Groupe de travail Réseau
Request for Comments : 3539
Catégorie : En cours de normalisation
Traduction Claude Brière de L'Isle

B. Aboba, Microsoft Corporation
J. Wood, Sun Microsystems, Inc.

juin 2003

Profil de transport d'authentification, autorisation et comptabilité (AAA)

Résumé
Le présent document discute des questions de transport qui se posent dans les protocoles pour l'authentification, l'autorisation et la comptabilité (AAA). Il formule aussi des recommandations sur l'utilisation du transport par les protocoles AAA. Cela inclut l'usage de RFC en cours de normalisation ainsi que de propositions expérimentales.

Table des Matières

1. Introduction

Le présent document discute des problèmes de transport qui se produisent avec les protocoles d'authentification, autorisation et comptabilité (AAA, *Authentification, Authorization and Accounting*). Il donne aussi des recommandations sur l'utilisation du transport par les protocoles d'AAA. Cela inclut l'usage de RFC sur la voie de la normalisation ainsi que de propositions expérimentales.

1.1 Langage 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].

1.2. Terminologie

Comptabilité : acte de collecte des informations sur l'utilisation des ressources pour les besoins de l'analyse de tendance, de vérification, de facturation, ou d'allocation des coûts.

Domaine administratif : internet, ou collection de réseaux, ordinateurs, et bases de données sous une administration commune.

Agent : un agent AAA est un intermédiaire qui communique avec les clients et serveurs AAA. Plusieurs types d'agents AAA existent, incluant les agents de relais, les agents de redirection, et les agents mandataires.

transport piloté par l'application : comportement de transport où le débit d'envoi des messages est limité par le taux de génération des données par l'application, plutôt que par la taille de la fenêtre d'encombrement. Dans le cas le plus extrême, l'intervalle entre transactions excède le délai d'aller retour entre envoyeur et receveur, impliquant que l'application opère avec une fenêtre d'encombrement effective de un. Le transport AAA est normalement piloté par l'application.

Paire attribut-valeur (AVP) : enchaînement de longueur variable d'un attribut unique (représenté par un entier) et d'une valeur contenant la valeur réelle identifiée par l'attribut.

Authentification : acte de vérification d'une identité revendiquée, sous la forme d'une étiquette pré existante provenant d'un espace de noms mutuellement connu, comme origine d'un message (authentification de message) ou de point d'extrémité d'un canal (authentification d'entité).

Autorisation : acte de déterminer si un droit particulier, comme d'accéder à une certaine ressource, peut être accordé au présentateur d'un accréditif particulier.

Facturation : acte de préparation d'une facture.

Identifiant d'accès réseau (NAI, *Network Access Identifier*) : c'est l'identifiant d'utilisateur soumis par l'hôte durant l'authentification de l'accès au réseau. En itinérance, l'objet du NAI est d'identifier l'usager tout autant que d'aider à l'acheminement de la demande d'authentification. Le NAI n'est pas nécessairement le même que l'adresse de messagerie électronique de l'usager ou d'identifiant d'utilisateur soumis dans l'authentification de couche application.

Serveur d'accès réseau (NAS, *Network Access Server*) : appareil auquel les hôtes se connectent pour obtenir l'accès au réseau.

Mandataire *(Proxy)* : en plus de transmettre les demandes et les réponses, les mandataires mettent en application les

politiques relatives à l'usage et au provisionnement des ressources. Cela est normalement accompli en retraçant l'état des appareils NAS. Bien que normalement les mandataires ne répondent pas aux demandes des clients avant de recevoir une réponse de la part du serveur, ils peuvent générer des messages de rejet dans les cas où les politiques sont violées. Par suite, les mandataires doivent comprendre la sémantique des messages qui passent à travers eux, et peuvent ne pas prendre en charge toutes les extensions.

Mandataire local : c'est un mandataire qui existe dans le même domaine administratif que l'appareil réseau (par exemple NAS) qui a produit la demande AAA. Normalement, un mandataire local est utilisé pour multiplexer les messages AAA de et vers un grand nombre d'appareils réseau, et peut mettre en œuvre la politique.

Mandataire à livraison différée : ils se distinguent des autres espèces de mandataires par l'envoi d'une réponse au NAS avant de relayer la demande au serveur. Par suite, les mandataires à livraison différée doivent mettre en œuvre la fonction de client et serveur AAA pour les messages qu'ils traitent. Les mandataires à livraison différée conservent normalement l'état sur les conversations en cours afin d'assurer la livraison des demandes et réponses mandatées. Bien que les mandataires à livraison différée soient le plus fréquemment déployés pour la comptabilité, ils peuvent aussi être utilisés pour mettre en œuvre la politique d'authentification/autorisation.

Transport piloté par le réseau : le comportement de transport est dit "piloté par le réseau" lorsque le taux d'envoi des messages est limité par la fenêtre d'encombrement, et non par le taux auquel l'application peut générer les données. Le transfert de fichiers est un exemple d'application où le transport est piloté par le réseau.

Redirection : plutôt que de transmettre les demandes et réponses entre clients et serveurs, les redirections renvoient les clients aux serveurs et leur permettent de communiquer directement. Comme les redirections ne se tiennent pas dans le chemin de transmission, elles n'altèrent aucune transmission d'AVP entre client et serveur. Les redirections ne génèrent pas de messages et sont capables de traiter tout type de message. Une redirection peut être configurée à ne rediriger que des messages de certains types, tout en agissant comme relais ou mandataire pour les autres types. Comme avec les relais, les redirections ne gardent pas l'état par rapport aux conversations ou aux ressources de NAS.

Relais : ils transmettent les demandes et réponses sur la base des AVP en rapport avec l'acheminement et des entrées de tableau d'acheminement du domaine. Comme les relais n'appliquent pas les politiques, ils n'examinent ni n'altèrent pas les AVP qui ne sont pas d'acheminement. Par suite, les relais ne génèrent jamais de message, n'ont pas besoin de comprendre la sémantique des messages ou des AVP qui ne sont pas d'acheminement, et ne sont pas capables de traiter des extensions ou des types de message. Comme les relais prennent des décisions sur la base des informations des AVP d'acheminement et des tableaux de transmission de domaine, ils ne conservent pas l'état sur l'utilisation des ressources de NAS ni sur les conversations en cours.

2. Questions sur l'utilisation du transport AAA

Les questions que soulèvent l'utilisation du transport AAA incluent : le pilotage par l'application par opposition à celui piloté par le réseau, la reprise lente sur échec, l'utilisation de l'algorithme de Nagle, les connexions multiples, la détection de dupliqués, l'invalidation des estimations des paramètres de transport, l'incapacité d'utiliser la retransmission rapide, l'évitement d'encombrement, les accusés de réception retardés, la reprise prématurée sur échec, le blocage de tête de ligne, l'équilibrage de charge de la connexion.

Ces points seront examinés successivement.

2.1 Pilotage par l'application contre pilotage par le réseau

Le comportement de transport AAA est normalement piloté par l'application plutôt que par le réseau. Cela signifie que le taux d'envoi des messages est normalement limité par la vitesse de leur génération par l'application, plutôt que par la taille de la fenêtre d'encombrement.

Par exemple, supposons un NAS à 48 accès avec une durée moyenne de session de 20 minutes. Cet appareil va, en moyenne, envoyer seulement 44 demandes d'authentification/autorisation à l'heure, et un nombre équivalent de demandes de comptabilité. Cela représente un espacement inter paquets moyen de 25 secondes, ce qui est bien supérieur au délai d'aller retour (RTT, *Round Trip Time*) dans la plupart des réseaux.

Même sur des appareils NAS beaucoup plus gros, l'espacement inter paquets est souvent supérieur au RTT. Par exemple, considérons un NAS à 2048 accès avec une durée moyenne de session de 10 minutes. Il va en moyenne envoyer

3,4 demandes d'authentification/autorisation par seconde, et un nombre équivalent de demandes de comptabilité. Cela se traduit par un espacement inter paquets moyen de 293 ms.

Cependant, même lorsque le comportement de transport est largement piloté par l'application, des périodes de comportement piloté par le réseau peuvent se produire. Par exemple, après le réamorçage d'un NAS, les enregistrements comptables mémorisés précédemment peuvent être envoyés au serveur de comptabilité en succession rapide. De même, après une récupération d'une panne d'alimentation, les usagers peuvent répondre par un grand nombre de connexions simultanées. Dans les deux cas, les messages AAA peuvent être générés plus rapidement que ce que le réseau peut permettre, et une file d'attente va se construire.

L'encombrement du réseau peut se produire lorsque le comportement de transport est piloté par le réseau ou piloté par l'application. Par exemple, alors qu'un seul NAS ne peut pas envoyer un trafic AAA substantiel, de nombreux NAS peuvent communiquer avec un seul mandataire ou serveur AAA. Par suite, les routeurs proches d'un mandataire ou serveur lourdement chargé peuvent subir de l'encombrement, même si le trafic provenant de chaque NAS individuel est léger. Un tel "encombrement convergent" peut résulter en abandons de paquets dans les routeurs proches du serveur AAA, ou même au sein du serveur AAA lui-même.

Considérons ce qui arrive lorsque 10 000 NAS à 48 accès, dont chacun a une durée de session moyenne de 20 minutes, sont configurés avec le même agent ou serveur AAA. Le malheureux mandataire ou serveur va recevoir 400 demandes d'authentification/autorisation par seconde et un nombre équivalent de demandes de comptabilité. Pour des demandes de 1000 octets, cela va générer 6,4 Mbit/s de trafic entrant chez l'agent ou serveur AAA.

Bien que cette charge de transaction soit dans les capacités des plus rapides agents et serveurs AAA, il existe des mises en œuvre qui ne peuvent pas traiter de si fortes charges. Donc de forts délais de mise en file d'attente et/ou taux d'abandon de paquets peuvent être subis chez l'agent ou le serveur, même si les routeurs sur le chemin ne sont pas encombrés. Donc, un protocole AAA bien conçu doit être capable de traiter l'encombrement qui survient au serveur AAA, aussi bien que l'encombrement subi sur le réseau.

