Groupe de travail Réseau
Request for Comments : 3782
RFC rendue obsolète : 2582
Catégorie : En cours de normalisation
Traduction Claude Brière de L'Isle

S. Floyd, ICSI
T. Henderson, Boeing
A. Gurtov, TeliaSonera
avril 2004

Modification NewReno à l'algorithme de récupération rapide de TCP

Résumé
L'objet du présent document est de faire avancer le statut des algorithmes de retransmission rapide et de récupération rapide NewReno de TCP de la RFC 2582 de Expérimental à En cours de normalisation.

Le principal changement de ce document par rapport à la RFC 2582 est de spécifier la variante Careful des algorithmes de retransmission rapide et de récupération rapide NewReno de TCP. L'algorithme de base décrit dans la RFC 2582 n'essayait pas d'éviter les multiples retransmissions rapides inutiles qui pouvaient survenir après une fin de temporisation. Cependant, la RFC 2582 définissait aussi les variantes "Careful" et "Less Careful" qui évitent ces retransmissions rapides inutiles et recommandait la variante Careful. Le présent document spécifie la variante précédemment nommée "Careful" comme version de base du NewReno TCP.

1. Introduction

Pour la mise en œuvre normale de l'algorithme TCP de récupération rapide décrit dans la [RFC2581] (d'abord mis en œuvre dans la publication Reno BSD 1990, et appelé Algorithme Reno dans [FF96]) l'envoyeur de données TCP ne retransmet un paquet qu'après que soit intervenue une fin de temporisation de retransmission, ou après que trois accusés de réception dupliqués soient arrivés, déclanchant l'algorithme de retransmission rapide. Une seule temporisation de retransmission peut résulter en la retransmission de plusieurs paquets de données, mais chaque invocation de l'algorithme de retransmission rapide dans la RFC 2581 conduit à la retransmission de seulement un paquet de données.

Des problèmes peuvent donc survenir lorsque plusieurs paquets sont abandonnés à partir d'une seule fenêtre de données et que les algorithmes de retransmission rapide et de récupération rapide sont invoqués. Dans ce cas, si l'option SACK est disponible, l'envoyeur TCP a les informations pour prendre des décisions intelligentes sur quels paquets retransmettre et quels paquets ne pas retransmettre durant la récupération rapide. Le présent document ne s'applique qu'aux connexions TCP qui sont dans l'incapacité d'utiliser l'option TCP d'accusé de réception sélectif (SACK, *Selective Acknowledgement*) soit parce que l'option n'est pas prise en charge localement, soit parce que l'homologue TCP n'a pas indiqué sa volonté d'utiliser SACK.

En l'absence de SACK, il y a peu d'informations disponibles à l'envoyeur TCP pour prendre ses décisions de retransmission durant la récupération rapide. À partir des trois accusés de réception dupliqués, l'envoyeur déduit une perte de paquet, et retransmet le paquet indiqué. Après cela, l'envoyeur des données pourrait recevoir des accusés de réception dupliqués supplémentaires, car le receveur des données accuse réception des paquets de données supplémentaires qui étaient déjà envoyés dans le réseau lorsque l'envoyeur est passé en récupération rapide.

Dans le cas où plusieurs paquets ont été abandonnés à partir d'une seule fenêtre de données, la première nouvelle information disponible à l'envoyeur arrive lorsque l'envoyeur reçoit un accusé de réception pour le paquet retransmis (c'est-à-dire, le paquet retransmis lorsque la retransmission rapide a été activée). Si il y a un seul abandon de paquet et pas de réarrangement, alors l'accusé de réception pour ce paquet va acquitter tous les paquets transmis avant le passage à la retransmission rapide. Cependant, si il y a plusieurs abandons de paquets, alors l'accusé de réception pour le paquet retransmis va acquitter certains, mais pas tous, des paquets transmis avant la retransmission rapide. On appelle cet accusé de réception un accusé de réception partiel.

Parmi plusieurs autres suggestions, [Hoe95] suggérait que durant la récupération rapide, l'envoyeur des données TCP réagisse à un accusé de réception partiel en déduisant que le prochain paquet en séquence a été perdu, et en retransmettant ce paquet. Le présent document décrit une modification de l'algorithme de récupération rapide de la RFC 2581 qui incorpore une réponse aux accusés de réception partiels reçus dans la récupération rapide. On appelle NewReno cet algorithme de récupération rapide modifié, parce qu'il est une variation légère mais significative de l'algorithme Reno de base de la RFC 2581. Le présent document ne discute pas les autres suggestions de [Hoe95] et [Hoe96], telles qu'un changement du paramètre ssthresh durant le démarrage lent *(Slow-Start)*, ou la proposition d'envoyer un nouveau paquet tous les deux accusés de réception dupliqués durant la récupération rapide. La version de NewReno dans le présent document s'appuie aussi sur d'autres discussions de NewReno dans la littérature [LM97], [Hen98].

On ne prétend pas que la version NewReno de la récupération rapide décrite ici soit une modification optimale de la récupération rapide pour répondre aux accusés de réception partiels, pour les connexions TCP qui ne sont pas capables d'utiliser SACK. Sur la base de nos expériences des modifications de NewReno dans le simulateur [NS] et avec de nombreuses mises en œuvre de NewReno, nous pensons que cette modification améliore les performances des algorithmes de retransmission rapide et de récupération rapide dans une large variété de scénarios.

2. Terminologie et définitions

Les mots clés "DOIT", "NE DOIT PAS", "EXIGE", "DEVRA", "NE DEVRA PAS", "DEVRAIT", "NE DEVRAIT PAS", "RECOMMANDÉ", "PEUT", et "FACULTATIF" dans le présent document sont à interpréter comme décrit dans la [RFC 21191]. La présente RFC indique les niveaux d'exigence pour les mises en œuvre conformes de TCP qui prennent en charge les algorithmes NewReno Retransmission rapide et Récupération rapide décrits dans le présent document.

