Groupe de travail Réseau
Request for Comments : 2018
Catégorie : En cours de normalisation
Traduction Claude Brière de L'Isle

M. Mathis, J. Mahdavi, PSC
S. Floyd, LBNL
A. Romanow, Sun Microsystems
octobre 1996

Options d'accusé de réception sélectif sur TCP

Résumé
TCP peut accomplir des performances médiocres lorsque plusieurs paquets sont perdus à partir d'une fenêtre de données. Avec les informations limitées qui sont disponibles à partir des accusés de réception cumulés, un envoyeur TCP peut seulement connaître une seule perte de paquet par délai d'aller-retour. Un envoyeur agressif pourrait choisir de retransmettre les paquets plus tôt, mais de tels segments retransmis peuvent avoir déjà été bien reçus.

Un mécanisme d'accusé de réception sélectif (SACK, *Selective Acknowledgment*), combiné avec une politique sélective de répétition de retransmission, peut aider à surmonter ces limitations. Le receveur TCP renvoie des paquets SACK à l'envoyeur pour informer l'envoyeur des données qui ont été reçues. L'envoyeur peut alors retransmettre uniquement les segments de données manquants.

Le présent mémoire propose une mise en œuvre de SACK et expose ses performances et les problèmes qui s'y rapportent.

(La présente traduction incorpore l'errata 1610 relatif au paragraphe 5.1.)

Remerciements
Beaucoup du texte de ce document est tiré directement de la RFC1072 "Extensions à TCP pour les chemins à long délai" de Bob Braden et Van Jacobson. Les auteurs tiennent à remercier Kevin Fall (LBNL), Christian Huitema (INRIA), Van Jacobson (LBNL), Greg Miller (MITRE), Greg Minshall (Ipsilon), Lixia Zhang (XEROX PARC et UCLA), Dave Borman (BSDI), Allison Mankin (ISI) et d'autres pour leur relecture et leurs commentaires constructifs.

1. Introduction

Des pertes de paquet multiples à partir d'une fenêtre de données peuvent avoir un effet catastrophique sur le débit TCP. TCP [RFC1323] utilise un schéma d'accusé de réception cumulatif dans lequel les segments reçus qui ne sont pas sur le bord gauche de la fenêtre de réception ne sont pas acquittés. Cela force l'envoyeur soit à attendre pendant un délai d'aller-retour pour découvrir ce qu'il en est de chaque paquet perdu, soit de retransmettre sans nécessité des segments qui ont été correctement reçus [Fall95]. Avec le schéma d'accusé de réception cumulatif, plusieurs segments abandonnés causent généralement la perte par TCP de son horloge fondée sur les ACK, ce qui réduit le débit global.

L'accusé de réception sélectif (SACK, *Selective Acknowledgment*) est une stratégie qui corrige ce comportement en présence de plusieurs abandons de segments. Avec l'accusé de réception sélectif, le receveur des données peut informer l'envoyer de tous les segments qui sont bien arrivés, de sorte que l'envoyeur n'a besoin de retransmettre que les segments qui ont réellement été perdus.

Plusieurs protocoles de transport, parmi lesquels NETBLT [RFC0998], XTP [Strayer92], RDP [RFC0908], NADIR [Huitema81], et VMTP [RFC1045] ont utilisé l'accusé de réception sélectif. Il y a une certaine évidence empirique en faveur de l'accusé de réception sélectif – la simple expérience de RDP a montré que désactiver la facilité d'accusé de réception sélectif augmente considérablement le nombre de segments retransmis sur un chemin Internet à pertes et fort délai [Partridge87]. Une étude récente de simulation de Kevin Fall et Sally Floyd [Fall95], démontre la supériorité de TCP avec SACK sur les mises en œuvre TCP non SACK, Tahoe et Reno.

La [RFC1072] décrit une mise en œuvre possible des options SACK pour TCP. Malheureusement, elle n'a jamais été

déployée dans l'Internet, car il y avait un désaccord sur la façon dont les options SACK devraient être utilisées en conjonction avec l'option TCP de fenêtre glissante (décrite initialement dans la RFC1072 et révisée dans la [RFC1323]).

