Vers un mécanisme de contrôle de congestion dirigé par les récepteurs pour le multicast

Vincent Lucas, Jean-Jacques Pansiot, Dominique Grad, Benoit Hilt

Vers un mécanisme de contrôle de congestion dirigé par les récepteurs pour le multicast

Vincent Lucas* — **Jean-Jacques Pansiot*** — **Dominique Grad*** — **Benoît Hilt****

** Laboratoire LSIIT (UMR 7005) - Université Louis Pasteur*
Boulevard Sébastien Brant - BP 10413
F-67412 Illkirch Cedex

{lucas,pansiot,grad}@dpt-info.u-strasbg.fr

*** Laboratoire MIPS (EA2032) - Université de Haute Alsace*
34 rue du Grillenbreit - BP 50568
F-68008 Colmar Cedex

benoit.hilt@uha.fr

RÉSUMÉ. *Pour généraliser le déploiement et l'utilisation du multicast, il est indispensable de disposer d'un mécanisme de contrôle de congestion. De plus, ce mécanisme doit être équitable avec TCP. Plusieurs méthodes dirigées par les récepteurs et utilisant un codage en couches ont été proposées. Ces méthodes sont extensibles aux très grands groupes, mais aucune n'est encore totalement satisfaisante. Dans cet article, nous proposons une nouvelle approche utilisant un codage en couches et s'inspirant de l'analyse de TCP. À partir de mesures expérimentales, nous avons dégagé certains paramètres influant sur l'efficacité et l'équité du contrôle de congestion. Nous proposons plusieurs mécanismes pour réduire significativement le temps de convergence vers un débit stable, et présentons des résultats encourageants en terme d'utilisation du réseau et d'équité avec TCP.*

ABSTRACT. *To generalize the deployment and the use of multicast, a congestion control mechanism is compulsory. Moreover, this mechanism must be fair with TCP. Several receiver-driven layered multicast congestion control scheme have been proposed. They are scalable to very large groups, but none of these are yet fully satisfying. In this article, we describe a new approach taking advantage of the layered scheme and of the analysis of the TCP congestion control scheme. Based on experimental results, we highlight parameters affecting the congestion control efficiency and fairness. Then, we propose several mechanisms to reduce significantly the time to converge to a stable rate, with interesting experimental results in term of throughput and of fairness with TCP.*

MOTS-CLÉS : *Contrôle de congestion, multicast, équité, TCP*

KEYWORDS: *Congestion control, multicast, fairness, TCP*

1. Introduction

Pour généraliser le déploiement et l'utilisation du multicast, un mécanisme de contrôle de congestion est indispensable, comme énoncé dans le RFC 2887 [HAN 00]. Nous nous intéressons aux mécanismes régis par les récepteurs, car utilisables à très grande échelle. La principale proposition est l'utilisation d'un codage en couches cumulatives. Ce principe demande à la source de subdiviser son flux en plusieurs couches ou canaux afin que chaque récepteur puisse indépendemment recevoir le débit qu'il souhaite en s'abonnant au nombre de canaux approprié. Plusieurs variantes ont déjà été proposées dont WEBRC [LUB 02][LUB 04] est sans doute la plus aboutie, notamment en ce qui concerne l'équité envers TCP.
Cette étude débute par l'état de l'art du contrôle de congestion pour le multicast à la section 2, avant d'analyser à la section 3 le comportement de TCP. La section 4 s'inspire de cette analyse pour proposer un premier protocole de contrôle de congestion, dont nous améliorons le temps de convergence dans la section 5. Enfin, la section 6 conclut sur les perspectives et les améliorations à venir de ce protocole.

2. État de l'art

Les méthodes de contrôle de congestion multicast qui exigent une signalisation entre récepteurs et émetteurs ne sont pas extensibles. De plus, les mécanismes utilisant un réseau actif sont actuellement difficilement déployables. Par conséquent, nous nous intéressons au contrôle de congestion multicast utilisant un codage en couches côté source et une gestion indépendante des abonnements aux canaux côté récepteur.

2.1. *Principes généraux côté source*

Les premières solutions, telles que RLM [MCC 96], RLC [VIC 98][RIZ 99] et FLID-SL [BYE 02], proposent de diviser le flux de la source en plusieurs couches ou canaux, chacun identifié par un groupe multicast. Le récepteur adapte son débit en s'abonnant au nombre de canaux adéquat. Cependant, se désabonner d'un groupe n'est pas immédiat et peut prolonger la congestion. Le récepteur n'est alors plus capable de déterminer si une congestion persistante est due au fait que le désabonnement n'est toujours pas effectif au niveau du lien congestionné ou s'il doit se désabonner d'un canal supplémentaire.
Pour corriger cela, FLID-DL [BYE 02] et WEBRC utilisent des canaux dynamiques. Le principe est que la source diminue continuellement le débit des canaux en fonction du temps. Le temps est alors divisé en cycles réguliers dont la durée est nommée *Time Slot Duration* (TSD). À chaque nouveau cycle, un canal est créé au débit maximum et le canal de débit le plus faible se termine et devient muet. Un récepteur peut ainsi réduire son débit en attendant simplement que les canaux diminuent d'eux-mêmes leur débit.

