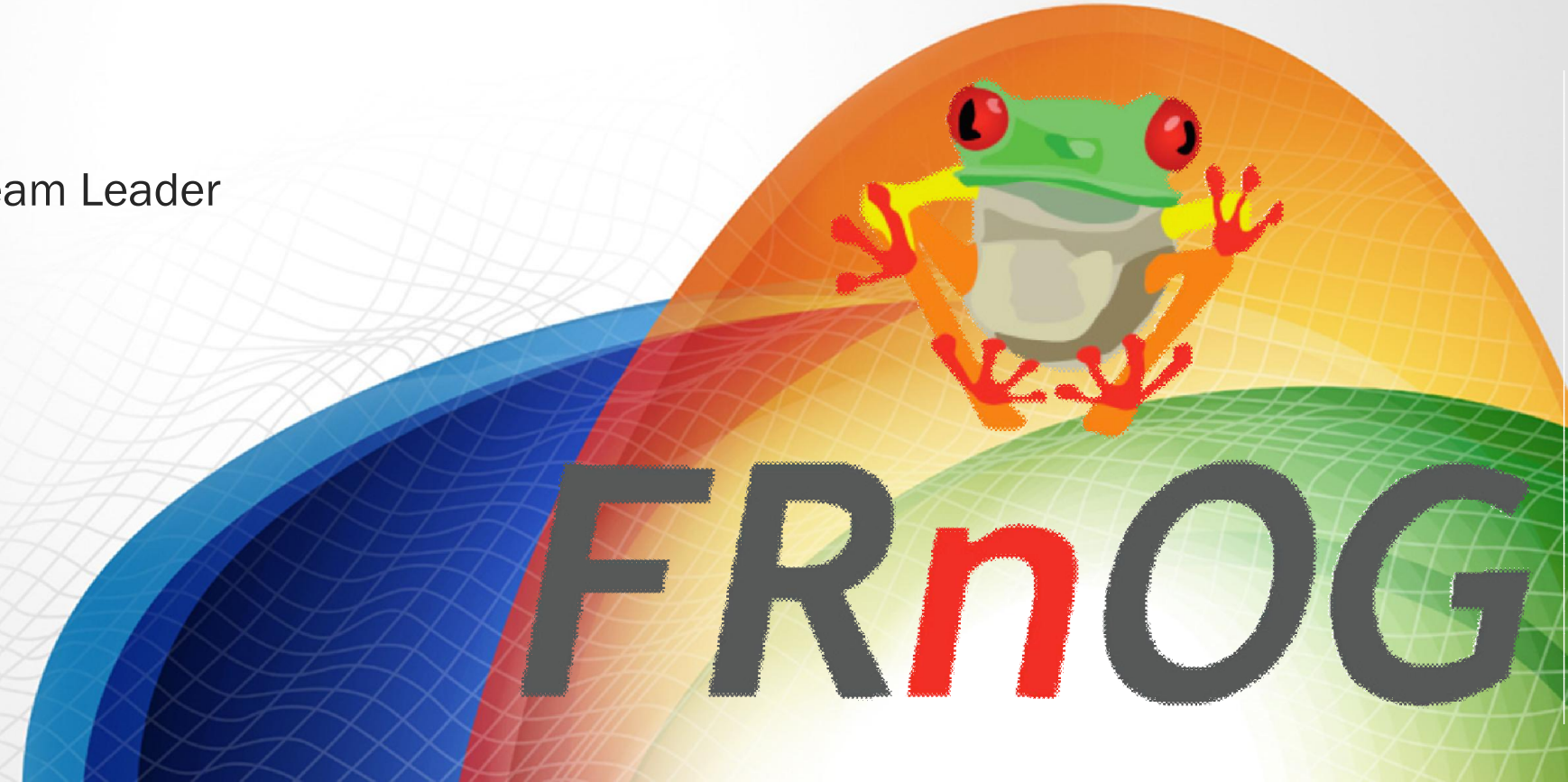
Optimisation TCP & Web

Benjamin David

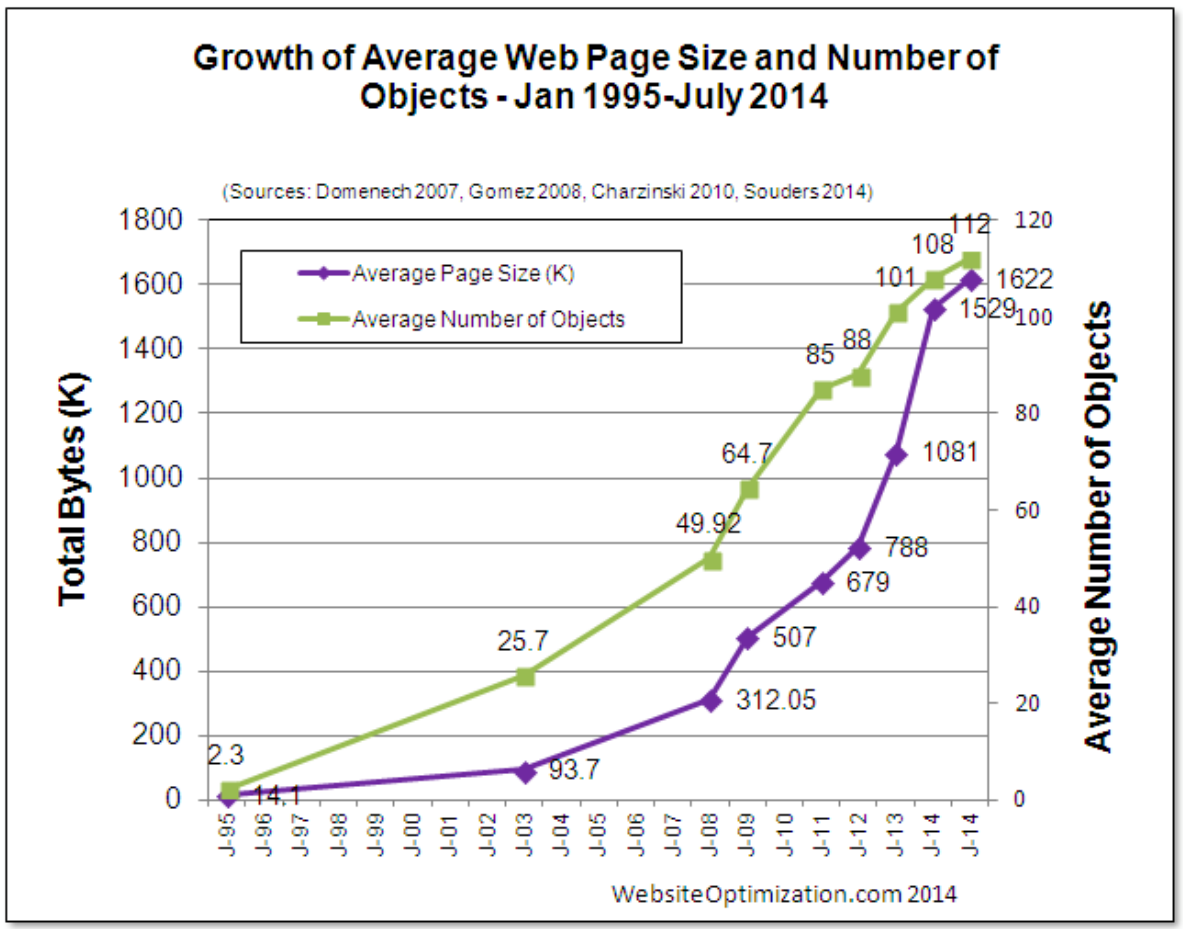
Service Provider Technical Team Leader

b.david@f5.com



TCP et Page Web

Les sites Web



