

# RFC 4340 : Datagram Congestion Control Protocol (DCCP)

Stéphane Bortzmeyer  
<stephane+blog@bortzmeyer.org>

Première rédaction de cet article le 26 Avril 2006. Dernière mise à jour le 27 Avril 2006

Date de publication du RFC : Mars 2006

<http://www.bortzmeyer.org/4340.html>

---

Si IP est la base de l'Internet, la grande majorité des applications utilisent un protocole de transport situé au dessus d'IP et ne parlent pas directement IP. Les deux protocoles de transport les plus connus sont TCP et UDP mais il en existe d'autres dont, désormais, DCCP.

Pour comprendre l'intérêt de DCCP, qui vient d'être normalisé, il faut se souvenir des services que fournissent TCP et UDP :

- TCP fournit un service de flot de données (il ne transporte pas des messages séparés mais un flot continu d'octets) fiable, c'est-à-dire que c'est TCP, pas l'application, qui se charge d'envoyer et de surveiller les accusés de réception, et de demande des retransmissions si un paquet se perd. Outre ce service rendu à l'application, TCP rend un service à l'Internet : il assure le contrôle de congestion, c'est-à-dire qu'il ralentit le débit sortant si les accusés de réception lui montrent que le réseau est surchargé. TCP est donc gentil pour le réseau. Enfin, les données ne sont échangées qu'après l'établissement d'une connexion, établissement pendant lequel les deux parties peuvent s'accorder sur des options.
- UDP, lui, ne fournit presque rien de plus qu'IP : son service est de datagrammes, c'est-à-dire de messages séparés, et il n'a aucune fiabilité, l'application ne sait même pas si le datagramme est arrivé ou pas. Si elle a besoin d'accusé de réception ou de fiabilité, c'est à elle de les gérer. Si l'application ne pense pas à gérer la congestion, elle risque d'aggraver les problèmes de l'Internet en (ré)envoyant d'avantage de données justement lorsque le réseau est saturé. Contrairement à TCP, aucune connexion n'est nécessaire.

Il n'est donc pas étonnant que la plupart des applications utilisent TCP, bien plus simple pour le concepteur et pour le programmeur. Mais TCP consomme d'avantage de ressources et introduit des délais non négligeables, surtout à l'ouverture de connexion. UDP est donc utilisé si le temps de réponse est primordial (cas du DNS) ou bien si l'application peut tolérer la perte d'un certain nombre de paquets (cas des protocoles de transmission de son ou d'image comme RTP, où on peut accepter quelques silences ou quelques trames manquantes).

Entre TCP et UDP, d'autres protocoles de transport ont été créés pour des usages divers mais tous sont restés marginaux.

DCCP aura peut-être d'avantage de succès :

- Comme TCP, DCCP fournit un service d'accusé de réception (donc l'application peut savoir quels messages sont arrivés) et nécessite une connexion, ce qui permet de choisir des options. Mais il ne retransmet pas les données perdues, l'application est juste tenue au courant.
- Comme TCP, DCCP fournit à l'Internet un service de contrôle de la congestion. Il y en a même plusieurs, choisis lors de la négociation initiale.
- Comme UDP, DCCP fournit un service de messages non fiable.

On trouvera des détails et des articles plus détaillés sur le site de l'UCLA (<http://www.read.cs.ucla.edu/dccp/>) ou bien sur celui de l'ICIR (<http://www.icir.org/kohler/dcp/>) (qui semble moins bien maintenu et contient des informations dépassées).

Une mise en œuvre limitée de DCCP (<http://linux-net.osdl.org/index.php/DCCP>) est incluse dans le noyau Linux mais je n'ai pas encore eu l'occasion de la tester ni de voir quelle API devait être utilisée par echoping (<http://echoping.sourceforge.net/>) (un exemple de programmes figure dans <http://wand.net.nz/~iam4/dccp/dccp-cs-0.01.tar.bz2>).