GXCAST : UNE GENERALISATION DU PROTOCOLE XCAST

Ali Boudani,
Doctorant en informatique
ali.boudani@irisa.fr , + 33 2 99 84 25 37

Alexandre Guitton,
Doctorant en informatique
alexandre.guitton@irisa.fr , + 33 2 99 84 71 30

Bernard Cousin,
Professeur en informatique
bernard.cousin@irisa.fr , +33 2 99 84 73 33

Adresse professionnelle
Université de Rennes I ★ Campus de Beaulieu ★ 35 042 Rennes Cedex

Résumé : Dans ce papier, nous étudions une généralisation du protocole Xcast. Après avoir introduit les deux protocoles Xcast et Xcast+, leurs avantages et leurs inconvénients, nous présentons notre protocole GXcast : le format d'un paquet GXcast et l'algorithme de routage. La résistance au facteur d'échelle du protocole GXcast est comparée à celle d'autres protocoles *multicast*. Nous proposons notamment une fonction permettant de réduire les inconvénients de ce type de protocoles et évaluons le protocole GXcast en termes de surcoût et de délai. Les résultats de simulation confirment notre analyse.

Summary : In this paper, we study a generalization of the Xcast protocol. After having introduced both Xcast and Xcast+ protocols, their advantages and drawbacks, we present our protocol Gxcast: the GXcast packet format and the forwarding algorithm. The scalability of the GXcast protocol is compared to the scalability of other multicast protocols. We propose a function that allow to reduce the drawbacks of this kind of protocols and we evaluate the GXcast protocol in terms of overhead and delay. The simulation results confirm our analysis.

Mots clés : Routage explicite, analyse de performance, protocole *multicast*.

GXcast : une généralisation du protocole Xcast

1 – INTRODUCTION

Un unique protocole de routage *multicast* est incapable de servir les différents types d'applications *multicast*, comme l'indique la charte du groupe de travail *Reliable Multicast Transport* de l'IETF (2003). En effet, le nombre d'applications *multicast* ayant souvent des exigences multiples voire contradictoires ne cesse de croître. Les protocoles de routage *multicast* doivent présenter une certaine flexibilité selon les besoins des différentes applications. Dans un réseau où il y a un très grand nombre de groupes *multicast* de petite taille (technique SGM : *Small Group Multicast*, présentée par Ooms (2000)) dont les destinataires sont largement dispersés, le modèle de *multicast* traditionnel ne convient pas.

1.1 - Le protocole Xcast

Explicit Multicast (Xcast) a été proposé par Boivie *et al* (2000) pour résoudre le problème de passage a l'échelle des protocoles *multicast* et pour servir des groupes ayant peu de membres tout en minimisant la consommation de la bande passante. La source encode explicitement la liste des destinations dans l'en-tête Xcast d'un paquet au lieu d'utiliser une adresse *multicast* et envoie le paquet vers son routeur désigné (c'est-à-dire le routeur Xcast en charge de cette source). Chaque routeur du chemin analyse l'en-tête Xcast, classe les destinations selon leur prochain routeur au sens *unicast* et envoie une copie[1] vers chacun des prochains routeurs. Dans le cas où il reste une seule destination dans la liste, le paquet Xcast est transformé par l'algorithme X2U (Xcast à *unicast*) en un paquet *unicast*.

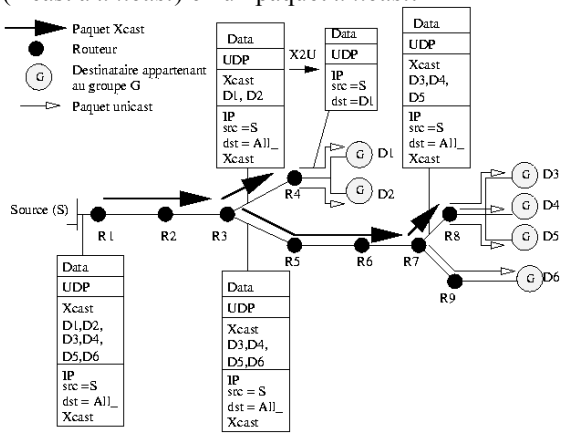

Figure 1 - La transmission d'un paquet Xcast.

