Réseau « avec connexion » ou « sans connexion », un peu d'histoire et de technique

Stéphane Bortzmeyer
<stephane+blog@bortzmeyer.org>

Première rédaction de cet article le 7 juillet 2026

https://www.bortzmeyer.org/connexion-ou-pas.html

Dans les débats sur la conception d'un réseau informatique de grande taille, on entend souvent des références à une discussion « commutation de paquets » vs. « commutation de circuits ». Cela revient souvent, par exemple, quand on parle en France des choix de Transpac par rapport à l'Internet. Mais la discussion est parfois mal informée sur la réalité de ces architectures et elle oublie une autre discussion, « commutation de paquets » vs. « datagramme pur ».

Comme toute création humaine, l'Internet a son « roman national ». Dans la version la plus simpliste, les gens des télécommunications se seraient cramponnés trop longtemps à la « commutation de circuits » et l'Internet, grâce à l'invention par les informaticiens de la « commutation de paquets » se serait imposé en raison de cette supériorité technique. Bon, peu de gens vont défendre une version aussi caricaturale mais ce récit a quand même laissé des traces. Alors, approfondissons. D'abord, c'est quoi, la différence entre « commutation de paquets » et « commutation de circuits » ?

La commutation de circuits vient effectivement du monde de la téléphonie. À l'époque où une standardiste manipulait des connexions physiques, chaque coup de téléphone avait un circuit physique, un fil de cuivre continu, rien que pour lui. Ce système manquait évidemment de souplesse, et a fini par être remplacé par des circuits virtuels. On était toujours en commutation de circuits (il fallait établir le circuit avant de pouvoir échanger un mot, et le circuit était ensuite entièrement dédié) mais cette fois on n'avait plus de lien physique. Par exemple, les liens physiques étaient partagés temporellement.

La commutation de circuits est relativement simple à réaliser. Par exemple, si on l'applique à l'informatique, l'émetteur n'a rien d'autre à faire que d'envoyer les bits sur le circuit obtenu (c'est ainsi que ça fonctionnait avec les modems). Mais elle a des limites, notamment un certain gaspillage des ressources : le circuit est établi, et des ressources réservées, même si les deux participants n'ont rien à dire. Comme cette technique est celle de la téléphonie, elle est longtemps restée populaire dans le monde des télécommunications (entre autres car elle collait bien à leur modèle de facturation à la durée).

Et la commutation de paquets ? Cette fois, le flux de données est découpé en petites unités, les paquets, et chaque paquet est transmis « indépendamment ». Si des participants n'ont rien à dire, les

commutateurs peuvent consacrer leur temps à traiter des paquets d'autres participants. Chaque paquet comporte des informations d'adressage (qui étaient absentes en commutation de circuits puisque chaque conversation avait son propre circuit), permettant au commutateur de savoir quoi faire du paquet. Un tel système a traditionnellement plutôt la faveur des informaticiens, dont les ordinateurs ont un trafic réseau très irrégulier.

Mais c'est ensuite que les choses se compliquent. car « commutation de paquets » peut signifier deux choses différentes. On peut imposer la création d'un circuit virtuel avant tout échange, ou bien on peut utiliser des datagrammes, des paquets de données complètement autonomes, qui ne dépendent pas d'une connexion. (La terminologie dans ce domaine est souvent variable, et parfois floue. Le terme « datagramme » a pu être utilisé pour des paquets non autonomes et on parle alors de « datagramme pur » pour les paquets réellement autonomes. Ne vous lancez pas dans un débat sur ces techniques sans qu'il y ait auparavant un accord sur des définitions précises de tous les termes.)

Lorsqu'on crée un circuit virtuel, l'établissement préalable d'une connexion est nécessaire, exactement comme dans la commutation de circuits. La différence avec celle-ci est que les données seront ensuite découpées en paquets, et que chaque paquet portera une information d'adressage. Typiquement, un identifiant sera attribué à la connexion, lors de l'établissement initial de celle-ci, et cet identifiant sera mis dans chaque paquet. Les commutateurs sur le trajet sauront alors quoi faire du paquet.

