N° d'ordre: 3021

THÈSE

Présentée devant

devant l'université de Rennes 1

pour obtenir

le grade de : DOCTEUR DE L'UNIVERSITÉ DE RENNES 1
Mention INFORMATIQUE

par

Ali BOUDANI

Équipe d'accueil : ARMOR
École doctorale : MATISSE
Composante universitaire : IFSIC/IRISA

Titre de la thèse :

Routage multicast : gestion des petits groupes et ingénierie de trafic

soutenue le 28 juin 2004 devant la commission d'examen

M. :	Gerardo	RUBINO	Président
MM. :	Jean-Jacques	PANSIOT	Rapporteurs
	Samir	TOHMÉ	
MM. :	Jean-Marie	BONNIN	Examinateurs
	Dirk	OOMS	
	Vincent	ROCA	
	Bernard	COUSIN	

La moitié de ce que je te dis est dénué de sens,
mais je le dis afin que l'autre moitié puisse t'atteindre

Gibran Khalil Gibran, *Le sable et l'écume*

Remerciements

C’est avec la crainte d’oublier quelqu’un que j’écris cette partie consacrée à ceux que je souhaite remercier.

Je tiens tout d’abord à remercier Gerardo Rubino qui a entamé au Liban, lors du DEA commun à l’université de Rennes 1, de l’université libanaise et de l’université saint joseph sous l’égide de l’agence universitaire de la francophonie, les négociations (pas facile parfois) de mon accueil et par la suite m’a accueilli au sein du Projet Armor de l’IRISA de Rennes et a présidé mon jury.

C’est grâce à Bernard Cousin que j’ai eu la chance de découvrir l’ingénierie de trafic *multicast* lors d’un stage en 2000. Il m’a de nouveau accompagné et dirigé tout au long des recherches exposées dans cette thèse. Je tiens à l’en remercier très vivement.

J’aimerais aussi remercier Jean-Jacques Pansiot et Samir Tohmé qui ont accepté d’être les rapporteurs de mes travaux et qui m’ont aidé à améliorer la qualité de ce manuscrit. Je remercie également Alexandre Guitton pour son aide et ses nombreux conseils, et qui ont contribué à la progression de mes travaux.

Je les remercie également, ainsi que Jean-Marie Bonnin, Vincent Roca et Dirk Ooms d’avoir accepté de faire partie de mon jury.

Je tiens à remercier les autres membres du projet Armor de l’IRISA de Rennes. Je tiens à ne pas oublier les doctorants de mon équipe avec qui j’ai passé de nombreux bon moments ces trois dernières années.

Je tiens à remercier aussi la France, pays accueillant, généreux, pays de liberté d’expression, de droit, de fraternité, d’égalité et surtout les personnels administratifs du CROUS de Rennes, les personnels de l’ambassade de France au Liban et l’INRIA

de Rennes (IRISA) pour les aides et le soutien financier.

En particulier, je souhaite ne pas oublier ma famille (mon père, ma mère, mes deux sœurs et mon frère) et mes amis pour leur soutien durant ces trois années malgré la distance qui nous sépare.

Table des matières

Introduction

Dans cette thèse, nous étudions les mécanismes de routage *multicast* dans le cadre des réseaux informatiques et plus précisément, nous nous intéressons à la gestion des petits groupes et de l'ingénierie de trafic. Nous proposons donc, dans ce but des protocoles de routage efficaces et passant à l'échelle.

Contexte

La transmission de données a évolué pour faire face à de nouveaux besoins. Elle est passée de la communication en mode *unicast* (point à point) à la communication en mode *multicast* (d'un point vers plusieurs points ou de plusieurs points vers plusieurs points). Le *multicast* est devenu de plus en plus important avec l'apparition d'applications telles que la téléphonie sur IP, la vidéo-conférence, la diffusion de la radio et de la télévision sur Internet, et les simulations interactives distribuées. Mais, le *multicast* est aussi adapté à des applications comme les jeux distribués et le transfert de fichiers vers des localisations multiples. Ceci impose au réseau d'offrir un service *multicast* efficace pour accompagner le développement de ces applications.

En *unicast*, pour envoyer un paquet vers plusieurs destinataires, une source génère une copie du même paquet pour chaque destinataire. Pour acheminer un paquet *multicast*, un arbre *multicast* doit être construit entre la source et les destinataires (membres du même groupe). Une seule copie du paquet est acheminée sur chaque branche de l'arbre *multicast*. Le protocole chargé de construire l'arbre *multicast* et d'acheminer les paquets le long de cet arbre est appelé protocole de routage *multicast*.

Nous présentons dans cette thèse un ensemble de protocoles de routage *multicast* [LB00, SM97] et nous décrivons les facteurs influençant le choix d'un arbre *multicast* comme : le coût de l'arbre, le délai de la source aux membres, la résistance au facteur d'échelle, le temps de construction de l'arbre, l'agrégation des chemins et la stabilité de l'arbre. Par la suite, nous présentons différentes techniques utilisées pour construire un arbre *multicast*.

Une première technique consiste à construire des arbres dits "arbres basés à la source" dont les racines sont les sources et dont les feuilles sont les destinataires (membres du même groupe).

Une deuxième technique consiste à ne construire qu'un seul arbre, partagé par toutes les sources d'un groupe.

Une troisième technique consiste à optimiser les arbres basés à la source ou partagés. Les arbres basés à la source ou partagés optimisés sont appelés arbres réduits [Gra97]. Il ne sera conservé dans ces arbres que les nœuds de branchement, nœuds ayant une fonction de duplication lors de la transmission de données dans l'arbre.

Pour être efficace et utilisé, un protocole de routage *multicast* doit être simple, robuste, résistant au facteur d'échelle, consommer un minimum de ressources et être inter-opérable avec les autres protocoles de routage *multicast* [Ram00]. Ces objectifs étant souvent contradictoires, il convient d'étudier les protocoles de routage selon leur portée.

Les protocoles de routage *multicast* dont la portée est limitée à un même domaine d'administration sont appelés protocoles de routage *multicast* intra-domaine (DVMRP [WPD88], MOSPF [Moy94b, Moy94a], PIM-DM [Sia03], CBT [Bal97a, Bal97b], PIM-SM [EFH$^+$98]). Les routeurs appartenant à un arbre *multicast* dans un domaine doivent mémoriser dans une table de routage *multicast* un état de routage *multicast* correspondant à l'arbre *multicast*. Le nombre d'états de routage *multicast* dans un routeur croît avec le nombre d'arbres construits dans le domaine et passants par ce routeur. Cela constitue un problème de résistance au facteur d'échelle car le temps de recherche dans la table de routage *multicast* pour un paquet dépend du nombre d'états dans cette

table.

Les protocoles de routage *multicast* dont la portée s'étend sur plusieurs domaines d'administration sont appelés protocoles de routage *multicast* inter-domaine. Pour ces protocoles, d'autres problèmes s'ajoutent à celui de la résistance au facteur d'échelle. Dans Internet, on constate une grande complexité du routage *multicast* inter-domaine avec les solutions proposées (la combinaison de BGMP [SKTA+98] et de MASC [REG+00] ou la combinaison de PIM-SM [EFH+98], de MSDP [FM03] et de MBGP [BCKR98]), et des difficultés liées à l'allocation des adresses de groupes et à la sécurité [LB00]. En effet, une adresse IP classe D identifie un groupe. Les paquets *multicast* ont pour adresse de destination une adresse *multicast*. Des collisions des adresses de groupes représentant différentes applications *multicast* peuvent survenir. L'allocation d'adresses *multicast* uniques doit se faire d'une manière globale. En plus, il n'existe aucun mécanisme pour empêcher des sources (resp. des destinataires) non autorisées d'envoyer (resp. de recevoir) du trafic à (resp. vers) un groupe *multicast*.

Une solution pour gérer les problèmes décrits précédemment a été introduite avec les deux services *multicast* : SSM (*Source Specific Multicast*) [HC03, Bha03] et Xcast (*Explicit Multicast*) [BFI+03], mais ces deux services nécessitent des changements importants du modèle *multicast*. SSM et Xcast sont proposés comme étant des solutions efficaces pour le routage *multicast* inter-domaine assurant ainsi une certaine résistance au facteur d'échelle et plus de simplicité en éliminant la nécessité d'une allocation stricte d'adresses.

SSM a repris les idées de base d'un protocole de routage *multicast* : EXPRESS. EXPRESS [HC99] introduit la notion de canal : un destinataire s'abonne à un canal (S, G), où S est l'adresse *unicast* de la source et G une adresse de groupe classe D. Le routage *multicast* est basé sur (S, G) ce qui permet à un destinataire d'un groupe de spécifier la source dont il veut recevoir des paquets. L'arbre *multicast* (basé à la source S) est appelé ainsi arbre source spécifique. Puisque l'adresse *unicast* est par définition unique, la source produit des adresses de groupe différentes pour ses diverses applications. L'allocation d'adresse est simplifiée et l'unicité de cette adresse est assurée

localement à la source.

Xcast est basé sur l'idée qu'un unique protocole de routage *multicast* est incapable de servir les différents types d'applications *multicast* [Sit03]. Xcast est proposé pour être utilisé avec un grand nombre de groupes ayant peu de membres. En effet, les protocoles de routage *multicast* doivent présenter une certaine flexibilité selon les besoins des applications. Dans un réseau où il y a un très grand nombre de petits groupes *multicast* (*SGM : Small Group Multicast* [Oom00]) dont les destinataires sont largement dispersés, le modèle de *multicast* traditionnel ne convient pas [DHPP00]. Xcast utilise un routage explicite en encodant la liste complète des membres du groupe dans l'en-tête de chaque paquet au lieu d'utiliser une adresse *multicast*. Xcast élimine ainsi la signalisation nécessaire à la gestion des états de routage *multicast* puisqu'il utilisera les états de routage *unicast*.

Nous proposons une généralisation du service Xcast: le protocole GXcast. GXcast [BGC04] permet d'éviter la fragmentation de paquets Xcast en limitant le nombre d'adresses des destinations encodés dans l'en-tête du paquet Xcast. GXcast est adapté aux groupes *multicast* de taille petite à moyenne. L'adhésion aux groupes se fait à l'aide de messages d'adhésion (appelé messages d'adhésion source spécifique) adressés directement à la source. Ces messages d'adhésion sont identiques aux messages d'adhésion à un canal dans SSM.

Nous proposons aussi une autre solution adaptée aux groupes *multicast* de petite taille : le protocole SEM (*Simple Explicit Multicast*). SEM [BC03] utilise le service Xcast pour construire un arbre réduit [Gra97] entre la source et les destinataires. Une fois l'arbre réduit construit, les paquets *multicast* sont acheminés en utilisant les adresses *unicast* des différents nœuds de branchement. L'arbre réduit est appelé aussi arbre *unicast* récursif (à l'origine proposé dans REUNITE [SEZ00]).

En revanche, avec l'apparition des applications multimédias et temps réel comme la vidéo à la demande, la visioconférence, le service de transmission de données *best effort* de l'Internet ne suffit plus. En effet, ces applications ont des besoins en terme, par exemple, de bande passante, délai et gigue (qu'on appelle qualité de service) qui

ne peuvent pas être satisfaits et garantis par les mécanismes actuels. Un des modèles les plus prometteurs pour fournir la qualité des service à travers les réseaux à hauts débits est l'ingénierie de trafic. Les protocoles décrits précédemment ne prennent pas en considération l'ingénierie de trafic *multicast*. L'ingénierie de trafic *multicast* est en cours de normalisation au sein de l'IETF (*Internet Engineering Task Force*). Après avoir défini l'ingénierie de trafic *multicast* et sa particularité vis-à-vis de l'ingénierie de trafic *unicast*, nous étudions la combinaison du *multicast* et de MPLS (*Multi-Protocol Label Switching*) [RVC01] en tant qu'outil d'ingénierie de trafic. Nous proposons le protocole de routage *multicast* dans un réseau MPLS : MMT (*MPLS Multicast Tree*). MMT [BC02] utilise les chemins MPLS entre les nœuds de branchement de l'arbre *multicast* (arbre réduit) afin de rendre la communication en mode *multicast* simple, d'augmenter la résistance au facteur d'échelle et de réduire le nombre d'états de routage *multicast*.

Contribution personnelle

Évaluation des protocoles de routage *multicast*. Il s'est avéré nécessaire d'évaluer les différents protocoles de routage *multicast* afin de montrer que le modèle actuel de *multicast* ne résiste pas au facteur d'échelle et d'introduire les deux services SSM et Xcast.

Le protocole GXcast. Une description détaillée de l'algorithme de routage et une étude sur le délai et le coût de l'utilisation de ce protocole sont présentées. Enfin, une analyse détaillée du protocole est menée et les résultats sont validés à travers d'une simulation.

Simulateur de groupes *multicast* explicites. Un simulateur de groupes *multicast* explicites est développé au cours de cette thèse. À l'aide de ce simulateur nous avons pu analyser les mécanismes du protocole GXcast en terme du coût et de délai. Il est à

notre connaissance le seul de ce type et peut être un bon outil pour de futures recherches sur les groupes *multicast* explicites (utilisant le service Xcast).

Le protocole SEM. L'impact de l'existence de nœuds de branchement sur l'arbre *multicast* est étudié. Un protocole de routage *multicast* tirant profit de ces nœuds est proposé. Le protocole est comparé avec d'autres protocoles similaires et analysé à travers une simulation.

L'ingénierie de trafic ***multicast*****.** Il était important de proposer une définition claire de l'ingénierie de trafic *multicast* et sa particularité vis-à-vis de l'ingénierie de trafic *unicast* afin de clarifier et de justifier notre choix de MPLS comme outil d'ingénierie de trafic.

Évaluation des protocoles de routage ***multicast*** **MPLS existants.** Les propositions MPLS pour l'ingénierie de trafic *multicast* sont étudiées et servent de support à la présentation de notre protocole MMT.

Simulateur de ***multicast*** **MPLS.** Un simulateur de *multicast* MPLS est développé au cours de cette thèse. Il est à notre connaissance le seul de ce type et peut être un bon outil pour de futures recherches sur le *multicast* MPLS.

Le protocole MMT. Un protocole de routage *multicast* dans un domaine MPLS, MMT est proposé. Ce protocole est comparé avec d'autres propositions *multicast* MPLS et analysé à travers une simulation.

Plan de la thèse

Cette thèse est organisée en quatre chapitres.

Dans le premier chapitre, nous présentons les différentes techniques de construction d'arbre *multicast*. Nous décrivons l'évolution des protocoles de routage *multicast*

de l'intra-domaine à l'inter-domaine. Nous fournissons ainsi une vue globale sur les protocoles de routage *multicast* intra-domaine et les solutions adaptées pour le *multicast* inter-domaine et nous présentons brièvement les deux services Xcast et SSM.