Exemple : considérons le groupe représenté sur la figure 1, comportant une source *S* et six destinataires *D1*, *D2*, *D3*, *D4*, *D5* et *D6*. La source *S* envoie un paquet Xcast contenant la liste des destinations *(D1,D2,D3,D4,D5,D6)* à *R1*. *R1* traite ce paquet comme un paquet Xcast quelconque : *R2* est le prochain routeur sur chacun des chemins *unicast* de *R1* à *Di*, donc le paquet Xcast entier est transmis à *R2*. *R2* envoit à son tour le paquet à R3. Lorsque *R3* reçoit le paquet, il en envoie une copie au routeur *R4* avec dans l'en-tête Xcast la liste *(D1,D2)* et une copie au routeur *R5* avec dans l'en-tête Xcast la liste *(D3,D4,D5,D6)*. *R4*, recevant le paquet qui lui est destiné, enverra à son tour à *D1* le message Xcast contenant la liste *(D1)* et à *D2* le message Xcast contenant la liste *(D2)*. *D1* pourra extraire du message qu'il recevra les données qui lui sont utiles. Le comportement est similaire pour les routeurs *R5* à *R9* et pour les cinq autres destinataires.

1.2 – Le protocole Xcast+

Xcast+ a été proposé par Myung-Ki *et al* (2001) pour résoudre le problème de passage à l'échelle de Xcast pour les groupes de taille moyenne. Il est basé sur le protocole Xcast (il utilise d'ailleurs un en-tête très similaire) et permet de réduire la liste des destinataires en utilisant habilement le protocole de gestion locale des groupes d'Internet (protocole IGMP décrit dans la RFC 3376 (2002)). En effet, un récepteur désirant faire partie du groupe *(S,G)* (un groupe, appelé session *multicast* dans la terminologie Xcast, est identifié par le canal *(S,G)* où *S* est l'adresse de la source et *G* l'adresse du groupe) émet un message *join* IGMP à destination du groupe *(S,G)*. Quand le routeur désigné reçoit ce message, il envoie à la source *S* un message de demande d'enregistrement Xcast+ contenant l'adresse de la source *S*, l'adresse de groupe *G*, et sa propre adresse *DR*. Lorsque le DR associé à la source reçoit ce message, il maintient les adresses de tous les routeurs DR ayant des récepteurs appartenant au groupe *multicast (S,G)*.
Lorsque la source envoie les paquets *multicast*, le DR de la source crée un paquet Xcast+ dans lequel il encode explicitement la liste des DR associés aux destinations dans l'en-tête Xcast+, il complète le paquet avec les données à envoyer et émet ce paquet vers le(s) prochain(s) routeur(s) concerné(s) (M2X : *multicast* à Xcast+). Le chemin suivi par le paquet Xcast+ est le même que celui suivi par le paquet Xcast. La seconde différence a lieu aux DR récepteurs : les paquets Xcast+ qui leur parviennent sont convertis en paquets *multicast* et envoyés aux réseaux dont les DR ont la charge (X2M : Xcast à *multicast*).

[1] Cette copie est souvent légèrement modifiée.

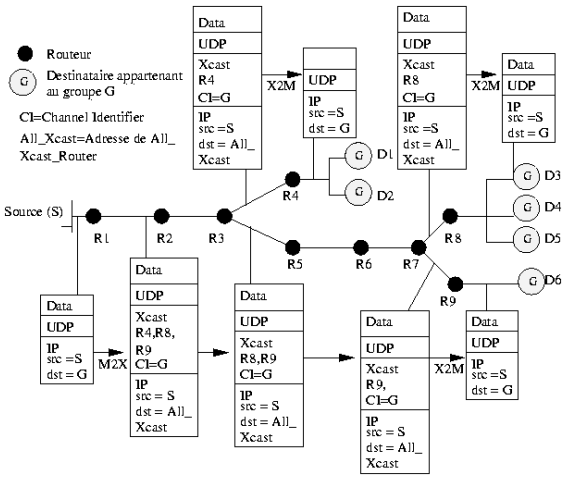

Figure 2 – Exemple de la transmission d'un paquet Xcast+.

