RFC 8724 : SCHC: Generic Framework for Static Context Header Compression and Fragmentation

Stéphane Bortzmeyer
<stephane+blog@bortzmeyer.org>

Première rédaction de cet article le 16 avril 2020

Date de publication du RFC : Avril 2020

https://www.bortzmeyer.org/8724.html

Ce nouveau RFC décrit un mécanisme de compression des données, et de fragmentation des paquets, optimisé pour le cas des réseaux à longue distance lents connectant des objets contraints en ressources, ce qu'on nomme les LPWAN ("*Low-Power Wide Area Network*", par exemple LoRaWAN.) L'un des buts est de permettre l'utilisation des protocoles Internet normaux (UDP et IPv6) sur ces LPWAN. SCHC utilise un contexte de compression et décompression **statique** : il n'évolue pas en fonction des données envoyées.

Le LPWAN (RFC 8376[1]) pose en effet des défis particuliers. Il a une faible capacité <https://www.bortzmeyer.org/capacite.html>, donc chaque bit compte. Il relie des objets ayant peu de ressources matérielles (par exemple un processeur très lent). Et la batterie n'a jamais assez de réserves, et émettre sur un lien radio coûte cher en énergie (1 bit reçu ou transmis = 1-1000 microjoules, alors qu'exécuter une instruction dans le processeur = 1-100 nanojoules.) . Le but de l'IETF est de pouvoir utiliser IPv6 sur ces LPWAN et la seule taille de l'en-tête IPv6 est un problème : 40 octets, dont plusieurs champs « inutiles » ou redondants, et qui auraient donc tout intérêt à être comprimés.

Trois choses importantes à retenir sur SCHC ("*Static Context Header Compression*") :

— N'essayez pas de prononcer le sigle « èsse cé hache cé » : on dit « chic » en français, et « sheek » en anglais.
— Comme tout algorithme de compression, SCHC repose sur un **contexte** commun au compresseur et au décompresseur, contexte qui regroupe l'ensemble des règles suivies pour la compression et la décompression. Mais, contrairement aux mécanismes où le contexte est dynamiquement modifié en fonction des données transmises, ici le contexte est **statique** ; il n'évolue pas avec les données. Cela élimine notamment tout risque de désynchronisation entre émetteur et récepteur. (Contrairement à des protocoles comme ROHC, décrit dans le RFC 5795.)

1. Pour voir le RFC de numéro NNN, https://www.ietf.org/rfc/rfcNNN.txt, par exemple https://www.ietf.org/rfc/rfc8376.txt

— SCHC n'est pas un protocole précis, c'est un cadre générique, qui devra être incarné dans un protocole pour chaque type de LPWAN. Cela nous promet de nouveaux RFC dans le futur. L'annexe D liste les informations qu'il faudra spécifier dans ces RFC.

Si les LPWAN posent des problèmes particuliers, vu le manque de ressources disponibles, ils ont en revanche deux propriétés qui facilitent les choses :

— Topologie simple, avec garantie que les paquets passent au même endroit à l'aller et au retour,
— Jeu d'applications limité, et connu à l'avance, contrairement à ce qui arrive, par exemple, avec un ordiphone.

La section 5 de notre RFC expose le fonctionnement général de SCHC. SCHC est situé entre la couche 3 (qui sera typiquement IPv6) et la couche 2, qui sera une technologie LPWAN particulière, par exemple LoRaWAN (pour qui SCHC a déjà été mis en œuvre). Après compression, le paquet SCHC sera composé de l'identificateur de la règle ("*RuleID*"), du résultat de la compression de l'en-tête, puis de la charge utile du paquet. Compresseur et décompresseur doivent partager le même ensemble de règles, le **contexte**. Le contexte est défini de chaque côté (émetteur et récepteur) par des mécanismes non spécifiés, par exemple manuellement, ou bien par un protocole d'avitaillement privé. Chaque règle est identifiée par son "*RuleID*" (section 6 du RFC), identificateur dont la syntaxe exacte dépend d'un profil de SCHC donc en pratique du type de LPWAN. (Rappelez-vous que SCHC est un mécanisme générique, les détails concrets de syntaxe sont spécifiés dans le profil, pas dans SCHC lui-même.)

Les règles sont expliquées dans la section 7. Chaque règle comporte plusieurs descriptions. Chaque description comprend notamment :

— L'identificateur du champ concerné (SCHC ne traite pas l'en-tête globalement, mais champ par champ), par exemple « port de destination UDP ».
— Longueur et position du champ.
— Valeur cible (TV, "*Target Value*"), qui est la valeur à laquelle on va comparer le champ.
— Opérateur de comparaison (MO, pour "*Matching Operator*"), l'égalité complète, par exemple, ou bien l'égalité de seulement les N bits de plus fort poids.
— Action (CDA, "*Compression Decompression Action*"), qui indique quoi faire si le contenu du champ correspond à la valeur cible. Par exemple, ne pas transmettre le champ (si sa valeur est une valeur connue), ou bien (pour le décompresseur) recalculer la valeur à partir des données. La liste des actions possibles figure dans un tableau en section 7.4. (Elle est fixée une fois pour toutes, il n'est pas prévu de procédure d'enregistrement de nouvelles possibilités.)

