

République Algérienne Démocratique et Populaire
Ministère de l'Enseignement Supérieur et de la Recherche Scientifique
Université de Mentouri – Constantine
Faculté des sciences de l'ingénieur
Département d'informatique

Magistère en informatique
Option : Génie logiciel et Intelligence artificielle

Mémoire présenté en vue de l'obtention du diplôme de
magistère en informatique
sous le thème de :

Analyse du cache des routeurs actifs
dans les protocoles multicast fiable

N° ordre :

Série :

Présenté par : Mlle. Bendali_Amor Doumia

Soutenu publiquement le : / /2008 devant le jury composé de :

Le président du jury : Dr. A. Chaoui (Université de Constantine)

Rapporteur : Pr. M. Benmohammed (Université de Constantine)

Examineur : Dr. S. Chikhi (Université de Constantine)

Examineur : Dr. A. Billami (Université de Batna)

Sommaire

<i>Introduction générale</i>	01
<u>Chapitre I : Généralités</u>	03
I. Introduction.....	04
II. Le IP multicast.....	04
II.1. Le protocole "Internet Group Management Protocol" (IGMP).....	05
II.2. Le schéma d'adressage de l'IP multicast.....	06
II.3. Le routage multicast.....	07
II.4. La mise en service des transmissions multicast.....	08
III. Les protocoles de routage multicast.....	08
III.1. Les approches de routage.....	09
III.1.1 Les protocoles en mode dense.....	09
A. Le protocole DVMRP "Distance Vector Multicast Routing Protocol".....	09
B. Le protocole PIM-DM "Independent Multicast Dense Mode".....	10
III.1.2 Les protocoles en mode épars.....	10
* Le protocole PIM-SM "Independent Multicast Sparse Mode".....	10
IV. Les routeurs actifs.....	11
IV.1. Introduction.....	11
IV.2. L'intérêt des routeurs actifs.....	12
IV.3. La mise en œuvre d'un réseau actif.....	12
IV.3.1 Le model fonctionnel des réseaux actifs.....	13
IV.3.2 Les types de réseau actif.....	14
IV.4. Les applications des réseaux actifs.....	16
V. Conclusion.....	16
<u>Chapitre II : La fiabilité multicast dans les réseaux best effort</u>	17
I. Introduction.....	18
II. Les applications du multicast.....	18
III. La fiabilité de transmission des flux multicast.....	19
IV. Le recouvrement de bout-en-bout.....	20
IV.1. Le recouvrement basé sur l'initiative de la source.....	20
IV.2. Le recouvrement basé sur l'initiative des récepteurs.....	21
V. Le recouvrement local.....	22
V.1. L'architecture en arbre.....	23

V.2. Les temporisateurs.....	25
V.3. Les codes détecteur/correcteur d'erreurs.....	25
V.4. Les services actifs.....	25
VI. Conclusion.....	26
<u>Chapitre III : Les protocoles de fiabilité multicast dans les réseaux actifs</u>	27
I. Introduction.....	28
II. Le protocole "Active Error Recovery" : AER.....	29
II.1. Vue d'ensemble du protocole AER.....	29
II.2. L'identification des serveurs de réparation.....	29
II.3. La technique de recouvrement des erreurs.....	29
II.4. La stratégie de libération du cache mémoire.....	30
III. Le protocole "Multicast Actif Fiable" : MAF.....	31
III.1. Vue d'ensemble du protocole MAF.....	31
III.2. La stratégie d'élection des répondeurs.....	32
III.3. La gestion du recouvrement des pertes.....	32
III.4. La gestion du cache mémoire.....	33
IV. Le protocole "Dynamic Relier Active reliable Multicast": DyRAM.....	34
IV.1. Vue d'ensemble du protocole DyRAM.....	34
IV.2. Les services actifs.....	35
IV.3. La structure des paquets.....	37
IV.4. Le comportement des différents éléments du réseau.....	38
IV.5. Les algorithmes fonctionnels du protocole DyRAM.....	42
V. Conclusion.....	48
<u>Chapitre IV : Le protocole AMRHy</u>	49
I. Introduction.....	50
II. Motivation.....	50
III. Présentation générale.....	51
III.1. La gestion des paquets de contrôle.....	52
III.2. La répartition de la charge de recouvrement.....	53
III.3. L'exploitation de la bande passante et des éléments du réseau.....	54
IV. Particularité du protocole AMRHy.....	54
V. Description fonctionnelle du protocole AMRHy.....	55
V.1. Comportement de la source.....	55
V.2. Comportement du récepteur.....	55
V.3. Comportement du routeur actif.....	56
VI. La structure des paquets.....	57

VI.1. Les paquets de données.....	57
VI.2. Les paquets de contrôle.....	58
VII. L'algorithme fonctionnel du protocole AMRHy.....	58
VII.1.Comportement de la source.....	58
VII.2.Comportement des récepteurs.....	59
VII.3.Comportement des routeurs actifs.....	61
VIII. Conclusion.....	63
<i>Chapitre V : Simulation comparative des protocoles AMRHy et DyRAM.....</i>	64
I. Introduction.....	65
II. Motivation.....	65
III. Le simulateur "Network Simulator 2" : NS2.....	66
IV. Le déroulement de la simulation.....	66
IV.1. La topologie du réseau.....	66
IV.2. Le comportement du réseau.....	68
V. L'étude comparative.....	69
V.1. Analyse du trafic engendré par les deux protocoles.....	69
V.2. Analyse du besoin en mémoire cache au niveau des routeurs actifs.....	73
V.3. Analyse du délai d'acheminement des paquets de données.....	76
V.4. Récapitulatif de l'ensemble de l'analyse accomplie.....	79
VI. Les améliorations apportées au protocole AMRHy.....	82
VII. Conclusion.....	85
<i>Conclusion et perspectives.....</i>	86
<i>Annexe.....</i>	88

Introduction

La communication numérique à travers Internet est de nos jours une partie omnipotente de nos activités quotidiennes. On communique soit en mode unicast quand on désire communiquer avec une destination précise soit en mode broadcast lorsqu'on désire atteindre l'ensemble des internautes de la planète. Avec les nouvelles applications émergentes comme la vidéoconférence, la vidéo à la demande ou les applications coopératives un nouveau mode de communication est apparu : le mode multicast. Celui-ci est un mécanisme de communication multipoint qui permet la diffusion d'un paquet, d'une source vers un sous-ensemble des hôtes d'un réseau. Le multicast IP fournit, au niveau de la couche réseau, un support efficace pour ce type d'applications. En plus de l'efficacité de routage, certaines de ces applications nécessitent une livraison fiable des différentes données transmises. Fournir une délivrance multicast fiable et efficace à grande échelle est un défi, particulièrement lorsque l'application nécessite un délai de livraison très court et une capacité en terme de bande passante assez élevée. Plusieurs protocoles ont été élaborés pour résoudre le problème de la fiabilité dans les réseaux à délivrance dite *best effort* tel que l'Internet où la perte de paquets est loin d'être rare. Les protocoles de multicast fiables, *Reliables Multicast (RM)*, traitent ce problème en imposant un compromis entre le délai d'acheminement et la capacité en bande passante. Les premiers protocoles multicast adoptaient une approche de bout en bout, qui fait intervenir la source et les récepteurs, pour assurer le recouvrement des pertes. Cependant cette approche souffre du problème de l'implosion des acquittements au niveau de la source de données. Afin d'alléger la charge de cette dernière et réduire le délai de recouvrement des pertes une approche de recouvrement local a été adoptée. Cette dernière consiste à désigner un récepteur ou toute autre entité qui se charge du recouvrement des pertes localement. Néanmoins, la solution proposée par ces derniers se limite à une échelle restreinte.

L'introduction du concept des réseaux actifs dans la fiabilité multicast a orienté les recherches vers l'exploitation des services actifs fournis par les routeurs pour élaborer des protocoles multicast fiables qui s'appliquent à une grande échelle offrant ainsi une solution plus générale et plus flexible. Les services actifs consistent principalement à faire :

- Le cache des paquets de données pour assurer le recouvrement des pertes localement afin de permettre d'une part la répartition de la charge de recouvrement entre la source et les

routeurs actifs et d'autre part la réduction d'une manière considérable de la latence de recouvrement des pertes.

- L'agrégation et/ou la suppression locale des NAK's pour éviter le problème de l'implosion des NAK's au niveau de la source.

Le travail que nous proposons consiste en l'étude et l'analyse des performances de deux protocoles multicast fiables basés sur les services actifs à savoir : "Active Multicast Reliable Hybrid protocol" (AMRHy) [Réf.17] et "Dynamic Replier Active reliable Multicast" (DyRAM) [Réf.23]. L'analyse de performance va porter sur trois métriques : la bande passante, le délai d'acheminement et la taille du cache mémoire au niveau des routeurs actifs. Cette analyse va nous permettre de montrer l'intérêt de la combinaison des classes basées sur l'initiative de la source "sender-initiated" et des classes basé sur l'initiative des récepteurs "receiver-initiated" adoptée par AMRHy par rapport à la classe basé sur l'initiative des récepteurs "receiver-initiated" uniquement adoptée par DyRAM. Dans la classe des protocoles de recouvrement basé sur l'initiative des récepteurs "receiver-initiated" la détection des pertes est attribuée aux récepteurs indépendamment du lien sur lequel la perte se produit. Par contre dans la combinaison des classes, le lien sur lequel se produit la perte est pris en compte ; la source détecte les pertes qui se produisent sur les liens source et les récepteurs détectent celles qui se produisent sur les liens terminaux. Ce qui permet de réaliser une meilleure répartition de la charge de recouvrement des pertes entre la source et les récepteurs avec la contribution des routeurs actifs. Pour accomplir ce travail nous allons implémenter les deux protocoles (AMRHy et DyRAM) dans un environnement de simulation NS2. Le choix de ces deux protocoles est motivé par le fait que ces derniers choisissent un répondeur parmi les récepteurs pour assurer le recouvrement des pertes localement. Cette élection a pour objectif de réduire la charge du routeur actif en termes de taille du cache et du temps de traitement. Néanmoins, chaque protocole adopte une approche particulière pour résoudre le problème de fiabilité en multicast.

Le présent mémoire est organisé comme suit : le premier chapitre présente des généralités relatives au domaine des réseaux de transmission ainsi qu'au mode de transmission multicast. Le deuxième chapitre est consacré à la fiabilité des transmissions multicast ainsi qu'aux différentes approches et protocoles élaborés dans le but d'atteindre une fiabilité totale et efficace des transmissions multicast. Le troisième chapitre représente un état de l'art de trois protocoles multicast fiables basés sur les services actifs. Chaque protocole est décrit brièvement en mettant en évidence les principaux points forts du protocole ainsi que les

inconvénients qu'il présente. Dans le quatrième chapitre, on présente le protocole AMRHy élaboré au sein du laboratoire LIRE de l'université de Constantine. Le cinquième chapitre est consacré à la simulation NS2 des deux protocoles AMRHy et DyRAM ainsi qu'aux résultats obtenus et leur analyse. La fin de ce chapitre, représente les améliorations apportées au protocole AMRHy suite aux résultats obtenus durant sa simulation. Pour clôturer ce mémoire, une conclusion des résultats obtenus grâce à ce travail et des perspectives futures seront présentés.



Remerciement

Je remercie Dieu tous puissant d'avoir guidé mes pas vers la science.

Je remercie toute ma famille pour sa confiance et son soutien. Sans eux : Maman, Nana, Nesrine, Anis, Mémé, Fouzia, Fatiha, Baya, Hassiba, Habiba, mes oncles et tous mes petits cousins et cousines, je ne serai surement pas arrivée là.

Je remercie Pr. Benmohammed pour la confiance qu'il a mise en moi et j'espère être à la hauteur de sa confiance.

Je remercie M. Derdouri pour son aide et son soutien. Ce travail est en grande partie grâce à sa persévérance et ses conseils.

A tous Merci.....

"Les réseaux informatiques, tout comme l'imprimerie il y a cinq cents ans, permettent au citoyen ordinaire de s'exprimer de différentes façons et de toucher un panel d'individus plus varié et plus large qu'auparavant. Hélas, le bien et le mal coexistent dans ce monde et l'Internet est un monde en lui-même."

-Andrew Tanenbaum-

Je dédie ce mémoire à Anis mon futur époux

« Comme promis je l'ai fait pour toi !!! »

"La science ne cherche pas à énoncer des vérités éternelles; loin de prétendre que chaque étape est définitive, elle cherche à cerner la vérité par approximations successives."

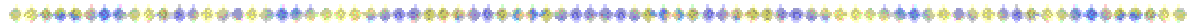
- Bertrand Russell-

Chapitre I

Généralités

*“Dans les sciences, le chemin est plus important que le but.
Les sciences n'ont pas de fin”*

- Erwin Chargaff -



I. Introduction

La transmission IP multicast a fait son entrée dans le monde des télécommunications il y a presque deux décennies de cela. Cependant, elle n'a rien perdu de sa popularité. Le nombre d'utilisateurs ne fait qu'augmenter de jour en jour et les travaux de recherche concernant ce mode de transmission sont nombreux. Deux problèmes majeures de ce mode de transmission intriguent les chercheurs et désappointent les utilisateurs : le manque de fiabilité et la congestion fréquente du réseau. Aussi, plusieurs travaux ont été consacrés uniquement à la résolution de ces deux problèmes.

La version IP multicast décrite dans le RFC 1112 [1] n'introduit aucun mécanisme assurant la fiabilité de la transmission. Donc, la fiabilité doit être assurée au niveau des couches supérieures de la pile protocolaire du réseau. Cette problématique a été largement explorée durant ces dernières années et un nombre très important de travaux ont abouti à une panoplie de techniques et d'approches de recouvrement des erreurs survenues lors de la transmission. Par ailleurs, des chercheurs renommés ont donné naissance au concept de réseaux actifs. Ce concept a révolutionné le recouvrement d'erreurs dans les réseaux multicast.

Ce chapitre présentera des généralités liées à la transmission multicast. On survolera les différentes techniques et approches qui existent pour une transmission multicast. Dans ce chapitre, on se focalisera aussi sur l'émergence du concept des services actifs dans les réseaux et leurs apports dans le recouvrement d'erreur.

II. Le IP multicast

La découverte de l'IP multicast peut être affiliée à un doctorant de l'université de Stanford "Steven Deering". Vers la fin des années 80, le travail de ce doctorant était de trouver un mécanisme permettant le transport de flux multicast à travers les sous-réseaux IP [2]. En Décembre 1991, "Steven Deering" soutient sa thèse intitulée : " Un routage multicast dans les réseaux à transmission datagramme " [multicast routing in a datagram network]. Cette thèse a donné naissance au premier document IETF sur la transmission IP multicast [1].

Le multicast désigne une transmission d'une source vers un groupe de destinataires. L'IP multicast est une forme de diffusion de données efficace : une seule copie de la donnée est transmise simultanément à tous les récepteurs du groupe. Cette technique permet non seulement de réduire la surcharge au niveau des routeurs du réseau mais aussi l'utilisation efficace de la bande passante.

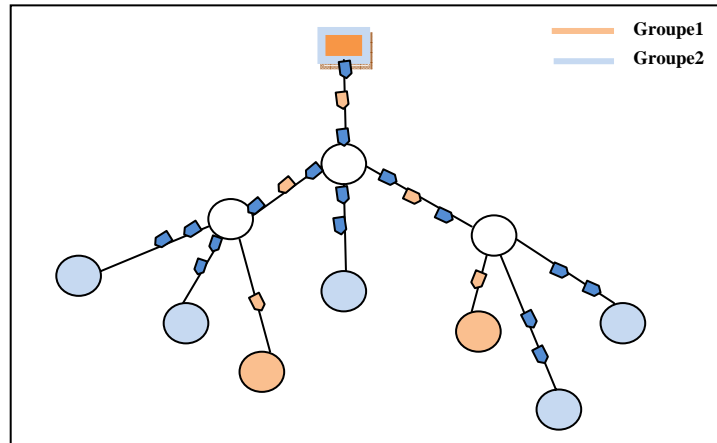


Fig1.1 La transmission en mode multicast

Dans une transmission multicast, la source transmet des paquets à une adresse de groupe. Les récepteurs voulant recevoir les paquets se joignent au groupe concerné. Pour cela, ils utilisent le protocole IGMP "Internet Group Management Protocol" [4] pour informer le routeur local de leur volonté.

II.1. Le protocole "Internet Group Management Protocol" (IGMP)

IGMP fait partie du protocole IP et comprend l'échange de deux types de messages. Le premier est un message d'interrogation émis par le routeur (host membership query) et le deuxième est un message de réponse émis par les hôtes du routeur (host membership report).

- **Interrogation:** Le routeur envoie une sollicitation aveugle à l'adresse 224.0.0.1 toutes les 60 secondes avec un TTL = 1 et pose la question « A quel groupe voulez vous vous abonner? ».
- **Réponse:** Les hôtes renvoient alors leur réponse dans laquelle ils spécifient les groupes auxquels ils souhaitent s'abonner. Pour cela, ils fixent un délai aléatoire avant de répondre de manière à éviter que toutes les réponses de tous les hôtes ne parviennent au routeur en même temps. Lorsqu'un hôte a répondu, les autres du même groupe n'ont plus besoin de répondre car le routeur va continuer d'envoyer les paquets vers ce groupe.

Le routeur temporise par la suite, et re-sollicite à nouveau périodiquement les hôtes. Si le routeur ne reçoit aucune réponse pour un groupe donné, alors il arrête la réémission des paquets vers ce groupe et le considère comme sans abonné.

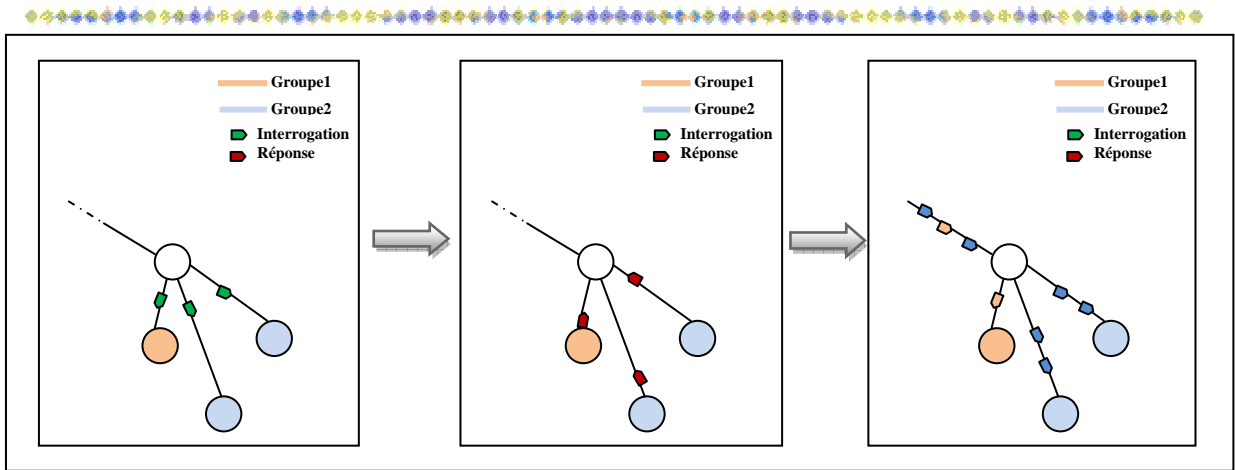


Fig.1.2. Gestion des groupes multicast avec IGMP

Remarque :

Dans le cas de plusieurs routeurs, un routeur est élu parmi tous les autres, il est appelé le routeur désigné « *Designated Router* » (*DR*). Il est le seul à émettre les messages *IGMP* mais ce n'est pas lui pour autant qui envoie forcément les paquets multicast. Dans la version 1 d'IGMP, le mécanisme d'élection est externe à IGMP.

Dans la version 2 d'IGMP, c'est le routeur qui possède la plus petite adresse IP qui est désigné comme DR.

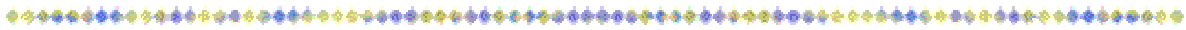
II.2. Le schéma d'adressage de l'IP multicast

Les paquets multicast sont identifiés grâce à une adresse IP de classe D comprise entre 224.0.0.0 à 239.255.255.255. Dans cette adresse, les 28 bits de poids faible identifient le groupe lui-même. Une partie (23 bits de droite) de ces 28 bits sera utilisée pour former l'adresse multicast Ethernet. Ainsi, une adresse multicast Ethernet permet de couvrir un ensemble de groupes multicast IP [9,10].



Fig.1.3. Schéma d'adressage de l'IP multicast

Les quatre (04) premiers bits de ces adresses correspondent toujours à la suite de bits de valeur "1110". Cela permet aux routeurs d'identifier rapidement la nature des paquets de données reçus.



II.3. Le routage multicast

L'acheminement des paquets multicast peut se faire grâce à 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.

L'acheminement du paquet peut aussi passer par un simple routeur où la notion de tunnel multicast est nécessaire : les routeurs sont des stations tunnels. 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 tunnel encapsule les paquets multicast destinés à d'autres parties du réseau dans des paquets unicast IP et les expédie à travers les tunnels appropriés. L'ordinateur destinataire, équipé d'un logiciel supportant le routage multicast IP, est chargé de la diffusion de ces paquets entre son sous-réseau et le réseau distant.

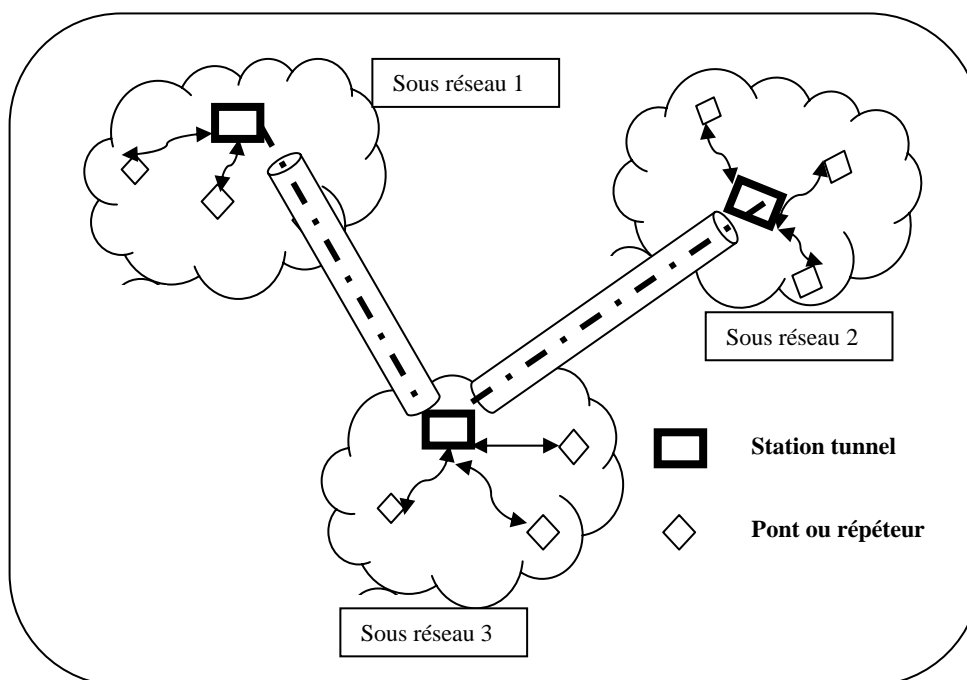


Fig.1.4 Acheminement des paquets multicast « stations tunnels »

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. Par conséquent, des protocoles de routage tels que CBT "Core Based Trees" [5], DVMRP "Distance Vector multicast Routing Protocol" [6], MOSPF "multicast Open Shortest



Path First" [7] et PIM "Protocol Independant multicast" [8] permettent d'acheminer les paquets transmis à tous les récepteurs membres d'un groupe.

II.4. La mise en service des transmissions multicast

En 1992, l'Internet Engineering Task Force (IETF) décide de construire un réseau virtuel au dessus du traditionnel réseau Internet : le MBone. Le MBone (Multicast Backbone) utilise l'Internet pour véhiculer un trafic multicast. Il se situe au dessus de la couche IP du réseau Internet afin de supporter un routage IP multicast entre des machines. En utilisant l'infrastructure des réseaux actuels, MBone évite le besoin d'avoir des réseaux dédiés pour la communication multimédia. Les informations transmises par le MBone sont acheminées dans le réseau Internet d'une manière standard [3].

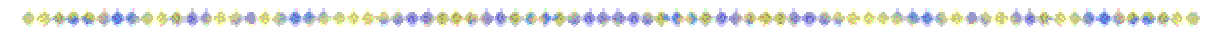
III. Les protocoles de routage multicast

Le routage d'un flux multicast est un peu plus complexe que le routage d'un flux unicast ou même broadcast. Une adresse multicast identifie une session de transmission particulière plutôt qu'une destination physique spécifique. Une session multicast peut concerner une multitude de récepteurs. Ainsi, la source n'est pas sensée connaître l'identité de tous ces récepteurs. Et par conséquent, les routeurs sont chargés de transformer les adresses multicast en adresses physiques pour chaque récepteur concerné. Ce type de routage se base sur l'interaction entre les routeurs du réseau afin d'échanger les informations concernant le voisinage de chacun d'eux.

Les données multicast sont dispatchées sur le réseau grâce à un arbre de diffusion établi par les routeurs du réseau. L'arbre de diffusion est construit en fonction de la connexion et l'accessibilité de tous les récepteurs du groupe multicast en cours. Cet arbre permet de spécifier le parcours des paquets de données multicast de la source vers chaque membre du groupe. Il existe deux types d'arbres de diffusion de base : ceux basés source et ceux distribués [28]

Le premier type d'arbre est construit à partir de l'emplacement de la source de données. Les routeurs construisent l'arbre en recherchant le plus court chemin entre la source et les récepteurs. Cela aboutit à une multitude d'arbres qui émanent des sous-réseaux directement reliés à la source.

Le second type d'arbre est construit selon la distribution des routeurs intermédiaires. Les routeurs construisent un arbre unique commun à l'ensemble des membres du groupe. Dans ce



type d'arbre, le trafic multicast est transmis et reçu à travers le même chemin indépendamment de la source des données.

III.1. Les approches de routage

Les protocoles de routage multicast facilitent l'échange d'informations entre les différents routeurs. Ils permettent la construction des arbres de diffusion et la transmission des paquets multicast. Il existe un certain nombre de protocoles de routage multicast, mais ils suivent généralement un des deux modes de base : le mode dense et le mode épars.

III.1.1 Les protocoles en mode dense

Les protocoles en mode dense sont fondés sur l'hypothèse qu'un certain nombre des membres d'un groupe sont distribués en masse à travers le réseau. C'est pour cette raison que ces protocoles inondent périodiquement le réseau avec des messages de contrôle multicast pour établir et mettre à jour l'arbre de diffusion. Les protocoles en mode dense sont plus adaptés aux environnements où il y a un nombre élevé de récepteurs voulant ou devant recevoir les paquets multicast.