2.2. *Principe de WEBRC côté récepteur*

La méthode de WEBRC se rapproche de celle de TFRC [FLO 00b][HAN 03]. En effet, WEBRC estime la bande passante disponible suivant la formule [1] basée sur l'évaluation de Padhye [PAD 98] du débit moyen de TCP en fonction des pertes et du temps aller-retour nommé *Round Trip Time* (RTT) :

$$debit_{WEBRC} := 1/\left(artt * \sqrt{p} * \left(0.816 + 7.35 * p * \left(1 + 32 * p^2\right)\right)\right) \quad [1]$$

où p représente le pourcentage de pertes et $artt$ une estimation du RTT selon WEBRC. En effet, comme il n'existe pas de communication bidirectionnelle entre les récepteurs et la source, il est difficile d'estimer directement le RTT. Les récepteurs doivent donc se contenter d'une estimation qui pour WEBRC est équivalente au temps d'adhésion à un canal. Un calcul des abonnements à effectuer est réalisé à chaque nouvelle époque de 500 ms, qui est une subdivision du TSD. Finalement, cette décision est prise par comparaison entre l'estimation du débit avec la bande passante utilisée à la prochaine époque.
WEBRC est un protocole qui se veut équitable avec TCP. Cependant dans une étude précédente [LUC 06], nous avons montré l'inadéquation de l'utilisation du temps d'adhésion pour calculer l'estimation du RTT. De plus, même sur le réseau de tests de cet article, où le temps d'adhésion est représentatif du RTT, WEBRC présente encore des problèmes d'équité selon les caractéristiques du réseau, comme nous le verrons aux figures 4 et 6 lors des comparaisons avec les solutions que nous proposons.

3. Analyse de TCP

Dans tout l'internet, le principal contrôle de congestion utilisé est celui de TCP. C'est pourquoi, le protocole de contrôle de congestion pour le multicast que nous voulons élaborer doit être équitable avec TCP. Nous devons donc commencer par comprendre en détail comment fonctionne un TCP standard, par exemple New-Reno.
L'émetteur TCP utilise 2 variables, une fenêtre de congestion ou *Congestion Window* (CW) et un seuil de démarrage lent ou *Slow Start Threshold* (SST), qui sont respectivement initialisés à la taille d'un paquet ou *Segment Size* (SS) et à 64 Ko [JAC 88][STE 97]. L'algorithme de TCP comporte 2 phases et 2 événements :

– La première phase est le **démarrage lent**, conçue pour sonder rapidement la bande passante disponible. Chaque acquittement reçu augmente CW de la taille d'un paquet, ce qui correspond à un doublement par RTT (voir formule [2]), donc à une croissance exponentielle. Cette phase continue tant que $CW \leq SST$.

– La deuxième phase est l'**évitement de congestion**, conçue pour converger lentement vers le débit équitable. La source incrémente CW de la taille d'un paquet par RTT (voir formule [3]). Cette phase correspond à une augmentation linéaire et continue tant que $CW > SST$.

– Le premier événement est une **perte** détectée par exemple par la réception d'un acquittement en triple ou sélectif. Une perte est la manifestation d'une congestion et

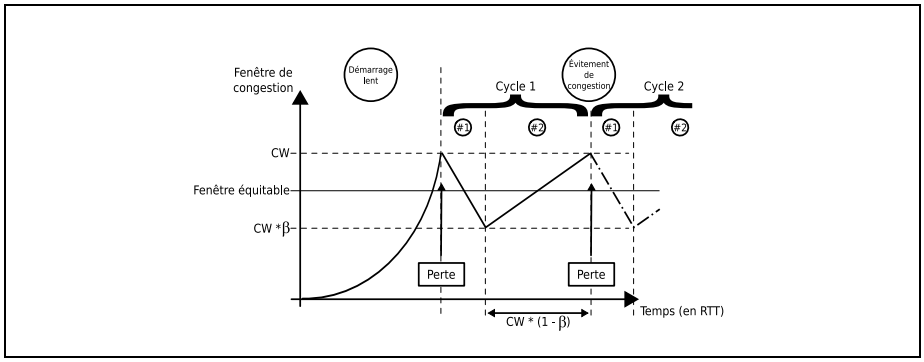


Figure 1. *Comportement de la fenêtre de congestion de TCP*

provoque la réduction de moitié de CW (voir formule [4]). De plus, SST est mis-à-jour et correspond à la nouvelle valeur de CW.

