RFC 8923 : A Minimal Set of Transport Services for End Systems

Stéphane Bortzmeyer
<stephane+blog@bortzmeyer.org>

Première rédaction de cet article le 22 octobre 2020

Date de publication du RFC : Octobre 2020

https://www.bortzmeyer.org/8923.html

Ce nouveau RFC s'incrit dans le travail en cours à l'IETF pour formaliser un peu plus les services que la couche transport offre aux applications. Le RFC 8095 [1] décrivait tous les services possibles. Ce nouveau RFC 8923 liste les services minimums, ceux à offrir dans presque tous les cas.

Comme le note le RFC 8095, les applications tournant sur la famille de protocoles TCP/IP ne peuvent en général pas aujourd'hui exprimer les **services** attendus de la couche transport. Elles doivent choisir un protocole de transport (comme TCP ou UDP), même si ce protocole ne convient pas tout à fait. Toute l'idée derrière le travail du groupe TAPS <https://datatracker.ietf.org/wg/taps/documents/> de l'IETF est de permettre au contraire aux applications d'indiquer des services (« je veux un envoi de données fiable et dans l'ordre, avec confidentialité et intégrité ») et que le système d'exploitation choisisse alors la solution la plus adaptée (ici, cela pourrait être TCP ou SCTP avec TLS par dessus, ou bien le futur QUIC). De la même façon qu'un langage de programmation de haut niveau utilise des concepts plus abstraits qu'un langage de bas niveau, on pourrait ainsi avoir une interface de programmation réseau de plus haut niveau que les traditionnelles prises normalisées par Posix <http://pubs.opengroup.org/onlinepubs/9699919799/functions/V2_chap02.html#tag_15_10>. Ce RFC ne normalise pas une telle API, il décrit seulement les services attendus. La liste des services possibles est dans le RFC 8095, et les protocoles qui les fournissent sont documentés dans le RFC 8303 et RFC 8304. Le cœur de notre nouveau RFC 8923 est la section 6, qui liste les services minimum attendus (je vous en dévoile l'essentiel tout de suite : établissement de connexion, fin de connexion, envoi de données, réception de données).

Ce RFC ne spécifie pas une API précise, cela sera éventuellement fait dans un autre document, draft-ietf-taps-interface.

1. Pour voir le RFC de numéro NNN, https://www.ietf.org/rfc/rfcNNN.txt, par exemple https://www.ietf.org/rfc/rfc8095.txt

À noter que les services minimums exposés ici peuvent être mis en œuvre sur TCP, et, dans certains cas, sur UDP. TCP offre un ensemble de services qui est quasiment un sur-ensemble de celui d'UDP, donc, sur le seul critère des services, on pourrait n'utiliser que TCP. (Exercice : quelle(s) exception(s) trouvez-vous ?) Comme il s'agit de services de la couche transport aux applications, l'implémentation peut être faite d'un seul côté (il n'est pas nécessaire que les deux partenaires qui communiquent le fassent).

Et, si vous lisez le RFC, faites attention à la section 2 sur la terminologie, section qui reprend le RFC 8303 : une **fonction** ("*Transport Feature*") est une fonction particulière que le protocole de transport va effectuer, par exemple la confidentialité, la fiabilité de la distribution des données, le découpage des données en messages, etc, un **service** ("*Transport Service*") est un ensemble cohérent de fonctions, demandé par l'application, et un **protocole** ("*Transport Protocol*") est une réalisation concrète d'un ou plusieurs services.

Comment définir un jeu minimal de services (section 3 du RFC) ? D'abord, il faut définir les services « sémantiques » ("*functional features*") que l'application doit connaitre. Par exemple, l'application doit savoir si le service « distribution des données dans l'ordre » est disponible ou pas. Et il y a les services qui sont plutôt des optimisations comme l'activation ou la désactivation de DSCP ou comme l'algorithme de Nagle. Si l'application ignore ces services, ce n'est pas trop grave, elle fonctionnera quand même, même si c'est avec des performances sous-optimales.