Exemple : prenons le même exemple que celui du paragraphe 1.1 : considérons le groupe représenté sur la figure 2, comportant une source *S* et les six destinataires *D1*, *D2*, *D3*, *D4*, *D5* et *D6*. *S* envoie un paquet *multicast* ayant *G* comme adresse destination. Ce message parvient au routeur désigné de son réseau, *R1* dans notre cas. Lorsque *R1* reçoit ce paquet, il génère (en utilisant l'algorithme de transformation M2X) un paquet Xcast+ avec la liste de destinations *(R4,R8,R9)* dans son en-tête Xcast+ et il applique alors au paquet l'algorithme d'émission Xcast+. Cet algorithme est similaire à celui de Xcast à une nuance près : quand un routeur désigné, par exemple *R4*, reçoit un paquet qui lui est destiné, il applique à ce paquet l'algorithme de transformation X2M et envoie le paquet *multicast* ainsi généré aux réseaux dont il est le routeur désigné.

Alors que Xcast ne permet que de gérer efficacement les petits groupes, Xcast+ permet la gestion de groupes *multicast* de taille moyenne. En effet, le facteur limitant est principalement le nombre d'entrées dans la liste des destinataires, et cette liste est réduite dans Xcast+ (Myung-KI *et al*, 2001).

1.3 – Avantages et inconvénients de la technique Xcast

Par la suite, nous désignerons sous le nom générique Xcast les deux protocoles Xcast et Xcast+ et nous appellerons technique Xcast le codage explicite d'une liste de destinataires dans les paquets ainsi que les algorithmes décrits précédemment.

L'utilisation de Xcast présente des avantages et introduit quelques inconvénients par rapport aux protocoles de routage traditionnels :

1) Avantages de la technique Xcast

a) Gestion des états de routage et des messages de signalisation : une des principales caractéristiques du protocole Xcast est qu'il élimine d'une part les états de routage *multicast* au sein des routeurs et d'autre part la nécessité de mettre en place un mécanisme de signalisation spécifique au routage *multicast* entre les routeurs. Il élimine de plus le besoin de protocoles de routage *multicast* intra-domaines et inter-domaines. Cette caractéristique lui permet d'être capable de gérer de nombreuses sessions simultanément.

b) Ingénierie de trafic simplifiée : l'ingénierie de trafic *multicast* est transformée en ingénierie de trafic *unicast*. En effet, le routage Xcast est basé sur le routage *unicast*, les outils *unicast* pouvant donc être utilisés. De plus, les changements de topologie dans Xcast sont pris en compte naturellement sans qu'il n'y ait besoin d'établir une communication supplémentaire entre le protocole *unicast* et le protocole Xcast. Ainsi, le temps de réaction aux pannes est plus court avec Xcast qu'avec les protocoles *multicast* traditionnels. Outre ces deux avantages, on peut citer les deux suivants :

- Xcast ne nécessite pas de mécanisme complexe d'allocation d'adresses *multicast* puisque l'adresse d'une session est *(S,G)* et non pas *G* uniquement.
- Xcast ne nécessite pas que les chemins soient symétriques puisqu'il se base entièrement sur les chemins *unicast*.

2) Inconvénients de la technique Xcast

a) Fragmentation des paquets Xcast : un paquet IP circulant sur un réseau peut être amené à être fragmenté si sa taille dépasse la capacité du lien qu'il doit emprunter. La fragmentation est un mécanisme IP qui tronque un paquet en plusieurs paquets IP autonomes, c'est-à-dire que chacun est muni d'un en-tête IP valide, et qui partage les données parmi ces deux paquets. La figure 3 montre l'effet qu'aurait une fragmentation sur un paquet Xcast. On peut voir que seul le premier paquet résultant est un paquet Xcast valide, puisqu'il est le seul paquet à contenir un en-tête Xcast (il ne comporte cependant aucune données provenant de l'application). Les trois autres paquets ne seront pas traités comme des paquets Xcast : ils ne parviendront pas aux destinataires du groupe. Pour interdire à un paquet Xcast d'être fragmenté, le drapeau DF (*Don't Fragment*) de son en-tête IP doit être positionné à 1. Dans ce cas, si un lien est incapable de transmettre le paquet sans le fragmenter, le paquet est détruit.

