REPUBLIQUE ALGERIENNE DEMOCRATIQUE ET POPULAIRE

MINISTERE DE L'ENSEIGNEMENT SUPERIEUR ET DE LA RECHERCHE SCIENTIFIQUE
UNIVERSITE MENTOURI CONSTANTINE
FACULTE DES SCIENCES DE L'INGENIEUR
DEPARTEMENT D'INFORMATIQUE

N° d'ordre: 43/TS/2009
N° Série: 03/INF/2009

THESE

Présentée pour obtenir le grade de

DOCTORAT EN SCIENCES EN INFORMATIQUE

Une Approche Hybride pour le Transport Multicast Fiable dans un Environnement Actif

Présentée par *:* **Dirigée par *:***

Mr. Derdouri Lakhdar Pr. Benmohammed Mohamed. Université Mentouri de Constantine.

Devant le jury

Président : Pr. Boufaida Mahmoud. Professeur, Université Mentouri de Constantine
Rapporteur : Pr. Benmohammed Mohamed. Professeur, Université Mentouri de Constantine.
Examinateurs: Pr. Koudil Mouloud. Professeur, ESI d'Alger.
Dr. Bilami Azzeddine. M.C, Université de Batna.
Dr. Chaoui Allaoua. M.C, Université Mentouri de Constantine

Date de Soutenance : 13 Juin 2009

REMERCIEMENTS

Tout d'abord, je voudrais exprimer mes vifs remerciements au Professeur Mohamed Benmohammed - mon directeur de thèse- qui m'a encadré avec enthousiasme, et a su me conseiller efficacement tout en me laissant travailler très librement. Pour cela, je lui suis très reconnaissant.

Mes profonds remerciements à tous les membres de jury : le Professeur Mahmoud Boufaida pour avoir accepté de présider le jury, le Professeur Mouloud Koudil, le Docteur Azzeddine Bilami et le Docteur Allaoua Chaoui, pour avoir accepté la lourde tâche d'examiner mon travail.

Je tiens également à remercier le Professeur Congduc Pham de m'avoir accueilli au sien de son laboratoire LIUPPA de Pau (France) et du temps qu'il m'a consacré pour réaliser une bonne partie de cette thèse.

Je tiens aussi à remercier le Docteur Faiza Belala du temps qu'elle m'a consacré pour lire et corriger cette thèse.

Enfin, je garde une place toute particulière pour toute ma famille. Je les remercie pour leur soutien de tous les instants.

A tous Merci……..

Résumé

Dans l'Internet, les communications multipoint (multicast) permettent l'échange de données entre plusieurs membres appartenant au même groupe. Au niveau de la couche réseau, un nouveau modèle de communication appelé multicast IP a été proposé ainsi qu'un certain nombre de protocoles de routage et de gestion de groupes. Au niveau de la couche transport, une autre classe de problèmes se pose. Cette classe comprend en particulier la fiabilité, le contrôle de congestion ainsi que la prise en charge de l'hétérogénéité des récepteurs qui appartiennent au même groupe. L'objectif de cette thèse est d'étudier l'apport de la combinaison des classes sender-initiated et receiver-intiated dans la solution du problème de la fiabilité au niveau transport tout en impliquant les routeurs. Pour valider nos propositions, nous avons adopté la technologie des réseaux actifs où les routeurs sont capables d'exécuter des services personnalisés en fonction des paquets reçus.

- *En premier lieu, nous avons proposé un protocole de multicast fiable actif nommé AMRHy. La particularité de notre protocole par rapport aux protocoles existants de multicast fiable actif, est que le notre, en combinant les deux classes sender-initiated et receiver-initiated, tient compte du lien sur lequel la perte se produise, permettant ainsi une meilleure répartition de la charge de recouvrement entre la source et les récepteurs avec la contribution des routeurs actifs. Dans cette combinaison de classes, la source détecte les pertes qui se produisent sur les liens proches d'elle (les liens source) et les récepteurs détectent celles qui se produisent sur les liens proches d'eux (les liens terminaux). Cette répartition équitable de la charge de recouvrement offre un mécanisme efficace pour résoudre les problèmes de la scalabilité qui surgissent à grand échelle a savoir la localité de perte et l'implosion d'acquittements en feedback.*
- *Ensuite, nous avons réalisé une évaluation analytique de performance, celle-ci montre l'intérêt de la combinaison des classes sender-initiated et receiver-initiated. Ainsi, pour valider nos résultats une étude comparative avec le protocole DyRAM a été dégagée. Ce dernier représente les protocoles de la classe receiver-initiated qui attribue la responsabilité de la détection de pertes aux récepteurs indépendamment du lien sur lequel la perte se produise, provoquant ainsi une mauvaise répartition de la charge de recouvrement des pertes.L'analyse de performance a porté sur deux métriques : le débit et la bande passante. Les résultats numériques ont montré que la combinaison des classes améliore de manière significative le débit et limite la bande passante requise par les messages de contrôle. Il est*

important de remarquer que les performances d'AMRHy surpassent celles de DyRAM surtout en augmentant la taille du réseau et la probabilité de perte.

Mots clés : Multicast IP, Multicast Fiable, Réseaux Actifs, Sender-initiated, receiver-initiated, Bande Passante, Débit.

Abstract

In the Internet, multipoint communications (multicast) allow the exchange of data between different members that belong to the same group. At the network layer level, a new communication model called IP multicast is introduced, and given number of membership management and routing protocols are proposed. At the transport layer level, another class of problems is to be addressed such as reliability, congestion control and the accommodation of the coexistence of heterogeneous receivers in the same multicast session. The objective of this thesis is to investigate the potential contribution of combining sender-initiated and receiver-initiated classes in solving reliability problem at transport level while implying the routers. To validate our proposals, we adopt the active networking technology where routers are able to perform customized services on packets they receive.

- *As a first step, a reliable multicast protocol called AMRHy was proposed. The particularity of our protocol compared to the existing reliable multicast protocols is that our protocol, by combining the sender-initiated and receiver-initiated classes, takes account of the link on which the loss occurs thus allowing a better distribution of loss recovery burden between the source and the receivers with the contribution of the active routers. In this combining the source detects the losses which occur on the links close to it (source links) and the receivers detect those which occur on the links close to them (tail links). This efficient distribution of loss recovery burden offers an effective mechanism to solve the problems which emerge on large scale such as the loss locality and the implosion of acknowledgement in feedback.*
- *Afterwards, in order to show the interest of this better distribution of loss recovery burden, an analytical analysis of performance was achieved. Thus, in order to validate our results a comparative study was made with the DyRAM protocol. This one represents the receiver-initiated class which attributes the responsibility for the detection of losses to the receivers regardless the link on which the loss occurs thus causing an inefficient distribution loss recovery burden. The analysis of performance related to two metric: throughput and the bandwidth. The numerical results show that combining classes significantly improves the throughput and limits the bandwidth needed by control messages. Interestingly, combining classes can outperform the receiver-initiated class depending on the network size and loss probability.*

Keywords: IP Multicast, Reliable multicast, Active networks, Sender-initiated, Receiver-initiated, Bandwidth, Throughput.

Table des matières

Table des figures

Liste des tableaux

Introduction générale

L’avènement de la technologie du protocole IP (*Internet Protocol*) a gagné la popularité et a beaucoup changé le visage de notre vie quotidienne. De nos jours, l’Internet est devenu indispensable pour la communication et l’échange d’information dans les différents domaines. Cependant, la communication actuelle est en grande partie limitée à l’échange de données en unicast (point à point), pendant qu’une large variété d’applications d’Internet émergeantes, couvrant un large spectre de domaines, impliquent plusieurs parties et exigent un mécanisme de communication multipoint tels que : l’enseignement à distance, la vidéo à la demande, la distribution des logiciels, les simulations interactives distribuées, la téléconférence, les applications coopératives, les jeux distribués, etc.

La solution la plus simple pour implanter un mécanisme de communication multipoint consiste à ouvrir plusieurs connexions unicast. Cependant, cette solution pose le problème de facteur d’échelle en présence d’un groupe important de récepteurs. Ainsi, la diffusion de paquets offre une solution plus simple que le mode unicast. Néanmoins, la diffusion dans un réseau large portée consomme une quantité excessive de ressources du réseau, principalement lorsque seul un petit groupe est intéressé par la réception. Une solution intermédiaire et meilleure, nommée multicast, consiste à diffuser les paquets de données seulement à un sous ensemble de nœuds du réseau. Le multicast est un service réseau qui traite les communications *one-to-many* ou *many-to-many* dans un réseau large portée tout en préservant la bande passante du réseau et la capacité de traitement de la source et évitant ainsi l’ouverture simultanée de plusieurs connexions. Le trafic redondant est évité par la duplication de paquets en cas de besoin et expédié étroitement aux destinateurs finaux du groupe.

En 1988, Deering a proposé un modèle de multicast ouvert, où les récepteurs intéressés à une session multicast, s’inscrivent seulement à une adresse de groupe. N'importe quel récepteur peut joindre ou quitter le groupe librement. La source ne se rend compte ni de l'identité ni du nombre de récepteurs inscrits. D'ailleurs, une source peut envoyer à une adresse multicast sans être inscrite. Dans ce modèle multicast, connu sous le nom de multicast IP (RFC 1112) [15], il est du sort des routeurs d'assurer la livraison de paquets à leurs destinations par des protocoles spéciaux de routage, qui maintiennent l'information d'adhésion et construisent des arbres de distribution multicast. Néanmoins, le multicast IP est une extension du traditionnel *best effort* unicast IP et,

par conséquent, il n'y a aucune garantie sur la livraison fiable de données. La fiabilité est une condition primordiale, exigée par plusieurs applications multicast tels que : la vidéo à la demande, les jeux distribués, les simulations interactives distribuées, etc. La prolifération de telles applications a motivé la conception de plusieurs protocoles de multicast fiable, qui constituent un sujet de recherches intenses et des efforts de développement pendant les deux dernières décennies.

Généralement, les différentes applications multicast sont classées selon deux critères fondamentaux : la fiabilité et le délai de livraison. Certaines peuvent tolérer la perte de quelques données tandis qu'elles exigent des délais très courts de livraison, comme par exemple, le cas des applications multimédia en temps réel. D'autres, par contre, sont peu sensibles à la contrainte de temps, mais elles exigent une fiabilité totale. Etant donné que le nombre de participants de telles applications se multiplie potentiellement par des centaines ou voire même par des milliers, n'importe quel protocole de multicast fiable doit donc fournir les mécanismes viables pour réaliser la scalabilité. La conception d'un protocole de multicast fiable permettant le passage l'échelle doit faire face aux problèmes de l'implosion en feedback des acquittements et de la localité de perte (duplication des retransmissions).

Implosion en feedback : Etant donné que le nombre de récepteurs dans certaines applications, se multiplie et que la probabilité pour qu'au moins un récepteur subisse une perte, augmente. Plusieurs récepteurs peuvent subir simultanément la même perte, dans ces cas, chaque récepteur est amené à retourner à la source une demande de réparation. Ce qui provoque un gaspillage de la bande passante et un écroulement de la source qui doit répondre à toutes les demandes reçues. Une augmentation de la latence de la livraison de données peut également résulter du problème de l'implosion en feedback des acquittements. Eviter, en présence d'un grand nombre de récepteurs, le problème de l'implosion des acquittements au niveau de la source est un défi qui va permettre le passage à l'échelle.

Localité de perte : Dans une large session multicast, un paquet de données est susceptible d'être perdu par plus d'un récepteur. Ainsi, la source est amenée à envoyer des réparations multiples comme réponse à des demandes semblables qu'elle reçoit. D'ailleurs, la source doit envoyer les réparations en multicast pour le groupe entier, certains récepteurs seront submergés par le traitement inutile des paquets de données déjà reçus. En plus de ce traitement inutile au niveau des récepteurs, les retransmissions des réparations consomment une quantité importante de la bande passante du réseau. D'où, l'importance de restreindre la portée des retransmissions aux

récepteurs qui ont effectivement perdu le paquet de données. Ce qui représente un autre défi permettant le passage à l'échelle.

Le consensus pour aborder ces problèmes est de déléguer la responsabilité de détection et de recouvrement des pertes aux récepteurs. Le protocole plus populaire de cette classe est SRM [20], qui permet aux récepteurs d'envoyer en multicast des paquets de contrôle au groupe en entier. N'importe quel récepteur qui possède le paquet demandé peut le soumettre en multicast. Avec une synchronisation parfaite des temporisateurs aléatoires, SRM peut résoudre le problème de l'implosion en feedback. Mais, le problème de localité de perte n'est pas résolu par SRM. Par contre, ce problème peut être allégé par la portée locale et hiérarchique [55]. D'autres protocoles adoptent une approche hiérarchique pour résoudre les problèmes d'implosion en feedback et de localité de perte tels que : RMTP [48], TMPT [62], LBRM [24], PGM [56] and LMS [47] en imposant une structure arborescente à la session multicast. Ainsi, ces protocoles nécessitent l'assistance d'une entité intermédiaire : un récepteur désigné [48], un serveur désigné [24] ou un routeur désigné [47, 56] qui doit orienter le paquet de réparation aux régions où la perte s'est produite effectivement. Cependant, la gestion d'un grand nombre de routeurs ou de récepteurs spécialisés sous une partition réseau où l'échec d'une machine créerait un énorme fardeau administratif. Une solution plus générale et plus flexible pour résoudre les problèmes de l'implosion en feedback et de localité de perte, est l'utilisation de l'approche réseaux actifs, où les routeurs peuvent faire des traitements personnalisés sur les paquets qui les traversent. Plusieurs protocoles de multicast fiable actif ont été proposé tels que : ARM [31], AER [29] et DyRAM [40]. Ces derniers sont tous basés sur la classe *receiver-initiated*, qui attribue la responsabilité de détection de pertes aux récepteurs indépendamment du lien sur lequel la perte se produit.

L'objectif de cette thèse est d'étudier l'apport de la combinaison des classes *sender-initiated* et *receiver-intiated* dans la solution du problème de la fiabilité au niveau transport tout en impliquant les routeurs. Pour valider nos propositions, nous adoptons la technologie des réseaux actifs où les routeurs sont capables d'exécuter des services personnalisés en fonction des paquets reçus.

- En premier lieu, nous proposons un protocole de multicast fiable actif nommé AMRHy. La particularité de notre protocole par rapport aux protocoles existants de multicast fiable actif, est que le notre, en combinant les deux classes *sender-initiated* et *receiver-initiated*, tient compte du lien sur lequel la perte se produise, permettant ainsi une meilleure répartition de la charge de recouvrement entre la source et les récepteurs avec la contribution des routeurs

actifs. Dans cette combinaison de classes, la source détecte les pertes qui se produisent sur les liens proches d'elle (les liens source) et les récepteurs détectent celles qui se produisent sur les liens proches d'eux (les liens terminaux). Cette répartition équitable de la charge de recouvrement offre un mécanisme efficace pour résoudre les problèmes de la scalabilité qui surgissent à grand échelle a savoir la localité de perte et l'implosion d'acquittements en feedback.

- Ensuite, nous réalisons une évaluation analytique de performance, celle-ci montre l'intérêt de la combinaison des classes *sender-initiated* et *receiver-initiated*. Ainsi, pour valider nos résultats une étude comparative avec le protocole DyRAM sera dégagée. Ce dernier représente les protocoles de la classe *receiver-initiated* qui attribue la responsabilité de la détection de pertes aux récepteurs indépendamment du lien sur lequel la perte se produise, provoquant ainsi une mauvaise répartition de la charge de recouvrement des pertes. L'analyse de performance a porté sur deux métriques : le débit et la bande passante.

Le reste de la thèse est organisé comme suit :

Le chapitre 1 introduit le multicast IP fourni au niveau de la couche réseau. Le chapitre 2 présente le concept des réseaux actifs et programmables. Le chapitre 3 résume l'état de l'art et notre analyse de la recherche dans le domaine de la fiabilité multicast. Un nouveau protocole de transport multicast fiable nommé AMRHy (***A****ctive* ***M****ulticast* ***R****eliable* ***Hy****brid protocol*) sera présenté dans le chapitre 4. L'évaluation des performances du protocole AMRHy et une étude comparative avec DyRAM seront détaillées dans le chapitre 5. Finalement, le travail se termine par une conclusion qui résume les contributions et donne les orientations futures.

Chapitre 1

Multicast IP

1. Introduction

Le multicast maintient le grand potentiel d'améliorer rigoureusement la livraison de données à un groupe de destinataires ayant un intérêt commun pour recevoir un flux particulier d'information, tel que : la voix, la vidéo, ou la donnée. Il n'existe aucune contrainte physique, géographique ou des frontières pour appartenir à un groupe, tant que les hôtes sont reliés à un réseau. Les hôtes désirant recevoir des données destinées à un groupe particulier joignent le groupe en utilisant un protocole de gestion de groupes : les hôtes/récepteurs doivent être des membres explicites du groupe pour pouvoir recevoir le flux de données, mais une telle adhésion peut être éphémère et dynamique. Ainsi le multicast permet à une source de transmettre simultanément une même copie de ses données vers un groupe identifié de récepteurs. Pour éviter la redondance d'un même paquet, celui-ci n'est transmis qu'une seule et unique fois sur un lien et sera dupliqué aux intersections en cas de besoin. C'est une manière d'imposer à un nœud de ne recevoir qu'un seul exemplaire d'un paquet. Plusieurs raisons peuvent motiver l'emploi des techniques de communication de groupe, en particulier, le passage à l'échelle en termes de nombre de participants, la réduction des coûts de transmission et l'augmentation des vitesses de transfert par rapport à la solution point à point. Cette dernière, malgré qu'elle représente le mode de communication le plus dominant dans Internet, est loin d'être optimale dans les situations où le nombre de destinataires est important, et lorsque les données sont volumineuses. Principalement deux problèmes sont rencontrés dans le cas de multiples envois unicast :

- un gaspillage de la bande passante,
- et une congestion autour de la source engendrée inutilement par les retransmissions du même paquet aux différents récepteurs.

Ainsi le multicast fournit une solution naturelle à ces problèmes en réduisant la charge de la source et des routeurs d'un réseau et en permettant une utilisation efficace de la bande passante (voir figure 1.1), par conséquent la conception d'un service multicast soulève plusieurs questions et autant de défis à résoudre tels que :

- l'identification et la gestion de groupes,
- la construction d'arbres de distribution multicast,
- et l'acheminement des paquets multicast à travers les routeurs d'un réseau.

Le travail de Deering vers la fin des années 80 a répondu à ces questions par la définition d'un modèle de service multicast IP, et la conception d'algorithmes permettant à des hôtes de joindre et de quitter arbitrairement un groupe multicast [3, 14, 15]. La transmission multicast IP introduit plusieurs mécanismes, qui seront discutés dans ce chapitre. La section 2 décrit le mode

d'adressage multicast, c'est-à-dire le type d'adresses attribué à la communication de groupes et la gestion de ces adresses. La section 3 couvre la gestion des groupes multicast. La section 4 présente les principes fondamentaux du routage multicast et les modèles de construction des arbres de distribution multicast. La section 5 examine les architectures des réseaux multicast et introduit d'une très brève les apports IPv6. Enfin une section 6 clôtura ce chapitre par une conclusion.

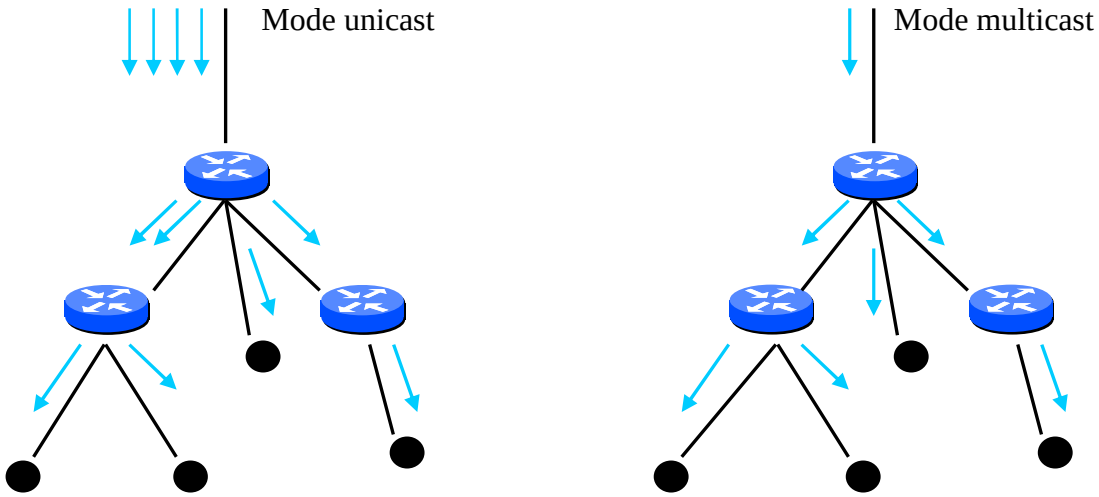

Figure 1.1 : La transmission en mode unicast et multicast

2. Adressage multicast

Le premier problème inhérent à la communication multicast est l'identification de groupes. Celui-ci est résolu par l'utilisation des adresses multicast. Dans le modèle de Deering, [15] spécifie les extensions demandées pour qu'un hôte puisse supporter le multicast IP. L'attribution des adresses multicast IP est contrôlée par l'IANA (*Internet Assigned Numbers Authority*) qui a réservé la plage d'adresses 224.0.0.0-239.255.255.255 (la classe D) pour la communication multicast IP. Pour chaque adresse multicast, il existe un ensemble de zéro ou plusieurs récepteurs qui sont à l'écoute des paquets transmis à cette adresse. Une source qui envoie des paquets à un groupe spécifique n'a pas besoin d'être un membre du groupe et elle n'est même pas sensée connaître les membres courants du groupe [36]. Ainsi, l'adresse source des paquets multicast IP est toujours une adresse unicast.

Cependant, il existe deux types de groupes d'hôtes [42] :

- Les groupes permanents : les applications qui font partie de ce type de groupe ont une adresse IP affectée par l'IANA de manière permanente. Un groupe permanent continue d'exister même s'il ne contient aucun membre. L'adhésion d'un hôte à ce type de groupe n'est pas permanente, un récepteur peut joindre ou quitter le groupe librement.
- Les groupes transitoires ou éphémères : tout groupe qui n'est pas permanent est par définition transitoire. Le groupe est disponible pour une affectation dynamique en cas de besoin. Les groupes transitoires cessent d'exister lorsqu'ils n'ont plus de membres.

Quelques adresses multicast IP ont été réservées pour des fonctions spécifiques. Les adresses qui existent dans la plage de 224.0.0.0 - 224.0.0.255 sont réservées pour être employées par les protocoles relatifs à un segment de réseau local. Ces adresses sont utilisées pour la découverte automatique du routeur et pour la communication de l'information de routage (par exemple, l'OSPF emploie 224.0.0.5 et 224.0.0.6 pour échanger l'information concernant l'état de lien). Les paquets IP avec ces adresses ne sont pas expédiés par un routeur ; ils restent locaux à un segment particulier du LAN (ils ont un paramètre du TTL, *Time To Live* égal 1 ; même si le TTL est différent de 1, ils ne sont toujours pas expédiés par le routeur). Ces adresses sont également dites adresses du lien-local.

Exemple de quelques adresses bien connues du lien-local :

- 224.0.0.1: Tous les systèmes du sous réseau
- 224.0.0.2: Tous les routeurs du sous réseau
- 224.0.0.4: Les routeurs DVMRP
- 224.0.0.5: Les routeurs OSPF
- 224.0.0.6: Les routeurs OSPF désignés
- 224.0.0.13: Tous les routeurs PIM

Les adresses de 224.0.1.0 à 238.255.255.255 sont des adresses connues à portée globale. Ces adresses sont employées pour transmettre l'information multicast à travers l'Internet et entre les organismes. Certaines de ces adresses ont été réservées pour des usages spécifiques tels que le NTP (Network Time Protocol) (224.0.1.1).

Exemple de quelques plages d'adresses à portée globale :

- 224.2.0.0–224.2.127.253: La vidéo conférence.
- 224.252.0.0–224.255.255.255: DIS des groupes transitoires.
- 224.0.12.000–224.0.12.063: Microsoft et MSNBC.
- 224.0.25.0–224.0.28.255: CME Market Data: affectées en Mars 2007.
- 224.0.254.000–224.0.254.255, Intelsat IPTV: affectées en Mars 2006.

Le processus d'allocation d'adresses est loin d'être évident, dans le sens où les groupes sont dynamiques, c'est à dire qu'un récepteur doit pouvoir joindre et quitter un groupe librement. Ceci pose deux problèmes :

- Premièrement, le maintient de l'arbre de distribution multicast correct. Par exemple, un ensemble de récepteurs peut à un instant *t* être abonné à un groupe, puis au fil de joindre et de quitter un groupe dynamiquement, il se peut qu'à un instant *t'* les récepteurs du groupe soient tous différents de ceux initialement présents. Il faut donc, un mécanisme de mise à jour de l'arbre.
- Deuxièmement, ce même phénomène de quitter un groupe dynamiquement peut générer un groupe vide, qui n'a plus raison d'exister. De la même manière, de nouveaux groupes peuvent se créer. Pour cela, il faut un mécanisme d'allocation et de récupération d'adresses dynamique.

La structure même des adresses multicast pose quelques problèmes, notamment celui du nombre de groupes pouvant existé simultanément (voir figure 1.2). Quatre bits sont réservés pour indiquer le type d'adressage, et le reste, c'est à dire les 28 bits, pour identifier les groupes. Est ce que c'est suffisant pour répondre au besoin de la planète? Ainsi plusieurs solutions ont été adoptées.

Figure 1.2 : Adresse IP classe D

2.1 Adressage à portée limitée

Une première solution est l'adressage à portée limitée. Ce mode d'adressage étant limité à un groupe local ou à une organisation, il a été défini dans le standard Internet [43]. Les adresses se trouvent dans la plage de 239.0.0.0 à 239.255.255. Tout comme une adresse de réseau local classique, ce mode permet la réutilisation d'adresses, en limitant leurs champs. Ainsi, les routeurs sont appelés à être configurés avec des filtres de paquet empêchant le trafic multicast dans cette plage d'adresses de circuler, en dehors d'un système autonome. Dans un système autonome, l'espace d'adressage à portée limitée peut être subdivisé, de sorte que des frontières locales multicast puissent être définies. Ceci tient compte également de la réutilisation d'adresse entre ces sous domaines Un des avantages implicites de ce mode d'adressage est qu'il simplifie

l'unicité de l'adresse sur le réseau, puisque celle-ci n'est valable et accessible que dans un espace restreint. Cependant, ceci pose le problème d'accès et de la traversée du réseau MBone (*Multicast Backbone*).

2.2 Adressage GLOP

Le standard Internet RFC 2770 [35] a recommandé que la plage d'adresses 233.0.0.0 /8 soit réservé pour les adresses définies statiquement par des organismes qui ont déjà un numéro de système autonome réservé (voir figure 1.3). Le numéro de système autonome est encastré dans les deuxièmes et troisième octets du rang 233.0.0.0 /8. GLOP est un mécanisme qui assigne des adresses multicast au système autonome (GLOP n'est ni un acronyme ni une abréviation).

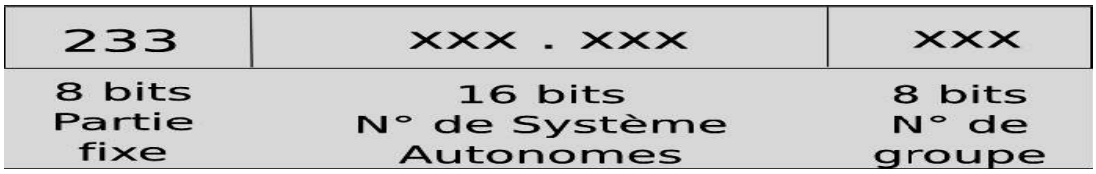

Figure 1.3 : Format d'adressage GLOP

Le problème de ce type d'adressage est qu'il n'y a que $2^8 = 256$ groupes par système autonome. Encore une fois cela n'est pas suffisant.

2.3 Adressage dynamique : Malloc

Une solution de gestion des adresses de groupe plus efficace était nécessaire. Le groupe de travail de l'IETF (*Internet Engineering Task Force*) a proposé le mode d'adressage du groupe Malloc. Celui-ci est composé de trois protocoles répartis sur trois niveaux (qui n'ont rien à voir avec les couches du modèle OSI) [4]:

- MASC : *Multicast Address Set Claim* (Niveau 3), ce protocole permet d'allouer des blocs d'adresses multicast à des domaines. Le temps de réservation est lent (il se compte en jours).
- MAAS : *Multicast Address Allocation Server* (Niveau 2), ce protocole permet aux serveurs d'adresses multicast de réserver des sous blocs d'adresses auprès de leur domaine.
- MADCAP : *Multicast Address Dynamic Client Allocation Protocol* (Niveau1), celui-ci permet la distribution d'adresses multicast aux clients.

Ces trois protocoles se partagent la tâche d'allocation d'adresses aux clients. Cette architecture hiérarchisée permet le dynamisme. En effet, l'unicité de l'adresse multicast exige du réseau une connaissance exhaustive des groupes existants, et des requêtes coûteuses. Ce qui justifie les

délais important au niveau des serveurs MASC. Au niveau le plus inférieur, le protocole MADCAP permet au client de demander une adresse au serveur MAAS. Il n'est pas acceptable de faire attendre les clients plusieurs jours ou même plusieurs heures, ainsi il peut y avoir des pré-allocations au niveau 2, pour pouvoir répondre rapidement. Un problème classique qui apparaît ici, et que l'on retrouve dans unicast, est celui de la fragmentation et la mauvaise répartition des adresses multicast, qui peuvent conduire à une pénurie apparente d'adresses pour certains domaines.

