SETIT 2007
4rth International Conference: **S**ciences of **E**lectronic,
Technologies of **I**nformation and **T**elecommunications
March 25-29, 2007 – TUNISIA

Analyse de la latence de recouvrement des pertes dans les protocoles multicast fiables de type APES

Lakhdar DERDOURI 1*, Cong Duc PHAM 2** and Doumia BENDALI-AMOR 3*

*Laboratoire LIRE, Université de Mentouri, 25000 Constantine, Algérie.
derdouril@yahoo.fr
bendaliamor-kh@caramail.com

**LIUPPA, Université de Pau, 64013 Pau Cedex, France.
congduc.pham@univ-pau.fr

Résumé: Fournir une délivrance multicast fiable et efficace pour des applications de dissémination de données à grande échelle est un défi, particulièrement lorsque l'application nécessite un délai de livraison très court et une capacité en terme de bande passante assez élevée. La combinaison d'une approche de recouvrement local basée sur des serveurs de réparation avec celle utilisant des codes FEC a donné naissance à une nouvelle classe de protocoles multicast fiables appelée APES « Active Parity Encoding Service ». Cet article réalise une étude comparative entre les protocoles multicast fiables appartenant à cette nouvelle classe en terme de latence de recouvrement des pertes. Notre étude montre qu'une approche de type Get-Repair Store-Repair, en plus de la réduction en espace de stockage au niveau des serveurs de réparation, apporte un gain considérable en terme de latence de recouvrement des pertes et contribue ainsi à l'amélioration de la fiabilité multcast temps réel.
Mots-clés: Multicast, fiabilité, codes FEC, services actifs

.

INTRODUCTION

Fournir une délivrance multicast fiable et efficace pour des applications de dissémination de données à grande échelle est un défi, particulièrement lorsque l'application nécessite un délai de livraison très court et une capacité en terme de bande passante assez élevée.

Plusieurs protocoles ont été élaborés pour résoudre le problème de la fiabilité dans les réseaux à délivrance dite *best effort* tel que l'Internet où la perte de paquets est loin d'être rare. Les protocoles de multicast fiables, *reliable multicast* (RM), traitent ce problème en imposant un compromis entre le délai d'acheminement et la capacité en bande passante. Les approches traditionnelles fonctionnent de bout-en-bout avec retransmission automatique (Automatic Repeat Request, ARQ) des paquets perdus à partir de la source. Plusieurs mécanismes comme le filtrage des messages de contrôle ou encore les retransmissions restreintes permettent d'éviter la surcharge de la source et des récepteurs en permettant dans une certaine mesure le passage à l'échelle en présence de groupes importants [FLO97].

D'autres propositions utilisent de manière conjointe la retransmission avec des techniques de codes correcteurs (*Forward Error Correction*, FEC) pour produire des paquets de réparation afin de répondre à des pertes éventuelles de paquets de données au niveau des récepteurs. Les récepteurs, dans cette approche, sont obligés d'effectuer des opérations de codage et de décodages supplémentaires pour pouvoir utiliser ou reproduire les données originales [BYE99, BYE98, RUB01, LIC99, NON98a].

Une certaine catégorie de protocoles RM nécessite l'assistance des routeurs comme ceux basés sur le recouvrement local des pertes en utilisant les serveurs/services de réparation (*Repair Services*, RS). Ces derniers sont placés à des points stratégiques du réseau, interceptent les paquets de données et les stockent au niveau de leurs caches pour assurer le recouvrement des pertes localement tout en traitant les accusés de réception négatifs (NAK) transmis par les récepteurs en cas de pertes [KAS98, LEH98, RUB00, SRI99].

