RFC 9868 : Transport Options for UDP

Stéphane Bortzmeyer
<stephane+blog@bortzmeyer.org>

Première rédaction de cet article le 5 décembre 2025

Date de publication du RFC : Octobre 2025

https://www.bortzmeyer.org/9868.html

Des protocoles de transport, comme TCP, ont le concept d'options, ajoutées à l'en-tête et permettant de régler divers paramètres. Ce RFC ajoute ce concept à UDP et standardise quelques options. Désormais, il y a des moyens standards de faire du « ping » en UDP.

Mais comment ajouter des options à un vénérable protocole (UDP a été normalisé en 1980, dans le RFC 768 [1]), qui n'a pas de place pour cela dans son en-tête très minimal ? L'astuce est que UDP indique une taille de paquet qui peut être différente de celle indiquée par IP. Si elle est supérieure, les octets ainsi disponibles peuvent stocker des options, à la fin du paquet. Ce sera donc un pied ("*trailer*") et pas un en-tête.

Voici un paquet UDP simplifié (j'ai aussi mis une partie de l'en-tête IP) analysé par tshark :

```
Internet Protocol Version 6, Src: 2a04:cec0:10fc:5bd8:efd1:79bb:7356:4583, Dst: 2001:db8::1
    0110 .... = Version: 6
...
    Payload Length: 56
    Next Header: UDP (17)
    Hop Limit: 64
    Source Address: 2a04:cec0:10fc:5bd8:efd1:79bb:7356:4583
    Destination Address: 2001:db8::1
User Datagram Protocol, Src Port: 57560 (57560), Dst Port: domain (53)
    Source Port: 57560 (57560)
    Destination Port: domain (53)
    Length: 56
...
    UDP payload (48 bytes)
```

1. Pour voir le RFC de numéro NNN, https://www.ietf.org/rfc/rfcNNN.txt, par exemple https://www.ietf.org/rfc/rfc768.txt

Vous avez vu ? Il y a deux champs qui indiquent la longueur du paquet UDP, un dans l'en-tête IP ("*Payload length*") et un dans l'en-tête UDP ("*Length*"). Ici, les deux sont identiques, ce qui est le cas « normal ». Mais s'ils sont différents ? C'est le point de départ de ce RFC. Les options se nicheront dans la différence entre la longueur du paquet UDP et celle du paquet IP qui l'encapsule, dans une zone connue sous le nom de surplus ("*surplus area*"). Normalement, elles seront ainsi ignorées de tous les vieux programmes (mais il y a parfois eu des bogues <https://svnweb.freebsd.org/base?view=revision&revision=334705>).

Comme TCP (RFC 9293), les protocoles de transport SCTP (RFC 9260) et DCCP (RFC 4340) ont dans leur en-tête un espace réservé pour d'éventuelles options. Pas UDP. C'est ce manque que comble notre RFC. Comme UDP est un protocole sans connexion, ces options ne serviront que pour un seul paquet (alors que TCP peut les garder pour toute la connexion). Des options, dans les autres protocoles de transport, il y en a plein. Pour TCP, le RFC cite par exemple, la dilatation de fenêtres (RFC 7323), l'authentification (RFC 5925) et d'autres. On en trouve pour des protocoles avec état (comme TCP, où elles vont s'appliquer à toute la connexion) ou sans état (comme IP, où elles ne s'appliquent qu'au paquet qui les porte). Comme vu plus haut, seul UDP (RFC 768) n'avait pas le moyen d'ajouter des options.

(Notez que Teredo - RFC 4380 - utilisait déjà un truc analogue.)

À quoi vont servir ces options (section 5 du RFC) ? À court terme, à permettre des mécanismes de fragmentation, de test (« ping UDP »), etc. À plus long terme (mais ce n'est pas encore normalisé), elles rendront possible d'ajouter de l'authentification ou du chiffrement à UDP (DTLS, RFC 9147 chiffre mais ne protège pas l'en-tête de la couche Transport). On pourra aussi faire de la découverte de la MTU du chemin, comme spécifié dans le RFC 9869. Et enfin, cela pourra permettre d'augmenter la taille des paquets DNS comme proposé dans `draft-heard-dnsop-udp-opt-large-dns-responses`.