– Le second événement est l'**expiration du minuteur** enclenché au dernier acquittement reçu. Ce minuteur équivaut à un petit multiple du RTT moyen et permet de détecter les congestions sévères. À expiration, TCP reprend la phase de démarrage lent et SST est actualisé à la moitié de CW, qui est lui-même réinitialisé à la taille d'un paquet.

$$\text{Démarrage lent :} \quad CW := \frac{CW}{\beta} \qquad [2]$$

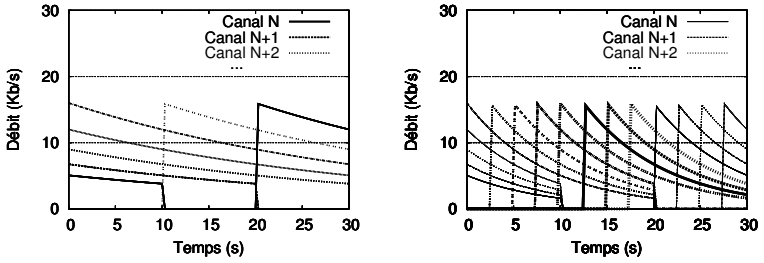
$$\text{Évitement de congestion :} \quad CW := CW + \alpha * \left(\frac{SS * SS}{CW}\right) \qquad [3]$$

$$\text{Perte :} \quad CW := CW * \beta \qquad [4]$$

$$\text{avec} \quad \beta := 0.5 \quad \text{et} \quad \alpha := 1$$

Quand CW atteint la taille correspondant à un partage équitable de la bande passante, le même débit est obtenu par chaque flux TCP de même RTT traversant le lien congestionné. Dans ce cas, chaque flux subit les cycles décrits dans [MAT 97] et représentés à la figure 1, composés d'une perte suivie d'une phase d'évitement de congestion. Chaque perte réduit CW de moitié (#1 de la figure 1). La phase d'évitement de congestion (#2 de la figure 1) nécessite $CW * (1 - \beta)$ RTT pour atteindre de nouveau la taille de CW et ainsi causer une nouvelle perte, signal du commencement d'un nouveau cycle. De plus, les analyses détaillées [FLO 00a][YAN 00] montrent qu'un protocole utilisant une augmentation additive et une diminution multiplicative avec une relation entre α et β telle que définie dans la formule [5] est équitable avec TCP.

$$\alpha = 3 * \frac{(1-\beta)}{(1+\beta)} \qquad [5]$$



(a) Les canaux de la source de WEBRC (b) Les canaux de la source avec réduction accélérée des débits

Figure 2. *Accélération de la réduction dynamique des canaux*

4. Contrôle de Congestion Multicast Inspiré de TCP (CCM-IT)

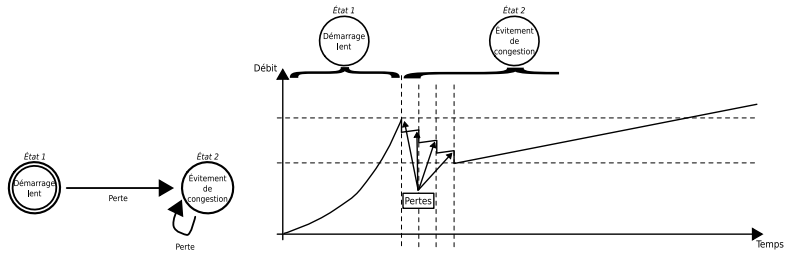
Notre première proposition est l'algorithme CCM-IT, basé sur l'analyse de TCP et sur la source de WEBRC. Cette partie présente une amélioration de la source, avant de se concentrer sur l'algorithme côté récepteur et d'évaluer cette proposition.

4.1. *Amélioration de la source*

Les canaux de la source de WEBRC diminuent trop lentement pour réagir convenablement à une congestion. En effet, cette diminution est de $1 - P = 25\%$ toutes les $TSD = 10$ secondes (voir figure 2(a)), avec $P = 0.75$ le facteur multiplicatif de diminution des canaux. Nous proposons d'utiliser une pente de diminution des canaux de la source plus importante tout en gardant une granularité de débit égale à celle de WEBRC. De plus, nous conservons TSD à 10 secondes pour éviter les problèmes liés au temps d'adhésion [LUC 06].
Par conséquent, nous proposons de passer le facteur multiplicatif de diminution P de WEBRC à P^K. Ainsi, la granularité inter-canal est toujours de P, mais ces derniers diminuent de P^K en un TSD (voir figure 2(b)). De ce fait les canaux réduisent leur débit K fois plus rapidement et au bout de TSD secondes il y a non pas 1 mais K canaux qui deviennent muets. Pour cette étude, nous définissons arbitrairement $K = 4$.