Source : http://www.websiteoptimization.com/speed/tweak/average-web-page/

Taille moyenne des objets Web :
14Ko

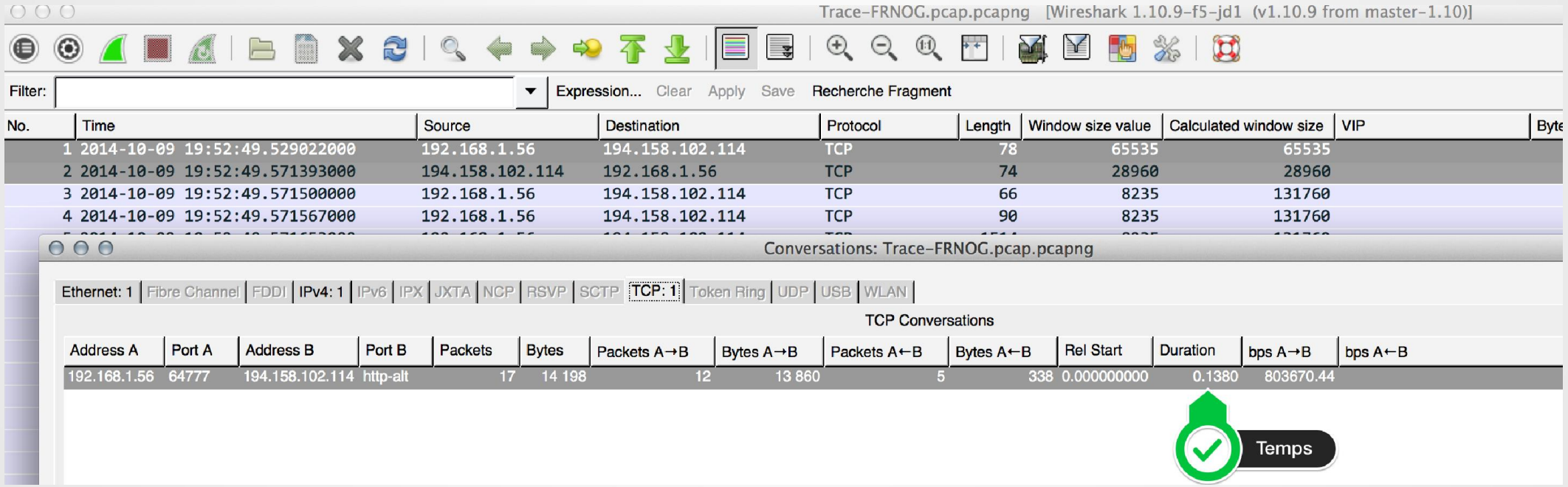
14 Ko cela donne quoi ?????

- En théorie :

14 Ko à 1 Mb	0,109375	seconde
14 Ko à 4 Mb	0,02734375	seconde
14 Ko à 20 Mb	0,00546875	seconde
14 Ko à 42 Mb	0,002604167	seconde
14 Ko à 100 Mb	0,00109375	seconde
14 Ko à 200 Mb	0,000546875	seconde
14 Kà à 1Gb	0,000106812	seconde

14 Ko cela donne quoi ?????

- En pratique :



