RFC 6458 : Sockets API Extensions for Stream Control Transmission Protocol (SCTP)

Stéphane Bortzmeyer
<stephane+blog@bortzmeyer.org>

Première rédaction de cet article le 12 décembre 2011

Date de publication du RFC : Décembre 2011

https://www.bortzmeyer.org/6458.html

Il n'y a pas de protocole qui réussit sans une API, c'est-à-dire une spécification d'une interface qui permet au programmeur d'utiliser le protocole en question. Le protocole de transport SCTP, normalisé dans le RFC 4960 [1], n'avait pas jusqu'à présent d'API standard et les programmes ne fonctionnaient donc qu'avec une seule bibliothèque. Désormais, avec ce RFC, SCTP a des chances de voir le développement de programmes portables.

SCTP est le principal concurrent de TCP dans la catégorie « protocole de couche 4 fiable » (i.e. qui garantit l'acheminement des données, contrairement à UDP). Mais il est beaucoup moins utilisé que TCP, plus ancien, et qui a une API standard depuis longtemps, interface décrite dans la norme POSIX. Il y a aussi d'autres raisons au moindre déploiement de SCTP. Mais l'absence d'API jouait certainement un rôle : pas moyen de programmer avec SCTP sans devoir se limiter, non seulement à un système donné, mais en prime à une mise en œuvre particulière de SCTP. Les instructions d'installation devenaient donc pénibles à lire. C'est ce problème que résout notre RFC.

L'API décrite dans ce RFC est longue (le RFC fait 113 pages) donc je résume plutôt sauvagement. Elle permet d'écrire des programmes qui font la même chose qu'avec TCP, comme des programmes qui tirent profit des fonctions spécifiques de SCTP. Car la sémantique de SCTP n'est pas exactement la même que celle de TCP, le remplacement de l'un par l'autre n'est pas entièrement invisible aux applications. Les points importants de cette nouvelle API :

— Elle est basée sur la traditionnelle interface "*socket*",
— Elle permet le style de programmation un-vers-N (section 3). TCP ne permet que un-vers-un, UDP permet d'utiliser une prise ("*socket*") pour du un-vers-N, et SCTP aussi. En termes SCTP, une prise permet de contrôler plusieurs **associations**. Par contre, pas de diffusion, que ne permet pas SCTP.

1. Pour voir le RFC de numéro NNN, https://www.ietf.org/rfc/rfcNNN.txt, par exemple https://www.ietf.org/rfc/rfc4960.txt

— Elle permet également le style un-vers-un (section 4), familier aux programmeurs qui utilisent TCP. Avec ce style (une seule association par prise), les applications existantes qui se servent de TCP peuvent être portées en SCTP avec très peu d'efforts.

Ces deux styles ne sont pas compatibles dans le même programme, le programmeur devra choisir.

Notez que les efforts de conception d'une API SCTP sont anciens. Résultat, il traîne dans la nature des tutoriels, des Howto, des exemples de programme utilisant des mécanismes qui ne sont plus les mécanismes recommandés. Plusieurs éléments des API expérimentales qui ont précédé celle-ci sont repris dans ce RFC, en les notant comme dépassés et ne devant plus être utilisés pour de nouveaux programmes.

Un programme serveur typique qui utilise le style un-vers-N ressemble à :

```
socket(..., SOCK_SEQPACKET, IPPROTO_SCTP): /* Le type SOCK_SEQPACKET indique le
style 1-vers-n. Section 3.1.1 */
bind(...);
listen(...); /* Pas besoin de accept(). Les nouvelles associations
sont gérées automatiquement (si le serveur le veut, il est prévenu
lors des recvmsg(), évenement SCTP_ASSOC_CHANGE). Section 3.1.3 */
recvmsg(...);
sendmsg(...);
close(...); /* close() ferme tout. Pour ne fermer qu'une seule association, on utilise
sendmsg() avec SCTP_EOF. */
```

pendant que le client fera plutôt :

```
socket(..., SOCK_SEQPACKET, IPPROTO_SCTP):
sendmsg(...);
recvmsg(...);
close(...):
```

Toutes les associations du serveur seront représentées par une seule prise et distinguées par leur identificateur d'association, de type `sctp_assoc_t`. La section 3.3 fournit les détails pour ce cas. Par exemple, si le programme n'utilise qu'un seul tampon par prise réseau, une association qui traîne peut bloquer toutes les autres. Le RFC recommande d'utiliser des prises non-bloquantes.

Quant au style un-vers-un, c'est celui de TCP et il est familier à tous les programmeurs réseau. Il est présenté en section 4. L'idée est qu'une application d'aujourd'hui, qui utilise TCP, sera ainsi portée en très peu de temps.