3. Gestion de groupes

Le second problème inhérent à la communication multicast est celui de la gestion de groupes qui a été résolu par le protocole IGMP (*Internet Group Management Protocol*) [15]. Ce protocole gère le mécanisme de joindre et de quitter un groupe par les hôtes. Il permet essentiellement d'informer les routeurs de l'existence d'un ou de plusieurs récepteurs appartenant à un groupe derrière chacune de leurs interfaces (voir figure 1.4). Il est important de noter que IGMP ne fournit ni l'identité, ni le nombre de destinataires présents dans un groupe derrière une interface. La seule information pertinente qu'il apporte, est qu'il y en a au moins un, et que par conséquent le paquet doit être transmis sur cette interface. Ce protocole existe en trois versions distinctes, la première est évidemment la plus simple et celle qui comporte le plus d'inconvénients. Les deux autres proposent des solutions de ces inconvénients. Les trois versions sont présentées d'une manière succincte :

3.1 IGMPv1

Dans la version 1 (définie dans RFC 1112 [15]), il existe deux types de messages IGMP échangés entre les routeurs et les hôtes :

- *IGMP membership Report* : ce type de message est envoyé par un hôte, soit désirant souscrire à un groupe, soit pour répondre à un message *IGMP membership Query*. L'adresse correspondant au groupe concerné se trouve dans le champ *destination address* du paquet IP. Ce message sera pris en compte par le routeur désigné, ou encore le DR (*Designated Router*).
- *IGMP membership Query* : ce type de message est envoyé par le DR en sortie de ses interfaces sur lesquelles les hôtes sont connectés. Afin d'atteindre tous ces hôtes, une sollicitation aveugle est envoyé à l'adresse 224.0.0.1 toutes les 60 secondes avec un TTL égal à 1. Cette adresse est affectée officiellement aux machines "parlant" IGMP. Ainsi tous les hôtes sous le DR recevront cette requête. Pour répondre à cette requête, chaque hôte

concerné envoie un rapport (via *IGMP membership Report*). Ce qui permet au DR de vérifier s'il existe encore, au moins un destinataire du groupe derrière son interface. Cependant, afin d'éviter trop de redondance, si plusieurs hôtes font partie d'un même groupe, chaque hôte marque un temps d'attente aléatoire et n'envoie pas la réponse si elle n'est que le double d'un message déjà envoyé. En effet, le DR lui suffit qu'il y ait un seul hôte intéressé pour qu'il continue à envoyer le trafic correspondant. Si le DR ne reçoit aucun message *IGMP membership Report* après trois requêtes consécutives, il considère qu'il n'y a plus de destinataires du groupe sur cette interface.

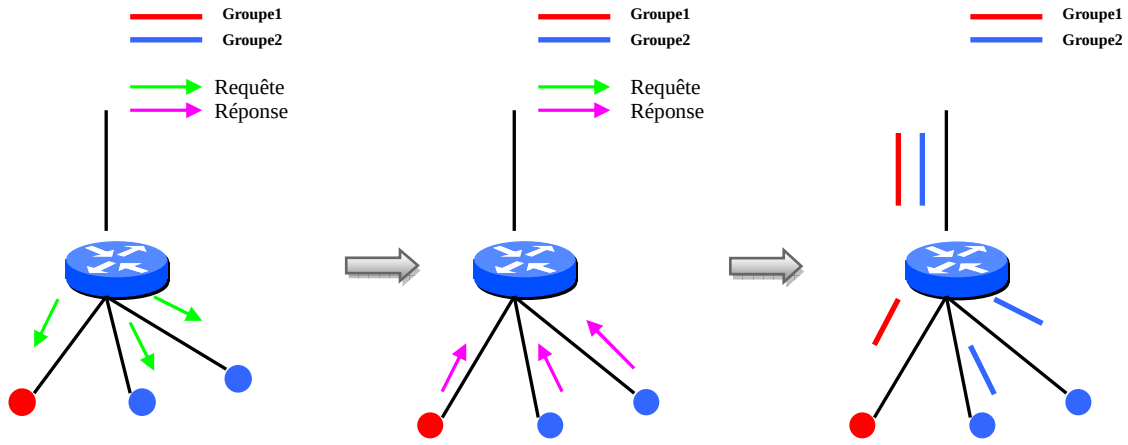

Figure 1.4 : Gestion des groupes multicast par le protocole IGMP

Cette version présente deux limites majeures :

- La première concerne la latence entre le moment où un hôte désire quitter un groupe et celui où l'émission s'arrête effectivement. En effet, ce délai dépend entièrement de la fréquence des messages *IGMP membership Query*, ainsi que du délai à partir duquel le DR considère que la non réponse à ce message signifie qu'il n'existe plus de récepteur intéressé par le flux destiné au groupe.
- La seconde est la sécurité : il n'existe aucun moyen pour qu'un hôte puisse filtrer une source de trafic donné. Ce qui signifie qu'il peut y avoir dans le réseau une source parasite qui diffuse ses données vers les hôtes sans que ceux-ci puissent l'empêcher. Ceci peut être considéré comme un défaut de sécurité puis qu'il est possible en théorie de causer des dénis de service, ou au moins fortement polluer le trafic.

3.2 IGMPv2

La deuxième version d'IGMP a été définie dans RFC 2236 [18], elle fonctionne d'une manière semblable à la version 1 mais avec l'addition de deux nouveaux types de messages :

- *Group Specific Query* : ce type de requête permet de cibler directement le groupe en permettant au DR de connaître s'il est toujours opportun d'envoyer le trafic à ce groupe,
- *Leave Group* : avec ce mécanisme, les hôtes peuvent explicitement communiquer leur intention de s'écarter du groupe au DR. Ainsi, lors de la réception d'un message LG, le DR interroge ce groupe par l'envoi d'une requête *Group Specific Query* pour déterminer s'il existe toujours des hôtes qui ont besoin de recevoir le trafic. S'il ne reçoit aucune réponse, le DR déclare le groupe comme étant vide et cesse d'expédier le trafic. Ce mécanisme permet au DR de détecter plus rapidement un groupe localement vide, et donc ne plus transmettre inutilement des paquets.

La version IGMPv2 a été optimisée afin de réduire le temps de la latence entre le retrait d'un membre et l'arrêt effectif du flux de paquets qui lui est destiné. Avec la version précédente, un routeur ne détectait le retrait du dernier membre d'un groupe que par l'absence de réponse à une requête *Query*. Désormais, un hôte peut signaler son retrait à travers un message LG. De plus, le DR a la possibilité d'émettre une sollicitation spécifique à un groupe.

3.3 IGMPv3

Cette version définie dans RFC 3376 [9] représente un véritable tournant par rapport aux deux versions précédentes. Elle permet de prendre en charge des inscriptions à un groupe pour une ou plusieurs adresses sources spécifiques. C'est ce qu'on appelle le filtrage. Cette information peut être employée par des protocoles de routage multicast pour éviter de livrer des paquets multicast à partir de sources spécifiques aux sous réseaux, où il n'y a aucun récepteur intéressé. Ainsi, la version 3 est grandement améliorée concernant les aspects sécuritaires. Cependant, il est possible pour un hôte de demander de recevoir le trafic multicast pour un groupe qu'en provenance de certaines sources. Ce filtrage peut se faire selon deux modes [64]:

- *Mode Inclusion* : le récepteur annonce l'adhésion à un groupe et fournit une liste d'adresses IP (la liste d'INCLUSION) à partir des quelles il souhaite recevoir le trafic. Le récepteur n'accepte les paquets multicast destinés à un groupe que si les sources font partie d'un ensemble d'adresses défini à l'avance et notifié au DR par le biais d'un message *Group-source Report*.

- *Mode Exclusion* : le récepteur annonce l'adhésion à un groupe et fournit une liste d'adresses IP (la liste d'EXCLUSION) à partir des quelles il ne souhaite pas recevoir le trafic. Ceci indique que le récepteur accepte tous les paquets multicast destinés à un groupe sauf ceux provenant des sources dont la liste est transmise par l'hôte au DR par le biais d'un message *Exclusion Group-source*.

Cette nouvelle considération de la paire *Group-Source* est très utile lorsqu'il existe plusieurs sources pour le même groupe. Nous notons que cette version est plus complexe que les deux précédentes, elle permet d'interdire une source, elle s'éloigne donc du modèle de Deering, qui considère que chaque membre peut émettre sans restriction à n'importe quel groupe. Néanmoins, les économies de la bande passante faites par ce filtrage restent locales dans la mesure où elles ne sont pas propagées.

4. Principes fondamentaux du routage multicast

Dans la section précédente, nous avons vu les mécanismes qui permettent aux hôtes de joindre et de quitter les groupes en utilisant un protocole de gestion de groupes (IGMP). Une fois que le groupe est constitué, il serait nécessaire d'acheminer de manière efficace le trafic vers l'ensemble des abonnés. Ceci présente le troisième problème inhérent à la communication multicast et qui sera résolu par les protocoles de routage multicast. Le routage et l'expédition en multicast sont très différents de ceux de l'unicast. D'abord, dans le routage multicast, la source envoie le trafic à un groupe dynamique de récepteurs. Pour les atteindre tous, le chemin de livraison du paquet multicast nécessite la création de plusieurs branches à travers le réseau construisant ainsi un arbre de distribution. Deuxièmement, et contrairement au routage unicast, l'adresse source a un rôle important dans l'expédition des paquets multicast basée sur la méthode propagation de paquets dite : "l'expédition de chemin inverse", RPF (*Reverse Path Forwarding*). En plus, le comportement des récepteurs affecte directement l'information de routage par l'intermédiaire du protocole de gestion de groupes.

4.1 L'expédition du chemin inverse

La méthode de l'expédition du chemin inverse RPF (*Reverse Path Forwarding*) est une notion fondamentale du routage multicast. Elle vérifie si un paquet multicast est reçu sur la bonne interface vers laquelle le routeur aurait expédié un paquet unicast vers la source. Cette méthode évite les boucles d'expédition tout en vérifiant la source du paquet multicast et l'interface d'entrée IIF (*Incoming InterFace*). Un routeur accepte un paquet multicast seulement si celui-ci

provient de l'interface appropriée. En effet, la méthode RPF utilise la table de routage unicast pour déterminer la liste des interfaces sortantes sur lesquelles le routeur expédiera ce paquet. Elle permet aux routeurs d'expédier le trafic multicast tout le long de l'arbre de distribution en adoptant la procédure de contrôle suivante [42]:

a) Le routeur cherche l'adresse source dans la table de routage unicast pour déterminer si le paquet est arrivé sur l'interface qui représente le chemin inverse vers la source.
b) Si le paquet est arrivé sur l'interface appropriée, le contrôle de RPF est réussi et le paquet sera expédié.
c) Si le contrôle de RPF échoue dans l'étape b), le paquet sera écarté.

4.2 Les modèles de routage multicast

Les routeurs créent les arbres de distribution multicast qui contrôlent le chemin qu'empreinte le trafic multicast IP, à travers un réseau afin de le délivrer à tous les récepteurs. Les mécanismes existent pour créer et pour maintenir ces arbres (par exemple, l'élagage). Les différents algorithmes de routage multicast, par exemple : PIM (*Protocol Independant Multicast*) [45], CBT (*Core Based Trees*) [3], et DVMRP (*Distance Vector Multicast Routing Protocol*) [58] emploient différentes techniques pour établir l'arbre de distribution multicast. Les membres d'un groupe multicast peuvent joindre ou quitter le groupe à n'importe quel moment qui est mis à jour de façon dynamique. Lorsque tous les récepteurs actifs sur une branche particulière cessent de demander le trafic pour un groupe particulier, les routeurs élaguent cette branche de l'arbre de distribution et arrêtent l'expédition du trafic vers cette branche. Si un récepteur sur cette dernière redevient actif et demande le trafic multicast, les routeurs modifieront dynamiquement l'arbre de distribution et reprendront l'expédition du trafic [64]. Ainsi, on peut classer les algorithmes de routage selon deux modèles de base [34]: le modèle d'inondation et d'élagage et le modèle de joint explicite. Les protocoles de routage multicast basés sur ces modèles sont souvent appelés le mode dense et le mode épars, respectivement.

a. Modèle d'inondation et d'élagage

Dans ce modèle, les routeurs inondent le réseau en entier par les paquets multicast, c'est-à-dire en se basant sur la méthode RPF, les routeurs expédient ces paquets sur toutes les interfaces sauf l'interface entrante. Lorsque le paquet est livré à un routeur DR, celui-ci vérifie si un des récepteurs l'a informé de son existence par l'intermédiaire du protocole de gestion de groupes. Si aucun récepteur n'existe, la procédure d'élagage commence. En se basant toujours sur la méthode RPF, le routeur DR envoie un message d'élagage au routeur ascendant. Si un routeur reçoit un

message d'élagage et aucun autre routeur ou destinataire ne demande la réception du trafic multicast, il cesse d'expédier des paquets multicast vers le groupe à travers l'interface d'arrivée du message d'élagage. En plus, lorsque le routeur arrête l'expédition vers toutes les interfaces sortantes possibles, il envoie à son tour le message d'élagage au routeur ascendant. En répétant la procédure d'élagage, la distribution de paquet converge vers une forme idéale où les paquets sont uniquement livrés aux récepteurs inscrits à un groupe particulier (voir figure 1.5).

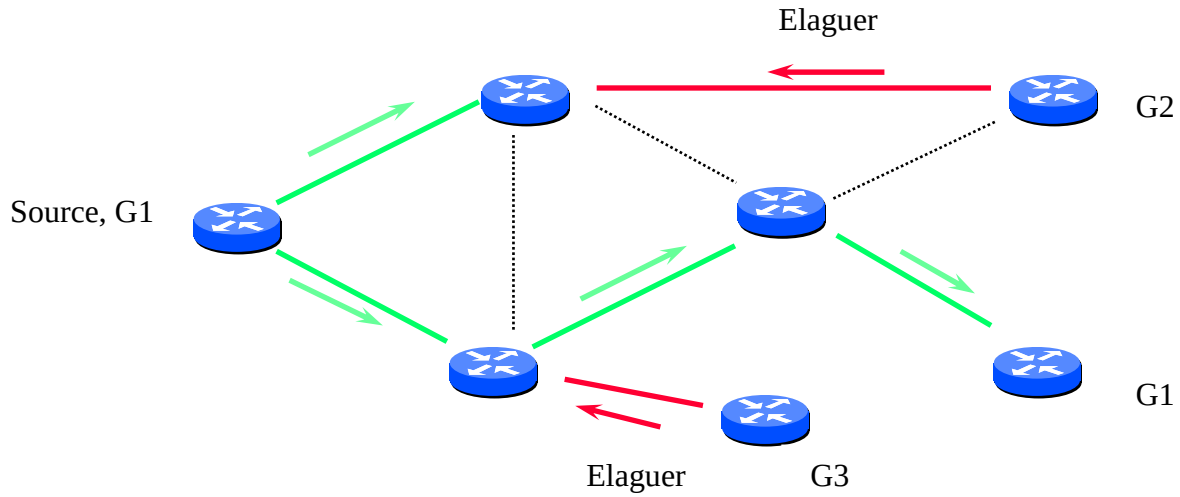

Figure 1.5 : Elagage d'un arbre

Les chemins d'expédition résultants forment un arbre de distribution pour les paires d'adresses source et de groupe. L'arbre de distribution ainsi construit est basé sur le plus court chemin, SPT (*Shortest Path Tree*), dont la racine est la source du groupe. La notation (S, G) est utilisée pour décrire un SPT où S est l'adresse IP de la source et G est l'adresse du groupe. Les SPTs aboutissent à une topologie de chemin optimal entre la source et les récepteurs en termes de nombre de sauts, ce qui permet d'obtenir une latence réseau minimale pour la distribution du trafic multicast. Cependant, les routeurs multicast sont requis pour maintenir l'information de chemin pour chaque source. Dans des réseaux à large portée, le maintient de cette information d'état peut surcharger les routeurs surtout en termes de ressources mémoire requises pour stocker la table de routage multicast, en particulier, lorsqu'il y a un nombre important de sources et de groupes. Nous notons que les algorithmes de multicast de Deering [13, 14, 15] construisent des arbres de distribution basés sur le plus court chemin, avec un arbre de distribution multicast par source [3]. L'inconvénient majeur de ce modèle est qu'il n'est pas approprié aux réseaux à grande échelle. Son principe d'inondation consomme inutilement une quantité importante de la bande passante.

b. Modèle de rejoint explicite

Le modèle de joint explicite adopte une approche totalement différente pour résoudre le problème de routage multicast. Il est basé sur les arbres partagés (*Shared Tree*) qui utilisent une racine commune placée à un point donné du réseau. Cette racine est appelée point de rendez-vous RP (*Rendez-vous Point*) ou le noyau de l'arbre (*core* ou *center*). En se servant d'un arbre partagé, les sources envoient leur trafic au nœud racine (RP) qui se charge de son expédition à travers l'arbre partagé pour atteindre tous les récepteurs actifs.

Dans ce modèle, un destinataire peut s'abonner à un groupe en envoyant un joint explicite vers le RP. Ce message est propagé jusqu'au RP. Lorsque le message joint atteint le RP, le chemin de distribution du paquet multicast, vers le groupe spécifié, est établi (voir figure 1.6).

Le RP utilise la méthode RPF pour tester les interfaces de l'arbre partagé. La notation (*, G) est utilisée pour représenter cet arbre, où '*' signifie toutes les sources. Les arbres partagés sont plus performants que ceux par source, ils nécessitent moins de mémoire pour stocker l'information d'état de routage, moins de bande passante et moins de calcul local. Malgré ces avantages, les arbres partagés présentent deux limites majeurs : d'une part, l'utilisation d'un arbre unique mène à la centralisation de trafic et provoque ainsi une congestion sur les liens près de la racine, et d'autre part les chemins entre chaque source et chaque destination ne sont pas forcément optimaux contrairement aux arbres par sources et par conséquent la latence réseau n'est pas minimale.

5. Matériel et Infrastructure

L'acheminement des paquets multicast peut se faire selon deux types de routeurs. Un paquet multicast peut transiter par un routeur adapté au trafic multicast. Dans ce cas, le routeur recherche dans sa table de routage toutes les sorties qui aboutissent aux différents membres du groupe multicast de destination et transmet le paquet uniquement sur ces sorties. Comme le paquet multicast, peut aussi transiter par un routeur simple, où la notion de tunnel multicast est nécessaire, dans ce cas on parle généralement de réseau virtuel appelé "Mbone".

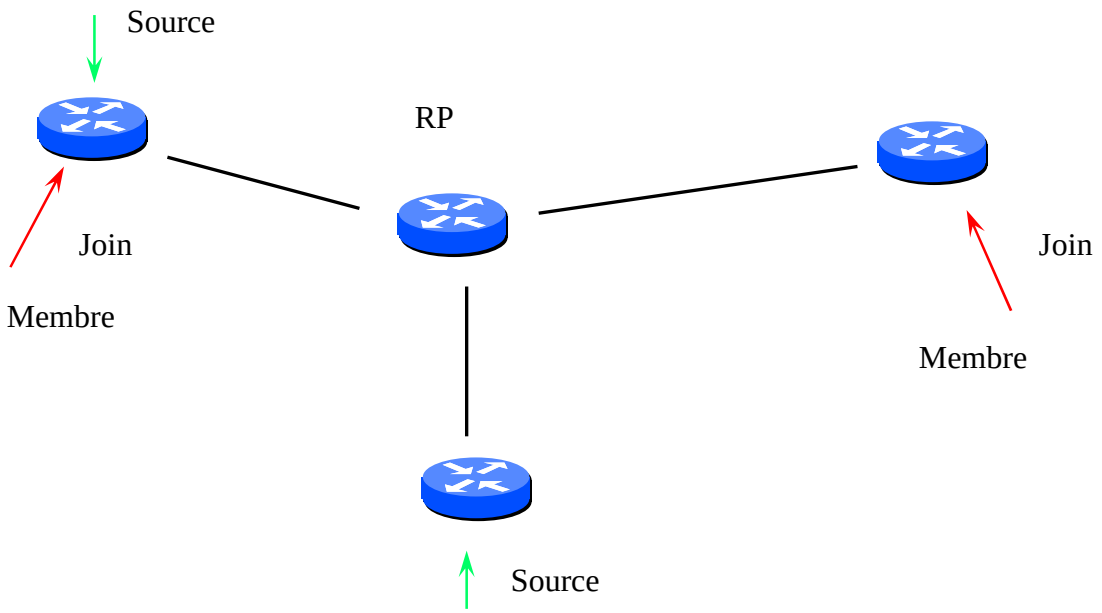

Figure 1.6 : Arbre partagé.

5.1 MBone

Le MBone (*Multicast Backbone*) est issu d'expérimentations de l'IETF (*Internet Engineering Task Force*) à San Diego en 1992, visant à diffuser le son puis la vidéo de ses réunions plénières. L'idée était de créer une zone de test grande nature et quasi permanente. Le MBone est alors un réseau virtuel, qui s'appuie sur le réseau Internet classique pour véhiculer un trafic multicast. Il forme une couche d'abstraction au dessus de la couche IP afin de supporter un routage IP multicast entre des machines. Or, la majorité des routeurs en exploitation ne supportaient pas le multicast, le MBone est composé d'îles multicast reliées entre elles par des tunnels (voir figure 1.7). En utilisant l'infrastructure des réseaux existants, le MBone évite le besoin d'avoir des réseaux dédiés pour la communication de groupe. Les informations transmises par le MBone sont acheminées dans le réseau Internet d'une manière standard. Les points finaux du tunnel sont typiquement des machines type "poste de travail" tournant sous un système d'exploitation supportant le Multicast [65]. Les stations tunnels ont un rôle important. Elles font le lien entre la partie du réseau dont le mode de transmission est limité à l'unicast et au broadcast et les parties où le mode multicast est pris en charge. La station tunnel déballe les trames contenant des paquets multicast, reçus à travers les tunnels, et les transmet aux destinataires se trouvant sur sa partie du réseau. Symétriquement, la station encapsule les paquets multicast destinés à d'autres parties du réseau dans des paquets unicast IP et les expédie à travers les tunnels appropriés.

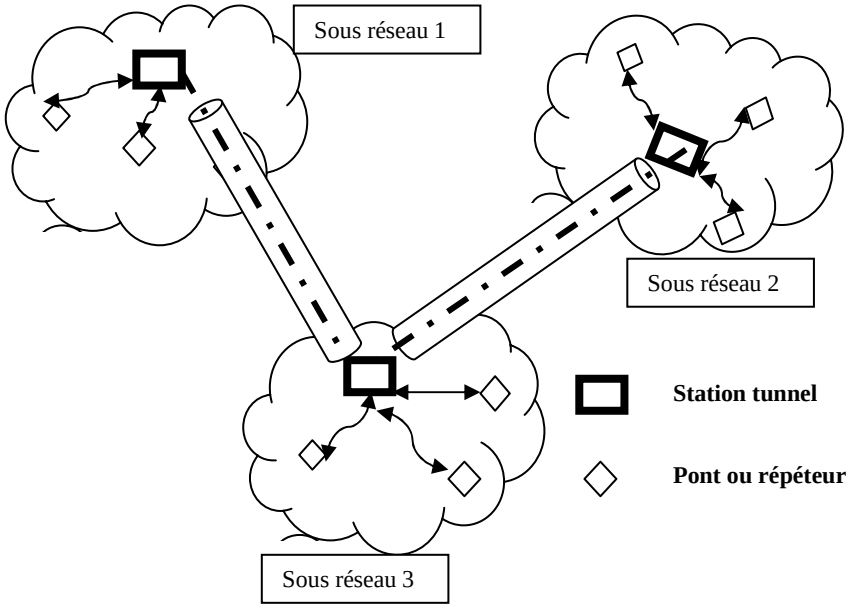

Figure 1.7 : Architecture logique du réseau MBone.

5.2 Routeurs multicast

Les routeurs récemment commercialisés sont généralement des routeurs qui prennent en charge l'acheminement des paquets multicast sans le recourt au '*tunneling*' qui tend à disparaître. En plus des tables de routage multicast, les routeurs ont besoin de connaître la route vers les adresses unicast de certains nœuds (par exemple la source ou le RP). Un routeur multicast a donc besoin de maintenir une table de routage unicast. Par conséquent, les protocoles de routage tels que CBT (*Core Based Trees*) [3], DVMRP (*Distance Vector Multicast Routing Protocol*) [58], MOSPF (*Multicast Open Shortest Path First*) [44] et PIM (*Protocol Independant Multicast*) [45] permettent d'acheminer les paquets multicast transmis à tous les récepteurs membres d'un groupe.

5.3 Limites de l'IPv4

La version IP en cours : IPv4, a été en service pendant presque trois décennies. Mais, elle commence à montrer des limites vis à vis des demandes émergentes en termes de la cardinalité d'espace d'adressage, la mobilité à haute densité, le multimédia, et la forte sécurité. Par conséquent, l'IPv6 crée un nouveau format d'adresse IP, de sorte que le nombre d'adresse IP ne s'épuise pas pendant plusieurs décennies ou bien aussi longtemps que possible à ce qu'une nouvelle récolte entière d'équipements soit reliée à l'Internet. IPv6 améliore également le routage

et la configuration automatique de réseau. Spécialement, lorsque les nouveaux dispositifs qui seront reliés à l'Internet sont des dispositifs prêts à l'emploi (“*plug-and-play*”).

5.4 IPv6

IPv6 est le protocole de la prochaine génération d'Internet, qui était d’abord appelé IPng (“*Internet Next Generation*”). L'IETF a développé les caractéristiques de base de ce protocole pendant les années 90 pour supporter la migration à un nouvel environnement. IPv6 défini dans RFC 2460 [16], est considéré comme étant une version améliorée d’IP. Celui-ci est conçu pour coexister avec IPv4 et pour fournir par la suite de meilleures possibilités d’interconnexion de réseaux qu’avec IPv4 [66].

a. Intérêts d’IPv6

Les avantages d'IPv6 peuvent être résumés comme suit :

- Facteur d’échelle (scalabilité) : les adresses IPv6 sont représentées sur 128 bits par contre celles d’IPv4 sont représentées sur 32 bits.
- Sécurité : IPv6 inclut la sécurité dans ses spécifications, tels que le chiffrement de données et l'authentification de la source de données.
- Application en temps réel : Pour fournir un meilleur support pour le trafic en temps réel (par exemple, VoIP), IPv6 inclut “des flux étiquetés” dans ses spécifications. Grâce à ce mécanisme, les routeurs peuvent identifier le flux de bout en bout auquel appartiennent les paquets transmis.
- Prêt à l’emploi (*plug-and-play*): IPv6 inclut un mécanisme prêt à l'emploi qui facilite le raccordement des équipements au réseau. La configuration requise est automatique.
- Mobilité : IPv6 inclut des mécanismes plus efficaces pour gérer la mobilité, particulièrement dans les réseaux mobiles.
- Protocole optimisé : IPv6 incarne les meilleures pratiques d’IPv4 mais omis les caractéristiques qui sont non utilisées ou périmées. En conséquence, il produit un protocole IP mieux optimisé.
- Adressage et routage : IPv6 améliore l’adressage et le routage hiérarchique.
- Extensibilité : IPv6 a été conçu pour être extensible et offre un support pour de nouvelles options.

b. Espace d'adressage IPv6

Le format d'adressage IPv6 est décrit dans RFC 2373 [22]. La taille relativement large de l'adresse IPv6 est conçue pour être subdivisé en domaines de routage hiérarchique qui reflètent la topologie de l'Internet moderne. L'utilisation de 128 bits fournit des niveaux multiples de hiérarchie et de flexibilité dans la conception d'adressage et de routage hiérarchiques. Une adresse IPv6 est représentée en tant que huit groupes de 16 bits, chacun séparé par le caractère ":". Chaque groupe de 16 bits est représenté par quatre chiffres hexadécimaux. Un exemple de l'adresse IPv6 est représenté comme suit : 3223:0BA0:01E0:D001:0000:0000:D0F0:0010.

Dans IPv6 les adresses multicast commencent par le préfixe 1111 1111 (voir figure 1.8). Le préfixe est suivi de deux champs, chacun de 4 bits : drapeaux et portée. Le drapeau T indique initialement si l'adresse est permanente ou transitoire. Le standard Internet RFC3306 [21] a ajouté le drapeau P (préfixe) ; ce drapeau permet à une partie de l'adresse du groupe d'inclure le préfixe de l'adresse unicast de la source, ce qui crée globalement une adresse unique du groupe. Le drapeau de R est employé pour indiquer que l'adresse RP est encastrée dans l'adresse du groupe ; avec le RP encastré, les drapeaux R, P, et T sont tous positionnés à 1. La portée est un sous-ensemble du réseau. Les 112 bits restants de l'adresse IPv6 sont l'identification du groupe.

c. Gestion des groupes multicast IPv6

Le multicast IPv6 n'emploie pas le protocole IGMP, pour gérer les groupes, mais plutôt celui de la découverte d'auditeur multicast MLD (*Multicast Listener Discovery*). Le protocole MLD est employé par un routeur IPv6 pour découvrir la présence des auditeurs multicast sur des liens locaux et pour découvrir quelles adresses multicast intéressent les nœuds voisins. MLDv1 est semblable à IGMPv2, et MLDv2 est semblable à IGMPv3 [42].

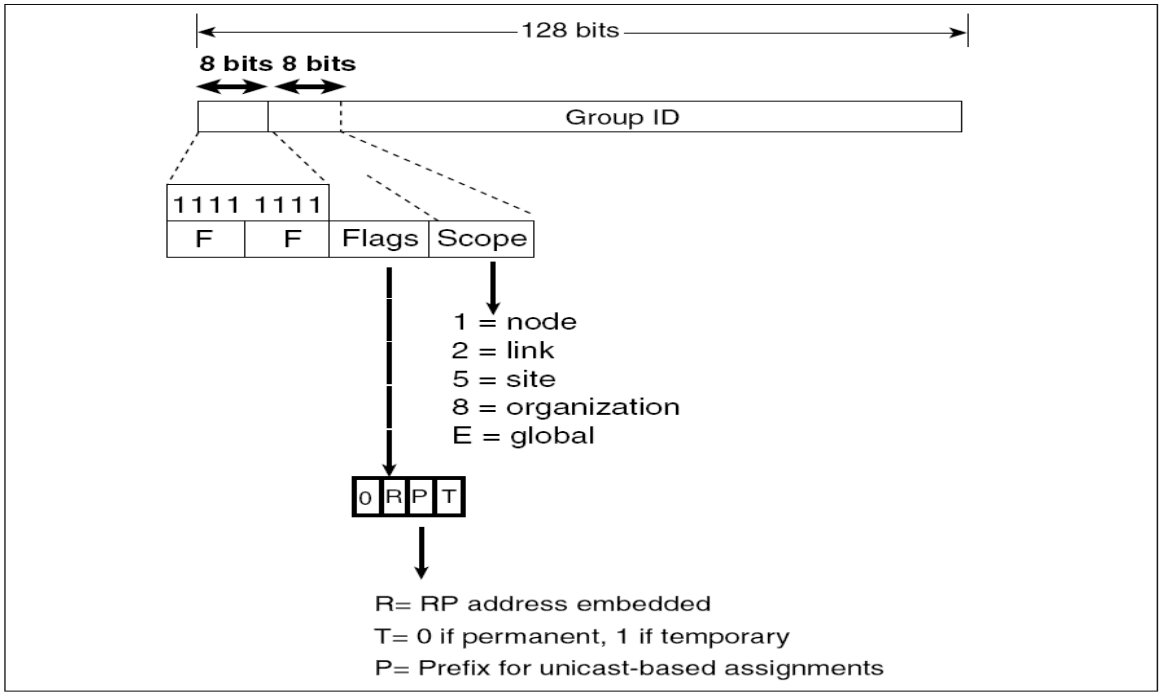

Figure 1.8 : Adresse multicast IPv6.