Les approches FEC/ARQ et RS permettent de réduire les besoins en termes de bande passante et la latence de recouvrement des pertes des protocoles RM. Le mariage de ces deux a donné naissance à une

nouvelle classe de protocoles multicast fiables appelée *Active Parity Encoding Services* (APES) [KER98, NON98]. Plusieurs variantes des protocoles APES sont apparues dans le but d'améliorer les performances de cette classe de protocoles. Récemment deux nouveaux protocoles APES ont été proposés dans la littérature : BRSR (*Built-Repairs Store-Repairs*) et GRSR (*Get-Repairs Store-Repairs*) [RUB04]. Une étude comparative met en valeur les améliorations, en terme de réduction en espace de stockage au niveau des serveurs de réparation et en terme de bande passante consommée, apportées par ces deux nouveaux protocoles APES [RUB04]. Les deux protocoles ont été comparés à une autre variante plus ancienne des protocoles APES : SDBR (Store-Data Built-Repairs) [RUB04, RUB98]. Cette étude comparative a montré qu'effectivement les protocoles BRSR et GRSR permettent une réduction considérable en espace de stockage au niveau des serveurs de réparation tout en maintenant la consommation en bande passante relativement identique à celui du protocole SDBR.

Dans cet article, nous proposons d'élargir l'étude comparative pour englober l'aspect latence de recouvrement des pertes dans un bloc de données envoyé par une source jusqu'à la réception du bloc par tous les récepteurs. Notre étude montre que la latence de recouvrement des pertes d'un paquet de données dans les trois protocoles croît linéairement en fonction du temps de codage et de décodage, et aussi en fonction du taux de pertes. Pour un temps de codage assez élevé, GRSR présente les meilleures performances. Par contre, pour un temps de codage raisonnable c'est SDBR qui minimise le temps pour la transmission d'un bloc de données.

Le reste de l'article est organisé comme suit : la section 2 donne une brève description du fonctionnement des trois protocoles APES (SDBR, BRSR et GRSR). La section 3 présente le modèle de réseau sur lequel se basera notre étude. Les hypothèses de l'étude comparative sont exposées dans la section 4. La section 5 est consacrée à l'étude comparative des trois protocoles en terme de latence de recouvrement des pertes dans un bloc de données. La section 6 clôturera l'article par une conclusion et des perspectives.

1. Description des variantes de protocoles APES

Dans cette section nous présentons les trois variantes de la classe de protocoles APES. Les protocoles BRSR et GRSR ont été proposés dans un objectif d'améliorer les performances des protocoles multicast fiable en termes de bande passante et de capacité de stockage au niveau de serveurs de réparation [RUB04, RUB98]. Ces deux protocoles ont été comparés à une variante plus ancienne SDBR pour mettre en évidence les améliorations qu'ils apportent. Pour pouvoir cerner les différences qui existent entre les trois protocoles nous présentons une description du comportement de chaque protocole. De manière générale**, c**es variantes diffèrent dans comment et quand générer les paquets de réparation au niveau des serveurs de réparation, et dans le choix des paquets d'un bloc à stocker au niveau du cache de ces derniers. Dans notre description, nous exposerons les tâches affectées aux serveurs de réparation pour chaque protocole afin d'assurer la fiabilité des transmissions des blocs de données destinés aux récepteurs dans leurs domaines de réparation.

1.1. Le protocole SDBR : (Store-Data Built-Repairs)

Dans ce protocole, la source transmet une combinaison de paquets de données et de réparation dans des blocs de k paquets à tous les serveurs de réparation concernés. Une fois qu'un serveur de réparation a bien reçu les k paquets d'un bloc, il les décode pour restaurer les paquets de données originaux qu'il met dans un cache. Il transmet ensuite les paquets (non décodés) aux récepteurs.

Au niveau des récepteurs, les paquets source reçus sont décodés par un décodeur FEC afin de restaurer les paquets de données originaux. Si un récepteur a besoin de paquets de réparation supplémentaires dans le but de réparer les pertes d'un bloc, il envoie un NAK au serveur de réparation en indiquant le nombre de paquets FEC supplémentaires qui sont nécessaire. A la réception du NAK, un serveur de réparation procède lui même, grâce à un encodeur FEC, à la génération des nouveaux paquets de réparation demandés, et les envoie uniquement vers les récepteurs de son domaine (*subcast*). Comme les paquets FEC sont distincts, tout récepteur peut utiliser les paquets de réparation obtenus pour réparer n'importe quelle perte survenue dans le bloc de k paquets.

