RFC 9293 : Transmission Control Protocol (TCP)

Stéphane Bortzmeyer
<stephane+blog@bortzmeyer.org>

Première rédaction de cet article le 20 août 2022

Date de publication du RFC : Août 2022

https://www.bortzmeyer.org/9293.html

Le protocole de transport TCP est l'un des piliers de l'Internet. La suite des protocoles Internet est d'ailleurs souvent appelée TCP/IP, du nom de ses deux principaux protocoles. Ce nouveau RFC est la norme technique de TCP, remplaçant l'antique RFC 793 [1], qui était vieux de plus de quarante ans, plus vieux que la plupart des lecteurices de ce blog. Il était temps de réviser et de rassembler en un seul endroit tous les détails sur TCP.

TCP est donc un protocole de transport. Rappelons brièvement ce qu'est un protocole de transport et à quoi il sert. L'Internet achemine des paquets IP de machine en machine. IP ne garantit pas leur arrivée : les paquets peuvent être perdus (par exemple parce qu'un routeur n'avait plus de place dans ses files d'attente), peuvent être dupliqués, un paquet peut passer devant un autre, pourtant envoyé avant, etc. Pour la grande majorité des applications, ce n'est pas acceptable. La couche de transport est donc chargée de remédier à cela. Première (en partant du bas) couche à être de bout en bout (les routeurs et autres équipements intermédiaires n'y participent pas, ou plus exactement ne devraient pas y participer), elle se charge de suivre les paquets, de les remettre dans l'ordre, et de demander aux émetteurs de ré-émettre si un paquet s'est perdu. C'est donc un rôle crucial, puisqu'elle évite aux applications de devoir gérer ces opérations très complexes. (Certaines applications, par exemple de vidéo, n'ont pas forcément besoin que chaque paquet arrive, et n'utilisent pas TCP.) Le RFC 8095 contient une comparaison détaillée des protocoles de transport de l'IETF.

Avant de décrire TCP, tel que spécifié dans ce RFC 9293, un petit mot sur les normes techniques de l'Internet. TCP avait originellement été normalisé dans le RFC 761 en 1980 (les couches 3 - IP et 4 étaient autrefois mêlées en une seule). Comme souvent sur l'Internet, le protocole existait déjà avant sa normalisation et était utilisé. La norme avait été rapidement révisée dans le RFC 793 en 1981. Depuis, il n'y avait pas eu de révision générale, et les RFC s'étaient accumulés. Comprendre TCP nécessitait donc de lire le RFC 793 puis d'enchainer sur plusieurs autres. Un RFC, le RFC 7414, avait même été écrit dans

1. Pour voir le RFC de numéro NNN, https://www.ietf.org/rfc/rfcNNN.txt, par exemple https://www.ietf.org/rfc/rfc793.txt

le seul but de fournir un guide de tous ces RFC à lire. Et il fallait également tenir compte de l'accumulation d'errata <https://www.rfc-editor.org/errata_search.php?rfc=793&rec_status=15&presentation=table>. Désormais, la situation est bien plus simple, tout TCP a été consolidé dans un RFC central, notre RFC 9293. (Notez que ce travail de consolidation a été aussi fait pour HTTP et SMTP, qui ne sont plus décrits par les RFC d'origine mais par des versions à jour, mais pas pour le DNS, qui reste le principal ancien protocole qui aurait bien besoin d'une remise à jour complète des documents qui le spécifient.) Ne vous faites toutefois pas d'illusion : malgré ses 114 pages, ce RFC 9293 ne couvre pas tout, et la lecture d'autres RFC reste nécessaire. Notamment, TCP offre parfois des choix multiples (par exemple pour les algorithmes de contrôle de la congestion, une partie cruciale de TCP), qui ne sont pas décrits intégralement dans la norme de base. Voyez par exemple le RFC 7323, qui n'a pas été intégré dans le RFC de base, et reste une extension optionnelle.