Le présent document suppose le lecteur familier avec les termes "taille maximum de segment d'envoyeur" (SMSS, *Sender Maximum Segment Size*), fenêtre d'encombrement (cwnd, *Congestion Window*), et taille en cours (*Flight Size*, FlightSize) définis dans la [RFC2581]. La [RFC2581] donne la définition suivante : Taille en cours : Quantité de données envoyées non encore acquittées.

3. Algorithmes de retransmission rapide et de récupération rapide dans NewReno

La mise en œuvre standard des algorithmes de retransmission rapide et de récupération rapide est donnée dans la [RFC2581]. Cette section spécifie l'algorithme NewReno de base. Les sections 4 à 6 décrivent des variantes facultatives, et les motivations qui les sous-tendent, et qu'une mise en œuvre peut vouloir considérer lorsque elle fait le réglage des performances pour certains scénarios de réseau. Les sections 7 et 8 donnent des lignes directrices aux mises en œuvre sur la base des expériences avec NewReno.

La modification NewReno concerne la procédure de récupération rapide qui commence lorsque trois accusés de réception dupliqués sont reçus et se termine soit lorsque une fin de temporisation de retransmission survient, soit qu'un ACK arrive qui accuse réception de toutes les données jusque et incluses les données qui étaient en cours lorsque a commencé la procédure de récupération rapide.

L'algorithme NewReno spécifié dans le présent document diffère de la mise en œuvre dans la [RFC2581] dans l'introduction de la variable "recover" à l'étape 1, dans la réponse à un accusé de réception partiel ou nouveau à l'étape 5, et en modifications à l'étape 1 et l'ajout de l'étape 6 pour éviter les multiples retransmissions rapides causées par la retransmission de paquets déjà reçus par le receveur.

L'algorithme spécifié dans ce document utilise une variable "recover", dont la valeur initiale est le numéro de séquence initial envoyé.

1) Trois ACK dupliqués :
Lorsque le troisième accusé de réception dupliqué est reçu et que l'envoyeur n'est pas déjà dans la procédure de récupération rapide, vérifier pour voir si le champ Accusés de réception cumulés couvre plus que "recover". Si oui, passer à l'étape 1A. Autrement, passer à l'étape 1B.

1A) Invocation de la retransmission rapide :
Si oui, régler ssthresh à la valeur donnée dans l'équation (1) ci-dessous. (C'est l'équation 3 de la [RFC2581]).

ssthresh = max (Taille en cours / 2, 2*SMSS) (1)

De plus, enregistrer le plus fort numéro de séquence transmis dans la variable "recover", et passer à l'étape 2.

1B) Non invocation de la retransmission rapide :
Ne pas entrer dans la procédure de retransmission rapide et de récupération rapide. En particulier, ne pas changer ssthresh, ne pas passer à l'étape 2 pour retransmettre le segment "perdu", et ne pas exécuter l'étape 3 à réception d'accusés de réception dupliqués suivants.

2) Entrer en retransmission rapide:
Retransmettre le segment perdu et régler cwnd à ssthresh plus 3*SMSS. Cela "gonfle" artificiellement la fenêtre d'encombrement du nombre de segments (trois) qui ont quitté le réseau et que le receveur a mis en mémoire tampon.

3) Récupération rapide:
Pour chaque accusé de réception dupliqué supplémentaire reçu pendant la récupération rapide, incrémenter cwnd de SMSS. Cela augmente artificiellement la fenêtre d'encombrement afin de refléter le segment supplémentaire qui a quitté le réseau.

4) Récupération rapide, suite :
Transmettre un segment, si c'est permis par la nouvelle valeur de cwnd et par la fenêtre annoncée du receveur.

5) Lorsque un ACK arrive qui accuse réception de nouvelles données, cet ACK pourrait être l'accusé de réception obtenu par la retransmission à partir de l'étape 2, ou obtenu par une retransmission ultérieure.

Accusés de réception pleins :
Si cet ACK accuse réception de toutes les données jusque et incluant "recover", alors le ACK accuse réception de tous les segments intermédiaires envoyés entre la transmission originale du segment perdu et la réception du troisième ACK dupliqué. Régler cwnd soit (1) au minimum de (ssthresh, Taille en cours + SMSS) soit (2) à ssthresh, où ssthresh est la valeur établie à l'étape 1 ; ceci est appelé "diminuer" la fenêtre. (On note que "Taille en cours" dans l'étape 1 se référait à la quantité de données en cours dans l'étape 1, quand on est entré dans la récupération rapide, alors que "Taille en cours" dans l'étape 5 se réfère à la quantité de données en cours dans l'étape 5, quand on sort de la récupération rapide.) Si la seconde option est choisie, la mise en œuvre est invitée à prendre des mesures pour éviter une possible salve de données, au cas où la quantité de données en cours dans le réseau serait très inférieure à ce que permet la nouvelle fenêtre d'encombrement. Un mécanisme simple est de limiter le nombre de paquets de données qui peuvent être envoyés en réponse à un seul accusé de réception ; c'est ce qui est appelé "maxburst_" dans le simulateur NS. Sortir de la procédure de récupération rapide.

Accusés de réception partiels :
Si cet ACK N'accuse PAS réception de toutes les données jusque et incluant "recover", c'est alors un ACK partiel. Dans ce cas, retransmettre le premier segment non acquitté. Diminuer la fenêtre d'encombrement de la quantité de nouvelles données acquittées du champ d'accusé de réception cumulatif. Si l'ACK partiel accuse réception d'au moins un SMSS de nouvelles données, rajouter alors SMSS octets à la fenêtre d'encombrement. Comme à l'étape 3, cela gonfle artificiellement la fenêtre d'encombrement afin de refléter le segment supplémentaire qui a quitté le réseau. Envoyer un nouveau segment si cela est permis par la nouvelle valeur de cwnd. Cette "diminution partielle de fenêtre" tente de s'assurer que, lorsque la récupération rapide finit par se terminer, une quantité approximative de ssthresh données seront en cours dans le réseau. Ne pas sortir de la procédure de récupération rapide (c'est-à-dire que si des ACK dupliqués arrivent ensuite, exécuter les étapes 3 et 4 ci-dessus).

