Groupe de travail Réseau
Request for Comments : 3545
Catégorie : Sur la voie de la normalisation

Traduction Claude Brière de L'Isle

T. Koren, Cisco Systems
S. Casner, Packet Design
J. Geevarghese, Motorola India Electronics Ltd.
B. Thompson & P. Ruddy, Cisco Systems
juillet 2003

RTP compressé (CRTP) amélioré pour liaisons à fort retard, perte et déclassement de paquet

Résumé
Le présent document décrit un schéma de compression d'en-tête pour liaisons point à point avec perte de paquets et longs délais. Il se fonde sur le protocole de transport compressé en temps réel (CRTP), la compression d'en-tête IP/UDP/RTP décrite dans la RFC 2508. CRTP ne fonctionne pas bien sur de telles liaisons : les pertes de paquets résultent en une corruption de contexte et à cause des longs délais, de nombreux paquets sont éliminés avant que le contexte soit réparé. Pour corriger le comportement de CRTP sur de telles liaisons, on spécifie ici quelques extensions au protocole. Les extensions visent à réduire la corruption de contexte en changeant la façon dont le compresseur met à jour le contexte au décompresseur : les mises à jour sont répétées et incluent des mises à jour des paramètres de contexte pleins et différentiels. Avec ces extensions, CRTP fonctionne bien sur des liaisons à pertes de paquets, déclassement de paquets et longs délais.

Table des Matières

1. Introduction

La compression d'en-tête RTP (CRTP) décrite dans la RFC 2508 a été conçue pour réduire les frais généraux d'en-têtes des datagrammes IP/UDP/RTP en compressant les trois en-têtes. Les en-têtes IP/UDP/RTP sont compressés à 2 à 4 octets la plupart du temps.

CRTP a été conçu pour des liaisons point à point fiables avec des délais courts. Il ne fonctionne pas bien sur des liaisons avec de forts taux de perte de paquets, un déclassement des paquet et de longs délais.

Un exemple d'une telle liaison est une session PPP qui est tunnelée en utilisant un protocole de tunnelage de niveau IP comme L2TP. Les paquets dans le tunnel sont portés par un réseau IP et donc peuvent être perdus et déclassés. Plus long est le tunnel, plus long est le délai d'aller retour.

Un autre exemple est un réseau IP qui utilise des technologies de couche 2 comme ATM et le relais de trame pour la portion accès du réseau. Les réseaux de transport de couche 2 comme ATM et le relais de trame se comportent comme des liaisons point à point en série en ce qu'elle ne réordonnent pas les paquets. De plus, les circuits virtuels en relais de trame et ATM utilisés comme technologies d'accès IP ont souvent un faible débit binaire. Ces circuits virtuels diffèrent des liaisons en série à faible vitesse en ce qu'ils s'étendent sur une plus grande distance physique que la liaison point à point en série. Le retard de la vitesse de la lumière au sein d'un réseau de transport de couche 2 va résulter en de plus forts délais d'aller retour entre les points d'extrémité du circuit. De plus, l'encombrement au sein du réseau de transport de couche 2 peut résulter en un taux d'abandon effectif pour le circuit virtuel significativement supérieur aux taux d'erreurs normalement rencontrés sur les liaisons séries point à point.

Il peut être souhaitable d'étendre les mises en œuvre existantes de CRTP pour les utiliser aussi sur les tunnels IP et autres circuits virtuels, où la perte de paquets, le réarrangement, et de long délais sont des caractéristiques communes. Pour traiter ces scénarios, le présent document définit des modifications et extensions à CRTP pour accroître la robustesse à la perte de paquet et au mauvais ordre entre le compresseur et le décompresseur. Cela se fait en répétant les mises à jour et en permettant l'envoi de valeurs absolues (non compressées) en plus des valeurs de delta pour les paramètres de contexte. Bien que ces nouveaux mécanismes imposent des frais généraux supplémentaires, la compression globale est quand même encore substantielle. Le CRTP amélioré, comme défini dans le présent document, convient donc pour de nombreuses applications dans les scénarios discutés ci-dessus, par exemple, le tunnelage et autres circuits virtuels.

La RFC 3095 définit un autre schéma de compression d'en-tête RTP appelé compression d'en-tête robuste (ROHC, *Robust Header Compression*) [RFC3095]. ROHC a été développé avec les liaisons sans fil comme cible principale, et introduit de nouveaux mécanismes de compression avec comme principal objectif de réaliser la combinaison de la robustesse contre la perte de paquet et une efficacité maximale de compression. ROHC est supposé être le mécanisme de compression préféré sur les liaisons où l'efficacité de compression est importante. Cependant, ROHC a été conçu avec les mêmes hypothèses de liaisons que CRTP, par exemple, que le schéma de compression NE DEVRAIT PAS avoir à tolérer de désordre des paquets compressés entre le compresseur et le décompresseur, ce qui peut arriver quand les paquets sont portés dans un tunnel IP sur plusieurs bonds.

À l'avenir, des améliorations pourront être définies pour ROHC afin de lui permettre de bonnes performances en présence de désordre des paquets compressés. Le résultat pourrait être plus efficace que le protocole de compression spécifié dans le présent document. Cependant, il y a de nombreux environnements pour lesquels le CRTP amélioré défini ici peut être préféré. En particulier, pour les environnements où CRTP est déjà mis en œuvre, l'effort supplémentaire requis pour mettre en œuvre les extensions définies ici est supposé petit. Il y a aussi des cas où la simplicité de mise en œuvre de ce CRTP amélioré par rapport à ROHC est plus importante que les avantages de performances de ROHC.

1.1 Fonctionnement de CRTP

Durant la compression d'un flux RTP, un contexte de session est défini. Pour chaque contexte, l'état de session est établi et partagé par le compresseur et le décompresseur. Une fois l'état de contexte établi, les paquets compressés peuvent être envoyés.

L'état du contexte consiste en les en-têtes IP/UDP/RTP complets, quelques valeurs différentielles de premier ordre, un numéro de séquence de liaison, un numéro de génération et un tableau de codage des deltas.