La séquence de commandes pour le serveur est typiquement :

```
socket(..., SOCK_STREAM, IPPROTO_SCTP); /* Le protocole IPPROTO_SCTP
est la seule différence avec TCP. */
bind(...);
listen(...);
accept(...);
/* Ensuite, recv() et send() avec la nouvelle prise retournée par
accept(). */
close(...);
```

Et chez le client :

```
socket(..., SOCK_STREAM, IPPROTO_SCTP);
connect(...);
/* Ensuite, recv()/recvmsg() et send()/sendmsg() */
close(...);
```

Voici pour le code. La section 5 présente ensuite les nouvelles structures de données, celles qui sont spécifiques à SCTP, lorsqu'on utilise `recvmsg()` et `sendmsg()` pour des opérations de contrôle (et pas seulement d'envoi et de récupération de données). Ces fonctions prennent un paramètre `message` de type `msghdr` (RFC 3542). On peut s'en servir pour définir des paramètres de la connexion (options `SCTP_INIT` et `SCTP_SNDINFO`), ou obtenir des informations supplémentaires lors de la réception de données (option `SCTP_RCVINFO`, par exemple le champ `rcv_assoc_id` indiquera par quelle association est venue le message).

Pendant la durée de vie d'une connexion SCTP, des tas d'évenements qui n'ont pas d'équivalent dans le monde TCP peuvent se produire : changement d'adresse IP d'un pair (`SCTP_PEER_ADDR_CHANGE`, section 6.1.2), établissement de nouvelles associations (ou arrêt des anciennes, cf. `SCTP_ASSOC_CHANGE`, section 6.1.1), etc. La section 6 décrit comment être informé de ces évenements. Quant aux options des prises spécifiques à SCTP, elles sont dans la section 8. Elles s'utilisent avec `setsockopt()` et `getsockopt()`. Par exemple, `SCTP_ASSOCINFO` permet d'obtenir (ou de modifier) les paramètres liés aux associations, `SCTP_PRIMARY_ADDR` d'obtenir l'adresse IP du pair pour une association donnée, etc.

Le RFC contient en annexe A deux exemples d'utilisation de cette API, un pour un serveur en un-vers-N (en ligne sur `https://www.bortzmeyer.org/files/sctp-sample-server.c`) et un pour un client en un-vers-un (en ligne sur `https://www.bortzmeyer.org/files/sctp-sample-client.c`). Pour les raisons expliquées plus haut (retard à normaliser l'API et bibliothèque non standards développées en attedant), ces exemples ne compilent pas sur une Debian ou une Ubuntu récente :

```
% gcc sctp-sample-client.c
sctp-sample-client.c: In function 'main':
sctp-sample-client.c:40:25: error: storage size of 'info' isn't known
sctp-sample-client.c:97:54: error: 'SCTP_SENDV_SNDINFO' undeclared (first use in this function)
sctp-sample-client.c:97:54: note: each undeclared identifier is reported only once for each function it appears
```

Je n'ai pas encore trouvé de système où ces exemples compilent. Il existe en effet plusieurs mises en œuvre de SCTP mais pas de cette API. Plus exactement, les implémentations de cette API sont ultra-récentes et ne sont pas encore arrivés chez l'utilisateur. On a :

- Sur Linux Linux Kernel Stream Control Transmission Protocol <http://lksctp.sourceforge.net/> (dont l'API est fondée sur une très vieille version de l'"*Internet-Draft*" qui a fini par donner naissance à ce RFC),
- Sur FreeBSD, SCTP existe depuis longtemps <http://lists.freebsd.org/pipermail/freebsd-current/2006-November/067218.html> mais l'API a le même problème.
- Et pareil pour sctplib <http://www.sctp.de/sctp-download.html> (une mise en œuvre de SCTP en espace utilisateur) et son API.

Le RFC note que des implémentations de l'API standard existent sur Linux, FreeBSD et Solaris mais leur déploiement effectif est inconnu. Elles ont apparemment déjà servi à ajouter SCTP à Firefox et Chrome.

Le livre de référence sur la programmation réseau, celui de Stevens <https://www.bortzmeyer.org/unix-network-programming.html>, ne parle pas de SCTP dans les deux premières éditions. C'est à partir de la troisième édition (réalisée par d'autres, après la mort de l'auteur) que SCTP apparaît.

En attendant, la bogue echoping <http://echoping.sourceforge.net> #1676608 <https://sourceforge.net/tracker/?func=detail&aid=1676608&group_id=4581&atid=354581> risque d'attendre longtemps (d'autant plus que je n'ai plus le temps de m'occuper d'echoping <https://www.bortzmeyer.org/echoping-nouveau-mainteneur.html>).