2.2 Reprise lente sur échec

Lorsque TCP [RFC0793] est utilisé comme transport, les mises en œuvre AAA vont subir des temps de reprise sur échec très longs si elles attendent jusqu'à ce que la connexion TCP arrive en fin de temporisation avant de renvoyer sur une autre connexion. Ce n'est pas un problème pour SCTP [RFC2960], qui prend en charge la détection de défaillance de point d'extrémité et de chemin. Comme décrit à la Section 8 de la [RFC2960], lorsque le nombre de retransmissions excède le maximum ("Association.Max.Retrans"), le point d'extrémité homologue est considéré comme injoignable, l'association entre dans l'état CLOSED, et la défaillance est rapportée à l'application. Cela permet une plus rapide détection des défaillances.

2.3 Utilisation de l'algorithme de Nagle

Les messages de protocole AAA sont souvent plus petits que la taille de segment maximum (MSS, *maximum segment size*). Bien qu'il y ait des exceptions lorsque des messages d'authentification fondés sur le certificat sont produits ou lorsque on rencontre une faible MTU de chemin, normalement, les messages de protocole AAA font moins de 1000 octets. Donc, lorsque on utilise TCP [RFC0793], le compte total du paquet et de la redondance réseau associée peut être réduit en combinant plusieurs messages AAA dans un seul paquet.

Lorsque AAA fonctionne sur TCP et que le comportement de transport est piloté par le réseau, comme après un réamorçage lorsque de nombreux utilisateurs se connectent simultanément, ou que de nombreux enregistrements comptables mémorisés doivent être envoyés, l'algorithme de Nagle va résulter en une "mise en lots à la couche transport" des messages AAA. Bien que cela ne réduise pas le travail d'analyse des paquets par l'application et de réponse aux messages, cela réduit le nombre de paquets traités par les routeurs le long du chemin. L'algorithme de Nagle n'est pas utilisé avec SCTP.

Lorsque le transport AAA est piloté par l'application, le NAS va normalement recevoir une réponse du serveur de rattachement avant d'avoir une autre demande à envoyer. Cela implique, par exemple, que les demandes de comptabilité vont normalement être envoyés individuellement plutôt que d'être mises en lots par la couche transport. Par suite, dans le régime piloté par l'application, l'algorithme de Nagle [RFC0896] est inefficace.

2.4 Connexions multiples

Comme le champ RADIUS [RFC2865] Identifier est d'un seul octet, un maximum de 256 demandes peut être en cours entre deux points d'extrémité décrits par un quintuplet : (adresse IP de client, accès du client, UDP, adresse IP du serveur,

accès du serveur). Afin de contourner cette limitation, les clients RADIUS ont utilisé plus d'un accès d'envoi, allant parfois jusqu'à l'extrémité d'utiliser un accès de source UDP différent pour chaque accès de NAS.

Si ce comportement devait être étendu aux protocoles AAA qui fonctionnent sur un transport fiable, le résultat serait une multiplication de la rentrée progressive de démarrage lent effectif par le nombre de connexions. Par exemple, si un client AAA a dix connexions ouvertes avec un agent AAA, et a utilisé une fenêtre initiale par connexion [RFC3390] de deux, la fenêtre initiale effective sera de 20. Ceci est inapproprié, car cela permettrait au client AAA d'envoyer une grosse salve de paquets dans le réseau.

2.5 Détection des dupliqués

Lorsque un client AAA entretient plusieurs connexions avec plusieurs agents ou serveurs AAA, et lorsque il prend en charge la reprise sur échec/restauration automatique ou l'équilibrage de charge de connexion, il est possible que plusieurs agents ou serveurs reçoivent des copies dupliquées de la même transaction. Une transaction peut être envoyée sur une autre connexion avant l'expiration de l'intervalle "d'attente" nécessaire pour garantir que tous les paquets envoyés sur la connexion d'origine ont quitté le réseau. Donc, il est concevable que les transactions envoyées sur la connexion de remplacement vont arriver avant celles envoyées sur la connexion défaillante. Par suite, les agents et serveurs AAA DOIVENT être prêts à traiter les dupliqués, et DOIVENT supposer que des dupliqués peuvent arriver sur toute connexion.

Par exemple, en facturation, il est nécessaire d'être capable d'éliminer les enregistrements comptables dupliqués, sur la base de l'identifiant de session comptable, de l'horodatage d'événement et des informations d'identification de NAS. Lorsque les demandes d'authentification sont toujours idempotentes, les réponses dupliquées résultantes provenant de plusieurs serveurs vont probablement être identiques, de sorte qu'il en résultera peu de dommages.

Cependant, il y a des situations où la réponse à une demande d'authentification va dépendre d'un état établi antérieurement, comme lorsque des restrictions d'utilisation simultanée sont en application. Dans ce cas, les demandes d'authentification ne seront pas idempotentes. Par exemple, alors qu'une demande initiale peut appeler une réponse Accept, une demande dupliquée pourrait appeler une réponse Reject de la part d'un autre serveur, si l'utilisateur était supposé être déjà connecté, et qu'une seule session est permise à un instant donné. Dans ces situations, le client AAA peut recevoir à la fois les réponses Accept et Reject à la même demande dupliquée, et le résultat va dépendre de celle qui arrive la première.

2.6 Invalidation des estimations de paramètre de transport

Les principes de contrôle d'encombrement [Congest], [RFC2914] exigent la capacité qu'un protocole de transport réponde efficacement à l'encombrement, comme perçu via des retards croissants, des pertes de paquets, ou une notification explicite d'encombrement.

Avec les applications pilotées par le réseau, il est possible de répondre à l'encombrement sur une échelle de temps comparable au délai d'aller retour (RTT, *round-trip time*).

Cependant, avec les protocoles AAA, le temps écoulé entre les envois peut être plus long que le RTT, de sorte que les conditions du réseau ne peuvent pas être supposées persister entre les envois. Par exemple, la fenêtre d'encombrement peut croître durant une période dans laquelle l'encombrement est subi à cause de quelques paquets envoyés, limitant les opportunités de retours. De même, après la détection de l'encombrement, la fenêtre d'encombrement peut rester petite, même si les conditions du réseau qui existaient au moment de l'encombrement ne s'appliquent plus au moment où le prochain paquet est envoyé. De plus, dû au faible intervalle d'échantillonnage, les estimations de RTT et de RTO faites via les procédures décrites dans la [RFC2988] peuvent devenir invalides.

2.7 Incapacité d'utiliser la retransmission rapide

Lorsque la validation de la fenêtre d'encombrement [RFC2861] est mise en œuvre, le résultat est que les protocoles AAA fonctionnent la plupart du temps en démarrage lent avec une fenêtre d'encombrement initiale réglée à 1 ou 2, selon la mise en œuvre [RFC3390]. Cela implique que les protocoles AAA tirent peu d'avantages des caractéristiques de fenêtres du transport fiable.

Comme la fenêtre d'encombrement est si petite, il n'est généralement pas possible de recevoir assez d'ACK dupliqués (3) pour déclancher la retransmission rapide. De plus, comme le trafic AAA est bidirectionnel, les ACK qui incluent des données ne vont pas compter au titre des ACK dupliqués nécessaires pour déclancher la retransmission rapide. Par suite, les abandons de paquet vont exiger une fin de temporisation de retransmission (RTO, *retransmission timeout*).

2.8 Évitement d'encombrement

La loi de conservation des paquets [Congest] suggère qu'un client ne devrait pas envoyer un autre paquet dans le réseau tant qu'il n'est pas raisonnablement sûr qu'un paquet est sorti du réseau sur le même chemin. Dans le cas d'un client AAA, la loi suggère qu'il ne devrait pas retransmettre au même serveur ou choisir un autre serveur tant qu'il ne peut être raisonnablement sûr qu'un paquet est sorti du réseau sur le même chemin. Si le client avance la fenêtre lorsque des réponses arrivent, le client va alors "s'auto temporiser", ajustant son taux de transmission à la bande passante disponible.

Alors qu'un client AAA qui utilise un transport fiable comme TCP [RFC0793] ou SCTP [RFC2960] va s'auto temporiser lorsque il communique directement avec un serveur AAA, l'auto temporisation de bout en bout n'est pas assurée lorsque des agents AAA sont présents.

Comme décrit dans l'Appendice, les agents AAA incluent des relais, des mandataires, des redirections, des mandataires à livraison différée, et des mandataires de transport. Parmi ces agents, seuls les mandataires de transport et les redirections fournissent une connexion de transport directe entre le client et le serveur AAA, permettant que l'auto temporisation de bout en bout se produise.

Avec les relais, les mandataires ou les mandataires à livraison différée, deux connexions de transport séparées et découplées sont utilisées. Une connexion fonctionne entre le client et l'agent AAA, et une autre entre l'agent et le serveur. Comme les deux connexions de transport sont découplées, les ACK de couche transport ne s'écoulent pas de bout en bout, et l'auto temporisation ne se produit pas.

Par exemple, considérons ce qui arrive lorsque il existe un goulot d'étranglement entre un relais AAA et un serveur AAA. L'auto temporisation va se produire entre le client AAA et le relais AAA, causant l'ajustement par le client AAA de son taux d'envoi à celui auquel les ACK de transport s'écoulent en retour vers le relais AAA. Cependant, comme ce taux est supérieur à la bande passante du goulot d'étranglement, le système global ne va pas s'auto temporiser.

Comme il n'y a pas de connexion de transport directe entre le client AAA et le serveur AAA, le client AAA n'a pas la capacité d'estimer les paramètres de transport de bout en bout et d'ajuster son taux d'envoi à la bande passante du goulot d'étranglement entre le relais et le serveur. Par suite, le taux entrant au relais AAA peut être supérieur au taux auquel les paquets peuvent être envoyés au serveur AAA.

Dans ce cas, les performances de bout en bout seront déterminées par les détails de la mise en œuvre d'agent. En général, les performances de transport de bout en bout en présence de relais, mandataires ou mandataires à livraison différée vont toujours être pires en termes de délai et de perte de paquet que si le client et le serveur AAA communiquaient directement.

Par exemple, si l'agent fonctionne avec une grande mémoire tampon de réception, il est possible qu'une longue file d'attente se développe sur le côté receveur, car le client AAA est capable d'envoyer des paquets à l'agent AAA plus rapidement que l'agent ne peut les envoyer au serveur AAA. À la fin, la mémoire tampon va déborder, causant une perte globale de paquets ainsi que de forts retards.