La partie en-tête du contexte est établie par le paquet FULL_HEADER qui commence toujours une session de compression. Les valeurs différentielles de premier ordre (valeurs de deltas) sont établies en envoyant des paquets COMPRESSED_RTP qui comportent les mises à jour des valeurs de deltas.

L'état du contexte DOIT être synchronisé entre le compresseur et le décompresseur pour qu'ait lieu une décompression réussie. Si le contexte n'est pas synchronisé, le décompresseur n'est pas capable de restaurer avec précision les en-têtes compressés. Le décompresseur invalide le contexte et envoie un paquet CONTEXT_STATE au compresseur pour indiquer que le contexte a été corrompu. Pour reprendre la compression, le compresseur DOIT rétablir le contexte.

Durant la période où le contexte est corrompu, le décompresseur élimine tous les paquets reçus pour ce contexte. Comme le mécanisme de réparation du contexte dans CRTP implique un retour de la part du décompresseur, la réparation de contexte prend au moins autant de temps que le délai d'aller retour de la liaison. Si le délai d'aller retour de la liaison est long, et en particulier si la bande passante de la liaison est élevée, de nombreux paquets vont être éliminée avant que le contexte soit réparé. Sur de telles liaisons, il est souhaitable de minimiser l'invalidation de contexte.

1.2 Comment est corrompu un contexte ?

Tant que les champs dans les en-tête combinés IP/UDP/RTP changent comme attendu pour la séquence de paquets d'une session, ces en-têtes peuvent être compressés, et le décompresseur peut pleinement restaurer les en-têtes compressés en utilisant l'état du contexte. Quand les en-têtes ne changent pas comme prévu, il est nécessaire de mettre à jour certaines des valeurs pleines ou de delta du contexte. Par exemple, l'horodatage RTP est supposé s'incrémenter du delta d'horodatage RTP (dT). Si la suppression de silence est utilisée, les paquets ne sont pas envoyés durant les périodes de silence. Alors quand l'activité vocale reprend, les paquets sont envoyés à nouveau, mais l'horodatage RTP est incrémenté d'une grande valeur et pas de dT. Dans ce cas, une mise à jour DOIT être envoyée.

Si un paquet qui comporte une mise à jour de certaines valeurs d'état du contexte est perdu, l'état au décompresseur n'est pas mis à jour. L'état partagé est alors différent au compresseur et au décompresseur. Quand le prochain paquet arrive au décompresseur, le décompresseur va échouer à restaurer précisément les en-têtes compressés car l'état du contexte au décompresseur est différent de l'état au compresseur.

1.3 Empêcher la corruption du contexte

Noter que le décompresseur n'échoue pas quand un paquet est perdu, mais quand le prochain paquet compressé arrive. Si le prochain paquet se trouve comporter la même mise à jour de contexte que dans le paquet perdu, le contexte au décompresseur peut être mis à jour avec succès et la décompression peut continuer sans interruption. Si le paquet perdu incluait une mise à jour d'un champ delta comme le delta d'horodatage RTP (dT), le prochain paquet ne peut pas compenser la perte car la mise à jour d'une valeur delta est relative au paquet précédent qui a été perdu. Mais si la mise à jour est pour une valeur absolue comme un plein horodatage RTP ou le type de charge utile RTP, cette mise à jour peut être répétée dans le paquet suivant indépendamment du paquet perdu. Donc, il est utile d'être capable de mettre à jour les valeurs absolues du contexte.

La section suivante décrit plusieurs extensions à CRTP qui ajoutent la capacité de mettre à jour de façon sélective les valeurs absolues du contexte, plutôt que d'envoyer un paquet FULL_HEADER, en plus des mises à jour existantes des valeurs de delta. Cette version améliorée de CRTP est destinée à minimiser les invalidations de contexte et donc améliorer les performances sur des liaisons à perte avec de longs délais d'aller retour.

1.4 Spécification des exigences

Les mots clés "DOIT", "NE DOIT PAS", "EXIGE", "DEVRA", "NE DEVRA PAS", "DEVRAIT", "NE DEVRAIT PAS", "RECOMMANDE", "PEUT", et "FACULTATIF" en majuscules dans ce document sont à interpréter comme décrit dans le BCP 14, [RFC2119].

2. CRTP amélioré

La présente section spécifie les changements introduits dans cette version améliorée de CRTP. Ce sont :

- Des extensions au paquet COMPRESSED_UDP pour permettre de mettre à jour les valeurs RTP différentielles dans le contexte du décompresseur et de mettre à jour de façon sélective l'identifiant IPv4 absolu et les valeurs RTP suivantes : numéro de séquence, horodatage, type de charge utile, compte de sources contributives (CSRC) et liste de CSRC. Cela permet de conserver la synchronisation du contexte même avec quelques pertes de paquets.

- Une "somme de contrôle d'en-tête" à insérer par le compresseur et à retirer par le décompresseur lorsque la somme de contrôle UDP n'est pas présente afin que la validation des en-têtes décompressés soit toujours possible. Cela permet au décompresseur de vérifier que la synchronisation de contexte n'a pas été perdue après une perte de paquet.

Un algorithme est ensuite décrit pour utiliser ces changements avec des mises à jour répétées pour réaliser un fonctionnement robuste sur les liaisons avec pertes de paquets et longs délais.

2.1 Paquets COMPRESSED_UDP étendus

Il est possible de s'accommoder de quelques pertes de paquets entre le compresseur et le décompresseur en utilisant l'algorithme "deux fois" du paragraphe 10.1 de la RFC 2507 pour autant que le contexte reste synchronisé. Dans cet algorithme, les valeurs de delta sont ajoutées deux fois (ou plus) au précédent contexte pour effectuer le changement qui se

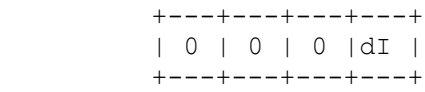
serait produit si les paquets manquants étaient arrivés. Le résultat est vérifié avec la somme de contrôle UDP. Garder la synchronisation du contexte exige de communiquer de façon fiable la valeur absolue et la valeur de delta chaque fois que la valeur de delta change. Pour de nombreux environnements, une fiabilité suffisante peut être obtenue en répétant la mise à jour avec chacun des paquets successifs.