1.2. Le protocole BRSR: (Built-Repairs Store-Repairs)

BRSR est une autre variante de la classe APES. Le serveur de réparation fixe à l'avance un nombre b de paquets de réparation à générer par bloc. Dans ce cas, les paquets de données reçus par le serveur de réparation sont introduits dans un encodeur FEC[1] [NON97, RIZ97] pour générer les b paquets de réparation qu'il va stocker (ce type de codeur permet un codage à la volée). La valeur de b est comprise entre 0 et la taille k, en paquets, des blocs ($0 \leq b \leq k$). Ces b paquets occupent l'espace du buffer cache dès la réception du premier paquet source et ne peuvent être utilisés pour la réparation que lorsque le $k^{ième}$ paquet source est introduit dans le codeur FEC du serveur de réparation.

Dans le meilleur des cas, lorsque tous les récepteurs perdent moins de b paquets source, les b paquets de réparation stockés au niveau des serveurs

[1] Deux types de codage FEC sont possibles : Vandermonde-based Reed-Solomon codes [RIZ97] ou Cauchy-based Reed-Solomon [NON97].

de réparation suffisent pour faire le recouvrement de n'importe quelle perte signalée par les récepteurs. Le serveur de réparation *subcast* les paquets de réparation demandés par les récepteurs de son domaine. Si un des récepteurs perd plus de *b* paquets source, alors le serveur de réparation doit obtenir les paquets de réparation supplémentaires en envoyant à la source un NAK indiquant le nombre de paquets FEC manquant. Le serveur de réparation est en effet incapable de générer lui même des paquets FEC supplémentaires (les paquets de données n'étant plus disponibles à son niveau).

Après avoir reçu ces paquets additionnels, il les *subcas*t à tous les récepteurs de son domaine. Ces protocoles supposent que la source est capable de transmettre des paquets de réparation supplémentaires à n'importe quel moment et sans perte. Les innovations apportées par ce protocole sont :

Au lieu de faire le cache des paquets de données, seul les paquets de réparation sont stockés au niveau des serveurs de réparation.

Il n'est pas nécessaire de générer de nouveau paquets de réparation à chaque perte : il suffit de transmettre les réparations stockées.

1.3. Le protocole GRSR : (Get-Repairs Store-Repairs)

De même que dans BRSR, GRSR fixe un nombre b de paquets de réparation à générer par bloc. Cependant, à la différence de BRSR, les serveurs de réparation n'effectuent pas d'opérations de codage-décodage[2]. Ils se procurent les b paquets de réparation en les demandant de la source pour ensuite les stocker dans leur cache dès leur arrivée.

Les paquets de réparation sont construits au niveau de la source et sont transmis directement à la suite des k paquets du bloc pour lequel ils ont été générés. Une fois les b paquets reçus et stockés, les serveurs de réparation se comportent comme dans BRSR pour le reste de la transmission.

2. Le modèle de réseau

Le modèle réseau sur lequel se basera notre étude est une topologie de réseau ayant une structure d'un arbre multicast qui admet la source comme racine, les serveurs de réparation comme nœuds intermédiaires et les récepteurs à l'extrémité de l'arbre. Les récepteurs sont répartis selon une topologie en *m* domaines de réparation $D_1, \ldots, D_m$ [RUB04]. La taille d'un domaine de réparation est définie comme étant le nombre de récepteurs dans ce domaine $|D_i|$. En réalité, les récepteurs peuvent être soit des LAN's contenant une ou plusieurs applications réceptrices soit tout simplement d'autres serveurs de réparation, ces derniers étant responsable de domaines de réparation d'un niveau hiérarchique inférieure. Dans notre étude, nous considèreront une hiérarchie à un seul niveau (figure 1.) où les serveurs de réparation sont placés à des points stratégiques dans le réseau (entre la source et un domaine de réparation). Cela permet aux serveurs de réparation d'intercepter tout paquet envoyé de la source vers les récepteurs de son domaine pour assurer le recouvrement local des pertes. Ils sont aussi capables de diffuser de manière restreinte (*subcast*) des paquets vers leur domaine de réparation de façon à ce que les données n'atteignent que les récepteurs de leur domaine.