4.2. *Algorithme côté récepteur*

Les récepteurs ajustent leur débit en s'abonnant au nombre de canaux adéquat. Pour évaluer ce nombre, chacun compare le débit actuellement reçu à une estimation du débit équitable. Dès que le débit équitable dépasse celui reçu plus le débit de N canaux, alors le récepteur s'abonne à N canaux supplémentaires. Ainsi, il suffit de



(a) Diagramme états-transitions

(b) Principe de l'estimation du débit équitable

Figure 3. *Contrôle de congestion inspiré de l'analyse de TCP*

définir le calcul de l'estimation du débit équitable.
Le temps aller ou *One Way Delay* (OWD) est utilisé en remplacement du RTT. Pour cela, on suppose que chaque source et chaque récepteur a préalablement synchronisé son horloge par un moyen dont la précision est de l'ordre de la milliseconde, comme le protocole NTP ou la synchronisation par GPS. Cette synchronisation et un champ spécifique contenant l'heure ajouté dans l'entête permettent à chaque récepteur de calculer l'OWD_{actuel} pour chaque paquet reçu. De plus, nous utilisons l'OWD_{moyen}, qui est calculé selon la formule [6].

$$OWD_{moyen} \quad := \quad OWD_{moyen} * \beta + OWD_{actuel} * (1 - \beta) \qquad [6]$$

Pour estimer le débit équitable, nous allons nous inspirer de l'analyse de TCP et définir 2 phases et un événement (voir figure 3) :

– La phase de **démarrage lent** (*état 1* de la figure 3(a)) sert à découvrir la bande passante disponible. Comme pour TCP, nous utilisons une croissance exponentielle. À chaque paquet reçu, l'*Estimation du Débit Équitable* (EDE) augmente selon la formule [7]. Ainsi, l'augmentation est inversement proportionelle à la diminution du débit des canaux de la source. De plus, pour éviter les trop grandes rafales de pertes, cette augmentation est figée tant que l'estimation du $OWD_{actuel} > OWD_{moyen}$.

– La phase d'**évitement de congestion** (*état 2* de la figure 3(a)). C'est une croissance linéaire pour découvrir progressivement la bande passante disponible. EDE est actualisé à chaque réception de paquet en utilisant la formule 8.

– La réaction à une **perte** est une diminution multiplicative (voir formule [9]), proportionnelle à la réduction continue du débit des canaux de la source.

$$\text{Démarrage lent :} \quad EDE \quad := \quad \frac{EDE}{\beta} \qquad [7]$$

$$\text{Évitement de congestion :} \quad EDE \quad := \quad EDE + \alpha * \left(\frac{SS * SS}{CW} \right) \qquad [8]$$

$$\text{Perte :} \quad EDE := EDE * \beta \qquad [9]$$

$$\text{avec} \quad \beta := P^{(\frac{TIP}{TSD} * K)} \qquad [10]$$

$$\text{Ainsi que :} \quad \alpha := 3 * \frac{(1-\beta)}{(1+\beta)} \quad \text{et} \quad CW := \frac{EDE}{8} * OWD_{moyen}$$

Le calcul de α est le même que pour TCP, mais dans notre cas β est dynamique. En effet, β est déterminé par la réduction du débit des canaux de la source pendant le dernier *Temps Inter-Paquets* (TIP). Comme β est exprimé en fonction de TIP (voir formule [10]), cela signifie que β est actualisé pour chaque nouveau paquet reçu.

4.3. *Méthode de mesure*

Nous avons implémenté les différents algorithmes et les avons évalués en utilisant la maquette (voir figure 4(d)) qui utilise les protocoles de routage multicast MLD et PIM-SM. Cette dernière comporte un lien faible à 1 Mb/s entre des routeurs Linux qui nous permettent de configurer le temps de propagation et la taille des files d'attente. Pendant les différents tests, l'utilisation de la bande passante est toujours supérieure à 95%, ce qui nous permet d'étudier l'équité selon la formule [11].

