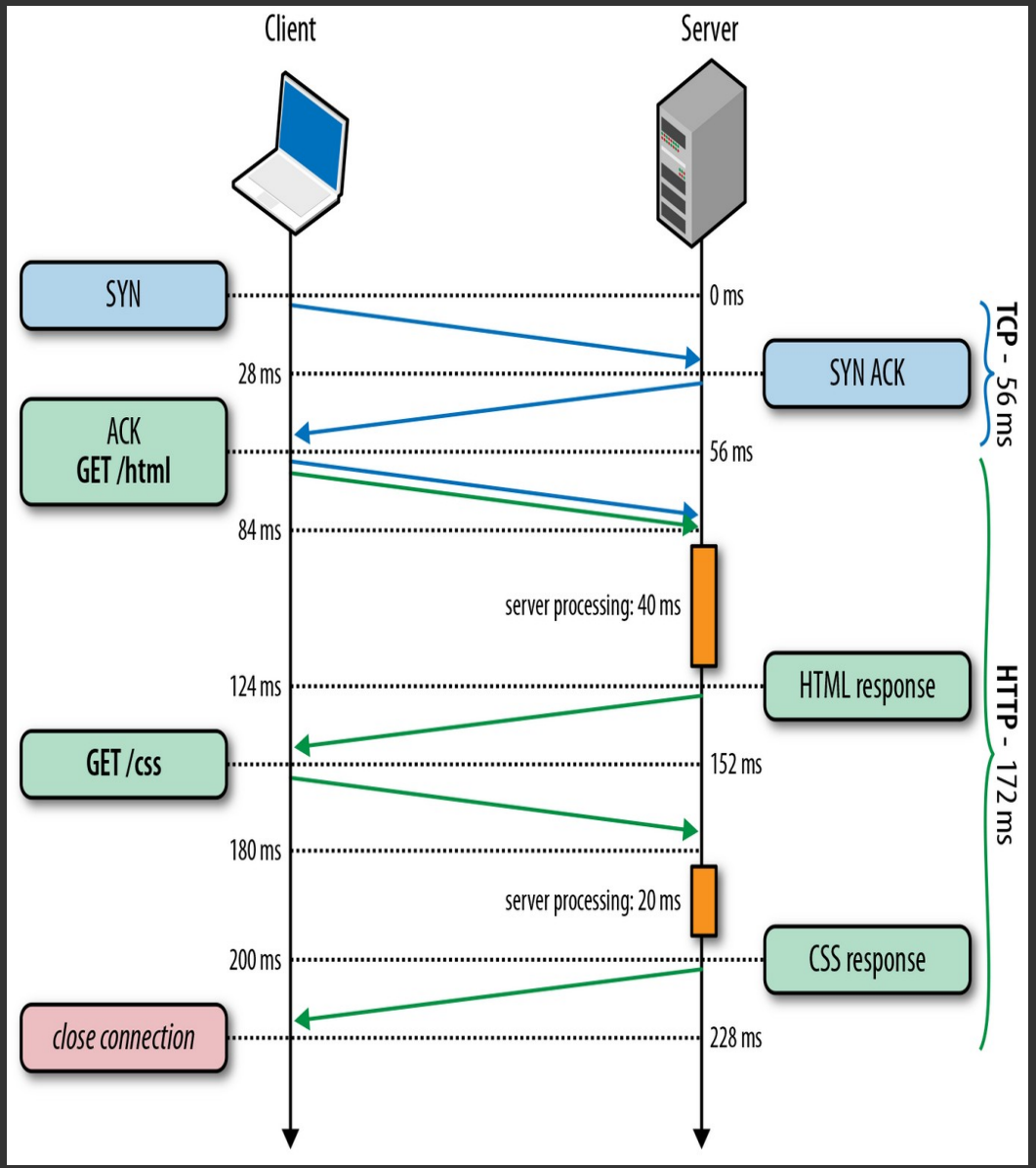
# Dov'è il problema?

Facebook carica oltre 200 oggetti al primo accesso. I siti web di quest'epoca sono troppo complessi?

HTTP riesce a soddisfare una richiesta per connessione.
Il protocollo HTTP 1.1 consiglia di aprire 2 sole connessioni per volta verso lo stesso host.
Molti browser ne aprono fino a 6.

HTTP non comprime le intestazioni anzi, le duplica!
La compressione lato server è disattivata di default.
Solo il client può eseguire la richiesta.