Dans le deuxième chapitre, nous étudions un protocole de routage explicite généralisé. Après avoir introduit le protocole Xcast [BFI+03] et son extension Xcast+ [MYKS01], leurs avantages et leurs inconvénients, nous présentons un protocole développé pendant cette thèse : le protocole GXcast. Nous décrivons le format d'un paquet GXcast puis son algorithme de routage. Nous étudions ensuite le protocole GXcast en terme de surcoût et de délai et la résistance au facteur d'échelle en comparaison à d'autres protocoles *multicast*. Nous simulons et validons enfin notre protocole grâce à des simulations effectuées par le simulateur NS [FV01].

Dans le troisième chapitre, nous étudions le protocole de routage explicite *multicast*, SEM [BC03]. Afin de construire un arbre *multicast*, la source SEM encode la liste des adresses IP des destinataires dans un message *branch*. Ce message utilise le principe du service Xcast et a pour rôle de découvrir les routeurs de branchement de l'arbre *multicast*. Pour acheminer les paquets *multicast*, SEM utilise les arbres *unicast* récursifs et les paquets sont acheminés d'un routeur de branchement à un autre en suivant l'arbre construit par le message *branch*. Nous décrivons le format d'un paquet SEM, l'algorithme de routage, la comparaison avec d'autres protocoles *multicast* (HBH [HCFCD01], PIM-SM [EFH+98] et GXcast [BGC04]). Nous simulons et validons enfin notre protocole grâce à des simulations effectuées par le simulateur NS.

Dans le chapitre quatre, nous étudions l'ingénierie de trafic *multicast* et la combinaison du *multicast* et du MPLS. Nous présentons une conclusion sur les exigences pour les algorithmes de routage MPLS *multicast*. Nous décrivons ensuite les motivations pour notre protocole *multicast* MPLS, le protocole MMT (*MPLS Multicast Tree*). Nous présentons le concept du MMT et son algorithme de routage. Nous analysons le protocole MMT [BC02] en terme de résistance au facteur d'échelle et d'efficacité. Nous discutons également de la mise en application d'un simulateur pour les protocoles de routage *multicast* MPLS [BCJD03] en utilisant l'outil de simulation NS et

nous validons enfin notre protocole grâce à des simulations effectuées par ce simulateur.

Nous concluons cette thèse par une synthèse de nos contributions, puis nous évoquons les perspectives offertes par nos travaux dans le domaine du routage *multicast* et dans le domaine de l'ingénierie de trafic *multicast*.

Chapitre 1

Le routage *multicast*

Le service *multicast* est de plus en plus employé par les applications multimédia. La transmission de données en mode *multicast* est la capacité d'un réseau à transmettre un message d'une application à de multiples destinataires pouvant être situés en différents emplacements mais appartenant à un même groupe.

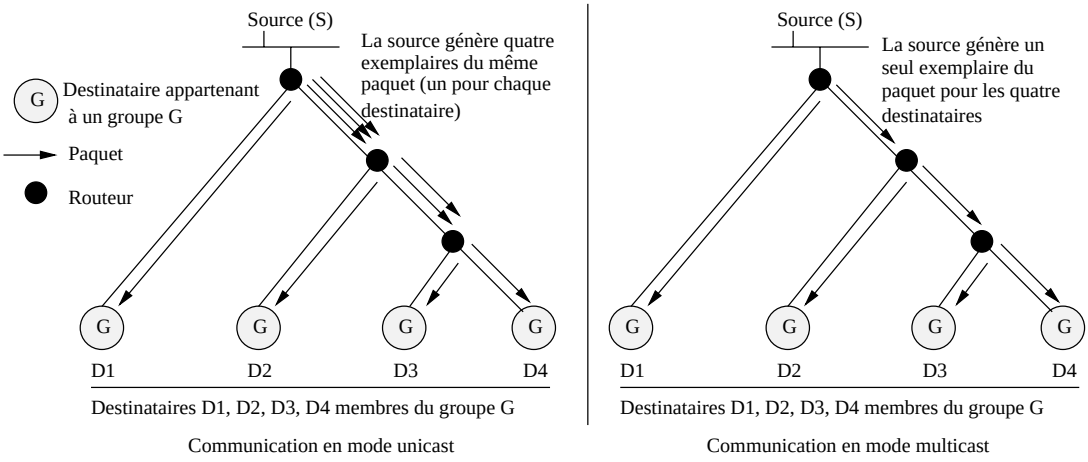


FIG. 1.1 – *Communication en mode unicast et communication en mode multicast.*

Une telle transmission peut être réalisée en utilisant des communications en mode *unicast* c'est-à-dire en envoyant un message (point à point) à chaque destinataire du groupe. Toutefois, cela conduit à faire circuler de multiples exemplaires d'un même paquet sur un même lien, consommant ainsi inutilement de la bande passante (*cf.* figure

1.1).

Dans Internet[1] une adresse *multicast*, par exemple une adresse IP classe D (de 224.0.0.0 à 239.255.255.255) [IANA01], identifie un groupe. Les paquets *multicast* ont pour adresse de destination une adresse *multicast.*

Sous IP, l'ensemble des membres d'un groupe est entièrement dynamique. Ainsi, n'importe quelle machine peut, à tout moment et indépendamment de toute localisation physique, adhérer à un groupe ou bien le quitter. De plus, une machine peut être, en même temps, membre de plusieurs groupes. Pour acheminer un paquet *multicast*, un arbre *multicast* doit être construit entre la source et les destinataires. Un protocole de routage *multicast* est chargé de construire un arbre *multicast* et ainsi permet d'acheminer les paquets le long de cet arbre. Les routeurs appartenant à un arbre *multicast* dans un domaine doivent mémoriser dans une table de routage *multicast* un état de routage *multicast* correspondant à l'arbre *multicast*.

Après avoir présenté les facteurs influençant le bon choix d'un arbre *multicast* et les différentes techniques de construction d'un arbre *multicast*, nous décrivons les principaux protocoles de routage *multicast*. Nous discutons l'évolution de ces protocoles de l'intra-domaine à l'inter-domaine et nous montrons que les solutions adaptées pour le *multicast* inter-domaine ne sont pas résistants au facteur d'échelle. Nous présentons à la fin les deux services Xcast et SSM et nous justifions notre choix d'utiliser ces deux services pour développer des protocoles de routage *multicast* originaux, efficaces, simples, robustes et résistants au facteur d'échelle.

1. Dans notre étude, nous nous intéressons essentiellement à Internet puisque ce réseau est omniprésent dans le monde. Cependant, Internet n'est qu'un exemple et les procédés décrits pourraient être utilisés par d'autres types de réseau.

1.1 Les facteurs influençant le bon choix d'un arbre *multicast*

Les différents facteurs influençant le bon choix d'un arbre *multicast* sont classés selon un ordre générale de priorité. Néanmoins, cet ordre peut varier d'une application *multicast* à une autre. Les paragraphes suivants résument la plupart de ces facteurs [LB00] :

Coût : on peut représenter les ressources réseau utilisées par un arbre empruntant un certain lien au moyen d'un coût pour ce lien. Le coût global d'un arbre *multicast* est la somme des coûts de tous ses liens. Un algorithme de routage *multicast* peut avoir comme but de minimiser ce coût.

Délai : le délai de la source vers un destinataire est la somme des délais sur chacun des liens du chemin entre la source et ce destinataire[2]. Un algorithme de routage *multicast* doit tendre à minimiser le délai entre la source et n'importe quel destinataire.

Un difficile compromis doit être trouvé lors de la recherche d'un algorithme de routage qui tend à minimiser le coût et le délai en même temps [LB00, Che99] puisqu'on peut ne pas trouver un arbre optimal pour ces deux contraintes à la fois.

Résistance au facteur d'échelle : la résistance au facteur d'échelle d'un arbre *multicast* se caractérise par deux aspects :

*** Premièrement,** pour n'importe quel groupe *multicast*, le temps nécessaire pour construire l'arbre doit être raisonnable tout en utilisant efficacement les ressources disponibles.

*** Deuxièmement,** les routeurs du réseau doivent permettre la gestion d'un nombre élevé d'arbres *multicast*. Ces arbres doivent coexister ensemble sur le même réseau.

2. À ce délai peut s'ajouter le délai induit dans les nœuds de l'arbre.

Les routeurs doivent donc mémoriser un minimum d'états de routage par groupe et les tables de routage doivent être de tailles réduites. La table de routage est partagée par l'ensemble des groupes d'où la nécessité de l'agrégation des états de routage.

Agrégation : chaque machine connectée au réseau (la source ou le destinataire) possède au moins une adresse *unicast*. Ainsi pour transmettre un message vers un destinataire, les routeurs situés sur le chemin de la source vers ce destinataire doivent mémoriser l'adresse du destinataire ainsi que l'adresse du prochain routeur vers ce destinataire. Mais en mode *unicast*, les entrées des tables de routage *unicast* peuvent être agrégées en utilisant une agrégation des adresses *unicast*. L'agrégation signifie qu'un ensemble d'adresses IP correspondant toutes au même prochain routeur, est remplacé par une seule entrée dans la table de routage. Lorsqu'un routeur reçoit un paquet, il recherche l'entrée de la table de routage qui correspond à l'adresse de destination contenue par ce paquet. L'entrée recherchée est celle qui possède la plus longue suite binaire commune avec l'adresse. La table de routage *unicast* est plus courte car plusieurs adresses contiguës sont regroupées dans une seule entrée.

L'agrégation est plus difficile en mode *multicast*. Ceci vient du fait que l'adresse *multicast* peut représenter des destinataires se trouvant dans différentes localisations physiques. L'agrégation devient encore plus difficile lorsque plusieurs sources existent. Le routage *multicast* est basé sur les adresses IP *multicast*. L'objectif final est d'acheminer le paquet à tous les destinataires du groupe *multicast* associé à cette adresse IP *multicast*. Puisque les destinataires peuvent être situés n'importe où dans l'Internet, il n'y a aucune autre alternative qu'une entrée par adresse IP *multicast* dans la table de routage *multicast*. Cela veut dire qu'il est très difficile d'agréger des adresses IP *multicast*. C'est un problème intrinsèque de résistance au facteur d'échelle *multicast* qui ne peut pas être évité. Un algorithme de routage *multicast* doit tenir compte du problème d'agrégation et avoir comme but de le résoudre ou bien de réduire son effet.

Stabilité : l'ensemble des membres d'un groupe est entièrement dynamique. Cela signifie que toute machine peut, à tout moment, demander de joindre le groupe ou bien

le quitter. Il n'y a *a priori* aucune restriction sur le nombre de membres d'un groupe ou sur leur localisation. Un algorithme de routage doit assurer que l'adhésion d'un membre à un groupe, ou son départ, ne doit influencer que d'une façon minime sur la qualité de l'arbre (délai, coût).

Le stress des liens : le nombre de copies du même paquet transmis sur un lien est dénoté comme le stress du lien. En *unicast*, le stress des liens augmente avec le nombre de destinataires dans le réseau et il peut avoisiner le nombre de destinataires pour les liens aux alentours de la source. En *multicast*, la cause principale du stress des liens est l'incapabilité d'un nœud sur l'arbre *multicast* à fournir le service de duplication de paquets *multicast* et donc les paquets doivent être dupliquer dans un nœud en amont. Un algorithme de routage *multicast* doit minimiser le stress des liens et éviter l'utilisation des nœuds qui ne fournit pas le service de duplication d'un paquet *multicast*.

1.2 Les techniques de construction d'un arbre *multicast*

Nous allons décrire les principales techniques de construction d'arbre *multicast*. Les différentes techniques de construction d'arbres seront utilisées par les protocoles de routage *multicast* que nous décrivons dans les sous-chapitres suivants.

1.2.1 La technique d'arbre basé à la source

La technique d'arbre basé à la source ou SBT (*Source Based Tree*) consiste à construire un arbre différent pour chaque source au sein d'un groupe. Cet arbre est enraciné à la source et utilise souvent le plus court chemin entre la source et chaque destinataire. Le plus court chemin entre la source et le destinataire est appelé aussi meilleur chemin entre la source et le destinataire. En effet, le réseau est représenté par un graphe, dont les arêtes sont affectées d'une fonction de poids strictement positive. Ce poids pourra représenter la distance entre les deux nœuds attachés aux arêtes, le délai de transmission de données sur les arêtes, ou encore le coût financier attribué aux

arêtes.

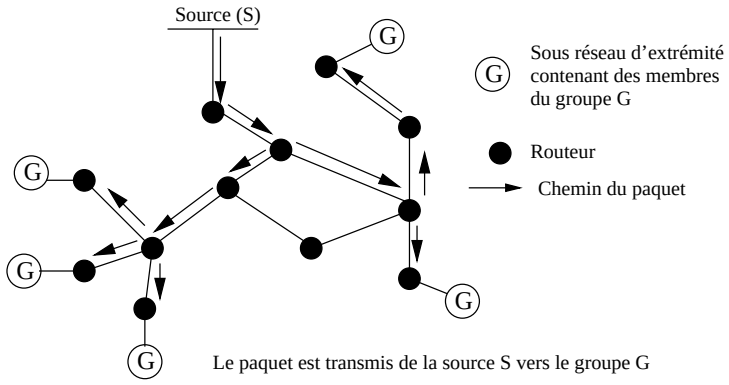


FIG. 1.2 – *Arbre basé à la source.*

1.2.2 La technique d'arbre partagé

La technique d'arbre partagé ou ST (*Shared Tree*) consiste à construire un arbre unique partagé par toutes les sources d'un groupe. En général, les différentes sources et destinataires dans un domaine de routage informent un routeur spécifique, appelé point de rendez-vous, de leur présence. La transmission de paquets entre les différentes sources et leurs destinataires s'effectue via le routeur point de rendez-vous (*cf.* figure 1.3).

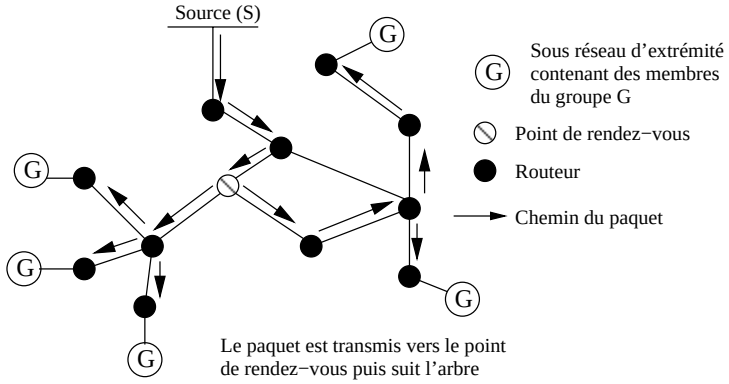


FIG. 1.3 – *Arbre partagé.*

1.2.3 Comparaison entre un arbre basé à la source et un arbre partagé

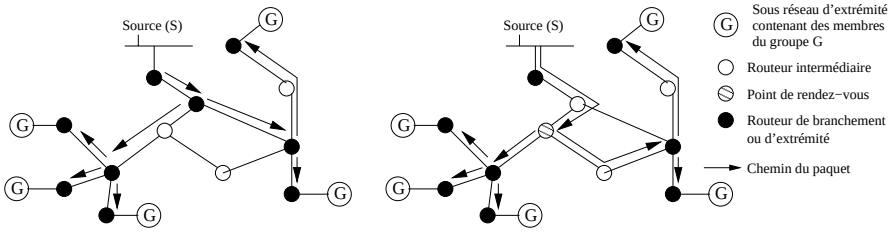
La technique d'arbre basé à la source est moins résistante au facteur d'échelle que la technique d'arbre partagé. Les algorithmes basés sur la technique d'arbres partagés utilisent d'une manière efficace les ressources des routeurs en évitant des entrées superflues dans leur table de routage. Ceci provient du fait qu'avec l'arbre partagé l'information à mémoriser dans chaque routeur concerne seulement le groupe tandis qu'avec l'arbre basé à la source cette information concerne la source et le groupe. En revanche, un arbre partagé peut créer des chemins non-optimaux vis-à-vis de la source, ce qui amène à un retard accru pouvant être critique pour certaines applications multimédia. En plus, il est probable, surtout dans le cas de nombreuses sources, que le trafic soit concentré sur le même ensemble de liens menant au point de rendez-vous de l'arbre.