Le paquet COMPRESSED_UDP satisfait au besoin de communiquer les valeurs absolues des champs différentiels RTP, mais il est spécifié dans la RFC 2508 qu'il réinitialise le delta d'horodatage RTP. Cette limitation peut être supprimée avec le changement simple suivant : la RFC 2508 décrit le format de COMPRESSED_UDP comme étant le même que celui de COMPRESSED_RTP sauf que les bits M, S et T sont toujours à 0 et que les champs de delta correspondants ne sont jamais inclus. Cette version améliorée de CRTP change cette spécification pour dire que le bit T PEUT n'être pas à zéro pour indiquer que le delta d'horodatage RTP est inclus explicitement plutôt que d'être remis à zéro.

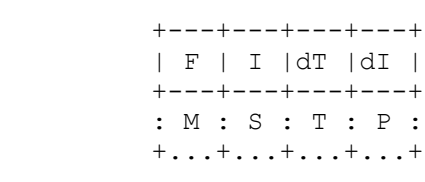
Un second changement ajoute un autre octet de bits de fanions au paquet COMPRESSED_UDP pour permettre de n'inclure que les champs individuels non compressés de l'en-tête RTP dans le paquet plutôt que de porter l'en-tête RTP complet au titre des données UDP. Les fanions supplémentaires augmentent bien un peu la complexité du calcul, mais l'augmentation en efficacité correspondante est importante quand les mises à jour des champs différentiels sont communiquées plusieurs fois dans des paquets COMPRESSED_UDP successifs. Avec ce changement, il y a des bits fanions pour indiquer l'inclusion des valeurs de delta et des valeurs absolues, de sorte que la nomenclature des fanions est changée. Les bits originaux S, T, I qui indiquent l'inclusion des deltas sont rebaptisés dS, dT, dI, et l'inclusion des valeurs absolues est indiquée par S, T, I. Le bit M est absolu comme avant. Un nouveau fanion P indique l'inclusion de la valeur de type de charge utile RTP absolue et un autre fanion indique l'inclusion du compte de CSRC (CC). Quand C=1, un octet supplémentaire est ajouté à la suite des deux octets de fanions pour inclure la valeur absolue du champ CC de quatre bits dans l'en-tête RTP.

Le dernier des trois changements au paquet COMPRESSED_UDP traite de la mise à jour du champ Identifiant IPv4. Pour ce champ, le paquet COMPRESSED_UDP spécifié dans la RFC 2508 peut déjà porter une nouvelle valeur pour le delta d'identifiant IPv4, mais pas la valeur absolue qui est seulement portée par le paquet FULL_HEADER. Donc, un nouveau fanion I est ajouté au paquet COMPRESSED_UDP pour indiquer l'inclusion de la valeur absolue d'identifiant IPv4. Le fanion I remplace le fanion dS qui n'est pas nécessaire dans le paquet COMPRESSED_UDP car le delta de numéro de séquence RTP reste toujours à 1 dans le contexte du décompresseur et donc n'a pas besoin d'être mis à jour. Noter que IPv6 n'a pas de champ ID IP, de sorte que lors de la compression de paquets IPv6 les fanions I et dI sont toujours réglés à 0.

Le format de l'octet fanions/séquence pour le paquet original COMPRESSED_UDP est montré ci-dessous pour référence :



La nouvelle définition de l'octet fanions/séquence plus un octet d'extension de fanions pour le paquet COMPRESSED_UDP est comme suit, où le nouveau fanion F indique l'inclusion de l'octet de fanions d'extension :

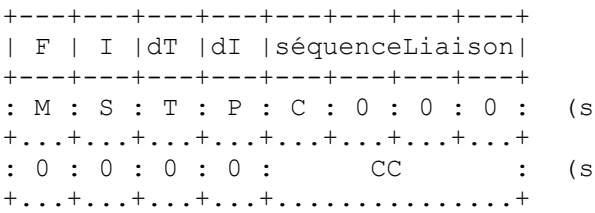


dI = delta d'identifiant IPv4
dT = delta d'horodatage RTP
I = identifiant IPv4 absolu
F = octet de fanions supplémentaire
M = bit marqueur
S = numéro de séquence RTP absolu
T = horodatage RTP absolu
P = type de charge utile RTP
C = compte de CSRC
CID = identifiant de contexte

Quand F=0, il y a seulement un octet de fanions, et les seuls fanions disponibles sont : dI, dT et I. Dans ce cas, le paquet inclut l'en-tête RTP complet. Comme dans la RFC 2508, si dI=0, le décompresseur ne change pas deltaI. Si dT=0, le décompresseur règle deltaT à 0.