6. Conclusion

Dans ce chapitre nous avons présenté les protocoles du multicast IP et les techniques utilisées pour permettre une distribution efficace de l'information à grande échelle destinée à une large population d'utilisateurs. L'avantage majeur et bien connue du multicast IP est qu'il préserve la bande passante d'un réseau par la duplication des paquets, qu'en cas de besoin. D'autre part le multicast IP facilite la gestion dynamique des groupes, en permettant à un membre de joindre et de quitter un groupe d'une manière transparente à la source. En outre, les sources parviennent à atteindre dynamiquement les récepteurs grâce aux protocoles de routage. Ces derniers s'articulent sur deux approches de bases : la première dite d'inondation et d'élagage, qui reste simple à mettre en œuvre. Par contre, elle est totalement inadaptée à grande échelle car, elle fait intervenir périodiquement tous les nœuds du domaine, et par conséquent, consomme énormément de bande passante. La seconde, dite à joint explicite, est plus adaptée à grande échelle, mais peut entrainer une congestion autour du point de rendez vous. Le multicast IPv6 ouvre de nouvelles perspectives pour plusieurs applications d'Internet notamment en termes de facteur d'échelle, de la mobilité et de la sécurité. Cependant, le multicast IP suit la vision de son ancêtre IP en n'assurant qu'une délivrance best effort. Pour assurer la fiabilité plusieurs solutions ont été proposées. Parmi celles-ci, une solution basée sur une nouvelle approche réseau (réseaux actifs) qui sera introduite dans le chapitre suivant.

Chapitre 2

Réseaux actifs

1. Introduction

Les réseaux de données traditionnels transportent les paquets de données d'un système final à un autre. Idéalement, ces paquets sont transmis d'une manière opaque, autrement dit, le réseau n'interprète pas le contenu informationnel (sauf l'entête protocolaire) des paquets qu'il véhicule. Dans ce type de réseaux, les routeurs ont un objectif unique, ils routent les paquets qui les traversent et les expédient vers leurs destinations. Face à l'apparition incessante de nouvelles applications s'appuyant sur ces réseaux ; il est impossible de mettre au point un service global capable de satisfaire toutes les applications.

L'approche des réseaux actifs est relativement différente puisque cette fois les éléments actifs du réseau peuvent dynamiquement se reconfigurer. Nous pouvons donc réaliser un nombre impressionnant d'applications, qui étaient jusqu'à présent difficilement faisables avec les réseaux traditionnels. Les réseaux actifs reposent sur des équipements réseau ouverts et programmables, ayant la possibilité de déployer dynamiquement des programmes qui seront appliqués à la volée sur les paquets de données. Ceux-ci permettent à des paquets de contenir, non seulement des données, mais aussi des fragments de code qui sont exécutés dans les environnements d'exécution des routeurs actifs. Ainsi, les paquets peuvent consulter des services, des informations dans le routeur, et décider ce qu'il faut faire au bon moment et au bon endroit. L'objectif des réseaux actifs est d'exploiter les réseaux de transport de données de manière plus "intelligente" par l'injection dynamique de services personnalisés dans les équipements réseaux.

Dans la section 2, nous présentons une classification des réseaux à partir de laquelle nous situons le contexte des réseaux actifs. La section 3 présente le principe de fonctionnement des réseaux actifs avec les différentes approches qui existent. La section 4 introduit le concept service actif. La section 5 couvre le domaine d'application des réseaux actifs. Et enfin, nous clôturons le chapitre par une conclusion.

2. Classification des réseaux

Afin de clarifier la notion de réseaux actifs. Nous allons définir dans cette section les principales caractéristiques de deux grandes familles de réseaux (les réseaux traditionnels et les réseaux programmables).

2.1 Réseaux traditionnels

Les réseaux traditionnels sont ceux qui sont actuellement utilisés dans la plus grande majorité des cas. Ces réseaux sont figés, puisqu'il est impossible de modifier les services implantés ou d'en ajouter de nouveaux. La seule modification que l'on puisse faire est la modification des tables de routage. Dans ce type de réseaux, les premières applications distribuées ont été caractérisées par des contraintes d'ordre et de fiabilité sur les données transmises (i.e. transfert de fichiers, courrier électronique, accès aux pages Web, etc.). De ce fait, les premiers services de communication ont été spécifiquement conçus pour répondre à ces besoins. Les protocoles de transport TCP et UDP sont des exemples de cette vision restrictive de la conception des services de communication. Le protocole TCP offre un service totalement fiable et totalement ordonné alors que le protocole UDP fournit un service non fiable et non ordonné. Aujourd'hui, de nouvelles applications présentant des contraintes de qualité de service plus complexes ont été conçues. Ces applications telles que la vidéo à la demande ou la visioconférence présentent, d'une part de fortes contraintes temporelles, de bande passante et de synchronisation multimédia, et d'autre part, sont capables de tolérer un service de communication imparfait (i.e. un service partiellement ordonné et/ou partiellement fiable). Malheureusement, les services de base fournis par ces protocoles existants sont mal adaptés à ce genre d'applications. Pour y remédier, des solutions à ce problème sont apparues véritablement à partir de 1995 avec la création de nombreux ateliers (*workshop*) consacrés aux réseaux programmables.

2.2 Réseaux programmables

Les réseaux programmables quant à eux, sont plus flexibles que les réseaux traditionnels puisqu'ils proposent un environnement de programmation à l'échelle du réseau permettant de programmer des nouveaux services [27]. Les principales motivations des réseaux programmables peuvent être résumées dans les points suivants :

- les délais entre la spécification initiale d'un nouveau service, sa normalisation, sa réalisation et son déploiement à grande échelle sont devenus énormes et ne conviennent plus aux besoins de la réactivité des opérateurs et des fournisseurs de services ;
- le réseau peut profiter des informations issues des applications sur la nature et la sémantique de leurs flux ;
- le besoin en fonctionnalités applicatives dans les composants du réseau (firewall, caches Web, etc.) est de plus en plus présent.

Deux écoles, sur la façon de rendre des réseaux programmables, ont émergé : l'école des réseaux à signalisation ouverte (OPENSIG) et celle des réseaux actifs.

a. Réseaux à signalisation ouverte

Une approche de signalisation ouverte est une approche dans laquelle le plan de signalisation est remplacé par un environnement logiciel distribué, ce qui permet de programmer de nouveaux protocoles et de services. Dans cette approche, on conserve la structure des unités de protocoles qui traversent le réseau; le code est placé de manière dynamique dans certains éléments, étendant ainsi les fonctionnalités du réseau. Le problème le plus important de cette approche est la normalisation de l'interface de contrôle d'un nœud programmable. Le développement des réseaux programmables a été entrepris par la communauté OPENSIG [67] et poursuivis dans le projet P1520 de l'IEEE [6]. Ce développement a porté sur la normalisation des interfaces des commutateurs ATM, des routeurs IP et des réseaux de télécommunications sans fil. Ceci va pouvoir permettre de manipuler les dispositifs physiques du réseau comme étant des objets informatique distribués avec des interfaces programmables bien définies. Ces interfaces ouvertes permettent aux fournisseurs de services de manipuler les états du réseau en utilisant des outils intergiciels tels que CORBA pour construire et gérer de nouveaux services [11]. Dans ce type de réseaux, le plan de signalisation est programmé en dehors du flux de données (*out-of-band control*) par l'opérateur ou le fournisseur de services et les plans de données et de supervision restent inchangés.

L'inconvénient est que ces modifications ne peuvent se faire dynamiquement, c'est-à-dire que les routeurs ne doivent pas être connectés au réseau ou alors qu'ils doivent être redémarrés pour que le nouveau service puisse fonctionner.

b. Réseaux actifs

Un réseau actif est un réseau dans lequel tout ou une partie de ses composants dans les différents plans (signalisation, supervision, données) sont programmables dynamiquement par des entités tierces (opérateur, fournisseur de services, applications ou usagers). Cette approche est plutôt issue des travaux sur l'insertion de services applicatifs dans le réseau Internet, par opposition aux réseaux à signalisation ouverte qui sont issus de travaux sur la signalisation pure dans les réseaux de télécommunication de type ATM. Elle étend le concept de programmation en partant du principe que virtuellement tout usager peut concevoir, déployer et utiliser un nouveau service de manière dynamique dans le réseau. Nous distinguons encore trois sous-familles dans les réseaux

actifs selon le cas où le déploiement des services se fait dans le flux de données, dans un flux séparé ou de manière hybride. La figure 2.1 montre la place des réseaux actifs dans les différentes familles de réseaux programmables.

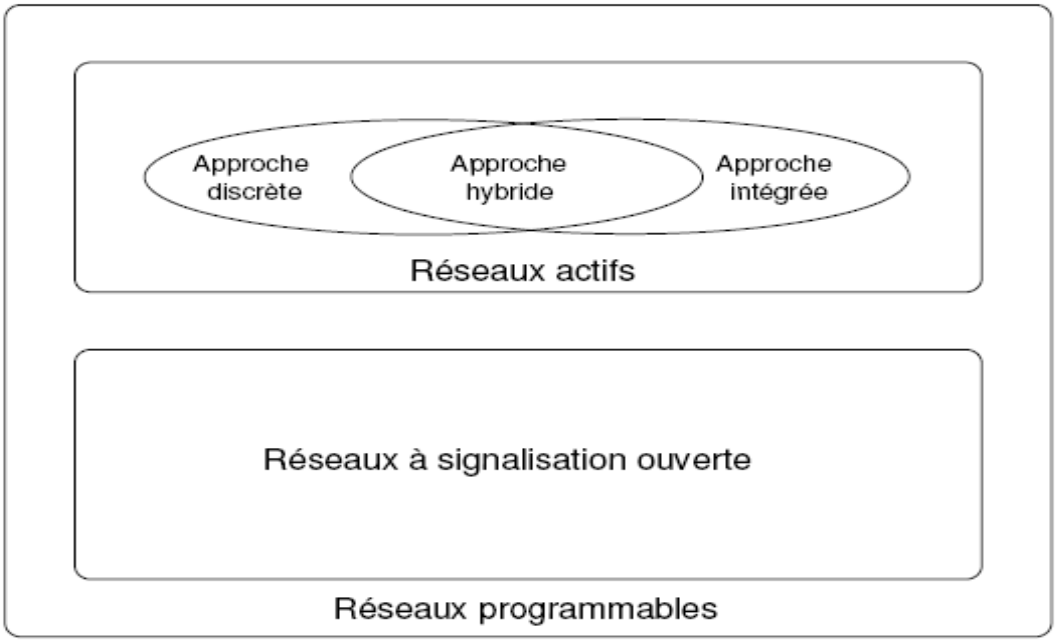

Figure 2.1 : Réseaux programmables

3. Fonctionnement des réseaux actifs

Les réseaux actifs reposent sur des équipements réseau ouverts et programmables ayant la possibilité de déployer dynamiquement des programmes qui seront appliqués à la volée sur les paquets de données. On peut aussi bien déployer de nouveaux protocoles dynamiquement et faire du traitement sur les paquets. Le caractère ouvert du réseau actif lui laisse également une liberté bien plus grande sur les choix d'architecture et de technologie. L'approche réseaux actifs peut être affinée en trois classes: celles qui se basent sur une approche paquet actif ou capsule (approche intégrée), celles qui se basent sur une approche nœud actif (ou approche discrète) et celles qui combinent les deux approches précédentes.

3.1 Approche intégrée

Une approche intégrée est une approche réseau actif, dans laquelle le déploiement des services dans les nœuds est conceptuellement intégré dans le flux usager (*in-band control*), et la durée de vie d'un service est liée à celle du trafic affecté. Les programmes peuvent être composés

d'instructions qui exécutent des calculs de base sur le contenu de la capsule ou invoquer des primitives qui permettent d'accéder aux ressources externes de l'environnement de transition. Il est clair que les capsules peuvent porter efficacement le code, seulement lorsque celui-ci est relativement simple et restreint.

Le code contenu dans une capsule peut invoquer :

- l'accès à l'environnement (pour demander l'adresse du nœud, la table de routage, etc.),
- la manipulation des capsules (accéder à l'entête et aux données des capsules),
- des opérations de contrôle (permettre à des capsules de créer d'autres capsules, de se copier ou de s'effacer elles-mêmes),
- des opérations pour manipuler les états des objets définis par une application pendant une courte durée de vie.

a. Avantages et limites de l'approche

Un avantage non négligeable de cette approche est la tolérance aux pannes. En effet, une architecture répartie est toujours moins fragile qu'une architecture centralisée. Néanmoins, le problème de l'approche intégrée est que le code inclus dans chaque capsule augmente le trafic du réseau et engendre donc une perte de la bande passante. Cela est d'autant plus vrai que chaque capsule transporte son protocole. En effet, si un grand nombre de capsules utilisant le même protocole transitent sur le même routeur, il y a une forte redondance de code. Si nous considérons le protocole ping-pong actif : les données transportées sont très faibles alors que le code doit contenir au moins l'algorithme de routage qui permet à la capsule d'aller de l'´emetteur au récepteur, de revenir et de calculer le temps de parcours. Nous constatons ici que le code à transporter est prédominant sur les données, ce qui n'est pas très intéressant. Dans ce cas le paquet non-actif homologue sera beaucoup plus petit.

b. Exemple

Comme exemple illustratif de cette approche, nous citons le projet *Smart Packets*, développée chez BBN [54]. Ce projet est motivé par la nécessité, de réduire la croissance exponentielle du trafic de gestion qui a tendance à augmenter avec le nombre de nœuds agissant ainsi négativement sur la capacité de traitement de réseau, et le besoin de décentraliser les fonctions de gestion pour soulager les stations de supervision. Afin de répondre à ces besoins, *Smart Packets* utilise intelligemment le paradigme de réseau actif pour décentraliser la gestion. Le

principe de l'approche est le suivant: un paquet de supervision est un programme véhiculé dans un paquet actif. La principale restriction sur ces paquets est la taille maximale qui est fixée à 1KO pour se tenir dans une trame Ethernet. Dans les différents nœuds du réseau, l'approche propose une architecture d'accueil et d'exécution de ces fonctions de gestion.

Le format d'un paquet *Smart Packets* est donné dans la figure 2.2. On y retrouve l'entête du paquet IP, l'encapsulation dans le standard de paquets actifs ANEP (*Active Network Encapsulation Protocol*) et finalement le paquet *Smart Packet*. Celui-ci est composé d'un numéro de version, d'un type, d'un contexte et des données véhiculées. Les différents types possibles sont :

- Program : le paquet contient un programme envoyé vers un agent ;
- Data : le paquet contient des données envoyées vers la station de supervision depuis un agent ;
- Error : le paquet véhicule une erreur d'un agent vers un superviseur ;
- Message : le paquet véhicule des données informationnelles plutôt que le code exécutable ;

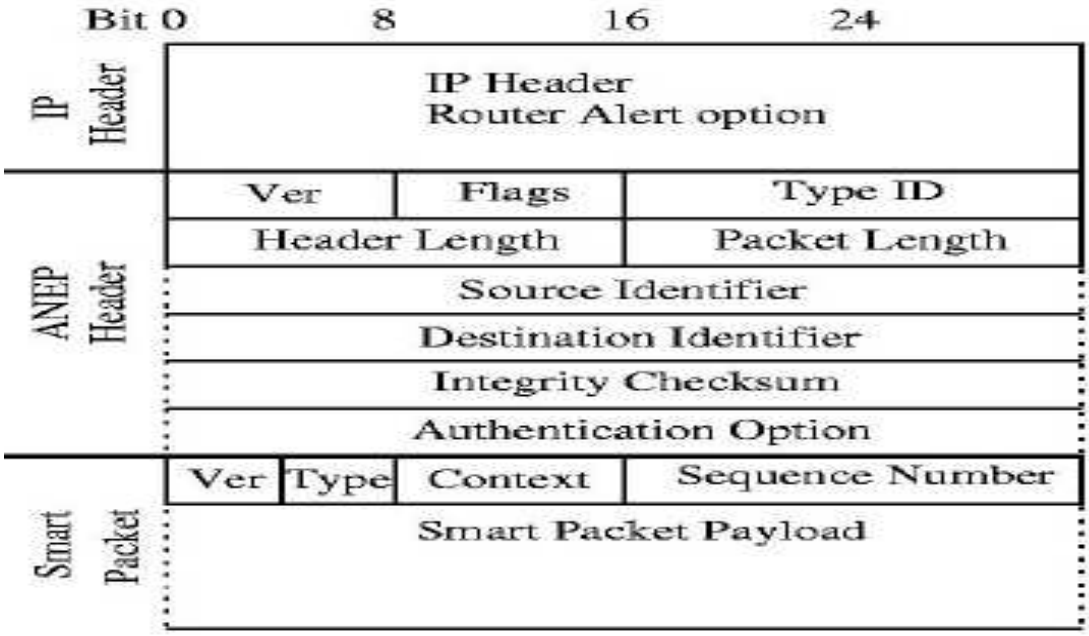

Figure 2.2 : Format d'un paquet *Smart Packet.*

3.2 Approche discrète

Dans l'approche nœud actif, les services sont également déployés dynamiquement dans les nœuds du réseau mais, en dehors des flux de données que ces services traitent. Ceci se rapproche fortement du concept de réseaux à signalisation ouverte. La différence avec ces derniers, repose

sur le fait que dans l'approche discrète les composants de service sont déployés le plus souvent grâce à des techniques de code mobile. Cette programmation externe permet également de donner le nom d'approche discrète à cette famille qui intègre également un sous-ensemble des propositions autour de la signalisation ouverte. Conceptuellement, l'approche paquet actif est beaucoup plus ouverte que celle du nœud actif, mais dans la première, les codes déployés sont nécessairement de taille plus petite (tout simplement pour des raisons de performance) et ne peuvent donc être utilisés que dans un nombre restreint de situations.

D'un point de vue conceptuel, un réseau actif est construit à partir des composants fondamentaux : le nœud actif et le paquet actif. Le nœud actif est composé d'un système d'exploitation du nœud (*Node Operating System* ou *Node OS*) responsable de l'allocation et de la gestion des ressources du nœud actif (la bande passante des liens, le CPU et le stockage) et fournit un ensemble de capacités de base utilisées par des environnements d'exécution (*Execution Environment,* EE) pour implémenter une machine virtuelle, java par exemple. Un environnement EE offre une interface de programmation qui peut être programmée par les applications actives (AA), dynamiquement chargées et interprétées par les environnements EE afin de mettre en œuvre de nouveaux services. L'organisation générale de ces composants est illustrée dans la figure 2.3. Il existe trois composants logiciels qui permettent de construire une architecture active pour supporter une exécution d'une AA :

- L'Application d'utilisateur (AA).
- L'Environnement d'Exécution (EE).
- Le système d'exploitation du nœud (node OS).

a. Avantages et limites de l'approche

L'avantage de l'approche discrète est que chaque protocole n'est téléchargé qu'une seule fois, il n'y a donc pas de perte de bande passante liée au transport des protocoles. Cependant, le déploiement d'une application est plus complexe puisque celle-ci doit au départ envoyer les protocoles qu'elle va utiliser à un nœud actif. Ainsi, les protocoles peuvent être vérifiés et signés avant d'être diffusés, ce qui est particulièrement intéressant en termes de sécurité. Mais l'approche discrète est moins flexible que l'approche intégrée.

b. Exemple

L'architecture du nœud actif définie à Georgia Tech [10] constitue un exemple illustratif de cette approche. C'est une approche typiquement nœud actif qui nécessite l'implantation de services

actifs dans le noyau d'un système d'exploitation, il s'agit de Solaris. Elle permet à des utilisateurs, à l'aide des options IP spécifiques à leur approche, de choisir parmi un ensemble de services, ceux qui seront appliqués dans chaque nœud traversé. La fonction à appliquer sur un paquet est définie dans l'option, par un identificateur unique. Cette fonction est paramétrable sur le nœud et les paramètres effectifs sont fournis dans l'entête de chaque paquet. Le nœud est caractérisé par une table de fonctions avec des pointeurs sur le code, permettant de dispatcher tout paquet entrant vers la fonction de traitement correspondante. Le nœud permet également l'accès à certains de ses états. L'extensibilité de cette architecture n'est pas facile. Cela est dû principalement à l'objectif des analyses qui ont été menées sur ce réseau, visant plus l'aspect performance de services plutôt que l'aspect lié à leur déploiement dynamique.

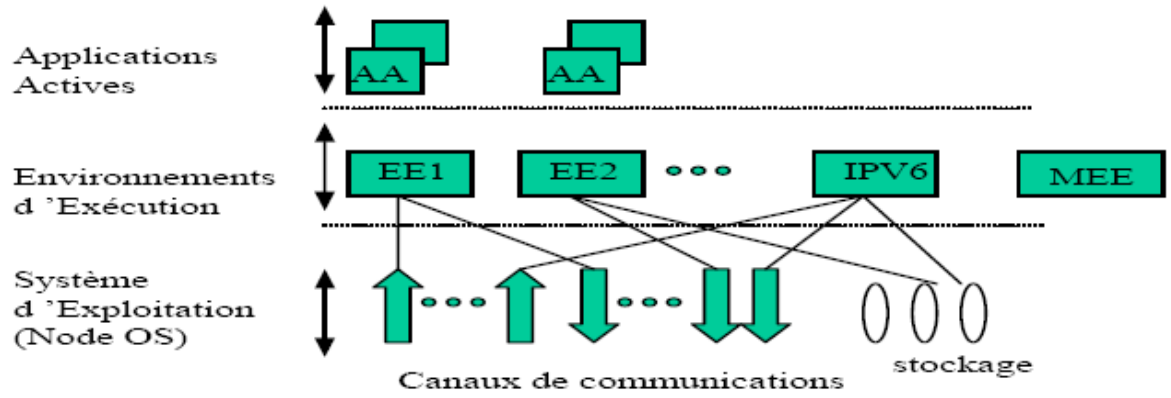

Figure 2.3 : Architecture d'un nœud actif.

3.3 Approche mixte

Les avantages de chacune des deux méthodes précédentes sont intéressants, cependant leurs limites sont tout de même importantes. Ainsi, leur fusion a entrainé l'apparition des méthodes mixtes.

Un exemple illustratif de l'approche mixte, est celui conçu et réalisé dans les labos du MIT, dite ANTS (*Active Network Transfer System*) [27]. Son principe permet le déploiement dynamique des services et des protocoles que les applications utilisent à travers tous les routeurs actifs traversés par les flux associés. Pour cela, ANTS offre une architecture de nœud qui permet de supporter des protocoles multiples, ainsi qu'un mécanisme permettant le déploiement dynamique de ces protocoles au sein de flux de données de l'application. La figure 2.4 présente l'architecture ANTS dont les principaux composants sont :

- Le système de répartition de code qui distribue les routines d'expédition aux nœuds où elles sont censées s'exécuter. Il est défini par un identificateur, un ensemble de classes décrivant le comportement (groupe du code) et des unités de distribution de ce code appelé capsule.
- La capsule est l'unité de base de programmation du réseau. Elle est employée non seulement pour véhiculer les données de l'application entre les nœuds actifs, mais aussi le code d'un protocole à déployer sur les nœuds. La capsule de données contient une adresse source et une adresse destination, un champ de TTL, un champ de version d'ANTS, un champ comportant l'adresse du dernier nœud actif traversé, un champ de données sur lesquelles le protocole référencé doit être appliqué. Par contre, dans une capsule de transfert de code, les données sont le code du protocole ou une référence au code du protocole. Ce dernier, peut être transporté dans plusieurs capsules successives. La figure 2.5 présente le format d'une capsule ANTS.
- Le nœud actif représente l'environnement d'exécution (EE) qui permet la réception, le traitement, et l'envoi d'une capsule suivant le protocole prédéfini par l'application ayant émis la capsule.

Les principaux apports de l'approche ANTS sont :

- Le déploiement dynamique rend l'architecture tolérante aux pannes et robuste vis-à-vis du passage à l'échelle en mettant en place un mécanisme de chargement à la demande depuis le nœud précédent. L'arrivée d'une capsule sur un nœud demandant un protocole qui n'est pas présent, nécessite une demande du nœud actif à l'émetteur de cette capsule de lui transmettre le code associé. La tolérance aux pannes est assurée par le routage dynamique qui redirige les flux dans le cas où un routeur n'est plus opérationnel. Le passage à l'échelle est également assuré par le mécanisme de chargement de protocole à la demande. D'une part, seuls les nœuds qui ont besoin le chargent. D'autre part, le chargement se fait systématiquement de proche en proche, c.-à-d. depuis le nœud actif précédent.
- La combinaison des deux approches (paquet actif et nœud actif) dans ANTS permet d'éviter à chaque capsule de données d'inclure le code permettant son traitement. Le mécanisme de cache de code fourni dans le nœud actif maintient les codes pendant des durées adaptées aux flux. Ceci est particulièrement intéressant pour les flux importants sur lesquels toutes les capsules subissent le même traitement dans les nœuds, par exemple, les flux MPEG.

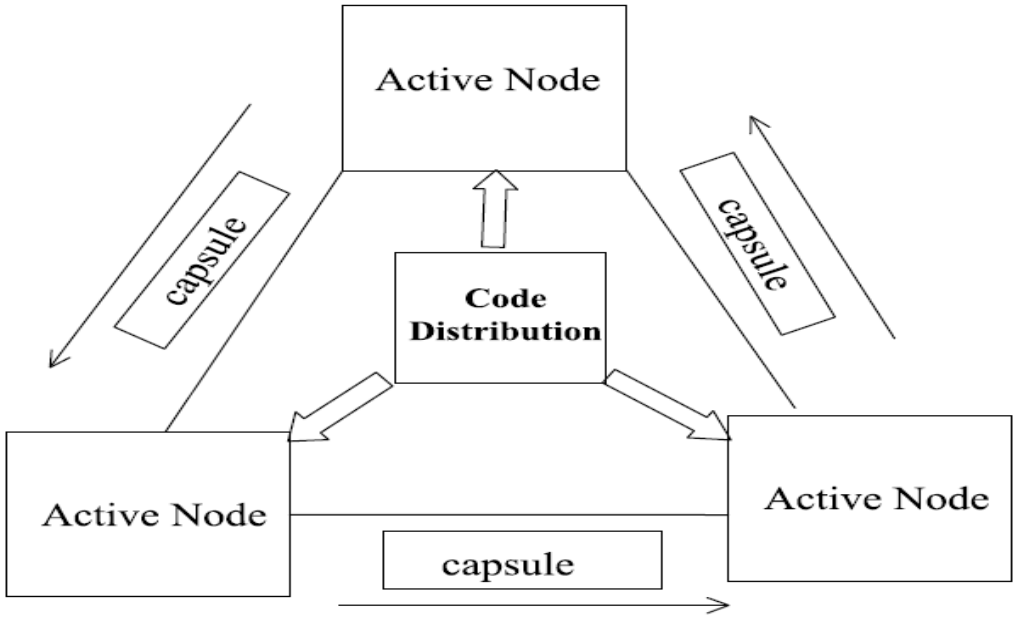

Figure 2.4 : Architecture ANTS (*Active Network Transfer System*).

Le principal inconvénient d'ANTS réside dans l'absence d'une stratégie globale de gestion et d'attribution de ressources pour les protocoles au niveau réseau et niveau nœud également.

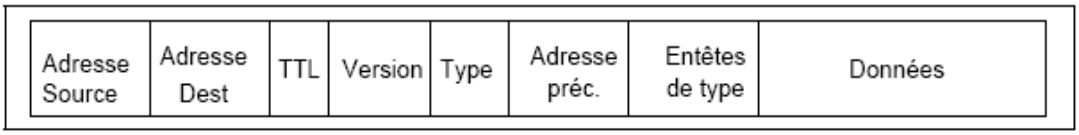

Figure 2.5 : Format d'une capsule ANTS.

4. Services actifs

En parallèle à la multitude de projets qui se concentrent sur l'architecture et l'implémentation des nœuds actifs ainsi que sur des langages spécifiques pour la composition des services actifs, une nouvelle approche a émergé comme une alternative aux réseaux actifs. Celle-ci prône le déploiement de code utilisateur dans le réseau mais de manière restreinte, en limitant ce déploiement à un niveau applicatif sans modifier le réseau sous-jacent. Dans cette approche, les applications peuvent être supportées par une architecture de services programmables construits au dessus des services d'Internet existants. Elle permet aux utilisateurs de télécharger et d'exécuter du code (application) à des endroits stratégiques dans le réseau. Ce code ou application est appelé "service actif". Le principal argument en faveur d'une telle approche est qu'elle est nettement plus facile à déployer et maîtriser dans des réseaux actuels que les solutions

opérant à des niveaux inférieurs. Ainsi, cette infrastructure de services actifs enrichit le domaine des réseaux actifs et elle résout le problème de compatibilité avec le réseau Internet actuel.

Comme exemple illustratif de cette approche, nous considérons l'infrastructure, proposée par E. Amir et al, appelée AS1 (*Active Service version* 1) [2]. Ses composants de base de cette (voir la figure 2.6) sont les serveurs d'agents, les agents de service ainsi que les protocoles de contrôle et de sollicitation. Les serveurs d'agents sont appelés cadre de services qui offre l'environnement d'exécution et l'interface d'accès aux ressources pour les agents de services. Ces derniers implantent la logique de service pouvant être réalisée par plusieurs agents distribués sur le réseau. Un client demande au cadre de services l'instanciation d'un agent de services (1). Cette demande est transmise via un canal multipoint et en utilisant un protocole spécifique appelé ASCP (*Active Service Control Protocol*). Le cadre de services instancie un agent en fonction de ses disponibilités actuelles et du type de service demandé sur l'un de ses serveurs. L'agent, une fois installé, met en place le service (flux, connexion, traitement) et offre une interface de contrôle au client pour lequel il travaille (2).

Les auteurs ont implémenté le modèle AS1 en se basant sur la plateforme "MASH" qui est un interpréteur du langage Tcl [68] avec des capacités de multimédia en temps réel et de réseaux. L'interface proposée par AS1 est un ensemble de classes d'objets Tcl qui peuvent être invoqués par des services actifs (des programmes Tcl). Un service de transcodage de vidéo a été ainsi réalisé comme un service actif.

L'intérêt principal de cette approche est de fournir une architecture propre pour le déploiement de services applicatifs. La limite d'une telle approche est la restriction au niveau applicatif. Ceci implique que de nombreux services ne sont pas traitables (contrôle de congestion, le routage, etc.)