Finalement, lorsqu'un arbre basé à la source est construit par les messages d'adhésion au groupe envoyés explicitement par les destinataires, l'arbre utilisé généralement est l'arbre des plus courts chemins inverses [Gra97] puisqu'il est construit à partir des chemins menant à la source d'un paquet (et non pas venant de la source). Dans le cas d'un réseau asymétrique, l'arbre peut ne pas être optimal pour une source donnée. De même un arbre partagé peut utiliser le chemin menant au point de rendez-vous (et non pas venant du point de rendez-vous). Nous appelons l'interface menant à la source ou au point de rendez-vous, l'interface RPF (*Reverse Path Forwarding Interface*).

1.2.4 La technique d'arbre réduit et l'*unicast* récursif

Les arbres réduits présentés dans [Gra97] sont une optimisation des arbres basés à la source ou partagés. Les algorithmes basés sur la technique d'arbre réduit construisent un arbre dont seuls les nœuds intermédiaires où la duplication se produit (appelés ci-après nœuds de branchement) sont conservés. Cette approche permet une économie importante de ressources, liée à l'absence d'informations d'état dans les

nœuds intermédiaires qui ne sont pas de branchement.



(a) Arbre réduit basé à la source.

(b) Arbre réduit partagé.

FIG. 1.4 – *Arbres réduits.*

L'acheminement d'un paquet selon l'*unicast* récursif peut être considéré comme une application de la technique d'arbre réduit. En effet, un paquet est acheminé selon une adresse *unicast*. Cette adresse peut être l'adresse du premier destinataire qui s'est abonné au groupe [SEZ00] ou bien l'adresse du prochain nœud de branchement [Gra97, HCFCD01, BC03]. Un mécanisme est mis en œuvre pour la découverte des nœuds de branchement là où la duplication se produit. Les routeurs de branchement dupliquent les paquets selon les entrées de l'état de routage. Les paquets dupliqués ont tous une adresse de destination *unicast*. Cette opération est répétée d'une manière récursive dans chaque routeur de branchement sur les chemins vers les destinataires.

1.3 Les protocoles de routage *multicast*

Trois types de protocoles sont utilisés pour l'implémentation du *multicast* sur Internet. Le premier type est employé au sein d'un sous réseau par un destinataire afin d'adhérer à un groupe ou bien de le quitter. Le protocole IGMP (*Internet Group Management Protocol*) [Fen97, CDT02] en est un exemple typique. Le second type, dont la portée est limitée à un même domaine d'administration, est appelé MIGP (*Multicast Interior Gateway Protocol*) ou bien protocole intra-domaine. DVMRP [WPD88], MOSPF [Moy94b, Moy94a], PIM [Sia03, EFH+98] et CBT [Bal97a, Bal97b] repré-

sentent des exemples de ce type de protocole. Ce type de protocole présente des limitations et des problèmes de résistance au facteur d'échelle s'il est utilisé à travers plusieurs domaines. Le troisième type, appelé inter-domaine, est employé par les routeurs de bordure pour permettre le service *multicast* à travers différents domaines. Le protocole BGMP [Tha03] représente un exemple de ce type de protocole.

1.3.1 Le protocole IGMP

Pour recevoir des paquets d'une source transmettant des données en utilisant la communication en mode *multicast*, un destinataire doit prévenir un routeur spécifique de son réseau local (appelé ci-après routeur désigné ou DR (*Designated Router*)) qu'il fait partie de ce groupe. Le DR, étant généralement élu parmi les différents routeurs qui fonctionnent en mode *multicast* sur le réseau local, est chargé de prendre contact avec les autres routeurs de l'Internet qui fonctionnent en mode *multicast* pour leur transmettre les informations d'appartenance aux groupes et pour déterminer le routage. Pour communiquer avec leur DR, les destinataires doivent utiliser le protocole de gestion des groupes *multicast* IGMP [Fen97, CDT02].

IGMP comporte deux phases : dans la première, un destinataire qui désire adhérer à un groupe, diffuse un message IGMP pour indiquer à toutes les machines de son réseau local son adhésion au groupe. Dans la deuxième, le DR interroge périodiquement les machines du réseau local, pour connaître leur statut par rapport au groupe. Le DR reçoit les messages d'appartenance aux groupes mais il ne tient pas à jour une table de destinataires : il suffit qu'il y ait au moins un destinataire appartenant à un groupe dans son réseau local pour qu'il continue à envoyer le trafic correspondant. Lorsque le DR émet un message d'interrogation IGMP, tous les destinataires membres d'un groupe répondent après un délai aléatoire et ceci pour deux raisons : éviter les réponses massives simultanées et donner la possibilité à certains destinataires membres du groupe de ne pas répondre (un destinataire qui entend la réponse émise à l'adresse du groupe par un autre destinataire annule sa réponse). Si aucun destinataire appartenant à un groupe ne répond, le DR considère qu'aucun destinataire du réseau local appartient à ce groupe.

Il cesse alors de diffuser aux autres routeurs *multicast* de l'Internet des informations d'appartenance au groupe.

Dans sa version deux, IGMPv2 [Fen97], le protocole IGMP a été optimisé afin de réduire le temps de latence entre le retrait d'un destinataire du groupe et l'arrêt effectif du flux de paquets lui étant destiné. Lorsque le DR reçoit d'un destinataire un message indiquant sa volonté de quitter un groupe il envoie une requête spécifique pour s'informer de l'état d'appartenance à ce groupe dans le réseau local. Dans le cas où il n'y a aucune réponse, le DR considère qu'aucun destinataire du réseau n'appartient plus à ce groupe. Le DR cesse alors de diffuser vers les autres routeurs de l'Internet qui fonctionnent en mode *multicast* des messages d'appartenance au groupe.

Afin de permettre à des destinataires membres d'un certain groupe de spécifier la source dont ils veulent recevoir les paquets, la version trois d'IGMP, IGMPv3 [CDT02], utilise la notion de canal : un destinataire s'abonne à un canal (S, G), où S est l'adresse *unicast* de la source et G une adresse de groupe IP classe D. Le routage *multicast* est basé sur l'identificateur de canal (S, G) ce qui permet à un destinataire d'un groupe de spécifier la source dont il veut (ou il veut pas) recevoir des paquets.

1.3.2 Les protocoles de routage *multicast* intra-domaine

Les protocoles de routage *multicast* se regroupent en deux familles:

Les protocoles dits à forte densité de membres (*Dense mode*) comme DVMRP, MOSPF et PIM-DM qui est le plus récent. Ces protocoles supposent qu'il y a un très grand nombre de membres dans un domaine restreint et que l'absence de membre dans un sous-domaine est une exception. Ils sont basés sur la technique de l'arbre basé à la source.

Les protocoles dits à membres épars (*Sparse mode*) comme CBT et PIM-SM. Ces protocoles supposent, au contraire, que les membres du groupe *multicast* sont dispersés. Ils sont basés sur la technique d'arbre partagé.

1.3.2.1 Le protocole DVMRP

DVMRP (*Distance Vector Multicast Routing Protocol*) [WPD88] est un protocole de routage *multicast* dérivé du protocole de routage *unicast* RIP [Mal98]. Le protocole DVMRP associe à chaque route, une métrique permettant l'évaluation du coût de la route en nombre de sauts.

Le premier paquet est propagé par inondation à tout le réseau. Afin d'éviter les cycles, un routeur accepte un paquet émis par une source S vers un groupe G, s'il le reçoit sur l'interface RPF (*cf.* sous-chapitre 1.2.3), sinon il se débarrasse du paquet. Le routeur utilise les tables de routage DVMRP pour déterminer la meilleure route vers la source S.

Si le routeur accepte le paquet, il crée une entrée (S, G) en mémoire, et dresse la liste des interfaces de sortie vers lesquelles il doit réémettre ce paquet.

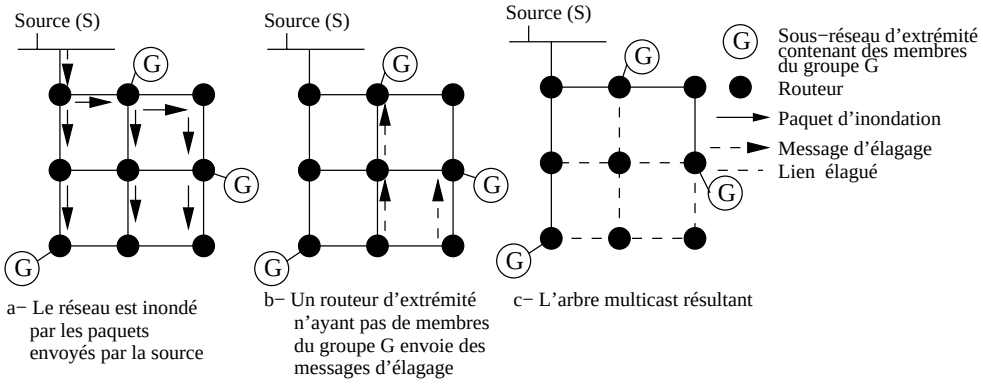


FIG. 1.5 – *La construction de l'arbre en DVMRP.*

La liste des interfaces de sortie pour les paquets (S, G) peut être réduite si le routeur détermine qu'aucun membre actif du groupe G ne peut être atteint *via* une interface de sortie. Ce mécanisme qui consiste à restreindre l'arbre (S, G) aux branches menant à des membres actifs du groupe G, est appelé l'élagage. Si un routeur en aval découvre qu'il est routeur d'extrémité, et qu'aucun membre de son réseau local n'est adhérent au groupe G (il n'a pas reçu de messages IGMP d'adhésion), il envoie un message d'élagage vers toutes les sources S émettant pour G. Les messages d'élagage peuvent remonter de proche en proche vers les sources S, élaguant ainsi les arbres. Le routeur

conserve dans l'entrée (S, G) de sa table de routage l'état élagué de l'interface de sortie par laquelle il a reçu un message d'élagage et déclenche un compte à rebours associé à cet état. Quand ce compte à rebours se termine, le routeur réémet de nouveau les paquets (S, G) vers cette interface jusqu'à la réception éventuelle d'un nouveau message d'élagage, afin de connecter d'éventuels nouveaux membres (*cf.* figure 1.5).

Le greffage évite d'attendre que les comptes à rebours déclenchés par les précédents messages d'élagages n'expirent. Les messages de greffe comme ceux d'élagage, remontent de proche en proche vers les sources S, de façon à reconstruire les arbres (S, G).

DVRMP inclut également le principe de *tunneling*, un procédé qui permet de passer des paquets *multicast* à travers des routeurs qui ne permettent pas le routage *multicast*. Ce principe utilise un en-tête IP particulier. Les adresses de source et de destination sont encaspsulées dans un élément optionnel appelé *Loose Source Routing*. L'adresse source est celle du routeur *multicast* qui émet le paquet et l'adresse de destination est celle du routeur *multicast* destinataire. Ce dernier, après réception du paquet, rétablit l'adresse d'origine et la destination d'origine.

1.3.2.2 Le protocole MOSPF

MOSPF (*Multicast Extensions to OSPF*) [Moy94b, Moy94a], est une extension *multicast* du protocole *unicast* OSPF [Moy91]. MOSPF diffuse en mode *multicast* les paquets selon l'arbre basé à la source des plus courts chemins dans un domaine d'administration.

OSPF est un protocole de routage *unicast* à état des liens qui divise un domaine d'administration en plusieurs zones. Chaque routeur OSPF conserve et met à jour une base de données sur les états des liens, qui décrit la topologie dans chaque zone. Cette base de données est construite grâce à la diffusion des différents états des liens par des messages LSA (*Link State Advertisement*). Chaque message LSA est diffusé par inondation à travers toute la zone de routage.

MOSPF introduit les messages d'appartenance aux groupes (*Group Membership*

LSA) afin de permettre aux routeurs de construire l'arbre de routage *multicast*. Un routeur désigné dans une zone utilise IGMP pour s'informer de l'appartenance au groupe des membres de son sous-réseau et il est responsable de la distribution de cette information par inondation de messages *Group Membership LSA* vers tous les routeurs dans la zone OSPF. Alors, en utilisant la base de données de l'état des liens, chaque routeur calcule l'arbre des plus courts chemins. Par la suite, les messages *Group Membership LSA* sont utilisés pour élaguer les branches de l'arbre qui n'aboutissent pas à des membres. Pour chaque groupe *multicast*, chaque routeur MOSPF peut ainsi déterminer sa position sur l'arbre des plus courts chemins et mettre à jour sa une table de routage *multicast*. Cette table de routage n'est pas régénérée périodiquement, mais change seulement quand il y a un changement de la topologie du réseau ou un changement de l'appartenance aux groupes. Pour ménager les ressources de tous les routeurs du réseau, le plus court chemin peut être calculé à la demande : à l'arrivée du premier paquet *multicast* pour un groupe.

1.3.2.3 Le protocole CBT

CBT (*Core Based Tree*) [Bal97a, Bal97b] est un protocole de routage *multicast* basé sur la technique d'arbre partagé.

Les tables de routage *multicast* dans les routeurs CBT sont de taille inférieure que celles des protocoles utilisant les arbres basés à la source car l'information à mémoriser ne concerne pas l'ensemble des couples (source, groupe) mais seulement l'ensemble des groupes.