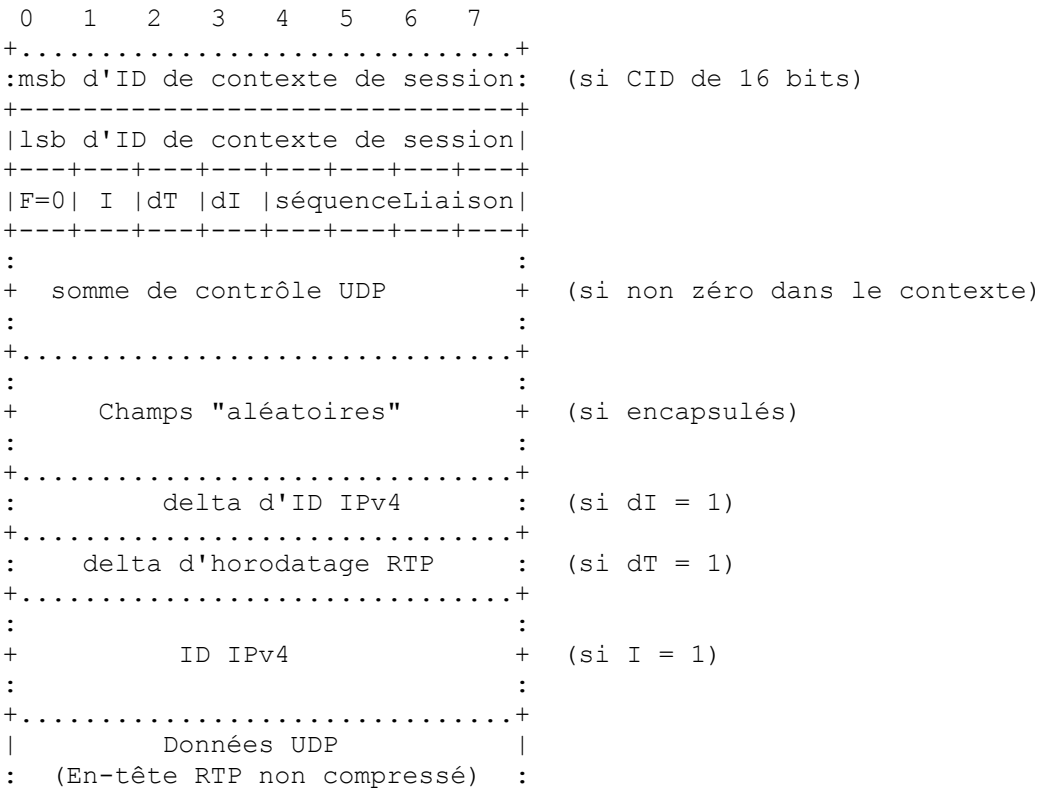
Quand C=1, un octet supplémentaire est ajouté à la suite des deux octets de fanions. Cet octet inclut le CC, le compte

d'identifiants de CSRC, dans ses 4 bits de moindre poids:



Les bits marqués "0" dans le second octet de fanions et l'octet CC DEVRAIENT être réglés à zéro par l'envoyeur et DEVRAIENT être ignorés par le receveur.

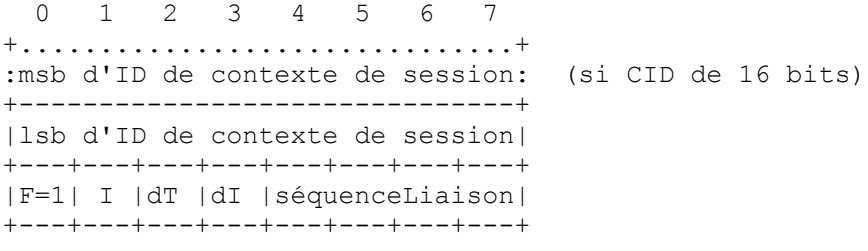
Des exemples de format de paquet illustrent l'utilisation des nouveaux fanions. D'abord, quand F=0, le paquet "traditionnel" COMPRESSED_UDP qui porte l'en-tête RTP complet au titre des données UDP :

