Groupe de travail Réseau
Request for Comments : 1323
RFC rendues obsolètes : RFC1072, RFC1185
Traduction Claude Brière de L'Isle

V. Jacobson, LBL
R. Braden, ISI
D. Borman, Cray Research
mai 1992

Extensions TCP pour hautes performances

Statut de ce mémoire
La présente RFC spécifie un protocole en cours de normalisation par l'IAB pour la communauté de l'Internet et appelle à des discussions et suggestions pour son amélioration. Prière de se référer à l'édition en cours des "normes officielles des protocoles de l'IAB" pour l'état de normalisation et le statut de ce protocole. La distribution du présent mémoire n'est soumise à aucune restriction.
(La présente traduction incorpore les errata n° 3685 et 2278)

Résumé
Le présent mémoire présente un ensemble d'extensions à TCP pour améliorer ses performances sur des chemins à grande bande passante*délai et pour fournir un fonctionnement fiable sur des chemins à très haut débit. Il définit de nouvelles options TCP pour les fenêtres adaptées et les horodatages, qui sont conçues pour fournir un inter fonctionnement compatible avec les TCP qui ne mettent pas en œuvre les extensions. Les horodatages sont utilisés pour deux mécanismes distincts : la mesure du temps d'aller-retour (RTTM, *Round Trip Time Measurement*) et la protection contre le retour à zéro du numéro de séquence (PAWS, *Protect Against Wrapped Sequences*). Les accusés de réception sélectifs ne sont pas inclus dans ce mémoire.

Le présent mémoire combine et subroge les RFC1072 et RFC1185, ajoutant des précisions supplémentaires et une spécification plus détaillée. L'Appendice C résume les changements par rapport aux RFC précédentes.

Table des Matières

1. Introduction

Le protocole TCP [RFC0793] a été conçu pour fonctionner de façon fiable sur presque tout support de transmission sans considération du taux de transmission, du délai, de la corruption, de la duplication, ou du réarrangement des segments. La production de mises en œuvre de TCP s'adapte actuellement aux taux de transfert dans la gamme de 100 bit/s à 10**7 bit/s et des délais d'aller-retour dans la gamme de 1 ms à 100 s. Des travaux récents sur les performances de TCP ont montré que TCP peut bien fonctionner sur divers chemins de l'Internet, allant de canaux I/O à 800 Mbit/s à des modems à numérotation à 300 bit/s [Jacobson88a].

L'introduction de fibres optiques a résulté en vitesses de transmission toujours plus élevées, et les chemins les plus rapides sont en train de sortir du domaine pour lequel TCP été prévu à l'origine. Le présent mémoire définit un ensemble de modestes extensions à TCP pour étendre le domaine de son application afin de répondre à cette capacité réseau croissante. Il se fonde sur les [RFC1072] et [RFC1185] et les rend obsolètes.

On ne peut pas répondre en une ligne à la question: "À quelle vitesse TCP peut-il aller ?". Il y a deux sortes distinctes de problèmes, les performances et la fiabilité, et chacune dépend de paramètres différents. On les exposera tour à tour.

1.1 Performances de TCP

Les performances de TCP ne dépendent pas du taux de transfert lui-même, mais plutôt du produit du taux de transfert et du délai d'aller-retour. Ce produit "bande passante*délai" mesure la quantité de données qui vont "remplir le tuyau" ; c'est l'espace de mémoire tampon requis chez l'envoyeur et le receveur pour obtenir le débit maximum sur la connexion TCP sur le chemin, c'est-à-dire, la quantité de données non acquittées que TCP doit traiter afin de garder le tuyau plein. Le problème des performances de TCP survient lorsque le produit bande passante*délai est grand. On se réfère à un chemin Internet fonctionnant dans cette région comme à un "tuyau long, et gros", et à un réseau contenant ce chemin comme un "LFN" (prononcé "éléphan(t)").

Les canaux satellites à grande capacité de paquets (par exemple, le réseau large bande du DARPA) sont des LFN. Par exemple, un canal satellite DS1-speed a un produit bande passante*délai de 10**6 bits ou plus ; cela correspond à cent segments TCP en cours de 1200 octets chacun. Les chemins terrestres en fibre optique vont aussi tomber dans la classe des LFN ; par exemple, un délai à travers le pays de 30 ms à la bande passante DS3 (45 Mbit/s) excède aussi les 10**6 bits.

Il y a trois problèmes fondamentaux de performances avec le TCP actuel sur chemins LFN :

(1) Limite de taille de fenêtre
L'en-tête TCP utilise un champ de 16 bits pour rapporter la taille de la fenêtre du receveur à l'envoyeur. Donc, la plus grande fenêtre qui peut être utilisée est de 2**16 = 65 koctets. Pour tourner ce problème, la Section 2 du présent mémoire définit une nouvelle option TCP, "Adaptation de fenêtre", pour permettre des fenêtres supérieures à 2**16. Cette option définit un facteur d'échelle implicite, qui est utilisé pour multiplier la valeur de la taille de fenêtre trouvée dans un en-tête TCP pour obtenir la vraie taille de la fenêtre.

(2) Récupération des pertes
Les pertes de paquet dans un LFN peuvent avoir un effet catastrophique sur le débit. Jusqu'à il y a peu, des mises en œuvre de TCP fonctionnant correctement causaient l'écroulement du tuyau de données à chaque perte de paquet, et exigeaient une action de démarrage lent pour récupérer. Récemment, les algorithmes de retransmission rapide et de récupération rapide [Jacobson90c] ont été introduits. Leur effet combiné est de récupérer d'une perte de paquet par fenêtre, sans écrouler le tuyau. Cependant, plus d'une perte de paquet par fenêtre résulte normalement en une fin de temporisation de retransmission et en l'écroulement résultant du tuyau et le démarrage lent.

Étendre la taille de fenêtre pour correspondre à la capacité d'un LFN résulte en un accroissement correspondant de la probabilité d'abandon de plus d'un paquet par fenêtre. Cela pourrait avoir un effet dévastateur sur le débit de TCP sur un LFN. De plus, si un mécanisme de contrôle d'encombrement fondé sur une forme d'abandon aléatoire a été introduit dans les passerelles, des abandons de paquets espacés au hasard vont devenir courants, augmentant éventuellement la probabilité d'abandon de plus d'un paquet par fenêtre.

Pour généraliser le mécanisme de retransmission/récupération rapide afin de traiter plusieurs abandons de paquets par fenêtre, les accusés de réception sélectifs sont nécessaires. À la différence des accusés de réception cumulatifs normaux de TCP, les accusés de réception sélectifs donnent à l'envoyeur une peinture complète des segments qui sont en file d'attente chez le receveur et de ceux qui ne sont pas encore arrivés. Certaines évidences en faveur des accusés de réception sélectifs ont été publiées dans [NBS85], les accusés de réception sélectifs ont été inclus dans un certain nombre de protocoles expérimentaux de l'Internet -- VMTP [RFC1045], NETBLT [RFC0988], et RDP [RFC0908], et

proposés pour OSI TP4 [NBS85]. Cependant, dans le régime non LFN, les accusés de réception sélectifs réduisent le nombre de paquets retransmis mais n'améliorent pas les performances par ailleurs, rendant leur complexité de valeur discutable. Cependant, il est prévu que les accusés de réception sélectifs deviennent de plus en plus importants dans le régime LFN.

La RFC-1072 définissait une nouvelle option TCP "SACK" pour envoyer un accusé de réception sélectif. Cependant, il reste des questions techniques importantes à résoudre concernant aussi bien le format que la sémantique de l'option SACK. Donc, SACK a été omis de ce paquetage d'extensions. On espère que SACK pourra se "raccrocher" durant le processus de normalisation.

(3) Mesures de l'aller-retour
TCP met en œuvre la livraison fiable des données en retransmettant les segments qui ne sont pas acquittés dans un certain intervalle de temporisation de retransmission (RTO, *retransmission timeout*). La détermination dynamique précise d'un RTO approprié est essentielle pour les performances de TCP. Le RTO est déterminé en estimant la moyenne et la variance du temps d'aller-retour (RTT, *round-trip time*) mesuré, c'est-à-dire, l'intervalle de temps entre l'envoi d'un segment et la réception d'un accusé de réception pour lui [Jacobson88a].

Le paragraphe 3.2 introduit une nouvelle option TCP, "Horodatages", et définit ensuite un mécanisme qui utilise cette option qui permet que presque tous les segments, y compris les retransmissions, soient chronométrés pour un coût de calcul négligeable. On utilisera le mnémonique RTTM (*Round Trip Time Measurement*) pour ce mécanisme, pour le distinguer des autres utilisations de l'option Horodatage.

1.2 Fiabilité de TCP

On passe maintenant des performances à la fiabilité. Les taux de transfert élevés font passer les performances de TCP par le produit bande passante * délai. Cependant, le taux de transfert élevé seul ne peut menacer la fiabilité de TCP en violant les hypothèses sous-jacentes au mécanisme TCP de détection des dupliqués et de séquencement.

Une espèce d'erreur particulièrement sérieuse peut résulter d'une réutilisation accidentelle des numéros de séquence TCP dans des segments de données. Supposons qu'un "vieux segment dupliqué", par exemple, un segment de données dupliqué qui a été retardé dans des files d'attentes Internet, est livré au receveur au mauvais moment, de sorte que son numéro de séquence tombe quelque part dans la fenêtre en cours. Il n'y aura pas d'échec de somme de contrôle pour avertir de l'erreur, et le résultat pourrait être une corruption indétectée des données. La réception d'un vieux segment d'ACK dupliqué chez l'émetteur pourrait seulement être un peu moins grave : il va vraisemblablement verrouiller la connexion afin que cela ne puisse pas aller plus loin, forçant un RST *(rétablissement)* sur la connexion.

La fiabilité de TCP dépend de l'existence d'une limite à la durée de vie d'un segment : la "durée de vie maximum de segment" (MSL, *Maximum Segment Lifetime*). Une MSL est généralement requise par un protocole de transport fiable, car chaque champ Numéro de séquence doit être fini, et donc, tout numéro de séquence peut éventuellement être réutilisé. Dans la suite des protocoles de l'Internet, la limite de MSL est mise en application par un mécanisme de couche IP, la durée de vie" (TTL, *Time-To-Live*).