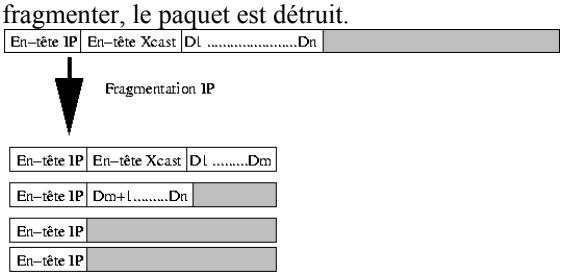

Figure 3 – Le problème de la fragmentation d'un paquet Xcast.

b) Surcharge introduite par l'en-tête Xcast : comme un paquet Xcast ne peut être fragmenté, le volume de données utiles qu'il peut contenir est restreint. Ce volume est d'autant plus petit que le nombre de destinataires est important, ce qui pose de nombreux problèmes dont la diminution notable de l'efficacité.

c) Changement des en-têtes des paquets à acheminer : un troisième problème est qu'à un paquet Xcast arrivant, un routeur Xcast doit associer plusieurs paquets Xcast dont les en-têtes sont différents : en effet, la liste des destinataires est différente pour chacun des prochains routeurs. Ce problème est atténué par la proposition d'utiliser une table d'indicateurs permettant de n'effectuer qu'un minimum de modifications dans l'en-tête afin de réduire l'impact sur le traitement et sur le calcul de la somme de contrôle des paquets à émettre.

3) Critiques de la technique Xcast

a) Utilisation d'*unicast* plutôt que du protocole Xcast : si le nombre de membres d'un groupe *multicast* est très limité, il est possible d'envoyer à chacun de ces membres un message *unicast*. Cette technique n'est pas toujours aisée à mettre en place, notamment lorsque la limitation de bande passante du « dernier mile » (*last mile*) rend nécessaire l'utilisation du *multicast* (Boivie *et al*, 2000).

b) Temps de réaction aux départs des membres : lorsqu'un membre désire quitter un groupe, il (ou bien son DR dans le cas de Xcast+) envoie un message à la source de ce groupe. Celle-ci prend en compte cette requête et cesse de lui faire parvenir des messages. Cependant, les paquets envoyés pendant le laps de temps entre lequel le membre a envoyé le message de désabonnement et où la source a pris en compte cette demande vont continuer d'arriver au destinataire. Une solution à ce problème a été proposée par Boudani et Cousin (2000).

2 – UNE GENERALISATION DE LA TECHNIQUE XCAST

GXcast (*Generalized Xcast*) est une adaptation de la technique Xcast conçue pour résoudre le problème de la fragmentation de la technique Xcast et pour permettre un plus grand nombre de membres par groupe.

2.1 – Description du protocole GXcast

GXcast utilise le même principe que Xcast mais il limite explicitement le nombre maximum n_m de destinataires autorisés dans un paquet Xcast dès la source. La source dans GXcast partitionne la liste initiale *L* de destinataires en plusieurs sous-listes de destinataires *Li*. Chacune de ces listes *Li* contiendra au plus n_m membres. Autant de paquets Gxcast que de listes *Li* seront envoyés, chacun contenant dans l'en-tête GXcast les membres contenus dans la liste correspondante. Ce mécanisme est en fait un mécanisme de fragmentation à la source.