$$EQ := \frac{debit_{multicast}}{debit_{TCP}} \qquad [11]$$

avec $debit_{multicast}$ le débit moyen par flux multicast

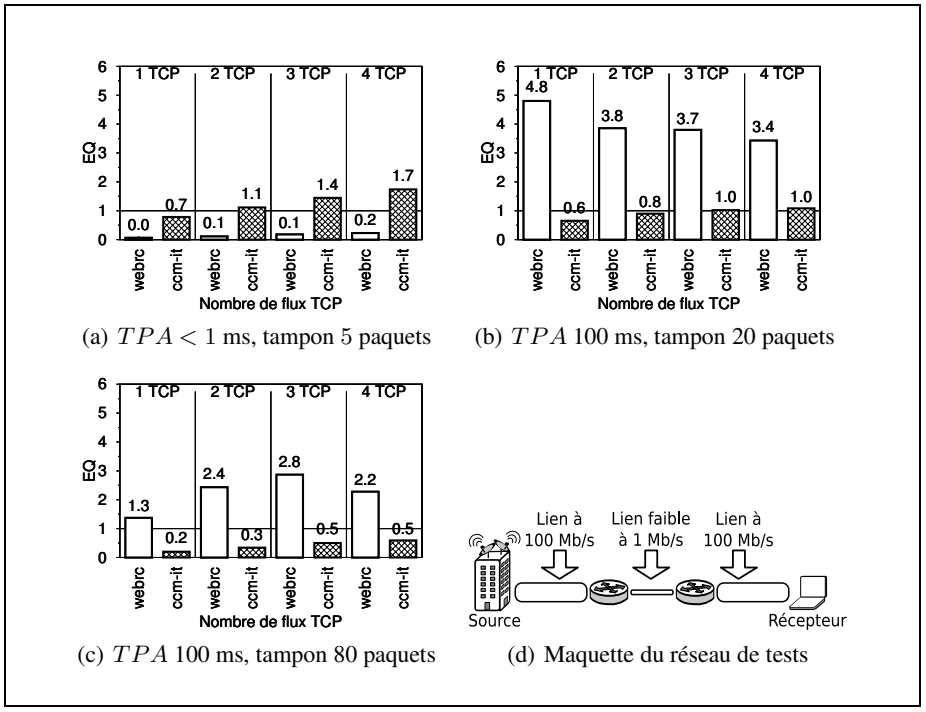
et $debit_{TCP}$ le débit moyen par flux TCP

Chaque test dure 10 minutes et révèle le comportement des protocoles en fonction du temps de propagation, de la taille des tampons et du nombre de flux TCP concurrents.

4.4. *Résultats*

Nous avons réalisé de nombreux tests en faisant varier la taille des files d'attente du lien faible de 5, 10, 20, 40, 60, 80 et 100 paquets, ainsi que le *Temps de Propagation Ajouté* (TPA) dans chaque sens de 0, 25, 50, 75 et 100 ms. Le OWD observé sera donc principalement la somme du TPA et du délai dans la file d'attente. Après avoir expérimenté toutes les combinaisons de ces paramètres, nous avons dégagé 3 comportements :

– Le comportement 1 apparaît quand les tampons des routeurs sont très petits, comme par exemple à la figure 4(a) avec un tampon de 5 paquets. Certes, CCM-IT semble équitable, mais le taux de perte d'environ 30% est beaucoup trop élevé. En ce qui concerne WEBRC, ce dernier présente un taux de perte d'environ 8% et il sous-utilise la bande passante. Par conséquent, les comportements de WEBRC et CCM-IT ont des dysfonctionnements qui montrent les limitations de ces méthodes quand les tampons sont sous-dimensionnés. Cependant, on peut considérer ce scénario comme



(a) $TPA < 1$ ms, tampon 5 paquets (b) TPA 100 ms, tampon 20 paquets

(c) TPA 100 ms, tampon 80 paquets (d) Maquette du réseau de tests

Figure 4. *Résultats de CCM-IT en fonction du nombre de flux TCP*

étant rare et fortement improbable au vu des configurations actuelles des routeurs qui par défaut utilisent bien plus d'espace mémoire pour les files d'attente.

– Le comportement 2 (voir figure 4(b)) apparaît avec des files d'attente de plus grande taille et OWD inférieur à environ 350 ms. Dans ces conditions, CCM-IT se montre équitable envers les flux TCP, avec un taux de perte inférieur à 4%. Pour WEBRC, le taux de perte est similaire, mais les flux utilisent 4 fois plus de bande passante que les flux TCP. Ce comportement montre que l'estimation de CCM-IT est plus précise que celle de WEBRC pour les configurations de réseaux les plus courantes.

– Le troisième comportement (voir figure 4(c)) apparaît quand OWD est supérieur à environ 350 ms. CCM-IT sous-utilise la bande passante alors que WEBRC est relativement équitable, même si cette équité varie en fonction du nombre de flux TCP concurrents. Dans les deux cas le taux de perte est toujours inférieur à 4%. En réalisant des tests de plusieurs heures dans les mêmes conditions, on peut observer que CCM-IT est finalement équitable avec TCP. Cela signifie que le problème vient du temps de convergence qui ici peut atteindre une heure pour un RTT d'une seconde.

CCM-IT est donc équitable pour les configurations de réseaux les plus courantes. Cependant, sa lenteur de convergence le rend difficilement utilisable en particulier pour le comportement 3.