L'algorithme est donc : pour chaque description dans une règle, voir si elle correspond au champ visé, en utilisant l'opérateur de comparaison et la valeur cible. Si toutes les descriptions collent, appliquer les actions, et envoyer les données comprimées, précédées de l'identificateur de la règle. Un "*RuleID*" spécial est utilisé pour attraper tout le reste, les paquets qui ne seront pas comprimés car aucune règle ne correspondait.

Prenons un exemple : la règle de "*RuleID*" 1 a deux descriptions, qui disent que les champs X et Y de l'en-tête ont une valeur connue (indiquée dans la valeur cible), mettons respectivement 42 et « foobar ». Dans ce cas, les actions de compression (CDA, "*Compression Decompression Action*") peuvent être simplement « omets ces champs » (`not-sent`). Le décompresseur, à l'autre bout, a la même règle (rappelez-vous que le contexte, l'ensemble des règles, est statique). Lorsque qu'il voit passer un paquet comprimé avec la règle 1, il crée simplement les deux champs, avec les valeurs définies dans la règle (valeurs TV, "*Target Value*".)

Un exemple figure en section 10 du RFC, avec des règles pour comprimer et décomprimer les entêtes IPv6 et UDP. Ainsi, le champ Version d'IPv6 vaut forcément 6. On met donc la valeur cible (TV) à 6, l'opérateur de comparaison (MO) à « ignorer » (on ne teste pas l'égalité, on est sûr d'avoir 6, si le paquet est correct), et l'action (CDA) à `not-sent` (ne pas envoyer). Le champ Longueur, par contre, n'a pas de valeur cible, l'opérateur de comparaison est « ignorer », et l'action est `compute` (recalculer à partir des données).

Pour UDP, on peut également omettre les ports source et destination si, connaissant l'application, on sait qu'ils sont fixes et connus, et on peut également recalculer le champ Longueur, ce qui évite de le transmettre. La somme de contrôle est un peu plus compliquée. IPv6 en impose une (RFC 8200, section 8.1) mais autorise des exceptions. Ne pas l'envoyer peut exposer à des risques de corruption de données, donc il faut bien lire le RFC 6936 et le RFC 6282 avant de décider d'ignorer la somme de contrôle. Les règles complètes pour UDP et IPv6 sont rassemblées dans l'annexe A du RFC.

Outre la compression, SCHC permet également la fragmentation (section 8 du RFC). La norme IPv6 (RFC 8200) dit que tout lien qui fait passer de l'IPv6 doit pouvoir transmettre 1 280 octets. C'est énorme pour du LPWAN, où la MTU n'est parfois que de quelques dizaines d'octets. Il faut donc effectuer fragmentation et réassemblage dans la couche 2, ce que fait SCHC (mais, désolé, je n'ai pas creusé cette partie, pourtant certainement intéressante .)

La section 12 de notre RFC décrit les conséquences de SCHC sur la sécurité. Par exemple, un attaquant peut envoyer un paquet avec des données incorrectes, dans l'espoir de tromper le décompresseur, et, par exemple, de lui faire faire de longs calculs, ou bien de générer des paquets de grande taille, pour une attaque avec amplification. Comme toujours, le décompresseur doit donc se méfier et, entre autres, ne pas générer de paquets plus grands qu'une certaine taille. Question sécurité, on peut aussi noter que SCHC n'est pas vulnérable aux attaques comme CRIME ou BREACH, car il traite les différents champs de l'en-tête séparément.

La fragmentation et le réassemblage amènent leurs propres risques, qui sont bien connus sur l'Internet (d'innombrables failles ont déjà été trouvées dans les codes de réassemblage de paquets fragmentés.) Par exemple, une attaque par déni de service est possible en envoyant plein de fragments, sans jamais en envoyer la totalité, forçant le récepteur à consommer de la mémoire pour stocker les fragments en attente de réassemblage. Là encore, le récepteur doit être prudent, voire paranoïaque, dans son code. Par contre, les attaques utilisant la fragmentation pour se dissimuler d'un IDS ne marcheront sans doute pas, puisque SCHC n'est utilisé qu'entre machines directement connectées, avec probablement aucun IDS sur le lien.

Merci à Laurent Toutain pour avoir attrapé une sérieuse erreur dans cet article et à Dominique Barthel pour sa relecture très attentive.