La section 6 du RFC spécifie le cahier des charges de ces options UDP :

— UDP est sans état (sans connexion) et doit le rester.
— UDP est unidirectionnel et doit le rester. Les protocoles requête/réponse (comme le DNS) sont bâtis sur UDP mais doivent s'occuper de faire correspondre requêtes et réponses sans l'aide d'UDP.
— Les options UDP n'ont pas de taille maximale (à part celle du paquet UDP lui-même, 65 536 octets.
— Les options UDP ne visent pas à remplacer des protocoles existants. Le « ping UDP » ne cherche pas à être équivalent au ping utilisant ICMP. Après tout, UDP est fait pour être minimal, et il doit le rester.
— Les options UDP ne constituent pas un protocole à elles seules : un protocole situé au-dessus d'UDP va devoir spécifier bien des détails (par exemple le RFC 9869 qui décrit le protocole PLPMTUD).
— Tout le mécanisme est conçu pour qu'un logiciel ancien, qui ne connaisse pas ces options, fonctionne comme aujourd'hui.

Un peu de terminologie avant d'attaquer les détails, notamment (section 3) :

— Option sûre ("*SAFE option*") : option qui pourra être ignorée par un receveur qui ne la comprend pas. Elle ne modifie pas la compréhension du paquet.
— Option non sûre ("*UNSAFE option*") : option qui ne doit pas être ignorée, car elle modifie la compréhension du paquet (par exemple car elle indique qu'il est chiffré).

La description technique de l'emplacement des options figure en section 7 du RFC. Le champ Longueur de l'en-tête UDP, qui a toujours été redondant avec celui d'IP, peut désormais prendre une valeur inférieure à celle du champ dans IP (mais supérieure à huit, la taille de l'en-tête UDP). La différence entre les deux longueurs est la taille du surplus, la zone à la fin du paquet où se logent les options (qui ne sont donc pas dans l'en-être mais dans le pied). Notez qu'il n'y a pas d'obligation que le surplus soit aligné sur des multiples (2 ou 4) d'octets. Si les longueurs IP et UDP sont identiques, c'est qu'il n'y a pas d'options.

La structuration du surplus en options est dans la section 8. Le surplus commence avec un champ de deux octets qui est une somme de contrôle. Cet OCS ("*Option CheckSum*") est une somme de contrôle Internet traditionnelle (RFC 791, section 3.1, et RFC 1071). Pourquoi cette somme de contrôle alors qu'UDP en a déjà une ? Pour détecter les cas où un autre mécanisme jouerait avec la différence des champs Longueur d'IP et d'UDP et aurait son propre pied, distinct de celui normalisé dans notre RFC. Simple somme de contrôle, l'OCS ne joue pas un rôle en sécurité, il ne sert qu'à vérifier que le surplus est bien utilisé conformément à la norme sur les options UDP. (Les options sont ignorées si l'OCS n'est pas correcte, voir la section 14.) La section 18 insiste sur ce point : comme cette distinction entre deux champs Longueur est ancienne, il est possible que certains s'en servaient et il faut donc faire face à la possibilité que des paquets UDP aient des champs Longueur différents sans pour autant avoir des options telles que normalisées ici. L'OCS est donc aussi une sorte de nombre magique mais en plus fort, puisqu'il n'est pas statique.

Comme toujours, il faut prévoir le cas de "*middleboxes*" boguées qui calculeraient la somme de contrôle UDP sur tout le paquet IP, au lieu de s'arrêter à la longueur indiquée dans l'en-tête UDP. L'OCS est conçu pour annuler l'effet d'une telle bogue (merveille des sommes de contrôle, qui n'ont pas de dispersion des résultats). La somme de contrôle UDP étant optionnelle (mais c'est évidemment très déconseillé), l'OCS l'est aussi (et peut donc valoir zéro). Rappelez-vous que ce qui marchait avant en UDP doit encore marcher avec les options donc, par défaut, même si l'OCS est incorrect, le paquet doit être transmis aux applications réceptrices.