Nous proposons de légères modifications aux options SACK telles que proposées dans la RFC1072. Précisément, l'envoi d'un accusé de réception sélectif pour les données les plus récemment reçues réduit le besoin des longues options SACK [Keshav94, Mathis95]. De plus, l'option SACK porte maintenant les numéros de séquence complets de 32 bits. Ces deux modifications représentent les seuls changements à la proposition de la RFC1072. Elles rendent SACK plus facile à mettre en œuvre et apaisent les inquiétudes sur sa robustesse.

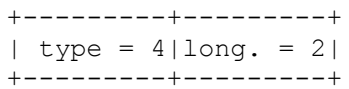
L'extension d'accusé de réception sélectif utilise deux options TCP. La première est une option d'activation, "SACK-permis", qui peut être envoyée dans un segment SYN pour indiquer que l'option SACK peut être utilisée une fois que la connexion est établie. L'autre est l'option SACK elle-même, qui peut être envoyée sur une connexion établie une fois que la permission a été donnée par SACK-permis.

L'option SACK est à inclure dans un segment envoyé d'un TCP qui reçoit des données au TCP qui envoie ces données ; nous nous réfèrerons respectivement à ces TCP comme "receveur des données" et "envoyeur des données". Nous considérons un flux de données simplex particulier ; toutes données s'écoulant dans la direction inverse sur la même connexion peuvent être traitées indépendamment.

2. Option SACK-permis

Cette option de deux octets peut être envoyée dans un SYN par un TCP qui a été étendu pour recevoir (et vraisemblablement traiter) l'option SACK une fois la connexion ouverte. Elle NE DOIT PAS être envoyée sur des segments non SYN.

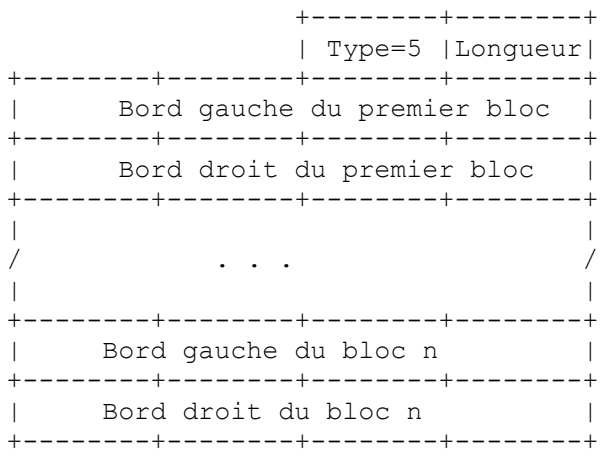
Option TCP SACK-permis :
Type : 4



3. Format de l'option SACK

L'option SACK est à utiliser pour convoyer des informations d'accusé de réception étendues du receveur à l'envoyeur sur une connexion TCP établie.

Option TCP SACK :
Type : 5
Longueur : Variable



L'option SACK est à envoyer par le receveur des données pour informer l'envoyeur des données que des blocs de données

non contigus ont été reçus et mis en file d'attente. Le receveur des données attend la réception des données (peut-être au moyen de retransmissions) pour combler les vides de l'espace des numéros de séquence entre les blocs reçus. Lorsque les segments manquants sont reçus, le receveur des données accuse normalement réception des données en avançant le bord gauche de la fenêtre dans le champ Numéro d'accusé de réception dans l'en-tête TCP. L'option SACK ne change pas la signification du champ Numéro d'accusé de réception.

Cette option contient une liste de certains des blocs de l'espace de séquence contigu occupé par les données reçues et mises en file d'attente dans la fenêtre.

Chaque bloc contigu de données mises en file d'attente chez le receveur des données est défini dans l'option SACK par deux entiers de 32 bits non signés dans l'ordre des octets du réseau :

* Bord gauche du bloc
  C'est le premier numéro de séquence de ce bloc.

* Bord droit du bloc
  C'est le numéro de séquence qui suit immédiatement le dernier numéro de séquence de ce bloc.

Chaque bloc représente les octets de données reçus qui sont contigus et isolés ; c'est-à-dire, les octets juste en dessous du bloc, (Bord gauche du bloc - 1), et juste au-dessus du bloc, (Bord droit du bloc) n'ont pas été reçus.

Une option SACK qui spécifie *n* blocs aura une longueur de 8**n*+2 octets, de sorte que les 40 octets disponibles pour les options TCP peuvent spécifier un maximum de 4 blocs. Il est prévu que SACK soit souvent utilisé en conjonction avec l'option Horodatage utilisée pour RTTM [RFC1323], qui prend 10 octets supplémentaires (plus deux octets de bourrage) ; donc un maximum de 3 blocs SACK seront admis dans ce cas.