Exemple : considérons le groupe représenté sur la figure 4, comportant une source *S* et six destinataires *D1*, *D2*, *D3*, *D4*, *D5* et *D6*. Dans un premier temps, décrivons le procédé d'adhésion d'un membre à un groupe. Chacun des destinataires envoie un message *join* IGMP qui parvient au routeur désigné de son sous-réseau. Dans notre exemple, *R4*, *R8* et *R9* reçoivent la demande d'adhésion IGMP. Ces routeurs désignés envoient alors des messages *join* vers la source *S* (message que *R1* peut intercepter) qui ajoute effectivement le routeur désigné dans la liste des routeurs destinataires. À présent, étudions l'algorithme d'émission de la source. Pour couvrir les *n* membres en limitant le nombre de destinataires par paquets à n_m, il faut envoyer $\lceil n/n_m \rceil$ paquets. Prenons comme exemple le cas où n_m vaut 2 et plaçons nous dans le cas du protocole Xcast+ qui considère n=3 membres[2]. Le nombre de paquets à générer va être égal à 2, le premier paquet GXcast étant à destination des membres de la liste *(R4,R8)* et le second à destination du membre de la liste *(R9)*. Ces paquets GXcast peuvent être considérés comme des paquets Xcast indépendants et subissent donc un traitement similaire.

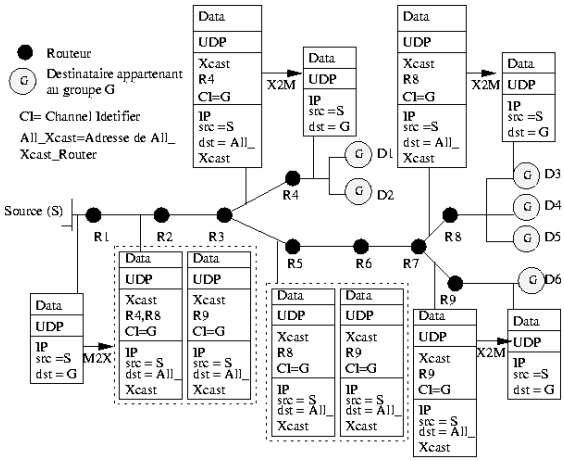

Figure 4 – Exemple de la transmission d'un paquet dans GXcast.

2.2 – Le format d'un paquet GXcast

Un paquet GXcast est très similaire à un paquet Xcast. Décrivons d'abord brièvement le format d'un paquet Xcast.

Un paquet Xcast est constitué de trois parties : un en-tête IP, un en-tête Xcast et une partie données. La source de l'en-tête IP est la source du groupe *multicast*, la destination de l'en-tête IP est l'adresse

[2] Ces petites valeurs de *nm* et de *n* ne sont données qu'à but pédagogique. Les valeurs réalistes de *nm* seront données ultérieurement. On peut néanmoins noter que, pour des petits groupes, GXcast a un comportement identique à Xcast+.

All_Xcast_Routers, le type de protocole de l'en-tête IP est XCAST_PROTO et le drapeau DF est positionné. Un champ *Destination* et un champ *Protocol* sont inclus dans l'en-tête Xcast pour indiquer respectivement l'adresse de groupe et potentiellement le protocole suivant l'en-tête Xcast (traditionnellement UDP).
Un paquet GXcast est lui aussi constitué de trois parties : un en-tête IP, un en-tête GXcast et une partie données. La différence entre un paquet GXcast et un paquet Xcast est minime :

- Le champ *Destination* de l'en-tête IP est l'adresse All_GXcast_routers.
- Le type de protocole de l'en-tête IP est GXCAST_PROTO.
- Le champ indiquant le nombre de destinataires dans un paquet GXcast doit pouvoir permettre de stocker un grand nombre de destinataires. Il semble que la limitation à 7 bits proposée dans Xcast soit insuffisante (en effet, le calcul du nombre maximal de destinataires *n*max effectué dans le paragraphe suivant aboutit à une valeur de 134, ce qui est au-dessus de la valeur maximale représentable avec 7 bits).
- Le champ indiquant la taille de l'en-tête GXcast doit lui aussi être suffisamment grand.

Il semble raisonnable de considérer qu'un en-tête GXcast peut être stocké sur 16 octets[3]. Combiné à l'en-tête IP dont la taille est de 20 octets pour IPv4, la taille totale des deux en-têtes sans la liste des destinataires de GXcast atteint donc E=36 octets.