La duplication des numéros de séquence pourrait survenir de l'une des deux façons suivantes :

(1) Retour à zéro du numéro de séquence sur la connexion en cours
Un numéro de séquence TCP contient 32 bits. À un taux de transfert suffisamment élevé, l'espace de séquence de 32 bits peut "revenir à zéro" (faire un cycle) pendant le temps qu'un segment est retardé dans des files d'attentes.

(2) Précédente incarnation de la connexion
Supposons qu'une connexion se termine, soit par une séquence de clôture appropriée, soit à cause d'une panne de l'hôte, et que la même connexion (c'est-à-dire, utilisant la même paire de prises) soit immédiatement rouverte. Un segment retardé de la connexion terminée pourrait tomber dans la fenêtre en cours pour la nouvelle incarnation et être accepté comme valide.

Les dupliqués provenant d'incarnations antérieures, cas (2), sont évités en appliquant le MSL fixe actuel de la spécification TCP, comme expliqué au paragraphe 5.3 et à l'Appendice B. Cependant, le cas (1), qui évite la réutilisation des numéros de séquence au sein de la même connexion, requiert qu'une limite de MSL dépende du taux de transfert, et à des débits suffisamment élevés, un nouveau mécanisme est nécessaire.

Plus précisément, si la bande passante effective maximum à laquelle TCP est capable de transmettre sur un chemin particulier est B octets par seconde, la contrainte suivante doit alors être satisfaite pour un fonctionnement sans erreur :

2**31 / B > MSL (s) [1]

Le tableau suivant montre la valeur de Twrap = 2**31/B en secondes, pour quelques valeurs importantes de la bande passante B :

Réseau	B*8 (bit/s)	B (octets/s)	Twrap (s)
ARPANET	56 kbit/s	7 kbit/s	3*10**5 (~3,6 jours)
DS1	1,5 Mbit/s	190 kbit/s	10**4 (~3 heures)
Ethernet	10 Mbit/s	1,25 Mbit/s	1700 (~30 min)
DS3	45 Mbit/s	5,6 Mbit/s	380
FDDI	100 Mbit/s	12,5 Mbit/s	170
Gigabit	1 Gbit/s	125 Mbit/s	17

Il est clair que le retour à zéro de l'espace de numéros de séquence n'est pas un problème pour la commutation de paquets à 56 kbit/s ou même pour les Ethernets à 10 Mbit/s. D'un autre côté, aux vitesses DS3 et FDDI, Twrap est comparable aux 2 minutes de MSL supposées par la spécification TCP [RFC0793]. En allant vers des vitesses de l'ordre du gigabit, Twrap devient trop petit pour la mise en application fiable du mécanisme de TTL de l'Internet.

Le champ de 16 bits de la fenêtre de TCP limite la bande passante effective B à 2**16/RTT, où RTT est le temps de l'aller-retour en secondes [RFC1110]. Si le RTT est assez grand, cela limite B à une valeur qui satisfait la contrainte [1] pour une grande valeur de MSL. Par exemple, considérons un cœur de réseau transcontinental avec un RTT de 60 ms (établi par les lois de la physique). Avec le produit bande passante*délai limité à 64 kbit par la taille de fenêtre TCP, B est alors limité à 1,1 Mbit/s, si élevé que soit le taux de transfert théorique du chemin. Cela correspond à faire faire un cycle à l'espace de numéros de séquence en Twrap = 2000 s, ce qui est sûr dans l'Internet d'aujourd'hui.

Il est important de comprendre que le coupable n'est pas la plus grande fenêtre mais plutôt la bande passante élevée. Par exemple, considérons un LAN FDDI (très grand) avec un diamètre de 10 km. En utilisant la vitesse de la lumière, on peut calculer le RTT à travers l'anneau comme étant (2*10**4)/(3*10**8) = 67 microsecondes, et le produit délai*bande passante est alors de 833 octets. Une connexion TCP à travers ce LAN en utilisant une fenêtre de seulement 833 octets va tourner avec le plein 100 Mbit/s et peut faire revenir à zéro l'espace de numéros de séquence en environ 3 minutes, très proche du MSL de TCP. Donc, la vitesse élevée seule peut causer un problème de fiabilité avec les retours à zéro de numéro de séquence, même sans fenêtre étendue.

Le protocole Delta-T de Watson [Watson81] inclut des mécanismes de couche réseau pour une application précise d'un MSL. À l'opposé, le mécanisme IP pour l'application du MSL est vaguement définie et encore plus vaguement mise en œuvre dans l'Internet. Il n'est donc pas sage de dépendre de la mise en application active du MSL pour les connexions TCP, et il serait irréaliste d'imaginer de régler les MSL plus bas que les valeurs actuelles (par exemple, que les 120 s spécifiées pour TCP).

Un remède possible à ce problème de cycles de l'espace de numéros de séquence serait d'augmenter la taille du champ TCP de numéro de séquence. Par exemple, le champ Numéro de séquence (et aussi le champ Accusé de réception) pourrait être allongé à 64 bits. Cela pourrait être fait soit en changeant l'en-tête TCP, soit au moyen d'une option supplémentaire.

La Section 5 présente un mécanisme différent, qu'on appelle protection contre le retour à zéro des numéros de séquence (PAWS, *Protect Against Wrapped Sequence numbers*) pour étendre la fiabilité de TCP aux taux de transferts bien au delà de la limite supérieure prévisible des bandes passantes du réseau. PAWS utilise l'option Horodatage de TCP définie à la Section 4 pour protéger contre les vieux dupliqués provenant de la même connexion.

1.3 Utilisation des options TCP

Les extensions définies dans le présents mémoire utilisent toutes les nouvelles options TCP. On doit régler deux problèmes possibles concernant l'utilisation des options TCP : (1) la compatibilité et (2) la redondance.

On doit prêter une grande attention à la compatibilité, c'est-à-dire, à l'interopération avec les mises en œuvre existantes. La seule option TCP définie précédemment, MSS, ne peut apparaître que sur un segment SYN. Chaque mise en œuvre devrait (et on s'attend à ce que la plupart le fassent) ignorer les options inconnues sur les segments SYN. Cependant, certaines mises en œuvre fautives de TCP pourraient être mises en panne par la première apparition d'une option sur un segment non SYN. Donc, pour chacune des extensions définies ci-dessous, les options TCP seront envoyées sur des segments non SYN uniquement lorsque un échange d'options sur les segments SYN a indiqué que les deux côtés comprennent l'extension. De plus, une option d'extension sera envoyée dans un segment <SYN,ACK> seulement si l'option correspondante a été reçue

dans le segment <SYN> initial.

Une question peut être soulevée sur la bande passante et le traitement de la redondance pour les options TCP. Ces options qui surviennent sur les segments SYN ne vont vraisemblablement pas causer de problèmes de performances. Ouvrir une connexion TCP exige l'exécution d'un code de casse particulière significatif, et le traitement des options ne va probablement pas accroître ce coût de façon significative.

D'un autre côté, une option Horodatage peut apparaître dans toutes données ou segment ACK, ajoutant 12 octets aux 20 octets de l'en-tête TCP. On espère que la bande passante économisée par la réduction des retransmissions inutiles va plus que compenser la bande passante du supplément d'en-tête.

Il y a aussi un problème avec le traitement de la redondance d'analyse du format variable aligné sur l'octet des options, particulièrement avec un CPU à architecture RISC. Pour répondre à ce souci, l'Appendice A contient une disposition recommandée pour les options dans les en-têtes TCP pour réaliser un alignement raisonnable des champs de données. Dans l'esprit de la prédiction d'en-tête, une mise en œuvre de TCP peut vérifier rapidement cette disposition et si elle est vérifiée, utiliser alors un chemin rapide. Les hôtes qui utilisent cette disposition canonique vont effectivement utiliser les options comme un ensemble de champs de format fixé ajouté à l'en-tête TCP. Cependant, pour conserver le cadre philosophique et protocolaire des options TCP, une mise en œuvre TCP doit être prête à analyser un champ d'option arbitraire, bien qu'avec une efficacité moindre.

Finalement, on observe que la plupart des mécanismes définis dans ce mémoire sont importants pour les LFN et/ou les réseaux à très haut débit. Pour les réseaux à faible débit, ce peut être une optimisation des performances de NE PAS utiliser ces mécanismes. Un fabricant de TCP concerné par l'optimisation des performances sur des chemins à bas débit pourrait considérer opportun de désactiver ces extensions pour les chemins à bas débit, ou de permettre à l'utilisateur ou gestionnaire d'installation de les désactiver.

2. Option d'adaptation de fenêtre TCP

2.1 Introduction

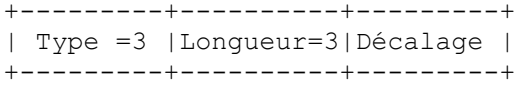
L'extension d'adaptation de fenêtre étend la définition de la fenêtre TCP à 32 bits et utilise ensuite un facteur d'adaptation pour porter ces valeurs de 32 bits dans le champ Fenêtre de 16 bits de l'en-tête TCP (SEG.WND dans la RFC-793). Le facteur d'adaptation est porté dans une nouvelle option TCP, Window Scale. Cette option n'est envoyée que dans un segment SYN (un segment avec le bit SYN établi) donc, l'adaptation de fenêtre est fixée dans chaque direction quand une connexion est ouverte. (Un autre choix de conception serait de spécifier l'adaptation de la fenêtre dans chaque segment. Il serait incorrect de n'envoyer une option d'adaptation de fenêtre que lorsque le facteur d'adaptation a changé, car une option TCP dans un segment d'accusé de réception ne sera pas livré de façon fiable (sauf si le ACK se trouve porté sur des données dans l'autre direction). Fixer l'adaptation lorsque la connexion ouvre présente l'avantage d'une plus faible redondance mais l'inconvénient que le facteur d'adaptation ne peut pas être changé durant la connexion.)