5. Applications des réseaux actifs

Les nouvelles applications émergeantes nécessitent d'exploiter de nouveaux services qui ne sont pas encore présents dans les réseaux traditionnels. L'introduction des réseaux actifs rend possible le contrôle dynamique permettant de confectionner et d'adapter des services par rapport à l'état courant du réseau. Au lieu d'utiliser la solution de bout-en-bout offerte par les réseaux traditionnels, les efforts ont été engagés pour améliorer les performances d'un large spectre d'applications au moyen des réseaux actifs. L'objectif de cette section est de présenter des exemples d'applications mettant en avant l'avantage et l'efficacité des réseaux actifs. La question qui se pose est de savoir si les changements induits par l'approche réseau actif vont

accroitre réellement les performances des applications qui s'exécutent au sein d'un réseau. Avant de présenter les points clefs et les principales caractéristiques de ces applications, nous résumons les points de vue de plusieurs chercheurs sur la relation qui existe entre réseaux actifs et ce qu'il convient d'appeler l'argument de bout-en-bout.

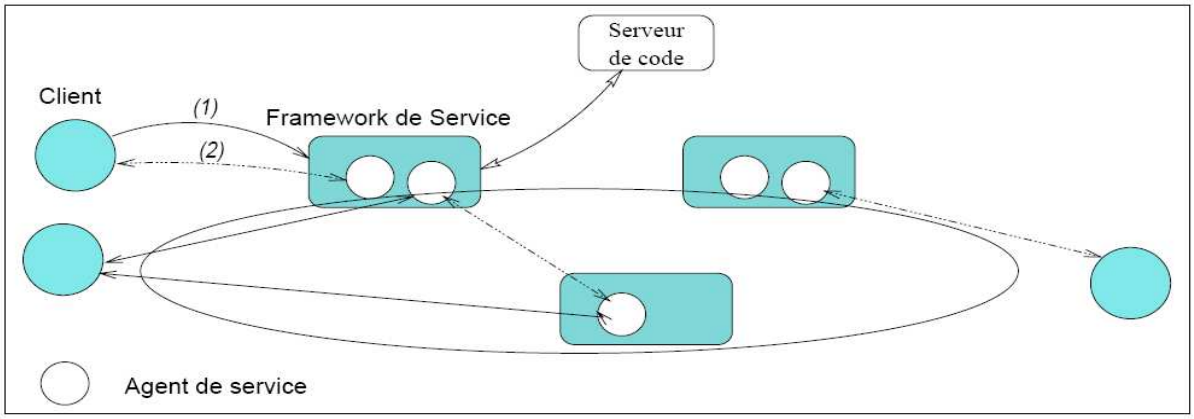

Figure 2.6 : Architecture d'un service actif.

5.1 Solution de bout-en-bout

Cette approche se base sur le fait que les décisions finales restent toujours du ressort de l'utilisateur et que tenter de le suppléer au sein du réseau devient redondant. Par conséquent, les fonctions de contrôle doivent, dans la mesure du possible, toujours être placées à l'extérieur du réseau [26]. Les points de vue des auteurs divergent, certains pensent que l'approche active remet en cause celle du bout-en-bout, d'autres soutiennent que les réseaux actifs abondent dans le sens de l'approche de bout-en-bout. Ces confrontations ont été publiées dans *Active and Programmable Networks* paru dans *IEEE Network* [12].

Le premier point de vue est donné par Bhattacharjee, Calvert, Zegura concluant que les réseaux actifs ne sont pas en conflit avec l'approche de bout-en-bout. Ceci vue est motivé par le fait que les différents services ne peuvent être efficacement déployés et améliorés que s'ils disposent d'informations uniquement disponibles et accessibles au sein du réseau. Comme par exemple, le lieu et l'instant où survient une congestion, ou les endroits où surviennent des pertes de paquets dans un arbre multicast. Ces auteurs considèrent que l'approche active est une conséquence naturelle de celle du bout-en-bout.

Le second point de vue est donné par Partridge, Strayer, Schwartz et Jackson où une évaluation de l'utilité des réseaux actifs dans le contexte du bout-en-bout a été faite pour chaque niveau des couches réseau Internet. Cette évaluation a montré que le principe du bout-en-bout va à

l'encontre d'une mise en œuvre d'un service actif dans la couche Internet. L'objectif de l'Internet est d'offrir une connectivité universelle et un moyen de communication entre un très grand nombre d'équipements hétérogènes. Ces auteurs pensent qu'il existe des cas pour lesquels les réseaux actifs n'exposent pas correctement ce service. Ils avancent comme argument que le chemin d'un paquet peut être affecté par le code qu'il véhicule (voire même le code d'un autre paquet). Les chances qu'un paquet arrive à destination sont réduites du fait puisqu'il peut se trouver dans environnement d'exécution endommagé, non à jour, et que les programmes véhiculés soient erronés et que leur débogage soit rendu extrêmement délicat. En outre leur analyse reste très favorable concernant toutes autres couches.

Le dernier point de vue revient à Reed, Saltzer et Clark, en examinant l'approche de bout-en-bout à la lumière du nouveau contexte introduit par les réseaux actifs. Ils remarquent que pour conserver un niveau de transparence élevé au sein des réseaux, le principe de bout-en-bout exige la définition d'une sémantique de toute fonctionnalité active. Celle-ci doit être définie et confinée avec beaucoup de précautions d'une manière à ce que les interactions entre différents utilisateurs partageant une même couche basse soient prédites par le concepteur qui utilise les services et les fonctionnalités de la couche active. Un manque de prédiction entraine un surcout pour tous les utilisateurs, y compris ceux qui ne font pas appel aux services actifs. Même si l'idée de l'actif n'est pas complètement écartée par le principe de bout-en-bout, les auteurs n'ont pas trouvé l'exemple mettant à jour l'impact indéniable de l'actif tout en garantissant une sémantique simple, flexible et transparente nécessaire pour une utilisation des couches basses.

5.2 Communications de groupe (multicast)

Divers travaux ont été entrepris pour l'étude de protocoles multicast dans les réseaux actifs. Ces derniers ont été utilisés pour la mise en œuvre d'un protocole inspiré de PIM (*Protocol Independant Multicast*) montrant ainsi leur facilité d'emploi pour le développement et déploiement de protocoles réels malgré leur complexité [60]. Ils ont été aussi utilisés pour fiabiliser les protocoles multicast, le cas qui nous intéresse dans cette dissertation, et qui permet de résoudre d'une manière élégante les problèmes d'implosion des acquittements et de la localité de perte. Il semble que les réseaux actifs offrent des améliorations prometteuses dans le domaine des protocoles multicast fiable, car ils font appel à un point clé qui est la possibilité de prendre une décision au sein même du réseau. Plusieurs expérimentations ont été réalisées, elles seront détaillées dans le chapitre suivant.

5.3 Applications multimédia audio/vidéo

La distribution des flux multimédia à tous les membres d'un groupe multicast est un problème ardu qui est du, non seulement à l'hétérogénéité des chemins au sein du réseau entre l'émetteur et les récepteurs, mais aussi au caractère extensible du groupe pour un grand nombre de récepteurs. L'approche classique résout les problèmes de l'hétérogénéité et de l'extensibilité par l'emploi d'une méthode de régulation orienté récepteur. Dans cette approche, plusieurs flux sont émis par la source, ceux-ci sont appropriés pour des classes de chemins du réseau dont les caractéristiques et les débits appartiennent à un certain éventail de valeurs. Chaque récepteur s'abonne au flux qui peut être supporté par le chemin le reliant à la source. Une autre alternative est offerte par l'approche réseaux actifs, elle permet de préciser l'endroit où l'adaptation et la régulation de flux peuvent être effectuées au sein du réseau. Dans cette approche, un seul flux est émis par la source à tous les récepteurs qui sont abonnés un groupe multicast. Les routeurs multicast détiennent les informations leur permettant de réduire le flux en cas de congestion d'un lien en aval. Les nœuds internes de l'arbre multicast diffèrent la régulation de flux jusqu'à ce qu'ils effectuent la duplication du paquet, afin de pouvoir traiter chaque lien en aval d'une manière indépendante. Ceci garantit que chaque récepteur reçoive un flux selon les caractéristiques du chemin le reliant à la source. La régulation est directement imposée par les conditions d'encombrement dans le réseau, ce qui permet de répercuter immédiatement toute baisse ou augmentation de la bande passante.

5.4 Application au Web

Plusieurs exemples simples d'applications peuvent bénéficier de nouveaux services introduits par l'approche réseaux actifs [59]. Ces exemples possèdent le même domaine d'application, à savoir l'Internet.

a. Quotations en directe

Le premier exemple concerne l'obtention des quotations boursières à travers le Web. Obtenir un accès rapide à des informations à jour est un point crucial surtout les périodes pendant lesquelles le serveur est chargé. Les techniques de cache traditionnelles s'avèrent inappropriées, puisque celles-ci ne stockent pas ce genre d'information. Ce dernier étant des données dynamiques et la granularité des objets stockés dans le cache Web (i.e., une page Web entière) est mal approprié car chaque boursicoteur génère une requête avec quelques titres d'actions, sur plusieurs centaines de titres disponibles, le nombre de combinaisons de pages Web est énorme. Dans l'approche

réseaux actifs, un système de cache met en œuvre une stratégie dédiée aux besoins de l'application. Le service actif fait une mise en cache des titres dans les nœuds du réseau, avec une granularité fine, ce qui permet de répondre à des requêtes concernant des titres sans tenir compte de leur ordre dans lequel la requête globale a été émise. Le service actif peut associer une périodicité minimale de mise à jour à chaque requête concernant un titre donné. L'idée générale est donc d'assurer un cache des quotations au niveau des nœuds du réseau. Les requêtes des clients seront interceptées dans les nœuds pour chercher si elles peuvent être satisfaites localement avec les données du cache.

b. Vente aux enchères

Le second exemple concerne la vente aux enchères de divers produits à travers l'Internet (voyages, matériel informatique, etc.). Un serveur qui propose ce genre de service doit collecter toutes les offres des clients et doit aussi informer le client du prix actuel de l'objet convoité. A cause des délais dans le réseau, un client peut proposer un prix en dessous du prix actuel de l'objet. Dans l'approche traditionnelle, le serveur est amené à collecter et de traiter toutes les offres. Par contre, l'approche réseaux actifs peut permettre de filtrer les offres trop basses. Ainsi lorsque le serveur devient surchargé, celui-ci active le mécanisme de filtrage dans les nœuds actifs du voisinage qui vont rejeter les offres basses tout en prévenant le client. Ces nœuds sont informés périodiquement, des mises à jour des prix des articles, par le serveur.

c. Caches Web

Les applications clients génèrent un trafic important d'Internet afin de récupérer des informations localisées sur des serveurs dispersés dans des différents endroits du réseau. Le cache des objets dans un voisinage proche des clients permet de réduire le trafic généré au sein d'un réseau ainsi que le temps nécessaire pour récupérer les objets. Le problème majeur d'un système de caches est de pouvoir localiser un objet et de savoir transmettre des requêtes entre les caches. L'Approche traditionnelle utilise une infrastructure basée sur une hiérarchie statique de caches SQUID [69]. Une configuration manuelle des clients est nécessaire pour pouvoir accéder à un cache particulier de la hiérarchie (le proxy du client). Si une requête échoue au niveau du proxy, la recherche est entamée dans les caches de même niveau ainsi que le cache père. Si elle n'est pas satisfaite, une nouvelle requête HTTP est générée avec comme cible le cache père et le processus est répéter d'une manière récursive dans toute la hiérarchie des caches jusqu'à ce que le requête soit satisfaite. Une fois l'objet récupéré, celui-ci sera mis en cache tout le long du chemin inverse le menant vers le cache proxy. Un tel système présente plusieurs limites dans le

fait qu'il est statique, que sa configuration soit manuelle et que la racine de la hiérarchie soit un goulot d'étranglement. L'approche réseaux actifs peut jouer un important dans ce cadre pour améliorer le cache Web. Une solution proposée dans [30] consiste à maintenir une table d'indirection au sein des nœuds actifs du réseau afin de pouvoir diriger les requêtes vers le cache le plus proche. Une autre solution qui consiste à mettre en œuvre et maintenir une distribution uniforme de petits caches ayant des informations sur le contenu des caches présents dans leur voisinage [5].

5.5 Autres types d'applications

Il existe d'autres domaines d'applications tels que : la sécurité [19], le déploiement à la volée [1].

6. Conclusion

Ce chapitre a fait un tour d'horizon de la technologie des réseaux actifs en présentant d'une part, les limites de l'approche traditionnelle et d'autre part, certaines solutions actives pouvant être proposées, ainsi que les nouveaux défis qu'elles posent. Le concept des réseaux actifs est considéré comme une piste prometteuse pour accroître la capacité du réseau, en permettant une grande réactivité et adaptabilité de l'infrastructure réseau aux nouveaux besoins des applications. Les réseaux actifs offrent des mécanismes qui accélèrent le déploiement de nouveaux protocoles et services, ainsi que l'augmentation de la flexibilité et l'innovation dans les réseaux, sans changer les structures actuelles, ni faire attendre la normalisation d'un nouveau protocole.

Par ailleurs, nous avons vu que les réseaux actifs s'appliquent à la communication multicast pour résoudre le problème inhérent à la fiabilité de ce genre de communication. Dans le chapitre suivant, nous allons présenter un état de l'art des protocoles qui traitent la fiabilité multicast et plus particulièrement ceux basés sur les réseaux actifs.

Chapitre 3

Multicast fiable

1. Introduction

Le multicast IP fournit au niveau réseau un support efficace pour un grand nombre d'applications : dissémination de données, simulation interactive distribuée, vidéoconférence, application coopératives, etc. Certaines de ces applications, en plus de l'efficacité de routage, nécessitent des services supplémentaires, et en particulier la fiabilité totale ou partielle des échanges. Ce problème est infiniment plus complexe à traiter en multicast qu'en unicast vu que la source ne connaît en général pas le nombre et l'identité des récepteurs (contrairement à TCP). Plusieurs points ont été identifiés comme critiques par la norme Internet RFC 2357 [41]:

- Le facteur d'échelle "la scalabilité" : il s'agit essentiellement du passage à l'échelle en termes de nombre de récepteurs.
- Le contrôle de congestion : les sessions multicast doivent avoir un comportement équitable vis à vis des flux TCP.
- La sécurité : ce point est adressé par le groupe SMUG, *Secure MUlticast Group*, de l'IETF. Déjà complexe en unicast, la sécurité appliquée au multicast l'est encore plus du fait de problèmes tels le dynamisme du groupe et l'anonymat des membres.

La notion de scalabilité, en termes de nombre de récepteurs, est appliquée au service de transmission multicast fiable pose de nombreux problèmes dont il faut en tenir compte en offrant :

- Un trafic de contrôle scalable : Il est clair qu'envoyer un acquittement positif (ACK) à chaque paquet transmis (à la TCP) n'est guère envisageable. Envoyer un acquittement négatif (NAK) systématiquement à chaque paquet non reçu pose également problème. Ainsi, si la perte a lieu près de la source, alors un très grand nombre de récepteurs vont générer un NAK ce qui risque l'écroulement de la source.
- Des retransmissions scalables : Transmettre les paquets perdus en unicast, pour celui qui a fait la demande n'est clairement pas envisageable. Mais, transmettre systématiquement en multicast à tout le monde les paquets perdus n'est guère aussi envisageable.
- Une gestion de l'hétérogénéité : Au sein d'un groupe important de récepteurs, qui ont de forte chance d'être largement hétérogènes du point de vue du réseau d'accès et des capacités de traitement.

Plusieurs protocoles de multicast fiable ont été développés pour garantir la fiabilité au niveau transport et pour assurer le contrôle de congestion du réseau sous-jacent. Ces protocoles ont été

regroupés selon les deux critères suivants: en terme de classes *sender-initiated* ou *receiver-initiated* et en terme d'approches hiérarchiques ou celles basées sur les temporisateurs [63]. La figure 3.1 montre cette classification ainsi que quelques protocoles de transport multicast fiable illustratifs : RMTP (*Reliable Multicast Transport Protocol*) [48], PGM (*Pragmatic General Multicast*) [56], TMTP (*Tree-based Multicast Transport Protocol*) [62], LBRM (*Log-Based receiver Reliable Multicast*) [24], LMS (*Light-weight Multicast Services*) [47], LGC (*Local Group Concept*) [23], SRM (*Scalable Reliable Multicast*) [20], AER (*Active Error Recovery*) [29], ARM (*Active Reliable Multicast*) [31], et DyRAM (*Dynamic Replier Active reliable Multicast*) [40].

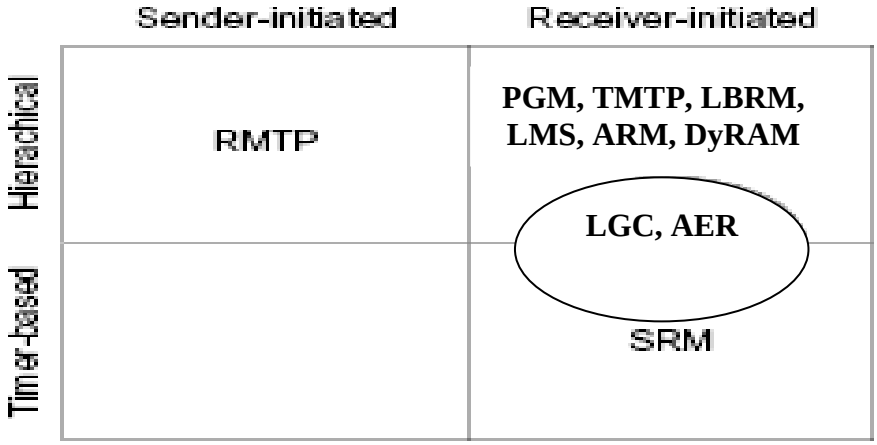

Figure 3.1 : Classification des protocoles de multicast fiable.

Le reste du chapitre est organisé comme suit : La section 2 présente les classes de recouvrement des pertes utilisées dans la stratégie de bout-en-bout. La section 3 décrit les approches de recouvrement local. La section 4 détaille les protocoles de multicast fiable actif les plus connus dans la littérature. La section 5 introduit l'approche FEC (*Forward Error Correction*) dans les protocoles de multicast fiable. Enfin le chapitre est terminé par une conclusion.

2. Stratégie de bout-en-bout

Pour assurer la fiabilité, les premiers protocoles de transport multicast adoptaient une approche de bout en bout qui fait intervenir uniquement la source et les récepteurs. La fiabilité consiste à détecter les pertes, puis de faire les retransmissions appropriées pour cela, on doit répondre aux questions suivantes : qui détecte la perte ? Qui prend en charge la retransmission (ou réparation) du paquet perdu ?

Ceci a mené à l'apparition de deux classes de protocoles, à savoir : la classe de protocoles *sender-initiated* et la classe de protocoles *receiver-initiated.*

2.1 Classe de protocoles "*sender-initiated*"

Dans cette classe de protocoles, seule la source est responsable de la détection et de la réparation des pertes. Chaque récepteur doit en effet signaler le succès de la réception du paquet de données par l'envoi d'un acquittement positif (ACK), entrainant ainsi une grande charge au niveau de la source qui doit traiter tous les ACKs. Ces messages de contrôle surchargent aussi le réseau en terme de bande passante consommée. Le problème s'aggrave plus lorsqu'il existe un grand nombre de récepteurs. Le défi est donc, d'assurer la fiabilité tout en garantissant le passage à l'échelle en présence d'un nombre important de récepteurs. Les protocoles de cette classe souffrent du problème de l'implosion des ACKs au niveau de la source. Une solution qui peut réduire le nombre de message de contrôle, c'est de décharger la source de la tâche de détection des pertes et de rendre chaque récepteur responsable des pertes qu'il subit.

2.2 Classe de protocoles "*receiver-initiated*"

En revanche, cette classe de protocoles attribue la responsabilité de la détection de pertes aux récepteurs. Les protocoles emploient les NAKs pour informer la source des pertes subies par un récepteur. Chaque récepteur peut détecter la perte d'un paquet sur l'expiration d'un délai de garde ou la réception d'un paquet altéré ou hors séquence. Une fois que le récepteur détecte une perte, il transmet un NAK à la source qui est censée retransmettre le paquet perdu. L'utilisation des NAKs au lieu des ACKs réduit d'une manière considérable la charge des messages de contrôle au niveau de la source ainsi que la bande passante consommée du réseau. Cependant, cette classe souffre également du problème de l'implosion de NAKs surtout lorsqu'un paquet de données est perdu par un grand nombre de récepteurs.

2.3 Limites de la stratégie de bout-en-bout

D'une manière générale, les protocoles de multicast fiable de la stratégie de bout-en-bout ne permettent pas le passage à l'échelle (le problème de la scalabilité) en présence d'un groupe important de récepteurs :

- Le problème de l'implosion des acquittements au niveau de la source persiste, puisque cette dernière subit toujours la charge de recouvrement des pertes.

- Le problème d’exposition des récepteurs existe toujours, car il n’existe aucun moyen pour limiter la portée de retransmission à un sous-groupe de récepteurs.
- La latence de recouvrement de perte est toujours aussi importante, puisqu’il n’existe aucun moyen pour la réduire.
- Une solution à ces problèmes est d'adopter une stratégie de recouvrement local des pertes qui sera assurée par une entité intermédiaire.

3. Stratégie de recouvrement local

Les protocoles de multicast fiable adoptent une stratégie de recouvrement local fondée sur deux approches : l’approche basée sur les temporisateurs (*timer based approach*) et l’approche basée sur une structure hiérarchique (*structured based approach*).

3.1 Approche basée sur les temporisateurs

L’approche basée sur les temporisateurs, utilise ces derniers pour résoudre le problème d'implosion de NAKs. Lorsqu’un récepteur détecte une perte de paquet, il attend pendant un temps aléatoire et puis diffuse une demande de réparation. Le temporisateur de l’hôte le plus proche du point de perte est probablement celui qui expire le premier. Cet hôte est élu comme récepteur demandeur (*requestor*), il envoie un NAK demandant le paquet de données perdu. Afin d’éviter la redondance, les hôtes qui ont également perdu le paquet de données, entendent cette demande et suppriment les siens. Ce comportement empêche le problème d'implosion de NAKs. N'importe quel hôte qui a une copie du paquet de données peut répondre à cette demande, celui-ci est appelé répondeur (*replier*). Cette approche est particulièrement robuste quant au changement de topologie, puisqu’elle ne dépend d’aucun nœud intermédiaire pour faire la suppression de NAKs et de retransmissions. Le protocole SRM [20] est l’exemple typique basé sur cette approche (voir figure 3.1). L'inconvénient de l'approche réside dans le problème de localité de réparation qui peut être allégé en organisant les membres d’un groupe dans une hiérarchie de zones administrativement limitées [55].

3.2 Approche basée sur une structure hiérarchique

L'approche hiérarchique partitionne l'arbre de distribution multicast en sous-groupes formant une hiérarchie à plusieurs niveaux qui admet la source comme racine. Chaque sous-groupe a un leader, qui peut être soit un récepteur désigné [48], un serveur dédié [24] ou un routeur désigné [47, 56]. Ce leader assure le recouvrement local des pertes qui se produisent au sein du sous-

groupe en maintenant en cache les paquets de données et agrège également les NAK identiques. L'implosion en feedback est limitée, parce que chaque leader manipule un nombre restreint de récepteurs. Limiter localement le flux en feedback et les retransmissions permet de préserver la bande passante et réduit le problème d'exposition des récepteurs. La latence de recouvrement des pertes est aussi réduite puisque les réparations viennent d'un représentant proche du point de la perte. LBRM, RMTP, PGM, TMTP, LMS [48, 24, 56, 62 et 47] sont des exemples typiques de protocoles basés sur cette approche (voir figure 3.1). Cependant, l'inconvénient principal de l'approche provient de la gestion d'un grand nombre de routeurs spécialisés ou de récepteurs et l'échec d'une machine créerait un énorme fardeau administratif. L'approche réseaux actifs permet d'offrir une solution à ce genre de problème.

3.2 Approche hybride

Le protocole LGC [23] peut être considéré comme un protocole hybride des deux approches précédentes (voir figure 3.1). Il regroupe les récepteurs dans des groupes locaux et désigne pour chaque groupe local un leader. Ce dernier est responsable de réparer les pertes qui se produisent au niveau des récepteurs appartenant à son groupe local. Les groupes locaux forment un arbre hiérarchique multi niveaux qui permet l'échange de données parmi eux. La perte d'un paquet de données est d'abord récupérée à l'intérieur du groupe local en suivant la manière du protocole SRM [20]. Le leader du groupe local demande le paquet perdu de la source ou d'un autre leader hiérarchique si est seulement si aucun des membres de son groupe ne tient une copie du paquet perdu.

4. Stratégie basée sur les réseaux actifs

Les réseaux actifs constituent une nouvelle approche réseau où les routeurs peuvent faire des traitements plus adaptés aux applications sur les paquets qui les traversent [57]. Ces dernières déploient, dynamiquement, dans le réseau les services et les protocoles qu'elles utilisent dans les routeurs actifs. Cette approche réseau offre une solution plus générale et plus flexible que celle adoptée par les protocoles PGM, LMS [56, 47] qui utilisent des routeurs dédiés et figés pour assurer un transport multicast fiable. Ainsi, et dans le but d'assurer le multicast fiable, des services actifs peuvent être implémentés, au niveau des routeurs, à des emplacements stratégique du réseau. Le rôle de ces services consiste principalement à faire :

- Le cache des paquets de données au niveau des routeurs actifs, pour assurer localement le recouvrement des pertes. Ceci garantit une bonne répartition de la charge de recouvrement

des pertes entre la source et les routeurs actifs et réduit d'une manière considérable la latence de recouvrement des pertes.

- L'agrégation et/ou la suppression locale des NAKs identiques évitant ainsi le problème de l'implosion des NAKs en feedback.
- La retransmission en multicast partiel (On parle de retransmission en subcast) en limitant la portée aux récepteurs qui ont effectivement perdu le paquet de données. Ceci permet de résoudre le problème d'exposition des récepteurs.
- L'élection d'un répondeur parmi les récepteurs pour assurer le recouvrement local des pertes à la place du routeur actif réduisant ainsi sa charge.

Plusieurs protocoles de transport multicast fiable basés sur l'approche réseaux actifs ont été développés (voir figure 3.1). Nous détaillons dans cette section les plus cités au niveau de la littérature. Les protocoles AER (*Active Error Recovery*) [29], ARM (*Active Reliable Multicast*) [31], et DyRAM (*Dynamic Replier Active reliable Multicast*) [40].

4.1 Protocole AER (*Active Error Recovery*)

Le protocole AER [29] est un protocole de transport développé afin de fiabiliser les transmissions multicast IP. Pour cela le protocole combine les deux approches hiérarchique et celle basée sur les temporisateurs, pour résoudre le problème de la scalabilité (voir figure 3.1). Dans cette structure hiérarchique proposée par AER, les services de réparation sont placés au niveau d'entités spécialisées et dédiées appelées les serveurs de réparation. Ces serveurs sont directement reliés aux routeurs de l'arbre de distribution multicast. Ils représentent les seules entités actives du réseau. Les serveurs de réparation sont chargés de faire le cache des paquets de données qui transitent à travers l'arbre multicast pour assurer le recouvrement local des pertes par la suite. Ainsi, les serveurs de réparation sont placés dans des emplacements stratégiques dans le réseau, de façon à minimiser le nombre de paquets de contrôle circulant dans l'arbre de distribution multicast et ce, en éliminant les acquittements redondants qui les traversent. Afin que les serveurs de réparations jouent leur rôle dans le réseau, il est nécessaire d'invoquer leurs services actifs. Dans ce cas, ils se joignent au groupe multicast pour lequel les paquets de données sont envoyés.

a. Identification des serveurs de réparation

Le protocole AER met en place un mécanisme de signalisation qui permet, d'une part, d'identifier les serveurs de réparation au sein du réseau et d'activer/désactiver les services actifs des serveurs, et d'autre part, la prise en charge des chemins asymétriques et leur changement. La source transmet périodiquement des messages de chemin source SPM (*Source Path Messages*) au groupe multicast concerné par les paquets de données. Chaque message SPM contient l'adresse IP du dernier serveur de réparation à travers lequel il a transité. L'adresse initiale est celle de la source. Grâce à ce mécanisme, les récepteurs pourront, d'une part, identifier le serveur de réparation le plus proche, et d'autre part, établir le chemin inverse vers la source.

b. Technique de recouvrement des pertes

Le protocole AER est basé sur la classe *receiver-initiated* dans laquelle la détection de pertes est attribuée aux récepteurs. Un récepteur détectant la perte d'un paquet, émet un NAK au serveur de réparation le plus proche. Cependant, pour pouvoir résoudre le problème de l'implosion des NAKs, AER enrichie sa technique de recouvrement de perte en introduisant un système de temporisation au niveau des récepteurs et serveurs de réparation intermédiaires. A chaque détection d'une perte, les récepteurs (respectivement le serveur de réparation) déclenchent un temporisateur. Afin d'éviter toute confusion entre les récepteurs ayant détecté la même perte simultanément, la durée du temporisateur est aléatoire. Lors de l'expiration du délai d'attente et si le récepteur n'a reçu aucun NAK identique durant ce délai de garde, alors le récepteur transmet son NAK au serveur de réparation ascendant le plus proche, et déclenche un temporisateur d'attente de réparation. Si un NAK parvient au récepteur (respectivement le serveur de réparation) avant l'expiration de son temporisateur, il supprime le sien et se comporte comme s'il l'a transmet. A l'expiration du délai d'attente d'une réparation, alors qu'aucune réparation n'a été reçue, le récepteur (respectivement le serveur de réparation) retransmet le même NAK à son serveur de réparation le plus proche et réinitialise le délai d'attente d'une réparation. Cette technique permet de réduire le nombre de paquets de contrôle transmis par les nœuds du réseau, victimes d'une perte du paquet de données. Lors de la réception d'un NAK au niveau d'un serveur de réparation, celui-ci peut avoir deux comportements distincts :

- Si le serveur de réparation possède le paquet de données demandé, il le diffuse à l'ensemble de récepteurs de son sous-arbre.
- Si le serveur de réparation ne possède pas le paquet de données demandé, il diffuse le NAK reçu à l'ensemble des récepteurs de son sous-arbre pour qu'ils puissent supprimer leurs

NAKs identiques localement et transmet un NAK à son serveur de réparation le plus proche si ce dernier ne l'a pas déjà fait.