# Oltretutto TCP è lento!



# molto lento!

3Way Handshake
0ms  1)  Host A invia un TCP SYN a Host B
50ms 2) Host B riceve il SYN di A
50ms 3) Host B invia a SYN-ACK
100ms 4) Host A riceve il SYN-ACK di B
100ms 5) Host A invia l'ACK di avvio connessione
150ms 6) Host B riceve l'ACK.

Con un ping medio di 50ms, questa operazione ci costa almeno 150ms!

# Come ovviamo?

# SPDY!

# Cos'è SPDY?

- un protocollo di rete aperto;
- inizialmente sviluppato da Google;
- la base del protocollo HTTP 2.0;
- **un sistema per minimizzare il tempo di caricamento delle pagine.**

# Come ci aiuta?

- riduce drasticamente il tempo di caricamento della pagina;
- aumenta la sicurezza dei dati in transito;
- facile da implementare.

# Come fa a fare tutto ciò?

- trasporta richieste HTTP multiple in una singola sessione TCP (potenzialmente infinite);
- Riduce il consumo di banda comprimendo le intestazioni;
- utilizza connessioni SSL per veicolare le informazioni.

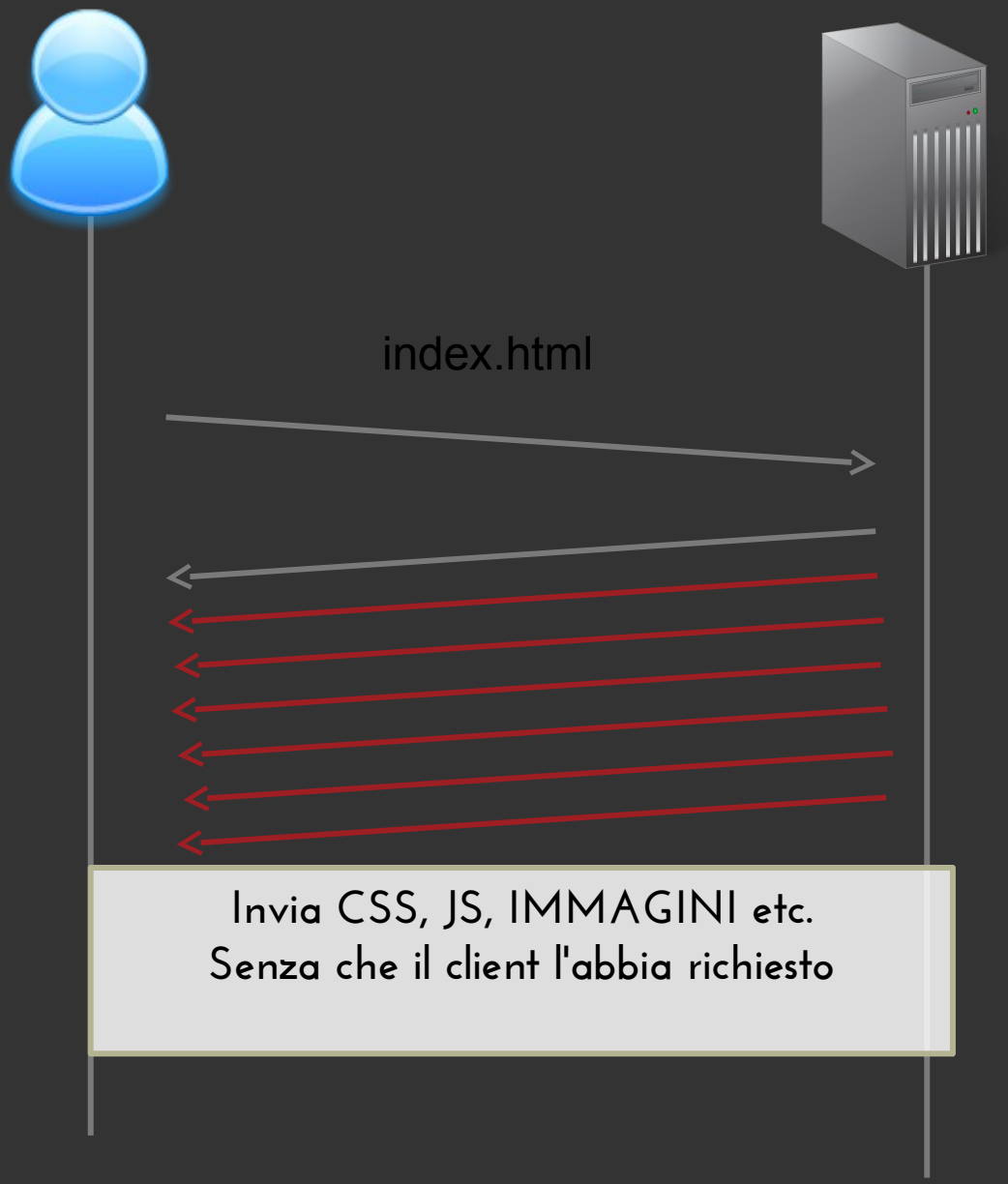
# Quindi?