Pour le premier ACK partiel qui arrive durant la récupération rapide, remettre aussi à zéro le temporisateur de retransmission. La gestion des temporisateurs est discutée plus en détail à la Section 4.

6) Fins de temporisation de retransmission :
Après une fin de temporisation de retransmission, enregistrer le plus fort numéro de séquence transmis dans la variable "recover" et quitter la procédure de récupération rapide si applicable.

L'étape 1 spécifie une vérification que le champ Accusé de réception cumulatif couvre plus que "recover". Comme le champ Accusé de réception contient le numéro de séquence que l'envoyeur s'attend à recevoir ensuite, l'accusé de réception "ack_number" couvre plus que "recover" quand : ack_number - 1 > recover ; c'est-à-dire, au moins un octet de données de plus est acquitté au delà de l'octet le plus élevé qui était en cours lors de la dernière entrée en retransmission rapide.

Noter que dans l'étape 5, la fenêtre d'encombrement est diminuée après la réception d'un accusé de réception partiel. La fenêtre d'encombrement avait probablement été gonflée considérablement lorsque l'accusé de réception partiel a été reçu. De plus, selon le schéma original de pertes de paquets, l'accusé de réception partiel pourrait acquitter presque une fenêtre de données. Dans ce cas, si la fenêtre d'encombrement n'a pas été diminuée, l'envoyeur des données pourrait être capable d'envoyer presque une fenêtre de données dos à dos.

Le présent document ne spécifie pas la réponse de l'envoyeur aux ACK dupliqués lorsque l'algorithme de retransmission rapide/récupération rapide n'est pas invoqué. Ceci est traité dans d'autres documents, tels que ceux qui décrivent la procédure de transmission limitée [RFC3042]. Le présent document ne traite pas non plus des questions d'ajustement du seuil d'accusé de réception dupliqué, mais suppose le seuil spécifié dans les normes de l'IETF ; la norme actuelle est la RFC2581, qui spécifie un seuil de trois accusés de réception dupliqués.

En note finale, on observera qu'en l'absence de l'option SACK, l'envoyeur des données travaille à partir d'informations limitées. Lorsque la question de la récupération de l'abandon de plusieurs paquets à partir d'une seule fenêtre de données est d'une importance particulière, la meilleure solution de remplacement serait d'utiliser l'option SACK.

4. Rétablissement du temporisateur de retransmission en réponse à des ACK partiels

Une variante possible de la réponse aux accusés de réception partiels spécifiés à la Section 3 concerne quand remettre à zéro le temporisateur de retransmission après un accusé de réception partiel. L'algorithme de la Section 3, étape 5, ne remet le temporisateur de retransmission qu'après le premier ACK partiel. Dans ce cas, si un grand nombre de paquets étaient abandonnés dans une fenêtre de données, le temporisateur de retransmission de l'envoyeur des données TCP va finalement arriver à expiration, et l'envoyeur des données TCP invoquera le démarrage lent *(Slow-Start)*. (Ceci est illustré à la page 12 de [F98].) On appelle cela la variante impatiente de NewReno. On note que la variante impatiente à la Section 3 ne suit pas l'algorithme recommandé dans l'étape 5.1 de la [RFC2988] de redémarrage du temporisateur de retransmission après chaque transmission ou retransmission de paquet.

À l'opposé, les simulations de NewReno dans [FF96] illustrent l'algorithme décrit ci-dessus avec la modification que le temporisateur de retransmission est remis à zéro après chaque accusé de réception partiel. On appelle cela la variante lente mais ferme de NewReno. Dans ce cas, pour une fenêtre avec un grand nombre d'abandons de paquets, l'envoyeur des données TCP retransmet au plus un paquet par délai d'aller-retour. (Ce comportement est illustré dans la simulation New-Reno TCP de la Figure 5 de [FF96], et à la page 11 de [F98]).

Lorsque N paquets ont été abandonnés d'une fenêtre de données pour une grande valeur de N, la variante lente mais ferme peut rester en récupération rapide pour N temps d'aller-retour, retransmettant un paquet abandonné de plus à chaque temps d'aller-retour ; pour ces scénarios, la variante impatiente donne une récupération plus rapide et de meilleures performances. Les essais "ns test-suite-newreno.tcl impatient1" et "ns test-suite-newreno.tcl slow1" dans le simulateur NS illustrent un tel scénario, où la variante impatiente a de meilleures performances que la variante lente mais ferme. La variante impatiente peut être particulièrement importante pour les connexions TCP qui ont de grandes fenêtres d'encombrement, comme illustré par les essais "ns test-suite-newreno.tcl impatient4" et "ns test-suite-newreno.tcl slow4" dans le simulateur NS.

On peut aussi construire des scénarios où la variante lente mais ferme donne de meilleures performances que la variante impatiente. À titre d'exemple, cela se produit quand seulement un petit nombre de paquets sont abandonnés, le RTO est suffisamment petit pour que le temporisateur de retransmission arrive à expiration, et les performances auraient été meilleures sans une temporisation de retransmission. Les essais "ns test-suite-newreno.tcl impatient2" et "ns test-suite-newreno.tcl slow2" dans le simulateur NS illustrent de tels scénarios.

La variante lente mais ferme peut aussi réaliser un meilleur débit utile que la variante impatiente, en évitant des retransmissions inutiles. Cela pourrait présenter un intérêt particulier pour les liaisons cellulaires, où chaque transmission coûte de l'énergie pour la batterie et de l'argent. Les essais "ns test-suite-newreno.tcl impatient3" et "ns test-suite-newreno.tcl slow3" dans le simulateur NS illustrent de tels scénarios. La variante lente mais ferme peut aussi être plus robuste aux variations de délai dans le réseau, alors qu'une pointe de délai peut forcer la variante impatiente à une fin de temporisation et à une récupération de N paquets.