À propos de "*middleboxes*", le RFC précise aussi (section 16) que les options sont de bout en bout et ne doivent pas être modifiées par les équipements intermédiaires (mais cela risque de rester un vœu pieux.) Des tests effectués ont montré que des systèmes d'exploitation courants (Linux, MacOS, Windows) n'avaient pas de problème en recevant des paquets UDP contenant des options (section 18). Ils n'envoient à l'application que la partie du paquet indiquée par le champ Longueur d'UDP, et ignorent donc les options, qu'on peut donc envoyer sans craindre de perturber les systèmes pré-existants. Toutefois, au moins un objet connecté (non nommé par le RFC) passe tout le datagramme (y compris, donc, le surplus contenant les options) à l'application. Et au moins un IDS, l'Alcatel-Lucent Brick considère les paquets UDP ayant des options comme des attaques (les IDS sont un des facteurs importants de l'ossification de l'Internet car ils interprètent tout ce qui ne colle pas à leur vision étroite comme une attaque).

Les options suivent la somme de contrôle et la plupart sont encodées selon un classique schéma TLV, comme pour la plupart des options TCP (RFC 9293, section 3.1). Les deux exceptions à cet encodage TLV sont les options NOP (un seul octet, qui vaut un, cette option, comme son nom l'indique, ne sert à rien, à part éventuellement à aligner les données) et EOL ("*End Of Options*", un seul octet, qui vaut zéro, elle indique la fin de la liste d'options). Autrement, les options sont en TLV, le premier octet indiquant le type, la longueur faisant un octet (un mécanisme existe pour des options faisant plus de 255 octets). Les options sûres ont un type de 0 à 191, les autres sont non sûres. Parmi les options sûres enregistrées par notre RFC, outre EOL et NOP, on trouve entre autres (section 11) :

— APC ("*Additional Payload Checksum*"), type 2, qui ajoute une somme de contrôle plus forte (CRC32c, la même que celle utilisée dans iSCSI, RFC 3385 et SCTP).
— FRAG ("*Fragmentation*"), type 3, qui permet une fragmentation au niveau UDP. Peut se combiner avec MDS ("*Maximum Datagram Size*", type 4), qui indique la taille maximale qu'un récepteur peut accepter sans que le datagramme soit fragmenté.
— REQ ("*Echo Request*", type 6) et RES ("*Echo Response*", type 7) permettent de faire un « ping UDP ». (Plus propre que les trucs abusivement baptisés « ping UDP » comme cet exemple `<https://fr.linux-console.net/?p=19728>` ou celui-ci `<https://serverfault.com/questions/193425/verify-connectivity-to-a-server-on-a-udp-port>`.)
— TIME, type 8, permet d'envoyer et de recevoir l'heure, sous forme d'une estampille temporelle de quatre octets. C'est l'analogue de l'option TCP Timestamp (RFC 7323, section 3). UDP ne garantit pas que ces estampilles correspondent à l'heure officielle, juste qu'elles seront monotones (toujours croissantes).

— AUTH ("*Authentication*"), n'est pas réellement spécifiée. Son type, 9, est réservé mais les autres points sont repoussés à un futur RFC, peut-être à partir de l'"*Internet-Draft*" `draft-touch-tsvwg-udp-auth`

Les options EOL, NOP et quelques autres doivent normalement être reconnues par toutes les mises en œuvres des options UDP (cf. section 14).

Et les options non sûres ? Rappelons que ce sont celles qui modifient l'interprétation du contenu du paquet. On y trouve (section 12 du RFC) entre autres :

— UCMP ("*Unsafe Compression*"), type 192, pour des contenus comprimés.
— UENC ("*Unsafe Encryption*"), type 193, pour des contenus chiffrés.

Ces deux options ne sont pas autrement spécifiées, seul un type est réservé. Le reste devra être précisé dans un RFC ultérieur.

De nouvelles options seront peut-être rajoutées dans le futur au registre des options `<https://www.iana.org/assignments/udp/udp.xml#udp-options>` (section 13 du RFC). La politique IANA (section 26) est « action de normalisation » ou « approbation par l'IESG », donc des procédures assez lourdes (RFC 8126) : il ne sera pas facile d'ajouter des options.

Question mise en œuvre, ce n'est pas forcément trivial à faire soi-même. Pour pouvoir lire et à plus forte raison écrire des options UDP, il faut actuellement court-circuiter UDP et écrire directement au-dessus d'IP ("*raw sockets*") ou pire, dans certains cas, directement sur le réseau. Notre RFC propose des extensions à l'API "*socket*" pour lire et écrire les options (avec aussi des variables sysctl) mais je ne connais pas encore de système d'exploitation qui les intègre. Notez que ces API ne permettent pas d'influencer l'ordre des options, et c'est exprès.