La fenêtre de réception maximum, et donc le facteur d'adaptation, est déterminée par l'espace maximum de mémoire tampon du receveur. Dans une mise en œuvre moderne normale, cet espace maximum de mémoire tampon, est réglé par défaut mais peut être outrepassé par un programme d'utilisateur avant l'ouverture d'une connexion TCP. Cela détermine le facteur d'adaptation, et donc aucun nouvel interface d'utilisateur n'est nécessaire pour l'adaptation de fenêtre.

2.2 Option Adaptation de fenêtre

Les trois octets de l'option Adaptation de fenêtre peuvent être envoyés dans un segment SYN par TCP. Cela répond à deux objets : (1) indiquer que TCP est prêt à faire l'adaptation de fenêtre à la fois en émission et en réception, et (2) communiquer un facteur d'adaptation à appliquer à sa fenêtre de réception. Donc, une mise en œuvre de TCP qui est prête à adapter sa fenêtre devrait envoyer l'option, même si son propre facteur d'adaptation est 1. Le facteur d'adaptation est limité à une puissance de deux et codé sous forme logarithmique, de sorte qu'il peut être mis en œuvre par une opération de glissement binaire.

Option TCP Adaptation de fenêtre (WSopt, *Window Scaling option*) :
Type : 3
Longueur : 3 octets



Cette option est une offre, non une promesse; les deux côtés doivent envoyer les options Adaptation de fenêtre dans leurs segments SYN pour activer l'adaptation de fenêtre dans l'une et l'autre direction. Si l'adaptation de fenêtre est activée, le TCP qui envoie cette option va alors décaler à gauche ses vraies valeurs de fenêtre de réception de 'shift.cnt' bits pour transmission dans SEG.WND. La valeur 'shift.cnt' peut être zéro (offrant d'adapter, tout en appliquant un facteur d'adaptation de 1 à la fenêtre de réception).

Cette option peut être envoyée dans un segment initial <SYN> (c'est-à-dire, un segment avec le bit SYN établi et le bit ACK à zéro). Elle peut aussi être envoyée dans un segment <SYN,ACK>, mais seulement si une option Adaptation de fenêtre a été reçue dans le segment <SYN> initial. Une option Adaptation de fenêtre dans un segment sans un bit SYN devrait être ignorée.

Le champ Fenêtre dans un segment SYN lui-même (c'est-à-dire, un segment <SYN> ou <SYN,ACK>) n'est jamais adapté.

2.3 Utilisation de l'option Adaptation de fenêtre

Un modèle de mise en œuvre d'adaptation de fenêtre suit, utilisant la notation de la [RFC0793] :

* Toutes les fenêtres sont traitées comme des quantités de 32 bits pour la mémorisation dans le bloc de contrôle de connexion et pour les calculs locaux. Cela inclut les valeurs de la fenêtre d'envoi (SND.WND) et de la fenêtre de réception (RCV.WND) ainsi que de la fenêtre d'encombrement.

* L'état de connexion est augmenté de deux comptes de décalage de fenêtre, Snd.Wind.Scale et Rcv.Wind.Scale, à appliquer, respectivement, aux champs de fenêtre entrante et sortante.

* Si TCP reçoit un segment <SYN> contenant une option Adaptation de fenêtre, il envoie sa propre option Adaptation de fenêtre dans le segment <SYN,ACK>.

* L'option Adaptation de fenêtre est envoyée avec shift.cnt = R, où R est la valeur que TCP voudrait utiliser pour sa fenêtre de réception.

* À réception d'un segment SYN avec une option Adaptation de fenêtre contenant shift.cnt = S, TCP règle Snd.Wind.Scale à S et règle Rcv.Wind.Scale à R ; autrement, il règle les deux Snd.Wind.Scale et Rcv.Wind.Scale à zéro.

* Le champ Fenêtre (SEG.WND) dans l'en-tête de chaque segment entrant, à l'exception des segments SYN, est glissé à gauche de Snd.Wind.Scale bits avant de mettre à jour SND.WND :

  SND.WND = SEG.WND << Snd.Wind.Scale

  (en supposant que les autres conditions de la RFC793 sont satisfaites, et en utilisant la notation C "<<" pour le glissement à gauche).

* Le champ Fenêtre (SEG.WND) de tout segment sortant, à l'exception des segments SYN, est glissé à droite de Rcv.Wind.Scale bits:

  SEG.WND = RCV.WND >> Rcv.Wind.Scale.

TCP détermine si un segment de données est "vieux" ou "nouveau" en vérifiant si son numéro de séquence est dans les 2**31 octets de la bordure gauche de la fenêtre, et si il ne l'est pas, élimine les données comme "vieilles". Pour s'assurer que de nouvelles données ne sont jamais à tort considérées comme vieilles et vice-versa, le bord gauche de la fenêtre de l'envoyeur doit être au plus à 2**31 du bord droit de la fenêtre du receveur. De même avec le bord droit de l'envoyeur et le bord gauche du receveur. Comme les bords droit et gauche de la fenêtre de l'envoyeur ou du receveur diffèrent de la taille de la fenêtre, et comme les fenêtres de l'envoyeur et du receveur peuvent être déphasées d'au plus la taille de la fenêtre, les contraintes ci-dessus impliquent que 2 * la taille maximum de fenêtre doit être moins que 2**31, ou fenêtre max < 2**30.

Comme fenêtre max est 2**S (où S est le compte de décalage d'adaptation) fois au plus 2**16 - 1 (la fenêtre maximum non adaptée) la fenêtre maximum est garantie d'être < 2*30 si S ≤ 14. Donc, le compte de décalage doit être limité à 14 (ce qui permet des fenêtres de 2**30 = 1 Goctet). Si une option Adaptation de fenêtre est reçue avec une valeur shift.cnt excédant 14, TCP devrait enregistrer l'erreur mais utiliser 14 au lieu de la valeur spécifiée.

Le facteur d'adaptation s'applique seulement au champ Fenêtre tel que transmis dans l'en-tête TCP ; chaque TCP utilisant

des fenêtres étendues va entretenir les valeurs de fenêtre en local comme des nombres de 32 bits. Par exemple, la "fenêtre d'encombrement" calculée par le démarrage lent et l'évitement d'encombrement n'est pas affectée par le facteur d'adaptation, de sorte que l'adaptation de fenêtre n'introduira pas de quantification dans la fenêtre d'encombrement.

3. Mesure du temps d'aller-retour (RTTM)

3.1 Introduction

Des estimations précises et actuelles du RTT sont nécessaires pour s'adapter aux conditions de trafic changeantes et pour éviter une instabilité connue sous le nom de "collapsus d'encombrement" [RFC0896] dans un réseau chargé. Cependant, une mesure précise du RTT peut être difficile aussi bien en théorie qu'à la mise en œuvre.

De nombreuses mises en œuvre de TCP fondent leurs mesures du RTT sur un échantillon d'un seul paquet par fenêtre. Bien que cela donne une approximation adéquate du RTT pour les petites fenêtres, il en résulte une estimation d'une pauvreté inacceptable pour le RTT d'un LFN. Si on regarde une estimation de RTT comme un problème de traitement de signal (ce qu'il est en réalité) un signal de données à une certaine fréquence, le taux de paquet, est échantillonné à une fréquence inférieure, le taux de fenêtre. Cette fréquence d'échantillonnage inférieure viole le critère de Nyquist et peut donc introduire un artifice "d'échantillon" dans l'estimation du RTT [Hamming77].

Un bon estimateur du RTT avec un calcul prudent de la temporisation de retransmission peut tolérer une erreur d'échantillonnage lorsque la fréquence d'échantillonnage est "proche" de la fréquence des données. Par exemple, avec une fenêtre de 8 paquets, le taux d'échantillon est 1/8 de la fréquence des données – moins d'un ordre de grandeur de différence. Cependant, lorsque la fenêtre est de dizaines ou centaines de paquets, l'estimation du RTT peut être sérieusement erronée, résultant en retransmissions parasites.

Si il y a des abandons de paquets, le problème empire. Zhang [Zhang86], Jain [Jain86] et Karn [Karn87] ont montré qu'il n'est pas possible d'accumuler des estimations fiables de RTT si les segments retransmis sont inclus dans l'estimation. Comme une pleine fenêtre de données aura été transmise avant une retransmission, tous les segments de la fenêtre devront être acquittés avant que le prochain échantillon de RTT puisse être prélevé. Cela signifie au moins le temps d'une fenêtre supplémentaire entre les mesures de RTT et, comme le taux d'erreur approche d'un par fenêtre de données (par exemple, 10**-6 erreurs par bit pour le réseau par satellite Wideband) il devient effectivement impossible d'obtenir une mesure valide de RTT.

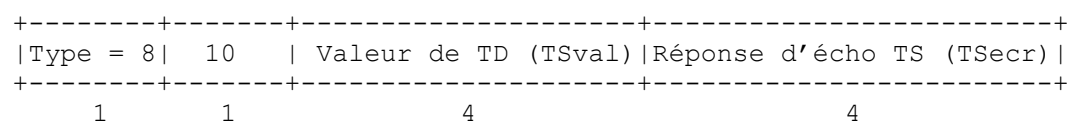
Une solution à ces problèmes, qui en fait simplifie substantiellement l'envoyeur, est la suivante : en utilisant les options TCP, l'envoyeur place un horodatage dans chaque segment de données, et le receveur reflète ces horodatages dans les segments ACK. Une seule soustraction donne alors à l'envoyeur une mesure précise du RTT pour chaque segment ACK (qui va correspondre à chaque autre segment de données, chez un receveur intelligent). On appelle cela le mécanisme de mesure du temps d'aller-retour (RTTM, *Round-Trip Time Measurement*).

Il est d'une importance vitale d'utiliser le mécanisme RTTM avec les grandes fenêtres ; autrement, la porte est ouverte à de dangereuses instabilités dues aux erreurs d'échantillonnage. De plus, l'option est probablement utile pour tous les TCP, car elle simplifie la tâche de l'envoyeur.