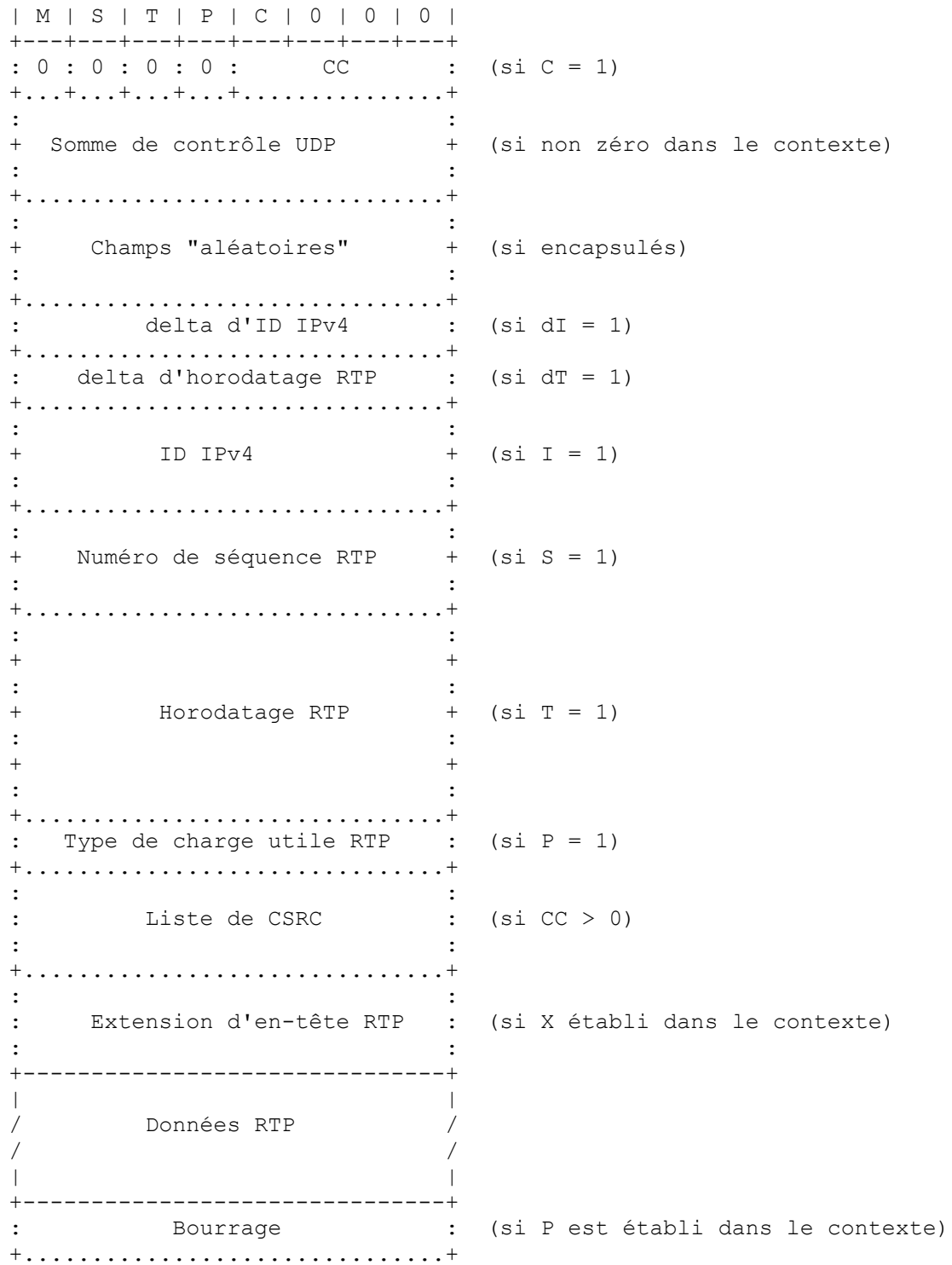


Quand F=1, il y a un octet de fanions supplémentaire et les fanions disponibles sont : dI, dT, I, M, S, T, P, C. Si C=1, il y a un octet supplémentaire qui inclut le nombre d'identifiants de CSRC. Quand F=1, le paquet n'inclut pas l'en-tête RTP complet, mais inclut des champs choisis de l'en-tête RTP comme spécifiés par les fanions. Comme dans la RFC 2508, si dI=0 le décompresseur ne change pas deltaI. Cependant, à la différence de la RFC 2508, si dT=0 le décompresseur CONSERVE le deltaT actuel dans le contexte (IL NE RÈGLE PAS deltaT à 0).

Un paquet COMPRESSED_UDP amélioré est similaire par son contenu et son comportement à un paquet COMPRESSED_RTP, mais il a plus de bits fanions, dont certains correspondent aux valeurs absolues pour les champs d'en-tête RTP.

COMPRESSED_UDP avec champs RTP individuels, quand F=1 :





Usage du paquet COMPRESSED_UDP amélioré :

Il est utile que le compresseur rafraîchisse périodiquement l'état du décompresseur pour éviter que le décompresseur envoie des messages CONTEXT_STATE en cas de perte de paquet irrécupérable. En utilisant les fanions F=0 et I=1, dI=1, dT=1, le paquet COMPRESSED_UDP rafraîchit tous les paramètres de contexte.

Quand la compression est faite sur une liaison à pertes avec un long délai d'aller retour, on veut minimiser les invalidations de contexte. Si les valeurs de delta changent fréquemment, le contexte peut être souvent invalidé. Dans ce cas, le compresseur PEUT choisir de toujours envoyer des valeurs absolues et jamais de valeurs de delta, utilisant les paquets COMPRESSED_UDP avec les fanions F=1, et ceux de S, T, I qui sont nécessaires.

2.2 Somme de contrôle d'en-tête CRTP

Le paragraphe 3.3.5 de la RFC 2508 décrit comment la somme de contrôle UDP peut être utilisée pour valider périodiquement la reconstruction d'en-tête ou quand l'algorithme "deux fois" est utilisé. Quand une somme de contrôle UDP n'est pas présente (a la valeur zéro) dans un flux, une telle validation ne sera pas possible. Pour couvrir ce cas, le CRTP amélioré donne une option par laquelle le compresseur PEUT remplacer la somme de contrôle UDP nulle par une somme de contrôle d'en-têtes de 16 bits (HDRCKSUM) qui est ensuite supprimée par le compresseur après validation. Noter que cette option n'est jamais utilisée avec IPv6 car une somme de contrôle UDP nulle n'est pas permise.

Un nouveau fanion C dans le paquet FULL_HEADER, comme spécifié ci-dessous, indique quand il est établi que tous les paquets COMPRESSED_UDP et COMPRESSED_RTP envoyés dans ce contexte auront une HDRCKSUM insérée. Le compresseur PEUT établir le fanion C quand le paquet UDP porté dans le paquet FULL_HEADER contenait à l'origine une valeur de somme de contrôle de zéro. Si le fanion C est établi, le paquet FULL_HEADER lui-même DOIT aussi avoir la HDRCKSUM insérée. Si un paquet dans le même flux arrive ensuite au compresseur avec une somme de contrôle UDP présente, un nouveau paquet FULL_HEADER DOIT alors être envoyé avec le fanion à zéro pour rétablir le contexte.

La HDRCKSUM est calculée de la même façon qu'une somme de contrôle UDP sauf qu'elle ne couvre pas toutes les données UDP. C'est-à-dire que la HDRCKSUM est sur 16 bits le complément à un de la somme des compléments à un du pseudo en-tête IP (comme défini pour UDP) de l'en-tête UDP, les 12 premiers octets des données UDP qui sont supposés contenir la partie fixe d'un en-tête RTP, et de la liste de CSRC. La partie étendue de l'en-tête RTP au delà de la liste de CSRC et les données RTP ne seront pas incluses dans la HDRCKSUM. La HDRCKSUM est placée dans le paquet COMPRESSED_UDP, ou COMPRESSED_RTP, où une somme de contrôle UDP aurait été. Le décompresseur DOIT mettre à zéro le champ Somme de contrôle UDP dans les paquets reconstruits.

Pour un contexte non RTP, il peut y avoir moins de 12 octets de données UDP présents. Les en-têtes IP et UDP peuvent encore être compressés dans un paquet COMPRESSED_UDP. Dans ce cas, la HDRCKSUM est calculée sur le pseudo en-tête IP, l'en-tête UDP, et les octets de données UDP qui sont présents. Si le nombre d'octets de données est impair, un octet de bourrage à zéro est alors ajouté afin de calculer la somme de contrôle, mais n'est pas transmis.

La HDRCKSUM ne valide pas les données RTP. Si la couche liaison est configurée à livrer les paquets sans vérification d'erreurs, les erreurs des données RTP ne seront alors pas détectées. Sur de telles liaisons, le compresseur DEVRAIT ajouter la HDRCKSUM si une somme de contrôle UDP n'est pas présente, et le décompresseur DEVRAIT valider chaque paquet reconstruit pour s'assurer qu'au moins les en-têtes sont correctes. Cela assure que le paquet sera livré à la bonne destination. Si seule la HDRCKSUM est disponible, les données RTP seront livrées même si elles comportent des erreurs. C'est une caractéristique qui peut être désirable pour des applications qui peuvent tolérer des erreurs dans les données RTP. C'est vrai aussi pour la partie étendue de l'en-tête RTP au delà de la liste de CSRC.