Les protocoles APES transmettent des paquets de réparation basés sur le codage FEC à la place des retransmissions. La source regroupe les paquets de données dans des blocs de taille ***k,*** et les introduit dans un codeur FEC pour générer les paquets de réparation. Le nombre de réparation qui peut être généré pour un bloc est fini et suffisamment large pour assurer le recouvrement des pertes éventuelles.

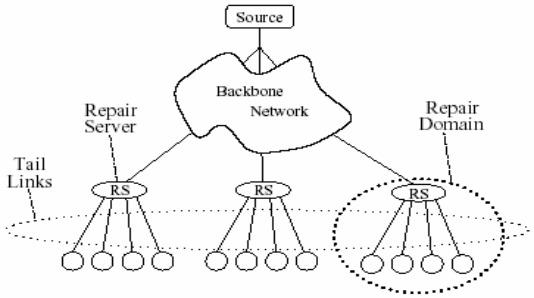

Figure 1 *Le modèle réseau des protocoles APES.*

La source est supposée pouvoir reproduire indéfiniment des réparations supplémentaires si celles-ci sont nécessaires. Les réparations, générées pour un bloc de *k* paquets de données, sont supposées appartenir à ce bloc. Les *k* premiers paquets d'un bloc (combinaison de données et de réparation) sont appelés : paquets source. En supposant que la transmission entre la source et les serveurs de réparation est fiable, les *k* paquets source sont tout simplement les paquets de données originaux. N'importe laquelle des entités (récepteur ou serveurs de réparation) munie d'un décodeur FEC est capable de restaurer les *k* paquets de données originaux quelle que soit la combinaison de *k* paquets reçus (données et réparation) appartenant à ce bloc. Les protocoles basés sur le code FEC doivent s'assurer que chaque récepteur a reçu au moins *k* paquets par bloc pour assurer la fiabilité. Le fait que différents paquets de données puissent être perdus et qu'un même ensemble de réparation puisse être utilisé pour les réparer permet de réduire le nombre de transmissions nécessaires au recouvrement. Cette réduction a été observée dans un environnement de bout en bout [NON97].

Les serveurs de réparation APES ont pour objectif

[2] L'opération de décodage des paquets de données n'est pas inhérente au protocole SDBR. Enl'éliminant, les récepteurs doivent, dans ce cas, effectuer un algorithme de décodage récursif pour pouvoir restaurer les paquets de données originaux, comme c'est le cas pour les récepteurs des protocoles BRSR et GRSR.

de s'assurer que chaque récepteur appartenant à son domaine de réparation a bien reçu au moins *k* paquets distinct par bloc. Pour accomplir cela, chaque protocole dérivant de la classe APES détermine quelles sont les tâches effectuées par les serveurs de réparation.

3. Les hypothèses de l'étude comparative

L'étude consiste à comparer les trois variantes de protocoles APES : SDBR, BRSR et GRSR en terme de latence de recouvrement des pertes dans un bloc de paquets émis par une source et reçu par l'ensemble des récepteurs. Celle-ci nous permettra de déterminer, parmi les trois variantes de protocoles, quelle est la variante qui s'adapte le plus au multicast fiable temps réel.

Notre étude est basée sur un modèle de réseau hiérarchique à un seul niveau de serveurs de réparation. L'étude des protocoles peut être facilement étendue à un modèle hiérarchique à plusieurs niveaux : chaque serveur de réparation de niveau hiérarchique *i* fournit ses services de réparation à un ensemble de serveurs de réparation du niveau *i*+1 de la hiérarchie, ($1 \le i \le n$ où *n* est la profondeur de la hiérarchie). L'intérêt du choix de ce modèle de réseau est que chaque domaine de réparation peut être analysé séparément des autres. Ainsi, on se basera sur la topologie de la figure 2.