A. Le protocole DVMRP "Distance Vector multicast Routing Protocol"

Il y a un certain nombre de protocoles en mode dense pour le IPv4, mais le plus vieux et le plus largement connu est DVMRP [27]. DVMRP utilise l'inondation inversée pour la recherche des chemins de l'arbre de diffusion. Ainsi, quand un routeur reçoit un paquet, il inonde toutes les sorties exceptée celle qui mène vers la source du paquet de données reçu afin d'atteindre tout le sous-réseau. Si un sous-réseau ne veut pas recevoir les paquets d'un groupe particulier, le routeur concerné envoie un message de résiliation vers son routeur ascendant dans l'arbre de diffusion. Cela permet d'arrêter le transfert des paquets vers une branche où il n'y a aucun membre concerné. DVMRP a son propre protocole de routage unicast, basé sur le comptage des sauts. Il permet de déterminer quelle interface mène de nouveau au point d'émission des données, ainsi les chemins de transfert des paquets multicast et unicast peuvent ne pas être identiques.

B. Le protocole PIM-DM "Independent multicast Dense Mode"

PIM-DM [29] est un protocole de routage qui s'adapte à la fois à l'IPv4 et à l'IPv6. PIM-DM fonctionne d'une manière semblable à DVMRP. Cependant, il est plus adapté aux cas où les membres d'un groupe multicast sont organisés en masse localisée. Comme pour DVMRP, PIM-DM inonde le réseau avec les paquets multicast pour éliminer par la suite les routeurs



ayant transmis un message de résiliation. La notion de protocole indépendant signifie que le protocole peut utiliser les tables unicast existantes au niveau des routeurs sans procéder à une reconstruction des tables de routage spécifiques au transfert multicast. Le remplissage de la table de routage peut se faire par n'importe quel protocole de routage unicast.

III.1.2 Les protocoles en mode épars

Les protocoles en mode épars sont fondés sur l'hypothèse que les membres d'un même groupe multicast sont abondamment distribués à travers le réseau et que la bande passante n'est pas nécessairement disponible. Du fait que les membres d'un groupe sont éparpillés dans tout le réseau, l'inondation gaspillerait la bande passante inutilement et pourrait causer des problèmes de performance. Les protocoles en mode épars sont donc plus sélectifs en ce qui concerne la façon dont ils distribuent les paquets de données multicast. Ils débutent avec des arbres de diffusion vides et ajoutent uniquement les branchements par où ils reçoivent des messages de demande d'adhésion. Les protocoles CBT "Core Based Trees" [30] et PIM-SM "Protocol Independent Multicast-Sparse Mode" [31] sont les deux protocoles les plus communs pour le mode épars.

** Le protocole PIM-SM "Independent multicast Sparse Mode"*

PIM-SM est un protocole de routage qui s'adapte à la fois à l'IPv4 et à l'IPv6 comme c'est le cas pour le protocole PIM-DM. PIM-SM a été conçu pour des environnements où les récepteurs sont largement distribués. PIM-SM utilise un point de rendez-vous où les différentes sources transmettent leurs données et d'où les récepteurs du réseau réclament les paquets qu'ils désirent. Ainsi, quand un récepteur veut recevoir un flux de données multicast, il s'inscrit au niveau du point de rendez-vous. A la réception des paquets de données venant de la source, le point de rendez-vous transmet les données aux récepteurs inscrits. Les routeurs automatiquement optimisent les chemins de transfert afin d'éliminer les sauts inutiles dans le réseau. Puisque PIM-SM est un protocole indépendant, il utilise les données contenues dans les tables de routage construites par le protocole de routage unicast existant.

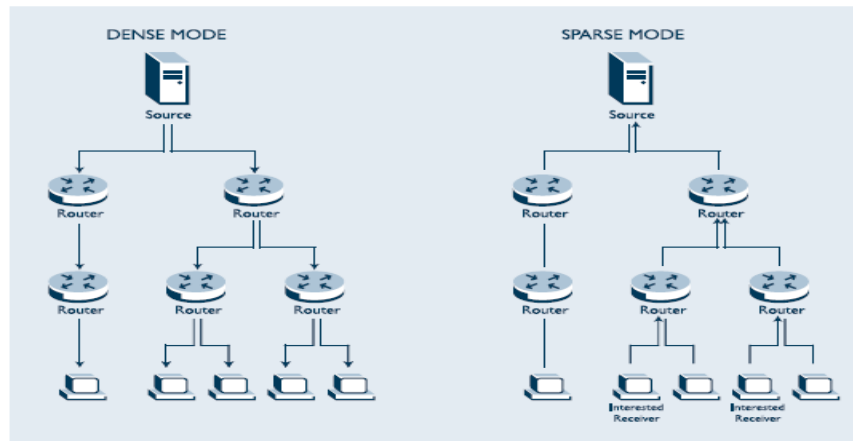
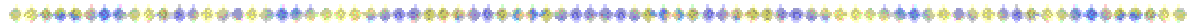


Fig.1.5. Mode dense vs Mode épars

Avec le routage en mode dense, la source diffuse les données dans tout le réseau et ne se rétracte que quand les données sont signalées comme indésirable. Avec le routage en mode épars, le récepteur intéressé demande les données de la source au premier routeur qui reçoit les paquets multicast.

IV. Les routeurs actifs

Commençons du commencement ; un routeur traditionnel (ou passif) [32] est un routeur qui possède un nombre restreint et fixe de services implantés à son niveau et qui n'offre aucun moyen d'en ajouter d'autres. Bien que de nombreux progrès aient été réalisés ces dernières années pour rendre plus souple sa configuration, son comportement reste relativement figé et spécifique à un type d'application. Des solutions à ce problème apparurent véritablement à partir de 1995 avec l'apparition des réseaux programmables. Un routeur programmable [21] est un routeur ouvert disposé à la mise en œuvre rapide de nouveaux services à son niveau. Contrairement aux réseaux traditionnels, un réseau programmable est un réseau de transport de données étendu par un environnement de programmation à l'échelle du réseau, comportant un modèle de programmation des services, des mécanismes de déploiements et un environnement d'exécution (EE). En 1996, sur une proposition de David Tennenhouse et David Wethrall est apparu le concept de réseau actif. Un routeur actif [21.32.33] est un routeur programmable dynamiquement par des entités tierces (opérateurs, fournisseurs de services, applications, usagers). Cette approche permet d'étendre le concept de programmation en ouvrant les équipements de l'ensemble du réseau et en partant du principe que tout usager peut concevoir et déployer, à l'aide d'interface de programmation (API), de nouveaux services. Ces nouveaux services peuvent alors être utilisés de manière dynamique dans le réseau.



IV.1. L'intérêt des routeurs actifs

Grâce aux routeurs actifs, les réseaux de transmission de données sont de plus en plus flexibles et dynamiques. Ils s'adaptent à chaque type d'application les utilisant et à chaque paquet qu'ils acheminent. Ainsi, les services actifs d'un routeur proposent trois types de fonctionnalités:

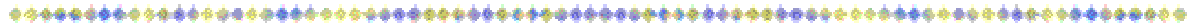
- Des fonctions d'accès à l'environnement du nœud pour la récupération de l'adresse IP, de la date, du MTU,.....
- Des fonctions de manipulation de paquets pour modifier les données d'un paquet, l'adresse source, l'adresse destination, la longueur, le protocole encapsulé et le TTL.
- Des fonctions de contrôle pour l'envoi de paquets IP et UDP, l'abandon de paquets, la réponse à un paquet et l'évaluation d'un paquet.

Ces fonctionnalités font que les réseaux actifs ont l'avantage de :

- Réduire le temps nécessaire au développement et au déploiement des nouveaux services réseau à mettre en place.
- Permettre à des usagers ou des parties tierces de créer ou d'adapter les services d'un composant du réseau selon les particularités de leurs applications ou en fonction de l'état courant du réseau.
- Fournir aux chercheurs une plateforme d'expérimentation grâce à leur aspect dynamique et programmable. Les services réseaux expérimentaux peuvent être injectés dans un environnement réel sans pour autant perturber le fonctionnement du réseau actif.
- Offrir des améliorations prometteuses dans le domaine des protocoles traitant la fiabilité en multicast car ils font appel à un point clé de ce genre de protocole qui est la possibilité de prendre une décision au niveau même du réseau.

IV.2. La mise en œuvre d'un réseau actif

Les réseaux actifs sont dynamiques dans le sens où le comportement de leurs composants (source, routeurs et récepteurs) s'adapte aux différents états du réseau. Pour la mise en pratique de tels réseaux, il est nécessaire de déterminer un modèle propre au fonctionnement des nœuds actifs mais aussi de définir les différentes façons possibles pour rendre les composants du réseau dynamiques et adaptatifs.



IV.2.1 Le model fonctionnel des réseaux actifs

Le model fonctionnel présenté dans cette section est extrait d'une publication de M. I-Hsuan Huang de l'université de Yuan Ze en Chine [19]. Il propose un model fonctionnel basé sur quatre couches. Chaque couche interagit avec ses couches adjacentes grâce à des interfaces comme c'est le cas pour le model OSI. (Fig.1.6)

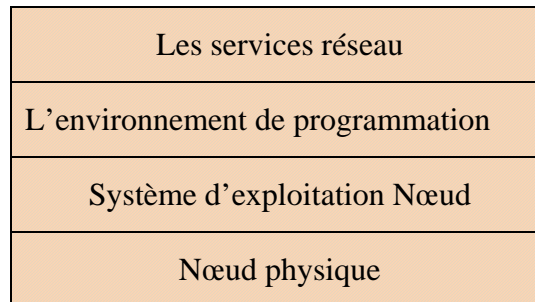


Fig.1.6 Le model fonctionnel des réseaux actifs

La couche la plus basse représente la structure physique du nœud actif du réseau. La deuxième couche représente le système d'exploitation qui gère le nœud actif. Cette couche définit les paramètres de sécurité tels que l'authentification et la vérification des données. Elle est responsable, aussi, de la gestion des ressources propres aux nœuds tels que la gestion du processeur et la réservation de bande passante dans le réseau. La couche qui suit supporte l'environnement de programmation. Cette couche procure une plateforme de programmation supplémentaire. Elle aide à la construction de nouveaux protocoles et applications au niveau du nœud actif. La couche la plus haute représente les services réseaux désirés. Ces services sont le plus souvent liés aux protocoles et applications réseaux : le protocole du Ping, le protocole du Trace Route, les protocoles de contrôle de congestion, les protocoles de transmissions multicast et des applications multimédia ou encore les protocoles de mise en cache des données. En plus de ces couches, il existe une interface entre les couches adjacentes. Elles permettent la communication entre les différentes couches du modèle. L'architecture fonctionnelle d'un nœud actif est représentée par la figure 1.7.

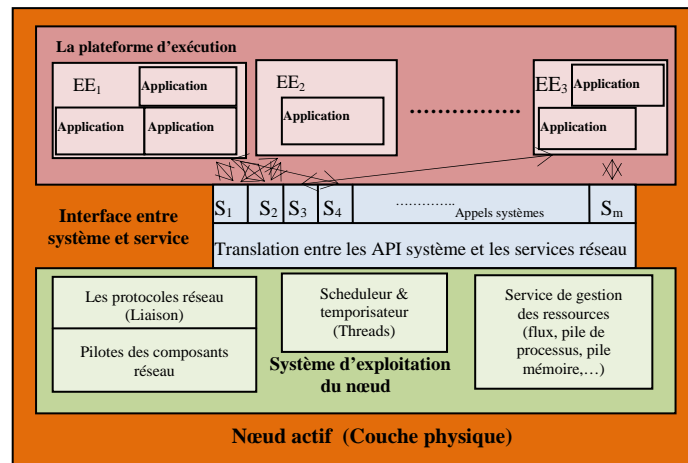


Fig1.7. Architecture fonctionnelle d'un nœud actif

IV.2.1 Les types de réseau actif

Dans la famille des réseaux actifs, on trouve deux approches pour la mise à jour régulières des équipements actifs [33, 34] :

A. L'approche intégrée

Elle permet de transporter le code des services réseau dans le même flux que les données traitées par ce service. Classiquement, le code du service est émis avant l'arrivée du flux de données. Dans le cas extrême, chaque paquet de données contient le code du service à y appliquer. On utilise le terme de paquet actif pour identifier un paquet qui encapsule à la fois des données et du code de programmation. Généralement la vie du service est liée à celle du trafic affecté.

✦ Les Avantages :

- Ouverte et dynamique.
- Flexible.
- Nécessite peu de traitement.

✦ Les inconvénients :

- Taille restreinte.
- Usage limité à des situations précises.
- Possibilité de perte.

Le PLAN [39] est un langage de programmation fonctionnelle pour des composants à forte contraintes de ressource. Il emploie une forme d'appel de procédures distantes pour procéder à la programmation des paquets actifs du réseau. Ce langage fait partie du projet "SwitchWare". Le projet "SwitchWare" décrit les efforts de recherche en cours dans le domaine des réseaux actifs au niveau du département d'informatique et des sciences de l'information de Penn et Bellcore.



B. *L'approche discrète*

Dans cette approche les services sont déployés dynamiquement sur les nœuds du réseau mais hors du flux de données utilisateurs. Cela se rapproche fortement du concept des réseaux programmables. La différence avec ces derniers, repose sur le fait que dans cette approche les composants d'un service sont déployés physiquement sur les nœuds du réseau, le plus souvent grâce à des techniques de code mobile. On appelle ce type de nœud, des nœuds actifs.

✦ *Les Avantages :*

- Fiable.
- Usage général.
- Taille du code important.

✦ *Les inconvénients :*

- Fixe.
- Nécessite des traitements conditionnels.
- Présente des défaillances de la sécurité.

Parmi les applications et projets réalisés grâce à cette approche, le projet ANN [38] (Active Network Node) représente la conception d'un nœud actif flexible qui permet d'atteindre de hautes performances en supportant des transferts de données en gigabit. Ce nœud fournit en plus de la flexibilité typique à la technologie des réseaux actifs, le déploiement des protocoles automatiquement, la rapidité du traitement des informations spécifiques aux applications et leur expédition.

Pour une question de capacité en bande passante, de fiabilité et d'efficacité, l'approche la plus utilisée est l'approche discrète.

Remarque :

Il existe des projets de recherche qui explorent la possibilité d'exploiter les avantages de chacune des deux approches. Le projet DAN "Distributed Code Caching for Active Networks" [37] propose une nouvelle architecture basée sur la combinaison de l'approche intégrée et distribuée et cela en transmettant dans les paquets de données (à la place du code de programmation) un pointeur vers des modules digitaux préalablement chargés au niveau des routeurs actifs. Ainsi, les avantages des deux approches sont réunis dans une même architecture.



VI.3 Les applications des réseaux actifs

Les protocoles appliqués aux réseaux actifs peuvent être de différents types :

1. *Les protocoles de filtrage* : ce sont les protocoles chargés de filtrer le flux de donnée en éliminant les paquets superflus afin d'éviter la congestion du réseau ou encore pour assurer une certaine sécurité comme c'est le cas pour les applications Pare-feux.
2. *Les protocoles d'encodage* : ce sont les protocoles chargés de rassembler des paquets de données en un seul paquet. Cela permet d'éviter les problèmes de congestion lors du passage des paquets d'un lien à haut débit vers un lien à faible débit.
3. *Les protocoles de codage/décodage de données.*
4. *Les protocoles de sécurité.*
5. *Les protocoles de gestion du réseau.*
6. *Les protocoles de routage.*
7. *Les protocoles de sauvegarde* : ce sont les protocoles dédiés à la sauvegarde des données. La sauvegarde peut se faire soit selon le contenu des paquets de données soit à la volée. On parle de sauvegarde active. Les proxys Web sont des exemples de ces applications.
8. *Les protocoles de déploiement* : ce sont les protocoles chargés de déployer les services réseau dans l'ensemble du réseau.
9. *Les protocoles d'usage divers* : ce sont tous les autres protocoles qui se basent sur les réseaux actifs pour accomplir une tâche bien définie au niveau du réseau.

V. Conclusion

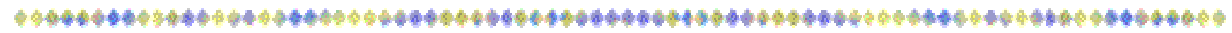
Ce chapitre est un résumé de tout ce qui est nécessaire à savoir pour mieux cerner le but du travail exposé dans ce mémoire. On a présenté très brièvement la transmission multicast et son utilité pour les applications de dissémination de données et coopératives. On a mis en évidence la nécessité des routeurs actifs et l'apport de ces derniers dans la transmission fiable des données de ces applications. Par conséquent, tout au long de ce mémoire, on s'intéressera plus particulièrement aux protocoles multicast fiables basés sur les réseaux actifs, vu les nombreux avantages que présente ce concept révolutionnaire.

Chapitre II



La fiabilité multicast dans les réseaux best effort

*“Au temps des temps où la sagesse étant reine,
la science était seine ”*



I. Introduction

Les applications multicast ont généralement des exigences différentes. Certaines applications exigent un délai d'acheminement des données très court tout en tolérant quelques pertes de données. D'autres applications ont des exigences inverses.

Le multicast IP ne garantit qu'une délivrance "best effort" et ne fournit aucun contrôle du débit de transmission, ce qui engendre deux types de problèmes :

- Parmi les paquets émis par la source certains peuvent éventuellement se perdre en cours de transmission.
- Le non contrôle du débit de transmission peut induire une surcharge au niveau des composants du réseau, provoquant ainsi une congestion et une occupation inutile de la bande passante.

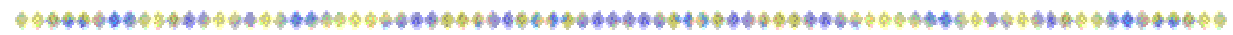
Une des solutions proposées est la participation des routeurs actifs dans le processus de recouvrement pour ce qu'ils présentent comme potentialités. Le support des services actifs permet de réduire les failles des approches de recouvrements antérieurs.

Ce chapitre est consacré aux différentes approches de recouvrement adoptées ou exploitées par les protocoles multicast fiables existants. On donnera un résumé de chaque approche de recouvrement qui a précédé l'introduction des services actifs. On présentera aussi les différentes architectures réseau élaborées afin d'acheminer efficacement les données multicast de la source à l'ensemble des récepteurs. On conclura cette section en exposant l'apport des services actifs dans l'élaboration des protocoles de recouvrement multicast.

II. Les applications du multicast

La transmission multicast s'est faite une place parmi les transmissions adoptées dans le réseau Internet. Cela ne pouvait être autrement vu les atouts de ce mode de transmission. Le multicast permet d'optimiser les performances du réseau : un seul flux est en sortie lors d'une transmission ainsi la bande passante du réseau est préservée et le transfert des paquets redondants est réduit.

Ainsi, la transmission multicast fournit une efficacité évidente en contrôlant le trafic du réseau et en réduisant la charge des différents dispositifs du réseau. De plus, ce mode de transmission s'adapte à tout type de réseau quel que soit son étendu. Mais pour une meilleure performance, l'utilisation multicast est plus réponde au niveau des réseaux étendus "Wide Area Network" (WAN). Le multicast permet à différents clients dispersés un peu partout dans le monde de



recevoir les mêmes données simultanément sans surcharger le réseau. Ainsi, il a facilité l'extension des applications qui nécessitent la transmission de données à un ensemble prédéfini de clients avec des délais d'acheminement restreints.

Les applications multicast peuvent être classées en deux classes selon l'importance des données transmises [10] :

- Les applications à fiabilité complète : Pour ces applications, chaque paquet transmis doit être impérativement bien reçu par l'ensemble des clients de destination. Parmi ces applications, on trouve les simulations interactives, les logiciels de mise à jour, les tableaux blancs, les jeux multi-joueurs, etc.
- Les applications à fiabilité partielle : Ce sont les applications dont la perte d'un paquet lors de la transmission n'affecte pas son exécution. Comme exemple, on a les vidéos conférences, les conversations audio, les services vidéo à la demande (video-on-demand), etc.

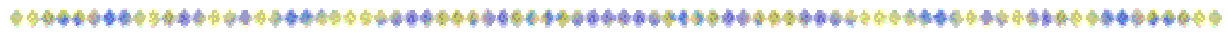
Un autre classement des applications se fait selon le délai de transmission exigé. On distingue dans ce sens deux autres classes :

- Les applications temps réel : Pour ces applications, la contrainte du délai d'acheminement est très importante. Les paquets en retard seront considérés comme perdus et inutiles. Ce sont des applications temps réel. Parmi ces applications : les applications audio et vidéo, les applications de télésurveillance, etc.
- Les applications sans contrainte temporelle : Ce sont toutes les applications multicast dont le déséquencement des paquets reçus n'affecte pas leur exécution.

Le multicast a aidé au développement d'un bon nombre de concepts informatiques dont l'utilisation, de nos jours, est de plus en plus demandée. Citons :

- La mise à jour des bases de données distribuées.
- La communication interprocessus dans les systèmes coopératifs.

Le succès de ces applications dépend de l'efficacité de la transmission multicast utilisée. Aussi, il est nécessaire de définir des protocoles multicast dignes des attentes exigées par ces différentes applications, à savoir, la fiabilité et la rapidité d'acheminement.



III. La fiabilité de transmission des flux multicast

La retransmission des données perdues a été la première solution pour les protocoles multicast fiables. Hélas, la mise en œuvre de ces retransmissions a engendré d'autres problèmes :

- La surcharge de la source et des liaisons rattachées à elle par les paquets retransmis, ce qui cause une dégradation significative du débit de transmission.
- Malgré la perte effective de paquets de données par un sous-ensemble de récepteurs, la retransmission de ce paquet par la source est expédiée vers tout le groupe multicast. Ce problème est désigné sous le nom de "Crying baby".

Une solution à ces problèmes consiste à concevoir un protocole de transport multicast au-dessus du multicast IP qui a pour but d'offrir un service fiable et de préserver les performances qu'offre la transmission multicast. Les protocoles multicast fiables (*Reliable Multicast*, RM) sont nés.

On distingue différentes classes de protocoles multicast fiables. Chacune des classes des protocoles multicast fiables existantes est venue combler les lacunes des classes précédentes.

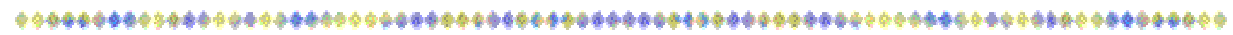
IV. Le recouvrement de bout-en-bout

Cette classe de recouvrement multicast donne l'exclusivité à la source pour réparer toutes les pertes de paquets de données survenues au niveau de l'ensemble du réseau. La signalisation des pertes se fait soit à l'initiative de la source soit à l'initiative des récepteurs finaux [11,12].

IV.1 Le recouvrement basé sur l'initiative de la source

Les protocoles multicast fiables basés sur l'initiative de la source nécessitent que la source soit capable de recevoir des acquittements positifs (ACK) venant de la totalité des récepteurs avant de mettre à jour sa fenêtre d'émission. La source ne supprime de sa mémoire que les données associées au ACK's reçus. L'approche traditionnelle des protocoles basés sur l'initiative de la source, charge la source de détecter les erreurs survenues lors de la transmission en définissant, à son niveau, un délai de retransmission. Ce délai de retransmission a pour but de cadencer la transmission fiable des données ainsi que la détection des erreurs éventuelles.

Un ACK peut référencier un paquet particulier comme il peut référencier un groupe de paquets, tout dépend de la stratégie utilisée.



La source préserve en mémoire le paquet envoyé jusqu'à ce que la totalité des récepteurs l'acquiesce positivement. Il est clair que la liste de l'ensemble des récepteurs doit être connue par la source.

Dans cette approche, on dit que la source « *pousse* » les récepteurs à acquiescer les paquets reçus en retransmettant les paquets non acquiescés dès l'expiration du délai d'attente.

Cette technique de recouvrement a une limitation majeure. Ce n'est pas l'utilisation des ACK's en lui-même qui pose problème mais cette façon de faire nécessite de la source le traitement de la totalité des ACK's du réseau. La source, aussi, doit à tout moment connaître l'ensemble des récepteurs concernés par la transmission. Cette limitation engendre des problèmes cruciaux :

- Le problème de l'implosion des acquiescements positifs dans le réseau induisant la congestion du réseau.
- Très mal adapté aux réseaux de grande envergure.

Pour éviter ces problèmes deux solutions sont proposées :

- L'utilisation des acquiescements négatifs à la place des acquiescements positifs.
- Déléguer la responsabilité du recouvrement à un ensemble de récepteurs et cela en organisant le réseau en une architecture prédéfinie.

Ces deux solutions sont décrites plus loin.

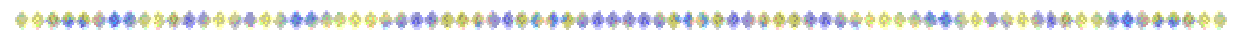
IV.2. Le recouvrement basé sur l'initiative des récepteurs

Cette technique charge les récepteurs de la détection des erreurs survenues lors de la transmission des paquets.

L'un des aspects importants de cette technique est l'absence des acquiescements positifs. Ils sont remplacés par des acquiescements négatifs transmis par les récepteurs dans les deux cas suivants :

- La réception d'un paquet ayant un numéro de séquence supérieur au numéro de séquence attendu. Cela implique une perte des paquets précédents.
- L'expiration du délai d'attente d'un paquet de données attendu par le récepteur.

Dans cette approche, la source ne reçoit de réponse des récepteurs que dans le cas d'une perte. Aussi, on dit que les récepteurs « *poussent* » la source à retransmettre en envoyant des acquiescements négatifs dès la détection d'une perte de données.



On peut remarquer que pour cette technique de recouvrement, la source est incapable de déterminer à quel moment elle doit éliminer de sa fenêtre d'émission les paquets envoyés. Les différents protocoles existants ne définissent pas explicitement le mécanisme utilisé pour la mise à jour de la fenêtre d'émission de leur source. Cependant, l'utilisation de cette technique a trois avantages essentiels par rapport à l'approche basée sur l'initiative de la source :

- La source ne garde pas la liste des récepteurs concernés par la transmission.
- La source n'a pas à traiter les ACK's transmis par l'ensemble des récepteurs.
- Le problème de l'implosion des paquets de contrôle ne se pose que lorsqu'un grand nombre de récepteurs perdent un paquet simultanément.

Cette technique présente des inconvénients malgré qu'elle permette de diminuer le nombre de paquets de contrôle circulant dans le réseau :

- La détection des pertes au niveau des récepteurs est retardée soit jusqu'à l'arrivée du prochain paquet ou jusqu'à expiration du délai d'attente. Cette technique est inadéquate aux applications temps réel où la contrainte du temps est très importante.
- Le mécanisme de mise à jour de la fenêtre d'émission de la source n'est pas déterminé. Cela implique une mauvaise gestion de l'espace mémoire de la source.

