

RFC 1644 : T/TCP – TCP Extensions for Transactions Functional Specification

Stéphane Bortzmeyer
<stephane+blog@bortzmeyer.org>

Première rédaction de cet article le 5 Décembre 2007

Date de publication du RFC : Juillet 1994

<http://www.bortzmeyer.org/1644.html>

Il n'y a pas que des normes qui réussissent, à l'IETF. Il y a aussi des échecs, comme cette amélioration du protocole TCP, **T/TCP**, qui portait de nombreux espoirs à une époque et est pourtant aujourd'hui abandonnée.

T/TCP, normalisé dans ce RFC, visait à résoudre un problème de TCP : le temps d'établissement de la connexion est souvent supérieur au temps passé à transmettre des données, dès que les données sont peu nombreuses. C'est le cas des protocoles de type requête/réponse comme HTTP (beaucoup de pages Web font moins d'un ou un et demi kilo-octet, soit moins qu'un seul paquet IP) mais aussi du DNS ou de SNMP, que notre RFC cite dans sa section 1. Pour ces protocoles, on souhaiterait typiquement n'envoyer que deux paquets, un pour la question et un pour la réponse, laissant le mécanisme normal de TCP aux transferts de gros fichiers, pour lesquels le temps d'établissement de la connexion est négligeable. Le DNS ou SNMP ont choisi d'utiliser UDP et, comme celui-ci n'offre aucune fiabilité, doivent donc gérer les paquets perdus, les retransmissions, etc. HTTP a choisi d'utiliser TCP, ce qui simplifie la tâche des développeurs d'applications Web mais augmente nettement la durée d'une requête Web typique.

Pourquoi ? À cause d'une fonction essentielle de TCP, le *"3-way handshake"*. TCP ne peut pas envoyer ne serait-ce qu'un octet de données avant d'avoir terminé l'ouverture de la connexion et cette ouverture nécessite trois paquets, le paquet de demande d'ouverture d'une connexion, SYN, le paquet d'acceptation et d'accusé de réception, SYN+ACK et le paquet d'accusé de réception ACK. Si le RTT du réseau est important (par exemple sur les liaisons satellite), la durée du *"3-way handshake"* pourra être l'essentiel de la durée de la requête. On peut voir cet effet avec `echoping` <<http://echoping.sourceforge.net/>>, sur un site Web (<<http://www.kame.net/>>) dont la page d'accueil est très légère :

```
% echoping -v -h / www.kame.net
```

```
This is echoping, version 5.2.0.
```

```
Trying to connect to internet address 2001:200:0:8002:203:47ff:fea5:3085 80 to transmit 88 bytes...
Trying to send 256 bytes to internet address 2001:200:0:8002:203:47ff:fea5:3085...
Connected...
TCP Latency: 0.331326 seconds
Sent (88 bytes)...
Application Latency: 0.357898 seconds
3241 bytes read from server.
Elapsed time: 0.711559 seconds
```

Le temps d'établissement de la connexion ("*TCP latency*") est la moitié du temps total. Voici le "*3-way handshake*" correspondant, vu avec `tcpdump` (le grand S après les adresses IP indique un paquet SYN) :

```
21:43:15.110842 2001:7a8:7509::1.35261 > 2001:200:0:8002:203:47ff:fea5:3085.80: S 845465777:845465777(0) win
21:43:15.425956 2001:200:0:8002:203:47ff:fea5:3085.80 > 2001:7a8:7509::1.35261: S 3136927327:3136927327(0) a
21:43:15.426014 2001:7a8:7509::1.35261 > 2001:200:0:8002:203:47ff:fea5:3085.80: . ack 1 win 5728 <nop,nop,t
```

À la fin de cette échange d'un tiers de seconde, aucune donnée « utile » n'a encore été envoyée.

(Un autre problème de TCP pour ce genre de liens est le délai d'attente à la fin d'une connexion, où chaque machine doit garder un état pour pouvoir traiter correctement les paquets arrivant en retard. Pour un serveur très chargé, gérant beaucoup de requêtes, c'est une vraie contrainte.)

L'idée de base de T/TCP est de transporter dès le premier paquet les données, de façon à ce que l'acceptation de la connexion puisse également porter la réponse. Cela se fait en utilisant une option de TCP, CC ("*Connection Count*") qui indique la présence de données dans le paquet SYN. Le "*Connection Count*" sert également de "*cookie*" pour garantir qu'on parle bien à la même machine. Un certain nombre de détails doivent être pris en compte pour que cela marche, en conservant la fiabilité de TCP, même si des paquets sont perdus. Ce qui explique que le protocole soit plus complexe que résumé ici (les sections 2 et 3 le détaillent).

Par exemple, comme la connexion ne dure pas longtemps, la mesure du RTT que fait normalement TCP pour optimiser l'utilisation du réseau ne peut avoir lieu. La section 4.3 demande donc qu'une machine T/TCP garde un cache des RTT des différentes machines à laquelle elle parle, pour obtenir approximativement le même effet.

L'idée était donc très tentante et a été promue par plusieurs experts comme W. Richard Stevens, qui s'en est fait l'ardent défenseur <<http://www.kohala.com/start/ttcp.html>>. Mais peu de systèmes d'exploitation l'ont implémenté (FreeBSD est la principale exception mais Linux a eu aussi sa version <<http://ttcplinux.sourceforge.net/>>), peu de machines l'ont activé et peu de logiciels s'en servaient (echoping <<http://echoping.sourceforge.net>> avait une option `-r` pour T/TCP mais elle a été retirée pour les raisons exposées plus loin, cf. bogue #1350714 <http://sourceforge.net/tracker/index.php?func=detail&aid=1350714&group_id=4581&atid=104581>).

Outre la simple inertie face à la nouveauté, T/TCP a en effet été rapidement victime d'une sécurité plus faible que celle de TCP. Il n'y a pas de miracle : si on va plus vite, on vérifie moins. Alors que les

adresses IP d'une connexion TCP sont relativement authentifiées (car le numéro de séquence TCP doit être renvoyé, ce qui impose de donner sa vraie adresse IP pour recevoir les messages), T/TCP n'est guère plus fiable qu'UDP face à l'usurpation d'adresses IP. Comme les réponses sont souvent plus grandes que les requêtes, cela permet en outre une amplification de l'attaque.

Ces failles ont été documentées dans plusieurs articles :

- "*VU#464113 : TCP/IP implementations handle unusual flag combinations inconsistently*" <<http://www.kb.cert.org/vuls/id/464113>>
- "*Security Problems Associated With T/TCP*" <<http://www.mid-way.org/doc/ttcp-sec.txt>> en 1996,
- "*Example of RFC-1644 attack*" <<http://seclists.org/bugtraq/1998/Apr/0034.html>> sur Bugtraq en 1998,
- "*Enhanced Transaction TCP - Design and Implementation qui proposait une amélioration du protocole, qui ne semble pas avoir eu de suite.*" <<http://ttcplinux.sourceforge.net/theses/ETTCP.pdf>>

Sur une machine FreeBSD, T/TCP semble coupé par défaut, désormais (`sysctl -a | grep -E 'rfc1644|dropsyn'` pour voir les variables pertinentes).

Finalement, c'est le RFC 4614¹ qui a marqué l'abandon officiel de T/TCP. Le RFC 6247 entérinera cet abandon en reclassifiant notre RFC 1644 comme « intérêt historique seulement ».

1. Pour voir le RFC de numéro NNN, <http://www.ietf.org/rfc/rfcNNN.txt>, par exemple <http://www.ietf.org/rfc/rfc4614.txt>