Les méthodes pour induire un couplage à fine granularité entre les deux connexions de transport sont difficiles à mettre en œuvre. Une solution possible est que l'agent AAA fonctionne avec une mémoire tampon de réception non supérieure à celle de son envoyeur. Si cela est fait, "la pression arrière" (par la clôture de la fenêtre de réception) va causer la réduction par l'agent du taux d'envoi du client AAA lorsque la mémoire tampon de l'agent envoyeur se remplit. Cependant, sauf si plusieurs connexions existent entre le client AAA et l'agent AAA, la fermeture de la fenêtre de réception va affecter tout le trafic envoyé par le client AAA, même le trafic destiné aux serveurs AAA où il n'existe pas de goulot d'étranglement. Comme plusieurs connexions entre un client et un agent AAA résultent en une multiplication du taux de montée effective du démarrage rapide, cela n'est pas recommandé. Par suite, l'utilisation de la "pression arrière" ne peut pas permettre aux conversations AAA individuelles client serveur de s'auto temporiser, et cette technique paraît impraticable pour AAA.

2.9 Accusés de réception retardés

Comme décrit à l'Appendice B, les ACK peuvent constituer jusqu'à la moitié du trafic généré dans un échange AAA. Cela se produit parce que les conversations AAA sont normalement pilotées par l'application, et donc il n'y a souvent pas assez de trafic pour permettre le portage des ACK. Par suite, les protocoles AAA qui fonctionnent sur le transport TCP ou SCTP peuvent subir un doublement du trafic par rapport à celui des mises en œuvre qui utilisent le transport UDP.

Il n'est normalement pas possible de régler ce problème avec les API de prises. Les paramètres de ACK (comme la valeur du temporisateur de ACK retardé) sont normalement fixés par les mises en œuvre de TCP et SCTP et ne sont donc pas réglables par l'application.

2.10 Reprise sur défaillance prématurée

Les mises en œuvre de reprise sur défaillance RADIUS se fondent normalement sur le concept de serveur principal et secondaire, dans lequel tout le trafic s'écoule sur le serveur principal sauf s'il est indisponible. Cependant, l'algorithme de reprise su défaillance n'était pas spécifié dans la [RFC2865] ou la [RFC2866]. Par suite, les mises en œuvre de reprise sur défaillance RADIUS sont de qualité variable, avec certaines qui font une reprise sur défaillance prématurée, violant la loi de "conservation des paquets".

Lorsque un relais, un mandataire ou un mandataire à livraison différé est présent, le client AAA n'a pas de connexion directe avec un serveur AAA, et il est incapable d'estimer les paramètres de transport de bout en bout. Par suite, un client AAA qui attend une réponse de couche application de la part du serveur n'a pas de mécanisme fondé sur le transport pour déterminer une temporisation appropriée de reprise sur défaillance.

Par exemple, si le chemin entre l'agent et le serveur AAA comporte une liaison à fort délai, ou si le serveur AAA est très lourdement chargé, il est possible que le NAS fasse sa reprise sur défaillance sur un autre agent alors que des paquets sont encore en cours. Cela viole le principe de "conservation des paquets", car le client AAA va injecter des paquets supplémentaires dans le réseau avant d'avoir la preuve qu'un paquet envoyé antérieurement a quitté le réseau. Un tel comportement résulte en une situation pire sur une liaison déjà encombrée, résultant en un effondrement dû à l'encombrement [Congest].

2.11 Blocage de tête de ligne

Le blocage de tête de ligne se produit durant des périodes de pertes de paquets lorsque la durée entre les envois est plus courte que la valeur de la temporisation de retransmission (RTO, *re-transmission timeout*). Dans de telles situations, les paquets sont archivés dans la file d'attente de l'envoyeur jusqu'à ce que le paquet perdu puisse être retransmis avec succès. Cela peut être un problème pour SCTP lorsque on utilise une livraison ordonnée sur un seul flux, et pour TCP.

Le blocage de tête de ligne n'est normalement un problème que sur les plus grands NAS. Par exemple, un NAS à 48 accès avec un espacement inter paquets moyen de 25 secondes va peu probablement avoir un RTO supérieur à cela, sauf perte de paquets sévère. Cependant, un NAS à 2048 accès avec un espacement moyen inter paquet de 293 ms peut subir un blocage de tête de ligne car l'espacement inter paquets est de moins que la valeur minimum de RTO de 1 seconde [RFC2988].

2.12 Équilibrage de charge de connexion

Afin de diminuer les délais de mise en file d'attente et le blocage de tête de ligne d'adresse, une mise en œuvre AAA peut souhaiter équilibrer la charge entre les connexions à des destinations multiples. Bien qu'il soit possible d'employer des techniques d'équilibrage de charge dynamiques, ce niveau de sophistication ne peut pas être exigé. Dans de nombreuses situations, une fiabilité et un équilibrage de charge adéquats peuvent être réalisés via un équilibrage de charge statique, où le trafic est réparti entre les destinations sur la base de "pondérations" statiques.

3. Profil de transport AAA

Afin de régler les questions de transport AAA, il est recommandé que les protocoles AAA utilisent des techniques sur la voie de la normalisation aussi bien qu'expérimentales. Des détails sont fournis dans la présente Section.

3.1 Transpositions de transport

Les serveurs AAA DOIVENT prendre en charge TCP et SCTP. Les clients AAA DEVRAIENT prendre en charge SCTP, mais DOIVENT prendre en charge TCP si SCTP n'est pas disponible. Comme la prise en charge de SCTP s'améliore, il est possible que la prise en charge de SCTP soit exigée à l'avenir sur les clients. Les agents AAA héritent de toutes les obligations des serveurs à l'égard de la prise en charge du transport.

3.2 Utilisation de l'algorithme de Nagle

Bien que les protocoles AAA fonctionnent normalement sous le régime piloté par l'application, il y a des circonstances dans lesquelles ils sont pilotés par le réseau. Par exemple, lorsque un NAS se réamorce, ou lorsque la connexité est restaurée entre un NAS et un agent AAA, il est possible que plusieurs paquets soient disponibles à l'envoi.

Par suite, il y a des circonstances où la mise en lots à la couche transport fournie par l'algorithme de Nagle [RFC0896] est

utile, et par suite, les mises en œuvre AAA fonctionnant sur TCP DOIVENT permettre l'algorithme de Nagle, [RFC0896]. L'algorithme de Nagle n'est pas utilisé avec SCTP.

3.3 Connexions multiples

Les protocoles AAA DEVRAIENT utiliser une seule connexion persistante entre un client AAA et un agent ou serveur AAA. Ils DEVRAIENT assurer le traitement en parallèle des demandes, afin que plus d'une demande puisse être en cours à la fois. Afin de minimiser l'utilisation de connexions inactives dans les situations d'itinérance, un client ou agent AAA PEUT interrompre une connexion avec un agent ou serveur AAA si la connexion a été inutilisée (en ne tenant pas compte du chien de garde) pendant une certaine période, qui NE DOIT PAS être inférieure à BRINGDOWN_INTERVAL (5 minutes).

Alors qu'un client/agent AAA DEVRAIT seulement utiliser une connexion persistante avec un certain agent ou serveur AAA, il PEUT avoir des connexions avec plusieurs agents ou serveurs AAA. Un client/agent AAA connecté à plusieurs agents/serveurs peut les traiter comme principal/secondaire ou équilibrer la charge entre eux.

3.4 Chien de garde de couche application

Afin de permettre aux mises en œuvre AAA de détecter plus rapidement les défaillances de couche transport et application, les protocoles AAA DOIVENT prendre en charge un message de chien de garde de couche application.

Le message de chien de garde de couche application permet des reprises sur défaillance de la part d'un homologue défaillant, soit parce qu'il est injoignable, soit parce que ses fonctions d'application ont eu une défaillance. Ceci est distinct de l'objet du battement de cœur SCTP, qui est d'activer la reprise sur défaillance entre interfaces. Le battement de cœur SCTP peut activer une reprise sur défaillance sur un autre chemin pour accéder au même serveur, mais ne règle pas les situations où le système serveur ou le service d'application est en échec. Donc, les deux mécanismes PEUVENT être utilisés ensemble.

Le chien de garde est utilisé afin de permettre à un client ou agent AAA de déterminer quand renvoyer sur une autre connexion. Il fonctionne sur toutes les connexions ouvertes et est utilisé pour suspendre et éventuellement clore des connexions qui rencontrent des difficultés. Le chien de garde est aussi utilisé pour rouvrir et valider des connexions qui sont redevenues actives. Le chien de garde peut être utilisé au sein de configurations principales/secondaire ou d'équilibrage de charge. Cependant, il n'est pas destiné à être un mécanisme de battement de cœur de grappe.

Le chien de garde de couche application est conçu pour détecter les défaillances de l'homologue immédiat, et non pour être affecté par les défaillances des mandataires ou serveurs vers l'aval. Cela empêche l'instabilité des composants AAA vers l'aval de se propager vers l'amont. Bien que la réception d'une réponse AAA d'un homologue soit prise comme preuve que l'homologue est actif, l'absence d'une réponse est insuffisante pour conclure que l'homologue est mort. Comme l'absence de réponse peut résulter de problèmes avec le mandataire ou serveur aval, c'est seulement après l'échec d'une réponse au message de chien de garde qu'on peut déterminer que l'homologue est mort.

Comme l'algorithme de chien de garde prend toute réponse AAA en compte pour déterminer la vivacité des homologues, la diminution de l'intervalle du temporisateur de chien de garde n'augmente pas significativement le niveau du trafic de chien de garde sur les réseaux lourdement chargés. C'est parce que les messages de chien de garde n'ont pas besoin d'être envoyés lorsque d'autre trafic de réponse AAA sert de rappel constant de la vivacité de l'homologue. Le trafic de chien de garde n'augmente que lorsque le trafic AAA est léger, et donc qu'un "signal" de réponse AAA n'est pas présent. Néanmoins, diminuer l'intervalle de temporisation TWINIT augmente significativement la probabilité de fausse reprise sur défaillance, et donc cette décision devrait être prise avec précaution.