Source : Test Iperf vers fibreinfo Massy depuis DSL SFR

14 Ko cela donne quoi ?????

En théorie depuis chez moi

14 Ko à 4 Mb

0,027 s

En pratique depuis chez moi

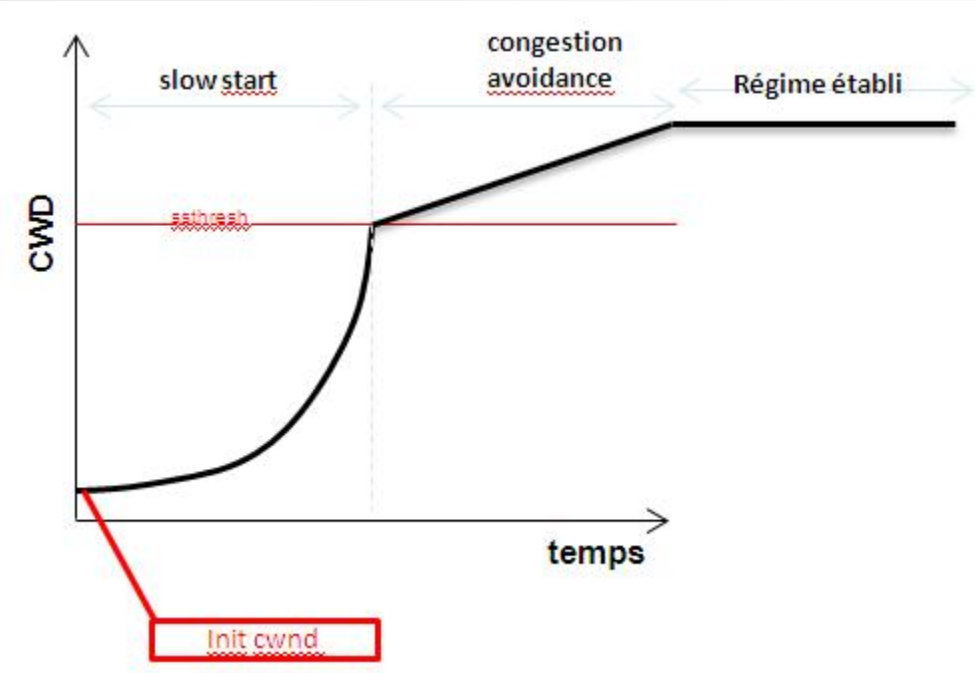
14 Ko à 4 Mb

0,14s

Alors pourquoi ?

Une histoire de TCP

- Dans le passé la qualité de la bande passante cliente était généralement mauvaise (perte de paquets, latence importante).
- TCP intègre des mécanismes pour gérer cette “mauvaise qualité”



- Le BDP ne peut ne jamais être atteint à cause du slow start.
- Très forte sensibilité à la perte de paquets
- Très efficace pour le téléchargement de gros fichier (> 10Mo)

