RFC 9622 : An Abstract Application Layer Interface to Transport Services

Stéphane Bortzmeyer
<stephane+blog@bortzmeyer.org>

Première rédaction de cet article le 23 janvier 2025

Date de publication du RFC : Janvier 2025

https://www.bortzmeyer.org/9622.html

Voici un des RFC du projet TAPS <https://datatracker.ietf.org/wg/taps/>, qui vise à proposer une vision cohérente des protocoles de transport aux applications. Ce RFC décrit (mais de manière assez abstraite) l'API de TAPS.

Avant de le lire, il est très fortement recommandé de réviser le RFC 9621 [1], qui décrit les buts du projet TAPS, et les principes de l'API "*Transport Services*" (autrefois surnommée « "*post-sockets*" »). Mais le RFC 9621 restait assez abstrait, et notre RFC 9622 se fait un peu plus concret (sans aller jusqu'à une vraie API puisqu'il reste partiellement indépendant du langage de programmation utilisé).

Rappelons quelques principes de TSAPI ("*Transport Services API*", l'API qui est le but final du projet TAPS). D'abord, l'API est asynchrone (ce qui n'est pas idéal pour les langages avec parallélisme comme Go ou Elixir). On crée des Connexions (la majuscule au début est là pour se distinguer du concept flou de connexion, et pour désigner un type d'objets particulier), voire des Préconnexions si on veut les configurer avant, on leur définit des propriétés (en indiquant si elles sont impératives ou facultatives), on peut ensuite activer les Connexions (ou écouter passivement que l'autre machine les active), puis on envoie et on reçoit des Messages. La section 3 du RFC résume l'API, et, si vous êtes pressé·e, vous pouvez vous contenter de cette section. Elle contient quelques exemples pratiques, en pseudo-code.

À propos de pseudo-code, le RFC note que les conventions de nommage de l'API ne pourront pas forcément être suivies religieusement par toutes les mises en œuvre puisque certains langages de programmation ont des règles différentes. Mais c'est un détail.

1. Pour voir le RFC de numéro NNN, https://www.ietf.org/rfc/rfcNNN.txt, par exemple https://www.ietf.org/rfc/rfc9621.txt

Parmi les points dignes d'intérêt dans le RFC, on note que les applications peuvent indiquer le domaine d'avitaillement (PvD, "*ProVisioning Domain*", défini dans le RFC 7556) qu'elles veulent utiliser. Elles peuvent par exemple l'identifier par son nom de domaine (RFC 8801).

Autre point intéressant, les cadreurs ("*framers*"). Décrits dans la section 9.1.2, ce sont des couches logicielles qui vont mettre en œuvre les exigences du protocole sous-jacent. Lors d'une opération d'envoi ou de réception d'un Message, les cadreurs vont tenir compte du protocole pour transformer un Message en flux d'octets et réciproquement. (Le RFC suggère de regarder le RFC 9329, pour un exemple.)

Pour mettre en œuvre cette API assez abstraite dans un langage de programmation spécifique, l'annexe A du RFC donne quelques conseils. Il y a des mises en œuvre de TAPS (plusieurs sont listées dans le RFC 9623, annexe C). On peut citer :

— En Python, PyTaps <https://github.com/fg-inet/python-asyncio-taps>, mais qui n'est plus maintenu depuis longtemps (le projet TAPS existe depuis des années et ne semble pas susciter d'enthousiasme délirant),
— En C, NEAT <https://github.com/NEAT-project/neat>, qui semble également abandonné.

Notez un point positif dans le RFC : l'annexe C du RFC 9623, qui décrit les mises en œuvre existantes donne non seulement les URL « normaux » du projet, et ceux sur GitHub mais aussi un lien Software Heritage, pour la pérennité. C'est une très bonne chose, car certains RFC ont une longue durée de vie). Par exemple, NEAT est mais aussi .