3.4.1 Vue d'ensemble de l'algorithme

Le comportement de chien de garde est contrôlé par un algorithme défini dans cette section. Cet algorithme est d'une utilisation appropriée dans les configurations de principal/secondaire ou d'équilibrage de charge. Les mises en œuvre DEVRAIENT appliquer cet algorithme, qui fonctionne comme suit :

(1) Le comportement de chien de garde est contrôlé par un simple temporisateur (Tw). La valeur initiale de Tw, avant tout changement, est Twinit. La valeur par défaut de Twinit est de 30 secondes. Cette valeur a été choisie parce qu'elle minimise la probabilité qu'une reprise sur défaillance soit générée suite à une faute d'acheminement, comme noté dans [Paxson]. Bien que Twinit PUISSE être réglé aussi bas que 6 secondes (la gigue non incluse) il NE DOIT PAS être réglé à moins. Noter que régler Twinit à une valeur aussi basse va probablement avoir pour résultat d'augmenter la probabilité de dupliqués, ainsi qu'une augmentation des reprises sur défaillance parasites et des tentatives de restauration

automatique. Afin de d'éviter les comportements de synchronisation qui peuvent se produire avec les temporisateurs fixes dans les systèmes répartis, à chaque fois l'intervalle de chien de garde est calculé avec une gigue en utilisant la valeur Twinit et en ajoutant une valeur aléatoire entre -2 et 2 secondes. D'autres calculs pour créer la gigue PEUVENT être utilisés. Ils DOIVENT être pseudo aléatoires, générés par un PRNG avec un germe selon la [RFC1750].

(2) Lorsque un message AAA est reçu, Tw est rétabli. Ceci n'a pas besoin d'être une réponse à une demande de chien de garde. Recevoir une réponse de chien de garde d'un homologue constitue une activité, et Tw devrait être rétabli. Si le temporisateur de chien de garde arrive à expiration et si aucune réponse de chien de garde n'est en cours, un message de chien de garde est alors envoyé. À l'envoi d'une demande de chien de garde, Tw est rétabli. Les paquets de chien de garde ne sont pas retransmis par le protocole AAA, car les protocoles AAA fonctionnent sur des transports fiables qui vont traiter en interne toutes les retransmissions. Par suite, une demande de chien de garde n'est envoyée que lorsque il n'y a pas de réponse de chien de garde en cours.

(3) Si le temporisateur de chien de garde arrive à expiration et si une réponse de chien de garde est en cours, la reprise sur défaillance est alors initiée. Afin qu'un client ou agent AAA effectue les procédures de reprise sur défaillance, il est nécessaire de tenir une file d'attente de messages en cours pour un homologue donné. Lorsque un message de réponse est reçu, la demande correspondante est retirée de la file d'attente. Le champ Identifiant bond par bond PEUT être utilisé pour confronter la réponse avec la demande de la file d'attente. Lorsque la reprise sur défaillance est initiée, tous les messages de la file d'attente sont envoyés à un autre agent, si disponible. Plusieurs demandes ou réponses identiques peuvent être reçues par suite d'une reprise sur défaillance. La combinaison d'un identifiant de bout en bout et de l'hôte d'origine DOIT être utilisée pour identifier les messages dupliqués. Noter que lorsque le trafic est fort, le chien de garde de couche application peut prendre jusqu'à 2 Tw pour déterminer qu'un homologue est mort. Pour les homologues qui reçoivent un fort volume de demandes AAA, les réponses AAA vont continuellement rétablir le temporisateur, de sorte qu'après une défaillance cela va prendre Tw pour que le manque de trafic soit remarqué, et pour que le message de chien de garde soit envoyé. Un autre Tw va s'écouler avant que la reprise sur défaillance soit initiée. Sur un réseau légèrement chargé sans beaucoup de trafic de réponses AAA, le temporisateur de chien de garde va normalement arriver à expiration sans être rétabli, de sorte qu'une réponse de chien de garde va être en suspens et que la reprise sur défaillance sera initiée après l'expiration d'un seul intervalle de temporisation.

(4) Le client NE DOIT PAS clore la connexion principale avant que son temporisateur de chien de garde ne soit arrivé à expiration au moins deux fois sans une réponse (noter que le chien de garde n'est cependant pas envoyé une seconde fois). Une fois que ceci s'est produit, le client DEVRAIT causer un rétablissement de transport ou clore la connexion. Une fois que la connexion principale a échoué, les demandes suivantes sont envoyées à l'autre serveur jusqu'à ce que le temporisateur de chien de garde soit rétabli sur la connexion principale. La suspension de la connexion principale empêche les oscillations entre les connexions, principale et secondaire, et assure que le comportement de reprise sur défaillance reste cohérent. L'application peut ne pas recevoir de réponse au message de demande de chien de garde du fait d'un problème de connexité, auquel cas un ACK de couche transport n'aura pas été reçu, ou le manque de réponse peut être dû à un problème de l'application. Sans visibilité à la couche transport, l'application est incapable de faire la différence, et doit se comporter avec prudence. Dans les situations où aucun ACK de couche transport n'est reçu sur la connexion principale après plusieurs retransmissions, le RTO sera retardé exponentiellement comme décrit dans la [RFC2988]. Dû à l'algorithme de Karn tel que mis en œuvre dans SCTP et TCP, l'estimateur de RTO ne sera pas rétabli tant qu'un autre ACK n'est pas reçu en réponse à une demande non retransmise. Donc, dans les cas où le problème se produit à la couche transport, après que le client a repris sur le serveur de remplacement, le RTO du principal va rester à une faible valeur sauf si un ACK est reçu sur la connexion principale. Dans le cas où le problème se produit à la couche transport, les demandes suivantes envoyées sur la connexion principale ne recevront pas le même service que ce qui était fourni à l'origine. Par exemple, au lieu d'une reprise sur défaillance se produisant après trois retransmissions, la reprise sur défaillance peut se faire sans même une seule retransmission si le RTO a été suffisamment retardé. Bien sûr, si l'absence de réponse de chien de garde était due à un problème de couche d'application, le RTO n'aura alors pas été retardé. Cependant, sans visibilité à la couche transport, il n'y a aucun moyen pour que l'application le sache. Suspendre l'utilisation de la connexion principale jusqu'à ce qu'une réponse à un message de chien de garde soit reçue garantit que le temporisateur de RTO aura été rétabli avant que la connexion principale soit réutilisée. Si aucune réponse n'est reçue après l'expiration du second temporisateur de chien de garde, la connexion principale est alors close et la suspension devient permanente.

(5) Pendant que la connexion est dans l'état clos, le client AAA NE DOIT PAS tenter d'envoyer d'autres messages de chien de garde sur la connexion. Cependant, après la clôture de la connexion, le client AAA continue de tenter périodiquement de rouvrir la connexion. Le client AAA DEVRAIT attendre que la couche transport rapporte la défaillance de la connexion avant de tenter à nouveau, mais PEUT choisir de limiter ce temps d'attente par l'intervalle de chien de garde, Tw. Si la connexion réussi à être ouverte, le message de chien de garde est alors envoyé. Une fois que trois messages de chien de garde ont été envoyés et ont reçu réponse, la connexion est remise en service, et les transactions sont une fois encore envoyées sur elle. La validation de connexion via la réception de plusieurs chiens de garde n'est pas obligée lorsque une connexion est initialement activée – dans ce cas, la connexion peut immédiatement être mise en service.

(6) Lorsque on utilise SCTP comme transport, il n'est pas nécessaire de désactiver les battements de cœur de la couche transport de SCTP. Cependant, si les mises en œuvre AAA ont accès aux paramètres de battement de cœur de SCTP, elles PEUVENT choisir de s'assurer que l'intervalle de battement de cœur de SCTP est plus long que l'intervalle de chien de garde AAA, Tw. Cela va assurer que des chemins de remplacement sont toujours sondés par SCTP, tandis que le chemin principal a un minimum de redondance de battement de cœur.

3.4.2 Prise en charge de la reprise sur échec principal/secondaire

Le temporisateur de chien de garde PEUT être intégré au style de reprise sur défaillance principal/secondaire afin de fournir une fiabilité améliorée et l'équilibrage de charge de base. Pour équilibrer la charge entre plusieurs serveurs AAA, chacun est désigné comme principal pour une portion des clients, et désigné comme secondaire de priorités diverses pour le reste. De cette façon, la charge peut être équilibrée parmi les serveurs AAA.

Dans les configurations principal/secondaire, le temporisateur de chien de garde fonctionne comme suit :

(1) On suppose que chaque client ou agent est initialement configuré avec un seul agent ou serveur principal, et une ou plusieurs connexions secondaires.

(2) Le mécanisme de chien de garde est utilisé pour suspendre et éventuellement clore les connexions principales qui rencontrent des difficultés. Il est aussi utilisé pour rouvrir et valider les connexions qui sont revenues à la santé.

(3) Une fois qu'une connexion secondaire a été promue au statut de principale, temporairement ou de façon permanente, le prochain serveur sur la liste des secondaires est promu pour combler le trou des secondaires.

(4) Le client ou agent tente périodiquement de rouvrir les connexions closes, de sorte qu'il est possible qu'une connexion précédemment fermée revienne en service et redevienne éligible à l'utilisation. Les mises en œuvre vont normalement établir une limite au nombre de connexions ouvertes en même temps, de sorte que lorsque une connexion précédemment close est remise en ligne, la connexion secondaire de la plus faible priorité sera fermée. Afin de prévenir la fermeture et l'ouverture périodique de connexions secondaires, il est recommandé que les connexions qui fonctionnent restent ouvertes pendant un minimum de 5 minutes.

(5) Afin de permettre le diagnostic du comportement de reprise sur défaillance, il est recommandé qu'un tableau des événements de reprise sur défaillance soit établi au sein de la MIB. Ces événements de reprise sur défaillance DEVRAIENT inclure les identifiants de transaction appropriés pour que les données de client et de serveur puissent être comparées, fournissant des indices sur la cause du problème (de couche transport ou application).

3.4.3 Équilibrage de charge de connexion

La reprise sur défaillance principal/secondaire est capable de fournir une résilience améliorée et un équilibrage de charge de base. Cependant, elle ne règle pas le blocage de tête de ligne d'adresse TCP, car une seule connexion est utilisée à la fois.

Un client ou agent AAA qui entretient des connexions avec plusieurs agents ou serveurs PEUT faire l'équilibrage de charge entre elles. Établir des connexions avec plusieurs agents ou serveurs réduit, mais ne supprime pas, les problèmes de blocage de tête de ligne rencontrés sur les connexions TCP. Ce problème n'existe pas avec les connexions SCTP qui utilisent plusieurs flux.

Dans les configurations d'équilibrage de charge entre connexions, le chien de garde d'application fonctionne comme suit :

(1) On suppose que chaque client ou agent est initialement configuré avec des connexions à plusieurs agents ou serveurs AAA, avec une connexion entre un certain client/agent et un agent/serveur.