3.2 Option Horodatage TCP

TCP est un protocole symétrique, qui permet d'envoyer les données à tout moment dans l'une et l'autre direction, et donc l'écho d'horodatage peut se faire dans les deux directions. Pour la simplicité et la symétrie, on spécifie que les horodatages sont toujours envoyés dans les deux directions et qu'il leur est fait écho de même. Pour l'efficacité, on combine les champs Horodatage et réponse d'horodatage dans une seule option TCP Horodatage.

Option TCP Horodatage (TSopt, *TimeStamp option*) :
Type : 8
Longueur : 10 octets



L'option Horodatage porte deux champs Horodatage de quatre octets. Le champ Valeur d'horodatage (TSval) contient la

valeur actuelle de l'horloge d'horodatage du TCP qui envoie l'option.

Le champ Réponse d'écho d'horodatage (TSecr) n'est valide que si le bit ACK est établi dans l'en-tête TCP ; si il est valide, il fait écho à la valeur d'horodatage qui a été envoyée par le TCP distant dans le champ TSval d'une option Horodatage. Lorsque TSecr n'est pas valide, sa valeur doit être zéro. La valeur TSecr va généralement provenir de l'option Horodatage la plus récente reçue ; cependant, il y a des exceptions qui sont expliquées ci-dessous.

TCP peut envoyer l'option Horodatage (TSopt) dans un segment <SYN> initial (c'est-à-dire, un segment contenant un bit SYN et pas de bit ACK) et ne peut envoyer une TSopt dans d'autres segments que si il a reçu une TSopt dans le segment <SYN> initial pour la connexion.

3.3 Mécanisme de RTTM

La valeur d'horodatage à envoyer dans TSval est obtenue d'une horloge (virtuelle) qu'on appelle "horloge d'horodatage". Ses valeurs doivent être au moins approximativement proportionnelles à l'heure réelle, afin de mesurer le RTT réel.

L'exemple suivant illustre un flux de données unidirectionnel avec des segments qui arrivent en séquence sans perte. Ici, A, B, C... représentent des blocs de données qui occupent les blocs successifs de numéros de séquence, et ACK(A),... représente les accusés de réception cumulatifs correspondants. Les deux champs d'horodatage de l'option Horodatage sont montrés de façon symbolique par <TSval= x,TSecr=y>. Chaque champ TSecr contient la valeur reçue le plus récemment dans un champ TSval.

```
TCP A     TCP B
   <A,TSval=1,TSecr=120> ------>
            <---- <ACK(A),TSval=127,TSecr=1>
   <B,TSval=5,TSecr=127> ------>
            <---- <ACK(B),TSval=131,TSecr=5>
              . . . . . . . . . . . . . . . . . . . . .
    <C,TSval=65,TSecr=131> ------>
          <---- <ACK(C),TSval=191,TSecr=65>
           (etc.)
```

La ligne pointillée marque une pause (longue de 60 unités de temps) dans laquelle A n'a rien à envoyer. Noter que cette pause gonfle le RTT que B pourrait déduire en recevant TSecr=131 dans le segment de données C. Donc, dans les flux de données unidirectionnels, le RTTM en direction inverse mesure une valeur qui est gonflée des trous de l'envoi des données. Cependant, la règle suivante empêche une inflation résultante du RTT mesuré :

> Une valeur TSecr reçue dans un segment n'est utilisée pour mettre à jour la mesure du RTT moyen que si le segment accuse réception de nouvelles données, c'est-à-dire, seulement si elle avance le bord gauche de la fenêtre d'envoi.

Comme le TCP B n'envoie pas de données, le segment de données C n'accuse pas réception de nouvelles données quand il arrive à B. Donc, la mesure gonflée de RTTM n'est pas utilisée pour mettre à jour la mesure du RTTM de B.

3.4 À quel horodatage faire écho ?

Si plus d'une option Horodatage est reçue avant l'envoi d'aucun segment de réponse, le TCP doit choisir seulement une des TSval pour écho, ignorant les autres. Pour minimise l'état conservé par le receveur (c'est-à-dire, le nombre de Tsval non traitées) le receveur devrait être obligé de conserver au plus un horodatage dans le bloc de contrôle de connexion.

Il y a trois situations à considérer :

(A) ACK retardés.
De nombreux TCP n'accusent réception que tous les K^{ème} segments d'un groupe de segments arrivant dans un petit intervalle de temps ; cette politique est connue généralement comme "ACK retardés". Le TCP envoyeur de données doit mesurer le RTT effectif, incluant le temps additionnel dû aux ACK retardés, ou autrement il va retransmettre inutilement. Donc, quand des ACK retardés sont utilisés, le receveur devrait répondre avec le champ TSval provenant du premier segment non acquitté.

(B) Trou dans l'espace de numéros de séquence (un ou des segments ont été perdus).
L'envoyeur va continuer d'envoyer jusqu'à ce que la fenêtre soit remplie, et le receveur peut générer des ACK lorsque

ces segments déclassés arrivent (par exemple, pour aider la "retransmission rapide").

Le segment perdu est probablement un signe d'encombrement, et dans cette situation, l'envoyeur devrait être prudent dans ses retransmissions. De plus, il est préférable de surestimer que de sous estimer le RTT. Un ACK pour un segment déclassé devrait donc contenir l'horodatage provenant du segment le plus récent qui a avancé la fenêtre.

La même situation survient si des segments sont réarrangés dans le réseau.

(C) Un trou bouché dans l'espace de numéros de séquence.
Le segment qui bouche le trou représente la plus récente mesure des caractéristiques du réseau. D'un autre côté, un RTT calculé à partir d'un segment précédent inclurait probablement la temporisation de retransmission de l'envoyeur, biaisant gravement l'estimation moyenne de RTT de l'envoyeur. Donc, l'horodatage provenant du dernier segment (qui bouche le trou) doit recevoir un écho.

Un algorithme qui couvre tous les trois cas est décrit dans les règles suivantes pour le traitement de l'option Horodatage sur une connexion synchronisée :

(1) L'état de connexion est augmenté de deux intervalles de 32 bits : TS.Recent contient un horodatage à faire écho dans TSecr chaque fois qu'un segment est envoyé, et Last.ACK.sent contient le champ ACK provenant du dernier segment envoyé. Last.ACK.sent sera égal à RCV.NXT sauf quant des ACK ont été retardés.

(2) Si Last.ACK.sent tombe dans la gamme des numéros de séquence d'un segment entrant :
SEG.SEQ ≤ Last.ACK.sent < SEG.SEQ + SEG.LEN
alors la TSval provenant du segment est copiée à TS.Recent ; autrement, la TSval est ignorée.

(3) Lorsque une TSopt est envoyée, son champ TSecr est réglé à la valeur TS.Recent en cours.

Les exemples suivants illustrent ces règles. Ici, A, B, C... représentent des segments de données qui occupent des blocs successifs de numéros de séquence, et ACK(A),... représente les segments correspondants d'accusé de réception. Noter que ACK(A) a le même numéro de séquence que B. On ne montre qu'une direction de l'écho d'horodatage pour être clair.

o Les paquets arrivent en séquence, et certains des ACK sont retardés.
Dans le cas (A), il est fait écho de l'horodatage provenant du plus ancien segment non acquitté.

```
                          TS.Recent
<A, TSval=1> ------------------->                    1
<B, TSval=2> ------------------->                    1
<C, TSval=3> ------------------->                    1
     <---- <ACK(C), TSecr=1>          (etc.)
```

o Les paquets arrivent déclassés, et chaque paquet est acquitté.
Dans le cas (B), il est fait écho à l'horodatage provenant du dernier segment qui a avancé le bord gauche de fenêtre, jusqu'à ce qu'arrive le segment manquant ; il y est fait écho conformément au cas (C). La même séquence va survenir si les segments B et D sont perdus et retransmis.

```
                          TS.Recent
<A, TSval=1> ------------------->                    1
     <---- <ACK(A), TSecr=1>                         1
<C, TSval=3> ------------------->                    1
     <---- <ACK(A), TSecr=1>                         1
<B, TSval=2> ------------------->                    2
     <---- <ACK(C), TSecr=2>                         2
<E, TSval=5> ------------------->                    2
     <---- <ACK(C), TSecr=2>                         2
<D, TSval=4> ------------------->                    4
     <---- <ACK(E), TSecr=4>          (etc.)
```

4. Protection contre le retour de numéro de séquence (PAWS)

4.1 Introduction

Le paragraphe 4.2 décrit un mécanisme simple pour rejeter les vieux segments dupliqués qui pourraient corrompre une connexion TCP ouverte ; on appelle ce mécanisme "protection contre le retour de numéro de séquence (PAWS, *Protect Against Wrapped Sequence numbers*). PAWS fonctionne au sein d'une seule connexion TCP, utilisant l'état qui est

sauvegardé dans le bloc de contrôle de connexion. Le paragraphe 4.3 et l'Appendice C exposent les implications du mécanisme PAWS pour éviter des vieux dupliqués provenant de précédentes incarnations de la même connexion.

4.2 Mécanisme PAWS

PAWS utilise les mêmes options Horodatage TCP que le mécanisme RTTM décrit prédéterminer, et suppose que chaque segment TCP reçu (y compris les segments de données et ACK) contient un horodatage SEG.TSval dont les valeurs sont monotones non décroissantes dans le temps. L'idée de base est qu'un segment peut être éliminé comme vieux dupliqué si il est reçu avec un horodatage SEG.TSval inférieur à un horodatage reçu plus récemment sur cette connexion.

Dans les deux mécanismes PAWS et RTTM, les "horodatages" sont des entiers non signés de 32 bits dans un espace modulaire de 32 bits. Donc, "inférieur à" est défini de la même façon que le sont les numéros de séquence TCP, et les mêmes techniques de mise en œuvre s'appliquent. Si s et t sont des valeurs d'horodatage, s < t si 0 < (t - s) < 2**31, calculés dans une arithmétique de 32 bits non signés.