Attaquons-nous maintenant à TCP. (Bien sûr, ce sera une présentation très sommaire, TCP est forcément plus compliqué que cela.) Son rôle est de fournir aux applications un flux d'octets fiable et ordonné : les octets que j'envoie à une autre machine arriveront dans l'ordre et arriveront tous (ou bien TCP signalera que la communication est impossible, par exemple en cas de coupure du câble). TCP découpe les données en **segments**, chacun voyageant dans un datagramme IP. Chaque octet est numéroté (attention : ce sont bien les octets et pas les segments qui sont numérotés) et cela permet de remettre dans l'ordre les paquets, et de détecter les pertes. En cas de perte, on retransmet. Le flux d'octets est bidirectionnel, la même connexion TCP permet de transmettre dans les deux sens. TCP nécessite l'établissement d'une connexion, donc la mémorisation d'un état, par les deux pairs qui communiquent. Outre les tâches de fiabilisation des données (remettre les octets dans l'ordre, détecter et récupérer les pertes), TCP fournit du démultiplexage, grâce aux numéros de port. Ils permettent d'avoir plusieurs connexions TCP entre deux machines, qui sont distinguées par ces numéros de port (un pour la source et un pour la destination) dans l'en-tête TCP. Le bon fonctionnement de TCP nécessite d'édicter un certain nombre de règles et le RFC les indique avec "*MUST-n*" où N est un entier. Par exemple, "*MUST-2*" et "*MUST-3*" indiqueront que la somme de contrôle est obligatoire, aussi bien en envoi qu'en vérification à la réception.

Après ces concepts généraux, la section 3 de notre RFC rentre dans les détails concrets, que je résume ici. D'abord, le format de cet en-tête que TCP ajoute derrière l'en-tête IP et devant les données du segment. Il inclut les numéros de port, le numéro de séquence (le rang du premier octet dans les données), celui de l'accusé de réception (le rang du prochain octet attendu), un certain nombre de booléens ("*flags*", ou bits de contrôle, qui indiquent, par exemple, s'il s'agit d'une connexion déjà établie, ou pas encore, ou la fin d'une connexion), la taille de la fenêtre (combien d'octets peuvent être envoyés sans accusé de réception, ce qui permet d'éviter de surcharger le récepteur), une somme de contrôle, et des options, de taille variable (l'en-tête contient un champ indiquant à partir de quand commencent les données). Une option de base est MSS ("*Maximum Segment Size*") qui indique quelle taille de segment on peut gérer. Il existe d'autres options <https://www.iana.org/assignments/tcp-parameters/tcp-parameters.xml#tcp-parameters-1> comme les accusés de réception sélectifs du RFC 2018 ou comme le facteur multiplicatif de la taille de fenêtre du RFC 7323.

Avant d'expliquer la dynamique de TCP, le RFC définit quelques variables indispensables, qui font partie de l'état de la connexion TCP. Pour l'envoi de données, ce sont par exemple `SND.UNA` (ces noms sont là pour comprendre le RFC, mais un programme qui met en œuvre TCP peut évidemment nommer ces variables comme il veut), qui désigne le numéro de séquence du début des données envoyées, mais qui n'ont pas encore fait l'objet d'un accusé de réception, ou `SND.NXT`, le numéro de séquence du début des données pas encore envoyées ou encore `SND.WND`, la taille de la fenêtre d'envoi. Ainsi, on ne peut pas envoyer d'octets dont le rang serait supérieur à `SND.UNA + SND.WND`, il faut attendre des accusés de réception (qui vont incrémenter `SND.UNA`) ou une augmentation de la taille de la fenêtre. Pour la réception, on a `RCV.NXT`, le numéro de séquence des prochains paquets si tout est normal, et `RCV.WND`, la taille de la fenêtre de réception.

TCP est un protocole à état et il a donc une machine à états, décrite en section 3.3.2. Parmi les états, `LISTEN`, qui indique un TCP en train d'attendre un éventuel pair, `ESTAB` (ou `ESTABLISHED`), où TCP