Aucune des deux variantes exposées ci-dessus n'est optimale. Notre recommandation est en faveur de la variante impatiente, comme spécifié à la Section 3 de ce document, à cause des performances médiocres de la variante lente mais ferme pour les connexions TCP avec de grandes fenêtres d'encombrement.

Une possibilité d'algorithme plus optimal serait qu'il récupère de multiples abandons de paquet aussi vite que le fait le démarrage lent, tout en rétablissant les temporisateurs de retransmission après chaque accusé de réception partiel, comme décrit dans la section suivante. On note cependant, qu'il y a une limitation aux performances potentielles dans ce cas en l'absence de l'option SACK.

5. Retransmissions après un accusé de réception partiel

Une variante possible à la réponse aux accusés de réception partiels spécifiés à la Section 3 serait de retransmettre plus d'un paquet après chaque accusé de réception partiel, et de rétablir le temporisateur de retransmission après chaque retransmission. L'algorithme spécifié à la Section 3 retransmet un seul paquet après chaque accusé de réception partiel. C'est la solution la plus prudente, en ce qu'elle est celle qui a le moins de chances de résulter en une retransmission inutile de paquet. Une variante qui récupèrerait plus vite d'une fenêtre avec de nombreux abandons de paquets serait effectivement le démarrage lent, qui retransmet deux paquets après chaque accusé de réception partiel. Une telle approche prendrait moins de N temps d'aller-retour pour récupérer de N pertes [Hoe96]. Cependant, en l'absence de SACK, récupérer aussi vite que le démarrage lent introduit la probabilité de retransmissions de paquets inutiles, et cela pourrait compliquer significativement les mécanismes de récupération.

On note que la réponse aux accusés de réception partiels spécifiés à la Section 3 du présent document et dans la RFC 2582 diffère de la réponse dans [FF96], même si les deux approches ne retransmettent qu'un paquet en réponse à un accusé de réception partiel. L'étape 5 de la Section 3 spécifie que l'envoyeur TCP répond à un ACK partiel en diminuant la fenêtre d'encombrement de la quantité de nouvelles données acquittées, rajoutant les octets SMSS si l'ACK partiel accuse réception d'au moins SMSS octets de nouvelles données, et en envoyant un nouveau segment si c'est permis par la nouvelle valeur de cwnd. Donc, seul un paquet envoyé précédemment est retransmis en réponse à chaque accusé de réception partiel, mais de nouveaux paquets supplémentaires peuvent être transmis aussi, selon la quantité de nouvelles données acquittées par l'accusé de réception partiel. À l'opposé, la variante de NewReno illustrée dans [FF96] règle simplement la fenêtre d'encombrement à ssthresh quand un accusé de réception partiel a été reçu. L'approche de [FF96] est plus prudente, et ne tente pas de garder précisément trace du nombre réel de paquets en cours après la réception d'un accusé de réception partiel. Alors que ces deux approches donnent des performances acceptables, la variante spécifiée à la Section 3 récupère plus en douceur lorsque plusieurs paquets sont abandonnées d'une fenêtre de données. (Le comportement de [FF96] peut être vu dans le simulateur NS en réglant la variable "partial_window_deflation_" à 0 pour "Agent/TCP/Newreno" ; le comportement spécifié à la Section 3 est réalisé en réglant "partial_window_deflation_" à 1.)

6. Éviter des retransmissions rapides multiples

Cette section décrit les motifs de la variable d'état "recover" de l'envoyeur, et expose les heuristiques possibles pour distinguer entre un paquet retransmis qui a été abandonné, et trois accusés de réception dupliqués de la retransmission inutile de trois paquets.

En l'absence de l'option SACK ou Horodatages, un accusé de réception dupliqué ne porte aucune information pour identifier le ou les paquets de données chez le receveur TCP qui ont déclanché cet accusé de réception dupliqué. Dans ce cas, l'envoyeur des données TCP est incapable de distinguer entre un accusé de réception dupliqué qui résulte d'une perte ou d'un paquet de données retardé, et un accusé de réception dupliqué qui résulte de la retransmission inutile par l'envoyeur d'un paquet de données qui a déjà été reçu chez le receveur des données TCP. À cause de cela, avec les algorithmes de retransmission et de récupération rapide dans Reno TCP, plusieurs pertes de segment provenant d'une seule fenêtre de données peuvent parfois résulter en plusieurs retransmissions rapides inutiles (et plusieurs réductions de la fenêtre d'encombrement) [F94].

Avec les algorithmes de retransmission rapide et de récupération rapide dans Reno TCP, les problèmes de performances causés par plusieurs retransmissions rapides sont relativement mineurs comparés aux problèmes potentiels avec Tahoe TCP, qui ne met pas en œuvre la récupération rapide. Néanmoins, des retransmissions rapides inutiles peuvent survenir avec Reno TCP sauf si des mécanismes explicites sont ajoutés pour l'éviter, comme l'utilisation de la variable "recover". (Cette modification est appelée "bugfix" dans [F98], et est illustrée aux pages 7 et 9 de ce document. Les retransmissions rapides inutiles pour Reno sans "bugfix" sont illustrées à la page 6 de [F98].)

La Section 3 de la [RFC2582] définissait une variante par défaut de NewReno TCP qui n'utilisait pas la variable "recover", et ne vérifiait pas si les ACK dupliqués couvrait la variable "recover" avant d'invoquer la retransmission rapide. Avec cette variante par défaut de la RFC2582, le problème de retransmissions rapides multiples à partir d'une seule fenêtre de données peut se produire après une temporisation de retransmission (comme à la page 8 de [F98]) ou dans des scénarios avec réarrangement (comme dans l'essai de validation "./test-all-newreno newreno5_noBF" dans le répertoire "tcl/test" du simulateur NS. Cela donne des performances similaires à celles de la page 8 de [F03].) La RFC2582 définissait aussi les variantes Careful et Less Careful de l'algorithme NewReno, et elle recommandait la variante Careful.