(2) En équilibrage de charge statique, les transactions sont réparties entre les connexions sur la base du nombre total de connexions d'une "pondération" allouée à chaque connexion. Le hachage de Pearson [RFC3074] appliqué au NAI [RFC2486] peut être utilisé pour déterminer quelle connexion va traiter une certaine transaction. Le hachage sur le NAI permet une granularité très fine d'équilibrage de charge, tout en assurant que tout le trafic pour une certaine conversation sera envoyé au même agent ou serveur. En équilibrage de charge dynamique, la valeur de la "pondération" peut varier sur la base de conditions comme la charge d'un serveur AAA. De telles techniques, bien que sophistiquées, sortent du domaine d'application du présent document.

(3) Les transactions sont réparties entre les connexions sur la base du nombre total de connexions disponibles et de leur pondération. Un changement du nombre de connexions disponibles force à recalculer le hachage du tableau. Afin que cela ne cause pas la commutation des conversations en cours sur de nouvelles destinations, une période de transition est

nécessaire lors du recalcul, dans laquelle les deux tableaux, ancien et nouveau, sont nécessaires afin de permettre de tenir compte des conversations en cours. Noter que cela exige un moyen pour déterminer facilement si une demande représente une nouvelle conversation ou la continuation d'une conversation existante. Par suite, supprimer et ajouter des connexions est une opération coûteuse, et il est recommandé que le tableau de hachage ne soit recalculé qu'une fois qu'une connexion est close ou est revenue en service. Les connexions suspendues, bien qu'elles ne soient pas utilisées, ne forcent pas la reconfiguration du tableau de hachage tant qu'elles ne sont pas closes. De même, les connexions rouvertes qui n'ont pas accumulé suffisamment de réponses de chien de garde ne forcent la reconfiguration tant qu'elles ne reviennent pas en service. Quand une connexion est suspendue, les transactions qui lui auraient été allouées sont affectées au prochain serveur disponible. Bien que ceci résulte en un déséquilibre momentané, on estime que c'est un prix assez faible à payer pour éviter de bouleverser le tableau de hachage.

(4) Afin de permettre le diagnostic du comportement d'équilibrage de charge, il est recommandé qu'en plus du tableau des événements de reprise sur défaillance, un tableau de statistique soit tenu par client, indexé par un serveur AAA. De cette façon, on peut évaluer l'efficacité de l'algorithme d'équilibrage de charge.

3.5 Détection des dupliqués

Plusieurs facilités sont requises pour permettre la détection des dupliqués. Cela inclut des identifiants de session ainsi que des identifiants de messages bond par bond et de bout en bout. Les identifiants bond par bond dont les valeurs peuvent changer à chaque bond ne sont pas suffisants, car un serveur AAA peut recevoir le même message de plusieurs agents. Par exemple, un client AAA peut envoyer une demande à l'Agent1, puis faire une reprise sur défaillance et renvoyer la demande à l'Agent2 ; les deux agents transmettent la demande au serveur AAA de rattachement, avec des identifiants bond par bond différents. Un identifiant de session est insuffisant car il ne fait pas la distinction entre les différents messages pour la même session.

Un traitement approprié de l'identifiant de message de bout en bout assure que les opérations AAA sont idempotentes. Par exemple, sans identifiant de bout en bout, un serveur AAA qui garde trace des connexions simultanées peut envoyer un Accept en réponse à une demande initiale, et ensuite un Reject en réponse à une demande dupliquée (où l'usager n'a la permission que d'une seule connexion simultanée). Selon la réponse qui est arrivée en premier, l'accès sera permis ou non à l'usager.

Cependant, si le serveur avait mémorisé l'identifiant de message de bout en bout avec les informations de connexions simultanées, la demande dupliquée (qui utilise le même identifiant de message de bout en bout) pourrait être identifiée et la réponse correcte aurait pu être retournée.

3.6 Invalidation des estimations de paramètres de transport

Afin de traiter l'invalidation des estimations de paramètres de transport, les mises en œuvre de protocoles AAA PEUVENT utiliser la validation de fenêtre d'encombrement [RFC2861] et la validation de RTO en utilisant TCP. La présente spécification recommande aussi une procédure pour la validation du RTO.

Les [RFC2581] et [RFC2861] recommandent toutes deux qu'une connexion passe en démarrage lent après une période où aucun trafic n'a été envoyé dans l'intervalle de RTO. La [RFC2861] recommande seulement d'augmenter la fenêtre d'encombrement si elle était pleine lorsque le ACK est arrivé. La fenêtre d'encombrement est réduite de moitié à chaque intervalle de RTO si aucun trafic n'est reçu.

Lorsque la validation de fenêtre d'encombrement est utilisée, la fenêtre d'encombrement ne va pas se construire durant les périodes pilotées par l'application, et sera plutôt supprimée. Par suite, les applications AAA qui fonctionnent sous le régime piloté par l'application vont normalement tourner avec une fenêtre d'encombrement égale à la fenêtre initiale la plupart du temps, opérant en "démarrage lent perpétuel".

Durant les périodes dans lesquelles le comportement AAA est piloté par l'application, cela n'aura pas d'effet. Comme le temps entre les paquets sera supérieur au RTT, AAA va fonctionner avec une fenêtre d'encombrement effective égale à la fenêtre initiale. Cependant, durant les périodes pilotées par le réseau, l'effet sera d'espacer l'envoi des paquets AAA. Donc au lieu d'être capable d'envoyer une grosse salve de paquets dans le réseau, un client devra attendre plusieurs RTT car la fenêtre d'encombrement se construit pendant le démarrage lent.

Par exemple, un client opérant sur TCP avec une fenêtre initiale de 2, avec 35 demandes AAA à envoyer va prendre approximativement 6 RTT pour les envoyer, car la fenêtre d'encombrement se construit durant le démarrage lent : 2, 3, 3, 6, 9, 12. Après l'apurement de l'arriéré, la mise en œuvre va à nouveau être pilotée par l'application et la taille de la fenêtre d'encombrement va diminuer. Si le client utilisait SCTP, le nombre de RTT nécessaire pour transmettre toutes les demandes

serait généralement moindre, et dépendrait de la taille des demandes, car SCTP suit les progrès de l'ouverture de la fenêtre d'encombrement par octets, non par segments.

Noter que la [RFC2861] et la [RFC2988] ne traitent pas la question de la validation du RTO. C'est aussi un problème, en particulier lorsque le gestionnaire d'encombrement [RFC3124] est mis en œuvre. Durant les périodes de forte perte de paquets, le RTO peut être augmenté de façon répétée via un retard exponentiel, et peut atteindre une forte valeur. Du fait de l'absence de retours à temps sur le RTT et le RTO durant les périodes pilotées par l'application, la forte estimation du RTO peut persister longtemps après que les conditions qui l'ont généré ont disparu. La validation de RTO PEUT être utilisée pour traiter ce problème pour TCP, via la procédure suivante :

Après que la fenêtre d'encombrement a été diminuée conformément à la [RFC2861], rétablir le RTO estimé à 3 secondes. Après l'arrivée du paquet suivant, recalculer RTTavg, RTTdev, et RTO selon la méthode décrite dans la [RFC2581].

Pour régler ce problème pour SCTP, les mises en œuvre AAA DEVRAIENT utiliser les battements de cœur SCTP. La [RFC2960] déclare que les battements de cœur devraient être activés par défaut, avec un intervalle de 30 secondes. Si cet intervalle se révèle trop long pour résoudre le problème, les mises en œuvre AAA PEUVENT réduire l'intervalle de battement de cœur.

3.7 Incapacité d'utiliser la retransmission rapide

Lorsque la validation de fenêtre d'encombrement [RFC2861] est utilisée, les mises en œuvre AAA vont fonctionner la plupart du temps avec une fenêtre d'encombrement égale à la fenêtre initiale. Par suite, la taille de la fenêtre va souvent n'être pas assez large pour permettre d'utiliser la retransmission rapide pour TCP. De plus, comme le trafic AAA est bidirectionnel, les ACK qui portent des données ne vont pas compter pour déclancher la retransmission rapide. SCTP est moins susceptible de rencontrer ce problème, de sorte que les mesures décrites ci-dessous s'appliquent à TCP.

Pour régler ce problème, les mises en œuvre AAA DEVRAIENT prendre en charge l'accusé de réception sélectif décrit dans les [RFC2018] et [RFC2883]. Les mises en œuvre AAA DEVRAIENT aussi mettre en œuvre la transmission limitée pour TCP, comme décrite dans la [RFC3042]. Plutôt que de réduire le nombre d'ACK dupliqués exigé pour déclancher la récupération rapide, qui augmenterait le nombre de retransmissions inappropriées, la transmission limitée permet d'augmenter la taille de la fenêtre, permettant ainsi l'envoi de paquets supplémentaires qui à leur tour peuvent déclancher la retransmission rapide sans changer l'algorithme.

Cependant, si la validation de la fenêtre d'encombrement [RFC2861] est mise en œuvre, cette proposition n'aura un effet que dans des situations où le temps entre paquets est inférieur au délai de retransmission (RTO) estimé. Si le temps entre paquets est supérieur au RTO, des paquets supplémentaires ne seront normalement pas disponibles à l'envoi afin de tirer parti de l'augmentation de la taille de fenêtre. Par suite, les protocoles AAA vont normalement fonctionner avec la plus petite taille possible de fenêtre d'encombrement, ce qui résulte en une fin de temporisation de retransmission pour chaque paquet perdu.

3.8 Blocage de tête de ligne

TCP ne fournit pas par lui-même de solution au problème du blocage de tête de ligne, bien que ses effets puissent être amoindris par la mise en œuvre de la transmission limitée [RFC3042], et l'équilibrage de charge de la connexion.