Lorsqu' un serveur de réparation reçoit un paquet de réparation de son ascendant, il garde une copie dans son cache et le diffuse aux membres de son sous-arbre.

c. Stratégie de la gestion du cache

Le protocole AER ne prévoit aucune stratégie pour la gestion du cache au niveau des serveurs de réparation. Cependant, il est mentionné qu'un serveur de réparation qui ne reçoit pas de message SPM pendant une certaine période de temps, procède à la désactivation de ses services actifs et à la suppression de tous les paquets de données stockés en cache.

d. Limites du protocole AER

Etant basé sur la classe *receiver-initiated*, ce protocole, comme c'est le cas pour de nombreux autres protocoles, n'assure un service parfait de recouvrement local que sur l'hypothèse d'un cache de capacité infinie.

4.2 Protocole ARM (*Active Reliable Multicast*)

Le protocole ARM [31] est un protocole de transport multicast fiable qui se base sur l'aspect actif de différents nœuds du réseau. Le protocole s'articule sur l'approche hiérarchique à plusieurs niveaux pour résoudre le problème de scalabilité. Il conçoit l'existence d'un protocole de routage multidestinataire qui construit et maintient un arbre de distribution multicast optimal dont la racine est la source. Cet arbre est réparti en sous-arbres, chaque sous-arbre admet un routeur actif comme racine et ses membres peuvent être aussi bien des récepteurs que des routeurs actifs. Chaque routeur actif effectue une mise en cache des paquets de données pour réparer les pertes qui se produisent au sein d'un sous-groupe. Les récepteurs ne sont pas censés connaitre la topologie de groupe ni conserver les paquets de données pour la réparation. A la différence avec d'autres approches qui exigent des membres du groupe multicast de savoir la topologie de ce dernier (les positions relatives des autres récepteurs), le protocole ARM n'est pas affecté par les changements de topologie de groupe.

a. Technique de recouvrement des pertes

Le protocole ARM appartient à la classe des protocoles *receiver-initiated* qui attribue la détection de pertes aux récepteurs. Pour cela, il emploie trois stratégies pour assurer le

recouvrement des pertes. D'abord, il contrôle la duplication des NAKs pour éviter le problème d'implosion. Les routeurs actifs du protocole ARM sont déployés de façon à ce qu'ils permettent de réduire le trafic généré par les NAKs, traversant des liens qui sont privés de la bande passante. En second lieu, un schéma de recouvrement de perte est instauré d'une manière à ce que la charge de réparation soit bien répartie et que la latence soit réduite. Les routeurs actifs font le cache des paquets de données évitant ainsi de chercher les paquets de données perdus à partir de la source. Cette manière de faire, assure un recouvrement local plus rapide sans imposer aucune surcharge inutile à la source et au groupe en entier. Troisièmement, pour préserver la bande passante, les routeurs sont permis d'effectuer un multicast partiel pour limiter la portée de la retransmission aux récepteurs, qui ont effectivement perdu le paquet de données, permettant ainsi d'éviter le problème d'exposition des récepteurs.

Dans le protocole ARM, les NAKs multiples de différents récepteurs sont agrégés au niveau des routeurs actifs tout le long de l'arbre de distribution multicast. Si tous les routeurs sont actifs, et l'agrégation des NAKs se fait en temps utile, la source ne reçoit qu'un seul NAK par paquet perdu. Ceci, est dû principalement à la fusion des NAKs ; autrement la source recevra plus d'un NAK. La source répond au premier NAK par une retransmission en multicast du paquet perdu. Par la suite, la source ignore tous les NAKs identiques demandant le même paquet pendant une période de temps bien fixée. Cette période est le RTT (*Round-Trip Time*) prévu entre la source et le récepteur le plus éloigné dans le groupe. Il est possible que les NAKs et le paquet retransmis soient également perdus ; dans ce cas, le récepteur doit renvoyer un NAK, s'il ne reçoit pas la réparation dans un certain délai. Celui-ci est supposé au moins égal au RTT entre le récepteur lui-même et la source originale du paquet de données. Pour identifier que le NAK entrant est nouveau, il utilise un compteur pour indiquer le nombre de fois que le récepteur a demandé le paquet perdu (voir figure 3.2). La source maintient le plus haut NAK lié à chaque demande. Si elle reçoit un NAK ayant un compteur plus élevé que celui maintenu, la source suppose que la retransmission précédente a été perdue et retransmet, en multicast, une nouvelle fois le paquet perdu.

b. Multicast partiel

Dans l'objectif d'éviter le problème d'exposition des récepteurs, les routeurs actifs du protocole ARM effectuent un multicast partiel des paquets de réparation, de sorte qu'ils soient envoyés aux récepteurs qui les ont précédemment demandés. Pour cela, chaque routeur actif maintient une structure de réparation qui permet d'enregistrer les liens sur lesquels proviennent toutes

demandes concernant un paquet de données. Lorsqu'un routeur actif reçoit le paquet de réparation, il consulte cette structure et envoie le paquet sur les liens enregistrés dans cette dernière.

c. Stratégie de la gestion du cache

La gestion du cache des routeurs actifs dans le protocole ARM est faite par l'utilisation d'un champ TTL du cache (*cache time-to-live*) spécifié dans l'entête du paquet de données (voir la figure 3.2). Celui-ci fixe la durée de vie d'un paquet de données original ou de réparation dans un cache du routeur actif. Ce temps est estimé approximativement à un RTT du routeur au récepteur le plus loin en aval. Passé ce temps, le paquet sera supprimé du cache du routeur.

Multicast Group Address	Source Address	Sequence Number	Cache TTL	NACK Count

Figure 3.2 : Format d'un entête du paquet de données ARM.

d. Limites du protocole ARM

- L'élimination de duplication des NAKs identiques au niveau des routeurs est fondée sur la supposition que tous les routeurs sont actifs, si le nombre de routeurs actifs diminue à travers l'arbre de distribution multicast, cela va augmenter le risque du problème d'implosion.
- L'utilisation d'un TTL qui fixe la durée de vie d'un paquet dans un cache risque de supprimer des paquets qui sont toujours utiles au recouvrement local de perte affectant ainsi la latence de réparation.

4.3 Protocole DyRAM (*Dynamic Replier Active reliable Multicast*)

DyRAM [40] est un protocole de transport multicast fiable actif, basé sur l'approche hiérarchique (voir figure 3.1). Il adopte un schéma de recouvrement local fondé sur la classe *receiver-initiated,* les récepteurs sont chargés de détecter les pertes qui se produisent, et retransmettent éventuellement les paquets de données perdus. Ce protocole se distingue des autres protocoles de multicast fiable actif par ses fonctionnalités innovatrices :

- Pour chaque paquet perdu, une élection dynamique d'un répondeur est effectuée. Celui-ci est élu parmi les récepteurs qui ont correctement reçu le paquet de données.

- L’émulation des acquittements positifs grâce à l’ajout de nouveaux champs au niveau des entêtes des paquets de contrôle du protocole.

a. Services actifs de DyRAM

a.1. Sauvegarde d’états

Les différents services actifs peuvent être implémentés, au niveau des routeurs actifs, simplement en maintenant les informations concernant les paquets de données et les NAKs reçus. Cet ensemble d'informations est uniquement identifié par une source et l'adresse multicast :

- Pour chaque NAK reçu, le routeur crée ou simplement met à jour une structure d'état NS (*NAK State*) qui a une durée de vie limitée. La structure NS est supprimée dès la réception de la réparation correspondante. Par conséquent, cette durée de vie peut être estimée par le temps maximum demandé pour que la réparation correspondante soit reçue par le routeur. Cette structure maintient pour chaque NAK reçu, une liste de multicast partiel qui garde la trace des liens ayant signalé la perte. Cela, permettra par la suite, d’envoyer le paquet de réparation uniquement aux récepteurs qui ont effectivement perdu le paquet de données.
- Afin de réaliser une élection d’un répondeur d’une manière rapide et précise, chaque routeur actif maintient une structure d'état LS (*Link State*) qui contient les informations concernant l’état d’un lien tels que : le numéro du dernier paquet reçu en ordre, le numéro du dernier paquet reçu et la variation du RTT sur ce lien.

a.2. Suppression des NAKs redondants

A la réception d'un NAK valide pour un paquet de données, le routeur actif crée la structure NS correspondante, initialise un temporisateur NST (*NAK Suppression Timer*) et expédie le NAK à son ascendant dans l’arbre de distribution multicast. Pendant ce temps, tout NAK reçu sera seulement employé pour la mise à jour des structures LS et NS et puis il sera ignoré. Un NAK est considéré valide, s’il a été reçu pendant qu’il n'y a aucun NST actif. Ceci, permettra à DyRAM d’éliminer les NAKs redondants au niveau des routeurs actifs.

a.3. Emulation des acquittements positifs

Parmi les fonctionnalités novatrices de DyRAM, on trouve l’émulation des acquittements positifs (voir figure 3.3). Cela se fait grâce aux champs l_o, l_r introduits au niveau des paquets de

contrôles du protocole. Ces champs ont l'utilité d'informer les routeurs actifs de l'état du lien d'où provient chaque NAK.

Sachant que, l_o : représente le numéro de séquence du dernier paquet de données reçu en ordre, et l_r : représente le numéro de séquence du dernier paquet de données reçu.

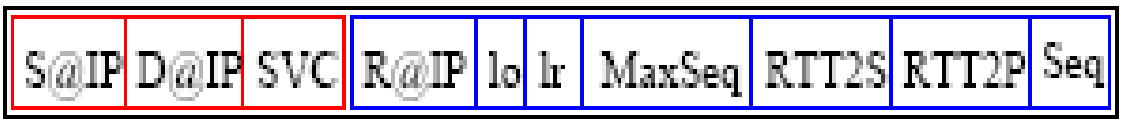

Figure 3.3 : Format d'un paquet de contrôle DyRAM.

a.4. Election dynamique d'un répondeur

DyRAM est le premier protocole qui a introduit la notion d'élection dynamique d'un répondeur par paquet. Ce mécanisme permet d'une part, de répartir la charge de recouvrement local des pertes entre les récepteurs d'un sous-groupe et d'autre part, de décharger les routeurs actifs de ce recouvrement. Pour élire un répondeur parmi les récepteurs, le candidat doit satisfaire l'un des deux critères suivants selon l'ordre indiqué :

- Le récepteur ayant acquitté positivement la bonne réception du paquet de donnée demandé. DyRAM n'utilise pas l'envoi explicite d'acquittement positif, mais plutôt, grâce à la fonctionnalité de l'émulation de ce dernier.
- L'état du lien menant à ce récepteur, indique un poids maximum par rapport aux autres récepteurs candidats.

Ainsi, lorsqu'il n'existe aucun récepteur qui a acquitté positivement le paquet perdu, le champ de poids de la structure LS sera utilisé pour élire un répondeur. Ce poids est proportionnel à la capacité de réponse affectée au lien descendant. L'estimation du poids est calculée par la formule suivante :

$$weight = \kappa\ (-RTTvar) + (1-\kappa)\ \frac{1}{RTT \times (lr - lo + 1)}$$

Sachant que :

- *K* : représente un paramètre de poids à choisir dans l'intervalle [0,1].
- RTT_{var} : exprime la dernière variation du RTT éprouvée par ce lien. Cette valeur varie inversement avec la bande passante disponible du lien considéré.
- *RTT* : représente le RTT entre le routeur actif et récepteur de ce lien.

- (l_r - l_o +1) : exprime une estimation grossière du taux de perte relatif à un lien, où l_r est le numéro de séquence du dernier paquet de données reçu et l_o est le numéro de séquence du dernier paquet de données reçu dans l'ordre sur ce lien.

b. Comportement des différentes entités de DyRAM

b.1. Comportement de La source

Avant chaque émission, la source stocke le paquet de données dans le cache, puis elle le transmet à une adresse multicast à laquelle sont inscrits tous les récepteurs d'un groupe. Lorsque la source reçoit un NAK, elle vérifie si le temporisateur NST (*NAK Suppression Time*) est désarmé, si c'est le cas, cela signifie que c'est un NAK valide, et par conséquent, la source déclenche le temporisateur NST. Pendant ce délai, la source entame une période d'ignorance des NAKs identiques qui dure un temps égal au RTT de la source jusqu'au récepteur le plus éloigné du groupe. A l'expiration de ce délai, la source retransmet le paquet de données demandé à l'adresse multicast. La gestion du cache au niveau de la source, s'effectue grâce au champ "l_o" des NAKs transmis par les routeurs. Chaque routeur agrège les NAKs reçus de ses récepteurs en un seul NAK ayant une valeur du champ "l_o", qui est égale au minimum de tous les "l_o" reçus. Ainsi, la valeur du champ "l_o" qui atteint la source indique le numéro de séquence maximum des paquets de données qui ont été correctement reçus par la totalité des récepteurs. Cependant, la source prend le minimum de tous les champs "l_o" reçus par ses liens descendants et élimine du cache tous les paquets de données qui ont un numéro de séquence inférieur ou égal à la valeur du champ "l_o" minimal. Par conséquent, la source inclut cette valeur au niveau du champ "Maxseq" de tous les paquets de données qu'elle transmettra par la suite. Ceci permettra de mettre à jour les caches des récepteurs du groupe.

b.2. Comportement des récepteurs

Lorsqu'un récepteur reçoit un paquet de données original, dont le numéro de séquence correspond au paquet attendu, il fait la mise à jour de numéro de séquence en réception. Le cas contraire signifie qu'une perte s'est produise, dans ce cas le récepteur envoie immédiatement un NAK à son ascendant dans l'arbre multicast et il initialise un délai de garde qui doit être égal au RTT du récepteur à la source. Si, au cours de cette période, il reçoit le paquet, il fait la mise à jour de numéro de séquence en réception. Mais, quand le délai de garde expire et le paquet n'a pas été reçu, le récepteur renvoie un autre NAK et réinitialise de nouveau le délai de garde. En plus de la détection par ordre d'arrivée, DyRAM adopte une stratégie complémentaire de

détection, qui repose sur l'initialisation d'un temporisateur à chaque réception d'un paquet de données, si le temporisateur arrive à sa fin sans que le récepteur ne reçoive le paquet suivant, celui-ci déduit alors qu'une perte s'est produise et retransmet un NAK relatif au paquet attendu.

Un récepteur peut être élu comme répondeur. Celui-ci peut alors recevoir des NAKs, indiquant les pertes survenues au niveau des autres récepteurs. Dans ce cas, le répondeur se comporte de la même manière que la source, lorsqu'il reçoit un NAK. Il est clair que les récepteurs ne sont pas dotés d'une capacité de cache infinie. Ainsi, une bonne gestion du cache s'impose. Les récepteurs s'appuient sur l'information transmise par la source pour mettre à jour leurs fenêtres de réception. Le champ "Maxseq" contenu dans les paquets de données représente le numéro de séquence maximum du paquet ayant été correctement reçus par l'ensemble des récepteurs du groupe. Le récepteur peut, ainsi, supprimer sans risque tous les paquets du cache qui ont un numéro de séquence inférieur ou égale à la nouvelle valeur du champ "Maxseq" reçue.

b.3. Comportement des routeurs actifs

Dans DyRAM, les routeurs actifs contribuent, aussi bien au recouvrement local, qu'à la détection des pertes qui se produisent à leurs niveaux. Pour cela, chaque routeur actif maintient une structure TL (T*rack List*) qui contient les numéros de séquence du dernier paquet reçu en ordre et du dernier paquet reçu sur chaque lien des descendants, ces deux informations permettent au routeur actif d'en déduire la liste des paquets perdus sur chacun des liens descendants.

A la réception d'un paquet de données attendu, le routeur actif fait la mise à jour du numéro de séquence et transmet le paquet à ses descendants dans l'arbre multicast. Dans le cas contraire le routeur se rend compte qu'il a perdu un paquet, il lance une procédure de réparation en envoyant un NAK à son ascendant et il initialise un délai de garde qui doit être égal au RTT du routeur actif à la source. Si le délai de garde arrive à sa fin et que le paquet n'est pas reçu alors le routeur actif envoie un autre NAK et initialise à nouveau le délai de garde.

A la réception d'un NAK provenant d'un lien descendant, le routeur actif procède de la manière suivante :

- Il vérifie si le temporisateur NST n'est pas actif, cela veut dire que c'est le premier NAK signalant la perte d'un paquet (un NAK valide). Dans ce cas, le routeur crée la structure NS correspondante, initialise un temporisateur NST, et consulte la structure LS pour trouver un lien qui a acquitté positivement le paquet demandé. Si ce lien existe, le routeur actif transmet immédiatement le NAK du paquet demandé sur ce lien (vers le récepteur élu répondeur), sinon un temporisateur DTD (*Delay To Decide*) est initialisé pour élire un répondeur.

- Si le NAK reçu n'est pas un NAK valide, le routeur actif met à jour les structures NS et LS et entame des vérifications supplémentaires.
  - Si la liste de subcast de la structure NS contient tous les liens descendants de ce routeur, cela veut dire que tous les récepteurs ont perdu le paquet. Dans ce cas, le temporisateur DTD est désactivé et le NAK est transmis vers la source.
  - Sinon le récepteur demandant le paquet perdu est ajouté à la liste de subcast de la structure NS et le NAK sera ignoré.

A la réception d'un paquet de réparation, le routeur actif consulte la structure NS correspondante et extrait la liste des liens sur lesquels la perte s'est produite et le transmet sur ces liens évitant ainsi les récepteurs qui ont déjà reçu le paquet. Si la liste de subcast est absente, le cas où la perte est détectée par le routeur lui-même, celui-ci transmet le paquet vers tous les liens descendants.

A l'expiration de la durée de l'élection d'un répondeur (DTD), le routeur actif procède à l'élection d'un répondeur parmi les descendants qui n'ont pas signalé la perte pour ce paquet. L'élection se fait selon la valeur du poids attribuée à chacun de ses liens. Un poids maximum permet au lien d'être élu comme répondeur pour cette perte de paquet. L'élection d'un répondeur se fait à chaque expiration d'un temporisateur DTD.

c. Limites du protocole DyRAM

- Le maintient et la gestion des structures d'états (NS, LS, LT) au niveau des routeurs actifs peuvent entrainer une surcharge sur ces derniers surtout en terme de temps consacré à leur traitement.
- Le temps de l'élection d'un répondeur peut devenir considérable surtout si l'élu ne possède pas le paquet perdu, ce qui influe négativement sur la latence de recouvrement des pertes.

5. Approche basée sur le code FEC (*Forward Error Correction*)

Le code FEC est une approche alternative qui permet la réparation des pertes de paquets dans un réseau sans avoir recours à la retransmission de ces paquets par un mécanisme ARQ (*Automatic Repeat reQuest*)[8, 7, 55, 46, 51, 52]. Un paquet FEC peut ainsi se substituer à n'importe quel paquet de données qui aurait pu être perdu, évitant ainsi la génération des NAKs. Deux schémas complémentaires peuvent être utilisés :

- Proactif : la source suppose qu'il y aura un certain taux de pertes et ajoute systématiquement dans le flux de données transmis des paquets FEC supplémentaires. Le problème ici, est de savoir à l'avance combien de FEC faut il ajouter. Il ya un équilibre à déterminer entre la

surcharge de transmission et la capacité de recouvrement de pertes. Les conditions réseaux évoluant dynamiquement, cet équilibre devrait en théorie constamment être remis en cause.

- Réactif : demander la retransmission d'un ou plusieurs paquets, chaque récepteur demande la retransmission d'un certain nombre de paquets. La source transmettra, en fait, un certain nombre de paquets FEC du fait de leur propriété de se substituer à n'importe quel paquet de données et donc satisfaire plusieurs récepteurs ayant subit des pertes différentes.

En raison de son efficacité, plusieurs protocoles de multicast fiable utilisent le code FEC selon l'un des deux schémas ci-dessus (le deuxième schéma est le plus systématiquement utilisé).

5.1 Limites du code FEC

- Le code FEC ne peut pas à lui seul garantir une fiabilité totale, puisque si le nombre de paquets originaux perdus est supérieur au nombre de paquets FEC, les récepteurs ne peuvent pas reconstruire les paquets originaux.
- Le gaspillage de la bande passante du réseau qui est dû principalement à la redondance des paquets FEC.

Une solution à ces problèmes est d'introduire le code FEC dans les protocoles de multicast fiable actif. Cette alliance des deux approches à donner naissance à une nouvelle classe de protocoles appelée APES (*Active Parity Encoding Services*).

5.2 Protocoles APES (Active Parity Encoding Services)

Les protocoles RS (*Repair services*) basé sur les services de réparation et le FEC/ARQ (*hybrid parity encoding and automatic repeat request*) [50, 53] sont deux approches alternatives de recouvrement des pertes qui permettent de réduire les besoins en bande passante des protocoles de multicast fiable. L'approche RS utilise pour cela, les serveurs de réparation, localisés à des emplacements stratégiques du réseau. Ils stockent les données pour assurer un recouvrement local des pertes et traitent les NAKs transmis par les récepteurs. Étant donné que, chaque serveur de réparation est responsable d'un nombre variable de récepteurs dans le réseau, et que la capacité de stockage des paquets de réparation au niveau de leurs caches, est limitée. Il est important de réduire les besoins en buffer de cette approche.

L'approche FEC est une approche de bout-en-bout qui utilise la technique des codes correcteur pour produire des paquets de réparations spéciales, afin de répondre à des éventuelles pertes de données au niveau des récepteurs. Typiquement, l'ensemble des paquets de réparation qui permet de recouvrir n'importe quelle perte au niveau des récepteurs, est plus petit que celui des

retransmissions nécessaires. Quoique les récepteurs, dans cette approche, sont obligés d'effectuer des opérations de codage et de décodage supplémentaires pour pouvoir utiliser ou transmettre les données originales.

Ainsi, cette nouvelle classe de protocoles à pour objectif de réduire, non seulement la bande passante consommée, mais aussi, le cache et le traitement à l'intérieur d'un réseau.

6. Conclusion

Dans ce chapitre nous avons présenté les différentes approches qui ont été élaborées pour traiter le problème de la fiabilité en multicast. La plupart de ces approches contribuent à résoudre deux types de problèmes inhérents à ce genre de communication : l'implosion en feedback et la localité de réparation. Dans la classe des protocoles fondée sur l'approche de bout-en-bout, il n'existe aucun moyen ni pour éviter le problème de l'implosion en feedback ni pour localiser les récepteurs concernés par la perte, le problème s'aggrave davantage avec des groupes importants. La classe des protocoles basée sur l'approche de recouvrement local a permis d'éviter le problème de l'implosion en feedback et à pu réduire l'effet d'exposition des récepteurs. Cependant, le recouvrement local est assuré par des nœuds intermédiaires spécialisés (récepteurs désignés, serveurs ou routeurs dédiés) où la panne d'un nœud risque de compromettre la fiabilité multicast. Une solution plus générale et plus flexible, est celle introduite par l'approche réseaux actifs, où les routeurs contribuent au recouvrement des pertes à des emplacements stratégiques du réseau très proches du lieu de la perte. Le tableau 3.1 : récapitule les différentes stratégies utilisées par les protocoles de multicast actif pour résoudre le problème de scalabilité tel que : l'implosion en feedback, la répartition de la charge de recouvrement des pertes, l'exposition des récepteurs.

Table 3.1. Tableau comparatif

	AER	ARM	DyRAM
Implosion des acquittements	Agrégation et suppression	Agrégation	Agrégation
Répartition de charge de recouvrement	Source et serveur de réparation	Source et routeur actif	Source et récepteur élu
Portée de retransmission	Contrôlée par le serveur de réparation	Contrôlée par le routeur actif	Contrôlée par le routeur actif

Néanmoins, les protocoles de multicast fiable actif les plus connus en littérature, adoptent une approche de recouvrement local basé sur la classe *receiver-initiated*. Cette dernière présente quelques limites qui seront présentées dans le chapitre suivant.

Chapitre 4

Protocole AMRHy

1. Introduction

Fournir une délivrance multicast fiable et efficace pour les applications de dissémination de données à grande échelle est un défi. Plusieurs protocoles ont été élaborés pour résoudre le problème de la fiabilité multicast dans les réseaux à délivrance dite *best effort* tel que l'Internet. Ils traitent ce problème en imposant un compromis entre le délai d'acheminement et la capacité de la bande passante. Néanmoins, les solutions proposées par ces protocoles se limitent à une échelle réduite. L'introduction du concept des réseaux actifs dans la fiabilité multicast a orienté les recherches vers le déploiement des services actifs au niveau des routeurs pour élaborer des protocoles de multicast fiable, qui s'appliquent à grande échelle, offrant ainsi, une solution plus générale et plus flexible. La plupart des protocoles de multicast fiable actif existants adoptent une approche de recouvrement local qui se base sur la classe "*receiver-initiated*". Cette dernière attribue la détection de pertes aux récepteurs indépendamment du lien sur lequel la perte se produit. Contrairement à ces protocoles, le protocole que nous proposons est un protocole de transport multicast fiable basé sur le concept des réseaux actifs appelé AMRHy acronyme du "***A**ctive **M**ulticast **R**eliable **Hy**brid protocol* ". Notre protocole combine les classes "*sender-initiated*" et "*receiver-initiated*". Cette combinaison de classes permet d'assurer une répartition équitable de la charge de recouvrement des pertes entre la source et les récepteurs, avec la contribution des routeurs actifs. Dans cette combinaison de classes, la source détecte les pertes qui se produisent sur les liens source (liens proches de la source) et les récepteurs détectent celles qui se produisent sur les liens terminaux (les liens proches des récepteurs).

Le reste du chapitre est organisé comme suit : La section 2 présente les limites de la classe "*receiver-initiated*" tout en montrant surtout la mauvaise répartition de la charge de recouvrement des pertes dans cette classe. La section 3 présente les caractéristiques du protocole proposé, sa contribution et ces objectifs. La section 4 est consacrée aux détails du protocole AMRHy entre autres les formats des paquets et les algorithmes descriptifs du comportement des différentes entités (la source, les routeurs actifs et les récepteurs). La section 5 terminera le chapitre.

2. Limites des protocoles de la classe *"receiver-initiated"*

Les protocoles de la classe "*receiver-initiated*" sont caractérisés par l'utilisation de l'acquittement négatif. Ce dernier est envoyé lorsqu'un récepteur se rend compte d'une perte d'un paquet de données. Malgré que cette classe ait pu réduire le nombre de paquets de contrôle

par rapport à la classe "*sender-initiated*", néanmoins, elle présente plusieurs inconvénients dont nous rappelons les plus essentiels dans cette section :

- La détection de la perte d'un paquet de données au niveau des récepteurs est différée jusqu'à l'arrivée du prochain paquet de données. Ceci s'avère inadéquat pour les applications qui sont sensibles au délai de livraison. Dans ce type d'applications, non seulement la fiabilité est exigée, mais également la latence de délai d'acheminement doit être réduite.
- La mauvaise répartition de la charge de recouvrement des pertes entre la source et les récepteurs, où la responsabilité de la détection des pertes est attribuée aux récepteurs sans tenir compte du lien sur lequel se produit la perte. Par conséquent, si une perte se produit sur un lien source, cette perte ne sera détectée qu'au niveau des récepteurs, générant ainsi en feedback un flux important d'acquittement négatifs qui demandent le paquet perdu.
- La mauvaise gestion de cache au niveau des routeurs actifs, où le moment de leur libération des anciens paquets reste inconnu. Car, il se peut qu'un paquet soit toujours utile au recouvrement local des pertes et que sa suppression du cache influe négativement sur la latence de recouvrement. Cependant, cette classe exige la disponibilité d'un cache ayant une taille illimitée pour pouvoir garantir un recouvrement local parfait, pourtant, ce n'est pas le cas dans un réseau réel.
- Le risque qu'un paquet de données perdu n'atteint jamais sa destination, lorsque la source dispose d'un nombre restreint de buffers en émission. Celle-ci, sera amenée à supprimer de son cache les paquets de données les plus anciens, il se peut que parmi ces paquets l'un soit toujours utile au recouvrement des pertes. Dans ce cas, même si la source reçoit un NAK demandant le paquet de données, elle n'a aucun moyen pour le récupérer.
- Le temps de l'élection d'un répondeur proposé dans [40], peut devenir important lorsque les NAKs sont perdus. Dans ce cas, le routeur actif est amené à faire plusieurs tentatives afin d'élire le répondeur approprié.

3. Présentation du protocole AMRHy

Les problèmes suscités de la classe "*receiver-initiated*", nous ont amené à proposer un protocole qui pallie à ces inconvénients en combinant les deux classes "*sender-initiated*" et "*receiver-initiated*". L'alliance de ces deux classes est traduite par la cohabitation de l'acquittement positif et l'acquittement négatif. L'acquittement positif est utilisé de bout-en-bout entre les récepteurs et la source pour confirmer la bonne réception d'un paquet de données par au moins un

récepteur, écartant ainsi la possibilité de sa perte sur un lien source. Il permet aussi au routeur actif :

- d'inviter les membres de son groupe, ayant perdu le paquet de données, de signaler cette perte avant que ce paquet ne soit supprimé du cache,
- d'informer les membres de son groupe de l'adresse du répondeur élu dynamiquement pour des réparations futures sans passer par les services actifs,
- d'informer les membres de son groupe, ayant reçu correctement le paquet de données, de faire une suppression locale de leurs ACKs correspondants,
- de supprimer du cache le paquet de données en question,

D'autre part, il permet à la source de libérer le buffer en émission associé au paquet de données acquitté et d'ajuster sa fenêtre d'émission.
Par contre l'acquittement négatif est utilisé localement entre le routeur actif et les récepteurs de son groupe pour signaler la perte d'un paquet de données.

3.1 Contribution du protocole AMRHy

Le protocole que nous proposons un protocole hybride. Il englobe l'approche hiérarchique et l'approche basée sur les temporisateurs, ainsi que les classes "*sender-initiated*" et "*receiver-initiated*" (cohabitation des ACKs et NAKs). Le regroupement de ces différentes approches et classes au sein de ce protocole a motivé son appellation. **AMRHy** pour "***A**ctive **M**ulticast **R**eliable **Hy**brid protocol*". La solution hybride adoptée par notre protocole lui permet de résoudre les problèmes inhérents à une communication de groupes qui surgissent à grande échelle tels que : l'implosion des paquets de contrôle au niveau de la source, la répartition de la charge de recouvrement des pertes, l'exposition des récepteurs et l'influence des récepteurs de faible capacité sur le groupe (*drop to zero*). L'intérêt de la combinaison des deux approches a été déjà étudié et analysé dans [2 3, 29]. La principale contribution de ce protocole dans le domaine du traitement de la fiabilité multicast réside dans la combinaison des deux classes "*sender-initiated*" et "*receiver-initiated*". L'intérêt de cette combinaison de classes est mis en évidence tout au long de ce chapitre et encore plus dans le chapitre 5 durant l'analyse de notre protocole. La figure 4.1 permet de situer le protocole AMRHy par rapport aux protocoles évoqués précédemment.
Dans cette alliance de classes, la source détecte les pertes qui se produisent sur les liens source et les récepteurs détectent celles qui se produisent sur les liens terminaux. Cette bonne répartition

de la charge de recouvrement des pertes entre la source et les récepteurs, permet de combler la limite de la classe "*receiver-initiated*" concernant la mauvaise répartition de la charge. Entre autre, elle permet aussi :

- d'éliminer le trafic des NAKs généré en feedback par les récepteurs, lorsque la perte se produit sur un lien source, et par conséquent, garantir un gain considérable en termes de temps de traitement au niveau des routeurs actifs et ainsi que la bande passante consommée,
- d'éviter le problème "*drop to zero*" puisque la source n'est plus soumise aux contraintes des récepteurs de faibles capacités. Celle-ci, dès qu'elle reçoit le premier ACK, libère le buffer en émission associé au paquet de données acquitté, réajuste sa fenêtre en émission et procède à l'envoi d'une nouvelle série de paquets de données, si le contrôle de flux le permet,
- d'optimiser l'utilisation du cache au niveau des routeurs actifs en supprimant à chaque fois le paquet de données acquitté.