2.3 - Liens entre GXcast et l'unité de transfert maximale

L'unité de transfert maximale (MTU, *Maximum Transfer Unit*) est la taille du plus grand paquet pouvant circuler sur un lien sans subir de fragmentation. Comme un paquet GXcast ne peut être fragmenté au sens IP, cette taille va limiter le nombre de destinataires par paquet, c'est-à-dire la valeur de n_m. L'étude que nous proposons se base sur le fait que la MTU est connue *a priori*. Par la suite, nous utiliserons la valeur M=576 octets qui est la MTU minimale garantie pour le protocole IPv4[4]. Le dernier paramètre à fixer est IP=4 qui est la taille en octets d'une adresse IPv4[5]. De ces valeurs, on peut calculer le nombre de destinataires maximal par paquet *n*max de la manière suivante :

$$n\max=\left\lfloor \frac{M-E-1}{IP} \right\rfloor =134.$$

Cette expression intègre le fait qu'un paquet GXcast est censé contenir au moins un octet de donnée[6].

2.4 – À propos du nombre de paquets envoyés

Rappelons que n_m désigne le nombre maximal de destinataires autorisé par paquet GXcast. Le nombre de paquets générés pour envoyer un volume D de données à un ensemble de n destinataires va dépendre de cette valeur. On s'intéresse alors à la fonction f^* qui va associer à un nombre de destinataires n et à un certain volume de données D la valeur n_m permettant d'envoyer un nombre minimal de paquets.
À titre d'exemple, considérons un ensemble de n=100 destinataires et une source désirant leur transmettre un volume D=300 octets de données. Dans le cas du protocole Xcast, la quantité de données disponibles par paquet est égale à M-E-IP*n, soit 140 octets. Xcast va donc nécessiter l'envoi de trois paquets pour transmettre la totalité des données. Plaçons nous à présent dans le cas du protocole GXcast et supposons que n_m soit égal à 50. La quantité de données disponibles par paquets est égale à M-E-IP*n_m, soit 340 octets. Chaque paquet GXcast peut donc contenir la totalité des données. Il va cependant être nécessaire d'envoyer deux paquets : le premier sera à destination des 50 premiers membres, le second à destination des 50 restants.
La valeur optimale $n_m=f^*(n,D)$ peut être calculée comme suit :

$$f^*(n,D)=\arg\min_{nm\in[1,134]}\left\{\left\lceil n/n_m \right\rceil \left\lceil D/(M-E-IP.n_m) \right\rceil\right\}$$

$\left\lceil n/n_m \right\rceil$ représente le nombre de paquets à envoyer pour atteindre les n destinataires en limitant à n_m le nombre de destinataires de chaque paquet. $\left\lceil D/(M-E-IP.n_m) \right\rceil$ est la quantité de données utiles qu'il est possible de placer dans un paquet comportant au plus n_m destinataires.
Cette définition de f^* présente deux désavantages principaux :

- le calcul associé est coûteux puisqu'il faut effectuer le calcul du minimum sur un ensemble de 134 valeurs,
- le calcul dépend de D.

Pour résoudre ces deux inconvénients, nous nous proposons de chercher une fonction f telle que :

- f est simple à calculer,
- f ne dépend que de n,
- pour tout D, $f(n)$ s'approche de $f^*(n,D)$.

L'écart entre $f(n)$ et $f^*(n,D)$ sera alors, dans le pire des cas, le surcoût en nombre de messages par rapport à l'optimal. La fonction $\left\lceil n/n_m \right\rceil \left\lceil D/(M-E-IP.n_m) \right\rceil$ admet un minimum

[3] Rappelons que l'en-tête Xcast est, quant-à lui, codé sur 12 octets.

[4] Elle est de M=1280 octets pour le protocole IPv6.

[5] Cette taille est de IP=16 octets pour le protocole IPv6.