Le choix des horodatages entrants à sauvegarder pour cette comparaison doit garantir une valeur d'accroissement monotone. Par exemple, on peut sauvegarder l'horodatage provenant du segment qui a le dernier avancé le bord droit de la fenêtre de réception, c'est-à-dire, le segment en séquence le plus récent. À la place, on choisit la valeur TS.Recent introduite au paragraphe 3.4 pour le mécanisme RTTM, car utiliser une valeur commune pour PAWS et RTTM simplifie la mise en œuvre des deux. Comme expliqué au paragraphe 3.4, TS.Recent diffère de l'horodatage du dernier segment en séquence seulement dans le cas d'ACK retardés, et donc de moins d'une fenêtre. L'un ou l'autre choix protègera donc contre le retour à zéro du numéro de séquence.

RTTM a été spécifié de manière symétrique, afin que les horodatages TSval soient portés à la fois dans les segments de données et d'ACK et que leur fassent écho les champs TSecr portés en retournant les segments ACK ou de données. PAWS soumet tous les segments entrants au même essai, et protège donc contre les segments ACK aussi bien que de données dupliqués. (Un autre algorithme non symétrique protègerait contre les vieux ACK dupliqués : l'envoyeur des données rejetterait les segments ACK entrants dont les valeurs de TSecr seraient inférieures à celles du TSecr sauvegardé à partir du dernier segment dont le champ ACK a avancé le bord gauche de la fenêtre d'envoi. Cet algorithme est réputé manquer d'économie de mécanisme et de symétrie.)

Les horodatages TSval envoyés sur les segments {SYN} et {SYN,ACK} sont utilisés pour initialiser PAWS. PAWS protège contre les vieux segments non SYN dupliqués, et contre les segments SYN dupliqués reçus lorsque il y a une connexion synchronisée. Les segments {SYN} et {SYN,ACK} dupliqués reçus lorsque il n'y a pas de connexion seront éliminés par la prise de contact à trois phases normale et par les vérifications de numéro de séquence de TCP.

Il est recommandé que les segments RST NE portent PAS d'horodatage, et que les segments RST soient acceptables sans considération de leur horodatage. Les vieux segments RST dupliqués devraient être très peu probables, et leur fonction de nettoyage devrait prendre le pas sur l'horodatage.

4.2.1 Algorithme PAWS de base

L'algorithme PAWS exige que le traitement suivant soit effectué sur tous les segments entrants pour une connexion synchronisée :

R1) Si il y a une option Horodatage dans le segment arrivant et si SEG.TSval < TS.Recent et si TS.Recent est valide (voir l'exposé qui suit) traiter alors le segment arrivant comme non acceptable:

Envoyer un accusé de réception en réponse, comme spécifié dans la RFC-793 page 69 et éliminer le segment.

Note : Il est nécessaire d'envoyer un segment ACK afin de conserver le mécanisme de TCP de détection et de récupération des connexions semi ouvertes. Par exemple, voir la Figure 10 de la RFC-793.

R2) Si le segment est en dehors de la fenêtre, le rejeter (traitement TCP normal).

R3) Si un segment arrivant satisfait à SEG.SEQ ≤ Last.ACK.sent (voir le paragraphe 3.4) enregistrer alors son horodatage dans TS.Recent.

R4) Si un segment arrivant est en séquence (c'est-à-dire, au bord gauche de la fenêtre) l'accepter alors normalement.

R5) Autrement, traiter le segment comme un segment TCP normal dans la fenêtre, hors séquence (par exemple, le mettre en file d'attente pour sa livraison ultérieure à l'usager).

Les étapes R2, R4, et R5 sont les étapes normales du traitement TCP spécifié par la RFC-793.

Il est important de noter que l'horodatage est vérifié seulement quand un segment arrive au receveur, sans considérer si il est en séquence ou si il doit être mis en file d'attente pour livraison ultérieure. Considérons l'exemple suivant.

Supposons que la séquence de segments A.1, B.1, C.1, ..., Z.1 a été envoyée, où la lettre indique le numéro de séquence et le chiffre représente l'horodatage. Supposons aussi que le segment B.1 a été perdu. L'horodatage dans TS.TStamp est 1 (de A.1), de sorte que C.1, ..., Z.1 sont considérés comme acceptables et sont mis en file d'attente. Lorsque B est retransmis comme segment B.2 (en utilisant le dernier horodatage) il remplit le trou et cause l'acquittement de tous les segments jusqu'à Z et leur passage à l'usager. L'horodatage des segments mis en file d'attente *n'est pas* inspecté à nouveau à ce moment, car ils ont déjà été acceptés. Lorsque B.2 est accepté, TS.Stamp est réglé à 2.

Cette règle permet des performance raisonnables en cas de pertes. Une pleine fenêtre de données est en transit à tout moment, et après une perte une pleine fenêtre moins un paquet va se trouver hors séquence pour être mise en file d'attente chez le receveur (par exemple, jusqu'à ~2**30 octets de données) ; l'option Horodatage ne doit pas résulter en l'élimination des données.

Dans certaines circonstances peu probables, l'algorithme des règles R1-R4 pourrait conduire à éliminer inutilement certains segments, comme le montre l'exemple suivant :

> Supposons encore que les segments A.1, B.1, C.1, ..., Z.1 ont été envoyés en séquence et que le segment B.1 a été perdu. De plus, supposons que la livraison de certains parmi C.1, ... Z.1 soit retardée jusque APRÈS que la retransmission de B.2 arrive au receveur. Ces segments retardés seront éliminés inutilement lorsque ils finissent par arriver, car leur horodatage est maintenant périmé.

Ce cas a très peu de chances de se produire. Si la retransmission a été déclanchée par une fin de temporisation, certains des segments C.1, ... Z.1 doivent avoir été retardés plus longtemps que le délai de RTO. On présume que c'est un événement improbable, ou il y aurait de nombreuses temporisations et retransmissions parasites. Si la retransmission de B a été déclanché par l'algorithme de "retransmission rapide", c'est-à-dire, par des ACK dupliqués, alors les segments en file d'attente qui ont causé ces ACK doivent avoir été déjà reçus.

Même si un segment était retardé plus longtemps que le RTO, le mécanisme de retransmission rapide [Jacobson90c] va causer la retransmission des paquets retardés en même temps que B.2, évitant un RTT supplémentaire et causant donc une très faible pénalité de performances.

On ne connaît pas de cas avec une probabilité d'occurrence significative dans lequel les horodatages causeraient une dégradation des performances par une inutile élimination de segments.

4.2.2 Horloge d'horodatage

Il est important de comprendre que l'algorithme PAWS n'exige pas de synchronisation d'horloge entre l'envoyeur et le receveur. L'horloge d'horodatage de l'envoyeur est utilisée pour horodater les segments, et l'envoyeur utilise l'écho de l'horodatage pour mesurer les RTT. Cependant, le receveur traite l'horodatage comme un simple numéro de série à accroissement monotone, sans aucune connexion nécessaire à son horloge. Du point de vue du receveur, l'horodatage agit comme une extension logique des bits de poids fort du numéro de séquence.

L'algorithme du receveur a bien quelques exigences quant à la fréquence de l'horloge d'horodatage :

(a) L'horloge d'horodatage ne doit pas être "trop lente".
Elle doit battre au moins uns fois tous les 2**31 octets envoyés. En fait, afin d'être utile pour que l'envoyeur mesure le délai d'aller-retour, l'horloge devrait battre au moins une fois par plein de données de la fenêtre, et même avec l'extension de fenêtre de la RFC-1072, 2**31 octets doivent faire au moins deux fenêtres. Pour rendre ceci plus quantitatif, toute horloge plus rapide que un tic/s va rejeter les vieux segments dupliqués pour des vitesses de liaison d'environ 8 Gbit/s. Une horloge d'horodatage à 1 ms va fonctionner à des vitesses de liaison jusqu'à 8 Tbit/s (8*10**12 bit/s !).

(b) L'horloge d'horodatage ne doit pas être "trop rapide".
Son temps de cycle doit être supérieur à MSL secondes. Comme l'horloge (l'horodatage) est de 32 bits et que le plus mauvais cas de MSL est de 255 secondes,la fréquence maximum acceptable d'horloge est d'un tic toutes les 59 ns.

Cependant, il est souhaitable d'établir une période de cycle bien plus longue, afin de traiter les horodatages périmés sur les connexions inactives (voir au paragraphe 4.2.3) et de relâcher les exigences de MSL pour empêcher le retour à zéro du numéro de séquence. Avec une horloge d'horodatage de 1 ms, l'horodatage de 32 bits va inverser son bit de signe en

24,8 jours. Donc, il va rejeter les vieux dupliqués sur la même connexion si le MSL est de 24,8 jours ou moins. Cela paraît être un chiffre très sûr ; un MSL de 24,8 jours ou plus long peut probablement être supposé par le système passerelle sans exiger une application précise du MSL par une valeur de TTL à la couche IP.

Sur la base de ces considérations, on choisira une fréquence d'horloge d'horodatage dans la gamme de 1 ms à 1 s par tic. Cette gamme correspond aussi aux exigences du mécanisme de RTTM, qui n'a pas besoin de beaucoup plus de résolution que la granularité du temporisateur de retransmission, par exemple, des dizaines ou centaines de millisecondes.

Le mécanisme PAWS fait aussi peser une forte exigence de monotonie sur l'horloge d'horodatage de l'envoyeur. La méthode de mise en œuvre de l'horloge d'horodatage pour satisfaire à cette exigence dépend du matériel et du logiciel du système.

* Certains hôtes ont une horloge matérielle dont la monotonie est garantie à travers les réinitialisations du matériel.
* Une interruption d'horloge peut être utilisée pour simplement incrémenter périodiquement de 1 un entier binaire.
* L'horloge d'horodatage peut être déduite d'une horloge système qui est soumise à des changements abrupts, par l'ajout d'une valeur de décalage variable. Ce décalage est initialisé à zéro. Lorsque une nouvelle valeur d'horloge d'horodatage est nécessaire, le décalage peut être ajusté en tant que de besoin pour rendre la nouvelle valeur égale ou supérieure à la valeur précédente (qui a été sauvegardée à cette fin).

4.2.3 Horodatages périmés

Si une connexion reste inactive pendant assez longtemps pour que l'horloge d'horodatage de l'autre TCP change son bit de signe, la valeur sauvegardée dans TS.Recent va alors devenir trop vieille; par suite, le mécanisme PAWS va causer le rejet de tous les segments suivants, gelant la connexion (jusqu'à ce que l'horloge d'horodatage change à nouveau son bit de signe).