L'option SACK est consultative, en ce que, alors qu'elle notifie à l'envoyeur des données que le receveur des données a reçu les segments indiqués, il est permis au receveur des données d'éliminer ensuite les données qui ont été rapportées dans une option SACK. Dans la Section 8 figure une discussion sur les conséquences du caractère consultatif de SACK, en particulier que le receveur des données peut renoncer, ou abandonner des données pour lesquelles SACK avait été fourni.

4. Génération des options SACK : comportement du receveur des données

Si le receveur des données a reçu une option SACK-permis sur le SYN pour cette connexion, le receveur des données PEUT choisir de générer les options SACK comme décrit ci-dessous. Si le receveur des données génère les options SACK, en toutes circonstances, il DEVRAIT les générer dans toutes les circonstances permises. Si le receveur des données n'a pas reçu une option SACK-permis pour une connexion donnée, il NE DOIT PAS envoyer les options SACK sur cette connexion.

Si elles sont envoyées, les options SACK DEVRAIENT être incluses dans tous les ACK qui n'accusent pas réception du plus haut numéro de séquence dans la file d'attente du receveur des données. Dans cette situation, le réseau a perdu ou dérangé les données, de sorte que le receveur détient des données non contiguës dans sa file d'attente. Le paragraphe 4.2.2.21 de la RFC 1122 discute les raisons pour que le receveur envoie des ACK en réponse aux segments supplémentaires reçus dans cet état. Le receveur DEVRAIT envoyer un ACK pour chaque segment valide qui arrive et contient de nouvelles données, et chacun de ces ACK "dupliqué" DEVRAIT porter une option SACK.

Si le receveur des données choisit d'envoyer une option SACK, les règles suivantes s'appliquent :

* Le premier bloc SACK (c'est-à-dire, celui qui suit immédiatement les champs Type et Longueur dans l'option) DOIT spécifier le bloc de données contigu qui contient le segment qui a déclanché ce ACK, à moins que le segment n'avance le champ Numéro d'accusé de réception dans l'en-tête. Ceci assure que le ACK avec l'option SACK reflète le changement le plus récent dans la file d'attente de la mémoire tampon du receveur des données.

* Le receveur des données DEVRAIT inclure autant de blocs SACK distincts que possible dans l'option SACK. Noter que l'espace maximum d'option disponible peut n'être pas suffisant pour rapporter tous les blocs présents dans la file d'attente du receveur.

* L'option SACK DEVRAIT être remplie en répétant les bloc SACK les plus récemment rapportés (sur la base des premiers blocs SACK dans les précédentes options SACK) qui ne sont pas des sous-ensembles d'un bloc SACK déjà inclus dans l'option SACK en cours de construction. Cela assure qu'en fonctionnement normal, tout segment qui reste au titre d'un bloc non contigu de données détenu par le receveur des données est rapporté dans au moins trois options

SACK successives, même pour des mises en œuvre de TCP à fenêtre large [RFC1323]). Après le premier bloc SACK, les blocs SACK suivants dans l'option SACK peuvent être énumérés dans n'importe quel ordre.

Il est très important que l'option SACK rapporte toujours le bloc qui contient le segment le plus récemment reçu, parce que cela donne à l'envoyeur les informations les plus à jour sur l'état du réseau et sur la file d'attente du receveur des données.

5. Interprétation de l'option SACK et stratégie de retransmission : comportement de l'envoyeur des données

Lorsque il reçoit un ACK qui contient une option SACK, l'envoyeur des données DEVRAIT enregistrer l'accusé de réception sélectif pour les références futures. L'envoyeur des données est supposé avoir une file d'attente de retransmission qui contient les segments qui ont été transmis mais n'ont pas encore été acquittés, dans l'ordre des numéros de séquence. Si l'envoyeur des données effectue une remise en paquets avant la retransmission, les limites de bloc dans une option SACK qu'il reçoit peuvent ne pas tomber sur les limites des segments qui sont dans la file d'attente de retransmission ; cependant, cela ne pose pas de difficultés sérieuses à l'envoyeur.