3.8.1 Utilisation des flux SCTP pour empêcher le blocage de tête de ligne

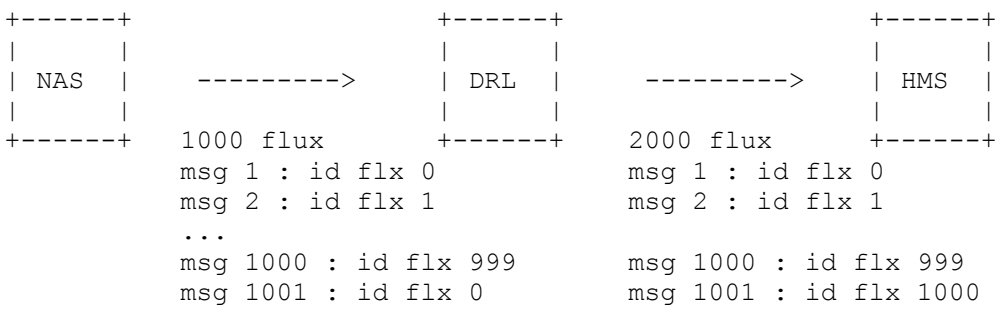
Chaque nœud AAA DEVRAIT distribuer ses messages équitablement à travers la gamme de flux SCTP sur laquelle lui et son homologue se sont mis d'accord. (Un message perdu dans un flux ne causera le blocage d'aucun autre flux.) Une mise en œuvre triviale et efficace de cela va simplement incrémenter un compteur pour l'identifiant de flux sur lequel envoyer. Lorsque le compteur atteint le nombre maximum de flux pour l'association, il revient à 0.

Les homologues AAA DOIVENT être capables d'accepter des messages sur tous flux. Noter que les flux ne sont utilisés *que* pour empêcher le blocage de tête de ligne. Toutes les informations d'identification sont portées dans la charge utile Diameter. Les messages distribués à travers plusieurs flux peuvent n'être pas reçus dans leur ordre d'envoi.

Les homologues SCTP peuvent allouer jusqu'à 65 535 flux pour une association. Le coût des flux inactifs peut être ou non de zéro, selon la mise en œuvre, et le coût des flux non inactifs est toujours supérieur à 0. Ainsi les administrateurs peuvent souhaiter limiter le nombre de flux possibles sur leurs nœuds Diameter en fonction des ressources (c'est-à-dire de la mémoire, puissance de CPU, etc.) d'un nœud particulier.

Sur un client Diameter, le nombre de flux peut être déterminé par le nombre maximum d'utilisateurs sur le NAS. Si un flux est disponible par usager, cela devrait être suffisant pour prévenir le blocage de tête de ligne. Sur un mandataire Diameter, le nombre de flux peut être déterminé par le nombre maximum de sessions en cours à partir de ce mandataire vers chaque serveur AAA aval.

Les identifiants de flux n'ont pas besoin d'être préservés par les agents de relais. Cela simplifie la mise en œuvre, car les agents peuvent facilement traiter la transmission entre deux associations qui ont des numéros de flux différents. Par exemple, considérons le cas suivant, où un serveur relais DRL transmet les messages entre un NAS et un serveur de rattachement, HMS. Le NAS et DRL se sont accordés sur 1000 flux pour leur association, et DRL et HMS se sont accordés sur 2000 flux pour leur association. La figure suivante montre les flux de messages du NAS à HMS via DRL, et les allocations d'identifiant de flux pour chaque message:



DRL peut transmettre les messages 1 à 1000 à HMS en utilisant le même identifiant de flux que le NAS a utilisé pour envoyer à DRL. Cependant, comme l'association NAS / DRL a seulement 1000 flux, le NAS revient à l'identifiant de flux 0 lorsque il envoie le message 1001. L'association DRL / HMS, de son côté, a 2000 flux, de sorte que DRL peut réallouer le message 1001 à l'identifiant de flux 1000 lorsque il le transmet à HMS.

Ce schéma de distribution fonctionne comme une tableau de hachage. Il est possible, bien que peu probable, que deux messages finissent dans le même flux, et encore moins probable qu'il y ait une perte de message résultant en un blocage lorsque ceci se produit. Si cela se révélait être un problème, les administrateurs locaux peuvent augmenter le nombre de flux sur leurs nœuds pour améliorer les performances.

3.9 Évitement d'encombrement

Afin d'améliorer les estimations de temporisateur par défaut, les mises en œuvre AAA PEUVENT utiliser le gestionnaire d'encombrement (CM, *Congestion Manager*) [RFC3124]. CM est un module de système d'extrémité qui :

(i) permet à un ensemble de plusieurs flux concurrents, à partir d'un envoyeur destinés au même receveur et partageant les mêmes propriétés d'encombrement, d'effectuer un évitement et contrôle d'encombrement approprié, et

(ii) permet aux applications de s'adapter facilement à l'encombrement du réseau.

Le CM aide à intégrer la gestion de l'encombrement à travers toutes les applications et tous les protocoles de transport. Le CM conserve les paramètres d'encombrement (bande passante disponible agrégée et par flux, temps d'aller retour par receveur, etc.) et exporte une API qui permet aux applications d'apprendre les caractéristiques du réseau, de passer les information au CM, de partager les informations d'encombrement aux autres CM, et de programmer la transmission des données.

Le CM permet aux applications AAA d'accéder aux paramètres de transport (RTTavg, RTTdev) via des rappels. Les estimations de RTO ne sont pas disponibles actuellement via l'interface de rappel, bien qu'elles devraient probablement l'être. Lorsque disponibles, les paramètres de transport DEVRAIENT être utilisés pour améliorer les valeurs de temporisateur par défaut.

3.10 Reprises sur échec prématurées

Une reprise sur défaillance prématurée est empêchée par la fonction de chien de garde décrite ci-dessus. Si le prochain bond ne retourne pas de réponse, le client AAA va lui envoyer un message de chien de garde pour vérifier sa vivacité. Si une réponse de chien de garde est reçue, le client AAA va alors savoir que le serveur du prochain bond fonctionne à la couche application. Par suite, il est seulement nécessaire de fournir des messages d'erreur terminaux, tels que les suivants :

"Occupé" : l'agent/serveur est trop occupé pour traiter des demandes supplémentaires, le NAS devrait effectuer la reprise sur défaillance de toutes les demandes sur un autre agent/serveur.

"Ne peut localiser" : l'agent ne peut localiser le serveur AAA pour le domaine indiqué ; le NAS devrait effectuer la reprise sur défaillance de cette demande sur un autre mandataire.

"Ne peut transmettre" : l'agent a essayé les deux serveurs AAA, principal et secondaire sans réponse ; le NAS devrait effectuer la reprise sur défaillance de la demande à un autre agent.

Noter que ces messages diffèrent par leur portée. Le message "Occupé" dit au NAS que l'agent/serveur est trop occupé pour TOUTE demande. Les messages "Ne peut localiser" et "Ne peut transmettre" indiquent que la destination ultime ne peut être atteinte ou ne répond pas, ce qui implique une reprise sur défaillance par demande.

4. Considérations sur la sécurité

Comme clients, agents et serveurs AAA servent de portiers d'accès réseau, ils sont des cibles tentantes pour les attaquants. Les considérations générales de sécurité concernant le contrôle d'encombrement TCP sont discutées dans la [RFC2581]. Cependant, il y a quelques considérations supplémentaires qui s'appliquent à la présente spécification.

En permettant la reprise sur défaillance entre agents AAA, la présente spécification améliore la résilience des applications AAA. Cependant, cela peut aussi ouvrir des opportunités d'attaques de déni de service.

L'algorithme de reprise sur défaillance est piloté par l'absence de réponse aux demandes AAA et aux paquets de chien de garde. Sur un réseau chargé légèrement où les réponses AAA ne seront pas reçues avant l'arrivée à expiration du temporisateur de chien de garde, un attaquant peut submerger le réseau, causant l'abandon des paquets de chien de garde. Cela va amener le client AAA à passer sur un autre agent AAA, où l'attaque peut être répétée. En faisant que le client AAA circule entre les agents AAA, le service peut être refusé aux usagers qui désirent l'accès au réseau.

Lorsque TLS [RFC2246] est utilisé pour assurer la sécurité de AAA, il y a une vulnérabilité aux paquets de rétablissement factices, ainsi qu'à d'autres attaques de déni de service de couche transport (par exemple, l'inondation de SYN). Comme SCTP offre une résilience améliorée au déni de service par rapport à TCP, lorsque les applications AAA fonctionnent sur SCTP, cela peut être atténué dans une certaine mesure.

Lorsque IPsec [RFC2401] est utilisé pour assurer la sécurité, il est important que la politique IPsec exige IPsec sur les paquets entrants. Afin de permettre à un client AAA de déterminer quels mécanismes de sécurité sont utilisés sur un agent ou serveur sans en avoir connaissance à priori, il peut être tentant d'initier une connexion en clair, et d'avoir alors l'agent AAA qui répond avec IKE [RFC2409]. Bien que cette approche minimise la configuration client requise, elle augmente la vulnérabilité à l'attaque de déni de service, car une demande de connexion peut alors non seulement lier les ressources de transport, mais aussi des ressources au sein de la mise en œuvre de IKE.

5. Considérations relatives à l'IANA

Le présent document ne crée aucun nouvel espace de nombres pour l'administration de l'IANA.

6. Références

6.1 Références normatives

[RFC0793] J. Postel (éd.), "Protocole de commande de transmission – Spécification du protocole du programme Internet DARPA", STD 7, septembre 1981.

[RFC0896] J. Nagle, "Contrôle de l'encombrement dans l'inter-réseau IP/TCP", janvier 1984. *(Historique)*

[RFC1750] D. Eastlake 3rd et autres, "Recommandations d'aléa pour la sécurité", décembre 1994. *(Info., remplacée par la* RFC4086)

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

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

[RFC2486] B. Aboba, M. Beadles, "Identifiant d'accès réseau", janvier 1999. (*Obsolète, voir* RFC4282) *(P.S.)*

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

[RFC2883] S. Floyd et autres, "Extension à l'option d'accusé de réception sélectif (SACK) pour TCP", juillet 2000. *(P.S.)*

[RFC2960] R. Stewart et autres, "Protocole de transmission de commandes de flux", octobre 2000. (*Obsolète, voir* RFC4960) *(P.S.)*

[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.)*

[RFC3074] B. Volz et autres, "Algorithme DHC d'équilibrage de charge", février 2001. *(P.S.)*

[RFC3124] H. Balakrishnan et S Seshan, "Le gestionnaire d'encombrement", juin 2001. *(P.S.)*

6.2 Références pour information

[RFC2246] T. Dierks et C. Allen, "Protocole TLS version 1.0", janvier 1999.