Avec la gamme de fréquences d'horloge d'horodatage choisie (d'une seconde à 1 ms) le temps pour changer le bit de signe sera entre 24,8 jours et 24 800 jours. Une connexion TCP qui est inactive pendant plus de 24 jours et revient ensuite à la vie est excessivement inhabituelle. Cependant, il n'est pas souhaitable en principe de fixer de limitation aux durées de vie des connexions TCP.

On exige donc qu'une mise en œuvre de PAWS inclue un mécanisme pour "invalider" la valeur TS.Recent lorsque une connexion est inactive pendant plus de 24 jours. (Une autre solution au problème des horodatages périmés serait d'envoyer des segments Garder_en_vie à un rythme très faible, mais quand même plus souvent que le temps de retour à zéro pour les horodatages, par exemple, une fois par jour. Cela imposerait une redondance négligeable. Cependant, la spécification TCP n'a jamais comporté de Garder_en_vie, de sorte que la solution fondée sur l'invalidation a été choisie.)

Noter qu'un TCP ne connaît pas la fréquence, et donc, l'heure de retour à zéro, de l'autre TCP, de sorte qu'il doit supposer le pire. La validité du TS.Recent n'a besoin d'être vérifiée que si la vérification de l'horodatage PAWS de base échoue, c'est-à-dire, seulement si SEG.TSval < TS.Recent. Si TS.Recent est trouvé invalide, le segment est alors accepté, sans considération de l'échec de la vérification de l'horodatage, et la règle R3 met à jour TS.Recent avec le TSval provenant du nouveau segment.

Pour détecter depuis combien de temps la connexion est inactive, le TCP peut mettre à jour une horloge ou valeur d'horodatage associée à la connexion chaque fois que TS.Recent est mis à jour, par exemple. Les détails vont dépendre de la mise en œuvre.

4.2.4 Prédiction d'en-tête

La "prédiction d'en-tête" [Jacobson90a] est une technique de mise en œuvre de protocole de transport à hautes performances qui est de la plus grande importance pour les liaisons à grande vitesse. Cette technique optimise le code pour le cas le plus courant, recevant un segment correctement et dans l'ordre. En utilisant la prédiction d'en-tête, le receveur pose la question : "Ce segment est-il le suivant dans la séquence ?" Cette question peut trouver une réponse en moins d'instructions machine que la question :"Ce segment est-il dans la fenêtre ?"

Ajouter la prédiction d'en-tête à notre procédure d'horodatage amène la séquence recommandée suivante au traitement d'un segment TCP arrivant :

H1) Vérifier l'horodatage (comme dans l'étape R1 ci-dessus)
H2) Faire la prédiction d'en-tête : si le segment est le suivant dans la séquence et si il n'y a pas de condition particulière exigeant un traitement supplémentaire, accepter le segment, enregistrer son horodatage, et sauter H3.
H3) Traiter le segment normalement, comme spécifié dans la RFC-793. Cela inclut d'abandonner les segments qui sont en dehors de la fenêtre et éventuellement d'envoyer des accusés de réception, et mettre en file d'attente les segments dans la fenêtre qui sont hors séquence.

Une autre possibilité serait d'interchanger les étapes H1 et H2, c'est-à-dire, d'effectuer D'ABORD l'étape H2 de prédiction d'en-tête, et de n'effectuer H1 et H3 que lorsque la prédiction d'en-tête échoue. Cela pourrait améliorer les performances car, comme la vérification de l'horodatage de l'étape H1 a très peu de chances d'échouer, et qu'elle exige une arithmétique d'intervalle sur un champ fini, c'est une opération relativement coûteuse. Effectuer cette vérification sur chacun des segments est contraire à la philosophie de la prédiction d'en-tête. On pense que ce changement pourrait réduire le temps de CPU du traitement du protocole TCP d'environ 5-10 % sur des réseaux à haut débit.

Cependant, mettre H2 en premier créerait un risque : un segment provenant de 2**32 octets dans le passé pourrait arriver exactement au mauvais moment et être accepté par erreur par l'étape de prédiction d'en-tête. Le raisonnement suivant a été introduit dans la [RFC1185] pour montrer que la probabilité de cette défaillance est négligeable.

Si tous les segments ont une probabilité égale de se révéler de vieux dupliqués, la probabilité qu'un vieux dupliqué corresponde exactement au bord gauche de la fenêtre est la taille maximum de segment (MSS) divisée par la taille de l'espace de séquence. Ce ratio doit être inférieur à 2**-16, car le MSS doit être < 2**16 ; par exemple, il sera de (2**12)/(2**32) = 2**-20 pour une liaison FDDI. Cependant, plus un segment est vieux, moins il est probable qu'il soit conservé dans l'Internet, et selon tout modèle raisonnable de durée de vie de segment, la probabilité qu'un vieux dupliqué tombe exactement au bord gauche de la fenêtre doit être bien inférieure à 2**-16.

La somme de contrôle de 16 bits de TCP permet aussi une incertitude de base de un sur 2**16. Un mécanisme de protocole dont la fiabilité excède la fiabilité de la somme de contrôle TCP devrait être considéré comme "assez bon", c'est-à-dire qu'il ne va pas contribuer significativement au taux d'erreur global. On pense donc qu'on peut ignorer le problème d'un vieux dupliqué accepté en faisant la prédiction d'en-tête avant la vérifications de l'horodatage.

Cependant, cet argument probabiliste n'est pas universellement accepté, et le consensus présent est que le gain de performances ne justifie pas le risque dans le cas général. Il est donc recommandé que H2 suive H1.

4.3 Dupliqués provenant d'incarnations antérieures de la connexion

Le mécanisme PAWS protège contre les erreurs dues au retour à zéro du numéro de séquence sur les connexions à haut débit. Les segments provenant d'une incarnation précédente de la même connexion sont aussi une cause potentielle d'erreurs de vieux dupliqués. Dans les deux cas, le mécanisme TCP pour empêcher de telles erreurs dépend de l'application d'une durée de vie maximum de segment (MSL, *maximum segment lifetime*) par la couche Internet (IP) (voir l'exposé détaillé à l'Appendice de la RFC-1185). À la différence du cas du retour à zéro de l'espace de numéros de séquence, le MSL exigé pour empêcher les erreurs de vieux dupliqués provenant d'incarnations antérieures ne dépend pas du taux de transfert. Si la couche IP met en application le MSL recommandé de 2 minutes de TCP, et si les règles de TCP sont suivies, les connexions TCP seront protégées des incarnations antérieures, quel que soit le débit du réseau. Donc, le mécanisme PAWS n'est pas exigé pour ce cas.

On peut encore se demander si le mécanisme PAWS peut fournir une sécurité supplémentaire contre les vieux dupliqués provenant de connexions antérieures, nous permettant de relâcher la mise en application du MSL par la couche IP. L'Appendice B explore cette question, montrant que d'autres hypothèses et/ou mécanismes sont nécessaires, au delà de ceux de PAWS. Cela ne fait pas partie de la présente extension.

5. Conclusions et remerciements

Le présent mémoire a présenté un ensemble d'extensions à TCP pour assurer un fonctionnement efficace sur les chemins à large bande passante*retards et un fonctionnement fiable sur les chemins à très haut débit. Ces extensions sont conçues pour assurer un interfonctionnement compatible avec les TCP qui ne mettrent pas en œuvre ces extensions.

Ces mécanismes sont mis en œuvre avec de nouvelles options TCP pour des fenêtres adaptées et des horodatages. Les horodatages sont utilisés pour deux mécanismes distincts : RTTM (mesure du délai d'aller-retour) et PAWS (Protection contre le retour à zéro des numéros de séquence).

L'option Adaptation de fenêtre a été à l'origine suggérée par Mike St. Johns de USAF/DCA. La forme présente de l'option a été suggérée par Mike Karels de UC Berkeley en réponse à un schéma plus difficile défini par Van Jacobson. Lixia Zhang a aidé à formuler la description du mécanisme PAWS de la RFC-1185.

Finalement, beaucoup du présent travail trouve son origine dans les discussions qui ont eu lieu au sein de l'équipe End-to-End sur les limitations théoriques des protocoles de transport en général et de TCP en particulier. Plus récemment, les membres de l'équipe et d'autres de la liste de diffusion end2end-interest ont fait des contributions précieuses en révélant les

fautes des algorithmes et de la documentation. Les auteurs les remercient tous de ces contributions.

6. Références

[Garlick77] Garlick, L., R. Rom, et J. Postel, "Issues in Reliable Host-to-Host Protocols", Proc. Second Berkeley Workshop on Distributed Data Management et Computer Networks, mai 1977.

[Hamming77] Hamming, R., "Digital Filters", ISBN 0-13-212571-4, Prentice Hall, Englewood Cliffs, N.J., 1977.

[Jacobson88a] Jacobson, V., "Congestion Avoidance et Control", SIGCOMM '88, Stanford, CA., août 1988.

[Jacobson90a] Jacobson, V., "4BSD Header Prediction", ACM Computer Communication Review, Avril 1990.

[Jacobson90c] Jacobson, V., "Modified TCP congestion avoidance algorithm", Message to end2end-interest mailing list, avril 1990.

[Jain86] Jain, R., "Divergence of Timeout Algorithms for Packet Retransmissions", Proc. Fifth Phoenix Conf. on Comp. et Comm., Scottsdale, Arizona, mars 1986.

[Karn87] Karn, P. et C. Partridge, "Estimating Round-Trip Times in Reliable Transport Protocols", Proc. SIGCOMM '87, Stowe, VT, août 1987.

[NBS85] Colella, R., Aronoff, R., et K. Mills, "Performance Improvements for ISO Transport", Ninth Data Comm Symposium, published in ACM SIGCOMM Comp Comm Review, vol. 15, no. 5, septembre 1985.

[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.

[RFC0908] D. Velten, R. Hinden et J. Sax, "Protocole fiable de données", juillet 1984.