Une mise en œuvre possible du comportement de l'envoyeur est la suivante. Supposons que, pour chaque segment dans la file d'attente de retransmission, il y ait un (nouveau) bit fanion "SACK", à utiliser pour indiquer que ce segment particulier a été rapporté dans une option SACK.

Lorsque arrive un segment d'accusé de réception contenant une option SACK, l'envoyeur des données va établir les bits SACK pour les segments qui ont reçu l'accusé de réception sélectif. Plus précisément, pour chaque bloc dans l'option SACK, l'envoyeur des données va établir les fanions SACK pour tous les segments qui sont entièrement contenus dans ce bloc dans la file d'attente de retransmission. Cela exige des comparaisons directes de numéros de séquence.

Après l'établissement du bit SACK (par suite du traitement d'une option SACK reçue) l'envoyeur des données va sauter ce segment lors de toute retransmission ultérieure. Tout segment qui a le bit SACK ôté *(mis à 0)* et est inférieur au plus fort segment qui a le bit SACK est disponible pour la retransmission.

Après une temporisation de retransmission, l'envoyeur des données DEVRAIT mettre à zéro tous les bits SACK, car la temporisation pourrait indiquer que le receveur des données a renoncé. L'envoyeur des données DOIT retransmettre le segment au bord gauche de la fenêtre après une fin de temporisation de retransmission, que les bit SACK soit établi ou non pour le segment. Un segment ne sera pas sorti de la file d'attente et son espace de mémoire tampon libéré jusqu'à ce que le bord gauche de la fenêtre soit avancé jusqu'à lui.

5.1 Problèmes du contrôle de l'encombrement

Le présent document n'essaye pas de spécifier en détails les algorithmes de contrôle de l'encombrement pour les mises en œuvre de TCP avec SACK. Cependant, les algorithmes de contrôle de l'encombrement présents dans les mises en œuvre TCP standard DOIVENT être préservés [Stevens94]. En particulier, pour préserver la robustesse en présence de paquets réarrangés par le réseau, la récupération n'est pas déclanchée par un seul ACK rapportant des paquets déclassés chez le receveur. De plus, durant la récupération, l'envoyeur des données limite le nombre des segments envoyés en réponse à chaque ACK. Les mises en œuvre existantes limitent l'envoyeur des données à l'envoi d'un seul segment durant la récupération rapide de style Reno, ou à deux segments durant le démarrage lent de la [RFC1072]. Les autres aspects du contrôle d'encombrement, tels que la réduction de la fenêtre d'encombrement en réponse à l'encombrement, doivent aussi être préservés.

L'utilisation de temporisations comme mécanismes de sauvegarde pour détecter les paquets abandonnés n'est pas changée par l'option SACK. Comme le receveur des données peut éliminer des données qui ont reçu un SACK, lorsque survient une fin de temporisation de retransmission, l'envoyeur des données DEVRAIT (*) ignorer les informations de SACK antérieures pour déterminer quelles données retransmettre.

(*) L'errata 1610 a introduit le changement du DOIT original en DEVRAIT pour les raisons suivantes :

"Au moins un système d'exploitation (Linux) viole le DOIT pour son bien: Même lorsque la fin de temporisation intervient, il conserve les vieilles données de SACK de sorte qu'il peut éviter de retransmettre des données qui sont déjà chez le receveur. Donc, même sous des conditions sévères d'épuisement de bande passante, 100 % des données livrées au receveur causent un progrès de transmission et le système n'est pas sujet à l'écroulement par

encombrement classique (c'est-à-dire, à l'écroulement par encombrement provenant de retransmissions inutiles de paquets).

Lorsque ce projet sera rouvert, ce texte devrait être précisé pour régler un certain nombre de problèmes supplémentaires. En particulier :

- Il a été observé que de vider le tableau des résultats à la fin des temporisations cause parfois uee utilisation très inefficace du réseau, avec de grandes quantités de données dupliquées livrées au receveur.

- Il y a un risque de blocage si la fin de temporisation a été causée par un tableaudes résultats corrompu ou si le receveur renie les blocs SACK. Il est important que les vérifications pour les reniements et les tableaux de résultats incohérents soient robustes. De plus, il devrait probablement y avoir un mécanisme obligatoire de repli, comme d'exiger une retransmission rapide classique et un comportement new reno, ou finalement après des temporisations répétées sans progrès de transmission, la purge du tableau de résultats.