Pour combler les lacunes de cette technique ou plus au moins réduire leurs effets sur les performances du réseau, il est nécessaire de définir une architecture réseau adéquate et d'adopter un mécanisme efficace pour la suppression des paquets transmis au niveau de la source.

D'une manière générale, les protocoles se basant sur un recouvrement de bout en bout ont des performances limitées pour des réseaux à grande échelle. Cette limitation est due au débordement de la source. La source doit traiter un nombre croissant d'ACK/NACK. Très vite le problème de l'implosion des acquittements réduit considérablement les performances du réseau.

V. Le recouvrement local

Cette classe de protocole de fiabilité multicast est apparue comme une solution à l'implosion des acquittements. Le principe est de localiser le recouvrement des pertes de données survenues dans le réseau et ainsi réduire la charge de la source. Une solution été de désigner des dispositifs supplémentaire dédiés au recouvrement des pertes au niveau de leur localité. Des protocoles comme RMTP "Reliable Multicast Transport Protocol" [13] se base sur la

transmission d'acquittements positifs pour détecter les pertes de données au niveau des récepteurs. D'autres protocoles se basent sur la réception d'acquittements négatifs pour procéder au recouvrement comme c'est le cas pour le protocole PGM "Pragmatic General Multicast" [14].

Une autre solution a émergé constituant une nouvelle approche dans la conception des réseaux informatiques. Le concept des services actifs très approuvé par la communauté scientifique. Ces services actifs peuvent être implémentés au niveau des routeurs, on parle alors de routeurs actifs. Cette notion signifie que les routeurs en plus de leur tâche initiale à savoir le routage des données, participent au recouvrement des pertes survenues lors de la transmission des données dans le réseau. Citons comme exemple le protocole MAF "Multicast Actif Fiable" [15]

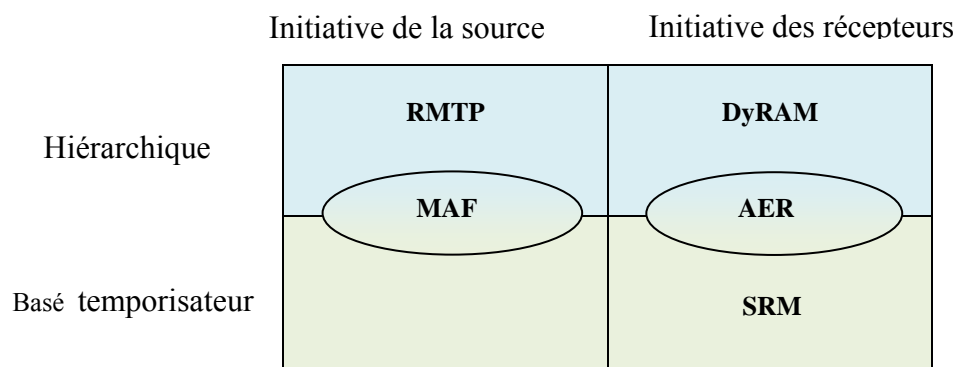
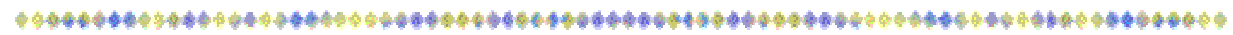


Fig.2.1 Classification des protocoles multicast fiables

Pour améliorer le recouvrement local des pertes, des techniques et stratégies nouvelles sont introduites. Entre autre, la nécessité d'organiser le réseau selon une architecture hiérarchique et d'utiliser des temporisateurs pour cadencer le recouvrement [16,17]. Mais aussi, le recourt aux codes détecteur/correcteur d'erreur tels que le code FEC ou l'introduction des services actifs au niveau des routeurs du réseau.

V.1. L'architecture en arbre

L'architecture en arbre ou hiérarchique du réseau consiste en l'organisation du réseau selon un arbre de transmission multicast où la source est la racine et les récepteurs sont les feuilles. Les récepteurs sont divisés en groupes multicast. Pour chaque groupe un nœud intermédiaire permet de faire la liaison entre les récepteurs du groupe et la source. Les nœuds intermédiaires peuvent être eux même des nœuds appartenant à un groupe multicast du réseau. Chaque groupe peut avoir plus d'un nœud intermédiaire et cela dans le cas des réseaux à source



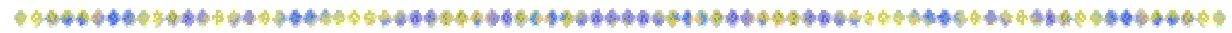
multiples. Les récepteurs qui ne sont pas des nœuds intermédiaires forment généralement la fin de l'arbre et ils sont appelés « les feuilles de l'arbre ».

Cette structure résulte du découpage en sous-arbres de profondeur 1, de l'arbre de diffusion. La source ainsi que chaque routeur actif de l'arbre de diffusion gèrent des sous-arbres comptant au plus D feuilles dont ils connaissent l'identité. Nous dirons de ce fait que la source et les routeurs actifs se répartissent la connaissance du groupe multicast destinataire. Cette limite D est définie à la création de l'arbre de diffusion et permet de réduire la charge de traitement de la source et des routeurs actifs. A la racine des sous-arbres ainsi construits, se trouve la source ou un routeur actif, alors que leurs feuilles peuvent être aussi bien des récepteurs que des routeurs actifs, eux mêmes racines de sous-arbres de niveau inférieur. L'ensemble des feuilles d'un même sous-arbre constitue ce que nous appellerons un sous-groupe dont est responsable la racine du sous-arbre. On suppose, aussi, que dans cette architecture les récepteurs ne sont pas en mesure de transmettre en mode multicast pour les trois raisons suivantes :

- Dans le cas du IP multicast commercialisé et disponible actuellement [24], les sources de transmission multicast sont excessivement chère à mettre en place.
- La plus part des réseaux nécessitent de la transmission multicast à sens unique.
- L'élaboration d'un arbre de diffusion multicast pour un nombre important d'expéditeurs multicast revient trop chère.

Cette organisation permet aux récepteurs de communiquer avec la source indirectement à travers des nœuds intermédiaires. Cela permet, aussi, la flexibilité du réseau et lui permet de s'étendre à un plus grand nombre de récepteurs. Les récepteurs appartenant à un groupe envoient aux nœuds intermédiaires les paquets destinés à la source. Ces transmissions sont dites locales (ACK local/NACK local).

L'architecture en arbre a l'avantage d'éviter le problème d'implosion due à la transmission des ACK/NACK cela grâce à l'agrégation des paquets au niveau des nœuds intermédiaires. Aussi, des analyses ont démontrées qu'en utilisant des schémas hiérarchiques de recouvrement, on peut avoir un débit maximal constant quelque soit le nombre de récepteurs et permet aussi un recouvrement local des pertes [18].



V.2. Les temporisateurs

Pour les approches qui se basent sur les temporisateurs, elles utilisent un mécanisme de temporisation pour élire un récepteur demandeur de la retransmission d'un paquet perdu. Il est de même pour l'élection d'un répondeur. Un délai d'attente est déclenché pour différencier les récepteurs ayant reçus correctement les paquets perdus. Un de ces récepteurs est élu afin de répondre à cette demande de réparation émise par les autres récepteurs.

V.3. Les codes détecteurs/correcteurs d'erreurs

Pour améliorer la fiabilité des transmissions multicast et minimiser le nombre de paquet de contrôle transmis en cas de perte, un code détecteur/correcteur d'erreur est introduit au niveau des paquets. La façon d'encoder et de décoder les paquets de données dépend du code utilisé. Aussi, les codes peuvent être transmis encapsulé dans le paquet de données ou encodé directement avec les données. D'une manière générale, cette technique consiste à générer un code à partir des données à transmettre divisées préalablement en un nombre fixé de petit paquet. Chaque code généré est transmis avec le paquet de données correspondant. A la réception, si une perte est détectée grâce au décodage du paquet alors une correction est possible grâce au code transmis. Parmi les codes les plus utilisés dans la fiabilité des transmissions multicast, on trouve le code FEC [35, 36]

V.4. Les services actifs

L'introduction des routeurs actifs a révolutionné le monde de la fiabilité multicast. L'idée consiste en l'utilisation des fonctionnalités fournies par les routeurs actifs dans l'élaboration des protocoles multicast fiables. Divers travaux ont été entrepris sur l'étude des protocoles de la fiabilité multicast dans les réseaux actifs. L'introduction du concept des réseaux actifs a permis de résoudre de façon élégante les problèmes dus aux retransmissions et au fait que les groupes multicast sont fortement dynamiques.

Parmi les solutions on trouve l'ARM (Active Reliable Multicast). L'ARM sous-ensemble des protocoles RM, il se base sur les nœuds actifs qui jouent un rôle primordial dans le mécanisme de fiabilité.



VI. Conclusion

Pour aboutir à une transmission fiable à grande échelle, il est bien clair que les chercheurs ont mis en œuvre tous leurs efforts et volonté pour aboutir à des résultats concluants. Cependant, jusqu'à présent, il semble que les meilleurs résultats ont été obtenus avec l'introduction des services actifs. Ainsi, le chapitre qui suit sera dédié à l'état de l'art des protocoles de fiabilité multicast dans les réseaux actifs.

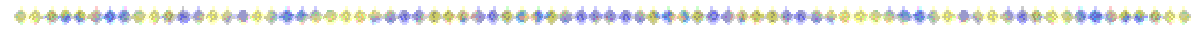
Chapitre III



Les protocoles de fiabilité multicast dans les réseaux actifs

*"Si j'ai vu plus loin que les autres, c'est par ce que je me suis
placé sur les épaules de géants."*

- Issac NEWTON -



I. Introduction

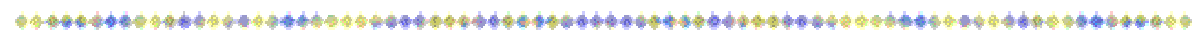
Avec l'apparition du paradigme des Réseaux Actifs, plusieurs protocoles de Multicast fiable, basés sur le soutien des routeurs, ont été proposés [13, 14, 15, 17, 23, 24, 25]. L'objectif de ces protocoles, que nous qualifierons d'actifs, est d'offrir une solution satisfaisante du point de vue fiabilité de transmission et résistance au facteur d'échelle. Cependant, chacun d'eux adopte une stratégie différente pour le problème de l'implosion des paquets de contrôle.

Selon la conception des protocoles, les routeurs actifs sont capables d'effectuer les actions suivantes :

- Le cache des paquets de données pour assurer le recouvrement local. Cela permet, d'une part, la répartition de la charge de recouvrement entre la source et les routeurs actifs et d'autre part une réduction considérable de la latence de recouvrement.
- L'agrégation et/ou la suppression locale des paquets de contrôles pour éviter le problème de l'implosion des paquets de contrôles au niveau de la source.
- La retransmission en limitant l'envoi aux récepteurs ayant effectivement perdu des paquets. On parle de retransmission en Subcast.
- L'élection d'un répondeur parmi les récepteurs ayant reçu le paquet perdu. Cela permet de réduire la charge des routeurs actifs en termes de cache mémoire.

L'ensemble des routeurs ainsi modifiés, qu'on appellera routeurs de réparation, est organisé selon une structure propre à chaque protocole. Si ces protocoles minimisent les risques d'implosion et dispensent la source et les routeurs de connaître l'ensemble de leurs fils, ils requièrent néanmoins, pour garantir une fiabilité totale, des capacités de stockage infinies au niveau de la source, voire même des routeurs de réparation. Aussi, chaque protocole a mis en place une technique de gestion du cache au niveau de la source et des routeurs de réparation.

Cette section est consacrée à l'état de l'art des protocoles de fiabilité multicast dans les réseaux actifs. On présentera les trois protocoles les plus récemment trouvés dans la littérature. Les protocoles MAF "Multicast Actif Fiable" [15], DyRAM "Dynamic Replier Active reliable Multicast" [23], AER "Active Error Recovery" [24] seront présentés brièvement tout le long de ce chapitre. On conclura cette section en donnant un tableau récapitulatif des différences qui existent entre les différents protocoles cités dans ce chapitre.



II. Le protocole “Active Error Recovery”: AER

II.1. Vue d’ensemble du protocole AER


Le protocole AER [24] est un protocole de transport développé afin d’assurer la fiabilité des transmissions Multicast IP. Ce protocole repose sur une architecture en arbre. Dans la hiérarchie que propose AER, les services actifs qu’implique l’utilisation des réseaux programmables 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 diffusion et ils représentent les seules entités actives du réseau. Les serveurs de réparation sont chargés de garder une copie des données qui transitent à travers eux pour par la suite les retransmettre en cas de perte de données. Aussi, les serveurs de réparation de ce protocole sont définis de façon à minimiser le nombre de paquets de contrôle circulant dans l’arbre de diffusion et cela en éliminant les acquittements redondants qui le traverse. Afin que les serveurs de réparations jouent leur rôle dans le réseau, il est nécessaire d’activer leurs services actifs. Dans ce cas, ils se joignent au groupe multicast pour le quel les paquets de données sont envoyés.

II.2. L’identification des serveurs de réparation

Le protocole AER met en place un mécanisme de signalisation qui permet, non seulement, d’identifier les serveurs de réparation au sein du réseau mais aussi d’activer/désactiver les services actifs des serveurs et la prise en charge des routes asymétrique et le changement de routes. Ce mécanisme est représenté par la transmission périodique de messages SPM (Source Path Messages) "Messages de chemin source". La source transmet périodiquement des SPM 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 l’adresse de la source. Grace à ce mécanisme, les récepteurs, lors de la réception d’un message SPM, pourront identifier le serveur de réparation le plus proche.

II.3. La technique de recouvrement des erreurs

Le protocole AER adopte une technique de recouvrement basée sur l’émission d’acquittements négatifs. Les récepteurs concernés par une quelconque perte de données les transmet au serveur de réparation le plus proche. Cependant, AER enrichie sa technique de recouvrement en introduisant un système de temporisation au niveau des récepteurs et des serveurs de réparation intermédiaires. A la détection d’une perte, les récepteurs (résp. serveur



de réparation) déclenchent un temporisateur. La durée du temporisateur est aléatoire afin d'éviter toute confusion entre les récepteurs ayant détecté la même perte simultanément. Lors de l'expiration du délai d'attente et si le récepteur n'a pas reçu de NACK identiques durant le délai de garde, alors le récepteur transmet son NACK au plus proche des serveurs de réparation ascendant et déclenche un temporisateur d'attente de réparation. Si un NACK parvient au récepteur (résp. serveur de réparation) avant l'expiration de son temporisateur, il supprime le sien et se comporte comme s'il l'a transmis.

A l'expiration du délai d'attente d'une réparation alors qu'aucune réparation n'a été reçue, le récepteur (résp. serveur de réparation) retransmet le même NACK à son plus proche serveur de réparation et réinitialise le délai d'attente d'une réparation.

Cette technique permet de diminuer le nombre de paquets de contrôle transmis par les composants du réseau victimes d'une perte de données.

Lors de la réception d'un NACK au niveau d'un récepteur (résp. serveur de réparation), un serveur de réparation peut avoir deux comportements distincts :

- Si le serveur de réparation possède le paquet de données concerné par l'acquittement reçu, il subcast¹ le paquet de données demandé.
- Si le serveur de réparation ne possède pas le paquet de données demandé, il subcast le NACK reçu à l'ensemble des récepteurs de son sous-arbre et transmet un NACK à son plus proche serveur de réparation si cela n'a pas été déjà fait.

Dans AER, si un serveur de réparation reçoit un paquet de réparation de son serveur de réparation ascendant, il garde en mémoire une copie du paquet reçu et le subcast¹ aux membres de son sous-arbre que s'il a reçu préalablement un NACK concernant ce paquet de données. Lorsque la source reçoit un NACK, elle multicast à nouveau le paquet de données à l'ensemble du groupe multicast. Elle s'appuie sur les serveurs de réparation intermédiaires pour restreindre le nombre de récepteurs recevant la retransmission du paquet de données.



(1). Subcast : est le fait de transmettre en Multicast uniquement vers les membres du sous-arbre de plus bas niveau.



II.4. La stratégie de libération du cache mémoire

Le protocole AER ne prévoit aucune technique précise pour ce qui concerne la libération du cache mémoire au niveau des serveurs de réparation.

Cependant, il est mentionné qu'un serveur de réparation ne recevant pas de message SPM² durant une certaine période de temps, procède à la désactivation de ses services actifs et à la destruction de tous les paquets de données stockés en mémoire.

III. Le protocole "Multicast Actif Fiable" : MAF

III.1. Vue d'ensemble du protocole MAF

Le protocole MAF [15] est un protocole de fiabilité multicast qui se base sur l'aspect actif des différents éléments du réseau. L'architecture sur laquelle repose ce protocole est une architecture hiérarchique à plusieurs niveaux. MAF repose sur l'existence d'un protocole de routage multidestinataire qui construit et maintient un arbre de diffusion optimal dont la racine est la source. A la racine des sous-arbres de l'arbre de diffusion se trouve la source ou un routeur actif, alors que leurs feuilles peuvent être aussi bien des récepteurs que des routeurs actifs. En effet, le protocole MAF affecte la réparation des pertes constatées au sein d'un sous-groupe, à la racine de ce sous-arbre. Nous dirons de ce fait qu'une racine de sous-arbre est le chef du sous-groupe constitué de ses feuilles. Un chef joue le rôle qu'attribuent à la source les approches de bout en bout : il assume, comme nous le verrons en détail plus tard, la détection et la réparation des pertes qui affectent le sous-groupe dont il est responsable. Pour réaliser ces fonctions, un chef doit connaître la composition du sous-groupe dont il est responsable. Aux fonctions de la source, s'y ajoute celles des récepteurs : un routeur de réparation retourne des acquittements positifs (ACK) à son propre chef, tout en étant responsable des chefs de niveau inférieur. Comme dans les approches orientées émetteur, la réception d'un ACK permet à un chef de détecter les éventuelles pertes.

De plus, la réception d'un ACK agrégé constitue une indication explicite pour la libération du cache. Le protocole MAF impose à la source de transmettre à un débit fixe maximum pour la durée d'une session de transmission donnée. Ce débit est établi lors de la création des groupes multicast.



(2). La réception d'un message SPM signifie que le serveur de réparation fait partie de l'arbre de diffusion pour la session multicast courante.



Cette limitation permet :

- La prévention d'une quelconque congestion au niveau du réseau lors de la transmission.
- La prévention de pertes liées à l'hétérogénéité des récepteurs du groupe.

III.2. La stratégie d'élection des répondeurs

En plus de l'arbre de diffusion, le protocole MAF définit une structure logique appelée arbre de contrôle. Cette structure est logique en ce sens où seuls les routeurs actifs qui séparent chacun des récepteurs de la source y sont représentés. Le protocole MAF suppose que les routes séparant les récepteurs de la source dans l'arbre de contrôle, coïncident avec les routes inverses dans l'arbre de diffusion. Cette contrainte est due aux états qu'installe un routeur actif de l'arbre de diffusion en fonction des informations de contrôle reçues des récepteurs : c'est sur la base de ces états que sont réalisées :

- La réparation des pertes,
- La réduction du nombre de récepteurs exposés à des retransmissions inutiles
- L'agrégation des informations de contrôle.

Pour que cette supposition soit toujours réalisée, le protocole met en œuvre une signalisation légère qui utilise des messages appelés (Leader Discovery Message) "Messages de découverte des chefs" ou LDM. Cette signalisation permet à chacun des récepteurs et des routeurs actifs de maintenir l'adresse du premier routeur actif situé immédiatement en amont dans l'arbre de contrôle. Les LDM sont régulièrement transmis par la source. Initialement, ces paquets comportent l'adresse de la source. A chaque passage par un routeur, l'adresse est modifiée pour contenir l'adresse du routeur. Les routeurs actifs de l'arbre de diffusion et les récepteurs du groupe de diffusion enregistrent l'adresse que véhiculent les LDM reçus.

III.3 La gestion du recouvrement des pertes

Le protocole MAF exécute au sein de chaque sous-arbre, une approche basée sur l'utilisation d'acquittements positifs (ACK) et de temporisateurs de retransmission. Un chef de sous-groupe, détecte, répare et maintient en mémoire une copie des unités de données qu'il retransmet tant qu'elles ne sont pas acquittées par l'ensemble de ses fils.

Les chefs des sous-groupes transmettent leur ACK lors de la réception correcte d'une unité de données sans attendre les acquittements de ses fils. Hors, au sein d'un même sous-groupe, les



membres retournent périodiquement des AACK's (Aggregated ACK) directement adressées à leur chef et qui concernent l'ensemble des unités de données correctement transmises depuis le dernier AACK émis.

Pour être en mesure de détecter les acquittements manquants, un chef utilise un temporisateur de retransmission et doit connaître la composition du sous-groupe dont il a la charge. Le temporisateur de retransmission représente l'intervalle de temps durant lequel les fils du routeur sont supposés avoir retourné leurs acquittements. Un chef détermine T_{ack} , la valeur de son temporisateur de retransmission, en fonction du RTT qui le sépare de son fils le plus éloigné. La retransmission des unités de données considérées comme perdues se fait uniquement à ceux de ses fils qui ne les ont pas acquittées et ce, jusqu'à ce qu'ils les aient positivement acquittées. Ce mécanisme préserve ainsi les fils qui n'ont pas enregistré la perte des retransmissions inutiles.

III.4. La gestion du cache mémoire

Un chef de sous-groupe enregistre une copie des unités de données diffusées. Il la maintient en mémoire jusqu'à ce que l'ensemble de ses fils ont aient accusé la réception. Dès lors que l'ensemble de ses fils acquitte positivement des unités de données, un chef libère les mémoires correspondantes. Il est alors en mesure de recevoir les unités de données suivantes dont il pourra fiabiliser la diffusion à l'ensemble de ses fils. Un chef indique explicitement à son supérieur le nombre d'unité de données dont il a libéré son cache, en retournant périodiquement des AACK's. Un AACK concerne l'ensemble des unités de données libéré depuis le dernier AACK émis. Les chefs déterminent la valeur de T_{aack} de telle sorte que $T_{aack} = n T_{ack}$, avec $n > 1$ et T_{aack} est l'intervalle de temps entre deux émissions de AACK.

Les AACK reflètent donc le nombre d'unités de données correctement reçues par l'ensemble des membres du sous-groupe depuis le dernier AACK émis. Ce mécanisme permet à chacun des chefs de fiabiliser la diffusion des unités de données à l'ensemble de ses fils, tout en utilisant une quantité finie de sa mémoire.

IV. Le protocole "Dynamic Relier Active reliable Multicast": DyRAM

IV.1. Vue d'ensemble du protocole DyRAM

DyRAM [23] est un protocole de fiabilité multicast basé sur une architecture en arbre. Le protocole adopte un schéma de recouvrement local basé sur la transmission d'acquittements



négatifs. Les récepteurs sont chargés de détecter les pertes de paquets de données et éventuellement de retransmettre des paquets de données perdus. En effet, ce protocole se distingue des autres protocoles de recouvrement multicast grâce à ses fonctionnalités innovatrices :

- Pour chaque paquet perdu, une élection dynamique d'un répondeur est effectuée.
- Les répondeurs peuvent être des récepteurs ayant bien reçus les paquets de données perdus.
- La retransmission des paquets perdus se fait en unicast vers les récepteurs ayant effectivement perdus les paquets demandés.
- L'émulation des acquittements positifs grâce à l'ajout de nouveaux champs au niveau de l'entête des paquets de contrôle du protocole.

Ainsi, on a fait le choix de comparer ce protocole au protocole sujet de notre étude : AMRHy³. Pour ce fait, on présentera en détaille le fonctionnement de ce protocole et on donnera l'algorithme relatif au comportement de chaque élément composant le réseau actif considéré.

IV.2. Les services actifs

IV.2.1 Suppression des NACK redondants

Le protocole DyRAM permet d'éliminer les NACK redondants transitant dans le réseau. Au niveau des routeurs actifs seul un NACK est transmis pour chaque paquet de données perdu au niveau du sous-arbre. Tous les autres NACK identiques reçus de ses liens descendants sont supprimés et durant une période de temps donnée. Aussi, les routeurs gardent la trace des liens ayant signalés une perte identique grâce à une structure logique. Cela permettra, par la suite, de procéder au recouvrement.

IV.2.2 L'adaptabilité du mode de transmission

Les routeurs actifs du protocole DyRAM sont capables de transmettre de deux manières différentes :


(3). AMRHy est un protocole de transport multicast fiable, développé par un groupe de chercheurs du laboratoire de recherche LIRE de l'université de Constantine.



- Transmission en mode Subcast : le routeur, lors de la réception d'un paquet de données original, transmet le paquet en Subcast vers tous ses liens ascendants.
- Transmission en mode Unicast : le routeur, lors de la réception d'un paquet de réparation, consulte une structure logique dédiée et ne transmet la réparation qu'aux récepteurs l'ayant réellement perdue.

IV.2.3 La sauvegarde d'états

DyRAM met en place une stratégie de sauvegarde d'états au niveau de chaque routeur actifs. Les routeurs sont chargés de garder et mettre à jour l'état de chacun de leurs liens descendants grâce à des structures logiques relatifs à chaque lien.

L'état d'un lien est représenté par trois informations :

- Le numéro de séquence du dernier paquet de données reçu dans l'ordre.
- Le numéro de séquence du dernier paquet de données reçu.
- Un poids assigné par le routeur à ce lien, indiquant sa capacité à mener vers un répondeur.

DyRAM maintient l'état de chaque lien grâce à une structure logique appelé LS (Link State) « Etat des liens ». Cette structure est mise à jour à chaque réception d'un NACK venant des liens descendants du routeur.

IV.2.4 L'émulation des acquittements positifs

Parmi les fonctionnalités de DyRAM, on trouve l'émulation des acquittements positifs⁴. Cela se fait grâce à deux champs supplémentaires 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 le paquet.

Remarque :

Le protocole DyRAM définit, aussi, un paquet de contrôle transmis dans le but d'un contrôle de congestion dans le réseau. Ce type de paquet englobe, entre autre, ses deux mêmes champs et cela pour obtenir une mise à jour plus fréquente de l'état des liens descendants au niveau des routeurs.



(4). DyRAM ne se base pas sur l'envoi explicite des acquittements positifs mais il les utilise grâce à la fonctionnalité d'émulation de ce dernier.



IV.2.5 L'élection dynamique d'un répondeur

DyRAM est le premier protocole à avoir introduit cette notion. Elle permet un recouvrement local des pertes sans surcharger les routeurs actifs. Pour DyRAM, un répondeur est un récepteur ayant la possibilité de répondre à une perte de paquet de données.

La capacité d'un répondeur est définie selon la satisfaction de l'un des critères suivant selon cet ordre :

- 1) L'élément ayant acquitté positivement⁴ la réception du paquet de donnée demandé.
- 2) L'état du lien menant à cet élément indique un poids maximum par rapport aux autres éléments candidats.