- quando il client richiede una pagina al server quest'ultimo fornirà subito sia la pagina che tutte le immagini, i fogli di stile, i javascript ed ulteriori files associati TUTTO su una singola connessione TCP ordinandoli per priorità!

# HTTP classico

# SPDY

# Server Push/Hint



# Quindi?

- niente più intestazioni duplicate (vengono inviate solo le differenze);
- tutti i dati in transito vengono compressi e cifrati;
- il server può indicare al client la variazione di contenuti.

# Client supportati

- IE 11 e successivi
- Firefox 13 e successivi
- Chrome 4 e successivi
- Opera 12.1 e successivi
- Android Browsers
- Opera Mobile
- etc...

Verifica se il tuo browser lo supporta all'indirizzo: https://isspdyenabled.com

# Server supportati

- **Apache** 2.2 e superiori con mod_spdy
- **NGINX** 1.5.10 e superiori con ngx_http_spdy_module
- **Jetty Web Server**
- **Node.js (SPDY Server)**
- **etc...**

# Abilitare Apache

Scaricare ed installare il modulo “mod_spdy” da https://developers.google.com/speed/spdy/mod_spdy

Nel file di configurazione di apache o nel file dei singoli vhost aggiungere la direttiva “SpdyEnabled on”.

Riavviare Apache.

E'necessario l'utilizzo di mod_fcgid, piccoli aggiustamenti alla configurazione di questo (come la modifica delle dimensioni delle richiede FcgidMaxRequestLen in fcgid.conf) potrebbero essere indispensabili.

# Abilitare NGINX

E' necessario aprire il file di configurazione del sito web in questione (che deve avere già attivo il supporto per SSL) e variare la stringa:
"*listen 443 ssl;*"

con

"*listen 443 ssl spdy;*"
Salvare e riavviare il demone con il comando service nginx reload

# Ottenere i certificati

In un ambito di produzione sarebbe necessario avere dei certificati digitali firmati da un'apposita autorithy globalmente riconosciuta.

**Qualche consiglio:**

- http://www.ssls.com (certificati a partire da pochi euro)
- https://letsencrypt.org (disponibile solo da luglio 2015)

# Qualche tool utile

- SPDY Indicator (Firefox)
- Speed Tracer (Google Chrome)
- FireBug

# Ma è davvero tutto rosa e fiori?

# Ma è davvero tutto rosa e fiori?

NO!

# Ma è davvero tutto rosa e fiori?

- Su TCP il ritardo o la perdita di un pacchetto si ripercuote su tutti gli stream;
- La prossima riconnessione richiede comunque l'attesa del 3WAY Handshaking e TCP richiede 1,5*RTT mentre SSL richiede 3*RTT

Come ovviamo a questo?

QUIC!

# QUIC

- Usa UDP al posto di TCP;
- i pacchetti persi non influenzano più tutti gli stream perché viene usato uno stream differente per ogni richiesta (ottimo dato che i client mobili tendono a spegnere i dispositivi di comunicazione wireless in modo aggressivo);
- niente handshake quindi 0ms di RTT;
- ovviamente su SSL;

# QUIC

- correzione degli errori a livello di pacchetto, riduzione della latenza per via della correzione che sostistuisce la ritrasmissione del pacchetto;
- generale riduzione del consumo di banda;
- quando il server termina una sessione idle può evitare di notificarla al client per non risvegliare i circuiti radio assopiti;

# QUIC

- utilizza un GUID (Globally Unique Identifier) di connessione composta da un numero randomico generato dal client. In questo modo in caso di roaming la connessione rimane stabile.
- Gli stream sono sequenze di dati bidirezionali e possono essere create sia dal client che dal server

# QUIC: stato attuale

Implementato in Chromium e  Google Chrome dalla versione 28.

(da definirsi ancora come sperimentale)

# Grazie per l'attenzione!

WEB:

AsCI - http://asci.cc.uniud.it

IGLU - http://iglu.cc.uniud.it

E-Mail:

Carmelo San Giovanni: carmelo.sangiovanni@linux.it

AsCI: asci@uniud.it