Une difficulté supplémentaire vient du fait qu'UDP est sans état, sans connexion. D'une part, les options ne peuvent pas être négociées lors de l'établissement de la connexion (puisqu'il n'y a pas de connexion), d'autre part, elles doivent être répétées dans chaque paquet (puisqu'il n'y a pas d'état). Le RFC précise donc (section 19) qu'il faut ignorer les options qu'on ne connait pas (du moins celles marquées comme sûres). D'autre part il impose qu'on passe les données à l'application même si on ne comprend pas les options. Ah, et aussi, les options UDP sont conçues pour l'"*unicast*" essentiellement.

Et leurs conséquences pour la sécurité (section 25 du RFC) ? Fondamentalement, les options UDP ne posent pas plus (ou pas moins…) de problèmes de sécurité que les options TCP, qui sont normalisées depuis longtemps. Comme pour TCP, elles ne sont **pas** sécurisées par TLS (ici, DTLS, RFC 9147), TLS qui protège uniquement les données applicatives. Si vous voulez les protéger, il faudra attendre la spécification des futures options AUTH (qui fournira à peu près le même service que le AO de TCP, normalisé dans le RFC 5925) et UENC. Ou alors, protégez toute la couche transport avec IPsec (RFC 4301).

Mais il y a un autre danger, dans la façon dont vont être traitées les options UDP par le récepteur. Si celui-ci boucle imprudemment, en attendant juste l'option EOL, le risque de débordement est élevé. Ou si le récepteur alloue de la mémoire en utilisant le champ Longueur d'UDP puis y copie tout le paquet, les options faisant alors déborder son tampon.

Voilà, je n'ai pas tout détaillé, la section 25 est longue, lisez-la.

Un petit point d'histoire (section 4) : pourquoi diable UDP a-t-il ce champ Longueur, alors que celui d'IP suffisait, d'autant plus que l'en-tête UDP est de taille fixe ? Aucun autre protocole de transport ne fait cela (à part peut-être Teredo, RFC 6081). Le RFC note que des historiens de l'Internet ont été consultés mais sans résultat. Les raisons de ce choix (qu'on apprécie aujourd'hui mais qui est resté sans utilité pratique pendant 45 ans) restent mystérieuses. Permettre de placer plusieurs paquets UDP dans

un seul datagramme IP ? Pour remplir les paquets de manière à ce qu'ils soient alignés sur des multiples d'octets ? Aucune explication ne semble satisfaisante et les archives sont muettes.

Notez aussi que d'autres propositions avaient été faites pour ajouter des options à UDP, comme celle décrite dans l'"*Internet Draft*" `draft-hildebrand-spud-prototype`.

Autre question que vous vous posez peut-être : et UDP-Lite (RFC 3828) ? Lui aussi jouait avec la différence des champs Longueur d'IP et d'UDP, pour indiquer la partie du paquet couverte par la somme de contrôle. De ce fait, le mécanisme d'options ne peut pas être utilisé pour UDP-Lite (section 17).

Et les mises en œuvre concrètes ? Il n'existe pas, à ma connaissance <https://mailarchive.ietf.org/arch/msg/tsvwg/4GbSb-xFjxCMNXv0rt3kiuIaudQ/>, de serveurs de test publics sur l'Internet. Mais vous avez une liste des programmes existants <https://github.com/tsvwg/draft-ietf-tsvwg-ud blob/main/Implementation.txt> (je n'en ai pas vu en Python utilisant Scapy, ce qui serait pourtant une expérience intéressante).

D'autres lectures sur ces options UDP ? Vous avez l'article de Raffaele Zullo <https://dl.ifip.org/db/conf/tma/tma2020/tma2020-camera-paper70.pdf> et les supports d'un de ses exposés <http://middleboxes.org/raffaelezullo/publications/tma2020-zullo-udp-options-slides.pdf>, qui expliquent bien la question. Et, sinon ChatGPT, consulté s'en ne s'en est pas trop mal tiré, inventant un mécanisme qui ressemble à celui de TCP.