Donc, l'élection d'un répondeur parmi l'ensemble des récepteurs n'ayant pas acquitté positivement le paquet perdu se fait grâce à la valeur donnée au poids du lien qui mène à ce récepteur. Aussi, les poids sont calculés à chaque fois qu'une élection est nécessaire et les valeurs résultantes sont affectées aux liens correspondants.

Le poids W d'un lien i est calculé comme suite :

$$W_i = \frac{\min_{j \rightarrow 1, n} (RTT_{jvar} \times RTT_j)}{RTT_{ivar} \times RTT_i \times (l_{ri} - l_{oi} + 1)}$$

Où :

- RTT_{ivar} = Varie inversement à la capacité en bande passante du lien i . Cette valeur est transmise au sein du paquet de contrôle de congestion.
- RTT_i = représente la distance qui sépare le routeur de l'élément de ce lien.
- $(l_{ri} - l_{oi} + 1)$ = peut être vue comme une estimation grossière du taux de perte relatif à ce lien, où l_r est le numéro de séquence du dernier paquet de données reçu dans l'ordre et l_o est le numéro de séquence du dernier paquet de données reçu.

IV.3. La structure des paquets

On distingue trois types de paquets : les paquets de données, les acquittements négatifs et les paquets de contrôle congestion. On omettra de donner la structure de ce dernier paquet dans le but de ne basé notre étude que sur l'aspect fiabilité du protocole DyRAM.



IV.3.1 Les paquets de données et de réparations⁵

Il contient deux parties. Une partie entête et une partie dédié aux informations supplémentaires.

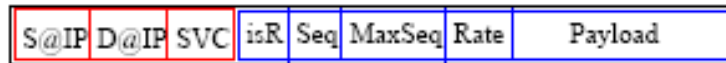


Fig.3.1. Paquet de données DyRAM

L'entête du paquet contient les champs suivants :

- S@IP: représente l'adresse IP de la source de données
- D@IP: représente l'adresse IP multicast du groupe de destination.
- SVC : représente un champ drapeau. Il indique si le service actif doit être activé pour ce paquet de donnée.

L'autre partie du paquet contient les informations suivantes :

- isR : ce champ permet de distinguer entre une transmission de données et une retransmission.
- Seq : représente le numéro de séquence du paquet transmis.
- Maxseq : Ce champ permet aux récepteurs de mettre à jour leur cache mémoire. Il indique les paquets qui peuvent être supprimés de leur cache. Il représente le numéro de séquence maximum des paquets de données reçus correctement par l'ensemble des récepteurs.
- Rate : ce champ contient le taux de transmission des données, défini en fonction du nombre de paquets.
- Payload : représente la donnée à transmettre.

IV.3.2 Les acquittements négatifs

La structure de ces paquets de contrôle se compose d'un entête identique à celui du paquet de données et d'une partie dédiée à des informations supplémentaires.

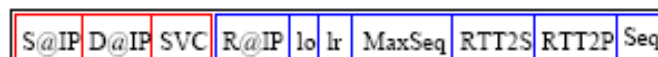


Fig.3.2. Les acquittements négatifs dans DyRAM



(5). Le protocole DyRAM utilise la même structure de paquet pour les paquets de données original et les retransmissions. Seule la valeur du champ "isR" permet de les différencier.



La deuxième partie du paquet se compose des champs suivants :

- R@IP: représente l'adresse IP du récepteur qui a généré ce NACK.
- l_o : représente le numéro de séquence du dernier paquet de données reçu dans l'ordre par le transmetteur du paquet.
- l_r : représente le numéro de séquence du dernier paquet de données reçu par le transmetteur du paquet.
- Maxseq : il indique quels sont les paquets de données dont le recouvrement local est possible.
- RTT2S & RTT2P : ils sont utilisés pour indiquer le RTT (Round Time Trip) jusqu'à la source ainsi que celui jusqu'au premier routeur actif ascendant respectivement.
- Seq : représente le numéro de séquence du paquet de données perdu.

IV.4. Le comportement des différents éléments du réseau

IV.4.1 Le comportement de la source

La source est responsable de la transmission des paquets de données. Avant chaque émission, la source stocke le paquet de données dans son cache mémoire, puis elle le transmet à l'adresse multicast de destination. Lors de la réception d'un NACK, la source déclenche un temporisateur NST (NACK Suppression Time) « Durée de suppression des NACK » pour ce NACK si cela n'est déjà fait. Si à la réception d'un NACK, un temporisateur NST est en cours pour le même paquet, cela signifie que ce NACK n'est pas valide et par conséquent celui-ci est ignoré. La période d'ignorance des NACK similaires dure un temps égal au RTT (Round Trip Time) « Temps d'un voyage circulaire » de la source jusqu'au routeur actif. La source retransmet le paquet de données demandé à l'adresse multicast de destination à l'expiration du délai d'attente.

La gestion du cache mémoire au niveau de la source s'effectue grâce au champ " l_o " des NACK transmis par les routeurs. Chaque routeur agrège les NACK reçus de ses récepteurs en un seul NACK avec une valeur du champ " l_o " 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 correctement reçus par la totalité des récepteurs de la branche descendant. Ainsi, 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 ayant 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 transmet par la suite. Cela permettra de mettre à jour les caches mémoire des récepteurs du groupe.

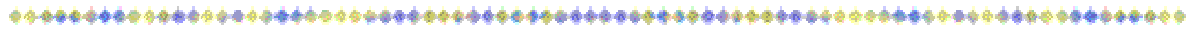
IV.4.2 Le comportement des récepteurs

Les récepteurs sont les utilisateurs finaux des paquets de données transmis dans le réseau de diffusion. Lorsqu'un récepteur reçoit un paquet de données original ayant le numéro de séquence attendu, il met à jour son numéro de séquence en réception et garde une copie du paquet de données reçu dans son cache de réception. Si le numéro de séquence du paquet de données reçu est supérieur à celui attendu cela signifie qu'une perte est survenue à son niveau. Ainsi, il procède à l'envoi immédiat d'un NACK vers son routeur actif ascendant et initialise un délai de garde égal au RTT du récepteur à la source. Lors de l'expiration du délai de garde, le récepteur renvoie un autre NACK et initialise à nouveau le délai de garde.

Il est nécessaire, au niveau des récepteurs, de déterminer une stratégie de détection des pertes autre que la détection du déséquence des paquets de données reçus. DyRAM définit une stratégie alternative. Cette stratégie permettra entre autre de détecter la perte du dernier paquet de données à recevoir. Le récepteur réinitialise un temporisateur à chaque réception d'un paquet de données ordonné. L'expiration de ce temporisateur indique que durant cette période le récepteur n'a reçu aucun paquet de données ordonné ou non. Le récepteur déduit aussi, qu'une perte est survenu et transmet un NACK relatif au paquet de données attendu.

Un récepteur peut être élu comme répondeur. Le récepteur peut, alors, recevoir des NACKs indiquant des pertes survenues au niveau des autres récepteurs. Dans ce cas, le répondeur se comporte identiquement à la source lors de la réception d'un NACK.

Il est clair que les récepteurs ne sont pas dotés d'une capacité mémoire infinie. Aussi, 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 avec un numéro de séquence inférieur ou égale à chaque nouvelle valeur du champ "Maxseq" reçue.



IV.4.3 Le comportement des routeurs actifs

Les routeurs actifs sont des dispositifs intermédiaires qui permettent l'acheminement des paquets de la source jusqu'aux récepteurs tout en permettant un recouvrement local des éventuelles pertes de données.

DyRAM adapte les routeurs actifs du réseau afin de leur permettre une détection rapide et facile des pertes de données éventuelles à leur niveau (ces pertes peuvent affecter la totalité du sous-arbre).

DyRAM définit, aussi, une structure TL pour chaque session multicast au niveau de chaque routeur actif. Cette structure à trois composantes :

- Le numéro de séquence du dernier paquet de données reçu dans l'ordre.
- Le numéro de séquence du dernier paquet de données reçu.
- La liste des paquets de données perdus. Ce sont les paquets de données ayant un numéro de séquence supérieur au numéro de séquence du dernier paquet reçu dans l'ordre et inférieur au numéro de séquence du dernier paquet reçu.

Cette structure permet aux routeurs de détecter plus rapidement les pertes éventuelles de données.

La structure NS est une structure logique où le routeur sauvegarde le numéro des liens d'où il a reçu des NACK similaires. Cela permet de procéder ultérieurement au recouvrement des pertes. Le temporisateur NST est la durée de vie d'une structure NS. Durant cette période de temps tous les NACK reçus et identiques au NACK de la structure sont considéré comme invalide. Le temporisateur DTD représente la durée de temps nécessaire à l'élection d'un répondeur parmi les récepteurs descendant du routeur actif.

D'où, si un routeur détecte une perte de données en recevant un paquet de données ayant un numéro de séquence supérieur au numéro de séquence attendue, il déclenche, alors, une procédure de réparation en envoyant un NACK à son ascendant, en construisant une structure NS et en initialisant un temporisateur NST égal au RTT du routeur à la source. Durant ce délai de garde, tous les acquittements identiques à celui transmis et provenant des liens descendants du routeur sont ignorés. A l'expiration du temporisateur, le routeur actif transmet le NACK et initialise à nouveau le temporisateur.



En revanche, si le routeur reçoit un paquet de données attendu, il procède comme suit :

- Il met à jour son numéro de séquence en réception.
- Il initialise un temporisateur qu'on désignera par T_f . Ce temporisateur servira à détecter la perte du dernier paquet de données transmis par la source. La valeur de ce temporisateur est égale au minimum au $1/D$ où D est le taux courant de transmission de la source. Le routeur extrait ce taux du dernier paquet de données reçu grâce au champ dédié à cet effet.
- Il transmet le paquet à ses descendants dans l'arbre multicast.

Lorsqu'un routeur reçoit un NACK provenant de l'un de ses descendants, il procède de la manière suivante :

- Il met à jour l'état du lien dans la structure LS correspondante et vérifie que ce lien n'a acquitté aucun paquet demandé par d'autres liens. S'il s'avère que le récepteur du lien a bien reçu un paquet demandé par d'autres récepteurs alors le lien est directement élu comme répondeur pour le paquet perdu et le DTD correspondant est désactivé.
- Il vérifie si un NACK similaire a été déjà reçu ou s'il a déjà été transmis au routeur ascendant.
- Si c'est le cas, le NACK n'est pas valide. Le routeur entame des vérifications supplémentaires. Il vérifie l'existence d'un temporisateur DTD actif pour ce NACK.
 - Si c'est le cas et si la liste de subcast de la structure NS correspondante contient tous les liens descendants de ce routeur alors le NACK est directement retransmis au lien ascendant, le temporisateur DTD est désactivé et un temporisateur T_N est initialisé. Si la liste est incomplète, le NACK reçu est ignoré après avoir mis à jour la liste des demandeurs de réparation.
 - Par contre, si aucun DTD n'est actif pour ce NACK, cela signifie qu'un répondeur a déjà été sélectionné et que le routeur est en attente d'une réparation. Le récepteur demandeur de réparation est ajouté à la liste de retransmission et le NACK est supprimé.
- Lors de la réception d'un NACK valide, le routeur actif construit une structure NS pour ce NACK et initialise un temporisateur NST. Par conséquent, le routeur actif



procède à la vérification des différentes structures LS construites au niveau de ce routeur à la recherche d'un lien ayant acquitté positivement le paquet demandé. Si ce lien existe, le routeur se charge de transmettre immédiatement le NACK à ce lien sinon un temporisateur DTD est initialisé.

A la réception d'un paquet de réparation, le routeur actif vérifie la structure NS correspondante et extrait la liste des liens l'ayant demandé. Le routeur ne transmet le paquet de réparation qu'aux descendants concernés par la perte de données.

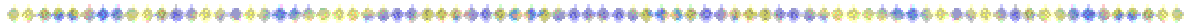
A l'expiration de la durée de l'élection d'un répondeur (DTD), le routeur actif procède à l'élection d'un répondeur parmi les descendants n'ayant pas signalé de perte pour ce paquet. L'élection se fait selon la valeur du poids attribuée à chacun de ses liens. Un poids maximum permet au lien d'être élu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

IV.5. Les algorithmes fonctionnels du protocole DyRAM

Dans ce qui suit, on donnera les algorithmes relatifs au fonctionnement de chaque entité du réseau tel que défini par le protocole DyRAM.

A. La source :

- ✦ Lors de l'envoi d'un paquet de données :
 - Stocker le paquet de données dans son cache mémoire.
 - Affecter la valeur adéquate au champ "Maxseq" du paquet de données.
 - Emettre le paquet de données à l'adresse multicast de destination.
- ✦ Lors de la réception d'un NACK :
 - Extraire la valeur de champ "l_o" du NACK reçu.
 - Mettre à jour le cache mémoire selon la valeur extraite.
 - **Si NACK valide alors**



- Initialiser un temporisateur NST égal au RTT de la source au prochain routeur.

Sinon

- Supprimer le NACK.

✦ A l'expiration du temporisateur NST :

- Retransmettre le paquet de données à l'ensemble des récepteurs du groupe multicast de destination.

B. Les récepteurs :

✦ A la réception d'un paquet de données :

- Extraire la valeur de champ "Maxseq" du paquet de données reçu.
- Mettre à jour le cache mémoire selon la valeur extraite.
- **Si** n° de séquence correspond au n° attendu **alors**
 - Mettre à jour le n° de séquence en réception.
 - Sauvegarder une copie au niveau du cache mémoire.
 - Armer un temporisateur T_f .

Sinon

- Envoyer un NACK au routeur actif ascendant dans l'arbre multicast.
- Initialiser un délai de garde égal au RTT du récepteur à la source.

✦ A la réception d'un paquet de réparation :



- Mettre à jour le n° de séquence en réception.
- Armer un temporisateur T_f .
- ✦ A la réception d'un NACK :
 - Envoyer un paquet de réparation au routeur actif ascendant.
- ✦ A l'expiration du délai de garde :
 - Envoyer un NACK au routeur actif ascendant.
 - Réinitialiser le délai de garde.
- ✦ A l'expiration du temporisateur T_f :
 - Envoyer un NACK au routeur actif ascendant.
 - Réinitialiser le temporisateur T_f .

C. Les routeurs actifs :

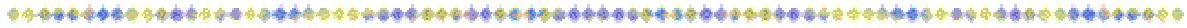
- ✦ A la réception d'un paquet de données :
 - Mettre à jour les informations de la structure LS.
 - **Si** le n° de séquence correspond au paquet attendu **alors**
 - Réinitialiser le temporisateur T_f .
 - Expédier le paquet de données aux descendants du routeur.

Sinon

- Envoyer un NACK vers le routeur actif ascendant.
- Initialiser un délai de garde égal au RTT du routeur à la source.



- ✦ A la réception d'un acquittement négatif provenant d'un descendant
 - Mettre à jour la structure LS correspondante.
 - **Si** le lien a bien reçu un paquet demandé **alors**
 - Le récepteur du lien est élu comme répondeur.
 - Expédier le NACK correspondant au répondeur élu.
 - Désactiver le temporisateur DTD correspondant.
 - **Si** un temporisateur NST est actif pour ce NACK **alors**
 - Le NACK reçu est invalide.
 - Mettre à jour la structure NS correspondante.
 - **Si** un temporisateur DTD est actif pour ce NACK **alors**
 - * Vérifier la liste de retransmission de la structure NS correspondante.
 - * **Si** les récepteurs descendant sont tous dans la liste **alors**
 - Expédier directement le NACK au routeur ascendant.
 - Désactiver le temporisateur DTD correspondant.
- Sinon**
- Le NACK est supprimé.

**Sinon**

- * Le NACK est supprimé.

Sinon

- Le NACK reçu est valide.
- Construire une structure NS pour ce NACK.
- Initialiser le temporisateur NST correspondant.
- Parcourir les structure LS du routeur à la recherche d'un récepteur ayant reçu le paquet demandé.
- **Si** le récepteur existe **alors**
 - * Le récepteur est élu comme un répondeur.
 - * Expédier immédiatement le NACK au répondeur élu.

Sinon

- * Initialisation d'un temporisateur DTD pour ce NACK.

✦ A la réception d'un acquittement négatif provenant du routeur ascendant

- Election d'un répondeur parmi les récepteurs ayant bien reçu le paquet demandé.
- Transmission du NACK au récepteur élu.



✦ A la réception d'un paquet de réparation

- Rechercher les récepteurs demandeurs de réparations.
- Subcaster le paquet de réparation uniquement au récepteurs demandeurs.

✦ A l'expiration du temporisateur T_f

- Transmettre un acquittement négatif relatif au paquet de données attendu.
- Initialiser un délai de garde égal au RTT du routeur à la source.

✦ A l'expiration du délai de garde

- Retransmettre l'acquittement négatif relatif au délai de garde a son ascendant.
- Réinitialiser le délai de garde.

✦ A l'expiration du temporisateur DTD

- Élire un répondeur parmi les récepteurs n'ayant pas transmis de NACK pour le paquet de données relatif au DTD.

✦ A l'expiration du temporisateur NST

- Transmettre au répondeur élu un NACK relatif au NST concerné.



V. Conclusion

Tout au long de ce chapitre, on a donné une présentation assez détaillée de trois protocoles de fiabilité multicast. Le protocole AER est un protocole de fiabilité multicast basé sur l'initiative des récepteurs. C'est un protocole assez simple à mettre en place, il n'utilise pas de techniques complexes et lourdes à implémenter. Cependant, il présente de nombreuses failles. Il réduit de très peu le problème d'implosion des NACK que connaissent les approches bout en bout. Aussi, ce protocole, comme c'est le cas pour de nombreux autres protocoles, ne fonctionne correctement que sur l'hypothèse d'une capacité de cache infinie. Le protocole MAF est un protocole qui rend un service de diffusion totalement fiable et adapté à des groupes de diffusion de grande taille, en définissant des mécanismes de recouvrement d'erreur que la technologie des réseaux actifs permet de déployer et d'implanter dynamiquement dans les routeurs actifs de l'arbre de diffusion. Le prototype de MAF a été implanté au-dessus de l'architecture de réseaux actifs développée dans le cadre du projet RNRT Amarrage [15] et il a été déployé à l'échelle géographique de la France. Les expérimentations menées, ont permis de valider le modèle de communication fiable point à multipoint actif de ce protocole. Toutefois, ce protocole nécessite un échange de paquets assez dense représenté par les paquets de données, les paquets de contrôles et les LDM périodiquement transmis par la source. Notre présentation s'est focalisée sur le protocole de fiabilité DyRAM étant donné qu'il est le plus récent des trois protocoles proposés. Aussi, on prendra ce protocole comme référence pour notre étude comparative du protocole AMRHy. Ce chapitre a permis de donner l'algorithme détaillé du fonctionnement des différents entités du protocole DyRAM ce qui sera utile lors de l'implémentation du protocole. Le protocole DyRAM permettra d'évaluer les performances du protocole AMRHy qu'on présentera dans le chapitre IV.

Chapitre IV



Le protocole AMRHy

"L'expérience fait l'art, l'inexpérience la fortune. On fait des découvertes en cherchant et des trouvailles par hasard."

-Joseph Joubert-

I. Introduction

Le but du présent mémoire de magistère est la présentation et l'évaluation du protocole AMRHy en comparant ses performances aux performances des protocoles existants. Ce chapitre est une présentation et une description de ce protocole. Il englobera une description détaillée de ses fonctionnalités et son algorithme complet. On mettra en valeur les potentialités de ce protocole et les avantages qu'il représente pour des réseaux à forte contraintes en terme de probabilité de perte et en terme de densité.

II. Motivation

En parcourant la littérature existante dans le domaine de la fiabilité multicast, il parut clair que la majorité des solutions proposées ont la particularité d'avoir été développées pour un type bien déterminé d'applications. Il existe des protocoles de transport multicast qui ont axé sur l'aspect acheminement des données en temps réel. Ces protocoles veillent à satisfaire pleinement les applications multicast sensibles au retard d'acheminement mais ne fournissent aucune ou très peu de garantie sur l'intégralité et l'intégrité des données reçus.

D'autres protocoles proposent des solutions pour le problème de la fiabilité des transmissions. Les solutions sont efficaces en termes de fiabilité mais nécessitent un délai d'acheminement assez important ce qui les rend inadéquats pour les applications temps réel. Les protocoles récemment développés, intègrent des fonctionnalités qui permettent à la fois une fiabilité totale des données transmises et une certaine maîtrise du délai d'acheminement. Cependant, ce compromis a engendré des protocoles gourmands en bande passante et nécessitant des traitements complexes au niveau des différents éléments du réseau.

Hors, une petite fouille dans les articles de recherche publiés par les plus grandes institutions de recherche en informatique laisse entre voir une nouvelle ère de réseau informatique : l'ère des réseaux sans fil. En effet, le besoin de mobilité est de plus en plus ressenti. Les interactions commerciales, les procédures de vente et d'achat et bien d'autres activités qui s'effectuaient naturellement au sein des bureaux administratifs se font aujourd'hui dans un café, dans le hall d'un hôtel ou encore coincé dans un embouteillage. La technologie sans fils gagne du terrain et s'impose comme l'avenir des réseaux informatique. Aussi, il est bien naïf de continuer à développer des solutions pour les réseaux informatiques sans tenir compte de l'avenir sans fils de ces derniers.

Les réseaux sans fils proprement dit représentent des entités mobiles ayant une autonomie en énergie réduite et communiquent grâce à un médium capricieux dans ce sens où :

- La diffusion des signaux est complexe et très peu maîtrisée.
- Son utilisation est règlementée et standardisée et nécessite des autorisations strictes.
- La puissance du signal en émission subit d'importantes atténuations ce qui nécessite l'ajout de dispositif d'amplification au niveau des récepteurs.
- L'existence de nombreuses sources de perturbation et de bruit altère la qualité du signal transmis.
- ...etc.

Aussi, il est clair, qu'un protocole de recouvrement des transmissions multicast sans fils est le bien venu. C'est dans cette perspective que ce présent travail expose une solution de recouvrement multicast conçu initialement pour des réseaux filaires et qui peut sans difficulté être ajusté aux réseaux sans fils.

III. Présentation générale

Le protocole proposé se base sur des services actifs implémentés au niveau des routeurs du réseau. Il se base sur une architecture hiérarchique où la disposition des routeurs actifs est étudiée de façon à éviter les problèmes survenue lors du déploiement des réseaux à forte densité [18]. La racine de l'arbre représente la source et les feuilles constituent les récepteurs. Les routeurs actifs adoptés par le protocole assurent en plus de leurs taches habituelles, les fonctions supplémentaires suivantes :

Tâches	Utilité
Le cache des paquets de données reçus.	Réduire la latence de recouvrement des pertes survenues
L'agrégation et la suppression des paquets de contrôle redondants.	Eviter le problème de l'implosion
La retransmission sélective des paquets de réparation	Limiter la diffusion des retransmissions ; Gain de bande passante ; Minimiser les traitements au niveau des composants du réseau ; Eviter la congestion.
L'élection d'un répondeur parmi les récepteurs.	Une distribution efficace de la charge de retransmission ; Réduction de l'espace mémoire nécessaire au niveau des entités actives du réseau.

L'activation des services actifs au niveau des routeurs dépend du type de transmission. Seules les transmissions multicast permettent d'activer ses services au niveau des routeurs tout le long de l'arbre multicast et cela grâce à un champ supplémentaire au niveau de l'entête des paquets de données.

Le protocole proposé peut être vu comme un protocole hybride. Il englobe l'approche hiérarchique et l'approche basée temporisateur mais aussi se base à la fois sur le recouvrement initié par la source et sur le recouvrement initié par les récepteurs (cohabitation des paquets ACK et NACK). La combinaison de toutes ces approches différentes au sein de ce protocole à motiver son appellation. Le protocole a été baptisé « AMRHy » pour "Active Multicast Reliable Hybrid protocol" en français "Protocole hybride de fiabilité Multicast actif".

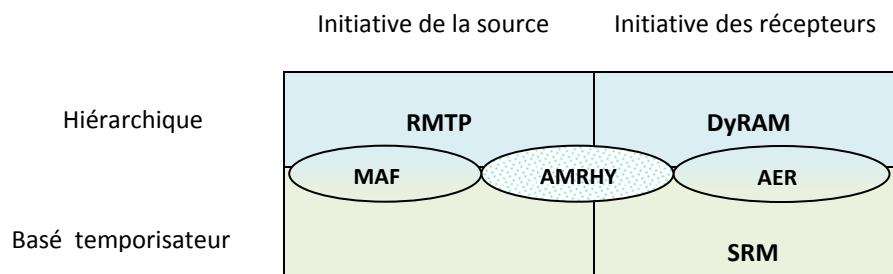


Fig.4.1. Positionnement du protocole AMRHy par rapport aux protocoles RM

La combinaison des différentes approches et classes de recouvrement a permis au protocole AMRHy d'hériter des avantages relatifs à chaque classe tout en comblant du mieux que possible les lacunes de chacune d'elles.

Aussi, cette combinaison a permis de résoudre les problèmes liés à l'implosion des paquets de contrôle, la surcharge des dispositifs du réseau et l'exploitation de la bande passante du réseau.

III.1. La gestion des paquets de contrôle

C'est la combinaison de l'approche hiérarchique et de l'approche basée sur les temporisateurs qui a permis de résoudre efficacement le problème de l'implosion des paquets de contrôle au niveau de la source. Une première suppression est réalisée grâce à l'agrégation des paquets redondants au niveau des routeurs actifs pour chaque niveau de l'arbre multicast. Une autre suppression est effectuée au niveau des récepteurs comme cela se fait dans les protocoles SRM [25] et AER [24]. Un routeur actif lors de la réception d'un ACK provenant de l'un de ses descendants, le transmet au reste des descendants. Un récepteur l'ayant reçu s'informe que le paquet a été déjà acquitté et supprime, ainsi, son propre acquittement s'il ne l'a pas déjà

transmis. Dans le cas contraire, le récepteur se rendra compte qu'il a perdu ce même paquet et pourra, ainsi, signaler sa perte avant l'expiration du délai d'attente. Cela représente un gain en délai de transmission.

III.2. La répartition de la charge de recouvrement

Commençant par définir le problème. Comme nous l'avons vu tout au long du chapitre précédent, chaque protocole propose une solution au problème de fiabilité dans les transmissions multicast en chargeant l'un des composants du réseau (source, routeur, récepteur) de la lourde tâche du recouvrement des pertes. Cela implique non seulement la surcharge de l'élément concerné comme c'est le cas pour la source mais aussi la nécessité d'implanter des traitements complexes à leur niveau ou encore de les munir d'un espace de stockage assez important comme c'est le cas pour les routeurs actifs et les récepteurs du réseau.