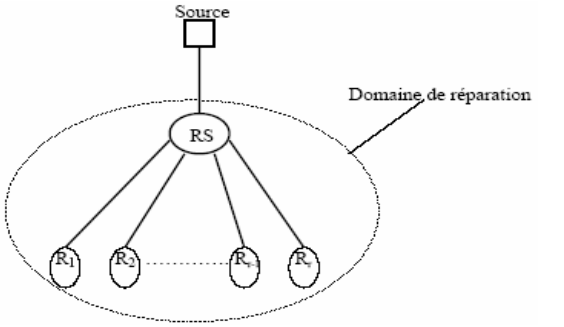

Figure 2 *Le modèle hiérarchique à un seul niveau***.**

Nous supposons quelques hypothèses essentielles à la réalisation de notre étude. Posons *r* comme étant le nombre de récepteurs du domaine de réparation. Ce nombre désigne la taille du domaine. Chaque récepteur est soumis à une probabilité de perte, *p*, entre le serveur de réparation et lui même. Les liaisons entre le serveur de réparation et ses récepteurs sont identiques. On suppose aussi que la probabilité de perdre des paquets entre la source et le serveur de réparation est négligeable. Le serveur de réparation affecte aux *k* paquets de source, un numéro de séquence allant de 1 à *k*. Les paquets de réparation ont des numéros de séquence supérieure à *k*.

Un récepteur ne peut détecter une perte qu'après avoir reçu les *k* paquets source. Un récepteur ayant perdu *m* des *k* paquets source demande au serveur de réparation la transmission des paquets de réparation, *k*+1,....., *k*+*m*. En pratique, un récepteur exigeant *m* réparations et ayant perdu l'un d'entre eux peut effectivement utiliser un paquet de réparation numéroté : *k*+*m'* tel que *m'*>*m* à la place des paquets perdus.

D'une part, $\gamma_j^k(p)$ est défini comme étant la probabilité de perdre exactement *j* paquets des *k* paquets source :

$$\gamma_j^k(p) = \binom{k}{j} p^j (1-p)^{k-j} \quad (1)$$

D'autre part, ϕ_i est la probabilité qu'un récepteur exige la transmission du $i^{ième}$ paquet. Pour $i \le k$, $\phi_i = 1$, car les *k* premiers paquets sont les paquets source qu'ils sont sûrement transmis par le serveur de réparation. Dans le cas où $i>k$, ϕ_i représente la probabilité qu'un récepteur reçoit moins de $2k-i+1$ paquets source donc il a besoin d'au moins $i-k-1$ paquets de réparation numérotés de $k+1$ à $i-1$.

$$\phi_i = \begin{cases} 1, & i \le k, \\ 1 - \sum_{j=0}^{i-k-1} \gamma_j^k(p), & k < i \le 2k, \\ 0, & i > 2k \end{cases} \quad (2)$$

On définit $q_i(j)$ la probabilité qu'au moins un récepteur a besoin que le paquet *i* lui soit transmis plus de *j* fois.

$$q_i(p) = \begin{cases} 1, & i \le k,\ j = 0, \\ 0, & i \le k,\ j > 0 \ \text{or}\ i > 2k, \\ 1 - (1 - \phi_i p^j)^r, & k < i \le 2k,\ j \ge 0. \end{cases} \quad (3)$$

Quatre métriques sont utilisées pour déterminer les performances des trois protocoles :

- Le nombre moyen de paquets transmis du serveur de réparation aux récepteurs de son domaine de réparation.
- Le nombre moyen de paquets transmis de la source vers le serveur de réparation.
- Le nombre moyen de paquets stockés au niveau du serveur de réparation, appelé "taille du buffer cache".
- Le temps moyen nécessaire pour transmettre de façon fiable un bloc de *k* paquets.