Un destinataire exprime sa volonté d'adhérer à un groupe en envoyant un message IGMP vers le routeur désigné DR de son réseau local. A la réception de ce message, le DR envoie un message d'adhésion vers le point de rendez-vous afin de joindre l'arbre (*cf.* figure 1.6a). Ce message sera explicitement reconnu et confirmé par le point de rendez-vous ou par un routeur intermédiaire qui a déjà adhéré à l'arbre avec succès. Un message d'acquittement sera alors envoyé, empruntant le même chemin dans le sens inverse. Ce message établit un état de routage dans les routeurs tout le long du

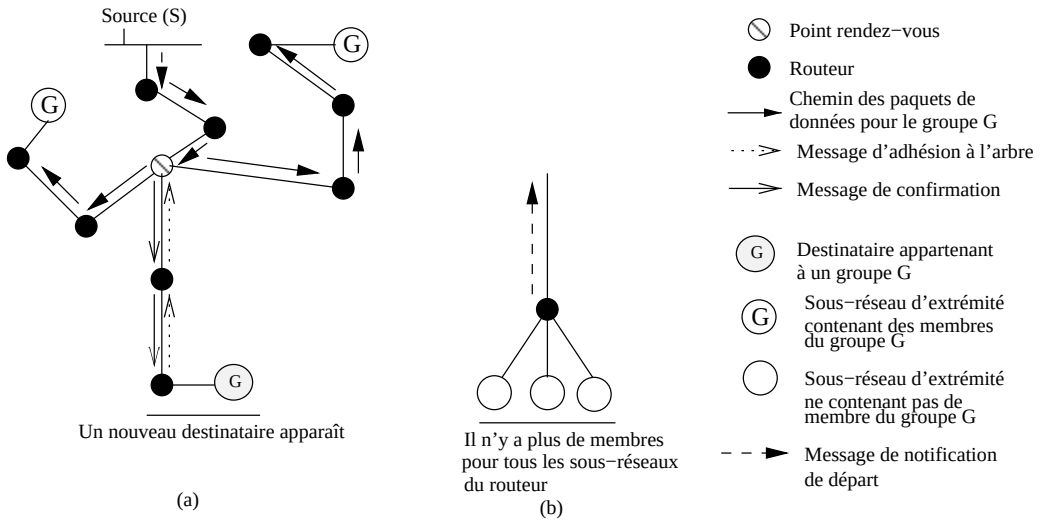


FIG. 1.6 – *(a) La procédure d'adhésion à un groupe en CBT, (b) la procédure de désabonnement d'un membre en CBT.*

chemin.

Cet état contient typiquement l'identificateur de groupe, l'interface sur laquelle le routeur père (c'est-à-dire le prochain routeur vers le point de rendez-vous) est connecté, l'interface de chacun des routeurs fils. Une fois que l'acquittement a atteint le DR du nouveau membre, le destinataire peut recevoir le trafic envoyé au groupe. Un routeur qui détecte qu'il n'y a plus aucun membre appartenant à un groupe sur ses liens en aval envoie un message de notification de départ (*cf.* figure 1.6b) vers le routeur en amont. Ceci se produit lorsque la liste des interfaces filles de ce routeur devient vide pour un groupe.

Si un routeur ou un lien devient inaccessible pour cause d'une défaillance, les routeurs qui sont en aval auront à rejoindre l'arbre de nouveau. La détection de l'inaccessibilité du routeur se fera grâce aux messages *Echo_Request* et *Echo_Reply* transmis périodiquement entre les routeurs (*cf.* figure 1.7b). Un message *Flush_Tree* sera transmis par le routeur qui a détecté l'inaccessibilité du routeur parent, vers toutes les interfaces filles (*cf.* figure 1.7a). Ce message servira à effacer tous les états relatifs à un groupe, en demandant aux récepteurs de rejoindre l'arbre de nouveau. Comme l'état créé dans les routeurs par le message d'acquittement d'adhésion est bidirectionnel (le

flux de données peut être échangé dans les deux sens le long d'une branche d'arbre), il n'y a pas de concept d'interfaces de sortie ou d'entrée, bien qu'il soit nécessaire de pouvoir distinguer les interfaces en amont de celles en aval. Ces interfaces sont connues comme les interfaces parentes et filles, respectivement.

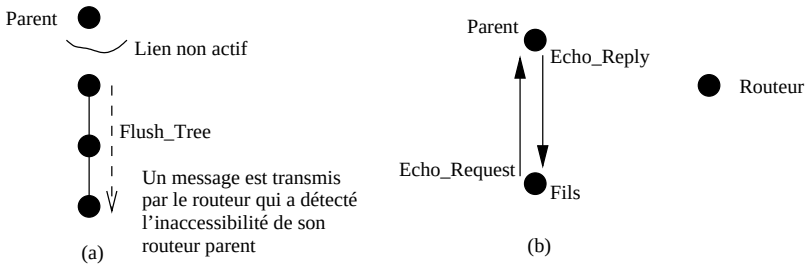


FIG. 1.7 – *(a) Parent inaccessible, (b) Maintenance de l'arbre.*

CBT ne sacrifie pas le facteur de passage à l'échelle pour assurer un délai minimal. CBT suggère qu'une solution du problème de délai (pour certaines applications) puisse être obtenue en plaçant le point de rendez-vous à coté de la source. CBT, de ce point de vue n'est pas robuste. En effet, les mécanismes du protocole doivent détecter la panne du point de rendez-vous et choisir ensuite rapidement un nouveau point de rendez-vous pour le remplacer. En revanche, CBT est simple et performant. CBT définit en plus des mécanismes d'interopérabilité avec DVMRP [Bal97a, Bal97b].

La concentration du trafic autour du point de rendez-vous et la construction d'un chemin non optimal (d'une source vers les destinataires passant par le RP) posent des limitations sur CBT. Ceci a conduit à chercher un protocole qui tient compte des deux techniques de construction de l'arbre de *multicast* : arbre basé à la source et arbre partagé. Le protocole PIM est un tel exemple.

1.3.2.4 Le protocole PIM

PIM (*Protocol Independent Multicast*) est un protocole de routage *multicast* indépendant d'un protocole de routage *unicast* particulier. PIM est conçu pour être plus efficace que les protocoles existants lors d'un déploiement assez large.

Il existe deux modes pour PIM : PIM-DM et PIM-SM. Ces deux modes correspondent à deux protocoles différents.

Le mode épars (mode SM) exige des routeurs connectés à des membres qu'ils envoient régulièrement un message d'adhésion à l'arbre afin que le trafic *multicast* soit envoyé vers ces membres. L'idée consiste à cesser d'envoyer des paquets sur les liens qui ne sont pas connectés à des membres du groupe. PIM-SM, comme CBT, est basé sur la technique d'arbre partagé et utilise le concept d'un point de rendez-vous (appelé ci-après RP). Les sources et les destinataires communiquent entre-eux à travers ce RP. Le RP doit être choisi de manière unique pour chaque groupe. Le mode dense (mode DM), comme pour DVMRP et MOSPF, est basé sur la technique d'arbre basé à la source.

Les deux modes de fonctionnement DM et SM peuvent coexister sur les mêmes routeurs. Les messages de contrôle et la diffusion des paquets de données de PIM-DM sont intégrés avec les opérations de PIM-SM pour qu'un seul routeur puisse exécuter des modes différents pour des groupes différents.

1.3.2.5 Le protocole PIM-DM

Un routeur PIM-DM [Sia03, SM97, Ban97] accepte un paquet (S, G) s'il le reçoit sur l'interface RPF. Le choix de cette interface se fait après consultation des tables de routage *unicast*.

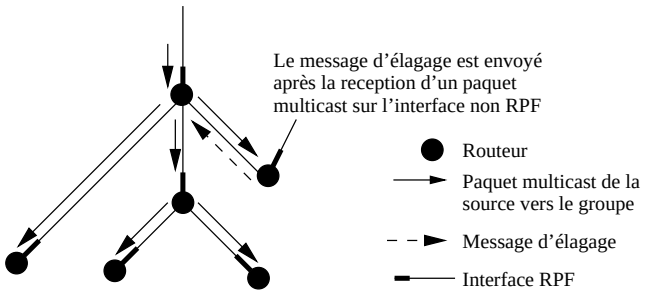


FIG. 1.8 – *Message d'élagage en PIM-DM.*

Le paquet accepté par le routeur sera réémis vers toutes les interfaces sur lesquelles

un routeur PIM-DM est présent, à l'exception de l'interface de réception. Si le paquet (S, G) est reçu par une interface différente de l'interface RPF, un message d'élagage est envoyé vers cette interface (*cf.* figure 1.8). Ce mécanisme permet à PIM-DM d'éviter la duplication des paquets. Un routeur recevant un message d'élagage sur une interface supprime cette interface de la liste des interfaces de sortie et déclenche un compte à rebours. Quand ce dernier expire, le routeur réémet les paquets (S, G) vers cette interface de sortie. Cette méthode permet à PIM-DM de prendre en compte les changements de topologie.

Si la liste des interfaces de sortie pour (S, G) est vide et qu'aucun membre du groupe G n'est présent sur tous les sous-réseaux connectés au routeur, ce dernier envoie alors un message d'élagage vers la source S, c'est-à-dire sur l'interface RPF menant à cette source S. À noter qu'un message d'élagage reçu en *multicast* sur un sous-réseau ne provoque la suppression de cette interface de la liste des interfaces de sortie qu'au bout de 3 secondes, alors que les messages d'élagage reçus sur une interface point à point sont traités immédiatement. Ceci parce qu'un autre routeur PIM en aval peut être présent sur le même sous-réseau. Ainsi si ce routeur veut continuer de recevoir les paquets destinés à G, il envoie un message d'adhésion au routeur en amont.

Le greffage permet de regreffer un sous-arbre élagué sans attendre que les comptes à rebours déclenchés par les messages d'élagage précédents n'expirent.

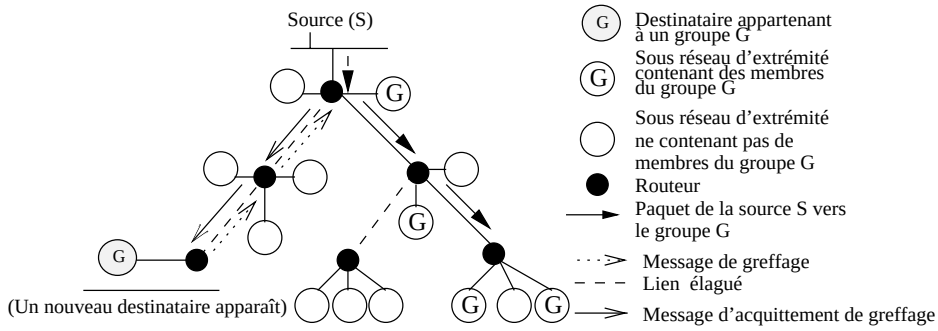


FIG. 1.9 – *Greffage en PIM-DM.*

Lorsqu'un routeur reçoit un message de greffage pour un groupe G sur une interface quelconque, il ajoute cette interface dans les listes des interfaces de sortie de tous

les couples (S, G) connus. Si, à l'arrivée de ce message, la liste d'interfaces de sortie (S, G) est vide, un message de greffage est envoyé pour le groupe G sur l'interface RPF conduisant au routeur en amont (c'est-à-dire placé sur le chemin menant à la source S). À noter qu'un routeur en aval réémettra un message de greffage jusqu'à ce que le routeur en amont l'ait acquitté par un message d'acquittement de greffage ou bien à la réception d'un paquet (S, G) sur l'interface RPF (*cf.* figure 1.9).

La maintenance de l'arbre est assurée par des messages *Hello* périodiquement envoyés par chaque routeur PIM vers toutes ses interfaces.

Les protocoles PIM-DM et DVMRP sont assez similaires mais contrairement à DVMRP, PIM-DM ne dépend pas d'un protocole de routage *unicast* particulier alors que DVMRP utilise la technique *distance vector* de diffusion des meilleures routes pour construire sa table de routage *multicast*.

1.3.2.6 Le protocole PIM-SM

PIM-SM [EFH+98] est un protocole de routage *multicast* basé sur la technique d'arbre partagé. PIM-SM peut aussi construire des arbres basés sur la technique d'arbre basé à la source. Ceci lui donne un avantage face aux autres protocoles basés seulement sur la technique d'arbre partagé comme le protocole CBT. Les décisions de routage sont basées, comme pour PIM-DM, sur l'existence d'une table de routage *unicast* quelconque et la construction de l'arbre est indépendante d'un protocole de routage *unicast* particulier.

Un point de rendez-vous doit être configuré de manière unique pour chaque groupe. Chaque source transmet les paquets *multicast* encapsulés en paquets *unicast* vers le point de rendez-vous. Un point de rendez-vous ayant déjà créé un état de routage pour le groupe (c'est-à-dire que des membres ont déjà adhéré au groupe), reçoit ces paquets, ôte l'encapsulation et transmet les paquets non encapsulés sur l'arbre partagé.

Dans PIM-SM, un destinataire annonce explicitement sa volonté de recevoir des paquets *multicast* transmis vers un groupe. Un arbre partagé, dont la racine est le RP, est formé pour chaque groupe.

Lorsqu'un DR reçoit un message IGMP qui indique la volonté d'un destinataire d'adhérer à un groupe, le DR détermine le RP de ce groupe et envoie un message *unicast* d'adhésion vers ce RP (*cf.* figure 1.10).

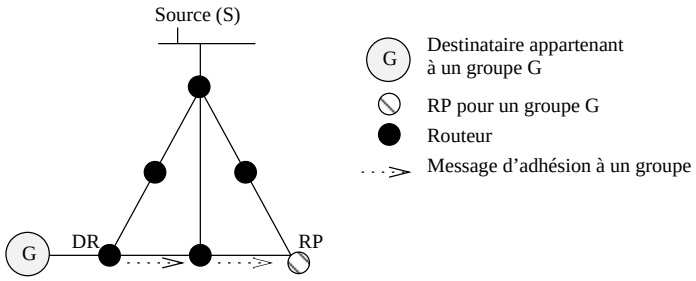


FIG. 1.10 – *Un nouveau destinataire adhère à un groupe multicast.*

Le DR et les autres routeurs *multicast* intermédiaires entre le DR et le RP créent une entrée notée $(*, G)$ indépendante de la source. Cette entrée permet de déterminer comment transmettre les paquets *multicast* venant du RP vers le DR et par la suite vers les destinataires membres du groupe.

Lorsqu'une source commence à transmettre des paquets vers un groupe, le DR de cette source encapsule les premiers paquets dans des paquets d'enregistrement et les transmet vers le RP de ce groupe en mode *unicast*. Après la réception de ces paquets d'enregistrement, le RP envoie un message d'adhésion vers le DR de cette source. Lors de l'envoi de ce message vers le DR, tous les routeurs intermédiaires ajoutent une nouvelle entrée spécifique pour la source S et le groupe G et cette fois notée (S, G) dans leur table de routage *multicast*.

Ainsi, les paquets *multicast* seront transmis vers le RP qui les transmet à son tour vers les destinataires membres du groupe. Il faut noter que jusqu'au moment où ces entrées sont ajoutées dans les tables de routage intermédiaires, tous les paquets *multicast* sont transmis comme des paquets *unicast* encapsulés (*cf.* figure 1.11).

Si le nombre de destinataires (ou la quantité de paquets transmis à travers cet arbre) augmente, l'utilisation d'un arbre des plus courts chemins peut être demandée pour remplacer l'arbre partagé.