[6] Pour IPv6, on aboutit à une valeur *n*max=76. En effet, pour IPv6 on a E=56.

pour $n_m=n\max/2$. On propose donc la fonction suivante : $f(n)=n\max/2$ [7]. Le paragraphe suivant étudie le comportement de cette fonction selon différents critères.

3 – ÉVALUATION ET SIMULATION DU PROTOCOLE GXCAST

3.1 – Le nombre de paquets générés

Le nombre de paquets générés par GXcast va dépendre des trois paramètres n, D et n_m. Le choix pour n_m de la valeur $f(n)$ a été justifié dans le paragraphe 2.4 (en tant que minimum). La figure 5 présente le surcoût de GXcast par rapport à un protocole *multicast* traditionnel. Rappelons qu'un protocole *multicast* traditionnel envoie $\lceil D/(M-E) \rceil$ paquets pour faire parvenir D octets de données à un groupe n de membres[8]. On remarque tout d'abord que le surcoût engendré par GXcast est pratiquement indépendant de la taille D de données à émettre. On remarque de plus que ce surcoût est linéaire en fonction de n : pour un groupe constitué de 150 membres, il sera nécessaire d'envoyer environ 5 fois plus de paquets avec GXcast qu'avec un protocole traditionnel. Pour un groupe de 300 membres, il sera nécessaire d'envoyer 10 fois plus de paquets.

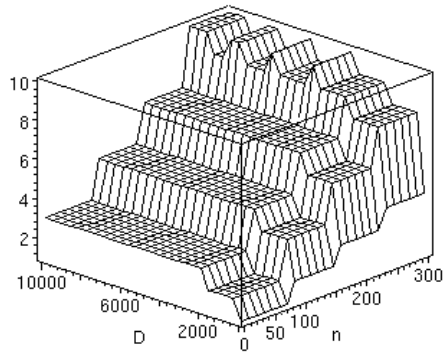

Figure 5 – Surcoût de GXcast par rapport à un protocole *multicast* traditionnel.

Pour transmettre à respectivement 150 et 300 destinataires un message de 1000 octets, un protocole basé sur des messages *unicast* aurait nécessité 150 et 300 paquets, soit respectivement 150 et 300 fois plus qu'un protocole *multicast* traditionnel[9].

3.2 – Le temps de traitement global

Le temps de traitement global d'un protocole est la somme des temps de traitement des paquets qu'il envoie. Dans ce paragraphe, nous étudions l'évolution du temps de traitement global en fonction du choix du paramètre n_m. Nous nous intéresserons plus particulièrement au cas où $n_m=f(n)$. Pour un paquet GXcast comportant nm destinataires, le temps de traitement est approché par $t_{nm}=\tau+n_m.\tau'$, où τ représente le temps de traitement des en-têtes IP et GXcast et τ' le temps de traitement associé à une entrée dans la liste des destinataires (recherche dans la table de routage, création des paquets par interface de sortie, etc.). On a donc le temps de traitement global défini par $t_G(n_m)=\lceil n/n_m \rceil t_{n_m} \approx \frac{n.\tau}{n_m}+n.\tau'$. La fonction $t_G(n_m)$ est une fonction strictement décroissante. Son minimum est donc en $t_G(n\max)$. Cependant, choisir $n_m=n\max$ est irréaliste. En effet, la formulation de $t_G(n_m)$ ne prend pas en compte les données à envoyer.

3.3 - Le délai supplémentaire moyen

Le délai d'un paquet est le temps qu'il lui faut pour atteindre sa destination à partir de la source. Il dépend en partie du nombre de paquets dans la file d'attente des routeurs et du temps de traitement de chacun d'eux. La différence de délai perçue par l'utilisateur final (c'est-à-dire le destinataire) joue un rôle important dans le choix des protocoles. Afin de mesurer le délai supplémentaire introduit au niveau du récepteur par l'utilisation de différentes valeurs de n_m, le protocole GXcast a été implémenté sous la plate-forme de simulation NS (Fall et Varadhan, 2001). Le réseau de test est un réseau aléatoire généré à l'aide de GT-ITM, algorithme proposé par Zegura *et al* (1996). Il possède 100 nœuds reliés entre eux par des liens bidirectionnels de capacité 20 Mb/s. On ne considère qu'un seul groupe *multicast* comportant une source et un ensemble de 18 destinataires qui s'abonnent et se désabonnent à des instants quelconques. Le nombre maximal de destinataires par paquet GXcast a été fixé à 1, 3, 6, 9 et 18 afin de pouvoir observer l'évolution du délai. La figure 6 présente le délai supplémentaire moyen par rapport à la valeur 18 de n_m. On remarque que le pourcentage de délai ajouté est assez faible, puisqu'il n'est que de 30% dans le pire des cas. Le délai ajouté pour la valeur $n_m=n/2=9$ est très faible puisqu'il ne représente que 5% du délai optimal.