4. Analyse de la latence de recouvrement des pertes

Dans cette section nous allons déterminer le temps nécessaire à la transmission d'un bloc de k paquets de la source jusqu'à sa réception, d'une manière fiable, par l'ensemble des récepteurs et ce en utilisant les trois variantes de protocoles étudiés. On définit ainsi TSDBR, TBRSR et TGRSR comme étant le temps total nécessaire à la transmission d'un bloc de k paquets en appliquant respectivement les protocoles SDBR, BRSR et GRSR. L'évaluation a été réalisée en utilisant le simulateur ns2 et en fixant un temps de codage et de décodage FEC (qui sera identique pour les 3 protocoles) au niveau de chaque entité. Avant de présenter les résultats de simulation, nous allons rappeler les opérations de codage et de décodage effectuées par chaque entité et cela pour chaque variante de protocoles :

SDBR	
Serveur	Décodage des k paquets source pour extraire les paquets de données originaux. Codage des données stockées dans le buffer pour générer les paquets de réparation demandés si une demande de réparation supplémentaire arrive.
Récepteur	Décodage des k paquets source pour restaurer les paquets de données originaux

BRSR	
Serveur	Décodage des k paquets source pour générer les b paquets de réparation à stocker.
Récepteur	Décodage des k paquets source pour restaurer les paquets de données originaux

GRSR	
Serveur	Aucune opération.
Récepteur	Décodage des k paquets source pour restaurer les paquets de données originaux

Notre première étude consiste à déterminer si le choix de la taille d'un bloc a une influence sur le temps de transmission pour les trois protocoles. La figure 3. représente T_{SDBR}, T_{BRSR} et T_{GRSR} en fonction de la taille d'un bloc, *k*. La taille du domaine est r =8, et la probabilité de perte est p=0.05

En observant la courbe, on peut constater que le temps total de transmission est une fonction linéaire de la taille du bloc. En effet, étant donné que les pertes ne seront détectées qu'à la réception des *k* paquets source au niveau du récepteur et que le temps de transfert d'un bloc, d'une manière fiable, dépend de la taille de celui-ci, plus la taille du bloc est petite plus le temps nécessaire à la détection et à la réparation des pertes est court**.** Pour des tailles de blocs inférieures à 40 paquets, les trois protocoles nécessitent un temps de transmission presque identique. Pour des tailles de bloc plus importantes, SDBR nécessite moins de temps par rapport aux autres protocoles car il procède à un recouvrement local quelque soit le nombre de pertes survenu au niveau de ses récepteurs. Par contre, BRSR nécessite beaucoup plus de temps car il effectue plusieurs opérations de codage et décodage FEC et nécessite une période de temps additionnel pour informer la source (dans le cas où il a besoin de réparation supplémentaire de la source).

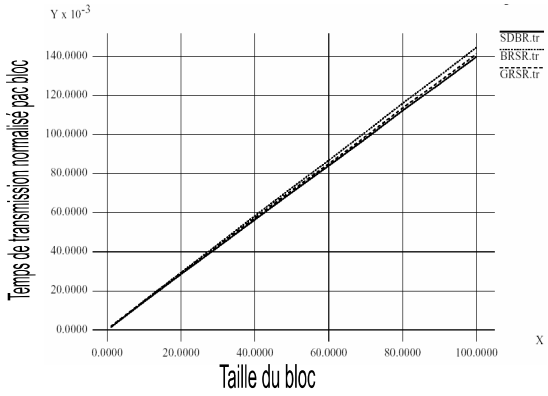

Figure 3 *Temps de transmission en fonction de la taille du bloc***.**

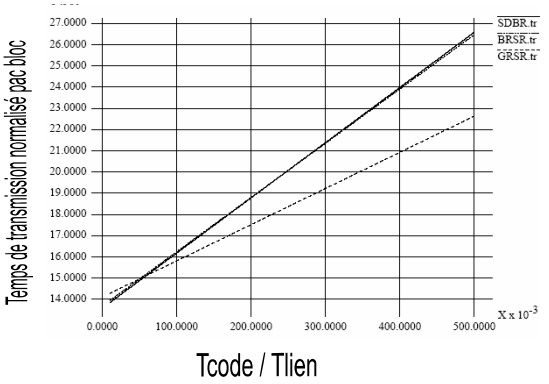

Figure 4 *Temps de transmission en fonction du temps de codage-décodage***.**

En se basant sur le graphe obtenu, on constate que le temps de transmission des trois protocoles croît linéairement avec le temps du codage FEC. Pour un temps de codage réduit, BRSR est meilleur que GRSR en terme de temps de transmission. Au-delà d'une certaine valeur du temps de codage, GRSR devient meilleur que BRSR et SDBR car la source GRSR envoie des paquets de réparation supplémentaire directement après les *k* paquets de source évitant ainsi au serveur de réparation de coûteuses opérations de codage.