Voici le format des champs de longueur FULL_HEADER avec le nouveau fanion C pour indiquer qu'une somme de contrôle d'en-tête sera ajoutée dans les paquets COMPRESSED_UDP et COMPRESSED_RTP :

Pour un identifiant de contexte de 8 bits :

```
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|0|1| Génération|      CID      |  premier champ de longueur
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+

+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|            0        |C|  séq  |  Second champ de longueur
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+  C=1 : HDRCKSUM sera ajoutée
```

Pour un identifiant de contexte de 16 bits :

```
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|1|1| Génération| 0   |C|  séq  |  premier champ de longueur
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+  C=1 : HDRCKSUM sera ajoutée

+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|              CID              |  Second champ de longueur
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

2.3 Réalisation d'un fonctionnement robuste

CRTP amélioré réalise un fonctionnement robuste en envoyant les changements plusieurs fois pour garder la

synchronisation entre le compresseur et le décompresseur. Cette méthode est caractérisée par un nombre "N" qui représente la qualité de la liaison entre les hôtes. Cela signifie que la probabilité de perte de plus de N paquets adjacents sur cette liaison est faible. Pour chaque changement d'une valeur pleine ou delta, si le compresseur inclut le changement dans N+1 paquets consécutif, le décompresseur peut alors conserver son état du contexte synchronisé avec le compresseur en utilisant l'algorithme "deux fois" tant que pas plus de N paquets adjacents sont perdus.

Comme la mise à jours est répétée dans N+1 paquets, si au moins un de ces N+1 paquets de mise à jour est reçu par le décompresseur, les valeurs pleines et de delta dans le contexte au décompresseur vont être mises à jour et son contexte va rester synchronisé avec le contexte du compresseur. On peut conclure que tant que moins de N+1 paquets adjacents sont perdus, le contexte au décompresseur a la garantie d'être synchronisé avec le contexte du compresseur, et l'utilisation de l'algorithme "deux fois" pour récupérer d'une perte de paquet va réussir à mettre à jour le contexte et restaurer les paquets compressés.

Le cycle de numéros de séquence de liaison se fait sur 16 paquets, de sorte que le nombre de paquets perdus n'est pas toujours clair. Par exemple, si le numéro de séquence de liaison précédent était 5 et si le numéro actuel est 4, une possibilité est que 15 paquets aient été perdus, mais une autre possibilité est que du fait d'un désordre, le paquet 5 soit arrivé avant le paquet 4 et qu'ils soient en réalité adjacents. Si il y a une interprétation des numéros de séquence de liaison qui pourrait être un trou de moins de N+1, l'algorithme "deux fois" peut être appliqué ce nombre de fois et vérifié avec la somme de contrôle UDP (ou la HDRCKSUM).

Lorsque plus de N paquets sont perdus, toutes les répétitions d'une mise à jour peuvent avoir été perdues. L'état du contexte peut alors être différent au compresseur et au décompresseur. Le décompresseur peut quand même essayer de récupérer en faisant une ou plusieurs suppositions quant au nombre de paquets perdus et appliquer ensuite l'algorithme "deux fois" ce nombre de fois.

Cependant, comme le champ Identifiant IPv4 n'est pas inclus dans la somme de contrôle, cela ne valide pas l'ID IPv4.

La conclusion est que pour IPv4 si plus de N paquets sont perdus, le décompresseur NE DEVRAIT PAS essayer de récupérer en utilisant l'algorithme "deux fois" et DEVRAIT plutôt invalider le contexte et envoyer un paquet CONTEXT_STATE. En IPv6, le décompresseur PEUT toujours essayer de récupérer de la perte de paquet en utilisant l'algorithme "deux fois" et en vérifiant le résultat avec la somme de contrôle UDP.

Il appartient à la mise en œuvre de déduire un N approprié pour une liaison. La valeur est conservée de façon indépendante pour chaque contexte et il n'est pas exigé qu'elle soit la même pour tous les contextes. Quand on compresse un nouveau flux, le compresseur règle une valeur de N pour ce contexte et envoie N+1 paquets FULL_HEADER. Le compresseur DOIT aussi répéter chaque mise à jour COMPRESSED_UDP suivante N+1 fois. La valeur de N peut être changée pour un contexte existant en envoyant une nouvelle séquence de paquets FULL_HEADER.

Le décompresseur apprend la valeur de N en comptant le nombre de fois que le paquet FULL_HEADER est répété et en mémorisant la valeur résultante dans le contexte correspondant. Si certains des paquets FULL_HEADER sont perdus, le décompresseur peut quand même être capable de déterminer la valeur correcte de N en observant le changement du numéro de séquence de 4 bits porté dans les paquets FULL_HEADER. Toute imprécision dans le compte va conduire le décompresseur à supposer une valeur de N plus petite que ce qu'envoie le compresseur. Ceci est sûr car la seule conséquence négative est que le décompresseur peut envoyer un paquet CONTEXT_STATE alors qu'il n'est pas réellement nécessaire de le faire. En réponse, le compresseur va envoyer à nouveau des paquets FULL_HEADER, fournissant une autre opportunité au décompresseur de compter le N correct.