5. Contrôle de Congestion Multicast avec Amélioration du Temps de Convergence (CCM-ATC)

La lenteur de convergence de l'algorithme CCM-IT est due à la phase d'évitement de congestion, créée pour atteindre en douceur le débit idéal. Ainsi, quand la phase de démarrage lent s'interrompt trop tôt, la phase d'évitement de congestion est utilisée alors que le débit est loin d'être équitable. Ce problème est similaire à celui de TCP pour les réseaux haut débit avec de longs délais. Pour améliorer CCM-IT, nous proposons CCM-ATC qui s'inspire de protocoles tels que BIC [LIS 04] et CUBIC[INJ 05].

5.1. *Analyse approfondie des cycles de pertes TCP*

Il faut donc déterminer la distance au débit équitable pour reprendre le cas échéant le démarrage lent et ainsi réduire le temps de convergence. Or, l'analyse de TCP montre que ce dernier est proche du débit équitable si le flux subit des pertes périodiques aux alentours du même débit. Ainsi, aux alentours du débit équitable :

– Une perte est attendue quand la période séparant deux pertes consécutives est écoulée. Cette période est estimée par $T := NB\_RTT * OWD_{moyen}$.

– Une perte est attendue quand le débit utilisé est proche de celui auquel la dernière perte a été détectée. On peut définir que le débit R est proche du débit D de la perte précédente si $seuil_{min} < R < seuil_{max}$ avec $seuil_{min} := D{*}\beta^2$ et $seuil_{max} := \frac{D}{\beta^2}$.

5.2. *Amélioration du récepteur*

Avec ces nouvelles définitions nous pouvons ajouter 2 nouveaux états pour améliorer notre algorithme d'évitement de congestion comme décrit figure 5 :

– La phase de **démarrage lent** à l'*état 3*. Quand la durée depuis la dernière perte est trop longue, c'est-à-dire depuis plus de T secondes. Nous utilisons alors une croissance exponentielle pour converger rapidement vers le débit de la dernière perte.

– La phase d'**évitement de congestion** à l'*état 4*. Quand le débit est proche du débit reçu au moment de la dernière perte, nous ralentissons la croissance de notre estimation. Cela signifie que pour $seuil_{min} < EDE < seuil_{max}$, EDE augmente linéairement pour découvrir progressivement la bande passante disponible. Si $EDE > seuil_{max}$ et que le temps depuis la dernière perte est supérieur à $\gamma{*}T$ [1], alors le débit reçu lors de la dernière perte est trop loin du débit équitable, ce qui permet de reprendre le démarrage lent de l'*état 1* et ce jusqu'à la prochaine perte.

– Une **perte** provoque toujours un retour à l'*état 2* et réinitialise le minuteur T. De plus, si l'état précédent n'était pas l'*état 2*, nous recalculons les valeurs de $seuil_{min}$

1. Dans la suite, nous prenons $\gamma = 3$. Comme T est calculé en fonction de OWD_{moyen} qui peut être 2 fois plus petit que le RTT, il faut donc au moins que $\gamma \geq 2$ pour équilibrer cette approximation et $\gamma = 3$ permet d'être prudent avant de repartir en mode de démarrage lent.