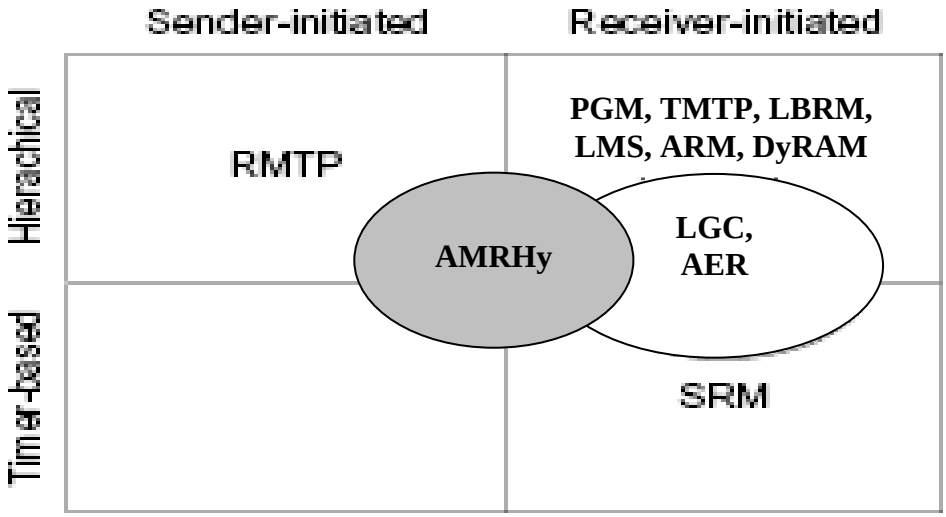

Figure 4.1 : Position d'AMRHy dans la classification des protocoles RM.

3.2 Objectifs du protocole AMRHy

La combinaison des différentes approches et classes de recouvrement de pertes a permis au protocole AMRHy d'hériter des avantages relatifs à chacune d'entre elles tout en comblant au mieux leurs limites. Ceci, permet à notre protocole de résoudre les problèmes qui surgissent à grande échelle et d'atteindre les objectifs de conception des protocoles de multicast fiable évoqués dans [29] :

a. Réduire l'implosion des ACKs au niveau de la source

Pour assurer une transmission fiable des paquets de données entre une source et un ensemble de récepteurs, il est important que chaque récepteur, retourne à la source un acquittement positif ACK, pour chaque paquet de données correctement reçu et un acquittement négatif NAK pour chaque paquet de données perdu. Eviter, en présence d'un grand nombre de récepteurs, le problème de l'implosion des acquittements au niveau de la source est un défi qui va permettre le passage à l'échelle. Notre protocole évite ce problème en combinant les deux approches hiérarchique et celle basée sur les temporisateurs :

- Une première suppression est faite selon le principe des protocoles SRM [20] et AER [29], lorsqu'un routeur actif reçoit le premier ACK provenant de l'un de ses descendants, avant de l'expédier à son ascendant dans l'arbre de distribution multicast, il le transmet au reste de ses descendants. Ceci permet :
  - aux récepteurs qui ont reçu le paquet de données de ne pas générer leurs ACKs correspondants (une suppression locale des ACKs),
  - aux récepteurs qui ont perdu le paquet de donnés de signaler cette perte avant que celui-ci ne soit supprimé du cache. Ce qui réduira la latence de recouvrement de perte d'une manière considérable. D'autre part, l'ACK sert à connaître l'adresse du répondeur élu pour une réparation future sans passer par les services actifs du routeur. Le répondeur élu est le premier à envoyer l'ACK.
- Une deuxième suppression est faite selon le principe des protocoles ARM [31] et DyRAM [40], par l'agrégation des duplications des ACKs au niveau hiérarchique de l'arbre de distribution multicast par les routeurs actifs en ignorant les ACKs identiques pendant une période de temps bien déterminée.

b. Répartir la charge de recouvrement des pertes

En présence d'un nombre important de récepteurs où les pertes sont fréquentes, la probabilité qu'au moins un récepteur perd un paquet est très élevée, d'où la nécessité de retransmettre chaque paquet à plusieurs reprises. Ainsi, la source et les liens associés deviennent surchargés et une dégradation des performances du réseau survient. Placer une puissance de traitement très proche de la source qui l'assiste dans le recouvrement des pertes ne résout pas le problème. D'où, l'importance d'une répartition efficace de la charge de recouvrement des pertes. Le protocole AMRHy assure une distribution équitable de la charge entre la source, les routeurs actifs et les récepteurs. Cela se fait grâce à une utilisation conjointe des deux classes "*sender-*

initiated" et "*receiver-initiated*" avec la contribution des routeurs actifs. Cette combinaison de classes comblera la limite constatée dans la classe "*receiver-initiated*" concernant le risque qu'un paquet de données n'atteint jamais sa destination en présence d'un cache restreint au niveau de la source.

Ainsi, le recouvrement des pertes est distribué entre les différentes entités comme suit :

- La détection de la perte du paquet de données au niveau de la source, est faite lors de l'expiration du délai de garde associé à chaque paquet de données émis. Cette expiration signifie que le paquet de données a été perdu sur un lien source et qu'aucun récepteur ne l'a reçu. Détecter ce genre de pertes par la source permet d'éliminer le problème de l'implosion des NAKs en feedback.
- La détection de la perte au niveau des récepteurs est faite lorsqu'un récepteur reçoit un ACK portant un numéro d'un paquet de données qui n'a pas été reçu, le récepteur signale la perte en envoyant un NAK au routeur actif le plus proche.
- Le routeur actif contribue aux recouvrements des pertes en envoyant le paquet de données aux récepteurs qui l'ont demandé, s'il existe dans son cache, dans le cas contraire, la demande sera expédié vers le répondeur (le premier récepteur ayant envoyé un ACK).

c. Limiter la portée de retransmission

Pour assurer la fiabilité de retransmission, le paquet perdu doit être retransmis soit par la source ou par toute autre entité intermédiaire. Lorsqu'on est en présence d'un groupe important de récepteurs et que les pertes sont fréquentes, la retransmission en unicast n'est pas efficace puisqu'elle demande plusieurs retransmissions identiques vers plusieurs récepteurs. Dans ce cas retransmettre en multicast est plus approprié. Ainsi, les paquets perdus seront retransmis à tous les récepteurs du groupe et non pas aux récepteurs qui ont effectivement perdu le paquet de données. Ce qui expose certains récepteurs à des transmissions dupliquées et provoque une consommation inutile de la bande passante. Le problème sera davantage aggravé au sein d'un groupe hétérogène où la source est amenée à s'adapter au rythme du récepteur le plus lent, ce qui peut avoir de forts impacts sur l'efficacité de la solution multicast fiable utilisée. D'où, l'importance de restreindre la portée des retransmissions aux récepteurs qui ont effectivement perdu le paquet de données. Le protocole proposé limite la portée de retransmission en maintenant une liste au niveau du routeur actif d'une manière similaire aux protocoles ARM [31] et DyRAM [40]. Le routeur actif inscrit dans cette liste l'adresse de chaque récepteur signalant la perte du paquet de données. Après la période d'attente, le temps de connaître tous les récepteurs

qui ont subi la perte du paquet de données, le routeur actif envoie le paquet de données perdu en multicast partiel aux récepteurs l'ayant effectivement perdu, évitant ainsi le problème d'exposition des récepteurs.

d. Garantir le support minimal du routeur

La charge supplémentaire attribuée au routeur pour assurer les services actifs ne doit pas être du même ordre que sa tâche principale et qui consiste à préserver les sémantiques de routage et de l'expédition de l'architecture courante de l'Internet. Notre protocole évite de surcharger le routeur actif par :

- L'élection dynamique d'un répondeur parmi les récepteurs assurant le recouvrement local des pertes sans passer par les services actifs du routeur.
- L'utilisation de l'acquittement positif permet une meilleure gestion du cache des routeurs actifs, en supprimant à chaque fois le paquet de données acquitté du cache. Une capacité d'un cache minimal est suffisante pour un recouvrement local efficace.

e. Restreindre le rôle des services actifs

La fonctionnalité d'un service actif ne consiste pas à corriger le fonctionnement du protocole multicast. Cet objectif assure que le service actif sera graduellement déployé dans le réseau sans affecter la continuité du bon fonctionnement du protocole. Notre protocole fonctionne correctement sans la présence des services actifs dont leur rôle consiste essentiellement à améliorer ses performances. En cas d'absence de ces services, notre protocole adopte une approche de bout en bout dans laquelle la charge de recouvrement des pertes est répartie entre la source et les récepteurs.

Le tableau ci-dessous résume les stratégies utilisées par les protocoles de transport multicast fiable basés sur les services actifs et met l'accent sur l'aspect fiabilité.

Tableau 4.1 : Comparaison des stratégies des protocoles de multicast fiable actif

	AER	**ARM**	**DyRAM**	**AMRHy**
Implosion des acquittements	Agrégation et suppression	Agrégation	Agrégation	Agrégation et suppression
Répartition de charge de recouvrement	Source et serveur de réparation	Source et routeur actif	Source et récepteur élu	Source, routeur actif et récepteur élu
Portée de retransmission	Contrôlée par le serveur de réparation	Contrôlée par le routeur actif	Contrôlée par le routeur actif	Contrôlée par le routeur actif

4. Caractéristiques du protocole AMRHy

Dans AMRHy, la source émet les paquets de données à une adresse multicast à laquelle sont inscrits tous les récepteurs d'un groupe. Nous supposons que les fonctionnalités du multicast IP créent un arbre de distribution multicast à travers le réseau et qu'elles assurent une délivrance " *best effort* " à tous les récepteurs du groupe. Nous supposons aussi que les paquets d'acquittement suivent le même chemin inverse des paquets de données. Cette supposition est essentielle pour tirer profit des services actifs.

4.1 Structure des paquets

On distingue deux sortes de paquets, les paquets de données et les paquets de contrôle.

a. Paquet de données

Pour assurer le recouvrement des pertes, le protocole AMRHy propose une structure d'un paquet de données englobant en plus des informations habituelles (adresses IP de la source et de la destination et les données), les informations supplémentaires suivantes :

- Un champ de numéro de séquence (*Numseq*) qui permet d'identifier le paquet de données.
- Un champ indicateur (*Ind*) qui permet de distinguer entre une transmission originale et une retransmission.
- Un champ identificateur du service actif (*Idsa*) qui sera appliqué au paquet de données.

- Un champ adresse (*Orgadr)* qui permet d'identifier l'origine du paquet. Ce champ est mis à jour, à chaque fois que le paquet traverse un nœud actif. Ce dernier sauvegarde l'adresse IP contenue dans ce champ et la substitue par sa propre adresse IP, permettant ainsi d'établir le chemin inverse d'un récepteur vers la source. Cette technique est inspirée du fonctionnement du protocole AER [29]. Cependant, le protocole AER procède à la transmission des paquets de signalisation supplémentaires et n'intègre pas ce mécanisme au niveau de ses paquets de données. L'intégration d'un champ '*Orgadr*' au niveau des paquets de données permet de réduire le nombre de paquets de contrôle échangés dans le réseau.

@IPsource	*@IPMulticast*	*Numseq*	*Ind*	*Idsa*	*Orgadr*	*Données*

Figure 4.2 : Structure d'un paquet de données du protocole AMRHy

b. Paquet de contrôle

La structure d'un paquet de contrôle du protocole AMRHy est assez simple. Il englobe en plus des champs d'adresses IP source et destination, le numéro de séquence spécifiant le prochain paquet de données attendu, et un champ '*type*' permettant de distinguer entre un acquittement positif (ACK) d'un acquittement négatif (NAK).

@IPsource	@IPdestination	Numseq	Type

Figure 4.3 : Structure d'un paquet de contrôle du protocole AMRHy.

4.2 Principaux algorithmes d'AMRHy

- AMRHy est un protocole de transport multicast fiable basé sur les services actifs. Ces derniers sont invoqués uniquement pendant les sessions multicast, l'invocation se fait à travers l'arbre de distribution multicast. Une fois que les services sont invoqués, chaque entité (source, routeurs actifs et récepteurs) se comporte de la manière suivante :
- *a. Comportement de la source*

– *Lors de l'envoi d'un paquet de données* :

- Initialisation d'un délai de garde. En pratique ce délai de garde est supérieur ou égal au RTT du récepteur le plus éloigné du groupe de la source et sera ajusté au fur et à mesure que la source reçoit les acquittements.
- Stockage du paquet de données dans un buffer du cache.
- Emission du paquet de données à une adresse multicast, adresse d'inscription de tous les récepteurs d'un groupe.

– *A la réception d'un ACK* :
- Libération du buffer de cache associé au paquet de données acquitté.
- Ajustement de la fenêtre en émission de la source.

– *A l'expiration du délai de garde* : /* Détection de la perte par la source */
- Retransmission du paquet de données à l'adresse multicast ;
- Réinitialisation du délai de garde ;

b. Comportement du récepteur

– *A la réception d'un paquet de données original* :
- Initialisation d'un délai d'attente aléatoire ;
- Mise à jour de la liste des paquets reçus ;

– *A la réception d'un paquet de données de réparation* :
- Mise à jour de la liste des paquets reçus ;
- Se comporter comme s'il a envoyé un ACK à l'ascendant dans l'arbre de distribution multicast ;

– *A la réception d'un ACK* :

Si le numéro de séquence n'existe pas parmi la liste des paquets reçus ***alors*** /* Détection de la perte par le récepteur */
- Initialisation d'un délai de garde avec une valeur égale au RTT entre le récepteur le plus éloigné du groupe et son routeur actif ;
- Envoi d'un NAK au routeur actif ;
- Enregistrement de l'adresse du répondeur ; /* Pour une réparation future */

Sinon
- ***Si*** le délai d'attente du récepteur est armé ***alors*** désarmer le délai d'attente ***fsi***.
- Se comporter comme s'il a envoyé un ACK ; /* suppression locale des ACKs */

Fsi.

– *A la réception d'un NAK* : /* Seul le récepteur élu répondeur par un routeur actif peut recevoir un NAK */
  - Envoi du paquet de réparation au récepteur l'ayant demandé ;
  - *A l'expiration du délai d'attente* : /*Le délai d'attente expire avant que le récepteur ne reçoit un ACK portant le même numéro de son ascendant */
  - Envoi d'un ACK vers l'ascendant dans l'arbre de distribution multicast ;
– *A l'expiration du délai de garde* :
  - Initialisation du délai de garde avec une valeur égale au RTT entre le récepteur en question et le répondeur ;
  - Envoi d'un NAK au répondeur ;

c. Comportement du routeur actif

– *A la réception d'un paquet de données original* :
  - Stockage du paquet de données dans un buffer du cache ;
  - Expédition du paquet de données aux fils directs dans l'arbre de distribution multicast ;
  - *A la réception d'un paquet de réparation* : /* ce paquet sera traiter de la même manière qu'un paquet de données original dans cette partie de l'arbre multicast */
– *A la réception d'un ACK* :

  Si l'ACK provient d'un fils ***alors***
  - *Subcast* de l'ACK aux autres fils ;
  - Initialisation d'un délai d'attente avec une valeur égale au RTT entre le routeur actif et son fils direct le plus éloigné dans l'arbre de distribution multicast ;
  - Ignorance des ACKs qui arrivent des fils pendant la durée d'attente ; /*agrégation des ACKs au niveau du routeur actif */

  Sinon **/*** l'ACK provient d'un fils */
  - ***Si*** le numéro de l'ACK ne figure pas parmi la liste des paquets reçus ***alors***
    - Envoi d'un NAK à son nœud père dans l'arbre de distribution multicast ;
    - Initialisation d'un délai de garde avec une valeur égale au RTT entre le routeur actif père et son fils direct le plus éloigné dans l'arbre de distribution multicast ;
    - Enregistrement de l'adresse du répondeur ;
  - ***sinon*** /* le paquet de données a été déjà reçu */
    - Désarmement du délai d'attente ;
    - Pas d'envoi d'un ACK au nœud père ; /*suppression locale des ACKs*/

Fsi ;

Fsi ;

- *A la réception d'un NAK* :
  - Ajout de l'adresse du fils à la liste des fils n'ayant pas reçu le paquet de données ;
- *Expiration du délai de garde* :
  - Expédition du NAK vers le répondeur ;
  - Initialisation d'un délai de garde égal au RTT du routeur actif au répondeur ;
- *Expiration du délai d'attente* :
  - Subcast du paquet de données aux fils ayant effectivement envoyé un NAK ;
  - Libérer le cache ;
  - Envoi d'un ACK au nœud père ;

5. Conclusion

Dans ce chapitre, nous avons proposé un nouveau protocole de transport multicast fiable basé sur le concept des réseaux actifs. Celui-ci adopte une approche hybride, qui combine les différentes approches et les différentes classes, nous menons ainsi de le nommer **AMRHy** pour "***A**ctif **M**ulticast **R**eliable **Hy**brid protocol*". Les protocoles de la classe "*receiver-initiated*" attribuent la responsabilité de la détection de perte aux récepteurs, indépendamment des liens sur lesquels les pertes se sont produites, entrainant une répartition inefficace de la charge de recouvrement des pertes. Contrairement à ces protocoles, notre protocole, en combinant les deux classes "*sender-initiated*" et "*receiver-initiated*", répartit d'une manière équitable la responsabilité de la détection des pertes entre la source et les récepteurs. Dans cette approche hybride, le lien sur lequel la perte s'est produite est pris en considération, où la source détecte les pertes qui se produisent sur les liens proches d'elle (liens source), tandis que les récepteurs se chargent de ceux qui se produisent sur leurs liens proches d'eux (liens terminaux). L'approche hybride adoptée par notre protocole permet l'héritage des avantages de chacune des classes, offrant les mécanismes efficaces pour résoudre les problèmes de la scalabilité qui surgissent à grande échelle tels que : l'implosion des acquittements en feedback, l'équilibrage de la charge de réparation, l'isolement du recouvrement ou l'exposition des récepteurs, et le problème (*drop to zero*) avec les récepteurs ayant des capacités limitées. Cependant, il est clair que les avantages présentés ne sont que théoriques et seule une analyse du protocole permettra de vérifier la véritable potentialité du protocole AMRHy. Ainsi et, afin de dégager des études comparatives

avec les protocoles de multicast fiable actif, une analyse des performances en termes de bande passante et débit sera détaillée dans le chapitre.

Chapitre 5

Analyse du protocole AMRHy

1. Introduction

La performance des protocoles peut être évaluée par une analyse mathématique, une simulation, ou par une mesure soit à travers Internet ou à l'aide d'un *test bed*. En effet, la mesure dans un environnement cible montre les performances réelles du protocole et de son implémentation disponible. Cependant, et plus particulièrement pour les groupes multicast, les mesures sont difficiles à réaliser puisqu'elles exigent souvent la coordination entre plusieurs sites multicast. C'est pourquoi la plupart des protocoles de multicast sont évalués soit par une analyse mathématique ou par une simulation.

L'analyse mathématique exige un modèle généralement simplifié de l'environnement et du protocole considéré. Elle permet de changer les paramètres et de superviser d'une manière tout à fait simple leur effet sur le comportement d'un protocole. Un autre avantage des analyses mathématiques est leur publication habituellement complète, permettant à d'autres personnes de vérifier l'exactitude (*correctness*) de l'analyse et du modèle considéré.

Avec les simulations, l'environnement et les protocoles considérés peuvent devenir plus complexes et par conséquent, plus réalistes. Cependant, les grands systèmes de simulation comme NS-2 laissent changer beaucoup de paramètres, influençant probablement les résultats. D'ailleurs, il est difficile de vérifier l'exactitude des simulations, puisque l'implémentation est souvent étendue et dans beaucoup de cas, le code de simulation n'est pas public. C'est la raison pour laquelle, nous croyons que l'analyse par modèle mathématique et par simulation sont les deux utiles et complémentaires.

Dans ce chapitre, nous analysons les performances du protocole AMRHy en termes de débit et de la bande passante et ensuite, pour montrer l'intérêt de l'approche hybride adoptée par AMRHy, nous comparons ses résultats avec ceux du protocole DyRAM qui représente les protocoles de la classe *receiver-initiated*.

La première analyse comparative entre les protocoles de multicast fiable des classes *sender-initiated* et *receiver-initiated* a été faite par Pingali et al [49]. Cette analyse a montré que les protocoles de la classe *receiver-initiated* sont de loin plus *scalables* que ceux de la classe *sender-initiated* parce que le débit maximal de la classe *sender-initiated* dépend du nombre de récepteurs, alors que ce n'est pas le cas de la classe *receiver-initiated*. Levine et al [32] ont étendu ce travail aux approches basées sur une structure arbre et ils ont prouvé que l'organisation de l'ensemble des récepteurs dans une structure hiérarchisée garantit la scalabilité et améliore la performance. Ils ont montré également que les protocoles de la classe *receiver-initiated* ne

peuvent pas garantir la fiabilité lorsqu'ils opèrent dans un environnement avec des caches limités. Une autre analyse comparative des classes *sender-initiated* et *receiver-initiated* a été présentée par Maihöfer et Rothermel [38]. Leur analyse a prouvé que les protocoles de la classe *receiver-initiated* réalisent la meilleure scalabilité mais ceux de la classe *sender-initiated* réalisent des latences réduites.

D'autre part, l'efficacité de la bande passante a été sujette à plusieurs études analytiques. L'analyse des protocoles multicast fiables génériques a été faite par Kasera et al [28] et cette analyse a prouvé que les approches de recouvrement local fournissent une performance significative en termes de consommation de la bande passante et de délai. Maihöfer [37] a présenté une évaluation analytique de la bande passante des protocoles de multicast fiable génériques et a prouvé que les approches hiérarchiques fournissent non seulement un débit élevé mais également consomme moins de bande passante.

Notre analyse permet d'étendre les travaux précédents en comparant la classe *receiver-initiated* avec une approche hybride qui combine les deux classes dans un environnement de réseaux actifs. La combinaison des deux classes peut tirer profit des avantages de chacune d'elles.

Les avantages de la classe *receiver-initiated* sont :

- la source ne connait pas l'ensemble de récepteurs,
- la source ne doit pas traiter un ACK de chaque récepteur,
- les récepteurs qui agissent (*pace*) sur la vitesse de la source.

Les avantages de la classe *sender-initiated* sont :

- la source et les routeurs actifs connaissent le moment pendant lequel ils peuvent libérer les buffers du cache sans risque,
- des latences réduites.

Par conséquent, leur combinaison permettra une distribution équitable et efficace de la charge de recouvrement de pertes entre la source et les récepteurs.

Le reste du chapitre est organisé comme suit : la section 2 décrit le comportement générique des protocoles AMRHy et DyRAM. La section 3 présente le modèle réseau et les hypothèses sur lesquels se basent notre analyse comparative entre AMRHy et DyRAM. Les sections 4 et 5 détaillent respectivement l'analyse analytique des deux protocoles en termes de débit et de bande passante. Le chapitre se termine par une conclusion.

2. Description des protocoles

AMRHy et DyRAM sont deux protocoles qui emploient des services actifs au niveau des routeurs. Chacun d'eux adopte une stratégie différente pour résoudre les problèmes de scalabilité. DyRAM est basé sur la classe *receiver-initiated* où la responsabilité de la détection de perte est attribuée aux récepteurs. En revanche, la responsabilité de la détection de perte dans AMRHy est distribuée entre la source et les récepteurs en combinant les classes *sender-initiated* et *receiver-initiated*. Dans cette approche hybride, la source prend en charge les pertes qui se produisent dans le lien source tandis que les récepteurs prennent soin de celles qui se produisent dans les liens terminaux.

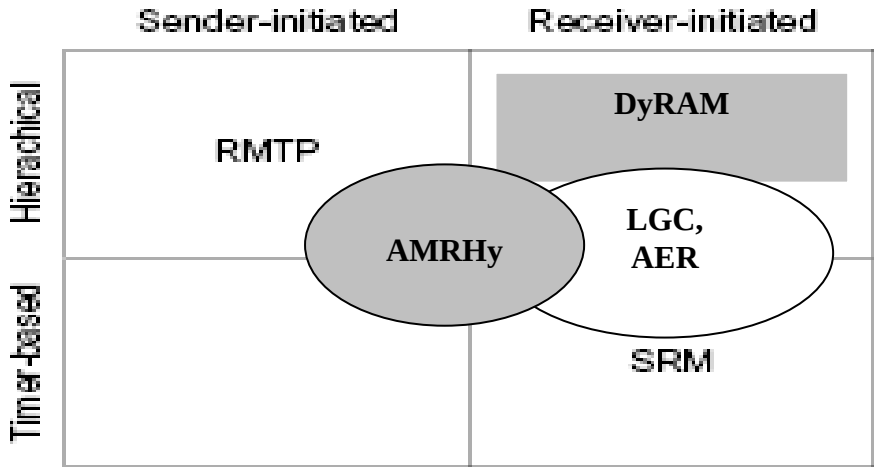

Figure 5.1 : AMRHy et DyRAM dans la classification de protocoles RM.

2.1 Description du protocole AMRHy

Le protocole AMRHy possède les propriétés suivantes :

- La source transmet en multicast le paquet de données à l'adresse de groupe souscrite par tous les récepteurs et initialise un temporisateur.
- A l'expiration du temporisateur, la source transmet en multicast le paquet de données à l'adresse de groupe souscrite par tous les récepteurs et initialise un temporisateur.
- Sur réception d'un ACK, la source supprime le paquet du cache et ajuste sa fenêtre en émission.
- Sur réception du premier ACK à partir d'un descendant, un routeur actif transmet en multicast l'ACK aux récepteurs de son groupe local et initialise un temporisateur WP

(*waiting period timer*). Au cours de cette période, le routeur ignore tous les ACKs redondants à partir des descendants.

- Sur réception d'un ACK à partir d'un ascendant pendant la période d'attente, un routeur actif vérifie si le paquet de données correspondant a été reçu. Si c'est le cas, il se comporte si comme s'il a envoyé un ACK, sinon il envoie un NAK vers son ascendant dans l'arbre de distribution multicast et initialise un temporisateur.
- A l'expiration du temporisateur, un routeur actif transmet un NAK vers ascendant dans l'arbre de distribution multicast et initialise un temporisateur.
- A l'expiration du temporisateur WP sans recevoir un ACK à partir de son ascendant, un routeur actif transmet un ACK vers son ascendant, transmet en multicast partiel le paquet de données vers les récepteurs l'ayant demandé et supprime le paquet de son cache.
- Sur réception d'un NAK à partir d'un ascendant ou d'un descendant, un routeur actif transmet le paquet s'il est disponible dans son cache, sinon il expédie le NAK vers le répondeur (le premier récepteur ayant envoyé l'ACK).
- Sur réception d'un paquet de réparation, un routeur actif l'expédie vers les nœuds qui l'ont demandé (ascendant ou descendant).
- Sur réception d'un paquet de données, un récepteur attend pendant une période aléatoire avant d'envoyer un ACK à son routeur actif. Si pendant la période d'attente, le récepteur reçoit un ACK alors il se comporte comme s'il l'a déjà envoyé.
- A l'expiration du temporisateur, le récepteur transmet un ACK vers son routeur actif.
- Sur réception d'un ACK, un récepteur vérifie le paquet de données correspondant. S'il a été déjà reçu alors le récepteur se comporte comme s'il a transmis un ACK sinon il envoie un NAK vers son routeur actif et initialise un temporisateur.

2.2 Description du protocole DyRAM

Le protocole DyRAM possède les propriétés suivantes :

- La source transmet en multicast le paquet de données à l'adresse de groupe souscrite par tous les récepteurs.
- Sur réception d'un NAK, la source transmet en multicast le paquet correspondant à tous les récepteurs (sur l'adresse de groupe).
- Sur réception d'un paquet de données, un routeur actif stocke le paquet dans le cache s'il le peut et l'expédie ensuite dans l'arbre de distribution multicast.
- Sur réception d'un paquet de réparation, un routeur actif le transmet en multicast partiel aux récepteurs l'ayant demandé.

- Sur détection d'une perte de paquet, un routeur actif transmet immédiatement un NAK vers son ascendant dans l'arbre de distribution et initialise un temporisateur pour une éventuelle requête.
- A l'expiration du temporisateur, un routeur actif transmet un NAK vers son ascendant dans l'arbre de distribution et initialise un temporisateur.
- Sur réception d'un NAK valide, un routeur actif initialise le temporisateur DTD (*Delay To Decide*) qui permet l'élection d'un répondeur.
- A l'expiration du temporisateur DTD, un routeur actif transmet un NAK vers le répondeur élu s'il existe, sinon il le transmet vers la source.
- Sur détection d'un paquet perdu, un récepteur transmet immédiatement un NAK vers la source et initialise un temporisateur.
- Sur réception d'un NAK, un récepteur transmet le paquet demandé s'il est disponible, sinon un NAK est retourné vers son routeur actif.