L'envoi de paquets FULL_HEADER est aussi déclenché par un changement dans un des champs gardés constants dans le contexte, comme le TOS IP. Si un tel changement DEVAIT se produire alors que le compresseur est en train d'envoyer les paquets FULL_HEADER N+1, le compresseur DOIT alors envoyer N+1 paquets FULL_HEADER après avoir effectué le changement. Cela pourrait causer la réception par le décompresseur de plus de N+1 paquets FULL_HEADER à la suite, avec pour résultat qu'il suppose une plus grande valeur de N que correct. Cela pourrait conduire à une perte non détectée de la synchronisation du contexte. Donc, le compresseur DOIT changer le numéro de "génération" dans le contexte et dans le paquet FULL_HEADER quand il commence à envoyer la séquence de N+1 paquets FULL_HEADER afin que le décompresseur puisse détecter la nouvelle séquence. Pour IPv4, c'est un changement du comportement par rapport à la RFC 2508.

Les paquets CONTEXT_STATE DEVRAIENT aussi être répétés N+1 fois (en utilisant le même numéro de séquence pour chaque contexte) pour fournir une mesure similaire de robustesse contre la perte de paquet. Ici N peut être le plus grand N de tous les contextes inclus dans le paquet CONTEXT_STATE, ou tout nombre que le décompresseur trouve nécessaire pour assurer la robustesse.

2.3.1 Exemples

Voici quelques exemples pour montrer le fonctionnement robuste de CRTP amélioré en utilisent N+1 répétitions de mises à jour. Dans ce flux, le codec audio envoie un échantillon toutes les 10 millisecondes. La première émission vocale dure une seconde. Puis il y a deux secondes de silence, puis une autre émission vocale. On suppose aussi dans ce premier exemple que le champ ID IPv4 ne s'incrémente pas à un taux constant parce que l'hôte génère d'autres flux de trafic non corrélés en même temps et donc le delta d'identifiant IPv4 change pour chaque paquet.

Dans ces exemples, on utilise des notations abrégées :
FH (FULL_HEADER) : en-tête complet
CR (COMPRESSED_RTP) : RTP compressé
CU (COMPRESSED_UDP) : UDP compressé

En fonctionnement sur une liaison à faibles pertes, on peut juste utiliser des paquets COMPRESSED_RTP dans la méthode CRTP de base spécifiée dans la RFC 2508. On pourrait avoir la séquence de paquets suivante :

N° de séquence	Temps	Type de paquet	Mises à jour et commentaires
1	10	FH	
2	20	CR	dI dT=10
3	30	CR	dI
4	40	CR	dI
...			
100	1000	CR	dI
101	3010	CR	dI dT=2010
102	3020	CR	dI dT=10
103	3030	CR	dI
104	3040	CR	dI
...			

Dans la séquence ci-dessus, si un paquet est perdu, on ne peut pas le récupérer ("deux fois" ne marche pas à cause de l'identifiant IPv4 imprévisible) et le contexte DOIT être invalidé.

Voici le même exemple en utilisant la méthode de CRTP amélioré spécifiée dans le présent document, lorsque N=2. Noter que le compresseur envoie seulement l'identifiant IPv4 absolu (I) et non le delta d'ID IPv4 (dI).

N° de séquence	Temps	Type de paquet	Fanions CU F I dT dI M S T P	Mises à jour et commentaires
1	10	FH		
2	20	FH		répète les champs constants
3	30	FH		répète les champs constants
4	40	CU	1 1 1 0 M 0 1 0	I T=40 dT=10
5	50	CU	1 1 1 0 M 0 1 0	I T=50 dT=10 répète mise à jour T & dT
6	60	CU	1 1 1 0 M 0 1 0	I T=60 dT=10 répète mise à jour T & dT
7	70	CU	1 1 0 0 M 0 0 0	I
8	80	CU	1 1 0 0 M 0 0 0	I
...	...	...	...	
100	1000	CU	1 1 0 0 M 0 0 0	I
101	3010	CU	1 1 0 0 M 0 1 0	I T=3010 T changé, garde les deltas
102	3020	CU	1 1 0 0 M 0 1 0	I T=3020 répète mise à jour de T
103	3030	CU	1 1 0 0 M 0 1 0	I T=3030 répète mise à jour de T
104	3040	CU	1 1 0 0 M 0 0 0	I
105	3050	CU	1 1 0 0 M 0 0 0	I
..	...	...	...	

Ce second exemple est la même séquence, mais suppose que delta ID IP est constant. D'abord le CRTP de base pour une liaison sans pertes :

N° de séquence	Temps	Type de paquet	Mises à jour et commentaires
1	10	FH	
2	20	CR	dI dT=10
3	30	CR	
4	40	CR	
...	...	...	
100	1000	CR	
101	3010	CR	dT=2010
102	3020	CR	dT=10
103	3030	CR	
104	3040	CR	
...	...	...	

Pour la séquence équivalente en CRTP amélioré, le paquet COMPRESSED_RTP le plus efficace peut encore être utilisé une fois que les deltas sont tous établis :

N° de séquence	Temps	Type de paquet	Fanions CU							Mises à jour et commentaires
			F	I	dT	dI	M	S	T	P	
1	10	FH									
2	20	FH									répète les champs constants
3	30	FH									répète les champs constants
4	40	CU	1	1	1	1	M	0	1	0	I dI T=40 dT=10
5	50	CU	1	1	1	1	M	0	1	0	I dI T=50 dT=10 répète les mises à jour
6	60	CU	1	1	1	1	M	0	1	0	I dI T=60 dT=10 répète les mises à jour
7	70	CR									
8	80	CR									
...	...	...	...								
100	1000	CR									
101	3010	CU	1	0	0	0	M	0	1	0	T=3010 T changé, garde les deltas
102	3020	CU	1	0	0	0	M	0	1	0	T=3020 répète la mise à jour de T
103	3030	CU	1	0	0	0	M	0	1	0	T=3030 répète la mise à jour de T
104	3040	CR									
105	3050	CR									
...	...	...	...								

Voici le second exemple quand on utilise IPv6. D'abord, le CRTP de base pour une liaison sans pertes :

N° de séquence	Temps	Type de paquet	Mises à jour et commentaires
1	10	H	
2	20	R	T=10
3	30	R	
4	40	R	
...	...		
100	1000	R	
101	3010	R	T=2010
102	3020	R	T=10
103	3030	R	
104	3040	R	
...			

Pour la séquence équivalente en CRTP amélioré, le paquet COMPRESSED_RTP plus efficace peut encore être utilisé une fois que les deltas sont tous établis :

N° de séquence	Temps	Type de paquet	Fanions CU							Mises à jour et commentaires