Comme avec la commutation de circuits, cette commutation de paquets nécessite un établissement de connexion initial (et donc ne satisfait pas les applications qui veulent une très courte latence <https://www.bortzmeyer.org/latence.html>), et surtout nécessite un état dans tous les équipements intermédiaires : si un commutateur redémarre et perd son état, les connexions en cours seront coupées.

Par compte, avec le datagramme, chaque paquet est complètement auto-suffisant. Il porte l'adresse de destination de l'autre machine et peut être acheminé par les routeurs sans avoir à faire partie d'une connexion. Si un routeur redémarre, ou bien si le routage change et qu'on passe par d'autres routeurs, cela n'a pas de conséquences fâcheuses, puisqu'il n'y a pas d'état dans le réseau.

Comme souvent avec les choix d'ingénierie, il n'y a pas de solution idéale. Chacune de ces trois architectures (commutation de circuits, commutation de paquets, datagramme) a ses avantages et ses inconvénients. Ainsi, le datagramme permet un meilleur passage à l'échelle, ce qui est important pour un gros réseau mondial comme l'Internet, puisque les routeurs n'ont pas besoin de mémoriser un état. Il est une des raisons du succès de l'Internet. Mais en revanche, il impose que chaque paquet porte une information d'adressage complète, pas une simple étiquette attribuée à la connexion, ce qui utilise davantage d'octets et, surtout, impose des adresses de taille fixe, puisqu'on ne peut pas demander à un routeur qui traite des centaines de millions de paquets par seconde de faire des analyses trop compliquées. Ce choix des adresses de taille fixe a à son tour la conséquence que, si on a mal calculé la taille nécessaire <https://www.bortzmeyer.org/epuisement-adresses-ipv4.html>, augmenter celle-ci nécessite de changer le format des paquets <https://www.bortzmeyer.org/ipv6-compatibilite.html> et donc tous les routeurs.

Et puis, bien sûr, un autre inconvénient du datagramme est que son modèle (on envoie chaque paquet indépendamment) ne colle pas avec les demandes de la plupart des applications. La très grande majorité des applications préfère envoyer des octets dans l'ordre et qu'ils arrivent tous, et dans l'ordre. Les applications ont davantage besoin d'un modèle de circuit. C'est pour cela que, dans l'Internet, les applications utilisent en général TCP (RFC 9293[1]) ou QUIC (RFC 9000), qui recréent un circuit virtuel au dessus des datagrammes.

1. Pour voir le RFC de numéro NNN, https://www.ietf.org/rfc/rfcNNN.txt, par exemple https://www.ietf.org/rfc/rfc9293.txt

Si on fait un peu d'histoire, il est intéressant de se rappeler qu'Arpanet, l'ancêtre de l'Internet, était à commutation de circuits, tout comme X.25. L'Internet n'est devenu complètement « à datagrammes » qu'en 1983 <https://www.bortzmeyer.org/tcpip-transition.html> avec l'arrivée d'IPv4. (Notons aussi que les protocoles utilisés par les machines terminales, aussi bien avec l'Arpanet qu'avec X.25, ne sont pas ceux utilisés à l'intérieur du réseau. C'est une des innovations d'IP d'avoir utilisé le même protocole partout. Pour Arpanet, vous pouvez lire par exemple le RFC 33.) Par contre, Cyclades a utilisé le datagramme mais il est très difficile de trouver des articles techniques détaillés sur le fonctionnement de Cyclades.

Sur ce sujet, vous pouvez aussi consulter l'article de Valérie Schafer, « Circuits virtuels et datagrammes : une concurrence à plusieurs échelles <https://shs.cairn.info/article/HES_072_0029?lang=fr&ID_ARTICLE=HES_072_0029> ».