3. Modèle de réseau et les hypothèses de l'analyse

Le modèle réseau que nous avons adopté pour réaliser notre étude, est inspiré de celui proposé dans [39, 49]. Il est basé sur une topologie de réseau ayant une structure d'un arbre multicast qui admet la source comme racine, les routeurs actifs comme nœuds intermédiaires et les récepteurs à l'extrémité de l'arbre (voir figure 5.2). La source multicast envoie les paquets de données aux R récepteurs qui sont répartis selon la topologie en N =R/B groupes locaux. La taille d'un groupe local B est définie comme étant le nombre de récepteurs dans ce groupe. R est supposé multiple de B. En réalité, les récepteurs peuvent être soit des LAN's contenant une ou plusieurs applications réceptrices soit tout simplement d'autres routeurs actifs. Ces derniers, sont responsables des groupes locaux d'un niveau hiérarchique inférieur. Dans notre étude, nous considérons que les routeurs actifs sont placés à des endroits stratégiques dans le réseau, où les pertes se produisent souvent. Ces endroits représentent la périphérie du *backbone* pour deux raisons essentielles : (a) le *backbone* est supposé fiable. Il a été montré dans [61] que les liens qui subissent le plus de pertes sont ceux qui se trouvent à la périphérie du *backbone*. On note qu'il existe deux types de liens : le lien source (*source link*), qui relie la source au *backbone*, et les liens terminaux (*tail link*), qui relient les récepteurs au *backbone*, (b) le *backbone* est un réseau à haut débit. Si on y place des routeurs actifs, celui-ci sera ralenti par les traitements effectués par ces derniers. Ces endroits stratégiques vont permettre aux routeurs actifs d'intercepter tout

paquet de données envoyé de la source vers les récepteurs de son groupe pour assurer le recouvrement local des pertes.

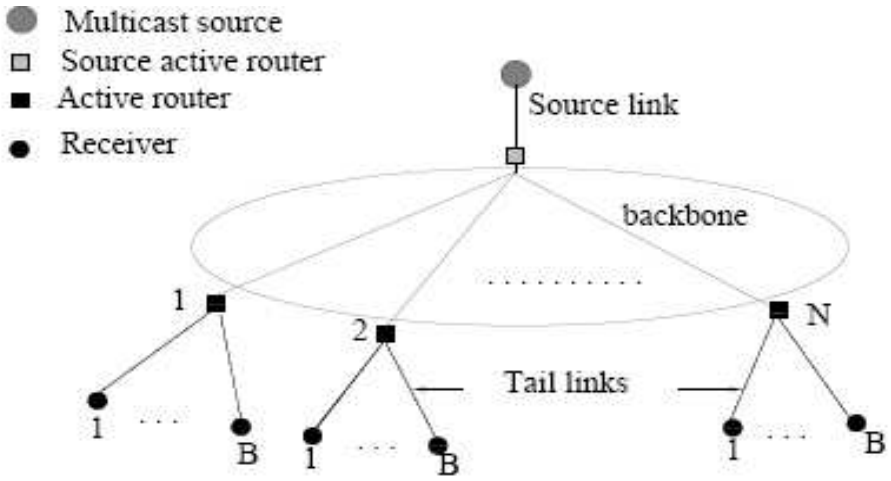

Figure 5.2 : Modèle de réseau.

Notre étude consiste à analyser les besoins du protocole AMRHy en termes de débit et de bande passante et les comparer à ceux du protocole DyRAM qui adopte une approche hiérarchique basée sur la classe *receiver-initiated*. Cette comparaison va nous permettre de montrer l'intérêt de la combinaison des classes adoptée par le protocole AMRHy dans un environnement non fiable, où le taux de pertes est supposé trop élevé. Le choix du protocole DyRAM est motivé par le fait que d'une part, il représente les protocoles basés sur la classe *receiver-initiated* et d'autre part, qu'AMRHy adopte la même stratégie que celle adoptée par ce dernier concernant l'élection dynamique d'un répondeur parmi les récepteurs afin de décharger le routeur actif du recouvrement local.

Nous admettons quelques hypothèses essentielles à la réalisation de notre analyse :

- Pour la modélisation des pertes, le *backbone* est considéré entièrement fiable, alors que sur les autres liens (lien source et liens terminaux), les pertes se produisent avec une probabilité P_l. Dans ce cas, la probabilité vue par un récepteur pour qu'une perte se produise de bout en bout est de $P=1-(1-P_l)^2$. Les pertes sont supposées temporellement indépendantes, celles relatives aux liens terminaux sont mutuellement indépendantes.
- Nous supposons qu'un NAK (ou ACK) envoyé en unicast, suit le même chemin en sens inverse du multicast pour pouvoir bénéficier des mêmes services actifs, cela peut être réalisé soit en implémentant un service de routage spécifique aux NAKs (ou ACKs), soit en

sauvegardant l'adresse de chaque nœud actif traversé par le paquet données, comme il a été mentionné dans [17].

- On dit qu'un NAK (ACK) est non valide, quand il est reçu par un routeur actif ou par la source, alors que celui-ci ou celle-ci vient de traiter un NAK (ACK) similaire. Afin d'éviter les traitements inutiles de ces derniers, la source et les routeurs actifs ignorent les NAKs (ACKs) pendant une durée de temps bien déterminée.
- Nous supposons aussi que le rôle d'un routeur actif se résume à faire le cache de paquets de données, l'agrégation, la suppression des acquittements et l'élection dynamique d'un répondeur parmi les récepteurs d'un groupe local.

4. Analyse du débit maximal

Dans cette section notre analyse se focalise sur les besoins en terme de temps de traitement des protocoles AMRHy (**A**) et DyRAM (**D**). Le débit maximal atteint par un protocole dépend du temps de traitement par paquet au niveau des différents nœuds de l'arbre multicast. Pour faire l'analyse du débit, nous utilisons des notations similaires à celles utilisées dans [39, 49]. Ces notations sont représentées dans le tableau 5.2. Le tableau 5.1 résume la distribution des probabilités et les moyennes des variables aléatoires. Pour le calcul de la moyenne d'une variable aléatoire M, nous utilisons la formule suivante :

$$E[M] = 1 + \sum_{m=1}^{\infty} 1 - P[M \leq m]$$

Sachant que : $E[M]$ représente la moyenne d'une variable aléatoire, et $P[M \leq m]$ représente la distribution de probabilités.

4.1 Analyse du protocole AMRHy (A)

Commençons par analyser les besoins de traitement au niveau de la source. Pour un paquet de données, la source l'envoie M_s fois jusqu'à ce qu'il soit correctement reçu par au moins un récepteur du groupe local, traite pour cela (M_s -1) expirations du délai de garde, et reçoit par conséquent un seul ACK grâce au service de suppression locale et de l'agrégation des ACKs au niveau des routeurs actifs. Le temps de traitement par paquet au niveau de la source peut être exprimé par :

$$E[X^{A}] = E[M_s]E[X_p] + (E[M_s] - 1)E[X_t] + E[X_a] \quad (1)$$

En remplaçant $E[M_s]$ par sa valeur du tableau 5.1 nous obtenons l'expression :

$$E[X^{A}] = \frac{1}{1-P} E[X_{p}] + \frac{P}{1-P} E[X_{t}] + E[X_{a}] \quad (2)$$

Tableau 5.1 : Distribution de probabilités et de moyennes des variables aléatoires.

Variable aléatoire X	Distribution de probabilités P [X≤ m]	Moyenne E[x]
M_s	$1 - P^m$	$\frac{1}{(1-P)}$ or $E[(M_s-2)^+] = \frac{p^2}{(1-P)}$
M_b	$1-P_l^m$	$\frac{1}{(1-P_l)}$ or $E[(M_b-2)^+] = \frac{p_l^2}{(1-P_l)}$
M_B	$(1-P_l^m)^B$	$1 + \sum_{m=1}^{\infty} 1-(1-P_l^m)^B$

Le temps de traitement au niveau du routeur actif dépend de la localisation de celui-ci s'il se trouve sur un lien terminal ou un lien source :

a. Routeur des liens terminaux (Ai : i=1..N)

Un routeur actif associé à un lien terminal reçoit le paquet de données M_b fois de son père et (M_B -M_b) de la part du répondeur de son groupe local ; donc M_B fois [M_b+ (M_B -M_b) = M_B], et envoie le même paquet de donnée vers les récepteurs de son groupe M_B fois. Par conséquent il reçoit (envoie) un ACK et traite pour cet ACK une expiration d'un délai d'attente égale au RTT du routeur actif en question jusqu'à son fils direct le plus éloigné. Il reçoit (M_s -1) NAKs de la part de chaque récepteur de son groupe ; donc B (M_s -1) de tous les récepteurs de son groupe, il envoie (M_B -1) NAKs vers le répondeur et traite pour ces NAKs (M_B -2) expirations du délai de garde. Le temps de traitement peut être exprimé par :

$$\begin{aligned} E[A_i^A] = {} & E[M_B](E[Y_p^a] + E[X_p^a]) + E[X_a^a] + E[Y_a^a] + E[A_t] \\ & + (E[M_B]-1)E[Y_n^a] + B(E[M_s]-1)E[X_n^a] \\ & + E[(M_B-2)]E[A_t] \end{aligned} \quad (3)$$

En remplaçant $E[M_s]$ par sa valeur du tableau 5.1 nous obtenons l'expression :

$$\begin{aligned} E[A_i^A] = {} & E[M_B](E[Y_p^a] + E[X_p^a]) + E[X_a^a] + E[Y_a^a] + E[A_t] \\ & + (E[M_B]-1)E[Y_n^a] + B(\frac{P}{1-P})E[X_n^a] + E[(M_B-2)]E[A_t] \end{aligned} \quad (4)$$

b. Routeur du lien source A_S

Le routeur actif associé au lien source (A_s) reçoit le paquet de données de la source M_s fois avec la probabilité ($1-P_l$) et envoie ($1-P_l$) *Ms* vers ses descendants et reçoit un seul ACK de l'un de ces fils, *subcast* un ACK vers les autres fils, traite pour cela une expiration de délai d'attente et envoie à la source un seul ACK**.** Le temps de traitement exprimé par :

$$E[A_s^A] = (1-P_l)E[M_s](E[Y_p^a] + E[X_p^a]) + E[Y_a^a] + E[A_t] + E[X_a^a] \quad (5)$$

En remplaçant *E [M_s]* par sa valeur du tableau 5.1 nous obtenons l'expression :

$$E[A_s^A] = \frac{1}{1-P_l}(E[Y_p^a] + E[X_p^a]) + E[Y_a^a] + E[X_a^a] + E[A_t] \quad (6)$$

Le besoin de traitement au niveau de chaque récepteur. Pour chaque paquet de données, un récepteur émet (M_s -1) NAKs, traite (M_s -2) expirations de délai de garde jusqu'à ce que le paquet de données soit correctement reçu, grâce au *subcast* il reçoit le paquet de données une seule fois, par conséquent il envoie ou reçoit un seul ACK. Si maintenant le récepteur est choisi comme répondeur, il envoie le paquet de données autant de fois qu'il reçoit le NAK**.**

Ce nombre est estimé à : $\frac{E[M_B]-1}{B}$. Le temps de traitement peut être exprimé par :

$$\begin{aligned} E[Y^A] = {} & E[Y_p] + E[Y_a] + (E[M_s]-1)E[Y_n] + E[(M_s-2)]E[Y_t] \\ & + \left(\frac{E[M_B]-1}{B}\right)(E[X_p] + E[X_n]) \end{aligned} \quad (7)$$

En remplaçant *(E [M_s]-1)* et *(E [M_s]-2)* par leurs valeurs correspondantes du tableau 5.1 nous obtenons :

$$\begin{aligned} E[Y^A] = {} & E[Y_p] + E[Y_a] + \frac{P}{1-P}E[Y_n] + \frac{P^2}{1-P}E[Y_t] \\ & + \left(\frac{E[M_B]-1}{B}\right)(E[X_p] + E[X_n]) \end{aligned} \quad (8)$$

4.2 Analyse du protocole DyRAM.

De la même manière qu'**AMRHy**, nous commençons par analyser les besoins de traitement au niveau de la source. Pour chaque paquet de données, La source envoie le paquet de données M_s fois jusqu'à ce qu'il soit correctement reçu par au moins un récepteur du groupe local, et donc reçoit (M_s -1) NAKs. Le temps de traitement peut être exprimé par :

$$E[X^D] = E[M_s]E[X_p] + (E[M_s]-1)E[X_n] \quad (9)$$

En remplaçant *(E [M_s])* et *(E [M_s]-1)* par leurs valeurs correspondantes du tableau 5.1 nous obtenons :

$$E[X^{D}] = \frac{1}{1-P} E[X_{p}] + \frac{P}{1-P} E[X_{n}] \quad (10)$$

Le temps de traitement au niveau du routeur actif dépend de la localisation de celui-ci s'il se trouve sur un lien terminal ou un lien source :

a. Routeur actif des liens terminaux (A_i : i=1..N)

Un routeur actif d'un lien terminal reçoit le paquet de données M_b fois de son père et (M_B -M_b) de la part du répondeur de son groupe local; donc M_B fois [M_b + (M_B - M_b)=M_B], et envoie le même paquet vers les récepteurs de son groupe M_B fois. Par conséquent il envoie (M_b -1) NAKs vers la source et traite pour ces NAKs (M_b -2) expiration de délai de garde. Il reçoit (M_s -1) NAKs de la part de chaque récepteur de son groupe ; donc B (M_s -1) de tous les récepteurs. Il envoie (M_B -1) NAKs vers le répondeur et traite pour ces NAKs (M_B -2) expirations de délai de garde et choisit le répondeur pendant un délai de décision (DTD : ce délai n'est pas représenté dans le tableau 5.2, nous supposons que ce délai est égal au délai de garde A_t). Le temps de traitement peut être exprimé par :

Tableau 5.2 : Notations utilisées dans l'évaluation analytique du débit.

X^A, X^D	Temps total de traitement par paquet de données au niveau de la source pour les protocoles **A** et **D**.
A_i^A , A_i^D	Temps total de traitement par paquet de données au niveau d'un routeur actif Ai (i = 1..N) associé à un lien terminal pour les protocoles **A** et **D**.
A_s^A , A_s^D	Temps total de traitement par paquet de données au niveau du routeur actif As associé au lien source pour les protocoles **A** et **D**.
Y^A, Y^D	Temps total de traitement par paquet de données au niveau d'un récepteur pour les protocoles **A** et **D**.
X_p , X_n , X_a	Temps nécessaire pour l'envoi d'un paquet de données et la réception d'un NAK (ACK) respectivement.
Y_p , Y_n , Y_a	Temps nécessaire pour la réception d'un paquet de données et l'envoi d'un NAK (ACK) respectivement.
X_p^a , X_n^a , X_a^a	Temps nécessaire pour l'envoi d'un paquet de données et la réception d'un NAK (ACK) par un routeur actif respectivement.
Y_p^a , Y_n^a , Y_a^a	Temps nécessaire pour la réception d'un paquet de données et l'envoi d'un NAK (ACK) par un routeur actif respectivement.
X_t	Temps nécessaire de traitement d'une expiration d'un délai de garde par la source.
Y_t	Temps nécessaire de traitement d'une expiration d'un délai de garde par un récepteur.
A_t	Temps nécessaire de traitement d'une expiration d'un délai de garde par un routeur actif.
M_s	Le nombre de transmissions nécessaire d'envois d'un paquet de données à partir de la source jusqu'à ce qu'il soit correctement reçu par un récepteur.
M_b, M_B	Le nombre de transmissions nécessaires d'envois d'un paquet de données à partir d'un routeur actif, jusqu'à ce qu'il soit correctement reçu par un ou tous les récepteurs de son groupe local.

$$
\begin{aligned}
E[A_i^D] = {} & E[M_B](E[Y_p^a] + E[X_p^a]) + (E[M_b] - 1)E[Y_n^a] + E[(M_b - 2)\ ]E[A_t] \\
& + (E[M_B] - 1)E[Y_n^a] + E[(M_B - 2)\ ]E[A_t] + E[A_{dtd}^a] \\
& + B*(E[M_s] - 1)E[X_n^a]
\end{aligned}
\qquad (11)
$$

En remplaçant *(E [M_s]-1), (E [M_b-1])* et *(E [M_b-1])* par leurs valeurs correspondantes du tableau 5.1 nous obtenons :

$$E[A_i^D] = E[M_B](E[Y_p^a] + E[X_p^a]) + \frac{P_l}{1-P_l}E[Y_n^a] + \frac{P_l^2}{1-P_l}E[A_t] + E[M_B - 1)E[Y_n^a] + E[(M_B - 2)\ ]E[A_t] + B*(\frac{P}{1-P})E[X_n^a] + E[A_{dtd}^a] \quad (12)$$

b. Routeur du lien source A_S

Le routeur actif (A_s) associé au lien source reçoit le paquet de données de la source M_s fois avec la probabilité ($1-P_l$) et l'envoie ($1-P_l$) Ms fois vers ses fils, il envoie pour le même paquet de données (M_s-1) NAKs vers la source et traite pour ces NAKs (M_s-2) expirations de délai de garde. Il reçoit en moyenne de N*(E [M_s]-1) NAKs de ces fils. Donc le temps de traitement est exprimé par :

$$E[A_s^D] = (1 - P_l)E[M_s](E[Y_p^a] + E[X_p^a]) + (E[M_s] - 1)E[Y_n^a] + E[(M_s - 2)\ ]E[A_t] + N*(E[M_s - 1)E[X_n^a]) \quad (13)$$

En remplaçant E [M_s], (E [M_s-1]) et (E [M_s-2]) par leurs valeurs correspondantes du tableau 5.1 nous obtenons l'expression :

$$E[A_s^D] = \frac{1}{1-P_l}(E[Y_p^a] + E[X_p^a]) + \frac{P}{1-P}(E[Y_n^a] + N*E[X_n^a]) + \frac{P^2}{1-P}E[A_t] \quad (14)$$

Le besoin de traitement au niveau de chaque récepteur. Pour chaque paquet de données, un récepteur reçoit qu'une seule fois le paquet de données grâce au service de *subcast*, il envoie (M_s -1) NAKs et traite pour ces NAKs (M_s -2) expirations de délai de garde. Si maintenant le récepteur est choisi comme répondeur, il envoie le paquet de données autant de fois qu'il reçoit de NAK**.**

Ce nombre est estimé à : $\frac{E[M_B]-1}{B}$. Le temps de traitement peut être exprimé par :

$$E[Y^D] = E[Y_p] + (E[M_s] - 1)E[Y_n] + E[(M_s - 2)\ ]E[Y_t] + \frac{E[M_B] - 1}{B}(E[X_p] + E[X_n]) \quad (15)$$

En remplaçant (E[M_s]-1) et (E[M_s-2]) par leurs valeurs correspondantes du Tableau 5.1 on obtient :

$$E[Y^D] = E[Y_p] + \frac{P}{1-P}E[Y_n] + \frac{P^2}{1-P}E[Y_t] + \frac{E[M_B] - 1}{B}(E[X_p] + E[X_n]) \quad (16)$$

4.3 Résultats numériques

Nous avons présenté dans les deux sous-sections précédentes le modèle analytique pour le calcul du temps de traitement par paquet pour chaque type de nœud, et ce pour chaque protocole. Dans ce paragraphe, nous utilisons ce modèle pour analyser numériquement les deux protocoles. L'étude est faite selon le débit atteint par chaque nœud et ensuite selon le débit global atteint par chaque protocole.

Le débit Λ_x^w atteint par le nœud x dans le protocole w $\in$ {**A, D**}, est calculé par la formule :

$$\Lambda^{w}_{x} = 1 / E[x^{w}],\ x \in \{X, Y, A_1 \ldots\ldots A_N, A_S\} \quad (17)$$

Le débit global Λ^w atteint par le protocole w est alors donné par :

$$\Lambda^{w} = Min\ (\Lambda^{w}_{x}) \quad (18)$$

Pour l'évaluation numérique du débit global nous prenons les mêmes mesures que celles qui ont été prises dans [39]:

$E[X_p] = E[Y_p] = E[X_p^a] = E[Y_p^a] = 500\,\mu\sec s$,

$E[X_n] = E[Y_n] = E[X_a] = E[Y_a] = E[X_n^a] = E[Y_n^a] = E[X_a^a] = E[Y_a^a] = 85\mu\sec s$,

$E[X_t] = E[Y_t] = 2E[A_t] = 32\mu\sec s$.

La figure 5.3 montre le débit atteint par chaque type de nœuds dans le protocole **A** en fonction du nombre de groupes locaux des récepteurs. Nous pouvons constater que : (**a**) le débit minimal est celui introduit par les routeurs actifs terminaux A_i, ceci est due principalement à la charge exercée sur ces derniers pour assurer le recouvrement local, (**b**) les récepteurs ont un débit maximal grâce au service de *subcast* qui permet aux récepteurs d'éviter le problème d'exposition, (**c**) le débit atteint par chaque type de nœuds demeure constant quelque soit le nombre de groupes locaux de récepteurs, (**d**) le débit global atteint par le protocole **A** est donc celui atteint par un routeur actif terminal A_i.

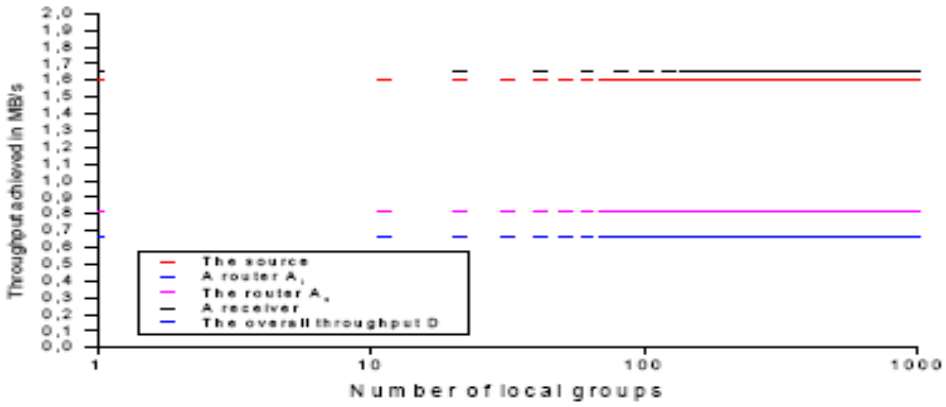

Figure 5.3 : Le débit atteint par les différents nœuds dans le protocole **A** (B=10, P=0.05).

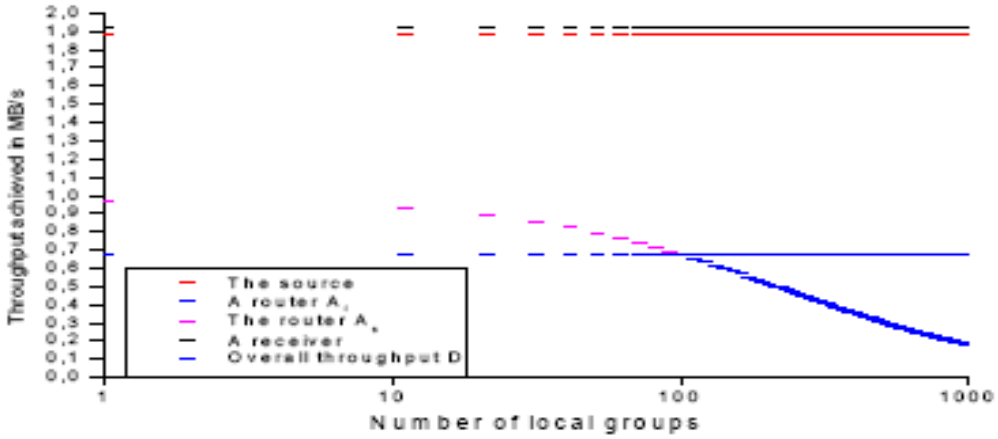

Figure 5.4 : Le débit atteint par les différents nœuds dans le protocole **D** (B=10, P=0.05).

La figure 5.4 montre le débit atteint par chaque type de nœuds dans le protocole **D** en fonction du nombre de groupes locaux des récepteurs. De la même manière que dans la figure précédente nous pouvons constater que : (**a**) le débit minimal est introduit au début par les routeurs actifs terminaux A_i, puis en augmentant le nombre de groupes locaux (environ 100) le routeur actif associé à la source A_s devient le goulot d'étranglement, (**b**) les récepteurs ont un débit maximal grâce au service de *subcast* qui permet aux récepteurs d'éviter le problème d'exposition, (**c**) le débit atteint par les autres types de nœuds demeure constant quelque soit le nombre de groupes locaux de récepteurs, (**d**) le débit global atteint par le protocole **D** est donc celui atteint par un routeur actif terminal A_i au début, puis par celui du routeur actif de la source A_s lorsque le nombre de groupes locaux de récepteurs augmente.

La figure 5.5 présente une comparaison des deux protocoles **A** et **D** en terme de débit global. Nous constatons qu'au début, celui-ci est presque le même pour les deux, avec un léger avantage pour le protocole **D**. Lorsqu'on augmente le nombre de groupes locaux des récepteurs (environ N> 95), le débit global du protocole **A** demeure constant, par contre celui, de **D** décroît d'une manière significative. Ce résultat s'explique par le fait que dans le protocole **A**, les acquittements positifs bénéficient d'un service de suppression locale et d'agrégation en combinant les deux approches hiérarchique et celle basée sur les temporisateurs. Par contre, dans le protocole **D**, les acquittements négatifs ne bénéficient que d'un service d'agrégation de l'approche hiérarchique. D'autre part, la combinaison des classes a permis une meilleure répartition de la charge de recouvrement des pertes entre la source et les récepteurs avec la contribution des routeurs actifs. La source s'occupe des pertes qui se produisent sur le lien source et les récepteurs s'occupent de celles qui se produisent sur les liens terminaux. Ceci permet au routeur actif, côté source, d'éviter de subir la charge des pertes des liens terminaux. Dans le protocole **D** par contre, le routeur actif, côté source, va subir toute la charge de toutes les pertes qui se produisent sur les liens terminaux. Par ailleurs, l'acquittement positif génère un flux en feedback une seule fois pour annoncer la bonne réception du paquet de données, alors que le flux généré par l'acquittement négatif se répète autant de fois, jusqu'à ce que le paquet de données soit correctement reçu par tous les récepteurs, lorsque la perte se produit sur le lien source.
La figure 5.6 montre qu'en augmentant la probabilité de perte le débit global décroît surtout pour le protocole **D**. Ce résultat nous incite à tracer les courbes du débit global des deux protocoles en fonction de la probabilité de perte.

La figure 5.7 montre que le débit global atteint par le protocole **D** est supérieur à celui de **A** lorsque la probabilité de perte varie dans l'intervalle [0, 0.05]. Par conséquent, si la probabilité de perte P est supérieure à 0.05, le débit atteint par le protocole **D** devient inférieur à celui de **A**.

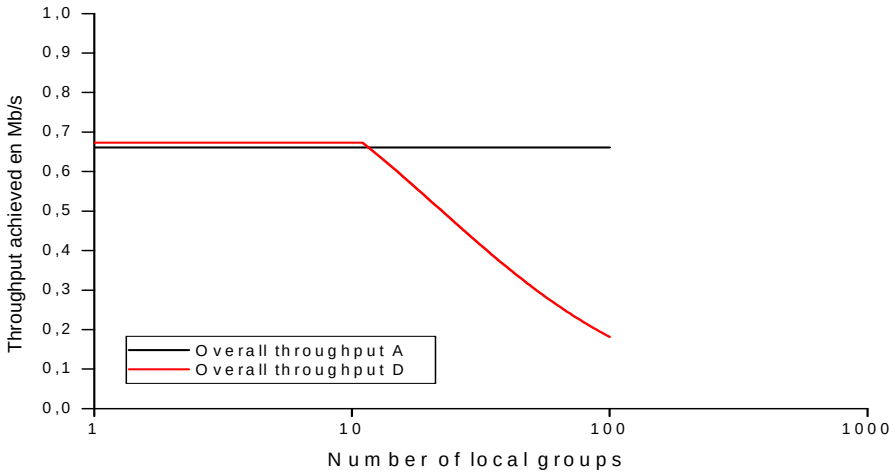

Figure 5.5 : Le débit atteint par les protocoles **A** et **D** (B=10, P=0.05).

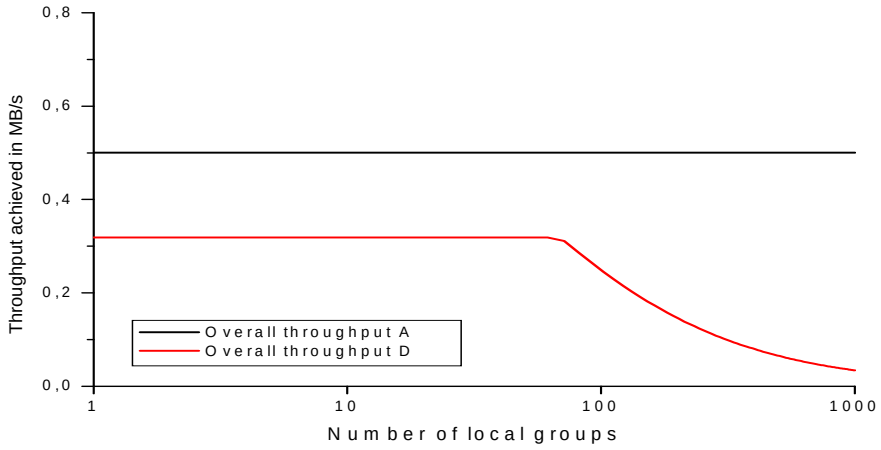

Figure 5.6 : Le débit atteint par les protocoles **A** et **D** (B=10, P=0.25).

La figure 5.8 montre que pour un nombre de groupes locaux assez important (N=500), la différence en rapidité de décroissance devient nettement claire. Le protocole **A** devient plus performant que le protocole **D** en terme de débit global lorsque la probabilité de perte est supérieure à (0.05). Cela est dû à la mauvaise répartition de la charge de recouvrement des pertes dans la classe *receiver-initiated* qui attribue la détection de pertes aux récepteurs quelque soit le lien sur lequel les pertes se produisent.

5. Analyse de la bande passante

Dans cette section, notre analyse va porter sur les besoins en terme de bande passante consommée par les protocoles AMRHy (**A**) et DyRAM (**D**). Nous utilisons des notations similaires à celles utilisées dans [28], celles-ci sont représentées dans le tableau 5.3.

Pour analyser la performance d'un protocole en terme de bande passante consommée, nous considérons trois types de lien : le lien source, le lien *backbone* et le lien terminal. La bande passante totale consommée sera donc donnée par l'expression suivante :

$$B^{w} = E[B_s^{w}] + N * E[B_b^{w}] + R * E[B_t^{w}] \quad (19)$$

Tableau 5.3 : Notations utilisées dans l'évaluation analytique de la bande passante.

[1]	B^w	[2]	La bande passante totale consommée par le protocole w.
[3]	B_s^w	[4]	La bande passante consommée sur le lien source par le protocole w.
[5]	B_b^w	[6]	La bande passante consommée sur le lien *backbone* par le protocole w.
[7]	B_t^w	[8]	La bande passante consommée sur un lien terminal par le protocole w.
[9]	B_p B_n B_a	[10]	La bande passante consommée par un paquet de données, un NAK ou un ACK.

