RFC 6951 : UDP Encapsulation of SCTP Packets for End-Host to End-Host Communication

Stéphane Bortzmeyer
<stephane+blog@bortzmeyer.org>

Première rédaction de cet article le 30 mai 2013

Date de publication du RFC : Mai 2013

https://www.bortzmeyer.org/6951.html

Normalement, un nouveau protocole de transport comme SCTP ne devrait avoir besoin de l'autorisation de personne pour être déployé entre deux machines majeures et consentantes. Le modèle de l'Internet, fondé sur le protocole IP, permet aux machines terminales, sans autorisation d'un intermédiaire, de communiquer en mettant ce qu'elles veulent dans les paquets IP. Cela, c'est la théorie. En pratique, par incompétence, sécurité ultra-rigide ou paresse, des tas d'intermédiaires se permettent de se placer sur le chemin entre deux machines et ne laissent passer qu'un petit nombre de protocoles de transport (typiquement uniquement TCP et UDP). Un nouvel arrivé (SCTP a quand même été normalisé dans le RFC 2960 [1] en 2000 donc, « nouveau » est relatif ; la norme actuelle est le RFC 4960) a donc le plus grand mal à se faire une place au soleil. On en est donc réduit à des astuces plus ou moins propres comme celle décrite dans ce RFC : encapsuler SCTP dans de l'UDP.

La principale cible visée est donc le grand nombre de machines coincées derrière un routeur NAT. Mais cette encapsulation peut aussi servir pour mettre en œuvre SCTP sans avoir accès au noyau, entièrement dans une application, sans avoir de privilèges particulier (ce qui est souvent nécessaire pour implémenter un protocole de transport). Attention, il ne s'agit pas de faire un tunnel pour mettre en relation deux réseaux, mais de faire communiquer deux machines terminales qui voudraient faire du SCTP et qui en sont empêchées par une "*middlebox*" (cf. section 4).

Pourquoi UDP, au fait, et pas TCP ? Parce que SCTP fournit tous les services de TCP (notamment le contrôle de congestion et la fiabilité) et qu'il ferait donc double emploi avec TCP.

Notez que rien n'empêcherait de faire un routeur NAT qui permettrait SCTP, comme ils permettent aujourd'hui de faire du TCP et de l'UDP. C'est juste que de tels routeurs, s'ils existent, sont très rares.

1. Pour voir le RFC de numéro NNN, https://www.ietf.org/rfc/rfcNNN.txt, par exemple https://www.ietf.org/rfc/rfc2960.txt

Bon, et comment SCTP sur UDP fonctionne (section 5) ? On utilise par défaut le port de destination **et** de source 9899. Toutefois, celui-ci aura pu être modifié en route par un routeur NAT et le port à l'arrivée sera donc peut-être différent. D'autre part, certaines machines peuvent se trouver dans l'obigation d'écouter sur un autre port et il faut donc qu'une mise en œuvre de ce RFC soit capable d'utiliser un port différent par correspondant (cf. l'API plus loin), et de se souvenir de ces ports. On met le paquet SCTP dans un paquet UDP qui sera lui-même transporté dans IP (peut-être avec des en-têtes d'extension, dans le cas d'IPv6). La somme de contrôle UDP **doit** être mise (même en IPv4 où elle est normalement facultative ; et, pour IPv6, on n'a pas le droit d'utiliser l'exception du RFC 6935).

C'est simple mais il y a quelques petits pièges. D'abord, avec les messages ICMP (section 5.5) : lors de leur réception, il n'y aura pas forcément assez d'octets du message original pour trouver l'association SCTP à qui envoyer le message ICMP. Il faudra alors jeter ce dernier sans remords. La mise en œuvre de SCTP doit donc être capable de se débrouiller sans les messages ICMP, dont la réception ne peut pas être garantie.

Deuxième piège, commun à toutes les techniques d'encapsulation, la MTU (section 5.6). SCTP doit penser à diminuer la MTU de la taille de l'en-tête UDP. Et se souvenir qu'ICMP est encore moins garanti lorsqu'on utilise l'encapsulation UDP donc les procédures de découverte de la MTU du chemin (RFC 4820, RFC 4821 et surtout RFC 8899) doivent fonctionner même si on ne reçoit pas les messages ICMP.

Dernier piège, mais crucial, le contrôle des adresses IP source des paquets UDP. Pour les associations SCTP à une seule adresse IP, il ne faut **pas** indiquer l'adresse IP de la machine dans les sous-paquets ("*chunks*") de type `INIT` (section 3.3.2 du RFC 4960) mais la mettre dans le paquet UDP (ce que fait UDP par défaut). Et si la machine a plusieurs adresses IP, il faut se servir des mécanismes des RFC 5061 et RFC 4895 pour les faire connaître au correspondant, et non pas mettre les adresses IP dans les sous-paquets SCTP car l'intervention d'un routeur NAT modifierait l'adresse source, rendant ces sous-paquets invalides.

Et pour les applications, quels mécanismes sont nécessaires pour faire fonctionner cette encapsulation ? La section 6 propose des changements à l'API du RFC 6458, avec l'ajout d'une option `SCTP_REMOTE_UDP_ENCAPS_PORT` qui prendra en paramètre une structure :

```
struct sctp_udpencaps {
     sctp_assoc_t sue_assoc_id;
     struct sockaddr_storage sue_address;
     uint16_t sue_port;
   };
```

Ce qui permettra de définir/connaître les adresses IP et les ports utilisés par l'encapsulation UDP.

Depuis sa version 9.1, FreeBSD a la capacité d'utiliser cette encapsulation pour envoyer des paquets SCTP. Un portage a été fait vers Mac OS X, disponible en <http://sctp.fh-muenster.de>.