C'est pour cette raison que le protocole AMRH_y propose une distribution équitable de la charge de recouvrement des pertes entre la source, les routeurs actifs et les récepteurs. Cela est possible grâce à l'utilisation conjointe du recouvrement orienté récepteur et orienté source. La détection des pertes au niveau de la source se fait à l'aide d'un temporisateur relatif à chaque paquet de données émis. Cette technique permet de combler la faille constatée des approches à l'initiative des récepteurs. Cette faille survient lorsqu'un paquet de données est perdu et que les NACK le désignant sont perdus en route alors que la nécessité en espace de stockage pousse à la suppression des paquets les plus anciens, tout cela fait que la récupération du paquet de données perdu est impossible même si au bout du compte un NACK parvient, malgré tout, à la source.

La détection des pertes de données au niveau des routeurs et des récepteurs est déclenchée lors du déséquencement des paquets reçus ou à la réception d'un ACK désignant le paquet de données attendu. Les récepteurs annoncent la perte par la transmission d'un NACK relatif au paquet perdu vers le routeur actif ascendant comme cela se fait pour l'approche orientée récepteurs.

Aussi, le recouvrement des pertes est partagé entre les entités du réseau comme suite :

- Au niveau de la source lors de l'expiration du délai d'attente.
- Au niveau du routeur lors de la réception d'un NACK, de ses descendants, relatif à un paquet de données existant dans son buffer.

- Au niveau des récepteurs élus comme répondeur lors de réception d'un NACK transmis par son routeur ascendant.

III.3. L'exploitation de la bande passante et des éléments du réseau

Pour qu'un protocole soit le plus performant possible, il est nécessaire de gérer au mieux les ressources du réseau en terme de bande passante, de capacité de traitement et d'espace mémoire. Ainsi, le protocole AMRHy procède à une retransmission des paquets perdus de manière à réduire la bande passante nécessaire au recouvrement et à minimiser les traitements qui en découle. La retransmission des paquets perdus se limite aux récepteurs ayant effectivement perdu des paquets. Pour ce fait, AMRHy s'inspire des protocoles ARM [26] et DyRAM [23] : chaque routeur construit une structure logique pour chaque paquet perdu. Cette structure permet de garder la trace des récepteurs ayant signalé une perte du paquet. Lors de la retransmission, le routeur consulte la structure correspondante et ne retransmet qu'aux récepteurs figurant dans cette liste. On parle de Subcast.

Le tableau ci-dessous résume la différence entre les stratégies de recouvrements adoptées par les protocoles multicast abordé précédemment et celle adoptée par le protocole AMRHy.

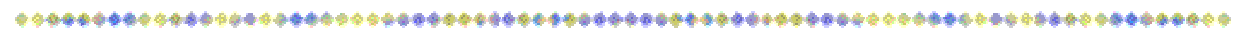
	AER	MAF	DyRAM	AMRHy
Explosion des acquittements	Agrégation et suppression	Agrégation	Agrégation	Agrégation et suppression
Charge de recouvrement	Source et serveur de réparation	Source et chef de sous-groupe	Source et récepteur élu	Source, routeur actif et récepteur élu
Portée de retransmission	Multicast et Subcast	Subcast et Unicast	Multicast, Subcast et Unicast	Multicast, Subcast et Unicast

Fig.4.2 Tableau récapitulatif et comparatif des protocoles de fiabilité multicast présentés

IV. Particularité du protocole AMRHy

Le Protocole AMRHy à la particularité de fonctionner correctement avec un support minimal des routeurs actifs :

- L'élection d'un répondeur parmi les récepteurs permet un recouvrement local fonctionnel sans l'intervention des services actifs des routeurs.



- Le recours à l'utilisation des acquittements positifs au niveau des récepteurs permet une bonne gestion du cache mémoire disponible au niveau des routeurs. Une capacité mémoire minimal est suffisante pour un recouvrement local efficace. Par conséquent, le protocole AMRHy se comporte parfaitement bien dans un environnement dénué de tout service actif. Dans ce cas, le protocole adopte une approche de bout en bout entre la source et les récepteurs. Le recouvrement des pertes est effectué conjointement par la source et les récepteurs. Ainsi, l'apport des services actifs est uniquement qualitatif, ils permettent d'améliorer les performances du protocole en terme de besoin en mémoire et en délai d'acheminement.

V. Description fonctionnelle du protocole 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 multicast construit cette effet par un protocole de routage. Une fois que les services sont invoqués, chaque entité (source, routeurs actifs et récepteurs) se comporte de la manière suivante :


V.1. Comportement de la source

La source émet les paquets de données vers une adresse multicast à laquelle sont inscrits tous les récepteurs d'un groupe. A chaque émission, la source initialise un délai de garde, en pratique ce délai de garde doit être égal au RTT (Round Trip Time) du récepteur le plus éloigné de la source et sera ajusté au fur et à mesure que la source reçoit les acquittements. Lorsque la source reçoit un ACK du groupe, elle libère l'espace mémoire occupé par le paquet acquitté et met à jour la fenêtre d'émission. Si le délai de garde arrive à sa fin avant la réception de l'acquittement positif, la source retransmet le paquet au groupe multicast.

V.2. Comportement du récepteur

En recevant le paquet de données attendu, le récepteur entame un délai d'attente aléatoire. Si au cours de cette période le récepteur reçoit un ACK portant le même numéro de séquence, il supprime le sien et se comporte comme s'il a envoyé cet acquittement. Si le délai d'attente arrive à sa fin, le récepteur envoie un ACK à son ascendant dans l'arbre multicast.

Si un récepteur reçoit un ACK dont le numéro correspond au prochain paquet, le récepteur se rend compte qu'il a perdu un paquet et envoie un NACK à son ascendant dans l'arbre multicast. Il initialise, aussi, un délai de garde qui doit être égal au RTT du récepteur au



routeur actif ascendant. Si le récepteur reçoit la retransmission, il met à jour le numéro de séquence en réception et se comporte comme s'il a envoyé un ACK. Par contre, si le délai de garde expire, le récepteur retransmet le NACK.

Lorsqu'un récepteur, élu répondeur par le routeur actif, reçoit un NACK, il retransmet le paquet demandé à son ascendant dans l'arbre multicast.

V.3. Comportement du routeur actif

Lorsqu'un routeur actif reçoit un ACK de l'un de ses descendant, il subcast cet ACK aux autres descendants, et entame une période d'attente égal au RTT du routeur en question et son descendant le plus éloigné. L'enclenchement de ce délai permet l'agrégation des acquittements provenant des descendants et de supprimer les acquittements positifs redondants transmis vers le routeur ascendant. La transmission de l'ACK aux descendants permet d'une part, la suppression de leur ACKs pour éviter la transmission d'acquittements identiques sur le réseau et d'autre part, d'informer les descendants n'ayant pas reçu le paquet en question de sa présence dans le cache de leur routeur ascendant, ce qui réduira le temps nécessaire au recouvrement des pertes. Pendant la période d'attente, le routeur actif va ignorer tous les ACK en provenance d'un des descendants et enregistre dans une liste les liaisons sur les quelles les NACKs parviennent. Le routeur choisit comme répondeur le premier descendant qui envoie un ACK. Toujours au cours de cette période, si le routeur reçoit un ACK de son ascendant portant le même numéro de séquence, il supprime le sien et se comporte comme s'il a envoyé cet acquittement.

Lorsque la période d'attente s'achève, le routeur consulte la liste des descendants ayant perdu le paquet de données, si celle-ci n'est pas vide, et il leur subcast la retransmission du paquet perdu. L'espace mémoire occupé par ce paquet est par la suite libéré et dans le cas où le routeur actif n'a pas reçu d'ACK concernant ce paquet de son routeur ascendant, il lui transmet l'ACK correspondant. Si le routeur actif reçoit un ACK transmis par le routeur ascendant portant un numéro de séquence égal au paquet de données attendu, le routeur se rend compte qu'il a perdu un paquet de données. Il déclenche, alors, une procédure de récupération en envoyant un NACK à son ascendant et initialise un délai de garde égal au RTT du routeur à son ascendant dans l'arbre multicast.

Lorsqu'un NACK parvient de l'ascendant, le routeur retransmet le paquet demandé, s'il existe dans son cache. Dans le cas contraire, un NACK sera expédié au répondeur désigné pour la retransmission du paquet demandé et un délai de garde égal au RTT du routeur au répondeur

désigné est initialisé. A la réception du paquet de réparation, il l'expédie vers le routeur ascendant l'ayant demandé.

VI. La structure des paquets

Le fonctionnement du protocole AMRHy dépend de la transmission de deux sortes de paquets : les paquets de données et les paquets de contrôle.

VI.1. Les paquets de données

Le protocole AMRHy propose une structure de paquet englobant bien entendu les informations habituelle à une transmission multicast (adresse IP de la source, l'adresse multicast de destination et l'information transmise) et les informations supplémentaires suivantes :

- **Numseq**: numéro de séquence identifiant le paquet de données.
- **Ind** : champ indicateur permettant de distinguer les paquets de données originaux des retransmissions.
- **Idsa** : champ indicateur de service actif.
- **Orgadr** : champ permettant d'identifier l'origine du paquet. L'adresse IP contenue dans ce champs est mise à jour à chaque passage par un routeur actif. Ce dernier sauvegarde l'adresse IP contenue et la remplace par sa propre adresse. Ainsi, le routeur actif identifie son ascendant dans l'arbre multicast. L'adresse initiale est l'adresse de la source.

Le mécanisme utilisé dans la mise à jour du champ "Orgadr" permet de construire le chemin inverse que suivront les paquets de contrôle. Cette technique est inspirée du fonctionnement du protocole AER [24]. Cependant, le protocole AER procède à la transmission de paquet de signalisation supplémentaire et n'intègre pas ce mécanisme au niveau de ses paquets de données. L'intégration du champ "Orgadr" au niveau des paquets de données a permis de réduire le nombre de paquets échangés dans le réseau¹.

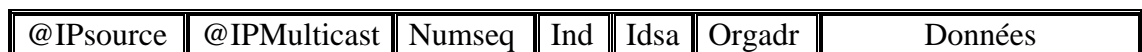
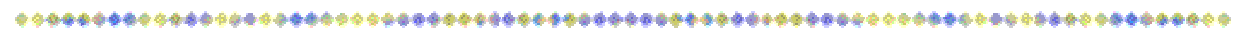


Fig.4.3. Structure des paquets de données du protocole AMRHy

(1) : Cette réduction est un atout supplémentaire dans l'adaptation du protocole AMRHy pour les réseau sans fils.



VI.2. Les paquets de contrôle

Les paquets de contrôle du protocole AMRHy sont assez simples. Ils englobent quatre champs : l'adresse source, l'adresse destination (l'ascendant direct dans l'arbre multicast), le numéro de séquence du paquet perdu, et un champ identifiant qui permet de distinguer entre un acquittement positif et acquittement négatif.

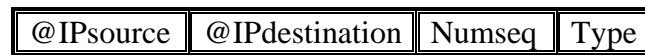


Fig.4.4. Structure des paquets de contrôle du protocole AMRHy

VII. L'algorithme fonctionnel du protocole AMRHy

VII.1. 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 à laquelle sont inscrits tous les récepteurs d'un groupe ;

✦ A la réception d'un ACK :

- Libération du buffer du cache associé au paquet de données acquitté ;
- Ajustement de la fenêtre en émission de la source ;



✦ A l'expiration du délai de garde :

- Retransmission du paquet de données à l'adresse multicast tout en réinitialisant le délai de garde ;

VII.2. Comportement des récepteurs

✦ 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 ;
/*Le récepteur se comporte comme s'il a envoyé un ACK à l'ascendant dans l'arbre multicast*/

✦ A la réception d'un ACK :

- **Si** le numéro de séquence n'existe pas parmi la liste des paquets reçus **alors**
/*Détection d'une perte d'un paquet de données 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 ;

***Sinon***

* *Si* le délai d'attente du récepteur est armé *alors*

- désarmer le délai d'attente ;

fsi.

/*Le récepteur se comporte comme s'il a envoyé un ACK*/

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 :

- Envoi d'un ACK vers le nœud parent dans l'arbre multicast ;

/* Le délai d'attente expire avant que le récepteur ne reçoit un ACK portant le même numéro*/

✦ 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 ;



VII.3. Comportement des routeurs actifs

✦ 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 multicast ;

✦ A la réception d'un paquet de réparation :

/* ce paquet sera traité 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 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 nœud père*/

- * *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 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 multicast ;



- Enregistrement de l'adresse du répondeur ;

Sinon /* le paquet de données a été reçu */

- Pas d'envoi d'un ACK au nœud père ; /*suppression locale des ACKs*/

Fsi ;

Fsi ;

✦ A la réception d'un NAK :

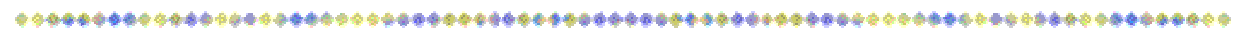
- Ajout de l'adresse du fils à la liste des fils n'ayant pas reçu le paquet de données ;

✦ 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.



VIII. Conclusion

Ce chapitre a permis de présenter le protocole AMRHy en détail afin de cerner les différentes facettes et avantages de ce protocole. Un parcours rapide de ce qui a été présenté dans ce chapitre permet de déduire le potentiel énorme que représente le protocole AMRHy. Il permet un recouvrement efficace et puissant tout en évitant les problèmes communément rencontrés par les autres protocoles de recouvrement multicast, il nécessite un support minimal des routeurs actifs et offre un délai d'acheminement adapté au application temps réel sans l'utilisation d'un espace de stockage infini. 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.

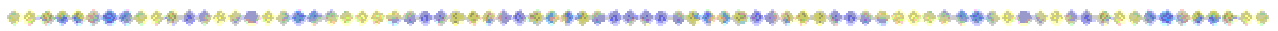
Chapitre V



Simulation comparative des protocoles AMRHy et DyRAM

"L'expérience est une observation provoquée dans le but de faire naître une idée."

-Claude Bernard-



I. Introduction

Ce chapitre est le cœur même du mémoire. Il présente la contribution principale de notre travail dans le développement du protocole AMRHy : la simulation et l'implémentation des protocoles de fiabilité multicast AMRHy et DyRAM. On commencera par donner les motivations d'une simulation NS2 ainsi qu'une brève présentation du simulateur. La suite du chapitre sera consacrée au déroulement de la simulation et de l'étude comparative : implémentation des protocoles, leur simulation et les résultats obtenus. Des propositions d'améliorations des performances du protocole AMRHy seront exposées en regard des résultats obtenus. On conclura ce chapitre en résumant les performances et les faiblesses du protocole AMRHy selon les résultats obtenus.

II. Motivation

Le protocole de fiabilité multicast AMRHy [17] a été le sujet d'une publication et a été soumis aux critiques de la communauté scientifique. L'une des remarques les plus pertinentes est l'absence de simulation confirmant le comportement attendu du protocole. En effet, une simulation MATLAB permet de représenter le comportement du protocole à un haut niveau ; les détails liés aux couches inférieures des nœuds ne sont pas pris en compte. Les problèmes qui peuvent surgir de ces détails ne sont pas visibles. Une implémentation en C++ permet, aussi, de valider le comportement du protocole au dessus de la couche transport en introduisant la programmation des sockets et la résolution des problèmes qu'elle présente. Cependant, la simulation MATLAB ainsi que l'implémentation C++ représentent un comportement dicté par la perception globale que possède le programmeur du protocole en question. De son côté, une simulation grâce au Network Simulator (NS2) permet de définir le comportement de chaque couche du modèle OSI d'un nœud. Tous les problèmes qui peuvent affecter une implémentation réelle du protocole sont visibles. Le comportement individuel de chaque composant du réseau contribue à l'émergence du comportement global du protocole : les résultats obtenus ne dépendent pas d'une perception mais dépendent uniquement de l'interaction entre le comportement de chaque nœud comme c'est le cas dans une implémentation réelle.



III. Le simulateur "Network Simulator 2" : NS2

NS2 est un simulateur orienté objet à événement discret. Il a été développé à l'université de Berkeley, écrit en C++ et muni d'un interpréteur Otcl (dérivé du langage Tcl avec une extension orienté objet développé par le MIT). NS2 est un outil de recherche très utilisé dans la conception et la mise au point des protocoles d'échange d'informations. Il est utile aussi bien dans l'étude des protocoles de routage qu'à l'étude des réseaux mobiles ou des communications par satellites. Cependant, son utilisation est assez complexe et nécessite une compréhension approfondie de toutes ses classes, sous-classe, types, attributs et même tous ses fichiers d'implémentation. Des détails supplémentaires sur le simulateur et son fonctionnement sont fournis en annexe.

IV. Le déroulement de la simulation

Le travail effectué est basé sur la simulation NS2 du protocole AMRHy et du protocole DyRAM. Durant cette étude, on s'est concentré sur le protocole AMRHy n'ayant jamais été sujet à une simulation NS2 approfondie.

Une simulation NS2 a essentiellement deux parties à implémenter :

- **La topologie** : elle représente la partie implémentée grâce au langage interprété "TCL/OTCL".
- **Le comportement** : elle décrit le comportement des éléments du réseau définis dans la topologie. Elle est implémentée en C++.

IV.1. La topologie du réseau

La topologie consiste en un ensemble de nœuds organisés selon un arbre hiérarchique multicast. La première idée de topologie a été de construire une topologie simple mais qui permet de bien percevoir la totalité du fonctionnement du protocole AMRHy. Il a été vite remarqué que la topologie illustrée dans la figure.5.1 est une topologie qui ne permet pas de visualiser tous les échanges de paquets que prévoit le protocole AMRHy.

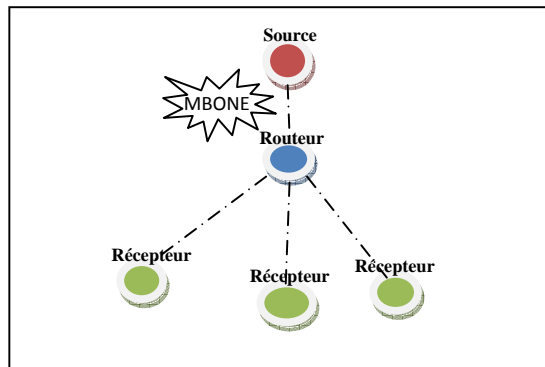


Fig.5.1. Topologie simple à un seul niveau

On a fait le choix d'utiliser malgré tout cette topologie pour vérifier son effet sur les performances du protocole en la comparant au résultat obtenue pour une autre topologie plus élaborée.

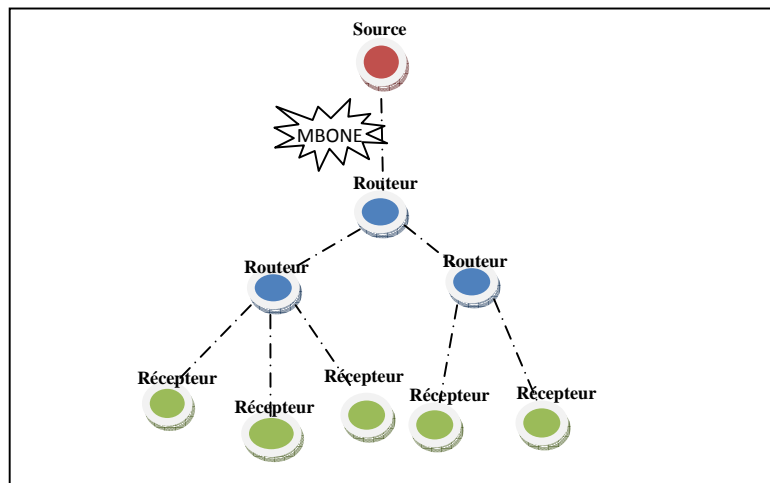


Fig.5.2. Topologie à deux niveaux

Les résultats obtenus pour les deux topologies sont presque identiques. Cependant, l'utilité des différentes fonctionnalités prévues lors de la conception du protocole AMRHy est plus visible pour des topologies à plusieurs niveaux.



Notre étude se base sur les hypothèses suivantes :

- Les liens entre la source de données et les routeurs actifs sont fiables. On suppose qu'ils représentent le MBONE qui sépare la source des routeurs actifs.
- Les liens entre les routeurs actifs sont identiques (même débit théorique).
- Les liens entre les routeurs actifs et leurs récepteurs sont identiques (même débit théorique).

IV.2. Le comportement du réseau

Une fois la topologie définie, l'étape suivante est la définition du comportement des différents éléments du réseau (source, routeurs actifs et récepteurs). Le comportement d'un nœud est le résultat de l'interaction entre toutes les couches de la pile protocolaire qui le constitue. Ainsi, pour définir le comportement d'un composant du réseau, il est nécessaire de déterminer le protocole à utiliser au niveau de chaque couche. On a fait le choix d'utiliser comme protocole de routage multicast le protocole PIM-SIM [31]. Etant donné que la topologie choisie n'est pas dynamique et que les groupes multicast formés ne subissent pas de changement, l'utilisation du protocole PIM-SM permettra de minimiser le transit des paquets de routage dans le réseau et cela afin de minimiser l'influence de ces paquets dans l'étude prévue. Une fois le protocole de routage désigné, il est important de définir les différents paramètres à mettre en place dans l'implémentation des dispositifs du réseau. Ces paramètres permettront d'estimer les performances de chaque protocole.

Trois métriques sont utilisées pour déterminer les performances d'un protocole multicast fiable :

- Le nombre moyen de paquets transmis dans le réseau. (estimation de la bande passante)
- Le nombre moyen de paquets stockés au niveau du serveur de réparation, appelé "taille du buffer". (estimation du cache mémoire)
- Le temps moyen nécessaire pour transmettre de façon fiable un paquet de données. (estimation du délai d'acheminement)

V. L'étude comparative

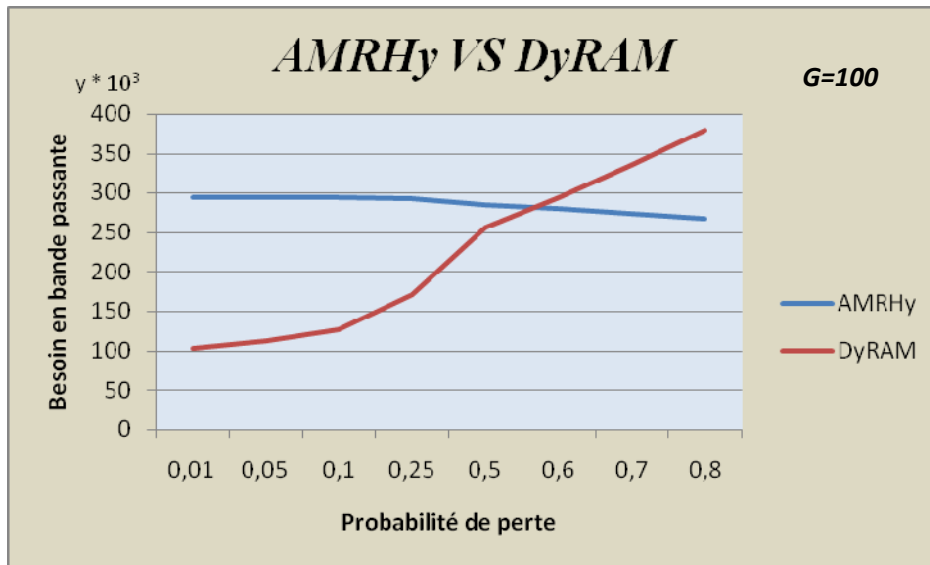
Dans ce qui suit, on exposera les résultats obtenus après l'implémentation des deux protocoles AMRHy et DyRAM au niveau de l'environnement NS2. À noter que les résultats de cette étude comparative, sont exposés avec une certaine réserve vue que rien ne prouve leur exactitude pour une implémentation réelle. Les résultats obtenus seront présentés en fonction de l'impact du taux de perte et de la taille des groupes multicast concernés par la transmission. L'accroissement de ces deux facteurs mène à la dégradation des performances d'un bon nombre de protocole multicast fiable. Les développeurs d'AMRHy l'ont conçu dans le but d'y faire face.

V.1. Analyse du trafic engendré par les deux protocoles

L'analyse du trafic engendré par la mise en place d'un protocole de fiabilité permet de déterminer le besoin en bande passante de ce dernier. Dans cette partie de l'étude, on présente une comparaison entre le besoin en bande passante des deux protocoles AMRHy et DyRAM. Soit T_{AM} , T_{Dy} , le nombre de paquets (données et réparations) circulant au-dessus des liens du réseau durant une session de transmission en utilisant respectivement les protocoles AMRHy et DyRAM. Rappelons que dans notre topologie les liens entre le serveur de réparation et ses récepteurs sont identiques et que la probabilité de perte est la même pour tous les récepteurs. T_{AM} , T_{Dy} se sont avérés sensible au taux de perte " p " et à la taille du groupe multicast " G ".

A. Analyse du besoin en bande passante en fonction de la probabilité de perte

Dans cette partie, on étudiera le besoin de chacun des deux protocoles en termes de bande passante en fonction de la probabilité de perte du réseau. Le besoin en bande passante est exprimé en nombre de paquets transitant dans le réseau durant une session de transmission donnée.