Pour ce faire, le DR envoie, un message d'adhésion vers la source, afin de construire

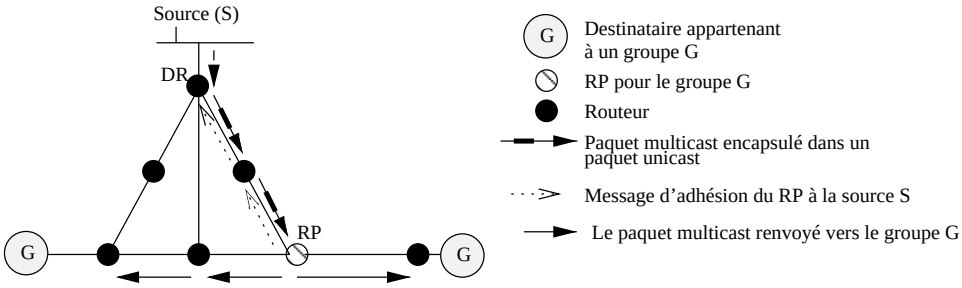


FIG. 1.11 – *Une source transmettant des paquets vers un groupe multicast dans PIM-SM.*

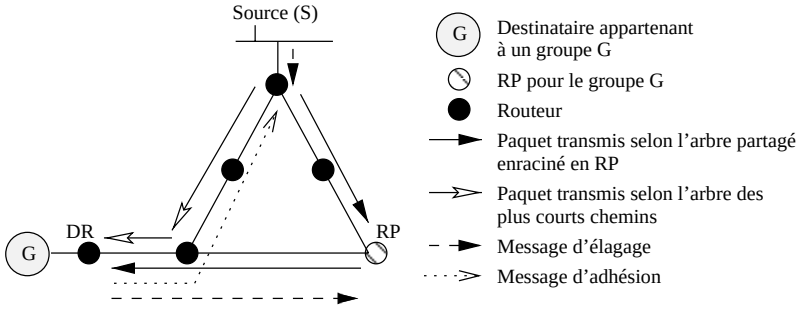


FIG. 1.12 – *Basculement entre l'arbre partagé enraciné en RP et l'arbre, des plus courts chemin, basé à la source.*

l'arbre des plus courts chemins basé à la source. Par la suite, le routeur peut envoyer un message d'élagage vers le RP (*cf.* figure 1.12).

1.3.2.7 Conclusion

Les protocoles à forte densité de membres utilisant la technique d'arbre basé à la source comme DVMRP, MOSPF et PIM-DM inondent le réseau par la diffusion périodique de paquets transmis vers tous les routeurs afin de construire l'arbre de routage *multicast*. Ces protocoles ne sont pas adaptés aux groupes de faible densité puisque les paquets vont atteindre des sous-réseaux dans lesquels il n'y a pas de destinataires. Ceci représente un handicap majeur pour ce genre de protocoles. On peut ajouter que les messages d'adhésion à un groupe et de désabonnement d'un groupe inondent le réseau périodiquement pour assurer la maintenance de l'arbre. De plus, la table de routage dans ces routeurs est de taille importante puisqu'elle concerne tous les couples (S, G).

Ainsi, les protocoles utilisant la technique d'arbre partagé comme CBT, présentent des avantages sur ceux utilisant la technique d'arbre basé à la source. En effet, ces protocoles assurent une résistance au facteur d'échelle vu la minimisation de la taille des états de routage à conserver dans le routeur ainsi que la rapidité d'adhésion à l'arbre.

Mais ces protocoles souffrent du problème de la concentration du trafic autour du point de rendez-vous, de l'utilisation des chemins non-optimaux et de la robustesse dans le cas où le point de rendez-vous tombe en panne.

PIM-SM tient compte de tous ces problèmes et ses mécanismes utilisent les deux techniques d'arbre partagé et d'arbre basé à la source. Des améliorations sur PIM-SM paraissent utiles pour assurer la bidirectionnalité de l'arbre construit par PIM-SM, la minimisation des messages de signalisation entre la source et les destinataires et la réduction de la taille des états à conserver dans les routeurs.

Comme conclusion, d'une part, le protocole PIM-SM paraît une solution élégante pour assurer une transmission *multicast* efficace et robuste à l'intérieur d'un domaine.

D'autre part, les mécanismes du protocole PIM-SM ne représentent pas la solution efficace pour le routage *multicast* entre plusieurs domaines de routage. En effet l'utilisation d'un seul point de rendez-vous pour un groupe paraît une solution inefficace et n'assure pas la résistance au facteur d'échelle lors de l'utilisation de PIM-SM à travers différents domaines de routage.

Nous présentons dans le sous-chapitre suivant deux solutions proposées pour le routage *multicast* inter-domaine.

1.3.3 Les protocoles de routage *multicast* inter-domaine

L'évolution du *multicast* à travers plusieurs domaines a crée le besoin d'un protocole passant à l'échelle, robuste et hiérarchique. En effet, les protocoles présentés dans le sous-chapitre précédent ne résolvent pas les problèmes de routage *multicast* entre différents domaines.

Pour DVMRP, les ressources demandées pour assurer la transmission *multicast* vont dépasser à terme les capacités des routeurs. Pour résoudre ce problème, une version hiérarchique de DVMRP a été proposée [TD95].

MOSPF propose aussi un modèle pour le routage entre domaines. Cependant, les protocoles en mode dense ne sont pas adaptés aux groupes ayant des membres épars, donc les efforts portent principalement sur l'adaptation du protocole PIM-SM, fonctionnant en mode épars, au traitement du routage inter-domaine.

Idéalement, un réseau *multicast* doit être divisé en plusieurs domaines de routage. Chaque domaine exécute son propre protocole de routage *multicast*. Un autre protocole ou un autre exemplaire du même protocole, est utilisé pour le routage entre différents domaines.

Nous allons étudier les problèmes de routage *multicast* inter-domaine en présentant deux solutions : la première solution utilisant les protocoles PIM-SM, MBGP et MSDP, et la deuxième solution utilisant le protocole BGMP. Nous verrons que la deuxième solution exige un protocole d'allocation d'adresse strict comme MASC.

1.3.3.1 La solution MBGP/MSDP/PIM-SM

BGP-4 [RL95] est un protocole pour IPv4 de routage *unicast* capable de transmettre les informations de routage *unicast* d'un domaine à un autre.

BGP-4, utilise le concept CIDR (*Classless InterDomain Routing*) [FLYV93] et assure l'agrégation des chemins.

MBGP [BCKR98], une extension de BGP-4, introduit de nouveaux attributs afin de porter l'information de routage pour différents protocoles.

De manière similaire au comportement de BGP dans le cas du routage *unicast*, MBGP fournit les informations d'accessibilité des différents liens et réseaux *multicast* d'un domaine. Tandis que BGP décrit la topologie *unicast* d'un domaine, MBGP peut être utilisé pour décrire la topologie *multicast*. La différence entre les routes MBGP et celles de BGP se caractérise par la façon dont ses routes sont utilisées : BGP utilise ses routes pour décider de l'interface vers laquelle les paquets doivent être envoyés. En revanche, les routes MBGP *multicast* sont utilisées pour exécuter le contrôle de l'interface RPF, c'est-à-dire pour décider des interfaces sur lesquelles les paquets ne doivent pas être diffusés.

Bien que MBGP est considéré comme étant la première étape vers l'inter-domaine *multicast*, il ne peut pas être considéré tout seul comme étant une solution complète. En effet, MBGP est capable de déterminer le prochain routeur vers une destination mais il n'est pas capable de construire l'arbre *multicast*.

Pour construire l'arbre de routage *multicast*, un protocole de routage inter-domaine est nécessaire.

PIM-SM, en tant que protocole de routage épars, est supposé être une bonne solution pour établir un arbre *multicast* entre les domaines. La proposition est d'avoir un seul RP par domaine pour tous les groupes et les sources appartenant à ce domaine. Cette proposition paraît logique afin d'assurer l'indépendance des mécanismes de routage *multicast* dans un domaine de ceux existants dans les autres domaines. Il est logiquement inacceptable que les destinataires adhérant à un groupe appartiennent à un domaine tandis que leur RP appartiendrait à un autre domaine où il n'y aurait aucun

membre de ce groupe. Cependant, le RP d'un domaine doit s'informer des sources se trouvant dans les autres domaines. Le problème se pose lorsque les membres d'un groupe sont dispersés à travers plusieurs domaines. Un arbre de transmission *multicast* est construit dans chaque domaine. Un mécanisme de fusion des différents arbres est donc nécessaire.

MSDP (*Multicast Source Discovery Protocol*) [FM03], est le protocole chargé de résoudre ce problème. MSDP utilise un routeur unique (généralement le RP appelé $RP/MSDP$) par domaine comme un représentant du domaine chargé d'annoncer aux autres domaines du réseau l'existence de sources actives dans ce domaine à travers des messages périodiques nommés SA (*Source Active*). Des sessions MSDP sont configurées entre les domaines et TCP est le protocole utilisé pour l'échange fiable des messages de MSDP. La construction d'un arbre inter-domaine *multicast* est ainsi déclenchée (*cf.* figure 1.13).

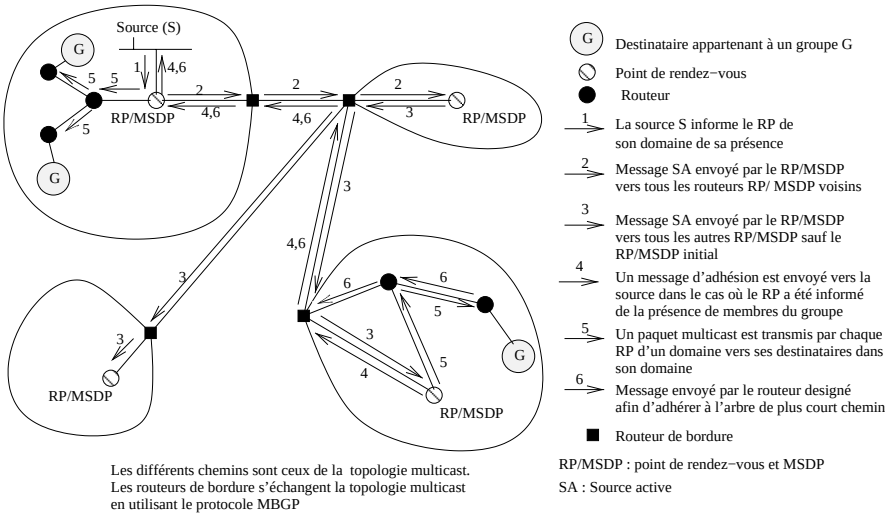


FIG. 1.13 – *Le mécanisme d'exécution du protocole MSDP.*

Le problème de résistance au facteur d'échelle constitue l'un des inconvénients de cette solution. En effet, avec la croissance du nombre de sources *multicast*, le nombre de messages SA devient très grand. L'autre inconvénient est celui des groupes dynamiques. En effet, deux problèmes peuvent être associés aux sources/groupes dynamiques, celui de la latence d'adhésion et celui de sources *bursty* décrits ci-dessous.

Le problème de la latence d'adhésion provient des messages SA qui sont envoyés périodiquement par le $RP/MSDP$. Un certain délai peut exister entre l'instant d'adhésion des nouveaux membres et l'instant de la réception du message SA. Pour résoudre ce problème, les routeurs $RP/MSDP$ peuvent être configurés afin de conserver les messages SA en cache. Un routeur $RP/MSDP$ sans cache peut demander à un routeur $RP/MSDP$ voisin avec cache de lui envoyer le message SA[3] qui correspond à la source. L'inconvénient de cette solution est l'état supplémentaire et la complexité de la gestion du cache.

Les sources *bursty* sont des sources caractérisées par l'envoi d'une rafale de paquets séparés par des périodes de silence de l'ordre de quelques minutes. Un problème se présente lors de la construction de l'arbre *multicast* pour ce genre de sources. En effet, une source transmet des paquets vers le RP qui déclenche un message SA en utilisant MSDP. Par la suite, les $RP/MSDP$ dans les autres domaines vont envoyer des messages d'adhésion vers la source. Mais, puisque la transmission des messages SA et la construction de l'arbre de diffusion par les $RP/MSDP$ vont prendre du temps, la rafale initiale peut ne pas atteindre les destinataires dans les autres domaines. Une fois l'arbre construit, tous les paquets sont normalement transmis et rapidement reçus. Le problème apparaît si la période de silence entre les rafales dépasse la valeur du temporisateur[4] associé à l'état de routage dans les routeurs (typiquement 3 minutes) [Alm00]. Aucun paquet d'une source *bursty* n'atteindra plus les destinataires. Pour ce problème, MSDP propose d'encapsuler les premiers paquets et de les envoyer avec les messages SA.

Pour ces raisons MBGP/PIM-SM/MSDP peut être considérée comme une solution à court terme (qui ne passe pas à l'échelle) et complexe du problème *multicast* inter-domaine. BGMP/MASC présente un premier essai de solution à long terme du problème inter-domaine *multicast*.

3. Le message SA existe déjà dans le cache du $RP/MSDP$ voisin.

4. À l'échéance de ce temporisateur, c'est-à-dire après une certaine durée d'inactivité du groupe, les états de routage sont supprimés.

1.3.3.2 La solution BGMP/MASC

La solution BGMP/MASC [SKTA+98] est constituée de deux protocoles : BGMP (*Border Gateway Multicast Protocol*) et MASC (*Multicast Address-Set Claim*). Nous présentons dans ce sous-chapitre ces deux protocoles.

Le protocole MASC Le protocole MASC [REG+00] forme la base d'une architecture hiérarchique d'allocation d'adresses. Un ou plusieurs nœuds (typiquement les routeurs de bordure) dans un domaine utilisent le protocole MASC pour allouer des intervalles d'adresses spécifiques au domaine. Le serveur d'allocation d'adresses (MAAS, *Domain Multicast Address Allocation Server*) dans un domaine utilise cet intervalle d'adresses pour les applications originaires du domaine.

MASC construit une hiérarchie de domaines : par exemple, les réseaux d'un campus considèrent un réseau régional comme parent, les réseaux régionaux considèrent les réseaux dorsaux comme parents.

MASC assigne dynamiquement les intervalles d'adresses pour les domaines en utilisant l'approche *listen and reclaim with collision detection*. En effet, les domaines fils apprennent les intervalles des adresses *multicast* sélectionnés par leur parents, et sélectionnent des sous-intervalles de ceux de leurs parents, puis propagent ces sous-intervalles vers leurs frères (*siblings*).

Les *claimers* attendent une période assez longue pour détecter une collision [5], avant de communiquer (sous forme de routes de groupes), en utilisant MBGP, les intervalles acquis vers les MAAS des domaines. Les différents MAAS peuvent ainsi attribuer des adresses *multicast* aux applications ayant des groupes (initiés) dans leur domaine.