L'algorithme spécifié à la Section 3 du présent document correspond à la variante Careful de NewReno TCP de la RFC 2582, et élimine le problème des retransmissions rapides multiples. Cet algorithme utilise la variable "recover", dont la valeur initiale est le numéro de séquence d'envoi initial. Après chaque temporisation de retransmission, le plus fort numéro de séquence transmis jusqu'alors est enregistré dans la variable "recover".

Si, après une temporisation de retransmission, l'envoyeur TCP des données retransmet trois paquets consécutifs qui ont déjà été reçus par le receveur des données, l'envoyeur des données TCP va alors recevoir trois accusés de réception dupliqués qui ne couvrent pas plus que "recover". Dans ce cas, les accusés de réception dupliqués ne sont pas une indication d'une nouvelle instance d'encombrement. Ils sont simplement une indication que l'envoyeur a inutilement retransmis au moins trois paquets.

Cependant, lorsque un paquet retransmis est lui-même abandonné, l'envoyeur peut aussi recevoir trois accusés de réception dupliqués qui ne couvrent pas plus que "recover". Dans ce cas, l'envoyeur aurait mieux fait d'initier la retransmission rapide. Pour un TCP qui met en œuvre l'algorithme spécifié à la Section 3 du présent document, l'envoyeur ne déduit pas un abandon de paquet des accusés de réception dupliqués dans ce scénario. Comme toujours, le temporisateur de retransmission est le mécanisme de repli pour déduire les pertes de paquet dans ce cas.

Il y a plusieurs heuristiques, fondées sur les horodatages ou sur la quantité d'avancement du champ Accusé de réception cumulatif, qui permettent à l'envoyeur de distinguer, dans certains cas, entre trois accusés de réception dupliqués suivant un paquet retransmis qui a été abandonné, et trois accusés de réception dupliqués provenant de la retransmission inutile de trois paquets [Gur03], [GF04]. L'envoyeur TCP PEUT utiliser une telle heuristique pour décider d'invoquer la retransmission rapide dans certains cas, même quand les trois accusés de réception ne couvrent pas plus que "recover".

Par exemple, lorsque trois accusés de réception dupliqués sont causés par la retransmission inutile de trois paquets, cela sera probablement accompagné par l'avancement du champ Accusé de réception cumulatif d'au moins quatre segments. De même, une heuristique fondée sur les horodatages utilise le fait que quand il y a un trou dans l'espace des numéros de séquence, l'horodatage reproduit en écho dans l'accusé de réception dupliqué est celui du paquet de données le plus récent qui a avancé le champ Accusé de réception cumulatif [RFC1323]. Si on utilise les horodatages, et si l'envoyeur mémorise l'horodatage du dernier segment acquitté, alors l'horodatage dont l'accusé de réception dupliqué fait écho peut être utilisé pour distinguer entre un paquet retransmis qui a été abandonné et trois accusés de réception dupliqués provenant de la retransmission inutile de trois paquets. Les heuristiques sont illustrées dans le simulateur NS par l'essai de validation "./test-all-newreno".

6.1 Heuristique de l'accusé de réception

Si l'heuristique fondée sur les ACK est utilisée, à la suite de l'avancement du champ Accusé de réception cumulatif, l'envoyeur mémorise alors la valeur du précédent accusé de réception cumulatif comme prev_highest_ack, et mémorise le dernier ACK cumulatif comme highest_ack. De plus, l'étape suivante est effectuée si l'étape 1 de la Section 3 échoue, avant de passer à l'étape 1B.

1*)Si le champ Accusé de réception cumulatif ne couvrait pas plus que "recover", vérifier si la fenêtre d'encombrement est plus grande que SMSS octets et si la différence entre highest_ack et prev_highest_ack est au plus de 4*SMSS octets. Si c'est vrai, les ACK dupliqués indiquent un segment perdu (passer à l'étape 1A de la Section 3). Autrement, les ACK dupliqués résultent probablement de retransmissions inutiles (passer à l'étape 1B de la Section 3).

La vérification de la fenêtre d'encombrement sert à protéger contre la retransmission rapide immédiatement après une temporisation de retransmission, similaire à la variable "exitFastRetrans_" dans NS. Des exemples d'application de l'heuristique de ACK sont dans les essais de validation "./test-all-newreno newreno_rto_loss_ack" et "./test-all-newreno newreno_rto_dup_ack" dans le répertoire "tcl/test" du simulateur NS.

Si plusieurs ACK sont perdus, l'envoyeur peut voir un bond dans le ACK cumulé de plus de trois segments, et l'heuristique peut échouer. Un essai de validation pour ce scénario est "./test-all-newreno newreno_rto_loss_ackf". La RFC 2581 recommande qu'un receveur devrait envoyer des ACK dupliqués pour chaque paquet de données déclassé, comme un paquet de données reçu durant une récupération rapide. L'heuristique de ACK va très probablement échouer si le receveur ne suit pas cet avis, parce qu'alors un plus petit nombre de pertes de ACK est nécessaire pour produire un bond suffisant dans l'ACK cumulatif.

6.2 Heuristique de l'horodatage

Si cette heuristique est utilisée, l'envoyeur mémorise l'horodatage du dernier segment acquitté. De plus, le second paragraphe de l'étape 1 de la Section 3 est remplacé comme suit :

1**) Si le champ Accusé de réception cumulatif ne couvrait pas plus que "recover", vérifier si l'écho d'horodatage dans le dernier accusé de réception non dupliqué est égal à l'horodatage mémorisé. Si c'est vrai, les ACK dupliqués

indiquent un segment perdu (passer à l'étape 1A de la Section 3). Autrement, les ACK dupliqués résultent probablement de retransmissions inutiles (passer à l'étape 1B de la Section 3).