[RFC2401] S. Kent et R. Atkinson, "Architecture de sécurité pour le protocole Internet", novembre 1998. (*Obsolète, voir* RFC4301)

[RFC2409] D. Harkins et D. Carrel, "L'échange de clés Internet (IKE)", novembre 1998. *(Obsolète, voir* RFC4306)

[RFC2607] B. Aboba, J. Vollbrecht, "Chaînage de mandataire et mise en œuvre de politique dans l'itinérance", juin 1999. *(Info.)*

[RFC2861] M. Handley, J. Padhye, S. Floyd, "Validation de fenêtre d'encombrement TCP", juin 2000. *(Expérimentale)*

[RFC2865] C. Rigney et autres, "Service d'authentification à distance de l'utilisateur appelant (RADIUS)", juin 2000. (*MàJ par* RFC2868, RFC3575, RFC5080) *(D.S.)*

[RFC2866] C. Rigney, "Comptabilité RADIUS", juin 2000. (*MàJ par* RFC2867, RFC5080) *(Information)*

[RFC2914] S. Floyd, "Principes du contrôle d'encombrement", BCP 41, septembre 2000.

[RFC2975] B. Aboba, J. Arkko, D. Harrington, "Introduction à la gestion comptable", octobre 2000. *(Information)*

[RFC3390] M. Allman, S. Floyd, C. Partridge, "Augmentation de la fenêtre initiale de TCP", octobre 2002. *(P.S.)*

[Congest] Jacobson, V., "Congestion Avoidance and Control", Computer Communication Review, vol. 18, n° 4, pages 314-329, août 1988. ftp://ftp.ee.lbl.gov/papers/congavoid.ps.Z

[Paxson] Paxson, V., "Measurement and Analysis of End-to-End Internet Dynamics", Ph.D. Thesis, Computer Science Division, University of California, Berkeley, avril 1997.

Appendice A. Algorithme de chien de garde détaillé

Dans cet Appendice "Algorithme de chien de garde détaillé" (DWA, *Detailed Watchdog Algorithm*) la structure de contrôle de mémoire qui contient toutes les informations concernant un homologue spécifique est appelée un bloc de contrôle d'homologue (PCB, *Peer Control Block*). Le PCB contient les champs suivants :

État :
OKAY : la connexion est active.
SUSPECT : la reprise sur défaillance a été initiée sur la connexion.
DOWN : la connexion a été fermée.
REOPEN : on tente de rouvrir une connexion fermée.
INITIAL : l'état initial du pcb à sa création. Le pcb n'a jamais été ouvert.

Variables :
Pending : réglé à VRAI si il y a une demande de chien de garde en cours sans réponse.
Tw : valeur du temporisateur de chien de garde.
NumDWA : nombre de DWA reçus durant REOPEN.

Tw est le temporisateur de chien de garde, mesuré en secondes. Chaque seconde, Tw est décrémenté. Lorsque il atteint 0, l'événement OnTimerElapsed (voir ci-dessous) est invoqué. Le pseudo code pour l'algorithme est inclus dans les pages suivantes.

```
SetWatchdog()
{
/* SetWatchdog() est invoqué chaque fois qu'il est nécessaire de réinitialiser la valeur du temporisateur de chien de garde
     Tw. La valeur du temporisateur de chien de garde est calculée sur la base de la valeur initiale par défaut TWINIT et
     d'une gigue dans la gamme de -2 à 2 secondes. La valeur par défaut de TWINIT est 30 secondes, et NE DOIT PAS être
     inférieure à 6 secondes. */

   Tw=TWINIT -2.0 + 4.0 * random() ;
   SetTimer(Tw) ;
   return ;
}

/* OnReceive() est invoqué chaque fois qu'un message est reçu de l'homologue. Ce message PEUT être une demande ou
     une réponse, et peut inclure des messages DWR et DWA. Pending (en cours est supposé être une variable globale. */

OnReceive(pcb, msgType)
{
  si (msgType == DWA) {
      Pending = FAUX;
      }
  switch (pcb->Status){
  cas OKAY:
      SetWatchdog();
      break;
  cas SUSPECT:
      pcb->Status = OKAY;
      Failback(pcb);
      SetWatchdog();
      break;
  cas REOPEN:
      si (msgType == DWA) {
        NumDWA++;
        si (NumDWA == 3) {
          pcb->status = OKAY;
          Failback();
        }
      } autrement {
        Throwaway(le paquet reçu);
      }
      break;
  cas INITIAL:
  cas DOWN:
      Throwaway(le paquet reçu);
      break;
  default:
      error("Ne devrait pas être là !");
      break;
  }
}

/* OnTimerElapsed() est invoqué chaque fois que Tw atteint zéro (0). */

OnTimerElapsed(pcb)
```

```
{
   switch (pcb->status){
     cas OKAY:
       si (!Pending) {
         SendWatchdog(pcb);
         SetWatchdog();
         Pending = VRAI;
         break;
       }
       pcb->status = SUSPECT;
       FailOver(pcb);
       SetWatchdog();
       break ;
       cas SUSPECT:
       pcb->status = DOWN;
       CloseConnection(pcb);
       SetWatchdog();
       break;
     cas INITIAL:
     cas DOWN:
       AttemptOpen(pcb);
       SetWatchdog();
       break;
     cas REOPEN:
       si (!Pending) {
         SendWatchdog(pbc);
         SetWatchdog();
         Pending = TRUE;
         break;
       }
       si (NumDWA < 0) {
         pcb->status = DOWN;
         CloseConnection(pcb);
       } autrement {
         NumDWA = -1;
       }
       SetWatchdog();
       break;
     default:
      error("Ne devrait pas être là !");
       break;
     }
}

/* OnConnectionUp() est invoqué chaque fois qu'une connexion est activée. */

OnConnectionUp(pcb)
{
   switch (pcb->status){
     cas INITIAL:
       pcb->status = OKAY;
       SetWatchdog();
       break;
     cas DOWN:
       pcb->status = REOPEN;
       NumDWA = 0;
       SendWatchdog(pcb);
       SetWatchdog();
       Pending = VRAI;
       break;
     default:
      error("Ne devrait pas être là !");
       break;
```

```
    }
}

/* OnConnectionDown() est invoqué chaque fois qu'une connexion est fermée. */

OnConnectionDown(pcb)
{
   pcb->status = DOWN;
   CloseConnection();
   switch (pcb->status){
     cas OKAY:
       Failover(pcb);
       SetWatchdog();
       break;
     cas SUSPECT:
     cas REOPEN:
       SetWatchdog();
       break;
     default:
      error("Ne devrait pas être là !");
       break;
     }
}

/*  Voici l'automate à états équivalent au code ci-dessus :
```

État	**Événement**	**Actions**	**Nouvel état**
OKAY	Reçoit DWA	En cours = FAUX	
		SetWatchdog()	OKAY
OKAY	Ne reçoitpas de DWA	SetWatchdog()	OKAY
SUSPECT	Reçoit DWA	En cours = FAUX	
		Failback() SetWatchdog()	OKAY
SUSPECT	Ne reçoit pas de DWA	Failback() SetWatchdog()	OKAY
REOPEN	Reçoit DWA & NumDWA == 2	En cours = FAUX	
		Failback() NumDWA++	OKAY
REOPEN	Reçoit DWA & NumDWA < 2	En cours = FAUX	
		NumDWA++	REOPEN
REOPEN	Ne reçoit pas de DWA	Throwaway()	REOPEN
INITIAL	Reçoit DWA	En cours = FAUX	
		Throwaway()	INITIAL
INITIAL	Ne reçoit pas de DWA	Throwaway()	INITIAL
DOWN	Reçoit DWA	En cours = FAUX	
		Throwaway()	DOWN
DOWN	Ne reçoit pas de DWA	Throwaway()	DOWN
OKAY	Expiration du tempo. & !En cours	SendWatchdog() SetWatchdog()	
		En cours = VRAI	OKAY
OKAY	Expiration du tempo. & En cours	Failover()	
		SetWatchdog()	SUSPECT
SUSPECT	Expiration du temporisateur	CloseConnection()	
		SetWatchdog()	DOWN
INITIAL	Expiration du temporisateur	AttemptOpen()	
		SetWatchdog()	INITIAL
DOWN	Expiration du temporisateur	AttemptOpen()	
		SetWatchdog()	DOWN
REOPEN	Expiration du tempo. & !En cours	SendWatchdog() SetWatchdog()	
		En cours = VRAI	REOPEN
REOPEN	Expiration du tempo. & En cours & NumDWA < 0	CloseConnection() SetWatchdog()	DOWN
REOPEN	Expiration du tempo. & En cours & NumDWA ≥ 0	NumDWA = -1 SetWatchdog()	REOPEN
INITIAL	Connexion active	SetWatchdog()	OKAY

DOWN	Connexion active	NumDWA = 0 SendWatchdog() SetWatchdog() En cours = VRAI	REOPEN
OKAY	Connexion morte	CloseConnection() Failover() SetWatchdog()	DOWN
SUSPECT	Connexion morte	CloseConnection() SetWatchdog()	DOWN
REOPEN	Connexion morte	CloseConnection() SetWatchdog()	DOWN

*/

Appendice B. Agents AAA

Comme décrit dans les [RFC2865] et [RFC2607], les agents AAA sont devenus courants afin de prendre en charge des services comme l'itinérance et les réseaux à utilisation partagée. De tels agents sont utilisés pour l'authentification / autorisation, ainsi que pour la comptabilité [RFC2975].

Les agents AAA incluent les relais, les mandataires, les redirections, les mandataires à livraison différée, les mandataires de couche transport.

Le comportement de couche transport de chacun de ces agents est décrit ci-dessous.

B.1 Relais et mandataires

Bien que le comportement de couche application des relais et mandataires soit différent, à la couche transport le comportement est similaire. Dans les deux cas, deux connexions sont établies : une provenant du client AAA (NAS) au relais/mandataire, et une autre du relais/mandataire au serveur AAA. Le relais/mandataire ne répond pas à une demande du client tant qu'il n'a pas reçu une réponse du serveur. Comme les deux connexions sont découplées, la conversation de bout en bout entre le client et le serveur ne peut pas s'auto temporiser.