			F	I	dT	dI	M	S	T	P	
1	10	FH									
2	20	FH									répète les champs constants
3	30	FH									répète les champs constants
4	40	CU	1	0	1	0	M	0	1	0	T=40 dT=10
5	50	CU	1	0	1	0	M	0	1	0	T=50 dT=10 répète les mises à jour
6	60	CU	1	0	1	0	M	0	1	0	T=60 dT=10 répète les mises à jour
7	70	CR									
8	80	CR									

...	...	...		
100	1000	CR		
101	3010	CU	1 0 0 0 M 0 1 0	T=3010 T changé, garde les deltas
102	3020	CU	1 0 0 0 M 0 1 0	T=3020 répète la mise à jour de T
103	3030	CU	1 0 0 0 M 0 1 0	T=3030 répète la mise à jour de T
104	3040	CR		
105	3050	CR		
...	...	...		

3. Négociation de l'usage de CRTP amélioré

L'utilisation de la compression IP/UDP/RTP (CRTP) sur une liaison particulière est une fonction du protocole de couche liaison. On suppose que la négociation de l'utilisation de CRTP sera définie séparément pour chaque couche de liaison.

Pour les couches de liaison qui ont déjà défini une négociation pour l'usage de CRTP comme spécifié dans la RFC 2508, une extension à cette négociation sera requise pour indiquer l'usage de CRTP amélioré défini dans le présent document car la syntaxe des formats de paquet existante a été étendue.

4. Considérations sur la sécurité

Comme le chiffrement élimine les redondances que le présent schéma de compression essaye d'exploiter, on est enclin à écarter le chiffrement pour effectuer un fonctionnement sur une liaison à faible bande passante. Cependant, dans les cas où le chiffrement des données et pas des en-têtes est satisfaisant, RTP spécifie une autre méthode de chiffrement dans laquelle seule la charge utile RTP est chiffrée et les en-têtes sont laissés en clair [RFC3711]. Cela permet quand même d'appliquer la compression.

Un dysfonctionnement du compresseur ou sa malveillance pourrait amener le décompresseur à reconstituer des paquets qui ne correspondent pas aux paquets d'origine mais ont quand même des en-têtes IP, UDP et RTP valides et éventuellement même des sommes de contrôle UDP valides. Une telle corruption peut être détectée avec des mécanismes d'authentification et d'intégrité de bout en bout qui ne seront pas affectés par la compression. Des portions constantes des en-têtes d'authentification seront compressées comme décrit dans la [RFC2507].

Aucune authentification n'est effectuée sur le paquet de contrôle CONTEXT_STATE envoyé par ce protocole. Un attaquant qui aurait accès à la liaison entre le décompresseur et le compresseur pourrait injecter des paquets CONTEXT_STATE falsifiés et causer la réduction de l'efficacité de la compression, résultant probablement en de l'encombrement sur la liaison. Cependant, un attaquant qui a accès à la liaison pourrait aussi perturber le trafic de nombreuses autres façons.

Une menace potentielle de déni de service existe lorsque on utilise des techniques de compression qui ont une charge de calcul non uniforme à l'extrémité de réception. L'attaquant peut injecter dans le flux des datagrammes pathologiques qui sont complexes à décompresser et causent la surcharge du receveur et dégradent le traitement des autres flux. Cependant, cette compression ne présente aucune non uniformité significative.

5. Remerciements

Les auteurs tiennent à remercier Van Jacobson, coauteur de la RFC 2508, et les auteurs de la RFC 2507, Mikael Degermark, Bjorn Nordgren, et Stephen Pink. Les auteurs tiennent aussi à remercier Dana Blair, Francois Le Faucheur, Tim Gleeson, Matt Madison, Hussein Salama, Mallik Tatipamula, Mike Thomas, Alex Tweedly, Herb Wildfeuer, Andrew Johnson, et Dan Wing.

6. Références

6.1 Références normatives

[RFC2119] S. Bradner, "Mots clés à utiliser dans les RFC pour indiquer les niveaux d'exigence", BCP 14, mars 1997. *(MàJ par* RFC8174)

[RFC2507] M. Degermark, B. Nordgren, S. Pink, "Compression d'en-tête IP", février 1999. *(P.S.)*

[RFC2508] S. Casner, V. Jacobson, "Compression d'en-têtes IP/UDP/RTP pour liaisons séries à bas débit", février 1999. *(P.S.)*

[RFC3544] T. Koren, S. Casner, C. Bormann, "Compression d'en-tête IP sur PPP", juillet 2003. (*Remplace* RFC2509) *(P.S.)*

[RFC3550] H. Schulzrinne, S. Casner, R. Frederick et V. Jacobson, "RTP : un protocole de transport pour les applications en temps réel", STD 64, juillet 2003. *(MàJ par* RFC7164, RFC7160, RFC8083, RFC8108)

6.2 Références pour information

[RFC3095] C. Bormann et autres, "Compression d'en-tête robuste (ROHC) : cadre et quatre profils", juillet 2001. (*MàJ par* RFC3759, RFC4815) *(P.S.)*

[RFC3711] M. Baugher et autres, "Protocole de transport sécurisé en temps réel (SRTP)", mars 2004. *(P.S.)*

7. Notice de propriété intellectuelle

8. Adresse des auteurs

Tmima Koren
Cisco Systems, Inc.
170 West Tasman Drive
San Jose, CA 95134-1706
USA
mél : tmima@cisco.com

Stephen L. Casner
Packet Design
3400 Hillview Avenue, Building 3
Palo Alto, CA 94304
USA
mél : casner@acm.org

John Geevarghese
Motorola India Electronics Ltd.
No. 33 A Ulsoor Road
Bangalore,
India
mél : geevjohn@hotmail.com

Bruce Thompson
Cisco Systems, Inc.
170 West Tasman Drive
San Jose, CA 95134-1706
USA
mél : brucet@cisco.com

Patrick Ruddy
Cisco Systems, Inc.
3rd Floor
96 Commercial Street
Leith, Edinburgh EH6 6LX
Scotland
mél : pruddy@cisco.com

9. Déclaration complète de droits de reproduction

Remerciement
Le financement de la fonction d'édition des RFC est actuellement fourni par l'Internet Society.