Considérons la hiérarchie MASC de la figure 1.14. Le domaine B essaye d'allouer l'intervalle 131.0.1.0/24, alors que le domaine C a déjà alloué cet intervalle. Dans ce cas une collision survient et B doit essayer d'allouer un nouvel intervalle, par exemple 131.0.128.0/24. Le MAAS du domaine B, après avoir alloué le nouvel intervalle doit

5. C'est le délai avant lequel on n'est pas sûr si l'allocation est sans collision.

attendre une période assez longue (de l'ordre de 48 heures [SKTA$^+$98]) avant de communiquer cet intervalle vers les MAAS des domaines.

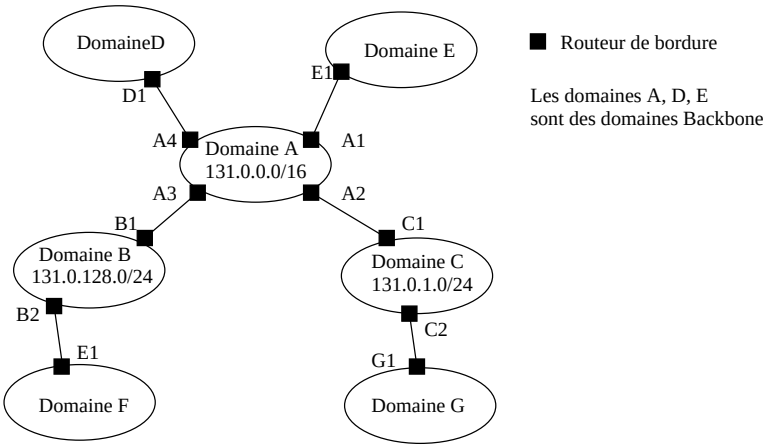


FIG. 1.14 – *L'allocation d'adresse en utilisant MASC.*

Une fois que le domaine B possède un intervalle d'adresse unique, cet intervalle est distribué comme route de groupe vers tous les routeurs de bordure du domaine B. Par exemple, la route de groupe est sauvegardée dans la base de données G-RIB (*Group Routing Information Base*) de $A3$ comme (131.0.128.0/24, $B1$) indiquant ainsi que $B1$ est le prochain routeur pour $A3$ pour atteindre le domaine racine de l'intervalle 131.0.128.0/24. Les autres routeurs de bordure du domaine A ($A1$, $A2$, $A4$) conservent l'information (131.0.128.0/24, $A3$) dans leur G-RIB indiquant que $A3$ est le prochain routeur pour atteindre le domaine racine de l'intervalle 131.0.128.0/24.

Lorsqu'un routeur de bordure X annonce une route de groupe R vers un routeur de bordure Y cela signifie que Y peut utiliser X comme le prochain routeur pour transmettre les paquets vers le domaine racine de l'intervalle d'adresse représenté par R.

Ainsi, BGMP peut utiliser les informations de la base de données G-RIB pour construire l'arbre *multicast* inter-domaine pour chaque groupe.

Le protocole BGMP Les routeurs de bordure emploient BGMP pour faciliter l'utilisation du service *multicast* à travers différents domaines. BGMP utilise les routes

de groupe de MBGP pour construire des arbres bidirectionnels pour chaque groupe *multicast*.

Chaque arbre associé à un groupe est enraciné dans le domaine dont l'intervalle d'allocation d'adresse contient l'adresse du groupe. Ce domaine qui a alloué un intervalle d'adresse est ainsi le domaine racine pour tous les groupes dont l'adresse est dans cet intervalle.

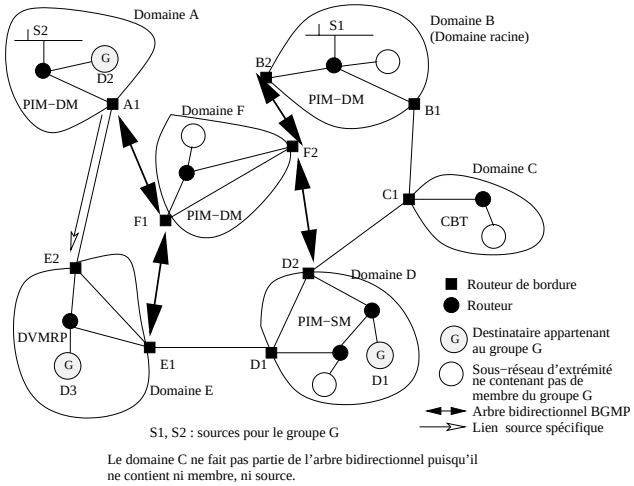


FIG. 1.15 – *Le mécanisme du protocole BGMP.*

Une des fonctions de BGMP est de décider dans quel domaine particulier l'arbre partagé doit être enraciné. BGMP est basé sur la notion de l'indépendance entre les domaines et ceci en utilisant une allocation stricte d'adresse afin d'éviter la collision des adresses. Cette allocation d'adresse permet à chaque domaine de posséder des intervalles spécifiques d'adresses. Le domaine racine d'une adresse *multicast* est celui qui l'a réclamé en appliquant un protocole d'allocation d'adresse (comme MASC). Les routeurs de bordure BGMP se connectent entre eux en utilisant le protocole de transport TCP.

Lorsque les appartenances aux groupes changent, les routeurs de bordure envoient des messages de mise à jour d'adhésion/d'élagage des uns aux autres.

Puisque le plus court chemin d'une source *multicast* à un destinataire peut être différent du chemin imposé par l'arbre partagé, BGMP permet (comme PIM-SM) de

construire des liens spécifiques aux sources si ce chemin ne coïncide pas avec le chemin de l'arbre partagé (*cf.* figure 1.15). Dans l'exemple de la figure 1.15, supposons que le plus court chemin *unicast* entre le domaine A et le domaine D passe par le lien $A1$-$E2$ et donc l'interface RPF pour le routeur $E2$ par rapport à A est celle menant à $A1$. Puisque le domaine E utilise DVMRP comme protocole de routage intra-domaine, le routeur de bordure $E1$ doit encapsuler les paquets arrivant sur l'arbre bidirectionnel BGMP et les transmet vers le routeur de bordure $E2$ afin de respecter l'interface RPF dans le domaine. $E2$ peut initier dans ce cas la construction d'un lien du plus court chemin de $E2$ vers $A1$. $E1$ envoie un message d'élagage vers $F1$ indiquant sa volonté de ne plus recevoir des paquets à travers l'arbre bidirectionnel. Le message est ainsi transmis vers le domaine racine du groupe (le domaine B est supposé le domaine racine de cet arbre).

1.3.3.3 Conclusion

Les deux solutions "BGMP/MASC" et "MBGP/MSDP/PIM-SM" souffrent des problèmes de résistance au facteur d'échelle ainsi que de problèmes de complexité et de sécurité. En effet, le modèle IP *multicast* actuel utilise une adresse *multicast* (classe D) pour représenter un groupe de destinataires quelconque. De plus, il n'existe pas de mécanismes servant à estimer la taille d'un groupe *multicast* ou à interdire les sources non autorisées. La combinaison de MBGP, de PIM-SM et de MSDP peut être considérée comme une solution à court terme du problème *multicast* inter-domaine, mais qui ne passe pas à l'échelle et complexe. Par ailleurs, BGMP nécessite l'utilisation d'un protocole comme MASC afin d'assurer l'allocation unique globale des adresses *multicast*. Les protocoles utilisés pour résoudre les problèmes d'allocation d'adresse, notamment *multicast* sont considérés comme complexes.

Les efforts portent à la fois sur l'amélioration de protocoles existants et sur la réalisation, de nouveaux protocoles à l'aide de nouveaux services comme Xcast et SSM, qui introduisent des changements importants au modèle *multicast* actuel.

1.4 Les nouveaux services

Le protocole EXPRESS (*Explicitly Requested Single Source*) [HC99] a introduit le modèle de canal : Un canal est identifié par le couple (S, G) où S est l'adresse *unicast* de la source et G est une adresse de groupe IP classe D. La notion de canal élimine la nécessité d'une allocation d'adresse *multicast* globale. La source S peut transmettre des paquets uniquement vers le canal (S, G) puisque les destinataires souscrivant à (S, G) ne sont pas abonnés à (S', G). Une adresse *multicast* globale G n'est plus nécessaire.

Le service SSM (*Source Specific Multicast*) [Bha03, HC03], basé aussi sur la notion de canal, est dérivé du protocole EXPRESS. L'arbre construit par SSM est un arbre basé à la source et appelé arbre source spécifique. L'arbre source spécifique est un arbre des plus courts chemins ce qui simplifie la méthode de construction de l'arbre. Les adresses 232/8 (de 232.0.0.0 à 232.255.255.255) sont réservées pour le protocole SSM [IANA01].

PIM-SM peut être adapté pour supporter le service SSM en utilisant son mécanisme d'arbre basé à la source. Dans une première phase le protocole PIM-SM utilise l'arbre partagé enraciné en un point de rendez-vous pour connaître les sources actives voulant transmettre des paquets vers les destinataires du groupe. Dans une deuxième phase si la quantité de données transmises par une source *multicast* dépasse un certain seuil, PIM-SM tend à utiliser le plus court chemin entre les destinataires et cette source. En revanche, dans SSM, un destinataire peut sélectionner certaines sources dont il veut recevoir les paquets au lieu de recevoir tous les paquets transmis par toutes les sources du groupe auquel il appartient.

IGMPv3 [CDT02] utilise aussi la notion de canal mais des modifications sur IGMPv3 doivent être introduites pour l'appliquer à SSM dans l'intervalle d'adresses 232/8. PIM-SM doit construire directement l'arbre du plus court chemin pour les applications ayant l'adresse du groupe *multicast* dans l'intervalle 232/8. Ceci amène des modifications sur PIM-SM aussi. Enfin, la quantité de données à mémoriser par les routeurs le long du chemin entre la source et les destinataires paraît coûteuse si les destina-

taires membres d'un groupe ne sont pas nombreux, d'où l'idée du service Xcast (*Explicit Multicast*) [BFI+03] qui peut servir un grand nombre de groupes ayant peu de membres. Cette idée sera détaillée dans le chapitre suivant.

1.5 Conclusion

Les protocoles de routage *multicast* actuels souffrent de plusieurs limitations. Les protocoles de routage *multicast* en mode épars ont des avantages sur ceux en mode dense. Premièrement, les protocoles en mode épars fournissent une résistance au facteur d'échelle en terme de nombre d'états de routage *multicast* à gérer. Les protocoles en mode épars exigent l'existence d'un état de routage *multicast* dans tous les routeurs appartenant à l'arbre construit pour un groupe tandis que les protocoles en mode dense exigent l'existence d'un état de routage *multicast* pour chaque couple (source,groupe). Deuxièmement, les protocoles en mode épars sont beaucoup plus efficaces car l'utilisation de messages d'adhésion explicites signifie que le trafic *multicast* ne s'écoule que sur les liens qui mènent à des destinataires qui ont demandé explicitement leur adhésion au groupe.

Mais les protocoles de routage en mode épars ont aussi des inconvénients généralement reliés à l'utilisation des points de rendez-vous qui peuvent tomber en panne alors qu'ils sont des points critiques. Le trafic est de plus concentré autour du point de rendez-vous. Finalement, le trafic transmis par une source vers le point de rendez-vous et ensuite vers les destinataires suit un chemin non-optimal. De plus, un destinataire est obligé de recevoir tous les paquets transmis vers le groupe par toutes les sources.

Un autre problème majeur avec les modèles existants est celui de l'allocation d'adresse. Ceci présente un problème majeur pour les protocoles de routage *multicast* inter-domaine. Actuellement, il y a un risque majeur d'avoir une collision d'adresses entre différentes applications. Les solutions proposées pour résoudre ce problème ne répondent pas aux exigences des applications vu la nature dynamique de l'allocation d'adresse *multicast*.

Les efforts portent à la fois sur l'amélioration de protocoles existants et sur la réalisation, de nouveaux protocoles à l'aide de nouveaux services comme Xcast et SSM, qui introduisent des changements importants au modèle *multicast* actuel. Ces protocoles sont en cours d'étude pour résoudre les limitations des protocoles existants et afin d'assurer une transmission *multicast* efficace, robuste et pouvant passer à l'échelle.

Chapitre 2

Le protocole GXcast

Un unique protocole de routage *multicast* est incapable de servir les différents types d'applications *multicast* [Sit03]. En effet, le nombre d'applications *multicast* ayant souvent des exigences multiples voire contradictoires ne cesse pas de croître. Les protocoles de routage *multicast* doivent présenter une certaine flexibilité selon les besoins des différentes applications. Dans un réseau où il y a un très grand nombre de groupes *multicast* de petite taille (*SGM : Small Group Multicast* [Oom00]) dont les destinataires sont largement dispersés, le modèle de routage *multicast* traditionnel ne convient pas [DHPP00].

Pour palier aux problèmes de passage à l'échelle en terme de nombre de groupes *multicast*, plusieurs mécanismes *multicast* ont été récemment proposés. Parmi ces propositions, un protocole de routage *multicast* explicite, le protocole Xcast. De tels protocoles encodent la liste complète des membres du groupe dans l'en-tête de chaque paquet.

Après avoir décrit le protocole Xcast et son extension, le protocole Xcast+, et avoir rappelé les avantages et inconvénients des protocoles de routage *multicast* explicite par rapport aux protocoles de routage traditionnel, nous montrons que le protocole Xcast ne supporte pas la fragmentation IP des paquets. Nous proposons un nouveau protocole nommé GXcast, extension de Xcast conçue pour éviter la fragmentation. Le protocole GXcast est basé sur un paramètre dont le choix influence les performances globales.

Nous analysons l'impact de ce paramètre sur les performances de ce protocole selon plusieurs critères et nous validons par des simulations. Par la suite, nous présentons des améliorations possibles du protocole GXcast et nous concluons finalement sur le fait que le protocole GXcast semble adapté aux groupes *multicast* de taille petite à moyenne.

2.1 Les protocoles Xcast et Xcast+

Les protocoles de routage *multicast* explicite semblent être une solution aux problèmes de passage à l'échelle des protocoles de routage *multicast* traditionnel. Le protocole Xcast proposé par Boivie *et al.* ainsi que son extension nommée Xcast+ sont deux protocoles de routage *multicast* explicite. Nous allons les décrire dans ce paragraphe.

2.1.1 Le protocole Xcast

Explicit Multicast (Xcast) [BFI+03] a été proposé pour résoudre le problème de passage à l'échelle des protocoles de routage *multicast* traditionnel et pour servir des groupes ayant peu de membres tout en minimisant la consommation de la bande passante. La source encode explicitement la liste des destinations dans l'en-tête Xcast d'un paquet au lieu d'utiliser une adresse *multicast*[1] et envoie le paquet vers son routeur désigné (c'est-à-dire le routeur Xcast en charge de cette source). Chaque routeur du chemin analyse l'en-tête Xcast, classe les destinations selon leur prochain routeur au sens *unicast* et envoie une copie[2] vers chacun des prochains routeurs. Dans le cas où il reste une seule destination dans la liste, le paquet Xcast est transformé par l'algorithme X2U (Xcast à *unicast*) en un paquet *unicast*.