Pour déterminer les différents termes de cette expression, nous avons besoin de trouver le nombre de paquets qui traverse ces liens (le lien source, le lien *backbone* et le lien terminal) pour qu'un paquet données transmis par la source, soit correctement reçu par tous les récepteurs. Pour simplifier notre analyse, nous considérons qu'il existe un même lien physique qui relie la source aux récepteurs.

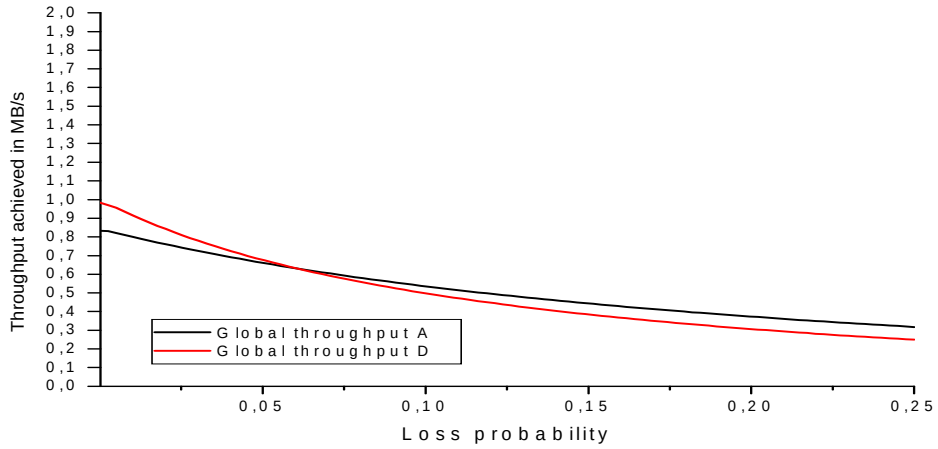

Figure 5.7 : Le débit atteint par les protocoles **A** et **D** (B=10, N=100).

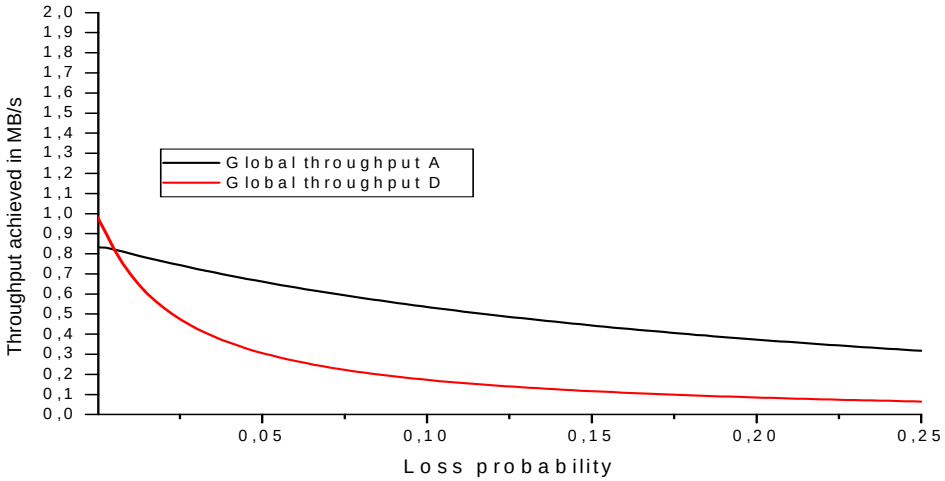

Figure 5.8 : Le débit atteint par les protocoles **A** et **D** (B=10, N=500).

5.1 Analyse du protocole AMRHy (A)

La bande passante consommée sur les différents liens par le protocole **A** est :

- *Lien source* : la source envoie M_s fois le paquet de données jusqu'à ce qu'il soit reçu par au moins un récepteur et reçoit par conséquent un seul ACK grâce aux services de suppression et d'agrégation.

$$E[B_s^A] = E[M_s]E[B_p] + E[B_a] \quad (20)$$

- *Lien backbone* : un routeur côté source expédie M_s fois le paquet de données jusqu'à ce qu'il soit reçu par au moins un récepteur et reçoit par conséquent un seul ACK grâce aux services de suppression et d'agrégation.

$$E[B_b^A] = (1 - P_l)E[M_s]E[B_p] + E[B_a] \tag{21}$$

- *Lien terminal* : un récepteur émet (M_s-1) fois le NAK jusqu'à ce qu'il reçoit le paquet de données et par conséquent envoie un seul ACK. Si le récepteur est élu répondeur, il envoie le paquet de données autant de fois qu'il reçoit le NAK. Ce nombre est estimé à $\frac{E[M_B]-1}{B}$.

$$\begin{aligned} E[B_t^A] &= E[B_p] + (E[M_s] - 1)E[B_n] + E[B_a] \\ &+ (\frac{E[M_B]-1}{B})(E[B_p] + E[B_n]) \end{aligned} \tag{22}$$

5.2 Analyse du protocole DyRAM (D)

La bande passante consommée sur les différents liens par le protocole **D** est :

- *Lien source* : la source envoie M_s fois le paquet de données jusqu'à ce qu'il soit reçu par au moins un récepteur et reçoit, par conséquent, (M_s-1) fois le NAK.

$$E[B_s^D] = E[M_s]E[B_p] + (E[M_s] - 1)E[B_n] \tag{23}$$

- *Lien backbone* : un routeur côté source expédie M_s fois le paquet de données jusqu'à ce qu'il soit reçu par au moins un récepteur et reçoit, par conséquent, (M_s-1) fois le NAK.

$$E[B_b^D] = (1 - P_l)E[M_s]E[B_p] + (E[M_s - 1)E[B_n] \tag{24}$$

- *Lien terminal* : un récepteur envoie (M_s-1) fois le NAK jusqu'à ce qu'il reçoit le paquet de données. Si le récepteur est élu répondeur, il envoie le paquet de données autant de fois qu'il reçoit le NAK. Ce nombre est estimé à $\frac{E[M_B]-1}{B}$.

$$\begin{aligned} E[B_t^D] &= E[B_p] + (E[M_s] - 1)E[B_n] \\ &+ (\frac{E[(M_B - 1)]}{B})(E[B_p] + E[B_n]) \end{aligned} \tag{25}$$

5.3 Résultats numériques

Pour l'évaluation de la bande passante, nous prenons les mêmes valeurs que celles qui ont été prises dans [28] : E [B_p] = 1024 octets et E [B_n] = E [B_a] = 32 octets.

La figure 5.9 montre qu'avec une probabilité de perte P=0.01 la bande passante consommée par **D** est inférieure à celle consommée par **A**.
La figure 5.10 montre qu'en augmentant la probabilité de perte P =0.05 la bande passante consommée par les deux protocoles **A** et **D** est la même.
La figure 5.11 montre qu'en augmentant encore la probabilité de perte (P=0.25), la bande passante consommée par le protocole **D** devient supérieure à celle consommée par le protocole **A**. Ce résultat s'explique par le fait que la combinaison des classes (la source détecte les pertes qui se produisent sur le lien source et les récepteurs détectent celles qui se produisent sur les liens terminaux) a permis au protocole **A** de réduire le flux d'acquittements négatifs surtout lorsque la perte se produit sur le lien source.

6. Conclusion

Nous avons évalué les performances, en termes de débit et de bande passante, de deux protocoles de multicast fiable dans le contexte de réseaux actifs. Le premier (**DyRAM)** appartient à la classe *receiver-initiated*, où la responsabilité de la détection de perte est attribuée aux récepteurs, sans tenir compte du lien sur lequel les pertes se produisent, entraînant ainsi une mauvaise répartition de la charge de recouvrement. Le deuxième (**AMRHy**) combine les deux classes *sender-initiated* et *receiver-initiated* où la responsabilité de la détection est distribuée entre la source et les récepteurs. Cette approche hybride tient compte du lien sur lequel les pertes se produisent, entrainant ainsi une répartition équitable et efficace de la charge de recouvrement entre la source et les récepteurs avec la contribution des routeurs actifs.

Pour évaluer ces deux protocoles en terme de débit, nous avons proposé des modèles analytiques pour calculer le temps de traitement par paquet pour chaque type de nœud, et ce pour chaque protocole. L'évaluation des performances est faite selon le débit global atteint par chaque protocole. L'analyse des débits atteints par chaque type de nœud, sert à connaitre l'élément qui introduit le débit minimal dans chaque configuration. Les résultats numériques ont montré que les routeurs actifs sont les éléments qui constituent le goulot d'étranglement, ceci est dû principalement à la charge du recouvrement local des pertes.

Pour évaluer ces protocoles en terme de bande passante, nous avons aussi proposé des modèles analytiques pour calculer la bande passante consommée par paquet sur chaque type de lien, et ce pour chaque protocole. Cette analyse sert à quantifier la bande passante consommée par chaque protocole.

L’étude comparative des deux protocoles nous a permis de montrer l’intérêt de la combinaison des classes adoptée par notre protocole par rapport à ceux basés uniquement sur la classe *receiver-initiated*. L’analyse analytique a montré le gain en termes de débit et de la bande passante de la combinaison des classes. Ce gain croit au fur et à mesure que la densité du réseau et la probabilité de perte augmentent.

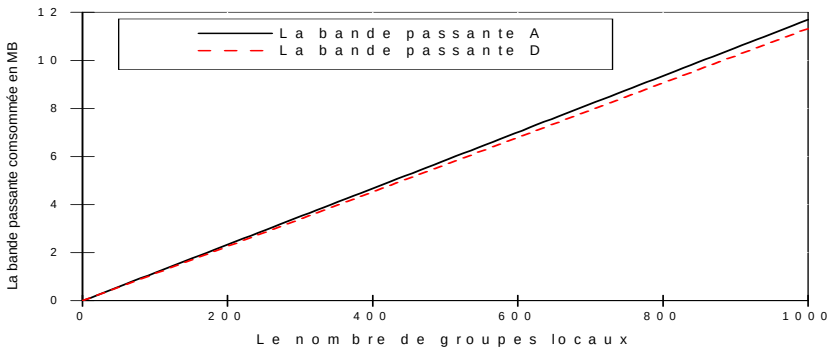

Figure 5.9 **:** La bande passante consommée par les protocoles **A** et **D** (B=10, P=0.01).

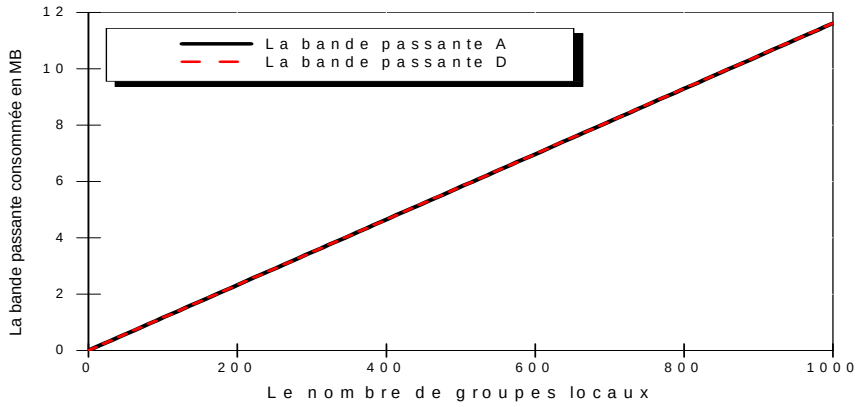

Figure 5.10 : La bande passante consommé par les protocoles **A** et **D** (B=10, P=0.05).

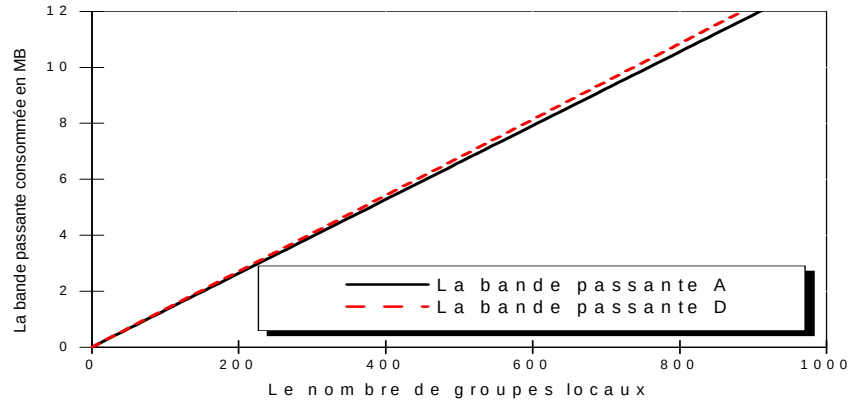

Figure 5.11 : La bande passante consommée par les protocoles **A** et **D** (B=10, P=0.25).

Conclusion générale

La croissance exponentielle de l'Internet et de ses applications, devient de plus en plus exigeante en termes de bande passante et de scalabilité. Un large spectre d'applications Internet courantes et émergeantes impliquent, dans leur communication, plusieurs parties et nécessitent ainsi, la disponibilité d'un mécanisme de communication multipoint. De nouvelles opportunités sont offertes à ces applications par la technologie multicast IP qui constitue un moyen efficace pour disséminer les données, tout en évitant les duplications multiples, à un grand nombre de destinataires. Pendant plusieurs années, le multicast a fait l'objet de recherches intenses, où une multitude de problèmes ont été abordés à différents niveaux. En dépit de l'effort considérable et du progrès dans la résolution des problèmes inhérents au multicast, il reste beaucoup de questions à traiter.

Dans cette thèse nous avons abordé un problème issu du niveau transport, où plus d'attention a été consacrée à la fiabilité multicast qui est essentiel pour plusieurs applications. Le souci est d'assurer une délivrance fiable, tout en garantissant le passage l'échelle (la scalabilité), à un grand nombre de récepteurs. Nous avons d'abord considéré le problème de recouvrement des pertes en proposant un nouveau protocole multicast fiable qui assure une utilisation efficace des ressources du réseau et celles des hôtes. Pour atteindre cet objectif, nous avons choisi d'adopter une solution plus dynamique et plus flexible fournit par le paradigme des réseaux actifs. Ce choix est motivé, principalement, par la disponibilité des informations réseau qui sont nécessaire pour développer des solutions sur la fiabilité plus précises. Ensuite, nous avons évalué analytiquement les performances de ce protocole. Pour valider notre proposition une étude comparative, avec le protocole de multicast fiable le plus récent DyRAM [40], a été réalisée.

a. Contributions

- Dans le chapitre 4 de cette dissertation, nous avons proposé un nouveau protocole de multicast fiable actif, nommé AMRHy. Ce dernier pallie aux inconvénients de la classe *receiver-initiated*, qui attribue la détection de pertes aux récepteurs, indépendamment du lien sur lequel la perte se produit, provoquant une mauvaise répartition de la charge de recouvrement des pertes. Toute fois, notre protocole combine les deux classes *sender-initiated* et *receiver-initiated*. Ceci, permet à la source de détecter les pertes qui se

produisent sur les liens proches d'elle (les liens sources) et aux récepteurs détectent celles qui se produisent sur des liens proches d'eux (les liens terminaux), garantissant ainsi une répartition équitable de la charge de recouvrement des pertes. AMRHy possède deux caractéristiques fondamentales, une qui concerne l'élection dynamique d'un répondeur, permettant d'instaurer un équilibre de la charge de recouvrement des pertes parmi les récepteurs appartenant à un même sous groupe ; la deuxième est la distribution équitable de la charge de recouvrement des pertes entre la source et les récepteurs en tenant compte du lien sur lequel la perte se produise. La répartition des routeurs actifs à travers l'arbre de distribution multicast permet à AMRHy de construire un arbre de recouvrement qui s'adapte parfaitement à la nature dynamique du groupe multicast. Cette arbre est utilisé pour accomplir la suppression et l'agrégation du flux, généré en feedback par les récepteurs au niveau des nœuds intermédiaires. Les services actifs ont été conçus d'une manière à assurer un support minimal des routeurs, afin de ne pas trop influencer la fonction principale d'expédition des routeurs qui appartiennent à l'arbre de distribution multicast.

- Le chapitre 5 a été consacré à l'évaluation analytique d'AMRHy. Le but était d'évaluer le potentiel des différents services actifs d'AMRHy à travers une étude comparative avec le protocole DyRAM. Cette évaluation a montré qu'AMRHy permet une répartition efficace de la charge sur les différents nœuds en termes de bande passante. Les différents services d'AMRHy permettent de réduire la latence de réparation et le problème d'implosion ce qui agit positivement sur le débit global. Finalement, nous avons prouvé qu'AMRHy possède de bonnes propriétés de scalabilité puisque les performances en termes de débit et de bande passante ne diminuent pas en augmentant le nombre de récepteurs. Les résultats numériques montrent que la combinaison des classes, adoptée par AMRHy, améliore de manière significative le débit et limite la bande passante requise par les messages de contrôle. Il est important de remarquer que les performances d'AMRHy surpassent celles de DyRAM, surtout lorsque la taille du réseau et la probabilité de perte sont importantes.

b. perspectives

Dans cette section nous indiquerons quelques orientations des travaux futurs dont certains constituent une suite naturelle des travaux représentés dans cette dissertation.

- Nous avons abordé un sujet du niveau transport concernant la fiabilité multicast, un autre sujet aussi important du même niveau est celui du contrôle de congestion dans une communication multicast. La résolution de ce problème nécessite l'élaboration des protocoles de contrôle de congestion qui permettent la prise en considération des états de

congestion, d'une manière scalable, dans les différentes parties de l'arbre de distribution multicast qui ne sont pas forcément les mêmes. Ces états sont utilisés par la source pour déterminer le débit global de transmission des paquets de données dans l'arbre de distribution multicast. Par ailleurs, le débit est ajusté de manière à assurer un partage équitable de la bande passante entre les sessions concurrentes multicast et unicast.

- Nous avons étudié les performances d'AMHRy en se basant sur deux métriques : la bande passante et le débit. Nous projetons élargir notre analyse à d'autres métriques tels que : le délai d'acheminement, la latence de recouvrement des pertes et son influence sur la taille du cache, au niveau des routeurs actifs, afin de déterminer le cache hit qui permet une latence minimale de recouvrements des pertes. D'autre part, les analyses de performances, que nous avons réalisées, sont de caractère analytique, nous prévoyons confirmer ces résultats dans un environnement de simulation tel que NS2 (*Network Simulator*).

- Les résultats obtenus montrent qu'AMRHy s'adapte parfaitement aux environnements de transmission très peu fiable et qui ont une population assez dense. A travers ces résultats, nous pouvons conclure que le protocole AMRHy a les potentialités nécessaires pour migrer facilement vers un environnement sans fil tel que : les réseaux maillés sans fil WMN (*Wireless Mesh Network*) [25]. Ces derniers constituent un paradigme émergeant pour la prochaine génération de l'Internet sans fil. Ils gagnent une attention particulière en offrant un environnement hautement flexible et une solution à moindre coût pour la couverture de vastes régions avec la technologie sans fil. Fournir une communication multicast fiable dans un tel environnement, où la maximisation du débit est en conflit avec la bande passante considérée comme une ressource rare, représente un défi.

- Les applications multicast peuvent être classées selon deux critères fondamentaux : la fiabilité et le délai de livraison. Dans cette dissertation seul le critère fiabilité a été pris en considération. Nous projetons inclure le critère délai exigé par les applications temps réel. Pour cela nous allons introduire le code FEC dans le protocole AMRHy. Par conséquent, notre protocole transmet les paquets de données dans des blocs de paquets et effectue les réparations en envoyant des paquets de réparation construits à partir des données originales. Ainsi, un bloc de paquets englobe k paquets de données et h paquets additionnels de réparation, où les h paquets de réparation sont construits à partir des k paquets originaux, avec h habituellement beaucoup plus petit que k. Si chaque récepteur reçoit k paquets des $k+h$ paquets transmis, celui-ci peut reconstruire les paquets de données originaux sans avoir recourt à la retransmission ce qui agit positivement sur le délai de livraison.

Bibliographie

[11] S. Alexander, M. Shaw, S. M. Nettles, J. M. Smith, Active Bridging, In Proceedings of SIGCOMM97, September 1997.

[12] E. Amir, S. McCanne, R.Katz, An Active Service Framework and its Application to Real-time Multimedia Transcoding, ACM Communication Review, vol. 28, no. 4, pp. 178-189, September 1998.

[13] A. Ballardie, Core Based Trees (CBT) Multicast Routing Architecture, Internet standard RFC 2201, September 1997.

[14] A. Benslimane, *Multicast Multimédia sur Internet*, Edited by Abderrahim Benslimane, *University of Avignon*, France, Junuary 2007.

[15] S. Bhattacharjee, K. Calvart, E. Zegura, An Architecture for Active Networking, High Performance Networking (HPN'97), White Plains, NY, April 1997.

[16] J. Biswas et al, The IEEE P1520 Standards Initiative for Programmable Network Interfaces, IEEE Communications Magazine, IEEE Communications Magazine, Vol. 36, no. 10, pp. 64-72, IEEE, October 1998.

[17] J. Byers, M. Luby, M. Mitzenmacher, A. Rege, A digital Fountain Approach to Reliable Distribution of Bulk data, ACM SIGCOMM Computer Communication Review, Vol. 28, no. 4, pp. 56-67, October 1998.

[18] 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, USA, March 1999.

[19] B. Cain, S. Deering, I. Kouvelas, B. Fenner, A. Thyagarajan, Internet Group Management Protocol, Version 3, Internet standard RFC 3376, October 2002.

[20] K.L. Calvert, Architectural Framework for Active Networks, University of Kentucky, July 1999.

[21] A.T. Campbell, H.G. De Meer, M.E. Kounavis, A Survey of Programmable Networks, ACM Computer Communications Review, vol. 29, no. 2, pp. 7-23, April 1999.

[22] T. Chen, et al., Commentaries on Active Networking and End -to-End Arguments, *IEEE Network*,1998, Special Issue on Active and Controllable Networks.

[23] S. E. Deering, Host Extensions for IP Multicasting, Internet standard RFC 988, July 1986.

[24] S. E. Deering, Host Extensions for IP Multicasting, Internet standard RFC 1054, May 1988.

[25] S. E. Deering, Host Extensions for IP Multicasting, Internet standard RFC 1112, August 1989.

[26] S. E. Deering, R. Hinden, Internet Protocol, Version 6 (IPv6) Specification, Internet standard RFC2460, December 1998.

[27] L. Derdouri, D. E. Saidouni, M. Benmohammed, Reliable Multicast Transport in Active Environment, in *Proceeding CSIT'06 Conference Computer Science and Information Technologie*, Amman, Jordan, Mars 2006.

[28] W. Fenner, Internet Group Management Protocol, Version 2, Internet standard RFC 2236, November 1997.

[29] P. Ferguson, D. Senie, Network Ingress Filtering: Defeating Denial of Service Attacks which employ IP Source Address Spoofing. Internet Standard RFC 2827, May 2000.

[30] S. Floyd, V. Jacobson, S. Mc Canne, C-G. Liu, L.Zhang, A reliable multicat framework for light-weight sessions and applications level framing, *IEEE/ACM Transactions on Networking*, Vol.5, no. 6, pp. 784-803, December 1997.

[31] B. Haberman, Unicast-Prefix-Based IPv6 Multicast, Internet standard RFC 3306, August 2002.

[32] R. Hinden and S. Deering, IP version 6 Addressing Architecture, Internet standard RFC 2373, July 1998.

[33] M. Hofmann, Enabling group communication in global networks, in *Proceeding Global networking'97*, Calgary, Alberta, Canada, November 1996.

[34] H. W. Holbrook, S. K. Singhal, D. R. Cheritoy, Log-based receiver reliable multicast for distributed interactive simulation, in *ACM SIGCOMM'95*, pp. 328-341, October 1995.

[35] E. Hossain, K. Leung, *Wireless Mesh Networks Architectures and Protocols*, Springer-Verlag New York Inc, December 2007.

[36] C. Huittema, *Le routage dans l'internet*, éditions : Eyrolles, Octobre 1994.

[37] S. A. Hussain, *Active and Programmable Networks for Adaptive Architectures and Services*, Auerbach Publications by Taylor & Francis Group, LLC, December 2006.

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

[39] S. K. Kasera, and al., Scalable Fair Reliable Multicast Using Active Services, *IEEE Network Magazine*, Vol. 14, no. 1, pp. 48-57, Jan/ Feb 2000.

[40] U. Legedza, J. Guttag, Using Network-level Support to Improve Cache Routing, 3rd International WWW caching Workshop, Manchester, England, June 1998.

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

[42] B. N. Levine, J. J. Garcia-Luna-Aceves, A comparison of reliable multicast protocols, *Multimedia systems*, 6, pp. 334-348, Spinger-Verlag, 1998.

[43] D. Li and D.R. Cheriton, Evaluating the utility of FEC with Reliable Multicast, in *Proceedings of the Seventh Annual international Conference on Network Protocols*,Oct/Nov 1999.

[44] Q. Li, T. Jinmei, and K. Shima, *IPv6 Advanced Protocols Implementation*, Morgan Kaufmann, Elsevier, 2006.

[45] P. Lothberg, D. Meyer,G LOP Addressing in 233/8, Internet standard RFC 2770, February 2000.

[46] P. Lydia, L. Wei, et al., TCP/IP Tutorial and Technical Overview, IBM Press, Redbook, IBM Form Number GG24-3376-07, December 2006.

[47] C. Maïhofer, A Bandwidth Analysis of Reliable Multicast Transport Protocols, Proceedings of the Second International Workshop on Networked Group Communication, November 2000.

[48] C. Maïhofer, and al., A Delay Analysis of Tree-based Reliable Multicast Protocols, in Proceedings of the Tenth International conference on computer communication and networks, Scottscale, USA, IEEE, October 2001.

[49] M. Maimour and C. Pham, A throughput Analysis of Reliable Multicast Protocols in Active Networking Environment, in Proceeding of the sixth IEEE Symposium on Computers and Communications, 2001.

[50] M. Maimour, C. Pham, "DyRAM: an Active Reliable Multicast framework for Data Distribution, *Journal of Cluster Computing*, vol. 7(2), pp. 163-176, April 2004.

[51] A. Mankin et al., IETF Criteria for Evaluating Reliable Multicast Transport and Application Protocols, Internet standard RFC 2357, June 1998.

[52] D. Minoli, *IP Multicast With Applications to IPTV and Mobile DVB-H*, Published by Wiley-IEEE Press, April 2008.

[53] D. Meyer, Administratively Scoped IP Multicast, University of Oregon, Internet standard RFC 2365, July 1998.

[54] J. Moy, Multicast Extensions to OSPF, Internet standard RFC 1584, March 1994.

[55] J. Nicholas, W. Siadak, Protocol Independent Multicast—Dense-Mode (PIM—DM): Protocol Specification (Revised), Internet standard RFC 3973, January 2005.

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

[57] C. Papadopoulos, G. Parulkar, nand G. Varghese, An Error Control Scheme for Large Scale Multicast Applications, in Proceedings of the seventeenth annual ACM symposium on Principles of distributed computing Proceedings of the seventeenth annual ACM symposium on Principles of distributed computing, 1988.

[58] S. Paul and K. Sabnani, A Reliable Multicast Transport Protocol, *IEEE JSAC, spec. Issue on Network Support for Multipoint Communications*, Vol. 15, no. 3, April 1997.

[59] S. Pingali, D. Towsley, and J.F. Kurose, A comparison of Sender-initiated and Receiver-initiated Reliable Multicast Protocols, ACM SIGMETRICS Performance Evaluation Review, Vol. 22 , no. 1, pp. 221-230, May 1994.

[60] D. Rubenstein, S. Kasera, D. Towsley, and J. Kurose, Improving Reliable Multicast Using Active Parity Encoding Services (APES), Computer Science Department.. University of Massachusetts at Amherst. Technical Report98-79. Department of Computer Science, July 1998.

[61] D. Rubenstein, N.F. Maxemchuk, D. Shur, A centralized approach to network repair service for multicast streaming media, in Proceedings of NOSSDAV'00, Chapel Hill, NC, June 2000.

[62] 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*, Vol. 24, no. 5-6, pp. 563-574, March 2001.

[63] D. Rubenstein, S. Kasera, D. Towsley, and J.Kurose, Improving Reliable Multicast Using Active Parity Encoding Services (APES), *Computer Networks* , Vol. 44, pp.63-78, 2004.

[64] B. Schwartz et al. Smart Packets: Applying Active Networks to Network Management. ACM Transactions on Computer Systems, Vol. 18, No. 1, February 2000.

[65] P. Sharma, D. Estrin, S. Floyd, and L. Zhang. Scalable Session Messages in SRM using Self-configuration. In USC Technical report, July 1998.

[66] T. Speakman and al, PGM reliable multicat protocol, *IEEE Network*, Vol. 7, no. 1, pp. 16-22, February 2003.

[67] D. L. Tennehouse, J. M. Smith, W. D. Sincoskie, D. J. Wethererall, and G. J. Winden, A survey of active network research, *IEEE Communication Magazine*, pp. 80-86, January 1997.

[68] D. Waitzman, C. Partridge, S. Deering, Distance Vector Multicast Routing Protocol, Internet standard RFC 3973, November 1988.

[69] D. Wetherall, U. Legedza, J. Guttag, Introducing New Internet Services: Why and How, *IEEE Network. Special Issue on Active and Programmable Networks*, Vol. 12, no. 3, pp. 12-19, May/Jun 1998.

[70] D. Wetherall, Service Introduction in Active Network, PhD thesis, Massachusetts Institute of Technology, February 1999.

[71] M. Yajnik, J. Kurose, and D. Towsley, Packet Loss Correlation in the MBone Multicast Network, in Proceedings of Global Telecommunications Conference, GLOBECOM'96, November 1996.

[72] R. Yavatkar, J. Griffoen, and M. Sudan, A reliable dissemination protocol for interactive collaborative applications, in Proceeding of the third ACM international conference on Multimedia, 1995.

[73] K. L. Yeung, H. T. Wong, Caching policy design and cache allocation in active reliable multicast, *Computer Networks , Elsevier*, 43 pp. 177-193, February 2003.

[74] Cisco Systems, Internet Protocol (IP) Multicast Technology Overview, White Paper, Cisco Systems, Inc., San Jose, CA, 2007.

[75] http://www.jukie.net/~bart/multicast/LinuxMroutedMiniHOWTO. Html

[76] IPv6Forum, IPv6Vendors TestVoice Wireless and Firewalls onMoonv6, http://www.ipv6forum.com/modules.php, November, 2004.

[77] OPENSIG: Open Signalling and Service Creation. http://comet.comlumbia.edu/opensig/

[78] Tcl (Tool Command Language). Disponible sur http://www.tcl.tk/software/tcltk/.

[79] http://squis.nlanr.net/Squid.