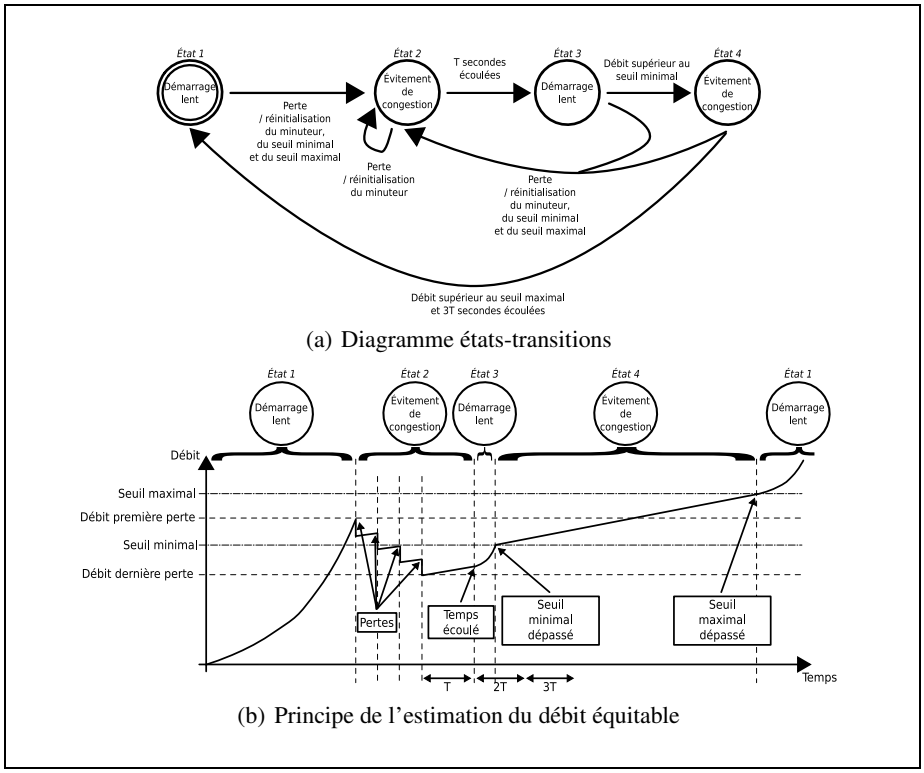


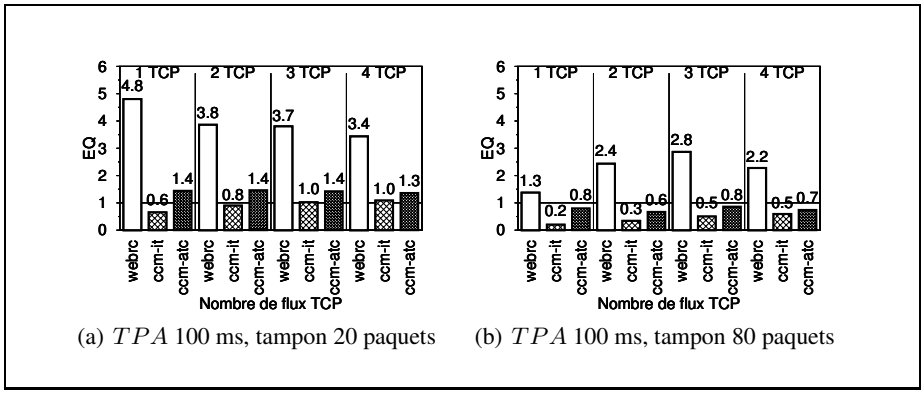
Figure 5. *Algorithme d'amélioration du temps de convergence*

et de $seuil_{max}$. De cette manière, en cas de rafale, ces seuils sont fixés à la valeur correspondant au débit lors de la première perte de la rafale.

D'une part, quand le débit est proche du débit équitable (*états 2 et 4*), cet algorithme croît linéairement, ce qui permet d'effectuer un partage équitable de la bande passante. D'autre part, quand le débit est éloigné du débit équitable (*états 1 et 3*), cet algorithme croît exponentiellement pour converger rapidement vers le débit équitable.

5.3. *Résultats*

Les mêmes tests que pour WEBRC et CCM-IT sont réalisés pour CCM-ATC afin de pouvoir comparer les 3 protocoles. Tout d'abord, le comportement 1 est identique pour CCM-IT et CCM-ATC en terme d'équité comme en terme de taux de pertes pour le cas des files d'attente sous-dimensionnées. Cependant, les tests de CCM-ATC (voir figure 6) ne présentent pas le comportement 3 de CCM-IT. Ainsi, CCM-ATC reste équitable pour tous les réseaux dont les files d'attente ne sont pas sous-dimensionnées tout en améliorant sensiblement le temps de convergence qui est divisé par un facteur



(a) TPA 100 ms, tampon 20 paquets (b) TPA 100 ms, tampon 80 paquets

Figure 6. *Résultats de CCM-ATC en fonction du nombre de flux TCP*

3 à 5 pour la figure 6(a) et converge en moins de 2 minutes là où CCM-IT en prenait plus de 60. Ces améliorations permettent donc de réduire le temps de convergence tout en stabilisant les flux au débit équitable. Ainsi, c'est l'auto-synchronisation de TCP qui permet de synchroniser indirectement les flux multicast.

6. Conclusion et perspectives

Cette étude nous a permis d'analyser le comportement de TCP et d'en dériver deux algorithmes de contrôle de congestion multicast. Le premier CCM-IT est fonctionnel tant que les tampons des routeurs ne sont pas sous-dimensionnés. Cependant, son temps de convergence très lent le rend inutilisable pour les réseaux où le RTT est supérieur à 350ms. Afin de résoudre ce problème, CCM-ATC propose de reprendre le mode de démarrage lent quand la probabilité d'être proche du débit équitable est trop faible. Cette probabilité est évaluée grâce à l'analyse des cycles de pertes que provoque TCP à un débit équitable. Finalement, les résultats de CCM-ATC montrent que ce dernier est équitable avec TCP tout en améliorant significativement le temps de convergence.
Ainsi ce protocole fonctionne dans les cas où TCP est majoritaire sur le réseau, ce qui est le cas le plus courant dans l'internet actuel. Cependant, si les flux TCP ne sont pas majoritaires, ces cycles de pertes ne sont plus respectés. Cela favorise le retour des flux multicast en mode de démarrage lent même si les récepteurs sont proches du débit équitable. La suite de nos travaux portera sur ce problème pour lequel il faudra pouvoir déterminer dynamiquement si les flux multicast sont majoritaires pour adapter l'agressivité du protocole. En outre, ces résultats doivent être complétés par de plus amples expérimentations impliquant des récepteurs ayant des OWD hétérogènes, ainsi que des tests avec des flux dynamiques. De plus, nous étudierons l'influence du facteur multiplicatif de diminution des canaux, ainsi que l'impact d'un tel contrôle de congestion sur la diffusion vidéo hiérarchique.