Comme le transport AAA est normalement piloté par l'application, il n'y a souvent pas assez de trafic pour permettre le partage des ACK. Par suite, l'algorithme de Nagle est rarement déclanché, et des ACK retardés peuvent composer la moitié du trafic. Donc, les protocoles AAA qui fonctionnent sur un transport fiable vont voir le trafic de paquets presque doubler par rapport à celui rencontré avec un transport UDP. Comme les paramètres de ACK (comme la valeur du temporisateur de ACK retardés) sont normalement fixés par la mise en œuvre TCP et ne sont pas réglables par l'application, on ne peut pas y faire grand chose.

Un schéma typique de conversation entre un NAS, un mandataire et un serveur est montré ci-dessous :

Instant	**NAS**	**Relais/mandataire**	**Serveur**
0	Demande ---------->		
OTTnp + Tpr		Demande ------->	
OTTnp + TdA		ACK retardé <-------	
OTTnp + OTTps + Tpr + Ts			Réponse/ACK <-------
OTTnp + OTTps + Tpr + Tsr + OTTsp + TpR		Réponse <-------	
OTTnp + OTTps + Tpr + Tsr + OTTsp + TdA		ACK retardé<br------->	
OTTnp + OTTps + OTTsp + OTTpn + Tpr + Tsr + TpR + TdA	ACK retardé ------->		

Légende :
OTT = durée d'un aller simple
OTTnp = durée d'un aller simple (du NAS au relais/mandataire)

OTTpn = durée d'un aller simple (du relais/mandataire au NAS)
OTTps = durée d'un aller simple (du relais/mandataire au serveur)
OTTsp = durée d'un aller simple (du serveur au relais/mandataire)
TdA = temporisateur d'ACK retardé
Tpr = temps de traitement d'une demande par le relais/mandataire
TpR = temps de traitement d'une réponse par le relais/mandataire
Tsr = temps de traitement d'une demande par le serveur
À l'instant 0, le NAS envoie une demande au relais/mandataire. Ignorant le moment de mise ne série, la demande arrive au relais/mandataire à l'instant OTTnp, et le relais/mandataire prend un Tpr supplémentaire afin de transmettre la demande au serveur de rattachement. À l'instant TdA après avoir reçu la demande, le relais/mandataire envoie un ACK retardé. L'ACK retardé est envoyé, plutôt que d'être porté sur la réponse, tant que TdA < OTTps + OTTsp + Tpr + Tsr + TpR.

Normalement Tpr < TdA, de sorte que l'ACK retardé est envoyé après que le relais/mandataire a transmis la demande au serveur, mais avant que le relais/mandataire ait reçu la réponse du serveur. Cependant, selon la mise en œuvre TCP sur le relais/mandataire et le moment où la demande est reçue, il est aussi possible que l'ACK retarde soit envoyé avant la transmission de la demande.

À l'instant OTTnp + OTTps + Tpr, le serveur reçoit la demande, et Tsr plus tard, il génère la réponse. Lorsque Tsr < TdA, la réponse va contenir un ACK porté. Cependant, selon la responsivité du serveur et la mise en œuvre de TCP, l'ACK et la réponse peuvent être envoyés séparément. Cela peut se produire, par exemple, lorsque une base de données ou un système de mémorisation lents doivent être consultés avant d'envoyer la réponse.

À l'instant OTTnp + OTTps + OTTsp + Tpr + Tsr la réponse/ACK atteint le relais/mandataire, qui prend alors TpR en plus pour transmettre la réponse au NAS. À TdA après la réception de la réponse, le relais/mandataire génère un ACK retardé. Normalement TpR < TdA de sorte que l'ACK retardé est envoyé au serveur après que le relais/mandataire transmet la réponse au NAS. Cependant, selon les circonstances et la mise en œuvre TCP du relais/mandataire, l'ACK retardé peut être envoyé en premier.

Comme avec un ACK retardé envoyé en réponse à une demande, qui peut être porté si la réponse peut être reçue assez rapidement, le portage de l'ACK envoyé en réponse à une réponse du serveur n'est possible que si du trafic de demandes supplémentaire est disponible. Cependant, dû au fort espacement entre les paquets dans les scénarios AAA normaux, ceci est peu probable sauf si le protocole AAA prend en charge un ACK de réponse.

À l'instant OTTnp + OTTps + OTTsp + OTTpn + Tpr + Tsr + TpR le NAS reçoit la réponse. TdA plus tard, un ACK retardé est généré.

B.2 Redirections

Les redirections opèrent en référant un NAS au serveur AAA, permettant au NAS de parler directement au serveur AAA. Comme une connexion de transport directe est établie, la connexion de bout en bout va s'auto temporiser.

Avec les redirections, les ACK retardés sont moins fréquents qu'avec les mandataires de couche application car la redirection et le serveur vont normalement porter les réponses avec les ACK.

La séquence des événements est la suivante :

Instant	**NAS**	**Redirection**	**Serveur**
0	Demande ------>		
OTTnp + Tp		Redirect/ACK <-------	
OTTnp + Tpr + OTTpn + Tnr	Demande ------>		
OTTnp + OTTpn + Tpr + Tsr + OTTns			Réponse/ACK <-------
OTTnp + OTTpn + OTTns + OTTsn + Tpr + Tsr + TdA	ACK retardé ------>		

Légende :
OTT = durée d'un aller simple
OTTnp = durée d'un aller simple (du NAS au relais/mandataire)

OTTpn = durée d'un aller simple (du relais/mandataire au NAS)
OTTns = durée d'un aller simple (du NAS au serveur)
OTTsn = durée d'un aller simple (du serveur au NAS)
TdA = temporisateur d'ACK retardé
Tpr = temps de traitement d'une redirection
Tnr = temps de traitement d'une redirection par le NAS
Tsr = temps de traitement d'une demande par le serveur

B.3 Mandataires à livraison différée

Avec un mandataire à livraison différée, le mandataire peut envoyer une réponse au NAS avant de transmettre la demande au serveur. Bien que les mandataires à livraison différée soient le plus fréquemment déployés pour la comptabilité [RFC2975], ils peuvent aussi être utilisés pour mettre en œuvre une politique d'authentification/autorisation, comme décrit dans la [RFC2607].

Comme noté dans la [RFC2975], les mandataires à livraison différée peuvent avoir un effet négatif sur la fiabilité comptable. En envoyant une réponse au NAS sans en avoir reçu une du serveur de comptabilité, les mandataires à livraison différée trompent le NAS en lui faisant penser que la demande de comptabilité a été acceptée par le serveur de comptabilité alors que ce n'est pas le cas. Par suite, le NAS peut supprimer un paquet comptable d'une mémorisation non volatile avant qu'il ait été accepté par le serveur de comptabilité. Cela laisse le mandataire responsable de la livraison des paquets de comptabilité. Si le mandataire implique des parties mobiles (par exemple, un pilote de disque) et pas le NAS, la fiabilité globale du système peut être réduite. Par suite, les mandataires à livraison différée NE DEVRAIENT PAS être utilisés.

La séquence des événements est la suivante :

Instant	**NAS**	**Mandataire**	**Serveur**
0	Demande ------->		
OTTnp + TpR		Réponse/ACK <-------	
OTTnp + Tpr		Demande ------->	
OTTnp + OTTph + Tpr + Tsr			Réponse/ACK <-------
OTTnp + OTTph + Tpr + Tsr + OTThp + TpR		Réponse <-------	
OTTnp + OTTph + Tpr + Tsr + OTThp + TdA		ACK retardé ------->	
OTTnp + OTTph + OTThp + OTTpn + Tpr + Tsr + TpR + TdA	ACK retardé ------->		

Légende :
OTT = durée d'un aller simple
OTTnp = durée d'un aller simple (du NAS au relais/mandataire)
OTTpn = durée d'un aller simple (du relais/mandataire au NAS)
OTTph = durée d'un aller simple (du mandataire au serveur de rattachement)
OTThp = durée d'un aller simple (du serveur de rattachement au mandataire)
TdA = temporisateur d'ACK retardé
Tpr = temps de traitement d'une demande par le mandataire
TpR = temps de traitement d'une réponse par le mandataire
Tsr = temps de traitement d'une demande par le serveur

B.4 Mandataires de couche transport

En plus d'agir comme mandataires à la couche application, les mandataires de couche transport transmettent les ACK de transport entre le client et le serveur AAA. Cela fusionne les connexions client-mandataire et mandataire-serveur en une seule connexion qui se comporte comme si elle opérait de bout en bout, exhibant l'auto temporisation. Cependant, comme les mandataires de transport opèrent à la couche transport, ils ne peuvent pas être mis en œuvre purement comme applications et sont rarement déployés.

Avec un mandataire de transport, la séquence des événements est la suivante :

Instant	NAS	Mandataire	Serveur de rattachement
0	Demande ------->		
OTTnp + Tpr		Demande ------->	
OTTnp + OTTph + Tpr + Tsr			Réponse/ACK <-------
OTTnp + OTTph + Tpr + Tsr + OTThp + TpR		Réponse/ACK <-------	
OTTnp + OTTph + OTThp + OTTpn + Tpr + Tsr + TpR + TdA	ACK retardé ------->		
OTTnp + OTTph + OTThp + OTTpn + Tpr + Tsr + TpR + TpD		ACK retardé ------->	

Légende :
OTT = durée d'un aller simple
OTTnp = durée d'un aller simple (du NAS au mandataire)
OTTpn = durée d'un aller simple (du mandataire au NAS)
OTTph = durée d'un aller simple (du mandataire au serveur de rattachement)
OTThp = durée d'un aller simple (du serveur de rattachement au mandataire)
TdA = temporisateur d'ACK retardé
Tpr = temps de traitement d'une demande par le mandataire
TpR = temps de traitement d'une réponse par le mandataire
Tsr = temps de traitement d'une demande par le serveurTpD = temps de traitement d'un ack retardé par le mandataire

Déclaration de propriété intellectuelle

Remerciements

Merci à Allison Mankin de AT&T, Barney Wolff de Databus, Steve Rich de Cisco, Randy Bush de AT&T, Bo Landarv de IP Unplugged, Jari Arkko de Ericsson, et Pat Calhoun de Blackstorm Networks pour les discussions fructueuses sur le transport AAA.

Adresse des auteurs

Bernard Aboba
Microsoft Corporation
One Microsoft Way
Redmond, WA 98052
téléphone : +1 425 706 6605
mél : bernarda@microsoft.com

Jonathan Wood
Sun Microsystems, Inc.
901 San Antonio Road
Palo Alto, CA 94303
US
mél : jonwood@speakeasy.net

Déclaration complète de droits de reproduction

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