TCP Slow Start

- Start with cwnd = 1 (slow start)
- On each successful ACK increment cwnd

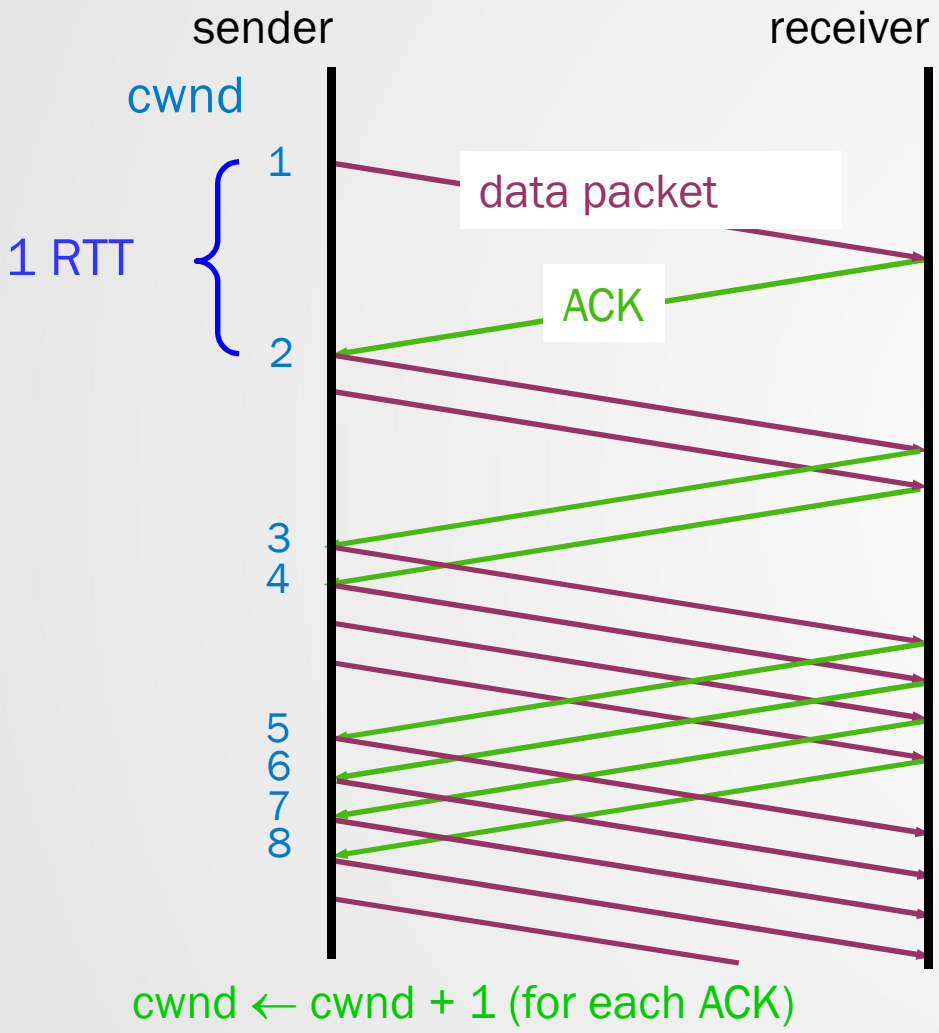
  cwnd ← cnwd + 1

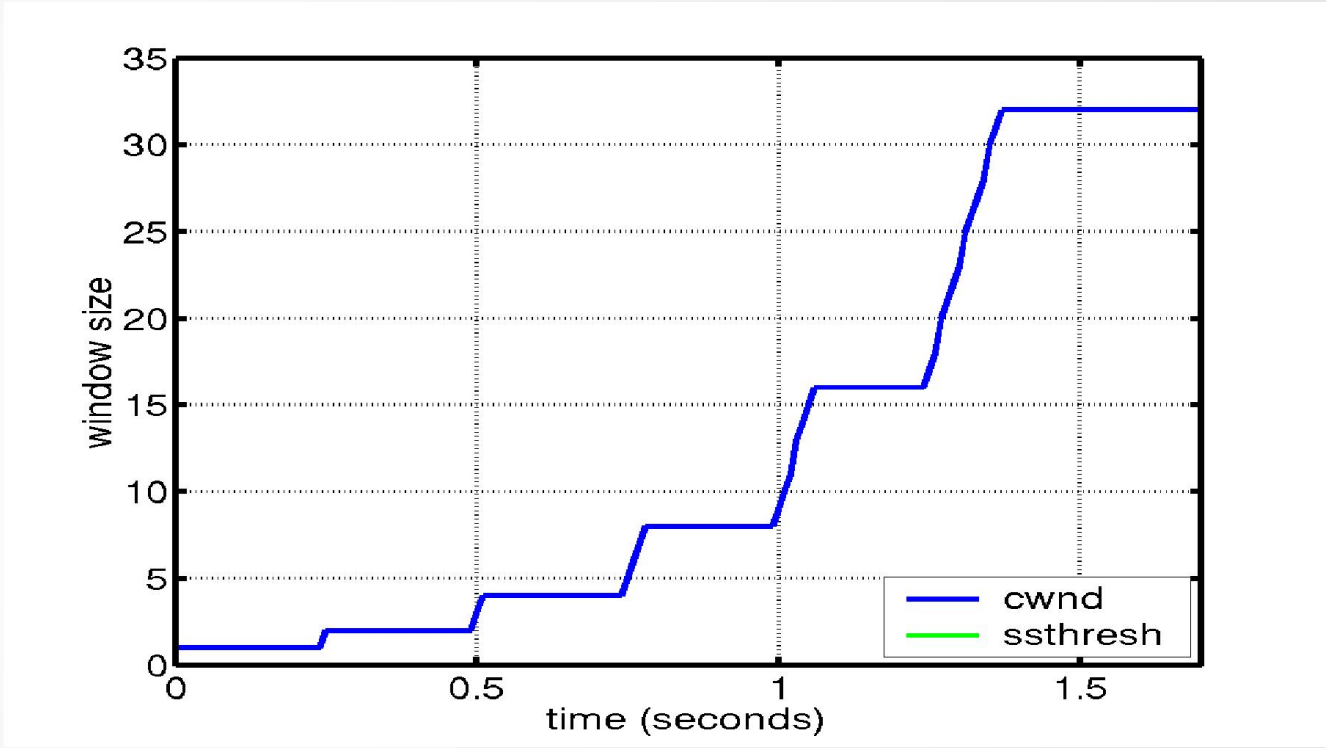
- Exponential growth of cwnd

  each RTT: cwnd ← 2 x cwnd

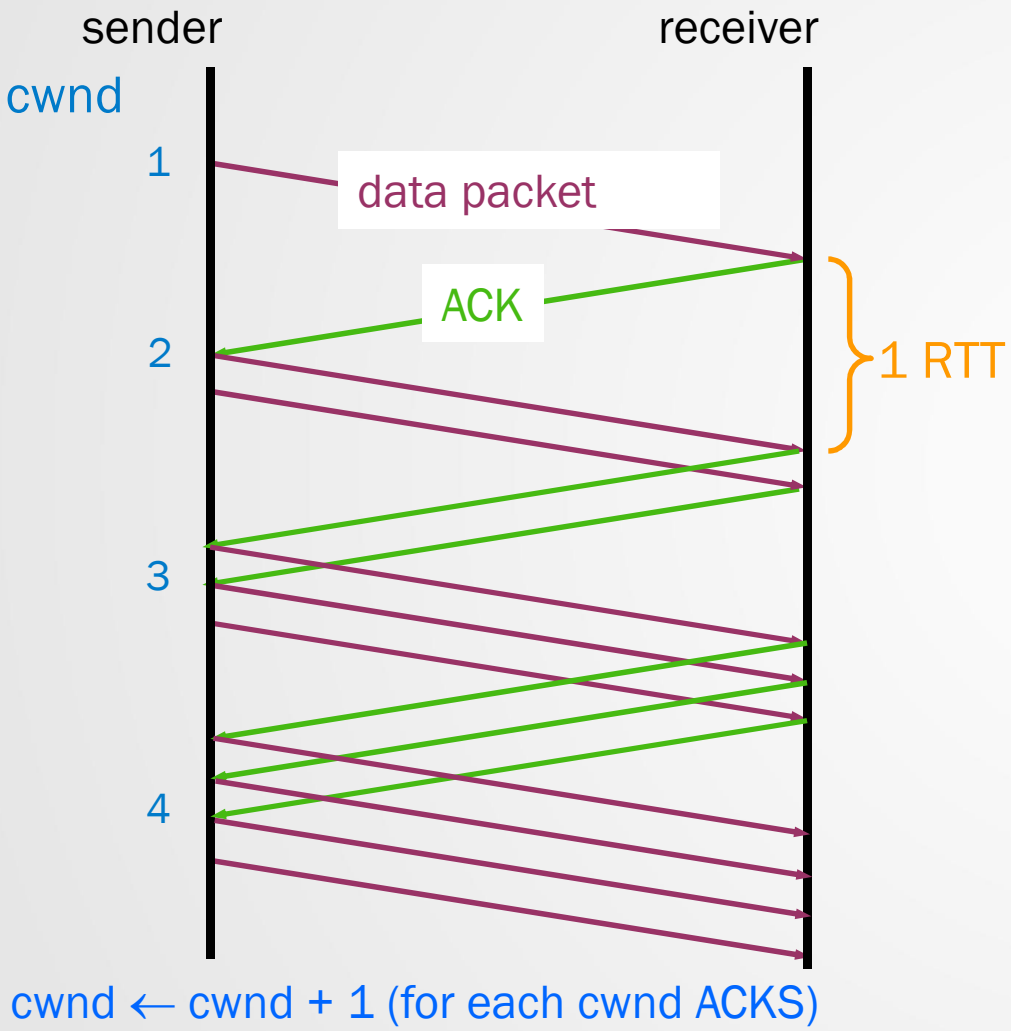
- Enter Congestion Avoidance when cwnd >= ssthresh
- Note
  - Initial ssthresh is typically high (so slow start continues until first packet loss)
  - After packet loss ssthresh is set equal to ½ of the current window size (which is minimum of cwnd and rwin)