La figure 5. représente T_{SDBR} et T_{GRSR} en fonction du taux de perte, p. Dans ce cas nous fixons r=8,

T_{code}/T_{lien}=0.05 et k=10. On voit que que le taux de pertes joue un rôle important dans la détermination de l'apport des protocoles SDBR et GRSR en terme de délai de transmission d'un bloc. Le temps de transmission des deux protocoles croît en fonction du taux de pertes au niveau des récepteurs. Plus il y a des pertes au niveau du réseau plus il y a d'opérations de codage/décodage et d'accès au serveur de réparation.

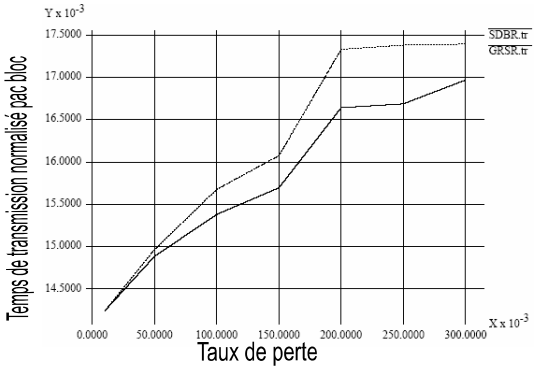

Figure 5 *Temps de transmission en fonction du taux de perte***.**

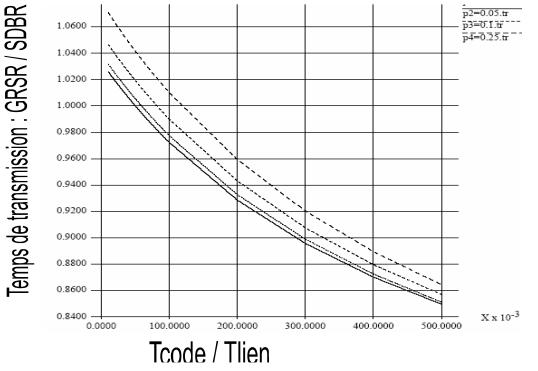

Figure 6 *Étude du rapport GRSR / SDBR en terme de temps de transmission en fonction du temps de codage pour différents taux de perte***.**

La figure 6. représente le rapport en terme de temps de transmission entre GRSR et SDBR en fonction du rapport T_{code}/T_{lien}. Grâce à ce graphe on remarque que plus le temps de codage tend vers le temps nécessaire au transfert des données du serveur de réparation aux récepteurs plus le temps de transmission du protocole GRSR s'éloigne du temps de transmission du protocole SDBR. Le temps de transmission de GRSR est de plus en plus réduit par rapport à celui du protocole SDBR. L'augmentation du temps de codage affecte plus nettement le temps de transmission du protocole SDBR que de celui du protocole GRSR car le serveur de réparation du protocole SDBR procède à un codage de données à chaque fois qu'une perte est signalée alors que sous GRSR le serveur de réparation n'effectue pas d'opérations de codage.

5. Conclusion

L'analyse effectuée sur la latence de recouvrement des pertes nous permet de déduire que la probabilité des pertes au niveau des récepteurs et le temps de codage /décodage FEC déterminent si un protocole peut être candidat à un multicast fiable temps réel ou non. Notre étude montre que le protocole GRSR, en plus de la réduction en espace de stockage au niveau des serveurs de réparation, apporte un gain considérable en terme de latence de recouvrement des pertes et contribue ainsi à l'amélioration de la fiabilité multicast temps réel. Pour un temps de codage assez élevé, GRSR présente les meilleures performances. Par contre, pour un temps de codage raisonnable c'est SDBR qui minimise le temps pour la transmission d'un bloc de données.