[RFC0998] D. Clark, M. Lambert et L. Zhang, "NETBLT : protocole de transfert de données en vrac", mars 1987. *(Exp.)*

[RFC1045] D. Cheriton, "VMTP : Spécification du protocole de transaction de message versatile", février 1988. *(Exp.)*

[RFC1072] V. Jacobson et R. Braden, "Extensions TCP pour les chemins à fort délai", octobre 1988. *(Obsolète, voir 1323 et 2018)*

[RFC1110] A. McKenzie, "Le problème de l'option grande fenêtre de TCP", août 1989.

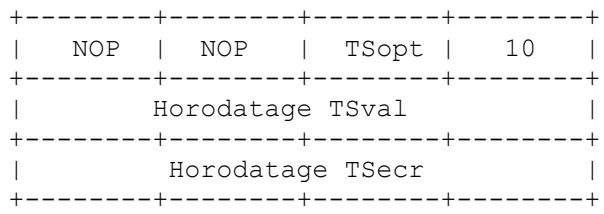
[[RFC1185] V. Jacobson, R. Braden et L. Zhang, "Extension TCP pour chemins à grande vitesse", octobre 1990.

[Watson81] Watson, R., "Timer-based Mechanisms in Reliable Transport Protocol Connection Management", Computer Networks, Vol. 5, 1981.

[Zhang86] Zhang, L., "Why TCP Timers Don't Work Well", Proc. SIGCOMM '86, Stowe, Vt., août 1986.

Appendice A Suggestions de mise en œuvre

Les présentations suivantes sont recommandées pour l'envoi des options sur des segments non SYN, pour réaliser l'alignement maximum réalisable de machines à 32 bits et à 64 bits.



Appendice B Dupliqués provenant d'incarnations antérieures de la connexion

Deux cas sont à considérer : (1) un système qui a une défaillance (et perd l'état de connexion) et redémarre, et (2) la même connexion est fermée et rouverte sans perte de l'état de l'hôte. Ils seront décrits dans les deux paragraphes suivants.

B.1 Panne système avec perte d'état

Le temps de repos de TCP d'un MSL au redémarrage du système traite la perte de l'état de connexion dans la panne/redémarrage d'un système. Pour une explication, voir par exemple "Quand rester tranquille" dans la spécification du protocole TCP [RFC0793]. Le MSL qui est exigé ici ne dépend pas de la vitesse du transfert. Le MSL TCP actuel de 2 minutes semble acceptable comme compromis de fonctionnement, car de nombreux systèmes hôtes prennent ce temps pour réamorcer après une panne.

Cependant, l'option d'horodatage peut être utilisée pour faciliter l'application des exigences de MSL (ou pour fournir une sécurité supplémentaire contre la corruption des données). Si les horodatages sont utilisés et si le fonctionnement monotone de l'horloge d'horodatage peut être garanti lors d'une panne/redémarrage de système, c'est-à-dire, si on peut garantir que la première valeur de l'horloge d'horodatage de l'envoyeur après une panne/redémarrage sera plus grande que la dernière valeur avant le redémarrage, un temps de repos sera inutile.

Se passer totalement du temps de repos exigerait que l'horloge de l'hôte soit synchronisée avec une source horaire qui soit stable sur la période de panne/redémarrage, avec une précision d'un tic d'horloge d'horodatage ou mieux. On peut s'appuyer sur cette exigence stricte pour tirer parti d'une synchronisation d'horloge approximative. Supposons que l'horloge soit toujours resynchronisée à N tics d'horloge d'horodatage et que l'amorçage (étendu d'un temps de repos, si nécessaire) prenne plus de N tics. Cela va garantir la monotonie des horodatages, qui peuvent alors être utilisés pour rejeter les vieux dupliqués même sans l'application d'un MSL.

B.2 Fermeture et réouverture d'une connexion

Lorsque une connexion TCP est fermée, un délai de 2*MSL dans l'état TIME-WAIT lie la paire de prises pour 4 minutes (voir le paragraphe 3.5 de la [RFC0793]. Les applications construites sur TCP qui closent une connexion et en ouvrent une nouvelle (par exemple, une connexion de transfert de données par FTP qui utilise le mode Flux) doit choisir à chaque fois une nouvelle pair de prises. Le délai TIME-WAIT sert deux objets différents :

(a) Mettre en œuvre la prise de contact de fermeture fiable bidirectionnelle de TCP.

Le moment approprié pour différer l'étape de clôture finale n'est pas en relation directe avec le MSL ; il dépend plutôt du RTO pour les segments FIN et donc du RTT du chemin. (On pourrait objecter que le côté qui envoie un FIN sait le degré de fiabilité dont il a besoin, et donc, il devrait être capable de déterminer la longueur du délai TIME-WAIT pour le receveur du FIN. Cela pourrait être réalisé avec une option TCP appropriée dans les segments FIN.)

Bien qu'il n'y ait pas de limite supérieure formelle au RTT, la pratique courante de l'ingénierie de réseau rend un RTT supérieur à une minute très improbable. Donc, le délai de 4 minutes de l'état TIME-WAIT fonctionne de façon satisfaisante pour fournir une fermeture TCP fiable en bidirectionnel. Noter une fois encore que ceci est indépendant de la mise en application du MSL et de la vitesse du réseau.

L'état TIME-WAIT pourrait causer un problème indirect de performances si une application avait besoin de clore de façon répétée une connexion et d'en rouvrir une autre à une très haute fréquence, car le nombre d'accès TCP disponibles sur un hôte est inférieur à 2**16. Cependant, les vitesses élevées de réseau ne sont pas le contributeur majeur à ce problème ; le RTT est le facteur qui limite la vitesse à laquelle les connexions peuvent être ouvertes et fermées. Donc, ce problème ne sera pas pire à des vitesses élevées de transfert.

(b) Permettre l'expiration des vieux segments dupliqués.

Pour remplacer cette fonction d'état TIME-WAIT, un mécanisme qui aurait à fonctionner à travers les connexions. PAWS est défini strictement au sein d'une seule connexion ; le dernier horodatage est TS.Recent qui est conservé dans le bloc de contrôle de la connexion, et est éliminé lorsque une connexion est close.

Un mécanisme supplémentaire pourrait être ajouté à TCP, une antémémoire par hôte du dernier horodatage reçu de toute connexion. Cette valeur pourrait alors être utilisée dans le mécanisme PAWS pour rejeter les vieux segments dupliqués provenant d'incarnations antérieures de la connexion, si l'horloge d'horodatage peut garantir d'avoir au moins eu un tic depuis l'ouverture de la vielle connexion. Cela exigerait que le délai TIME-WAIT plus le RTT soient ensemble au moins à un tic de l'horloge d'horodatage de l'envoyeur. Une telle extension ne fait pas partie des propositions de la présente RFC.

Noter qu'il y a une variante du mécanisme proposé par Garlick, Rom, et Postel [Garlick77], qui exige que chaque hôte conserve un enregistrement de connexion qui contienne le plus fort numéro de séquence de chaque connexion. En utilisant des horodatages à la place, il est seulement nécessaire de conserver une quantité par hôte distant, sans considération du nombre de connexions simultanées avec cet hôte.

Appendice C Changements depuis les RFC1072, RFC1185

Les extensions de protocole définies dans le présent document diffèrent de plusieurs façons importantes de celles définies dans les RFC-1072 et RFC-1185.

(a) SACK a été reporté dans un autre mémoire.

(b) Les règles détaillées pour l'envoi des réponses d'horodatage (voir le paragraphe 3.4) diffèrent d'une façon importante. Les règles antérieures pourraient résulter en une sous-estimation du RTT dans certains cas (paquets abandonnés ou déclassés).

(c) La même valeur TS.Recent est maintenant partagée par deux mécanismes distincts, RTTM et PAWS. Cette simplification est devenue possible à cause du changement précédent (b).

(d) Une ambiguïté de la RFC-1185 a été résolue en faveur de l'inclusion d'horodatage dans les ACK aussi bien que dans les segments de données. Cela prend en charge la symétrie du protocole TCP sous-jacent.

(e) Les options Écho et Réponse d'écho de la RFC-1072 ont été combinées en une seule option Horodatage, pour refléter la symétrie et pour simplifier le traitement.

(f) Le problème des horodatages périmés sur les connexions longtemps inactives, exposé au paragraphe 4.2.2, a été réalisé et résolu.

(g) La RFC-1185 recommandait que la prédiction d'en-tête prenne le pas sur la vérification d'horodatage. Sur la base d'un certain scepticisme à l'égard des arguments probabilistes donnés au paragraphe 4.2.4, il a été décidé de recommander que la vérification de l'horodatage soit effectuée en premier.

(h) La spécification a été modifiée de telle sorte que les options étendues ne soient envoyées sur les segments <SYN,ACK> que lorsque elles sont reçues dans les segments <SYN> correspondants. Cela assure les conditions les plus prudentes pour l'interopération avec les mises en œuvre qui n'ont pas les extensions.

En plus de ces changements de substance, la présente RFC tente de spécifier sans ambiguïté les algorithmes en présentant les modifications aux règles de traitement d'événement de la RFC-793 ; voir l'Appendice E.

Appendice D Résumé des notations

La notation suivante a été utilisée dans le présent document.

Options
Wsopt : Option TCP Adaptation de fenêtre *(Window Scale)*
TSopt : Option TCP Horodatages *(Timestamps)*

Champs d'option
shift.cnt : Octet d'adaptation de fenêtre dans WSopt.
Tsval : Champ Valeur d'horodatage de 32 bits dans TSopt.
Tsecr : Champ Réponse d'horodatage de 32 bits dans TSopt.

Champs d'option dans le segment en cours

SEG.Tsval : Champ TSval provenant de TSopt dans le segment en cours.
SEG.Tsecr : Champ TSecr provenant de TSopt dans le segment en cours.
SEG.Wsopt : Valeur de 8 bits dans WSopt

Valeurs d'horloge
my.Tsclock : Source locale de valeurs d'horodatage de 32 bits
my.TSclock.rate : Période de my.TSclock (de 1 ms à 1 s).