TCP Slow Start

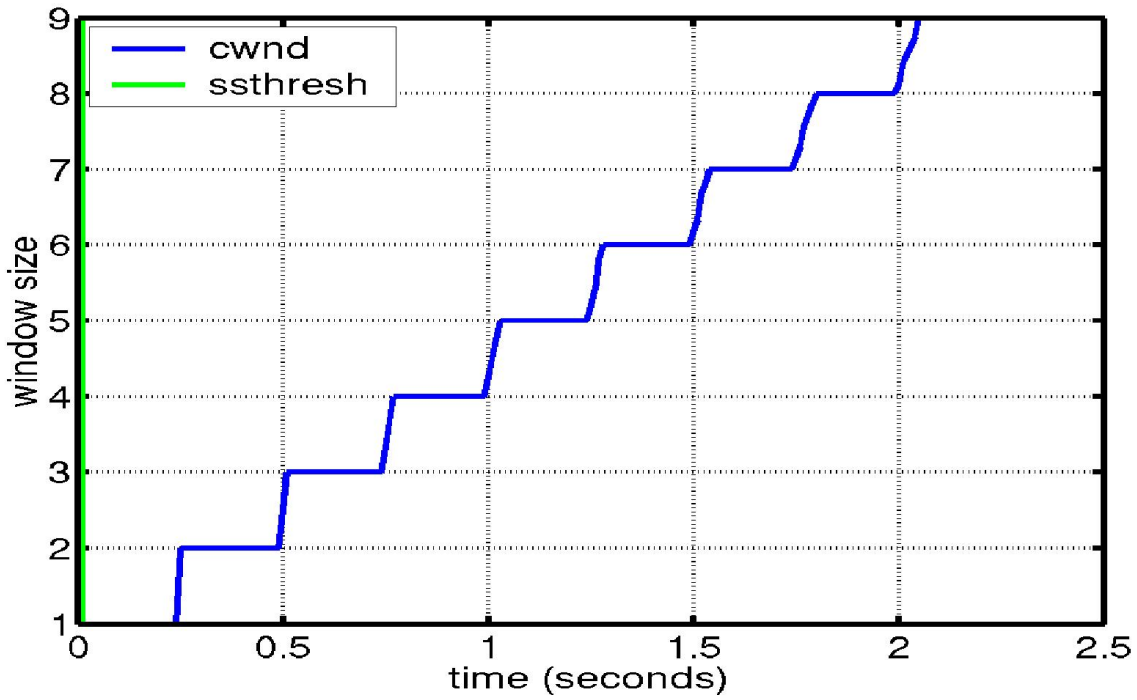




TCP Congestion Avoidance



cwnd ← cwnd + 1 (for each cwnd ACKS)