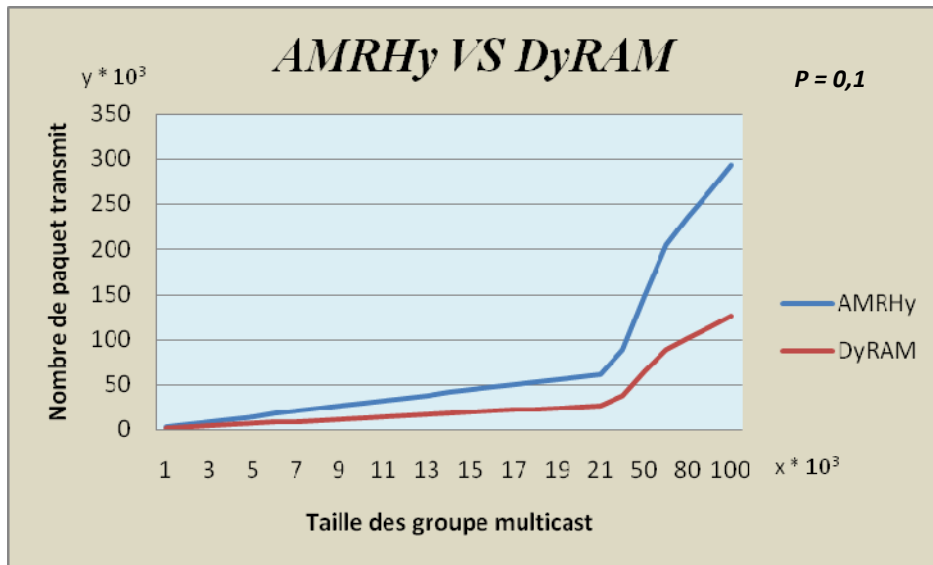


Graph 1.1 Graph 1.1 Graph comparatif des besoins en bande passante d'AMRHy et DyRAM en fonction de la probabilité de perte pour des groupes de $G=100$

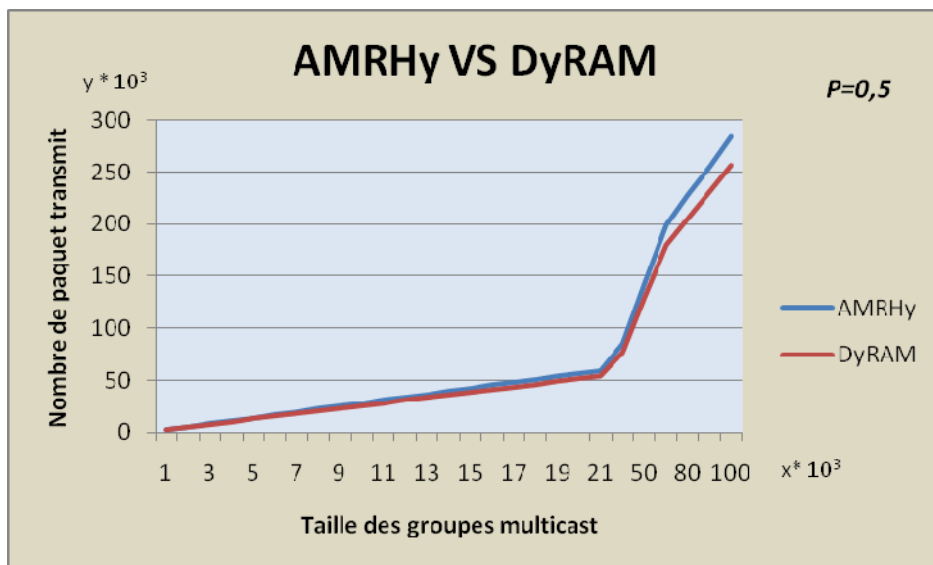
D'après les résultats obtenus, on remarque que le protocole DyRAM présente une meilleure utilisation de la bande passante pour des taux de pertes minimales. Cependant, le protocole DyRAM subit une dégradation considérable de ses performances pour des taux de perte élevés alors que le protocole AMRHy voit son besoin en bande passante diminué. Aussi, il est important de noter la constance avec laquelle le protocole AMRHy a fait face à la dégradation du moyen de transmission mais aussi l'amélioration du besoin en bande passante pour des pertes importantes. Cela peut être expliqué par le fait que les routeurs du protocole AMRHy retransmettent les acquittements positifs qu'il reçoit à l'ensemble de ces récepteurs. Pour des taux de pertes importants le nombre d'acquittements positifs diminue en réduisant ainsi le besoin en bande passante du protocole. On peut conclure que le protocole AMRHy supporte très bien les moyens de transmission non fiables alors que le protocole DyRAM présente des performances satisfaisantes uniquement pour des réseaux fiables.

B. Analyse du besoin en bande passante en fonction de la taille des groupes multicast

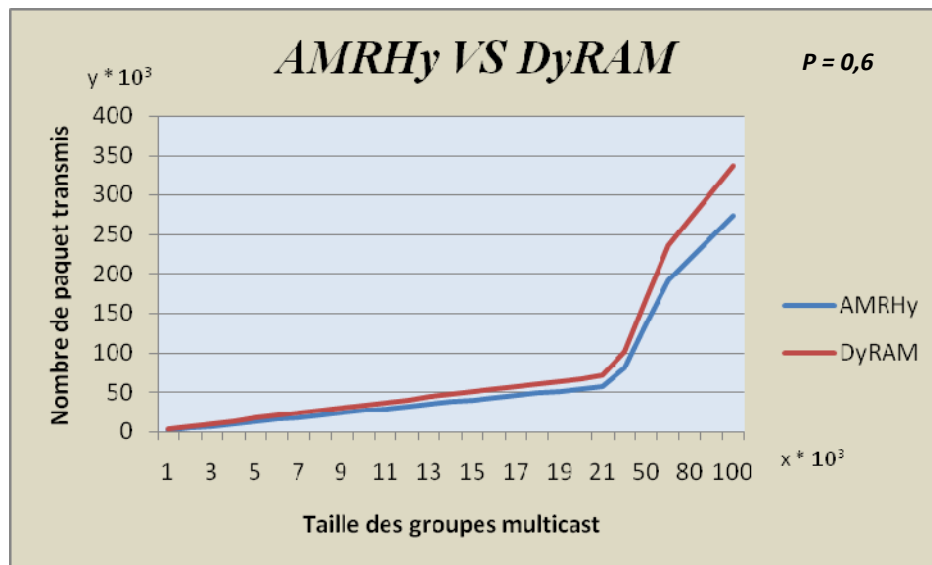
Dans cette partie, on étudiera le comportement de chaque protocole en termes de nombre de paquets transmis en fonction de la taille des groupes multicast du réseau. On effectuera notre étude en affectant à la probabilité de perte trois valeurs différentes $p=0,1$, $p=0,5$, $p=0,6$.



Graphe 1.2 Graphe comparatif des besoins en bande passante d'AMRHy et DyRAM en fonction de la taille des groupes pour une probabilité de perte p=0,1



Graphe 1.3 Graphe comparatif des besoins en bande passante d'AMRHy et DyRAM en fonction de la taille des groupes pour une probabilité de perte p=0,5



Graphe 1.4 Graphe comparatif des besoins en bande passante d'AMRHy et DyRAM en fonction de la taille des groupes pour une probabilité de perte $p=0,6$

L'analyse des trois graphes résultants, montre que pour des tailles de groupe réduit et une probabilité de perte minime le protocole AMRHy engendre un trafic nettement plus important que celui du protocole DyRAM. Cependant, le protocole DyRAM ne fournit pas la même performance pour des groupes plus larges avec un taux de perte supérieur à 50%.

AMRHy utilise beaucoup plus de bande passante que DyRAM car il se base sur la transmission explicite des acquittements positifs pour chaque paquet transmis. Le premier acquittement est par la suite retransmis à l'ensemble des récepteurs du groupe. Le protocole DyRAM, compte à lui, se base sur des acquittements positifs implicitement transmis avec les acquittements négatifs. A noter que cette étude s'est basé sur le nombre de paquets transitant dans le réseau pour déterminer le besoin en bande passante des deux protocoles et qu'aucune considération n'a été porté sur la taille des paquets transmit. En effet, les paquets de contrôle du protocole AMRHy sont basic et très léger alors que ceux du protocole DyRAM contiennent de nombreux champs de contrôle et peuvent être assez volumineux.

En résumé, AMRHy utilise moins de bande passante que DyRAM pour des taux de perte élevés et des groupes multicast importants. Cependant, pour des taux de perte moyens et des groupes multicast raisonnable, AMRHy et DyRAM ont des besoins en bande passante très proches.



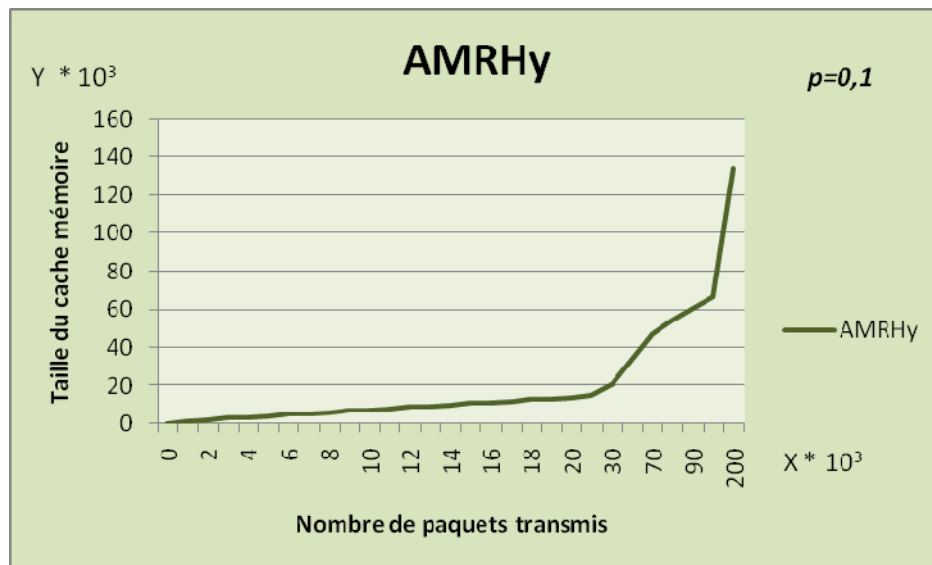
V.2. Analyse du besoin en mémoire cache au niveau des routeurs actifs

Le besoin en espace de stockage pour chaque protocole dépend du nombre de paquets qui doit être stocké et dépend aussi du temps de résidence des paquets dans le buffer. Notre analyse considère que les paquets venant de la source et allant vers un serveur de réparation arrivent avec un taux T et qu'il existe un délai R fixe qui détermine la valeur du Round Trip Time (RTT) entre chaque récepteur et son serveur de réparation. Ce délai inclut : le temps nécessaire à la réception d'un paquet, le temps de traitement du paquet et le délai nécessaire à la réception d'une réponse.

Le protocole AMRHy a été conçu de façon à garder une copie des données transmises au niveau des routeurs actifs. Le besoin en mémoire cache des routeurs du protocole AMRHy représente l'espace moyen nécessaire pour un recouvrement local efficace des pertes de données survenus au niveau de l'ensemble de son sous-arbre. Le protocole DyRAM, de son côté, ne procède pas à une sauvegarde des paquets de données au niveau des routeurs actifs. Lors de la réception d'un acquittement négatif en provenance de l'un de ses récepteurs, les routeurs actifs retransmettent directement l'acquittement à un répondeur ayant reçu correctement le paquet demandé. Pour ce fait, les routeurs doivent garder impérativement une trace de l'état de tous leurs liens descendants dans des emplacements mémoire à taille variable. Aussi, le besoin en mémoire cache des routeurs actifs du protocole DyRAM représente l'espace moyen nécessaire à la sauvegarde de l'état relatif à chaque lien descendant.

A. Analyse du besoin en mémoire cache du protocole AMRHy en fonction du nombre de paquet transmis dans le réseau

Le protocole AMRHy procède au stockage des données transmises au niveau de ses routeurs actifs. Il est intéressant de déterminer le cache hit nécessaire à un recouvrement local efficace.

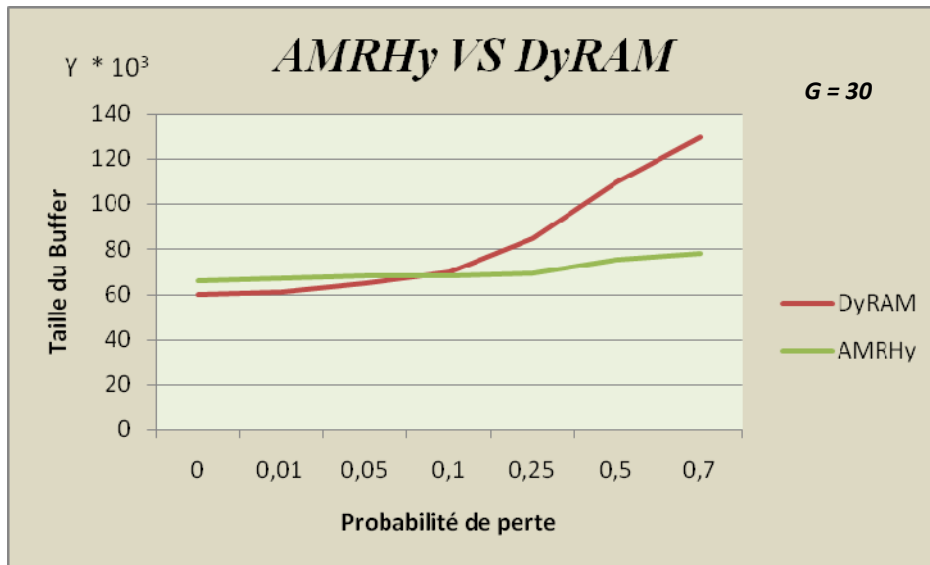


Graph 1.5 Graphique représentatif du besoin en mémoire cache du protocole AMRHy en fonction du nombre de paquets à transmettre pour une probabilité de perte $p=0,1$

D'après la courbe obtenue, on a pu estimer le cache hit à plus au moins 60% des paquets transférés à travers les routeurs actifs. Il est clair que cette estimation n'est pas très satisfaisante : la taille du buffer ne peut pas avoir une taille fixée au préalable. On peut expliquer cela par le fait que les routeurs actifs ne détruisent les paquets de données à leur niveau qu'après un temps d'attente trop long même dans le cas où tout les récepteurs ont bien reçu le paquet de données.

B. Analyse du besoin en mémoire cache en fonction de la probabilité de perte

Dans cette partie, on étudiera le comportement de chaque protocole en termes de capacité de stockage en fonction du taux de perte du réseau. On étudiera le cas d'un réseau avec des groupes multicast de 30 récepteurs ($G=30$)

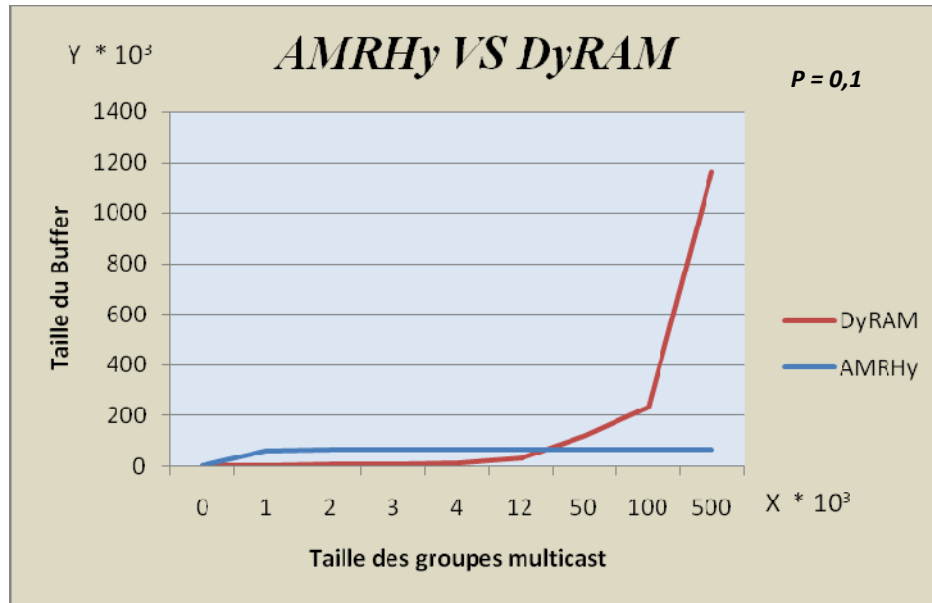


Graph 1.6 Graphe comparatif des besoins en mémoire cache des protocoles AMRHy et DyRAM en fonction de la probabilité de perte du réseau pour des groupes multicast $G=30$

Le protocole AMRHy présente, là encore, un comportement constant et positif vis-à-vis une dégradation considérable de la fiabilité du réseau de transmission alors que le protocole DyRAM présente une forte sensibilité à ce facteur. Ce comportement est justifié : le protocole DyRAM garde en mémoire la liste des paquets perdus par chacun de ses liens descendants. Plus le nombre de perte augmente plus l'espace dédié à cet usage accroît. Le protocole AMRHy, de son côté, n'est pas sensible au taux de perte du réseau car il préserve les paquets de données à son niveau pendant une période relative au RTT (Round Time Trip) des liens.

C. Analyse du besoin en mémoire cache en fonction de la taille des groupes multicast

Après avoir étudié l'effet de la probabilité de perte sur les deux protocoles, on présente une comparaison du comportement des protocoles AMRHy et DyRAM en fonction de la taille des groupes multicast pris en charge. On fixe le taux de perte à $p=0,1$.



Graph 1.7 Graphe comparatif des besoins en mémoire cache des protocoles AMRHy et DyRAM en fonction de la taille des groupes multicast pour une probabilité de perte $p=0,1$

Similairement, le protocole AMRHy présente toujours un comportement constant et positif vis-à-vis de l'élargissement des groupes multicast ce qui n'est pas le cas du protocole DyRAM qui subit une forte demande en espace de stockage au niveau de ses routeurs pour un nombre important d'adhérents aux groupes multicast. Ce comportement est dû au fait que le protocole DyRAM garde en mémoire l'état de tous ses descendants dans deux structures logique différentes. Le protocole AMRHy quant à lui procède à la sauvegarde des paquets de données de la même manière que ça soit pour un récepteur ou plusieurs milliers de récepteurs.

En somme, le protocole AMRHy, tels qu'il est décrit actuellement, ne permet pas de déterminer une taille fixe de la mémoire cache au niveau des routeurs actif et se contente de réduire la taille à 60% du nombre de paquets stockés. Cependant, le protocole a montré un comportement tout à fait convenable pour des réseaux très peu fiable et à forte densité multicast.

V.3. Analyse du délai d'acheminement des paquets de données

Cette partie de l'étude est consacrée à la détermination du temps nécessaire à la transmission d'un paquet de données de la source jusqu'au récepteur pour les deux protocoles AMRHy et DyRAM. Analyser le temps d'acheminement nous permettra de déterminer lequel des deux protocoles est le mieux adapté aux applications transmettant des données avec des contraintes temps réel. Pour

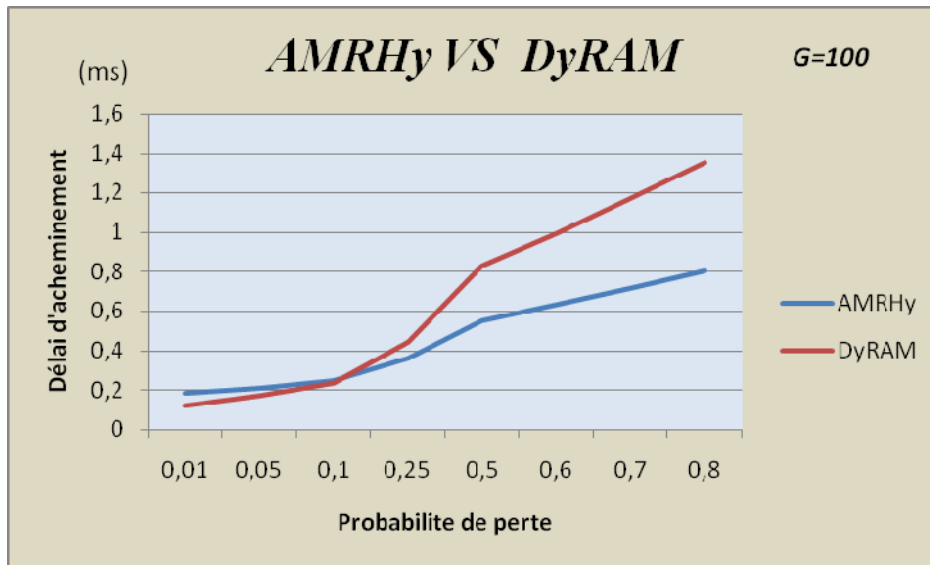


cette étude, le temps nécessaire de traitement des paquets au niveau des composants du réseau est pris en charge, on le désignera par β . Le temps β au niveau des routeurs actifs accroit avec la taille des groupes multicast : plus les routeurs sont sollicités pour le traitement des paquets de contrôle plus le temps de traitement augmente. Aussi, notre étude sera effectuée de façon à mettre en évidence l'effet de la probabilité de perte et l'effet de la taille que prend les groupes multicast sur le délai d'acheminement des deux protocoles AMRHy et DyRAM.

A. Analyse du délai d'acheminement des protocoles en fonction du taux de perte

Dans cette partie, on étudiera le délai nécessaire aux deux protocoles pour l'acheminement fiable des paquets de données en fonction de la probabilité de perte relatif au réseau de transmission. Pour notre étude, les valeurs de temps de transfert sont fixées comme suit :

- Le temps de transfert d'un paquet de la source au routeur actif descendant est de 0,02 ms.
Source $\xrightarrow{0,02}$ Routeur actif
- Le temps de transfert d'un paquet d'un routeur actif au routeur actif descendant est de 0,05 ms.
Routeur actif $\xrightarrow{0,05}$ Routeur actif
- Le temps de transfert d'un paquet d'un routeur actif à un récepteur descendant est de 0,05 ms.
Routeur actif $\xrightarrow{0,05}$ Récepteur
- La taille des groupes multicast est de 100 récepteurs ($G=100$).

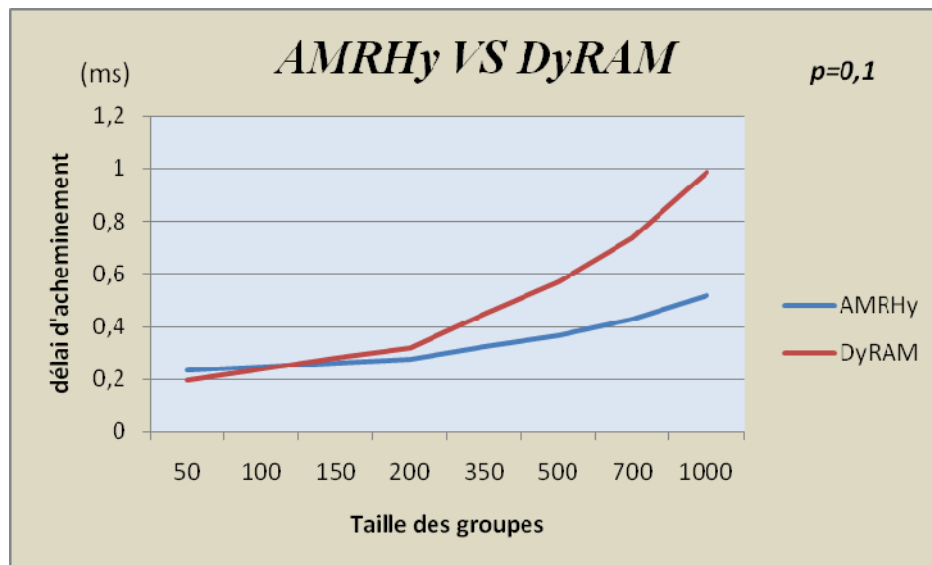


Graph 1.8 Graphe comparatif du délai d'acheminement des protocoles AMRHy et DyRAM en fonction de la probabilité de perte pour des groupes multicast de taille $G=100$

Grace à ce graphe, on constate que pour des probabilités de perte assez faible le protocole DyRAM permet un acheminement des paquets de données plus rapide que le protocole AMRHy. Pour une probabilité raisonnable et commune ($p=0,1$) les deux protocoles acheminent les paquets de données en un temps similaire. Une fois encore, le protocole AMRHy permet de meilleures performances que le protocole DyRAM pour des réseaux peu fiables.

B. Analyse du délai d'acheminement des protocoles en fonction de la taille des groupes multicast

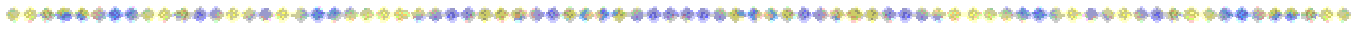
Après avoir étudié l'effet de la probabilité de perte sur les deux protocoles, on présente une comparaison du comportement des deux protocoles AMRHy et DyRAM en fonction de la taille des groupes multicast pris en charge. On fixe la probabilité de perte à $p=0,1$.



Graphe 1.9 Graphe comparatif du délai d'acheminement des protocoles AMRHy et DyRAM en fonction de la taille des groupes multicast pour une probabilité de perte $p=0,1$

Similairement, le protocole AMRHy présente un délai d'acheminement raisonnable vis-à-vis du déploiement des groupes multicast par rapport au protocole DyRAM. Plus la taille des groupes multicast concernés par la transmission est grande plus l'écart qui sépare le délai d'acheminement des deux protocoles augmente. Grâce à ce graphe, il est clair que le protocole DyRAM présente des défaillances de performances face aux réseaux de transmission à forte densité.

V.4. Récapitulatif de l'ensemble de l'analyse accomplie



Cette étude a permis de mettre en évidence les points positifs du protocole AMRHy mais aussi de détecter quelques défaillances relatives au comportement global du protocole.

Les résultats obtenus montrent, entre autre, que :

- 1) Le protocole AMRHy présente des performances très favorables dans un environnement à forte probabilité de perte et à densité élevée. Ceci est une victoire qu'emporte le protocole AMRHy sur un nombre important de protocoles renommés qui n'ont pas pu faire face à de telles contraintes.
- 2) Le protocole AMRHy, bien qu'il présente de meilleures performances pour des réseaux peu fiables, il offre des performances dans les normes pour les réseaux courants. Les différences obtenues pour une densité réduite des groupes multicast et pour des réseaux à faible probabilité de perte ne sont pas importantes et représentent quand même un bon résultat.
- 3) Le protocole AMRHy a l'avantage d'être un protocole facile à implémenter. Il ne procède pas à des traitements complexes au niveau des composants du réseau et sa compréhension est facile et instinctive contrairement au protocole DyRAM.

Durant la simulation, le protocole AMRHy a présenté quelques failles dans sa conception. On y a remédié très facilement comme cela est expliqué dans la section suivante. Les résultats présentés précédemment sont obtenus après correction du protocole AMRHy.

VI. Les améliorations apportées au protocole AMRHy

La simulation NS2 du protocole AMRHy a permis de détecter des anomalies dans le fonctionnement du protocole. Ces anomalies n'influencent pas beaucoup sur le comportement du protocole mais peuvent réduire les performances espérées lors de la conception du protocole AMRHy.

- A. La première anomalie détectée est le fait que lors de la réception d'un acquittement positif provenant d'un père, le routeur actif vérifie si un temporisateur d'attente d'acquittement négatif est actif pour ce paquet. Si c'est le cas, il le désactive et se comporte comme s'il a transmis un acquittement positif à son nœud père. Ce comportement a deux résultats négatifs :
- Le paquet acquitté n'est pas supprimé du buffer.
 - Les récepteurs ayant signalé une perte du paquet ne recevront pas le paquet de réparation et devront le redemander auprès du répondeur élu (perte de temps).

- B. Dans le cas où tous les récepteurs d'un routeur ont perdu un paquet, la détection de l'erreur est retardée jusqu'à l'arrivée d'un acquittement positif concernant ce paquet au niveau des routeurs actifs. Avec le comportement actuelle du protocole AMRHy, la détection ainsi présenté ne s'effectue pas et les paquets perdu ne seront pas réparés ni même détectés.
- C. Durant l'analyse du délai d'acheminement on a remarqué une performance réduite du protocole pour de petits taux de perte. L'explication est que les routeurs AMRHy en recevant un acquittement positif de l'un de leurs récepteurs déclenchent un temps d'attente d'acquittement négatif même si tous les récepteurs ont acquitté positivement le paquet.