On a des exemples d'application de l'heuristique d'horodatage dans les essais de validation "./test-all-newreno newreno_rto_loss_tsh" et "./test-all-newreno newreno_rto_dup_tsh". L'heuristique d'horodatage fonctionne correctement, à la fois quand le receveur fait écho aux horodatages comme spécifié par la [RFC1323], et par ses tentatives de révision. Cependant, si le receveur fait de façon arbitraire écho des horodatages, l'heuristique peut échouer. L'heuristique peut aussi échouer si une temporisation était parasite et retourne des ACK qui ne sont pas de segments retransmis. On peut empêcher cela par des algorithmes de détection tels que ceux de la [RFC3522].

7. Questions de mise en œuvre pour le receveur des données

La [RFC2581] spécifie que les "segments de données déclassés DEVRAIENT être acquittés immédiatement, afin d'accélérer la récupération de perte". Neal Cardwell a noté que certains receveurs de données n'envoient pas un accusé de réception immédiat lorsque ils envoient un accusé de réception partiel, mais attendent plutôt l'expiration de leur temporisateur d'accusé de réception retardé [C98]. Comme le note [C98], cela limite sévèrement l'avantage potentiel de NewReno en retardant la réception de l'accusé de réception partiel chez l'envoyeur des données. En faisant écho à la RFC 2581, notre recommandation est que le receveur des données envoie un accusé de réception immédiat pour un segment déclassé, même quand le segment déclassé bouche un trou dans la mémoire tampon.

8. Questions de mise en œuvre pour l'envoyeur des données

Dans l'étape 5 de la Section 3 ci-dessus, on note que les mises en œuvre devraient prendre des mesures pour éviter une possible salve de données en quittant la récupération rapide, au cas où la quantité de nouvelles données que l'envoyeur est en droit d'envoyer du fait de la nouvelle valeur de la fenêtre d'encombrement est grande. Cela peut arriver durant NewReno lorsque des ACK sont perdus ou traités comme de pures mises à jour de fenêtre, causant par là une surestimation, de la part de l'envoyeur, du nombre de nouveaux segments qui peuvent être envoyés durant la procédure de récupération. Précisément, les salves peuvent survenir lorsque Taille en cours est très inférieur à la nouvelle fenêtre d'encombrement lors de la sortie de récupération rapide. Un mécanisme simple pour éviter une salve de données en quittant la récupération rapide est de limiter le nombre de paquets de données qui peuvent être envoyés en réponse à un seul accusé de réception. (Ceci est appelé "maxburst_" dans le simulateur NS.) D'autres mécanismes possibles pour éviter les salves incluent le ralentissement fondé sur le taux, ou de régler le seuil de démarrage lent à la fenêtre d'encombrement résultante et ensuite de remettre la fenêtre d'encombrement à Taille en cours. Une recommandation sur le mécanisme général pour éviter les schémas excessivement saccadés sort du domaine d'application du présent document.

Une mise en œuvre peut vouloir utiliser un fanion distinct pour noter si elle est ou non présentement dans la procédure de récupération rapide. L'utilisation à cette fin de la valeur du compteur d'accusés de réception dupliqués n'est pas fiable, parce qu'elle peut être remise à zéro lors d'une mise à jour de fenêtre et à cause des accusés de réception décalés.

Lors de la mise à jour du champ Accusé de réception cumulatif en dehors d'une récupération rapide, la variable d'état "recover" peut devoir aussi être mise à jour afin de continuer de permettre une possible entrée en récupération rapide (étape 1 de la Section 3). Cette question se pose lorsque une mise à jour du champ Accusé de réception cumulatif résulte en un retour à zéro des numéros de séquence qui affecte l'ordre entre le champ Accusé de réception cumulatif et la variable d'état "recover". L'entrée en récupération rapide n'est possible que lorsque le champ Accusé de réception cumulatif couvre plus que la variable d'état "recover".

Il est important que l'envoyeur réponde correctement aux ACK dupliqués reçus lorsque l'envoyeur n'est plus en récupération rapide (par exemple, à cause d'une fin de temporisation de retransmission). La procédure de la transmission limitée de la [RFC3042] décrit les réponses possibles aux premiers et seconds accusés de réception dupliqués. Lorsque sont reçus trois accusés de réception dupliqués, ou plus, le champ Accusé de réception cumulatif ne couvre pas plus que "recover", et une nouvelle récupération rapide n'est pas invoquée, il est important que l'envoyeur n'exécute pas les étapes (3) et (4) de récupération rapide de la Section 3. Autrement, l'envoyeur pourrait se retrouver pris dans une chaîne de temporisations parasites. On ne mentionne cela que parce que plusieurs mises en œuvre NewReno ont eu cette bogue, y compris celle de ns-2 [NS]. (Cette bogue du simulateur NS a été réparée en juillet 2003, avec la variable "exitFastRetrans_".)

9. Simulations

Les simulations avec NewReno sont illustrées par l'essai de validation "tcl/test/test-all-newreno" dans le simulateur NS. La commande "../../ns test-suite-newreno.tcl reno" montre une simulation avec Reno TCP, illustrant le manque de réponse de l'envoyeur des données à un accusé de réception partiel. À l'opposé, la commande "../../ns test-suite-newreno.tcl newreno_B" montre une simulation avec le même scénario utilisant les algorithmes NewReno décrits dans le présent mémoire.

10. Comparaisons entre Reno et NewReno TCP

Comme on l'a déclaré dans l'introduction, on pense que la modification NewReno décrite dans le présent document améliore les performances des algorithmes de retransmission rapide et de récupération rapide de Reno TCP dans une grande variété de scénarios. Cela a été exposé en profondeur dans [FF96], qui illustre les pauvres performances de Reno TCP lorsque plusieurs paquets sont abandonnées dans une fenêtre de données, et illustre aussi les bonnes performances de NewReno TCP dans ce scénario.