1. Tous les routeurs Xcast du réseau appartiennent à un groupe dont l'adresse de destination est : `All_Xcast_Routers`.
2. Cette copie peut être légèrement modifiée.

Exemple : considérons le groupe représenté sur la figure 2.1, comportant une source S et six destinataires D_1, D_2, D_3, D_4, D_5 et D_6. La source S envoie un paquet Xcast contenant la liste des destinations $(D_1, D_2, D_3, D_4, D_5, D_6)$ à R_1. R_1 traite ce paquet comme un paquet Xcast quelconque : R_2 est le prochain routeur sur chacun des chemins *unicast* de R_1 à D_i, donc le paquet Xcast entier est transmis à R_2.

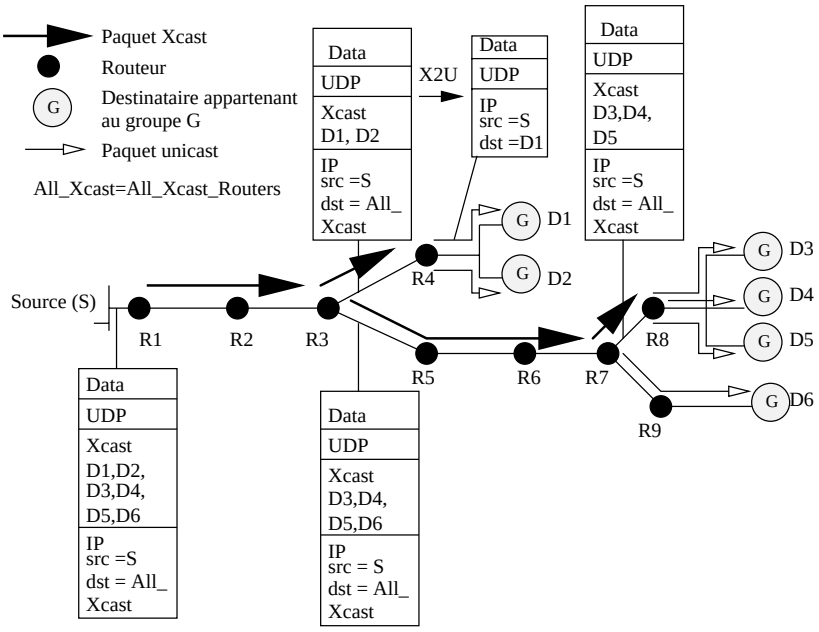


FIG. 2.1 – *Un exemple de la transmission d'un paquet dans Xcast.*

R_2 envoie à son tour le paquet à R_3. Lorsque R_3 reçoit le paquet, il en envoie une copie au routeur R_4 avec dans l'en-tête Xcast la liste (D_1, D_2) et une copie au routeur R_5 avec dans l'en-tête Xcast la liste $(D3, D4, D5, D6)$. R_4, recevant le paquet qui lui est destiné, enverra à son tour à D_1 le message Xcast contenant la liste (D_1) et à D_2 le message Xcast contenant la liste (D_2). D_1 pourra extraire du message qu'il recevra les données qui lui sont utiles. Le comportement est similaire pour les routeurs R_5 à R_9 et pour les cinq autres destinataires.

2.1.2 Le protocole Xcast+

Xcast+ [MYKS01] a été proposé pour résoudre le problème de passage à l'échelle de Xcast pour les groupes de taille moyenne. Il est basé sur le protocole Xcast (il utilise d'ailleurs un en-tête très similaire) et permet de réduire la liste des destinataires en utilisant habilement le protocole de gestion des groupes d'Internet (IGMP) [CDT02]. En effet, un destinataire désirant faire partie du groupe (S, G) (Un groupe est identifié par le canal (S, G) où S est l'adresse de la source et G l'adresse du groupe) émet un message *join* IGMP à destination du groupe (S, G). Quand le routeur désigné (DR : *designated router*) associé au destinataire reçoit ce message, il envoie à la source S un message de demande d'enregistrement Xcast+ contenant l'adresse de la source S, l'adresse du groupe G, et sa propre adresse DR. Lorsque le DR associé à la source reçoit ce message, il maintient les adresses de tous les routeurs DR ayant des destinataires appartenant au groupe (S, G).

Lorsque la source envoie des paquets *multicast*, le DR de la source crée un paquet Xcast+ dans lequel il encode explicitement la liste des DR associés aux destinataires dans l'en-tête Xcast+, il complète le paquet avec les données à envoyer et émet ce paquet vers le(s) prochain(s) routeur(s) concerné(s) (M2X : *multicast* à Xcast+). Le chemin suivi par le paquet Xcast+ est le même que celui suivi par le paquet Xcast. La seconde différence a lieu aux DR destinataires : les paquets Xcast+ qui leur parviennent sont convertis en paquets *multicast* et envoyés aux réseaux dont les DR ont la charge (X2M : Xcast à *multicast*).

Exemple : prenons le même exemple que celui du paragraphe 2.1.1 : considérons le groupe représenté sur la figure 2.2, comportant une source S et les six destinataires D_1, D_2, D_3, D_4, D_5 et D_6. S envoie un paquet *multicast* ayant G comme adresse de destination. Ce message parvient au routeur désigné de son réseau, R_1 dans notre cas. Lorsque R_1 reçoit ce paquet, il génère (en utilisant l'algorithme de transformation M2X) un paquet Xcast+ avec la liste de destinations (R_4, R_8, R_9) dans son en-tête Xcast+ et il applique alors au paquet l'algorithme d'émission Xcast+. Cet algorithme est similaire à celui de Xcast à une nuance près : quand un routeur désigné, par exemple

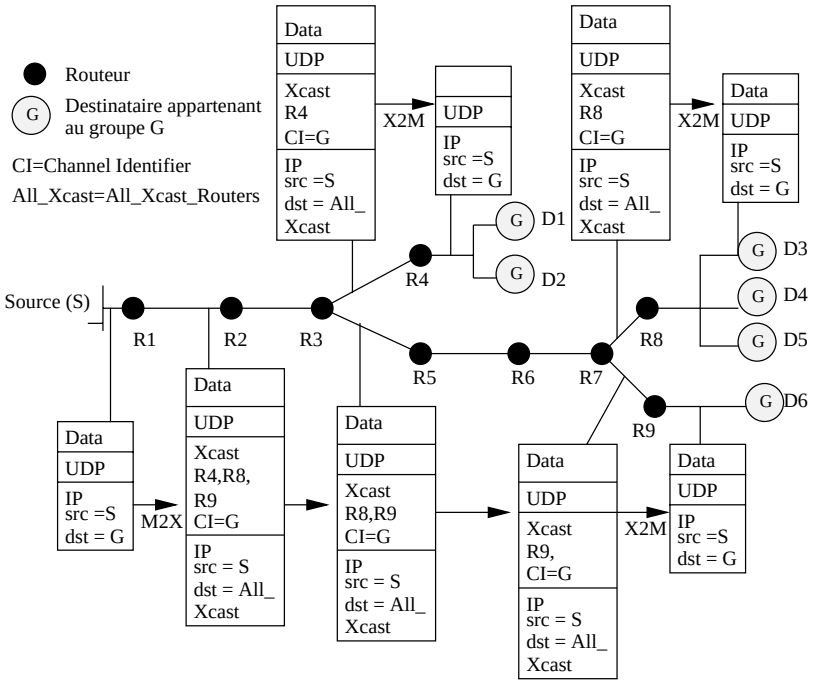


FIG. 2.2 – *Un exemple de la transmission d'un paquet dans Xcast+.*

R_4, reçoit un paquet qui lui est destiné, il applique à ce paquet l'algorithme de transformation X2M et envoie le paquet *multicast* ainsi généré aux réseaux dont il est le routeur désigné.

Alors que Xcast ne permet que de gérer efficacement les petits groupes, Xcast+ permet la gestion de groupes *multicast* de taille moyenne. En effet, le facteur limitant est principalement le nombre d'entrées dans la liste des destinataires, et cette liste est réduite dans Xcast+ [MYKS01]. Ce que nous montrerons dans les deux sous-chapitres suivants.

2.1.3 Les avantages et les inconvénients de la technique Xcast

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

L'utilisation de Xcast présente des avantages et introduit quelques inconvénients

par rapport aux protocoles de routage *multicast* traditionnel :

2.1.3.1 Les avantages de la technique Xcast

Gestion des états de routage et des messages de signalisation : une des principales caractéristiques du protocole Xcast est qu'il élimine d'une part les états de routage *multicast* au sein des routeurs [3] et d'autre part la nécessité de mettre en place un mécanisme de signalisation spécifique au routage *multicast* entre les différents routeurs. Il élimine de plus le besoin de protocoles de routage *multicast* intra-domaines et inter-domaines. Cette caractéristique lui permet d'être capable de gérer de nombreux groupes simultanément.

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

Outre ces deux avantages, on peut citer les deux suivants :

- Xcast ne nécessite pas de mécanisme complexe d'allocation d'adresses *multicast* puisqu'un groupe est identifié par le canal (S, G) et non pas par G uniquement.
- Xcast ne nécessite pas que les chemins soient symétriques puisqu'il se base entièrement sur les chemins *unicast*.

2.1.3.2 Les inconvénients de la technique Xcast

Fragmentation des paquets Xcast : pour des raisons techniques, la taille d'un paquet sur un lien est limitée à une valeur appelée l'unité de transmission maximale

3. Xcast utilise uniquement les états de routage *unicast* présents dans les routeurs.

(MTU) du lien. Le protocole IP [Pos81] est pourvu d'un mécanisme appelé *fragmentation* qui rend cette limitation transparente pour les équipements terminaux. Un paquet IP circulant sur un réseau peut être amené à être fragmenté si sa taille dépasse la capacité du lien qu'il doit emprunter. La fragmentation est un mécanisme IP qui tronque un paquet en plusieurs paquets IP autonomes, donc chacun munis d'un en-tête IP valide, et qui partage les données parmi ces paquets. Plus précisément, lorsqu'un routeur reçoit un paquet, il regarde dans sa table de routage vers quel prochain routeur et donc sur quel lien il doit acheminer le paquet. Une fois le lien identifié, le routeur contrôle que le MTU du lien n'est pas dépassé. Si c'est le cas et à moins que le paquet interdise explicitement sa fragmentation, le routeur coupe le paquet en fragments respectant les contraintes suivantes :

- chaque fragment résultant est un paquet IP autonome, avec notamment un en-tête IP valide,
- chaque fragment résultant de la fragmentation a une taille inférieure ou égale au MTU du lien,
- les données du paquet initial sont réparties dans les différents fragments.

L'algorithme utilisé pour fragmenter des paquets IPv4 est décrit dans [Pos81]. Bien que le protocole IPv6 tente d'éviter la fragmentation, il possède toutefois un mécanisme de fragmentation décrit dans [DH98].

La figure 2.3 montre l'effet qu'aurait une fragmentation sur un paquet Xcast. On peut voir que seul le premier paquet résultant est un paquet Xcast valide, puisqu'il est le seul paquet à contenir un en-tête Xcast (il ne comporte cependant aucune données provenant de l'application). Les trois autres paquets ne seront pas traités comme de paquets Xcast : ils ne passeront pas à l'ensemble des destinataires du groupe. Pour empêcher à un paquet Xcast d'être fragmenté, le champ DF (*Don't Fragment*) de son en-tête IP doit être positionné à 1. Dans ce cas, si le MTU du lien situé sur le chemin qui doit être emprunté par un paquet Xcast est inférieur à la taille du paquet, le paquet est détruit.

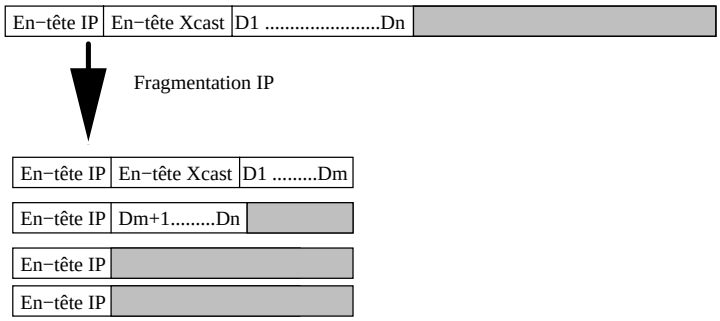


FIG. 2.3 – *Le problème de fragmentation d'un paquet Xcast.*

Afin d'atteindre les destinataires, la source peut limiter la taille de ses paquets à 576 octets qui est le MTU de minimum garanti par IPv4 sur un lien. Cette taille limite le nombre de destinataires pour un groupe Xcast à 135[4]. Cependant, le nombre de destinataires d'un groupe dans Xcast dépend d'un champ de 7 bits, ce qui limite la taille d'un groupe à 127 destinataires. Dans IPv6, puisque le MTU minimum est de 1280 octets et puisque la taille d'une adresse IPv6 est de 16 octets, la taille d'un groupe Xcast est limitée à 74 destinataires. Nous considérons que ces limites sont trop restrictives. Ce que nous proposons avec le protocole GXcast est un mécanisme simple pour éliminer ces limitations sur la taille des groupes Xcast.

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

Changement des en-têtes des paquets à acheminer : un troisième problème est qu'à un paquet Xcast arrivant, un routeur Xcast doit émettre des copies du paquet Xcast dont les en-têtes sont différents. En effet, la liste des destinataires est différente pour chacun des prochains routeurs. Ce problème est atténué par la proposition d'utiliser une table d'indicateurs (BITMAP) permettant de n'effectuer qu'un minimum de

4. Cette valeur sera étudiée dans le sous-chapitre 2.2.3.

modifications dans l'en-tête afin de réduire l'impact sur le traitement et sur le calcul de la somme de contrôle des paquets à émettre. Cela à l'opposé a comme conséquence de conserver inutilement la liste longue.

2.1.3.3 Les critiques de la technique Xcast

Utilisation d'*unicast* plutôt que du protocole Xcast : si le nombre de destinataires d'un groupe *multicast* est très limité, il est possible de servir en *unicast* les destinataires de ce groupe[5]. Cette technique n'est pas toujours aisée à mettre en place, notamment lorsque la limitation de bande passante en «dernier mille» (*last mile*) rend nécessaire l'utilisation du *multicast* [BFI+03].

2.2 Une généralisation de la technique Xcast

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

2.2.1 La description du protocole GXcast

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

Exemple : considérons le groupe représenté sur la figure 2.4, comportant une source S et six destinataires D_1, D_2, D_3, D_4, D_5 et D_6. Dans un premier temps, décrivons le

5. Un paquet *unicast* sera envoyé à chaque destinataire du groupe.

procédé d'adhésion d'un membre à un groupe. Chacun des destinataires envoie un message *join* IGMP qui parvient au routeur désigné de leur sous-réseau. Dans notre exemple, R_4, R_8 et R_9 reçoivent la demande d'adhésion IGMP. Ces derniers envoient alors des messages d'adhésion vers la source S (message que R_1 peut intercepter) qui ajoute effectivement le membre dans la liste des routeurs destinataires. À présent, étudions l'algorithme d'émission de la source. Pour couvrir les n membres en limitant le nombre de destinataires par paquet à n_M, il faut envoyer $\lceil \frac{n}{n_M} \rceil$ paquets. Prenons comme exemple le cas où n_M vaut 2 et plaçons nous dans le cas du protocole Xcast+ qui considère $n = 3$ membres[6]. Le nombre de paquets à générer va être égal à 2, le premier paquet GXcast étant à destination des membres de la liste (R_4, R_8) et le second à destination du membre de la liste (R_9)[7]. Ces paquets GXcast peuvent être considérés comme des paquets Xcast indépendants et subissent donc un traitement similaire.