Dans ce papier, nous avons volontairement étudié un modèle simple de réseau hiérarchique (1 niveau) pour faciliter l'interprétation des résultats. Nous souhaitons généraliser les résultats de notre étude pour un réseau à *n* niveaux hiérarchiques où les récepteurs peuvent avoir des taux de pertes différents dans des travaux futurs. Une autre priorité pour nous est d'étudier la relation entre la taille du buffer cache au niveau des réparateurs et la latence de recouvrement.

REFERENCES

[BYE 99] J. Byers, M. Luby, and M. Mitzenmacher, Accessing multiple mirror sites in parallel: using Tornado codes to speed up downloads, *In Proceedings of IEEE INFOCOM'99*, New York, NY, March 1999.

[BYE 98] J. Byers, M. Luby, M. Mitzenmacher, A. Rege, A digital fontain approach to reliable distribution of bulkdata, *In Proceedings of SIGCOMM'98*, Vancouver, Canada, Septembre 1998.

[FLO 97] S. Floyd, V. Jacobson, C. G. Liu, S. McCanne, and L. Zhang. A reliable multicast framework for light-weight sessions and application level framing. *IEEE/ACM Transactions on Networking*, 5(6), 1997.

[KAS 98] S. Kasera, J. Kurose, D. Towsley, A comparision of server-based and receiver-based local recovery approaches for scalable reliable multicast, *In Proceedings of INFO-COM'98*, San Francisco, CA, March 1998.

[KER 98] R. Kermode, Scoped hybrid automatic repeat request with forward error correction (SHARQFEC), *In Proceedings of SIGCOMM'98*, Vancouver, Canada, September 1998.

[LEH 98] L.H. Lehman, S.J. Garland, D.L. Tennenhouse, Active reliable multicast, *In Proceedings of INFOCOM'98*, San Francisco, CA, March 1998.

[LIC 99] D. Li, D.R. Cheriton, Evaluating the utility of FEC with Reliable Multicast, *Computer Science*, Stanford University, 1999.

[NON 98a] J. Nonnenmacher, E. Bircsack and D. Towsley, Parity-Based Loss Recovery for reliable multicast transmission, *IEEE/ACM Transactions on Networking*, Aug. 1998. vol.6, no.4, p. 349-361

[NON 98] J. Nonnenmacher, M. Lacher, M. Jung, E. Biersack, G. Carle, How bad is reliable multicast without local recovery, *In Proceedings of INFOCOM'98*, San Francisco, CA, March 1998.

[NON 97] J. Nonnenmacher, E. Bircsack and D. Towsley, Parity-Based Loss Recovery for reliable multicast transmission, ACM SIGCOMM '97, Sept .1997 Cannes, France, pp. 289-300.

[RIZ 97] L. Rizzo. Effective Erasure codes for Reliable Computer communication Protocols. Computer Communication Review, April 1997.

[RUB 01] D. Rubenstein, J. Kurose, and D. Towsley, A Study of Proactive Hybrid FEC/ARQ and Scalable Feedback Techniques for reliable Real Time Multicast, *Computer Communications* 24 (5-6) (March 2001) 563-574.

[RUB 04] D. Rubenstein, S. Kasera, D. Towsley, and J.Kurose, Improving Reliable Multicast Using Active Parity Encoding Services (APES), *Computer Networks* 44 (2004) 63-78. Responsible editor: K. Park.

[RUB 00] D. Rubenstein, N.F. Maxemchuk, D. Shur, Acentralized approach to network repair service for multicast streaming media, *In Proceedings of NOSSDAV'00*, Chapel Hill, NC, June 2000.

[RUB 98] D. Rubenstein, S. Kasera, D. Towsley, and J. Kurose, Improving Reliable Multicast Using Active Parity Encoding Services (APES), Technical Report 98-79. Computer Science Departement. University of Massachusetts at Amherst. July, 1998.

[SRI 99] K. Sripanidkulchai, A.C. Meyers, H. Zhang , A third-party value-added network service approche to realible multicast, *In Proceedings of ACM SIGMETRICS'99*, Atlanta, GA, May 1999.