On connaît cependant bien un scénario où Reno TCP donne de meilleures performances que NewReno TCP, qu'on décrit ici pour être complet. Considérons un scénario sans perte de paquet, mais avec des réarrangements suffisants pour que l'envoyeur TCP reçoive trois accusés de réception dupliqués. Cela va déclancher les algorithmes de retransmission rapide et récupération rapide. Avec Reno TCP ou SACK TCP, il va en résulter la retransmission inutile d'un seul paquet, combiné à une diminution par deux de la fenêtre d'encombrement (montré aux pages 4 et 6 de [F03]). Avec NewReno TCP, cependant, ce réarrangement va aussi résulter en la retransmission inutile d'une fenêtre entière de données (montré à la page 5 de [F03]).

Bien que Reno TCP ait de meilleures performances que NewReno TCP en présence de réarrangements, les performances supérieures de NewReno en présence d'abandons multiples de paquets l'emportent généralement sur ses performances moins optimales en présence de réarrangement. (L'accusé de réception sélectif (SACK) TCP est la solution préférée, avec de bonnes performances dans les deux scénarios.) Le présent document recommande les algorithmes de retransmission rapide et récupération rapide de NewReno TCP plutôt que ceux de Reno TCP pour les connexions TCP qui ne prennent pas en charge SACK. On notera aussi que les mécanismes de retransmission rapide et de récupération rapide de NewReno sont largement déployés dans les mises en œuvre de TCP de l'Internet d'aujourd'hui, comme exposé dans [PF01]. Par exemple, les essais de mises en œuvre de TCP dans plusieurs milliers de serveurs de la Toile en 2001 ont montré que pour les connexions TCP où le navigateur n'avait pas la capacité SACK, plus de serveurs utilisaient les algorithmes de retransmission rapide et récupération rapide de NewReno que ceux de Reno ou Tahoe TCP [PF01].

11. Changements par rapport à la RFC2582

L'objet du présent document est de faire avancer les algorithmes de retransmission rapide et récupération rapide de NewReno de la RFC2582 sur la voie de la normalisation.

Le principal changement de ce document par rapport à la RFC2582 est de spécifier la variante Careful des algorithmes de retransmission rapide et récupération rapide de NewReno. L'algorithme de base décrit dans la RFC2582 ne tentait pas d'éviter plusieurs retransmissions rapides inutiles qui pouvaient survenir après une fin de temporisation (décrite plus en détails dans la section précédente). Cependant, la RFC2582 définissait aussi les variantes "Careful" et "Less Careful" qui évitent ces retransmission rapides inutiles, et recommandait la variante Careful. Le présent document spécifie la variante précédemment nommée "Careful" comme version de NewReno. Comme décrit ci-dessous, cet algorithme utilise une variable "recover", dont la valeur initiale est le numéro de séquence envoyé.

L'algorithme spécifié à la Section 3 vérifie si le champ Accusé de réception d'un accusé de réception partiel couvre *plus* que "recover", comme défini à la Section 3. Une autre variante possible serait simplement d'exiger que le champ Accusé de réception couvre *plus que ou autant* que "recover" avant d'initier une autre retransmission rapide. On appelle cela la variante Less Careful dans la RFC2582.

Il y a deux scénarios distincts selon lesquels l'envoyeur TCP pourrait recevoir trois accusés de réception dupliqués qui acquittent "recover" mais pas plus que "recover". Un scénario serait que l'envoyeur des données a transmis quatre paquets avec des numéros de séquence supérieurs à "recover", que le premier paquet a été éliminé dans le réseau, et que les trois paquets suivants ont déclanché trois accusés de réception dupliqués qui acquittent "recover". Le second scénario serait que l'envoyeur a inutilement retransmis trois paquets en dessous de "recover", et que ces trois paquets ont déclanché trois accusés de réception dupliqués acquittant "recover". En l'absence de SACK, l'envoyeur TCP est incapable de distinguer les deux scénarios.

Pour la variante Careful de la retransmission rapide, l'envoyeur des données devrait attendre une fin de temporisation de retransmission dans le premier scénario, mais n'aurait pas une retransmission rapide inutile dans le second. Pour la variante Less Careful de la retransmission rapide, l'envoyeur des données ferait une retransmission rapide comme désiré dans le premier scénario, et ferait une retransmission rapide inutile dans le second scénario. Le présent document spécifie seulement la variante Careful dans la Section 3. Les retransmissions rapides inutiles avec la variante Less Careful dans les scénarios avec réarrangement sont illustrés à la page 8 de [F03].

Le document spécifie aussi deux heuristiques que l'envoyeur TCP PEUT utiliser pour décider d'invoquer la retransmission rapide même quand les trois accusés de réception dupliqués ne couvrent pas plus que "recover". Ces heuristiques, l'une fondée sur les ACK, et l'autre fondée sur les horodatages, sont décrites respectivement aux paragraphes 6.1 et 6.2.

12. Conclusions

Le présent document spécifie les algorithmes NewReno de retransmission et de récupération rapide pour TCP. Cette modification NewReno à TCP peut même être importante pour les mises en œuvre TCP qui prennent en charge l'option SACK, parce que celle-ci ne peut être utilisée pour les connexions TCP que lorsque les deux nœuds d'extrémité TCP l'acceptent. NewReno fonctionne mieux que Reno [RFC2581] dans un certain nombre de scénarios discutés ici.

Un certain nombre d'options pour les algorithmes de base présentés à la Section 3 sont aussi décrits. Cela inclut le traitement du temporisateur de retransmission (Section 4), la réponse aux accusés de réception partiels (Section 5), et la value de la fenêtre d'encombrement en quittant la récupération rapide (étape 5 de la Section 3). On pense que les différences entre ces variantes de NewReno sont petites comparées aux différences entre Reno et NewReno. C'est-à-dire que la chose importante est de mettre en œuvre NewReno plutôt que Reno pour une connexion TCP sans SACK ; il est moins important de savoir exactement quelle variante de NewReno est mise en œuvre.