- Rendre le SACK plus robuste en présence de temporisations peut augmenter le risque d'écroulement par encombrement associé à des embouteillages en cascade, parce qu'il permet à TCP de fonctionner avec des taux de perte déraisonnablement élevés".

De futures recherches sur les algorithmes de contrôle de l'encombrement pourront tirer parti des informations supplémentaires fournies par SACK. Un domaine pour ces recherches futures concerne les modifications à TCP pour un environnement sans fil ou par satellite où la perte de paquet n'est pas nécessairement une indication d'encombrement.

6. Efficacité et comportement au pire

Si le chemin de retour qui porte les ACK et les options SACK était sans perte, un bloc par paquet à option SACK serait toujours suffisant. Chaque segment arrivant alors que le receveur des données détient des données discontinues causerait l'envoi d'un ACK par le receveur des données avec une option SACK contenant le bloc altéré dans la file d'attente du receveur. L'envoyeur des données est donc capable de construire une réplique précise de la file d'attente du receveur en prenant l'union de tous les premiers blocs SACK.

Comme le chemin de retour n'est pas sans pertes, l'option SACK est définie comme comportant plus d'un bloc SACK dans un seul paquet. Les blocs redondants dans le paquet d'option SACK augmentent la robustesse de la livraison de SACK en présence d'ACK perdus. Pour un receveur qui utilise aussi l'option Horodatage de la [RFC1323], l'option SACK a de la place pour inclure trois blocs SACK. Donc, chaque bloc SACK va généralement être répété au moins trois fois, si nécessaire, une fois dans chacun des trois paquets ACK successifs. Cependant, si tous les paquets ACK qui rapportent un bloc SACK particulier sont abandonnés, l'envoyeur peut alors supposer que les données de ce bloc SACK n'ont pas été reçues, et retransmettre inutilement ces segments.

Le déploiement d'autres options TCP peut réduire le nombre de blocs SACK disponibles à 2 ou même à 1. Cela va réduire la redondance de livraison des SACK en présence de perte de ACK. Même ainsi, l'exposition du SACK TCP à l'éventualité de retransmission inutile de paquets est strictement inférieure à l'exposition des mises en œuvre actuelles. Les pires conditions nécessaires pour que l'envoyeur retransmette inutilement des données sont exposées plus en détails dans un autre document [Floyd96].

Les mises en œuvre TCP plus anciennes qui n'ont pas l'option SACK seront injustement désavantagées lorsque elles seront en compétition avec des TCP à capacité SACK. Cette question est exposée plus en détails dans [Floyd96].

7. Exemples d'option SACK

Les exemples suivants essayent de montrer le comportement approprié pour la génération de SACK par le receveur des données.

Supposons que le bord gauche de la fenêtre est 5000 et que l'émetteur des données envoie une salve de 8 segments, chacun contenant 500 octets de données.

Cas 1 : Les quatre premiers segments sont reçus mais les quatre derniers sont abandonnés.

Le receveur des données va retourner un segment ACK TCP normal pour accuser réception du numéro de séquence 7000, sans option SACK.

Cas 2 : Le premier segment est abandonné mais les sept restants sont reçus.

À réception de chacun des sept derniers paquets, le receveur des données va retourner un segment ACK TCP qui accuse réception du numéro de séquence 5000 et contient une option SACK spécifiant un bloc de données en file d'attente :

Segment déclencheur	ACK	Bord gauche	Bord droit
5000	(perdu)	-	-
5500	5000	5500	6000
6000	5000	5500	6500
6500	5000	5500	7000
7000	5000	5500	7500
7500	5000	5500	8000
8000	5000	5500	8500
8500	5000	5500	9000

Cas 3 : Les segments 2, 4, 6, et 8 (le dernier) sont abandonnés.