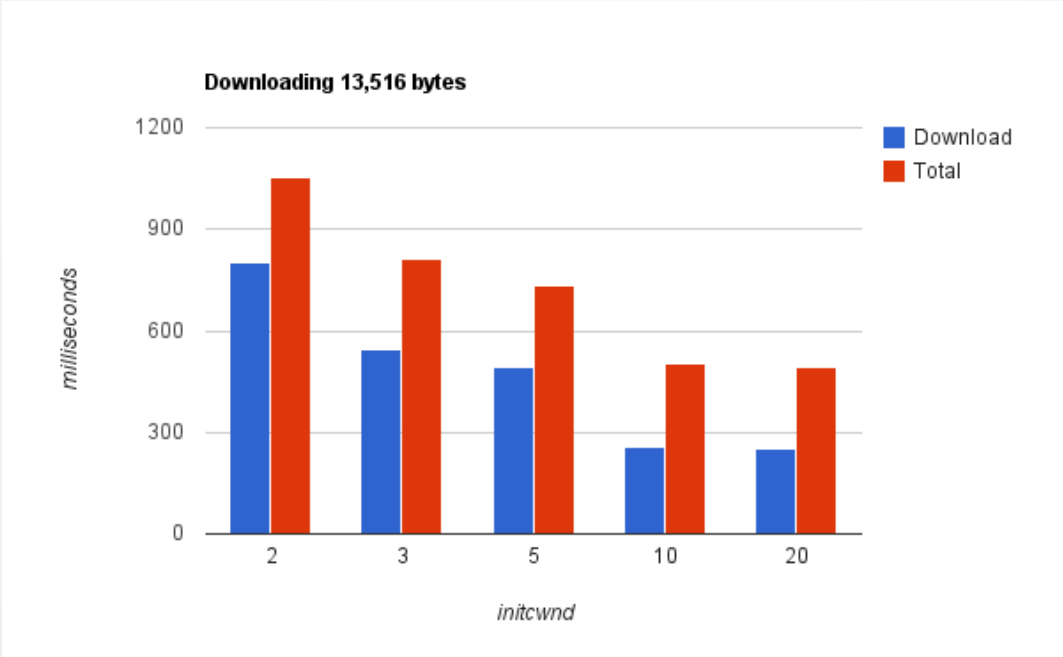


Pour booster les “petites connections”

- Supprimer le “slow start”
  - Enorme impact en fonction de l’algorithme de congestion
    - Algorithmes basés sur la perte de paquets
      - Reno, New Reno, High-Speed, Scalable, BIC, CUBIC
    - Algorithmes basés sur la latence (changement du RTT)
      - Vegas
    - Algorithmes basés sur la bande passante
      - Westwood, Westwood+
    - Hybride perte/latence
      - Illinois, Compound TCP, Woodside

Pour booster les “petites connections”

- Jouer sur la cwind du serveur: Proposition faite par google (https://developers.google.com/speed/articles/tcp_initcwnd_paper.pdf) d’augmenter cette valeur entre 10 et 16.



Source : http://www.cdnplanet.com/

Pour booster les “petites connections”

- En cas de perte de paquets, mise en place de “Rate Pacing” pour gérer les MicroBurst
- Utilisation de TCP Fast Open

- Attention certains paramétrages peuvent avoir beaucoup d’impacts sur des protocoles du niveau suppérieur (exemple : Nagle TCP_NODELAY et SSL)

Mais ne pas pousser l'optimisation trop loin ....

- Attention certains paramétrages peuvent avoir beaucoup d'impact sur des protocoles d'une niveau suppérieurs:

  exemple : Nagle TCP_NODELAY et SSL

  (http://www.extrahop.com/post/blog/performance-metrics/to-nagle-or-not-to-nagle-that-is-the-question/)

Surtout si maintenant de plus en plus de flux sont chiffrés..... SPDY/H2

SSL Vs HTTP2.0

HTTP 2.0 : Chiffrement ou pas ?

- HTTP/2 peut être géré sans SSL via le système “HTTP Upgrade”. This opens the door for all the optimizations that are currently done with HTTP
  - Cependant certain navigateur (Chrome and Firefox) ne supporterons pas cette déclinaison alors que d’autre oui (Internet Explorer).
- Proposition d’utiliser ALPN pour négocier entre le h2 & h2c
  - The former being the strong crypto/authenticated end-to-end connection which should be used when the user enters an https link.
  - The latter (h2c) is used for http (cleartext) links, for which the browser won’t perform authentication (it won’t check the server certificate)
    - This is implemented by Firefox