13. Considérations pour la sécurité

La RFC 2581 discute des considérations générales sur la sécurité concernant le contrôle d'encombrement TCP. Le présent document décrit un algorithme spécifique qui se conforme aux exigences du contrôle d'encombrement de la RFC 2581, et donc ces considérations s'appliquent aussi à cet algorithme. Il n'y a pas de problème supplémentaire de sécurité connu pour cet algorithme spécifique.

14. Remerciements

Tous nos remerciements à Anil Agarwal, Mark Allman, Armando Caro, Jeffrey Hsu, Vern Paxson, Kacheong Poon, Keyur Shah, et Bernie Volz pour les commentaires détaillés sur ce document ou son précurseur, la RFC 2582.

15. Références

15.1 Références normatives

[RFC2018] M. Mathis et autres, "Options d'accusé de réception sélectif sur TCP", octobre 1996. (*Remplace* RFC1072) *(P.S.)*

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

[RFC2581] M. Alman, V. Paxson et W. Stevens, "Contrôle d'encombrement avec TCP", avril 1999. *(Remplacée par la* RFC5681)

[RFC2582] S. Floyd, T. Henderson, "Modification NewReno à l'algorithme de récupération rapide de TCP", avril 1999. *(Expérimentale)* (*Obsolète, remplacée par la présente RFC)*

[RFC2988] V. Paxson, M. Allman, "Calcul du temporisateur de retransmission de TCP", novembre 2000. *(P.S. Obsolète, voir* RFC6298*)*

[RFC3042] M. Allman, H. Balakrishnan, S. Floyd, "Amélioration de la récupération de perte dans TCP avec la transmission limitée", janvier 2001. *(P.S.)*

15.2 Références pour information

[C98] Cardwell, N., "delayed ACKs for retransmitted packets: ouch!". novembre 1998, mél à la liste de diffusion tcpimpl, identifiant de message "Pine.LNX.4.02A.9811021421340.26785-100000@sake.cs.washington.edu", archivé à .

[F98] Floyd, S., "Révisions à la RFC 2001", présentation au groupe de travail TCPIMPL, août 1998. URL : ftp://ftp.ee.lbl.gov/talks/sf-tcpimpl-aug98.ps et ftp://ftp.ee.lbl.gov/talks/sf-tcpimpl-aug98.pdf .

[F03] Floyd, S., "Moving NewReno from Experimental to Proposed Standard?" présentation au groupe de travail TSVWG, mars 2003. URL : http://www.icir.org/floyd/talks/newreno-Mar03.ps et http://www.icir.org/floyd/talks/newreno-Mar03.pdf .

[FF96] Fall, K. et S. Floyd, "Simulation-based Comparisons of Tahoe, Reno et SACK TCP", Computer Communication Review, juillet 1996. URL : ftp://ftp.ee.lbl.gov/papers/sacks.ps.Z .

[F94] Floyd, S., "TCP et Successive retransmission rapides", Rapport technique, octobre 1994. URL : ftp://ftp.ee.lbl.gov/papers/fastretrans.ps .

[GF04] Gurtov, A. et S. Floyd, "Resolving Acknowledgment Ambiguity in non-SACK TCP", Next Generation Teletraffic et Wired/Wireless Advanced Networking (NEW2AN'04), février 2004. URL : http://www.cs.helsinki.fi/u/gurtov/papers/heuristics.html .

[Gur03] Gurtov, A., "[Tsvwg] resolving the problem of unnecessary fast retransmits in go-back-N", message à la liste de diffusion tsvwg, identifiant de message <3F25B467.9020609@cs.helsinki.fi>, juillet 28, 2003. URL : http://www1.ietf.org/mail-archive/working-groups/tsvwg/current/msg04334.html .

[Hen98] Henderson, T., "Re: NewReno and the 2001 Revision". septembre 1998. message à la liste de diffusion tcpimpl, Identifiant de message "Pine.BSI.3.95.980923224136.26134A-100000@raptor.CS.Berkeley.EDU", archivé à http://tcp-impl.lerc.nasa.gov/tcp-impl .

[Hoe95] Hoe, J., "Startup Dynamics of TCP's Congestion Control et Avoidance Schemes", Thèse de maîtrise, MIT, 1995.

[Hoe96] Hoe, J., "Improving the Start-up Behavior of a Congestion Control Scheme for TCP", ACM SIGCOMM, août 1996. URL http://www.acm.org/sigcomm/sigcomm96/program.html .

[LM97] Lin, D. et R. Morris, "Dynamics of Random Early Detection", SIGCOMM 97, septembre 1997. URL : http://www.acm.org/sigcomm/sigcomm97/program.html .

[NS] The Network Simulator (NS). URL http://www.isi.edu/nsnam/ns/ .

[PF01] Padhye, J. et S. Floyd, "Identifying the TCP Behavior of Web Servers", juin 2001, SIGCOMM 2001.

[RFC1323] V. Jacobson, R. Braden et D. Borman, "Extensions TCP pour de bonnes performances", mai 1992.

[RFC3517] E. Blanton et autres, "Algorithme de récupération de perte fondé sur l'accusé de réception sélectif prudent (SACK) pour TCP", avril 2003. *(P.S. Rendue obsolète par la* RFC6675*)*

[RFC3522] R. Ludwig, M. Meyer, "Algorithme Eifel de détection pour TCP", avril 2003. *(Expérimentale)*

Adresse des auteurs

Sally Floyd
International Computer Science Institute
téléphone : +1 (510) 666-2989
mél : floyd@acm.org
URL : http://www.icir.org/floyd/

Tom Henderson
The Boeing Company
mél : thomas.r.henderson@boeing.com

Andrei Gurtov
TeliaSonera
mél : andrei.gurtov@teliasonera.com

Déclaration complète de droits de reproduction

Propriété intellectuelle

Remerciement

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