[7] Notons que $f(n)$ ne dépend pas de n.

[8] On considère ici que la taille de l'en-tête associé au protocole *multicast* est aussi de E.

[9] Rappelons que Xcast est incapable de gérer un groupe ayant plus de 134 membres.

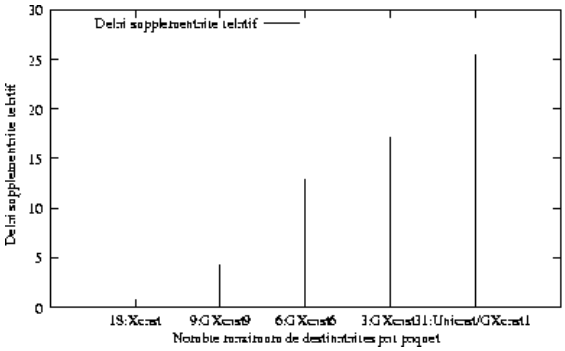

Figure 6 – Délai supplémentaire moyen en fonction de n_m.

4 – CONCLUSIONS

Les protocoles Xcast et Xcast+ permettent de gérer efficacement un grand nombre de petits groupes *multicast*. Une partie de leurs inconvénients vient de leur incapacité à gérer la fragmentation des paquets. De plus, une limite forte existe sur le nombre de membres du groupe *multicast*. Dans ce papier, nous avons proposé une extension à ces protocoles nommée GXcast. GXcast permet de résoudre le problème de la fragmentation et optimise des critères comme le nombre de paquets envoyés ou le temps de traitement des paquets au sein des routeurs. Une étude ultérieure effectuera l'analyse de GXcast en fonction de la topologie du réseau. À l'issue de l'étude de ce protocole, il semble néanmoins que GXcast puisse gérer un grand nombre de groupes de taille moyenne, dont les membres sont disséminés sur plusieurs centaines de sous-réseaux.

BIBLIOGRAPHIE

Boivie, R., Feldman, N., Imai, Y., Livens, W., Ooms, D., Paridaens, O. (2000), *Explicit multicast (Xcast) basic specification*, IETF Internet Draft.

Myung-KI, S., Yong-Jin, K., Ki-Shik, P., Sang-Ha, K. (2001), *Explicit multicast extension (Xcast+) for efficient multicast packet delivery*, ETRI Journal, vol 23, n° 4

Fall, K., Varadhan, K. (2001), *The NS Manual*, UC Berkeley, LBL, USC/ISI, and Xerox PARC.

Reliable Multicast Transport Working Group Web Site (2003), http://www.ietf.org/html.charters/rmt-charter.html

Ooms, D. (2000), *Taxonomy of Xcast/SGM proposals*, IETF Internet Draft.

Deering, S., Hares, S., Perkins, C., Perlman, R. (2000), *Overview of the 1998 IAB Routing Workshop*, IETF RFC 2902.

Cain, B., Deering, S., Kouvelas, I., Fenner, B., Thyagarajan, A. (2002), *Internet Group Management Protocol, Version 3*, IETF RFC 3376.

Boudani, A., Cousin, B. (2003), *SEM: a new small group multicast routing protocol*, Proceedings of the 10th International Conference on Telecommunications (ICT).

Zegura, E., Calvert, K., Bhattacharjee, S. (1996), *How to model an internetwork*, Proceedings of INFOCOM.