Le receveur des données fait un accusé de réception normal pour le premier paquet. Les paquets trois, cinq, et sept déclenchent les options SACK comme suit :

Segment déclencheur	ACK	Premier bloc		Second bloc		Troisième bloc	
		Bord gauche	Bord droit	Bord gauche	Bord droit	Bord gauche	Bord droit
5000	5500						
5500	(perdu)						
6000	5500	6000	6500				
6500	(perdu)						
7000	5500	7000	7500	6000	6500		
7500	(perdu)						
8000	5500	8000	8500	7000	7500	6000	6500
8500	(perdu)						

À ce point, supposons que le quatrième paquet soit reçu hors de son rang. (Cela pourrait être parce que les données ont été mal mises en ordre dans le réseau, ou parce que le second paquet a été retransmis et perdu, et que le quatrième paquet a été retransmis). À ce point, le receveur des données a seulement deux blocs SACK à rapporter. Le receveur des données répond avec l'Accusé de réception sélectif suivant :

Segment déclencheur	ACK	Premier bloc		Second bloc		Troisième bloc	
		Bord gauche	Bord droit	Bord gauche	Bord droit	Bord gauche	Bord droit
5500	7500	8000	8500				

Supposons à ce point que le second segment soit reçu. Le receveur des données répond alors avec l'accusé de réception sélectif suivant :

Segment déclencheur	ACK	Premier bloc		Second bloc		Troisième bloc	
		Bord gauche	Bord droit	Bord gauche	Bord droit	Bord gauche	Bord droit
6500	5500	6000	7500	8000	8500		

8. Renoncement par le receveur des données

Noter que le receveur des données est autorisé à éliminer les données de sa file d'attente quand elle n'ont pas été acquittées par l'envoyeur, même si les données ont déjà été rapportées dans une option SACK. Une telle élimination de paquets ayant fait l'objet d'un SACK est déconseillée, mais peut être utilisée si le receveur est à court d'espace de mémoire tampon.

Le receveur des données PEUT choisir de ne pas garder les données dont il est fait rapport dans une option SACK. Dans ce cas, la génération de SACK par le receveur répond à des règles supplémentaires :

* Le premier bloc SACK DOIT refléter le segment le plus nouveau. Même si le segment le plus nouveau va être éliminé et si le receveur a déjà éliminé des segments adjacents, le premier bloc SACK DOIT faire rapport, au minimum, des bords gauche et droit du segment le plus nouveau.

* Excepté pour le segment le plus nouveau, tous les blocs SACK NE DOIVENT PAS faire rapport de données anciennes qui ne sont plus réellement détenues par le receveur.

Comme le receveur des données peut ultérieurement éliminer des données rapportées dans une option SACK, l'envoyeur NE DOIT PAS éliminer des données avant qu'elles aient été acquittées par le champ Numéro d'accusé de réception dans l'en-tête TCP.

9. Considérations pour la sécurité

Le présent document ne renforce ni n'affaiblit les propriétés de sécurité actuelles de TCP.

10. Références

[Fall95] Fall, K. and Floyd, S., "Comparisons of Tahoe, Reno, and Sack TCP", ftp://ftp.ee.lbl.gov/papers/sacks.ps.Z, décembre 1995.

[Floyd96] Floyd, S., "Issues of TCP with SACK", ftp://ftp.ee.lbl.gov/papers/issues_sa.ps.Z, janvier 1996.

[Huitema81] Huitema, C., and Valet, I., "An Experiment on High Speed File Transfer using Satellite Links", 7th Data Communication Symposium, Mexico, octobre 1981.

[Jacobson88] Jacobson, V., "Congestion Avoidance and Control", Proceedings of SIGCOMM '88, Stanford, CA., août 1988.

[Keshav94] Keshav, présentation au groupe de recherche Internet de bout en bout, novembre 1994.

[Mathis95] Mathis, M., and Mahdavi, J., "TCP Forward Acknowledgment Option", présentation au groupe de recherche Internet sz bout en bout, juin 1995.

[Partridge87] Partridge, C., communication privée, février 1987.

[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.

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

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

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

[Stevens94] Stevens, W., "TCP/IP Illustrated, Volume 1: The Protocols", Addison-Wesley, 1994.

[Strayer92] Strayer, T., Dempsey, B., and Weaver, A., "XTP -- the xpress transfer protocol". Addison-Wesley Publishing Company, 1992.

11. Adresse des auteurs

Matt Mathis and Jamshid Mahdavi
Pittsburgh Supercomputing Center
4400 Fifth Ave
Pittsburgh, PA 15213
mathis@psc.edu
mahdavi@psc.edu

Sally Floyd
Lawrence Berkeley National Laboratory
One Cyclotron Road
Berkeley, CA 94720
floyd@ee.lbl.gov

Allyn Romanow
Sun Microsystems, Inc.
2550 Garcia Ave., MPK17-202
Mountain View, CA 94043
allyn@eng.sun.com