La méthode utilisée par les auteurs du RFC pour construire la liste des services minimums est donc :
— Catégorisation des services du RFC 8303 (sémantique, optimisation...) ; le résultat de cette catégorisation figure dans l'annexe A de notre RFC,
— Réduction aux services qui peuvent être mis en œuvre sur les protocoles de transport qui peuvent passer presque partout, TCP et UDP,
— Puis établissement de la liste des services minimum.

Ainsi (section 4 du RFC), le service « établir une connexion » peut être mis en œuvre sur TCP de manière triviale, mais aussi sur UDP, en refaisant un équivalent de la triple poignée de mains. Le service « envoyer un message » peut être mis en œuvre sur UDP et TCP (TCP, en prime, assurera sa distribution à l'autre bout, mais ce n'est pas nécessaire). En revanche, le service « envoyer un message avec garantie qu'il soit distribué ou qu'on soit notifié », s'il est simple à faire en TCP, ne l'est pas en UDP (il faudrait refaire tout TCP au dessus d'UDP).

La section 5 du RFC discute de divers problèmes quand on essaie de définir un ensemble minimal de services. Par exemple, TCP ne met pas de délimiteur dans le flot d'octets qui circulent. Contrairement à UDP, on ne peut pas « recevoir un message », seulement recevoir des données. Certains protocoles applicatifs (comme DNS ou EPP) ajoutent une longueur devant chaque message qu'ils envoient, pour avoir une sémantique de message et plus de flot d'octets. TCP n'est donc pas un strict sur-ensemble d'UDP.

Pour contourner cette limitation, notre RFC définit (section 5.1) la notion de « flot d'octets découpé en messages par l'application » ("*Application-Framed Bytestream*"). Dans un tel flot, le protocole applicatif indique les frontières de message, par exemple en précédant chaque message d'une longueur, comme le fait le DNS.

Autre problème amusant, certains services peuvent être d'assez bas niveau par rapport aux demandes des applications. Ainsi, le RFC 8303 identifie des services comme « couper Nagle », « configurer DSCP » ou encore « utiliser LEDBAT ». Il serait sans doute plus simple, pour un protocole applicatif, d'avoir une configuration de plus haut niveau. Par exemple, « latence minimale » désactiverait Nagle et mettrait les bits qui vont bien en DSCP.

Nous arrivons finalement au résultat principal de ce RFC, la section 6, qui contient l'ensemble minimal de services. Chaque service est configurable via un ensemble de paramètres. Il est implémentable uniquement avec TCP, et d'un seul côté de la connexion. Dans certains cas, TCP fournira **plus** que ce qui est demandé, mais ce n'est pas un problème. Je ne vais pas lister tout cet ensemble minimal ici, juste énumérer quelques-uns de ses membres :

— Créer une connexion, avec des paramètres comme la fiabilité demandée,
— Mettre fin à la connexion,
— Envoyer des données, sans indication de fin de message (cf. la discussion plus haut),
— Recevoir des données.

Pour l'envoi de données, les paramètres sont, entre autres :

— Fiabilité (les données sont reçues à l'autre bout ou, en tout cas, l'émetteur est prévenu si cela n'a pas été possible) ou non,
— Contrôle de congestion ou non, sachant que si l'application ne demande pas de contrôle de congestion au protocole de transport, elle doit faire ce contrôle elle-même (RFC 8085),
— Idempotence,
— Etc.

Pour la réception de données, le service est juste la réception d'octets, un protocole applicatif qui veut des messages doit utiliser les « flots d'octets découpés par l'application » décrits en section 5.1.

On notera que la sécurité est un service, également, ou plutôt un ensemble de services (section 8 du RFC). Dans les protocoles IETF, la sécurité est souvent fournie par une couche intermédiaire entre la couche de transport et la couche application à proprement parler. C'est ainsi que fonctionne TLS, par exemple. Mais la tendance va peut-être aller vers l'intégration, avec QUIC.