Des modifications ont été apportées uniquement sur le comportement des routeurs actifs du protocole, comme cela est présenté dans ce qui suite :

Les routeurs actifs :

✦ 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 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***

*Si délai d'attente non actif **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 multicast ;
- Positionner l'indicateur ACKgénéraler à vraie.
- NBACK =1 ;



Sinon /*délai d'attente actif*/

- Ignorance des ACKs qui arrivent des fils./*agrégation des ACKs au niveau du routeur actif*/
- NBACK ++ ;

Si NBACK égale au nombre de récepteur du groupe ***alors***

* Désactivation du délai d'attente ;

* Libération du cache mémoire ;

Si ACKgénéral est à vraie ***alors*** Envoie d'un ACK au nœud père ***Fsi*** ;

Fsi ;

Fsi ;

Sinon /* l'ACK provient d'un nœud père*/

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 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 multicast ;
- ◆ Enregistrement de l'adresse du répondeur ;

Sinon /* le paquet de données a été reçu */

Si le délai d'attente est actif pour ce paquet de données ***alors***

- Pas d'envoi d'un ACK à nœud père ; /*suppression locale des ACKs*/
- Positionner l'indicateur ACKgénéral à faux ;

Sinon /*le délai d'attente n'est pas actif */

- Pas d'envoi d'un ACK à nœud père ; /*suppression locale des ACKs*/
- Positionner l'indicateur ACKgénéral à faux ;
- Expédition du paquet de données aux fils directs dans l'arbre multicast ;

Fsi ;

Fsi ;



Fsi ;

✦ A la réception d'un NAK :

Si délai d'attente actif *alors*

◆ Ajout de l'adresse du fils à la liste des fils n'ayant pas reçu le paquet de données ;

Fsi ;

✦ 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 ;

Si ACKgénérer est à vraie *alors* Envoie d'un ACK au nœud père *Fsi ;*

VII. Conclusion

La totalité de ce chapitre a été consacré à l'étude comparative des deux protocoles AMRHy et DyRAM grâce à l'environnement de simulation NS2. Les résultats obtenus montrent un comportement exemplaire du protocole AMRHy pour des environnements de transmission très peu fiable et avec une population assez dense. Aussi, grâce à cette simulation, on a pu corriger quelques erreurs de conception du protocole qu'on n'a pas pu percevoir auparavant. En se basant sur les résultats obtenus, on peut conclure que le protocole AMRHy a les potentialités nécessaires pour migrer vers un environnement sans fil. Ces potentialités se résument en :

- La simplicité des traitements.
- Le support des taux de perte élevés.
- La tolérance d'une densité importante des récepteurs.

Conclusion et perspectives

Les transmissions de données à travers les réseaux "*best-effort*" tels que l'internet sont sujettes à une probabilité de perte très élevée. Pour réaliser un recouvrement des pertes avec un minimum de ressources en un temps réduit représente un véritable challenge. Le défi à été relevé par des dizaines de protocoles. Chaque protocole expose des solutions qui pourront remporter le défi. Hélas, au jour d'aujourd'hui, chaque protocole proposé présente des failles qui le rendent inadéquat pour certaines applications et sous certaines contraintes.

Dans ce mémoire, on essaie de classer ces protocoles selon l'approche de recouvrement adoptée. On s'est focalisé sur les protocoles se basant sur un recouvrement local et tout particulièrement sur les protocoles faisant appel aux services fournis par les réseaux actifs. On a présenté les trois protocoles les plus récents à savoir : MAF "Multicast Actif Fiable", AER "Active Error Recovery" et DyRAM "Dynamic Relier Active reliable Multicast". Le but de ce mémoire a été de présenter et d'évaluer les performances du protocole AMRHy "Active Multicast Reliable Hybrid protocol" par rapport à ceux d'un protocole de grande renommée tels que DyRAM. Le choix de ce protocole est dû au fait que ce dernier procède à l'élection d'un répondeur parmi les récepteurs tout comme le protocole AMRHy. Aussi, il a été intéressant de comparer entre leurs performances respectives en terme de besoin en bande passante, d'utilisation de cache mémoire au niveau des routeurs actifs et de délai d'acheminement nécessaire pour délivrer un paquet de données correctement. Une simulation NS2 a été réalisée pour les deux protocoles.

Les résultats obtenus montrent que le protocole DyRAM est plus performant que le protocole AMRHy pour des groupes multicast peu dense et un réseau assez fiable. Cependant, pour des réseaux très peu fiables avec des groupes multicast denses, le protocole AMRHy présente de meilleurs résultats. Grâce à cette simulation, il nous a été, aussi, permis de détecter quelques erreurs de comportement du protocole AMRHy. Une partie des corrections ont été d'ores et déjà apportées au protocole. Le reste des corrections feront partie des travaux futurs. Les perspectives futures engloberont, aussi, le projet d'intégrer le protocole AMRHy à un environnement sans fil. En effet, la simulation du protocole AMRHy reflète sa disposition et sa capacité d'assurer un recouvrement de perte de données au dessus des réseaux sans fil, connues pour leur densité et leurs taux de perte élevés.

Références

- [1]. www.faqs.org/rfcs/rfc1112.html
- [2]. IP-Multicasting Technology Part I: History and Overview; www.intelligraphics.com
- [3]. MBone, Internet à l'oeil; http://ditww.epfl.ch/sic/sa/publications/FI95/sp_95/sp95_19.html
- [4]. www.faqs.org/rfs/rfc3376.html
- [5]. www.faqs.org/rfs/rfc2901.html
- [6]. www.faqs.org/rfs/rfc1075.html
- [7]. www.faqs.org/rfs/rfc1584.html
- [8]. www.faqs.org/rfs/rfc4601.html
- [9]. Les protocoles Multicast; www.crir.univ-avignon.fr/visio/pfe/protocoles/protocoles.htm
- [10]. P .Mane; “WAIT: Selective Loss Recovery For Multimedia Multicast”, A Thesis Submitted to the Faculty of the worcester polytechnic institute, in partial fulfillment of the requirements for the Degree of Master of Science in Computer Science. Octobre 2000
- [11]. B. N. Levine, J. J. Garcia-Luna-Aceves; “A Comparison of Known Classes of Reliable Multicast Protocols”, Computer Engineering Department, University of California, Santa Cruz, CA 95064, USA.(1998)
- [12]. S. Pingali, D. Towsley, J. Kurose; “A Comparison of Sender-Initiated and Receiver-Initiated Reliable Multicast Protocols”, Dept. of Computer Science, University of Massachusetts and Dept. of Electrical and Computer Engineering, University of Massachusetts.(1998)
- [13]. J. C. Lin, S. Paul; "RMTP: A Reliable Multicast Transport Protocol", Department of Computer Sciences Purdue University, West Lafayette, Indiana and AT&T Bell Laboratories, Holmdel, New Jersey. In Proceedings of IEEE INFOCOM '96, March 1996, pp. 1414-1424.
- [14]. T. Speakman, J. Crowcroft & al.; “PGM Reliable Transport Protocol Specification”, Network Working Group, Request for Comments (RFC): 3208, Category: Experimental, December 2001.
- [15]. P. Spathis et K. L. Thai ; "MAF : un Protocole de Multicast Fiable", niversité Pierre et Marie Curie, No 4, place Jussieu, 75005 Paris, France, projet RNRT Amarrage Convention No 01.2.93.0128, (2001).

- [16]. K. L. Yeung, H. T. Wong ; "Caching policy design and cache allocation in active reliable multicast", Dept. Of Electrical and Electronic Engineering, The University of Hong Kong. February 2003.
- [17]. L. Derdouri, D. Saidouni, M. Benmohammed ; "Reliable Multicast Transport in an Active Environment", Lire Laboratory, University Mentouri of Constantine, Algeria.
- [18]. S. Guha, A. Markopoulou, F. Tobagi ; "Hierarchical reliable multicast : Performance evaluation and optimal placement of proxies", CIA Department USA, Harvard Ave USA and Electrical Engineering Département, Stanford University, USA. April 2003.
- [19]. I. Hsuan Huhang ; "Active Networks: An Overview", Department of Computer Science and Engineering, Yuan Ze University, China.(2002)
- [20]. J-P. Gelas ; "Vers la conception d'une architecture de réseaux actifs apte à supporter les débits des réseaux gigabits", Thèse de doctorat. Université Claude Bernard-Lyon I France, December 2003.
- [21]. O. Festor, I. Chrisment, and E. Fleury ; "Les réseaux programmables 1.0". Technical Report 3919, INRIA, March 2000.
- [22]. D. Wetherall, D. Tennenhouse; "The active IP option". Technical report, MIT, ACM SIGOPS European Workshop. September 1996.
- [23]. M. Maimour, C-D. Pham; "DyRAM : an Active Reliable Multicast Framework for Data Distribution". LIP, ENS de Lyon France. Kluwer Academic Publishers. January 2003
- [24]. S.K. Kasera, S. Bhattacharyya, M. Keaton, D. Kiwior, J. Kurose, D. Towsley and S. Zabele; "Scalable Fair Reliable Multicast Using Active Services". Bell Labs, Sprint Labs, Tasc Inc., UMASS Amherst. To appear in IEEE Network Magazine (Special Issue on Multicast), January/February 2000.
- [25]. S. Floyd, V. Jacobson, C-G Liu, S. McCanne, L. Zhang; "A reliable Multicast Framework for light weight sessions and application Level Framing". To appear in IEEE/ACM Transactions on Networking, December 1997.
- [26]. L-w.H. Lehman, S.J. Garland, D.L. Tennenhouse; "Active Reliable Multicast". MIT Laboratory for Computer Science. Research supported in part by DARPA, monitored by the Office of Naval Research, under contract No. N66001-96-C-8522. (1998)
- [27]. D. Waitzman, C. Partridge, S. Deering; "Distance Vector Multicast Routing Protocol". BBN STC, Stanford University, RFC: 1075. November 1988.
- [28]. www.alliedtelesyn.com; Multicasting White Paper
- [29]. T. Speakman, J. Crowcroft, J. Gemmell, D. Farinacci, S. Lin, D. Leshchiner, M. Luby, T. Montgomery, L. Rizzo, A. Tweedly, N. Bhaskar, R. Edmonstone, R. Sumanasekera, L. Vicisano; "PGM Reliable Transport Protocol Specification". Cisco

Systems, UCL, Microsoft, Procket Networks, Juniper Networks, Digital Fountain, Talarian Corporation, University of Pisa. RFC: 3208, December 2001.

- [30]. A. Ballardie; "Core Based Trees (CBT version 2) Multicast Routing". RFC:2189. September 1997.
- [31]. D. Estrin, D. Farinacci, A. Helmy, D. Thaler, S. Deering, M. Handley, V. Jacobson, C. Liu, P. Sharma, L. Wei; "Protocol Independent Multicast-Sparse Mode (PIM-SM): Protocol Specification". RFC: 2362. June 1998.
- [32]. J-P. Gelas; "Vers la conception d'une architecture de réseaux actifs apte à supporter les débits des réseaux gigabits". Thèse de doctorat. Université Claude Bernard-Lyon I-France. Décembre 2003.
- [33]. D. Wetherall, D. Tennenhouse; "The active Ip option". Technique report MIT. ACM SIGOPS European Workshop. September 1996.
- [34]. S. Bhattacharjee, K. Calvert, E. Zegura; "An architecture for active networking". In INFOCOMM 97, April 1997.
- [35]. R.E. Blahut; "Theory and Practice of Error Control Codes". Addison Wesley, MA,1984.
- [36]. L. Rizzo, L. Vicisano; "Effective erasure codes for reliable computer communication protocols". ACM Computer Communication Review, April 1997, vol.27,no.2, p. 24-36.
- [37]. The ns Manual (formerly ns Notes and Documentation).The VINT Project A Collaboration between researchers at UC Berkeley, LBL, USC/ISI, and Xerox PARC. Kevin Fall _ kfall@ee.lbl.gov_, Editor. KannanVaradhan_ kannan@catarina.usc.edu_, Editor. December 13, 2003.

ANNEXE A

A.1 Introduction

Suite à l'expansion impressionnante des réseaux informatiques, un besoin de plus en plus pressant de recherches dans ce domaine s'est fait sentir. Devant la complexité des problèmes concernés et la grande diversité des architectures rencontrées, de nombreuses solutions sont envisageables pour un même problème. Chacune d'elles doit alors être testée, évaluée, caractérisée et critiquée avant d'envisager son implémentation concrète dans un réseau. La simulation est le seul moyen permettant d'effectuer ces tests à moindres coûts.

La présente annexe a pour but de donner une idée sur la manière dont le simulateur Network Simulator [NS.2], élément essentiel dans la réalisation des études précédemment décrites dans le chapitre 4, fonctionne.

A.2 Le simulateur NS

NS (version 2) [29] est un simulateur orienté objet et à événement discret, développé à l'université de Berkeley, écrit en C++ et muni d'un interpréteur Otcl (dérivé du langage Tcl avec une extension orienté objet développé par MIT). Le projet NS fait partie du projet VINT chargé de développer des outils permettant non seulement d'afficher les résultats d'une simulation et d'une analyse, mais aussi de développer des convertisseurs de topologie réseau générés avec d'autres générateurs que NS.

NS est un outil de recherche très utile à la conception et à la compréhension des protocoles. Il est utile aussi bien dans l'étude des protocoles de routages au niveau des réseaux locaux ou à large porté qu'à l'étude des réseaux mobiles ou les communications par satellites. NS permet de simuler une variété de protocoles réseau : TCP et UDP, des générateurs de trafic : FTP, Telnet, Web, CBR et VBR, des protocoles de gestion des files d'attentes : Drop Tail, RED et CBQ, d'algorithmes de routages : Dijkstra, et bien d'autres. Il permet d'implémenter des trafics multicast et quelques protocoles de la couche MAC pour une simulation de LAN.

Cependant, NS est assez facile à utiliser une fois que l'utilisateur ait acquis une certaine maîtrise du simulateur. Mais il est relativement difficile pour un utilisateur non-initié. Cette difficulté est dû au manque de documents dédiés aux usagers non expérimentés et au fait que NS, afin d'introduire de nouveau protocole, est continuellement modifié et qu'ainsi, malheureusement, la documentation n'est pas mise à jour assez souvent.

Le simulateur supporte une hiérarchie de classes au niveau du compilateur C++ [aussi appelé : La hiérarchie compilée], et une autre hiérarchie de classes similaire à la précédente au niveau de l'interpréteur Otcl [aussi appelé : La hiérarchie interprétée].

Les deux hiérarchies sont intimement liées l'une à l'autre. Du point de vue utilisateur, il y a une correspondance une à une entre les différentes classes des deux hiérarchies.

La racine de cette hiérarchie est la classe : Tclobject.

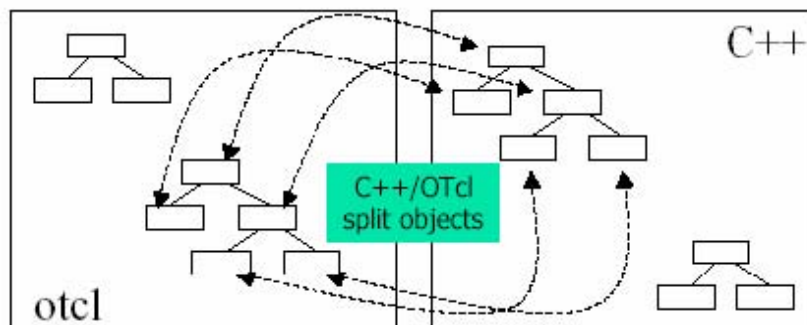


Figure A.1 La dualité entre Otcl et C++

Les utilisateurs créent de nouveaux objets de simulation grâce à l'interpréteur. Ces objets sont instanciés au sein de l'interpréteur (la hiérarchie interprétée) et sont fidèlement reproduits au niveau de la hiérarchie compilée grâce à des méthodes définies dans la classe Tclobject.

La hiérarchie interprétée est automatiquement établie grâce à des méthodes prédéfinies au niveau de la classe Tclclass.

A.3 Pourquoi deux langages ?

NS utilise deux langages car un simulateur est sensé faire deux choses différentes à la fois. D'une part, il doit réaliser une simulation détaillée d'un protocole donné, ce qui exige un langage de programmation capable de manipuler efficacement les bytes, les entêtes des paquets et peut implémenter des algorithmes qui se basent sur un ensemble très grand de données nécessaires à leurs exécutions. Pour ces tâches, le temps d'exécution est primordial mais le temps nécessaire à la mise au point (temps de simulation, de trouver les bugs et de les corriger, recompilation) est moins important.

D'autre part, une large partie des recherches sur les réseaux implique le plus souvent des légers changements dans les paramètres et les configurations et exige une rapide exploration des nombreux scénarios possibles. Dans ce cas là, le temps de l'itération (changement du modèle et ré-exécution) est plus important que le temps d'exécution.

NS rassemble les deux besoins en utilisant les deux langages C++ et Otcl.

C++ est rapide à exécuter mais il est lent lorsqu'on veut apporter des changements au code. Il est préférable de l'utiliser lors de l'implémentation détaillée d'un protocole.

Otcl s'exécute plus lentement mais son code peut être changé très rapidement et de manière interactive ce qui le rend idéal pour la simulation du comportement des différentes configurations.

NS, grâce à TCLCL, fournit un moyen pour construire les objets et les variables et ce en joignant les deux langages.

A.4 Quel langage utilisé et dans quel but ?

Ayant deux langages, la question qui se pose est : Quel langage doit-on utiliser et dans quel contexte l'utiliser ?

Il est conseillé d'utiliser l'Otcl pour les cas suivants :

- Pour la configuration et l'installation.
- Si l'utilisation des objets C++ déjà existant satisfait les exigences de votre travail il est conseillé d'utiliser Otcl.

Et il est conseillé d'utilisé C++ :

- Si votre travail exige le traitement séparé de chaque paquet d'un flux.
- Si vous voulez apporter des changements au comportement d'une classe C++ déjà existante et cela en ajoutant la prise en compte de situations non prévues.

Par exemple, les liens sont des objets Otcl qui regroupent : Les délais de transmission, la politique adoptée pour la gestion de la file d'attente et la probabilité de perte des paquets. Si votre expérimentation peut être réalisée à partir de cette structure vous n'aurez pas à déclarer un nouvel objet. Vous aurez à le définir grâce au C++ si au contraire, vous avez besoin d'une structure plus détaillé des liens.

A.5 Réaliser une simulation avec NS

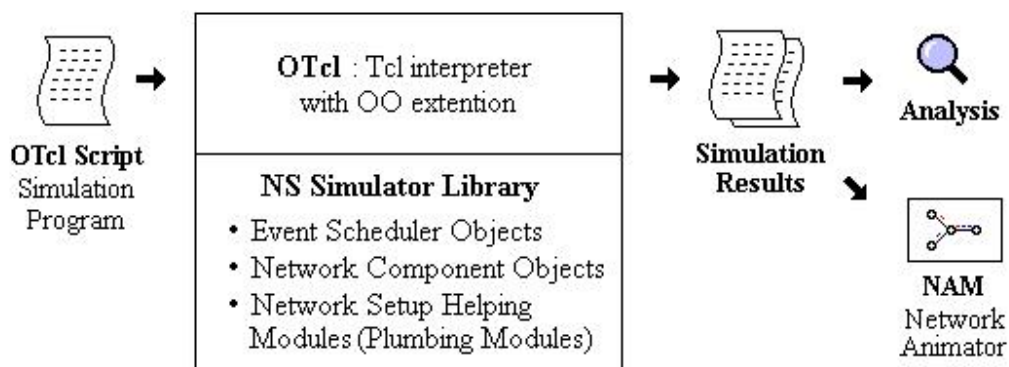


Figure A.2 Schéma des étapes nécessaire à l'utilisation de NS.

La **Figure A.2** présente les étapes d'une simulation réalisée à l'aide de NS. La toute première étape est la création d'un fichier d'extension ".tcl" qui contient la topologie du réseau concerné par la simulation.

La forme générale d'un programme de simulation Otcl est définie comme suite :

- ⊕ Créer une nouvelle instance du simulateur.
- ⊕ Créer les nœuds du réseau.
- ⊕ Créer les liens entre les différents nœuds en indiquant leurs caractéristiques (capacité, délai, politique de la file d'attente,.....).
- ⊕ Définir le type de routage à adopter.
- ⊕ La gestion du trafic sur le réseau nécessite la définition d'agents.
 - ⊕ Définir des agents de protocole de transport et les rattacher aux nœuds.
 - ⊕ Définir des agents de réception et les rattacher aux nœuds destinataires.
 - ⊕ Créer une source de trafic et la rattacher aux nœuds sources.
- ⊕ Définir l'instant de début et de la fin de simulation.
- ⊕ Lancer une procédure de terminaison préalablement déclarée dans le programme.

- ⊕ Créer une nouvelle instance du simulateur.
- ⊕ Créer les nœuds du réseau.
- ⊕ Créer les liens entre les différents nœuds en indiquant leurs caractéristiques (capacité, délai, politique de la file d'attente,.....).
- ⊕ Définir le type de routage à adopter.
- ⊕ La gestion du trafic sur le réseau nécessite la définition d'agents.
 - ⊕ Définir des agents de protocole de transport et les rattacher aux nœuds.
 - ⊕ Définir des agents de réception et les rattacher aux nœuds destinataires.
 - ⊕ Créer une source de trafic et la rattacher aux nœuds sources.
- ⊕ Définir l'instant de début et de la fin de simulation.
- ⊕ Lancer une procédure de terminaison préalablement déclarée dans le programme.

La procédure de terminaison est responsable des traitements post-opérateurs appliqués aux fichiers NS de sortie contenant tous les événements de la simulation.

Il faut noter que les agents utilisés dans le programme Otcl sont supposés préalablement définis au niveau de la bibliothèque d'objets NS, écrits en C++, respectant bien évidemment la dualité "OTCL-C++". Une fois le programme Otcl écrit, il est interprété grâce à l'interpréteur Otcl. Les résultats d'une simulation sont contenus dans deux types de fichiers : Le fichier NAM d'extension "*.nam" et le fichier trace d'extension "*.tr". Le premier fichier permet de visualiser la simulation grâce à l'interface graphique NAM et le second est utilisé par l'outil graphique XGRAPH pour analyser les résultats de la simulation.

A.6 Tool Command Language (TCL)

Tcl “Tool Command Language”, prononcé “tickle”, est un langage interprété permettant le contrôle et l’extension d’applications écrites en C++. Il fournit un système de programmation essentiel au développement et à l’utilisation des applications à interface graphique. Il est pourvu de nombreuses structures de programmation, comme les variables, les structures de boucles et les procédures. Aussi Tcl est facile à incorporer car son interpréteur est implémenté comme étant une bibliothèque de procédure C. Cela lui permet d’être incorporé facilement dans n’importe quelle application. Cette application peut même modifier le contenu des dispositifs Tcl en ajoutant des commandes supplémentaires qui lui sont spécifiques. L’Otel est l’extension du langage Tcl. Il introduit les différentes notions liées à la programmation orientée objet telles que la notion de classe, sous-classe et la notion d’héritage. Pour plus d’information sur le langage Tcl [30].

A.7 Le planificateur d’événements

Le planificateur d’événements [31] est l’élément principal du simulateur. Il est implémenté selon la structure d’une liste chaînée des événements qui doivent se produire. Ces événements sont de plusieurs types : Expiration d’un timer, arrivè d’un paquet, démarrage d’une source,...etc. ou simplement un appel de procédures.

Voici la définition d’un événement :

```
Class Event{  
    Public :  
    Event* next_;          /*event list*/  
    Handler* handler_;    /*handler to call when event ready*/  
    Double time_;        /*time at which event is ready*/  
    Int uid_;            /*unique ID*/  
    Event() : time_(0),uid_(0) {}  
};
```

Un événement possède donc quatre éléments: Un pointeur vers le prochain événement, un pointeur vers l’objet responsable du traitement de l’événement, l’instant où l’événement se produit et un identificateur unique pour chaque événement. Les principaux utilisateurs du planificateur d’événements sont les composants du réseau (nœuds, liaisons, files d’attente,...etc.) qui désirent simuler les délais de manipulation ou tout ce qui nécessite une synchronisation.

Le déroulement du programme consiste à prendre et à traiter systématiquement l’élément suivant dans la liste du planificateur.

Une des plus importantes utilisations du planificateur est la planification d’événements tels que l’activation d’un générateur de trafic, le lancement d’une procédure à un moment précis de la simulation ou encore l’évènement de fin de simulation. De tels événements appartiennent à la

classe "AtEvent", sous-classe de la classe "Event". Cette sous-classe dispose d'un champ supplémentaire permettant de contenir une chaîne de caractères passée en argument. Celle ci désigne le nom de l'événement à traiter. Ces événements sont traités de la même façon que ceux appartenant à la classe parente. Cependant, lorsque cet événement arrive en tête de liste, le traitement effectué sur lui consiste en l'exécution de la commande dont le nom est spécifié par la chaîne de caractères.

```

set ns [new Simulator]      ;# instantiation d'un objet Simulator
proc finish {}              ;#Déclaration d'une procédure
{
..                             ;# corps de la procédure
..
}
..
..
$ns at 10.5 "finish"      ;# appel de la procédure déclaré grâce
                           ;# au planificateur d'événements

```

A.8 Les composants réseau dans un programme Otcl

A.8.1 Les nœuds

Le nœud [31] est un des premiers éléments de la description de la topologie d'un réseau sous NS. Le nœud est composé de deux éléments : le classificateur d'adresses et le classificateur de ports. Le rôle des classificateurs est d'acheminer les paquets sur la bonne liaison (Addr Classifier) ou sur le bon port ⁽¹⁾ (Port Classifier). La classification des paquets se fait sur la base de l'adresse de destination contenue dans le paquet. Cette adresse contient un champ qui indique le nœud destinataire et un autre champ qui indique l'agent destinataire.

Il existe deux types de nœud : les nœuds unicast et les nœuds multicast. La structure des nœuds multicast diffère de celle des nœuds unicast par l'existence d'un duplicateur dont la fonction est de dupliquer les paquets entrants pour les acheminer vers plusieurs destinations.

(1) Chaque port est lié à un agent déclaré au niveau du nœud.

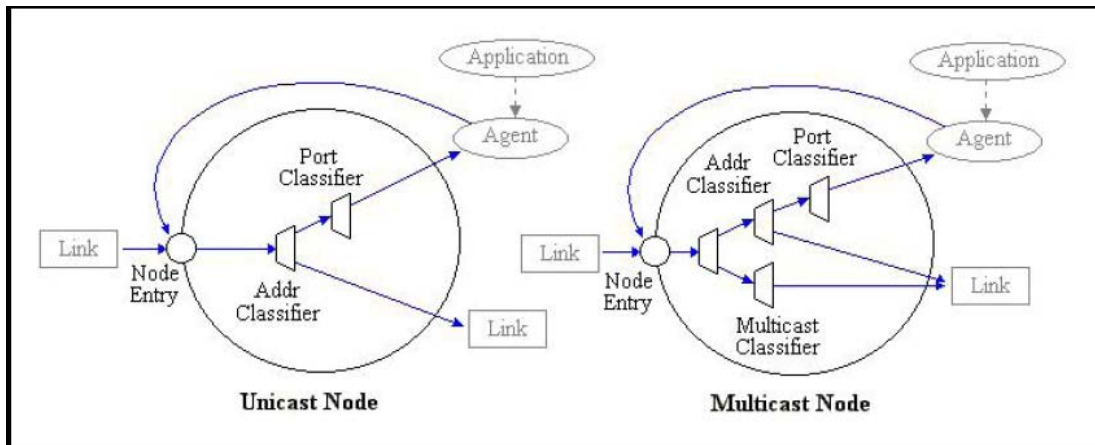


Figure A.3 Structures des nœuds

La déclaration d'un nœud dans un programme Tcl est comme suite :

```
set <nom_du_nœud> [ $<objet_simulator> node ]
```

Les nœuds sont par défaut des nœuds unicast. Pour créer un nœud multicast il faut déclarer son type explicitement au niveau du programme Otcl juste après l'avoir déclaré lui-même. Tous les nœuds du réseau sont ainsi déclarés comme des nœuds multicast.