Variables par état de connexion
TS.Recent : Dernier horodatage reçu
Last.ACK.sent : Dernier champ ACK envoyé
Snd.TS.OK : Fanion d'un bit
Snd.WS.OK : Fanion d'un bit
Rcv.Wind.Scale : Puissance d'adaptation de la fenêtre de réception
Snd.Wind.Scale : Puissance d'adaptation de la fenêtre d'envoi

Appendice E Traitement des événements

Traitement d'événement

Invoquer OPEN

...
Un numéro de séquence initial envoyé (ISS, *initial send sequence*) est choisi. Envoyer un segment SYN de la forme :

<SEQ=ISS><CTL=SYN><TSval=my.TSclock><WSopt=Rcv.Wind.Scale>

...

Invoquer SEND

État CLOSED (c'est-à-dire, TCB n'existe pas)

...

État LISTEN

Si la prise distante est spécifiée, changer alors la connexion de passive à active, choisir un ISS. Envoyer un segment SYN contenant les options : <TSval=my.TSclock> et <WSopt=Rcv.Wind.Scale>. Régler SND.UNA à ISS, SND.NXT à ISS+1. Entrer dans l'état SYN-SENT. ...

État SYN-SENT
État SYN-RECEIVED
...
État ESTABLISHED
État CLOSE-WAIT
Segmenter la mémoire tampon et l'envoyer avec un accusé de réception porté (la valeur de l'accusé de réception = RCV.NXT). ...

Si le fanion urgent est établi ...

Si le fanion Snd.TS.OK est établi, inclure alors l'option TCP Horodatages <TSval=my.TSclock,TSecr=TS.Recent> dans chaque segment de données.

Adapter la fenêtre de réception pour la transmission dans l'en-tête de segment :

SEG.WND = (SND.WND >> Rcv.Wind.Scale).

Le SEGMENT ARRIVE

...

Si l'état est LISTEN alors

premièrement, vérification d'un RST

...

deuxièmement, vérification d'un ACK

...

troisièmement, vérification d'un SYN

si le bit SYN est établi, vérifier la sécurité. Si le ...

...

Si le SEG.PRC est inférieur au TCB.PRC, continuer.

Vérifier qu'il y a une option Adaptation de fenêtre (WSopt) ; si il en est trouvé une, sauvegarder SEG.WSopt dans Snd.Wind.Scale et établir le fanion Snd.WS.OK. Autrement, régler Snd.Wind.Scale et Rcv.Wind.Scale à zéro et ôter le fanion Snd.WS.OK.

Chercher une option TSopt ; si il en est trouvé une, sauvegarder SEG.TSval dans la variable TS.Recent et établir le bit Snd.TS.OK.

Régler RCV.NXT à SEG.SEQ+1, IRS est réglé à SEG.SEQ et tout autre contrôle ou texte devrait être mis en file d'attente pour un traitement ultérieur. L'ISS devrait être choisi et un segment SYN devrait être envoyé sous la forme :

<SEQ=ISS><ACK=RCV.NXT><CTL=SYN,ACK>

Si le bit Snd.WS.OK est établi, inclure une option WSopt <WSopt=Rcv.Wind.Scale> dans ce segment. Si le bit Snd.TS.OK est établi, inclure un TSopt <TSval=my.TSclock,TSecr=TS.Recent> dans ce segment. Last.ACK.sent est réglé à RCV.NXT.

SND.NXT est réglé à ISS+1 et SND.UNA à ISS. L'état de connexion devrait être changé en SYN-RECEIVED. Noter que tout autre contrôle ou donnée entrant (combiné au SYN) sera traité dans l'état SYN-RECEIVED, mais le traitement du SYN et de l'ACK ne devrait pas être répété. Si le "listen" n'a pas été pleinement spécifié (c'est-à-dire, si la prise distante n'a pas été pleinement spécifiée) les champs non spécifiés devraient alors être remplis.

quatrièmement, autre texte ou contrôle

...

Si l'état est SYN-SENT alors

premièrement, vérifier le bit ACK

...

quatrièmement ,vérifier le bit SYN

...

Si le bit SYN est établi et si la sécurité/compartimentage et la préséance sont acceptables, alors RCV.NXT est réglé à SEG.SEQ+1, IRS est réglé à SEG.SEQ, et tout accusé de réception sur la file de retransmission qui est par là acquitté devrait être retiré.

Vérifier qu'il y a une option Adaptation de fenêtre (WSopt) ; si il en est trouvé une, sauvegarder SEG.WSopt dans Snd.Wind.Scale; autrement, régler Snd.Wind.Scale et Rcv.Wind.Scale à zéro.

Vérifier qu'il y a une option TSopt ; si il en est trouvé une, sauvegarder SEG.TSval dans la variable TS.Recent et établir le bit Snd.TS.Ok dans le bloc de contrôle de la connexion. Si le bit ACK est établi, utiliser my.TSclock - SEG.TSecr comme estimation initiale de RTT.

Si SND.UNA > ISS (notre SYN a été acquitté) changer l'état de connexion en ESTABLISHED, former un segment ACK :

<SEQ=SND.NXT><ACK=RCV.NXT><CTL=ACK>

et l'envoyer. Si le bit Snd.Echo.OK est établi, inclure une option TSopt <TSval=my.TSclock,TSecr=TS.Recent> dans ce segment ACK. Last.ACK.sent est réglé à RCV.NXT.

Les données ou contrôles qui ont été mis en file d'attente pour transmission peuvent être inclus. Si il y a d'autres contrôles ou texte dans le segment, continuer alors le traitement à l'étape six ci-dessous où le bit URG est vérifié, autrement, revenir.

Autrement, entrer SYN-RECEIVED, former un segment SYN,ACK :

<SEQ=ISS><ACK=RCV.NXT><CTL=SYN,ACK>

et l'envoyer. Si le bit Snd.Echo.OK est établi, inclure une option TSopt <TSval=my.TSclock,TSecr=TS.Recent> dans ce segment. Si le bit Snd.WS.OK est établi, inclure une option WSopt <WSopt=Rcv.Wind.Scale> dans ce segment. Last.ACK.sent est réglé à RCV.NXT.

Si il y a d'autres contrôles ou du texte dans le segment, les mettre en file d'attente pour traitement après avoir atteint l'état ESTABLISHED, revenir.

cinquièmement, si ni le bit SYN ni le bit RST ne sont établis, éliminer alors le segment et revenir.

Autrement,

Premièrement, vérifier le numéro de séquence

État SYN-RECEIVED
État ESTABLISHED
État FIN-WAIT-1
État FIN-WAIT-2
État CLOSE-WAIT
État CLOSING
État LAST-ACK
État TIME-WAIT

Les segments sont traités en séquence. Les vérifications initiales sur l'arrivée sont utilisées pour éliminer les vieux dupliqués, mais un autre traitement est effectué dans l'ordre SEG.SEQ. Si le contenu d'un segment est à cheval sur la frontière entre ancien et nouveau, seules les parties nouvelles devraient être traitées.

Réadapter le champ de fenêtre de réception :

TrueWindow = SEG.WND << Snd.Wind.Scale,

et utiliser "TrueWindow" à la place de SEG.WND dans les étapes suivantes.

Vérifier si le segment contient une option Horodatages et si le bit Snd.TS.OK est établi. S'il en est ainsi :

Si SEG.TSval < TS.Recent, alors vérifier que la connexion a été inactive moins de 24 jours ; si les deux sont vrais, alors le segment n'est pas acceptable ; suivre les étapes ci-dessous pour un segment inacceptable.

Si SEG.SEQ est égal à Last.ACK.sent, sauvegarder SEG.ECopt dans la variable TS.Recent.

Il y a quatre cas d'essais d'acceptabilité pour un segment entrant :

...

Si un segment entrant n'est pas acceptable, un accusé de réception devrait être envoyé en réponse (sauf si le bit RST est établi ; si il l'est, éliminer le segment et revenir) :

<SEQ=SND.NXT><ACK=RCV.NXT><CTL=ACK>

Last.ACK.sent est réglé au SEG.ACK de l'accusé de réception. Si le bit Snd.Echo.OK est établi, inclure l'option Horodatages <TSval=my.TSclock,TSecr=TS.Recent> dans ce segment ACK. Régler Last.ACK.sent à SEG.ACK et envoyer le segment ACK. Après l'envoi de l'accusé de réception, éliminer le segment inacceptable et revenir.

...

cinquièmement, vérifier le champ ACK.

si le bit ACK est à zéro, éliminer le segment et revenir.

si le bit ACK est établi

...

État ESTABLISHED

Si SND.UNA < SEG.ACK ≤ SND.NXT, régler alors SND.UNA <- SEG.ACK. Calculer aussi une nouvelle estimation du délai d'aller-retour. Si le bit Snd.TS.OK est établi, utiliser my.TSclock - SEG.Tsecr ; autrement, utiliser le temps écoulé depuis l'envoi du premier segment de la file d'attente de retransmission. Tous les segments de la file d'attente de retransmission sont donc par là entièrement acquittés...

...

Septièmement traiter le texte du segment.

État ESTABLISHED
État FIN-WAIT-1
État FIN-WAIT-2

...

Envoyer un accusé de réception de la forme :

<SEQ=SND.NXT><ACK=RCV.NXT><CTL=ACK>

Si le bit Snd.TS.OK est établi, inclure l'option Horodatages <TSval=my.TSclock,TSecr=TS.Recent> dans ce segment ACK. Régler Last.ACK.sent au SEG.ACK de l'accusé de réception, et l'envoyer. Cet accusé de réception devrait être porté sur un segment destiné à être transmis si possible sans subir de délais indus.

...

Considérations pour la sécurité

Les questions de sécurité ne sont pas abordées dans le présent mémoire.

Adresses des auteurs

Van Jacobson
University of California
Lawrence Berkeley Laboratory
Mail Stop 46A
Berkeley, CA 94720
téléphone : (415) 486-6411
mél : van@CSAM.LBL.GOV

Bob Braden
University of Southern California
Information Sciences Institute
4676 Admiralty Way
Marina del Rey, CA 90292
téléphone : (310) 822-1511
mél : Braden@ISI.EDU

Dave Borman
Cray Research
655-E Lone Oak Drive
Eagan, MN 55121
téléphone : (612) 683-5571
mél : dab@cray.com