7. Bibliographie

[BYE 02] BYERS J. W., HORN G., LUBY M., MITZENMACHER M., SHAVER W., « FLID-DL : Congestion Control for Layered Multicast », *IEEE journal on selected areas in communications*, vol. 20, No.8, 2002, p. 1558–1570.

[FLO 00a] FLOYD S., HANDLEY M., PADHYE J., « A Comparison of Equation-Based and AIMD Congestion Control », 2000, www.aciri.org/tfrc/aimd.pdf.

[FLO 00b] FLOYD S., HANDLEY M., PADHYE J., WIDMER J., « Equation-based congestion control for unicast applications », *SIGCOMM 2000*, Stockholm, August 2000, p. 43-56.

[HAN 00] HANDLEY M., FLOYD S., WHETTEN B., KERMODE R., VICISANO L., LUBY M., « The Reliable Multicast Design Space for Bulk Data Transfer », RFC : Informational n° 2887, August 2000, IETF.

[HAN 03] HANDLEY M., FLOYD S., PADHYE J., WIDMER J., « TCP Friendly Rate Control (TFRC) : Protocol Specification », RFC : Standards Track n° 3448, January 2003, IETF.

[INJ 05] INJONG R., LISONG X., « CUBIC : A New TCP-Friendly High-Speed TCP Variant », rapport, feb 2005, PFLDnet 2005, Ecole Normale Supérieure - LYON - FRANCE.

[JAC 88] JACOBSON V., « Congestion Avoidance and Control », *ACM SIGCOMM '88*, Stanford, CA, August 1988, p. 314-329.

[LIS 04] LISONG X., HARFOUSH K., INJONG R., « Binary increase congestion control (BIC) for fast long-distance networks », *IEEE INFOCOM'04*, vol. 4, 7-11 March 2004, p. 2514-2524.

[LUB 02] LUBY M., GOYAL V., SKARIA S., HORN G., « Wave and Equation Based Rate Control Using Multicast Round Trip Time », *ACM SIGCOMM'02*, 2002.

[LUB 04] LUBY M., GOYAL V., « Wave and Equation Based Rate Control building block », RFC : Experimental n° 3738, April 2004, IETF.

[LUC 06] LUCAS V., PANSIOT J.-J., HOERDT M., « Influence du temps d'adhésion sur le contrôle de congestion multicast », FLEURY E., KAMOUN F., Eds., *CFIP 2006*, Tozeur (Tunisie), october 2006, Hermès, p. 115–126.

[MAT 97] MATHIS M., SEMKE J., MAHDAVI J., « The Macroscopic Behavior of the TCP Congestion Avoidance Algorithm », *ACM - CCR*, vol. 27, n° 3, 1997.

[MCC 96] MCCANNE S., JACOBSON V., VETTERLI M., « Receiver-driven layered multicast », *SIGCOMM '96*, New York, NY, USA, 1996, ACM, p. 117–130.

[PAD 98] PADHYE J., FIROIU V., TOWSLEY D., KUROSE J., « Modeling TCP Throughput : A Simple Model and its Empirical Validation », *Proceedings of the ACM SIGCOMM '98*, 1998, p. 303–314.

[RIZ 99] RIZZO L., VICISANO L., CROWCROFT J., « The RLC multicast congestion control algorithm », 1999, http ://info.iet.unipi.it/ luigi/rlc99.ps.gz.

[STE 97] STEVENS W., « TCP Slow Start, Congestion Avoidance, Fast Retransmit, and Fast Recovery Algorithms », RFC : Standards Track n° 2001, January 1997, IETF.

[VIC 98] VICISANO L., RIZZO L., CROWCROFT J., « TCP-Like congestion control for layered multicast data transfert », *IEEE INFOCOM'98*, 1998.

[YAN 00] YANG Y. R., LAM S. S., « General AIMD congestion control », *ICNP '00*, Washington, DC, USA, 2000, IEEE Computer Society, p. 187–198.