A.8.2 les liens

Les liens [31] sont considérés comme des composants importants dans la topologie d'un réseau. Ils servent à connecter les nœuds entre eux. Il existe deux types de lien : *simplex_link* et *duplex_link*. Les liens *duplex_link* sont formés à partir de deux liens *simplex_link* dans les deux directions. Un lien est composé d'un ensemble de connecteurs.

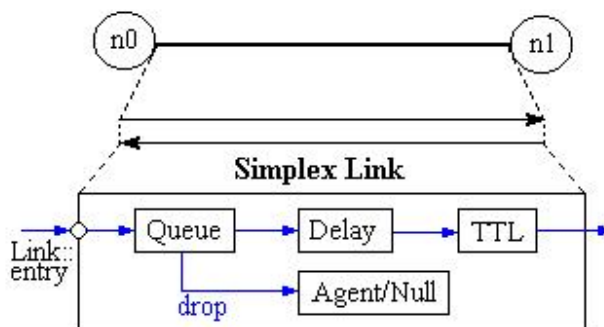


Figure A.4 La structure d'un lien (1).

Chaque connecteur a un rôle bien précis :

- **Queue** : Fait référence à la file d’attente principale du lien. Tout paquet en attente est envoyé à la file d’attente qui suit une politique d’élimination déterminée par l’usager.
- **Delay** : Fait référence à l’objet qui modélise les caractéristiques du lien en termes de délai et de bande passante.
- **Agent/Null** : Fait référence à l’objet qui traite les paquets perdus. Les paquets supprimés de la file d’attente lui sont envoyés pour être libérer.
- **TTL** : Fait référence à l’objet qui calcule le TTL de chaque paquet reçu et met à jour le champ TTL du paquet.

Dans le but de garder la trace des paquets transitant par un lien, on ajoute des connecteurs supplémentaires.

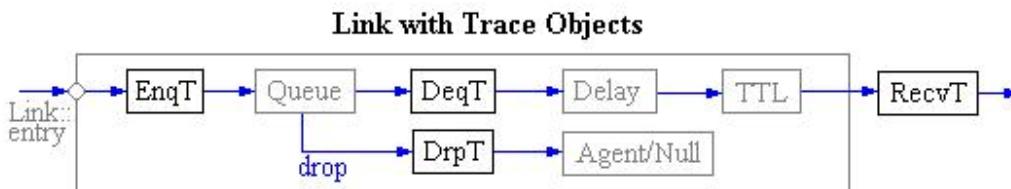


Figure A.5 La structure d’un lien

- **EnqT, DeqT, DrpT, RecvT** : Sont des objets qui permettent de garder la trace des paquets transitant par le lien. **EnqT** sauvegarde le nombre de paquets entrant dans le lien ; **DrpT** comptabilise le nombre de paquets perdus; **RecvT** donne le nombre de paquets reçus par l’élément de sortie.

Les paquets [31] utilisés dans une simulation NS sont formés d’un empilement d’entêtes et d’un pointeur vers des données. Chaque paquet comporte tous les entêtes déclarés au sein de NS, même si elles ne sont pas toutes nécessaires. Cette technique entraîne un gaspillage de la mémoire mais simplifie l’utilisation de l’entête.

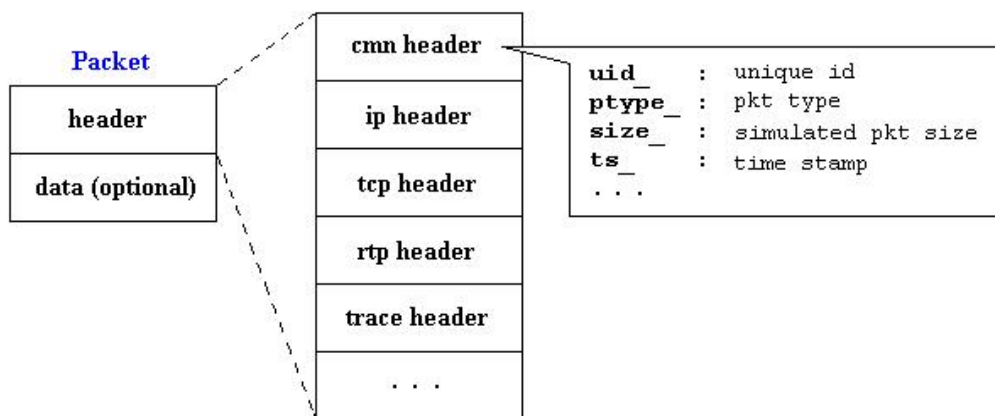


Figure A.6 La structure des paquets

A.9 Comment créé un nouvel agent ?

La structure des répertoires NS est donnée dans la **Figure A.7**.

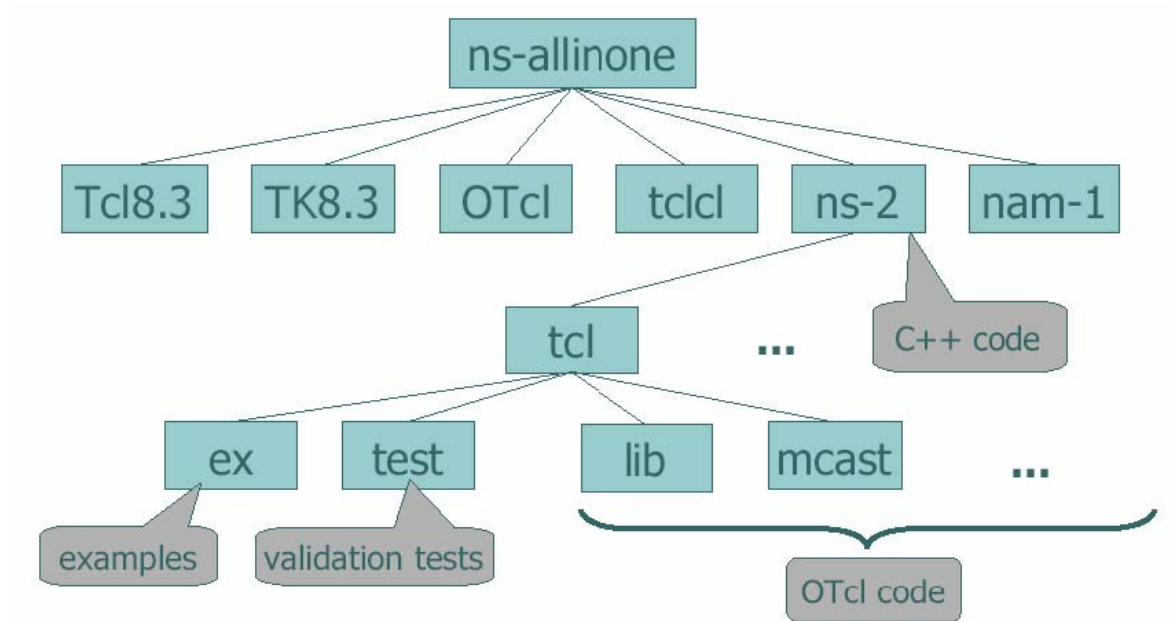


Figure A.7 La structure des répertoires de NS

La création d'un nouvel agent [32] est considérée comme une extension de NS au niveau du code C++. Cette extension peut s'effectuer de deux manières :

1. En modifiant le code d'agent déjà existant. Par la suite, pour pouvoir utiliser ces agents modifiés, il est nécessaire de recompiler NS en exécutant deux commandes : `make depend` et `make ns`.
2. En ajoutant votre propre agent. Là, il faut mettre à jour le fichier `makefile` ensuite recompiler NS pour rendre votre agent fonctionnel.

On se concentrera sur la deuxième manière. Les étapes de création d'un agent sont les suivantes :

- ✦ Déterminer la position hiérarchique de notre classe d'agent : à partir de quel classe notre agent est dérivé.
- ✦ Créer l'entête des paquets propres à notre agent si cela est nécessaire : il est possible d'utiliser l'entête des paquets d'un autre agent si cela satisfait notre agent.
- ✦ Créer la classe C++ de notre agent et définir ces différentes procédures.
- ✦ Définir les liens avec l'Otcl, s'ils existent.

- ✦ Déterminer la position hiérarchique de notre classe d'agent : à partir de quel classe notre agent est dérivé.
- ✦ Créer l'entête des paquets propres à notre agent si cela est nécessaire : il est possible d'utiliser l'entête des paquets d'un autre agent si cela satisfait notre agent.
- ✦ Créer la classe C++ de notre agent et définir ces différentes procédures.
- ✦ Définir les liens avec l'Otcl, s'ils existent.

La création d'une nouvelle entête à des étapes à suivre :

- ✦ Créer la structure de la nouvelle entête.
- ✦ Implanter la nouvelle entête au sein de NS pour qu'elle puisse être identifiée lors de l'utilisation.
- ✦ Créer une classe statique nécessaire aux liaisons Otcl (packet.h).
- ✦ Rendre l'entête fonctionnelle au niveau Otcl (tcl/lib/nspacket.tcl).

- ✦ Créer la structure de la nouvelle entête.
- ✦ Implanter la nouvelle entête au sein de NS pour qu'elle puisse être identifiée lors de l'utilisation.
- ✦ Créer une classe statique nécessaire aux liaisons Otcl (packet.h).
- ✦ Rendre l'entête fonctionnelle au niveau Otcl (tcl/lib/nspacket.tcl).

A.10 Post Simulation

A.10.1 Xgraph

Le Xgraph [33] fait partie de NS. C'est un programme de traçage. Il permet de créer une représentation graphique des résultats d'une simulation. Il traite les fichiers d'extension "*.tr".

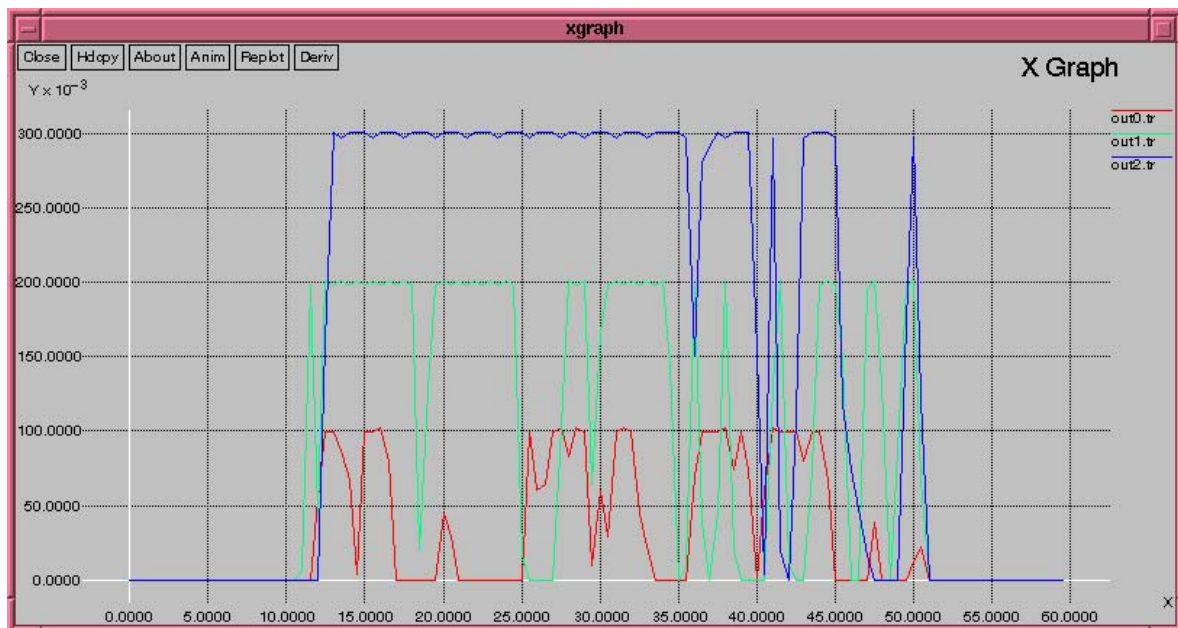


Figure A.8 l'interface de Xgraph.

L'exécution du programme Xgraph est réalisée grâce à la commande Otcl suivante :

```
exec xgraph <file_name.tr> -geometry 800x400 &
```

800x400 permet d'adapter la taille de la fenêtre xgraph à la taille de l'écran.

A.10.2 Network Animator

NAM (Network Animator) [34] est un outil d'animation graphique basé sur tcl/tk. Il permet de visualiser le déroulement d'une simulation en affichant la topologie du réseau et les déplacements des paquets durant la simulation.

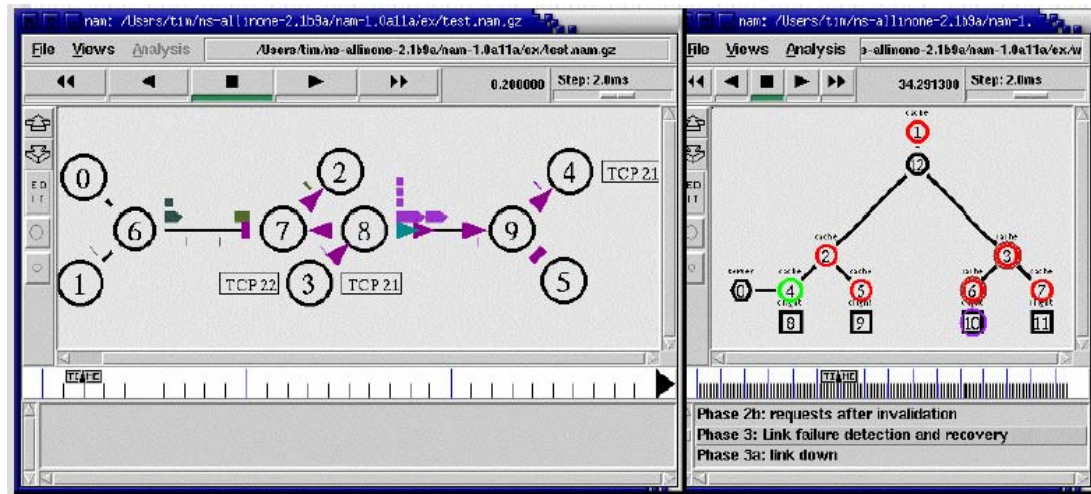


Figure A.9 L'interface de NAM

La création d'un fichier "*.nam" et l'exécution de NAM est possible grâce aux commandes suivantes :

set nf [open out.nam w] ;# ouverture d'un fichier "out.nam" en écriture désigné par la variable nf

\$ns namtrace-all \$nf ;# permet de sauvegarder les événements de la simulation dans le fichier nf

```

proc finish {} { ;# déclaration d'une procédure de fin de simulation
  global ns nf ;# indique que les variables ns et nf sont des variables globales
  $ns flush-trace ;# fixe les événements de la simulation dans le fichier nf
  close $nf ;# fermeture du fichier out.nam
  exec nam out.nam & ;# lecture du fichier out.nam en exécutant l'interface graphique
  exit 0
}

```

Résumé

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 l'exploitation des services actifs fournis par les routeurs pour élaborer des protocoles multicast fiables qui s'appliquent à grande échelle offrant ainsi une solution plus générale et plus flexible. Notre travail consiste en l'étude et l'analyse des performances de deux protocoles multicast fiables basés sur les services actifs : AMRHy et DyRAM. L'analyse de performance va porter sur trois métriques : la bande passante, le délai d'acheminement et la taille du cache mémoire au niveau des routeurs actifs. Cette analyse va nous permettre de montrer l'intérêt de la combinaison des classes basées sur l'initiative de la source "sender-initiated" et des classes basées sur l'initiative des récepteurs "receiver-initiated" adoptée par AMRHy par rapport à la classe basée sur l'initiative des récepteurs "receiver-initiated" uniquement adoptée par DyRAM. Dans la classe des protocoles de recouvrement basée sur l'initiative des récepteurs "receiver-initiated" la détection des pertes est attribuée aux récepteurs indépendamment du lien sur lequel la perte se produit. Par contre dans la combinaison des classes, le lien sur lequel se produit la perte est pris en compte ; la source détecte les pertes qui se produisent sur les liens source et les récepteurs détectent celles qui se produisent sur les liens terminaux. Pour ce fait, les deux protocoles AMRHy et DyRAM ont été implémentés dans un environnement de simulation NS2. Le choix des deux protocoles est motivé du fait qu'ils se basent sur l'élection d'un répondeur parmi les récepteurs pour assurer le recouvrement des pertes localement. Les résultats numériques montrent que la combinaison des classes améliore de manière significative le délai d'acheminement, limite la largeur de bande requise par les paquets de contrôle et la taille du cache au niveau des routeurs actifs. Cette amélioration augmente en augmentant la taille du réseau et la probabilité de perte.

Mots clés : *Multicast, Fiabilité, Les réseaux actifs, La bande passante, Le cache mémoire, Le délai d'acheminement, NS2.*

Abstract

To provide a reliable and efficient multicast deliverance for applications with scattering data on a large scale is a challenge. Several protocols were worked out to resolve the problem of reliability in networks said best deliverance effort such as the Internet. They handled this problem by imposing a compromise between the capacity of the bandwidth and the delay of transportation. However, the solutions proposed by these protocols limit themselves to a reduced scale. The introduction of the concept of active routers in the reliability multicast has orientated researches to work on the exploitation of the services provided by the routers to elaborate reliable multicast protocols which apply on a large scale giving so a more general and more flexible solution. Our work consists of a study and analysis of performances of two reliable multicast protocols based on the active services: AMRH_y and DyRAM. The analysis of performance is going to concern three metrics: the bandwidth, delay of transmission and the size of the cache memory at the level of the active routers. This analysis is going to allow showing the interest of the combination of the classes “sender-initiated” and “receiver-initiated” adopted by AMRH_y with regard to the “receiver-initiated” class adopted by DyRAM. In the receiver-initiated class the detection of loss is allocated to the receivers independently of the link on which the loss occurs. On the other hand in the combination of the two classes, the link on which occurs the lost is taken into account. The source detects the losses which occur on the links source and the receivers detect those who occur on the terminal links. For this fact, both protocols AMRH_y and DyRAM were implemented in an environment of simulation NS2. The choice of these two protocols is motivated due to the fact that they are based on the election of an answering machine among the receivers to insure the local covering of the losses. The numerical results show that the combination of the classes improves in a significant way the delay of transmission, limits the bandwidth required by the packages of control packets and reduce the size of the cache memory needed at the level of the active routers. This improvement increases by increasing the size of the network and the probability of loss.

Keywords: Multicasting, Reliability, The active network, The bandwidth, The cache memory, The delivery delay, NS2.

Analyses \ Protocoles	Facteur	AMRHy	DyRAM
<p style="text-align: center;">La bande passante</p>	<p style="text-align: center;">Probabilité de perte</p>	<ul style="list-style-type: none"> • Une performance minimale pour des taux de perte réduits. • Une utilisation de bande passante raisonnable pour des taux de perte élevés. • L'utilisation de bande passante est quasiment insensible (ou très peu) au niveau de fiabilité du réseau d'implémentation. 	<ul style="list-style-type: none"> • Une utilisation réduite de la bande passante pour des taux de perte réduits. • Une utilisation inconsidérée de la bande passante pour des taux de perte élevés. • L'utilisation de la bande passante est très sensible à la défaillance du réseau d'implémentation en termes de fiabilité.
	<p style="text-align: center;">Taille des groupes multicast</p>	<ul style="list-style-type: none"> • Pour des groupes multicast réduits, l'utilisation en bande passante est raisonnable. • Pour des groupes de grande taille, l'utilisation est élevée. Elle se réduit pour des réseaux peu fiables. • L'utilisation de la bande passante est sensible à la taille du groupe multicast. L'application du protocole au dessus d'un réseau peu fiable réduira sa nécessité en bande passante 	<ul style="list-style-type: none"> • Peu de bande passante est exploitée pour des groupes multicast réduits. • Pour des groupes de grande taille, l'utilisation augmente très légèrement. • Le protocole présente une bonne exploitation de la bande passante pour un taux de perte inférieur à 50%
<p style="text-align: center;">La mémoire cache</p>	<p style="text-align: center;">Probabilité de perte</p>	<ul style="list-style-type: none"> • La nécessite en mémoire cache est estimé a 60% du nombre de paquet de données transmit dans le réseau. • Pour des taux de perte allant jusqu'à 10%, le cache mémoire nécessaire est très proche de celui nécessaire au protocole DyRAM. • Le cache mémoire utile pour des taux de perte élevé est très satisfaisant. • Le protocole est efficace quelque soit la nature du réseau utilisé. 	<ul style="list-style-type: none"> • Le besoin en mémoire cache pour des taux de perte réduit est raisonnable. • Pour des taux de perte élevée, une très grande capacité en mémoire cache est nécessaire a un recouvrement efficace. • Le protocole est efficace pour des réseaux ayant un taux de perte réduit.

	<p style="text-align: center;">Taille des groupes multicast</p>	<ul style="list-style-type: none"> • Pour des tailles de groupe réduites, le cache mémoire nécessaire est un peu plus important que celui nécessaire au protocole DyRAM. • Le cache mémoire utile pour des groupes multicast important est constant et très satisfaisant. • Le protocole est efficace quelque soit la taille des groupes multicast pris en charge. 	<ul style="list-style-type: none"> • Le besoin en mémoire cache pour des groupes réduit est très satisfaisant. • Pour des tailles de groupe plus importantes, une très grande capacité en mémoire cache est nécessaire à un recouvrement efficace. • Le protocole est efficace pour la prise en charge de groupe multicast de petite taille.
<p style="text-align: center;">Le délai d'acheminement</p>	<p style="text-align: center;">Probabilité de perte</p>	<ul style="list-style-type: none"> • Pour des probabilités de perte inférieure a 10%, le délai d'acheminement des paquets de données est un peu plus important que celui nécessaire au protocole DyRAM. • Le délai utile à l'acheminement fiable des paquets de données pour des taux de perte raisonnable ($p=0,1$) est très satisfaisant et approche celui du protocole DyRAM. • Pour une probabilité de perte élevée, le protocole achemine les données dans des temps respectables. • Le protocole s'adapte parfaitement aux réseaux peu fiables. 	<ul style="list-style-type: none"> • Le délai d'acheminement pour des probabilités de perte réduit est très satisfaisant. • Pour des taux de perte plus importants, l'acheminement des paquets de données prend un temps trop conséquent. • Le protocole est inefficace pour les réseaux à forte probabilité de perte. ayant une faible probabilité de perte. la prise en charge de groupe multicast de petite taille.
	<p style="text-align: center;">Taille des groupes multicast</p>	<ul style="list-style-type: none"> • Pour des tailles de groupe réduites (<100), le délai d'acheminement nécessaire est un peu plus important que celui du protocole DyRAM. • Le délai utile à l'acheminement pour des groupes multicast important est assez réalisable. • Le protocole est efficace quelque soit la taille des groupes multicast pris en charge. 	<ul style="list-style-type: none"> • Le délai nécessaire pour des groupes réduit est très intéressant. • Pour des tailles de groupe plus importantes, un trop grand délai d'acheminement est nécessaire à un recouvrement efficace. • Le protocole est efficace uniquement en la prise en charge des groupes de petite taille.

Table des figures

Fig.1.1 : La transmission en mode multicast.....	05
Fig.1.2 : Gestion des groupes multicast avec IGMP	06
Fig.1.3 : Schéma d'adressage de l'IP multicast.....	06
Fig.1.4 : Acheminement des paquets multicast « Stations tunnels ».....	07
Fig.1.5 : Mode dense vs Mode épars.....	11
Fig.1.6 : Le model fonctionnel des réseaux actifs.....	13
Fig.1.7 : Architecture fonctionnelle d'un nœud actif.....	14
Fig.2.1 : Classification des protocoles multicast fiables.....	23
Fig.3.1 : Paquet de donnée DyRAM.....	37
Fig.3.2 : Les acquittements négatifs dans DyRAM.....	38
Fig.4.1 : Positionnement du protocole AMRHy par rapport aux protocoles RM.....	52
Fig.4.2 : Tableau récapitulatif et comparatif des protocoles de fiabilité multicast présentés.....	53
Fig.4.3 : Structure des paquets de données du protocole AMRHy.....	57
Fig.4.4 : Structure des paquets de contrôle du protocole AMRHy.....	58
Fig.5.1 : Topologie simple à un seul niveau.....	67
Fig.5.2 : Topologie simple à deux niveaux.....	67
Graphe 1.1 : Graphe comparatif des besoins en bande passante d'AMRHy et DyRAM en fonction de la probabilité de perte pour des groupes de $G=100$	70
Graphe 1.2 : Graphe comparatif des besoins en bande passante d'AMRHy et DyRAM en fonction de la taille des groupes pour une probabilité de perte $p=0,1$	71
Graphe 1.3 : Graphe comparatif des besoins en bande passante d'AMRHy et DyRAM en fonction de la taille des groupes pour une probabilité de perte $p=0,5$	71
Graphe 1.4 : Graphe comparatif des besoins en bande passante d'AMRHy et DyRAM en fonction de la taille des groupes pour une probabilité de perte $p=0,6$;.....	72

Graphe 1.5 : Graphe représentatif du besoin en mémoire cache du protocole AMRHy en fonction du nombre de paquets à transmettre pour une probabilité de perte $p=0,1$	74
Graphe 1.6 : Graphe comparatif des besoins en mémoire cache des protocoles AMRHy et DyRAM en fonction de la probabilité de perte du réseau pour des groupe multicast $G=30$	75
Graphe 1.7 : Graphe comparatif des besoins en mémoire cache des protocoles AMRHy et DyRAM en fonction de la taille des groupes multicast pour une probabilité de perte $p=0,1$	76
Graphe 1.8 : Graphe comparatif du délai d'acheminement des protocoles AMRHy et DyRAM en fonction de la probabilité de perte pour des groupe multicast de taille $G=100$	78
Graphe 1.9 : Graphe comparatif du délai d'acheminement des protocoles AMRHy et DyRAM en fonction de la taille des groupes multicast pour une probabilité de perte $p=0,1$	79