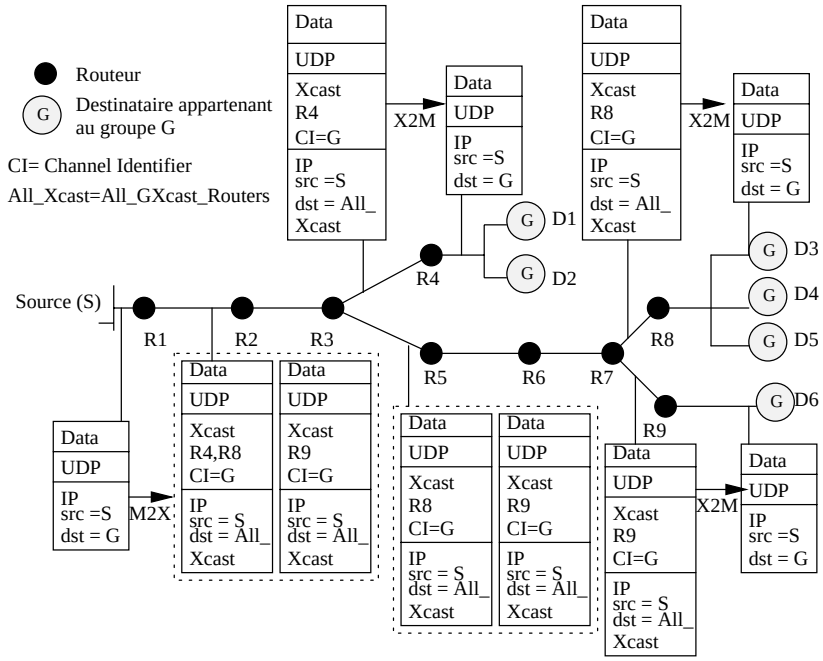


FIG. 2.4 – *Un exemple de la transmission d'un paquet dans GXcast.*

6. Ces petites valeurs de n_M et de n ne sont données qu'à titre indicatif. Les valeurs réalistes de n_M seront étudiées ultérieurement. On peut néanmoins noter que, pour des petits groupes, GXcast a toujours un comportement identique à Xcast+.

7. Une amélioration pour mieux choisir les sous-listes sera décrit dans le sous-chapitre 2.3.2.

2.2.2 Le format d'un paquet GXcast

Un paquet GXcast est très similaire à un paquet Xcast (*cf.* annexe A). Décrivons d'abord brièvement le format d'un paquet Xcast.

Un paquet Xcast est constitué de trois parties : un en-tête IP, un en-tête Xcast et une partie données. La source de l'en-tête IP est la source du groupe *multicast*, la destination de l'en-tête IP est l'adresse `All_Xcast_Routers`, le type de protocole de l'en-tête IP est `XCAST_PROTO` et le drapeau DF (*Don't Fragment*) est positionné. Un champ Destination et un champ Protocol sont inclus dans l'en-tête Xcast pour indiquer respectivement l'adresse de groupe et le protocole suivant l'en-tête Xcast (traditionnellement UDP).

Un paquet GXcast est lui aussi constitué de trois parties : un en-tête IP, un en-tête GXcast et une partie données (*cf.* annexe B). La différence entre un paquet GXcast et un paquet Xcast est minime :

- Le champ Destination de l'en-tête IP est l'adresse `All_GXcast_Routers`.
- Le type de protocole de l'en-tête IP est `GXCAST_PROTO`.
- Le champ indiquant le nombre de destinataires dans un paquet GXcast doit pouvoir permettre de stocker un grand nombre de destinataires. Il semble que la limitation à 7 bits proposée dans Xcast soit insuffisante (en effet, le calcul du nombre maximal de destinataires n_{max} effectué dans le paragraphe suivant aboutit à une valeur de 135).
- Le champ indiquant la taille de l'en-tête GXcast doit lui aussi être suffisamment grand.

Il semble raisonnable de considérer qu'un en-tête GXcast peut être stocké sur 12 octets[8]. Combiné à l'en-tête IP dont la taille est de 20 octets, la taille totale des deux en-têtes sans la liste des destinataires de GXcast atteint $E = 32$ octets[9].

8. Rappelons que l'en-tête Xcast est codé sur 12 octets.
9. Elle est de $E = 64$ pour le protocole IPV6.

2.2.3 Les liens entre GXcast et l'unité de transfert maximale

L'unité de transfert maximale (MTU, *Maximum Transmission Unit*) est la taille du plus grand paquet pouvant circuler sur un chemin sans subir de fragmentation. Comme un paquet GXcast ne peut pas être fragmenté au sens IP, cette taille va limiter le nombre de destinataires par paquet, c'est-à-dire la valeur de n_M. L'étude que nous proposons se base sur le fait que le MTU est connu *a priori*. Par la suite, nous utiliserons la valeur $M = 576$ octets qui est la MTU minimale garantie pour le protocole IPv4[10]. Le dernier paramètre à fixer est $IP = 4$ qui est la taille en octets d'une adresse IPv4[11]. De ces valeurs, on peut calculer le nombre de destinataires maximal par paquet n_{max} de la manière suivante[12] :

$$n_{max} = \lfloor \frac{M - E - 1}{IP} \rfloor = 135.$$

Cette expression intègre le fait qu'un paquet GXcast est censé contenir au moins un octet de donnée[13].

2.2.4 L'étude du paramètre de GXcast

Le comportement du protocole GXcast dépend grandement de la valeur du paramètre n_M. En effet, comme nous le soulignerons dans la suite de ce paragraphe, plusieurs critères sont influencés par le choix de cette valeur. On notera n le nombre de membres dans le groupe et d le volume en octets de données à transférer à ces membres.

10. Elle est de $M = 1280$ octets pour le protocole IPv6.
11. Cette taille est de $IP = 16$ octets pour IPv6.
12. Les bits A, D et P sont tous positionnés à 0 (voir annexes A et B). De plus, nous ne considérons pas dans ce calcul les champs utilisés par le BITMAP. En considérant le BITMAP le n_{max} se réduit à 130 destinataires seulement.
13. Pour IPv6, on aboutit à une valeur $n_{max} = 75$.

2.2.4.1 Le comportement basique

Le comportement le plus simple qu'il est possible d'adopter est de fixer n_M à n_{max}. Ce procédé correspond à une fragmentation : lorsque la taille critique est atteinte, un nouveau paquet est créé avec les membres restants.

Le comportement basique est peu efficace pour des groupes dont la taille est de l'ordre de n_{max}. À titre d'exemple, supposons que le protocole IPv6 soit utilisé et supposons un groupe *multicast* comportant $n = 70$ membres. La source du groupe désire transmettre un volume de $d = 10000$ octets de données. Lorsque $n_M = n_{max}$, 96 octets de données sont disponibles pour chaque message, une taille importante étant occupée par les 70 adresses. Ainsi, 105 paquets seraient nécessaires. Un choix plus approprié de n_M aurait causé la génération de beaucoup moins de paquets. En particulier, choisir $n_M = 37$ laisse 624 octets libres dans chaque paquet GXcast. Puisque deux copies des données doivent être envoyées, l'une à destination des 37 premiers membres et l'autre à destination des 33 suivants, on aboutit à un total de 32 paquets. En choisissant un n_M différent du n_{max}, le nombre de paquets générés a été réduit d'environ trois fois.

2.2.4.2 Le nombre de membres influencés par une défaillance

Si un paquet GXcast est perdu par un routeur, tous les destinataires contenus dans la liste de destination du paquet sont touchés par cette perte. Le nombre de membres influencés par une erreur ponctuelle peut être réduit en choisissant de petites valeurs pour n_M.

2.2.4.3 Le nombre de paquets générés

Pour envoyer d octets de données à n membres d'un groupe, le nombre de paquets $p(n, d, n_M)$ générés en limitant à n_M le nombre de destinataires par paquet est défini par la formule suivante :

$$p(n, d, n_M) = \lceil \frac{n}{n_M} \rceil \lceil \frac{d}{MTU - E - IP * n_M} \rceil. \tag{2.1}$$

La partie gauche de cette équation représente le nombre de sous-listes de taille au plus n_M nécessaires pour couvrir les n membres, c'est-à-dire le nombre de paquets comportant les mêmes données mais à destination de membres différents. La partie droite de cette équation représente le nombre de paquets à générer pour envoyer d octets en considérant la quantité maximale d'octets de données que l'on peut placer dans un paquet qui est à destination de n_M membres. Afin d'étudier le comportement de p en terme de n_M, nous considérerons deux cas: $n \leq n_M$ et $n > n_M$. Dans le premier cas, nous avons : $p(n, d, n_M) = \lceil \frac{d}{MTU - E - n.IP} \rceil$.

Cette expression de p ne dépend pas de n_M. Le protocole GXcast se comporte dans ce cas-ci de la même manière que le protocole Xcast. Dans le deuxième cas où $n > n_M$ et afin d'étudier le comportement de la fonction p, faisons l'hypothèse qu'elle peut être approchée par la fonction $\bar{p}$, définie comme suit :

$$\bar{p}(n, d, n_M) = \frac{n}{n_M} \frac{d}{MTU - E - IP * n_M}.$$

La fonction $\bar{p}$ admet un minimum en :

$$n_M = \frac{MTU - E}{2 * IP} \approx \frac{n_{max}}{2}.$$

Afin de garantir un faible nombre de paquets générés, il est donc important de choisir une valeur de n_M s'approchant de la moitié de n_{max}. Cette valeur a de plus l'avantage d'être indépendante de la taille du groupe n ou de la quantité d de données à envoyer. Nous proposons cette valeur comme valeur par défaut pour le protocole GXcast

2.3 Évaluation et simulation du protocole GXcast

2.3.1 Le taux de surcoût engendré par le protocole GXcast

Le nombre de paquets générés par GXcast va dépendre des trois paramètres n, d et n_M. Le choix de la valeur de n_M a été justifié dans le paragraphe 2.2.4.3. Rappelons qu'un protocole de routage *multicast* traditionnel envoie $p(n,d) = \lceil \frac{d}{MTU-E} \rceil$ paquets pour faire parvenir d octets de données à un groupe de n membres[14].

Nous définissons $\delta p_{GXcast/multicast}$ comme étant le taux de surcoût engendré par le protocole GXcast par rapport à un protocole de routage *multicast* traditionnel et nous distinguons les deux cas : $n \leq n_M$ et $n > n_M$.

$$\delta p_{GXcast/multicast} = \frac{p(n,d,n_M)}{p(n,d)} = \frac{\lceil \frac{n}{n_M} \rceil \lceil \frac{d}{MTU-E-IP*min(n,n_M)} \rceil}{\lceil \frac{d}{MTU-E} \rceil}.$$

Pour $n \leq n_M$:

$$\delta p_{GXcast/multicast} = \frac{p(n,d,n_M)}{p(n,d)} = \frac{\lceil \frac{n}{n_M} \rceil \lceil \frac{d}{MTU-E-n*IP} \rceil}{\lceil \frac{d}{MTU-E} \rceil} = \frac{\lceil \frac{d}{MTU-E-n*IP} \rceil}{\lceil \frac{d}{MTU-E} \rceil}.$$

La figure 2.5 présente le nombre de paquets générés par le protocole GXcast par rapport au nombre de paquets générés par un protocole de routage *multicast* traditionnel pour $n \leq n_M$. On remarque que la valeur du taux de surcoût ne dépasse pas 2 et que dans beaucoup de cas cette valeur est égale à 1, ce qui implique que le surcoût engendré par le protocole GXcast est faible si le nombre de destinataires est inférieur à n_M.

Pour $n > n_M$:

$$\delta p_{GXcast/multicast} = \frac{p(n,d,n_M = \frac{n_{max}}{2})}{p(n,d)} = \frac{\lceil \frac{2*n}{n_{max}} \rceil \lceil \frac{2*d}{2*(MTU-E)-n_{max}*IP} \rceil}{\lceil \frac{d}{MTU-E} \rceil}$$

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

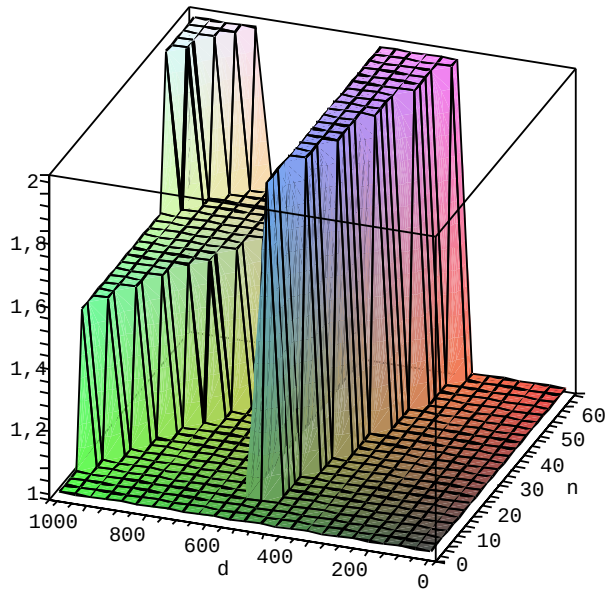


FIG. 2.5 – *Le taux de surcoût du protocole GXcast par rapport à un protocole de routage multicast traditionnel en terme de* n *et* d *pour* $n \leq n_M$.

$$\approx \frac{\lceil \frac{2*n}{n_{max}} \rceil \lceil \frac{2*d}{MTU-E} \rceil}{\lceil \frac{d}{MTU-E} \rceil} \leq 4 * \lceil \frac{n}{n_{max}} \rceil.$$

La figure 2.6 présente le nombre de paquets générés par le protocole GXcast par rapport au nombre de paquets générés par un protocole de routage *multicast* traditionnel pour $n > n_M$.

Nous remarquons d'abord que le taux de surcoût est faible si la taille de la charge utile est inférieure à 250 octets. On remarque de plus que ce taux de surcoût est presque linéaire en fonction de n : pour un groupe constitué de 150 membres, il sera nécessaire d'envoyer environ 5 fois plus de paquets avec le protocole GXcast qu'avec un protocole de routage *multicast* traditionnel. Pour un groupe de 300 membres, il sera nécessaire d'envoyer 10 fois plus de paquets.

Pour transmettre à respectivement $n = 150$ et $n = 300$ destinataires un message de $d = 1000$ octets, un protocole basé sur des messages *unicast* aurait nécessité 300 et

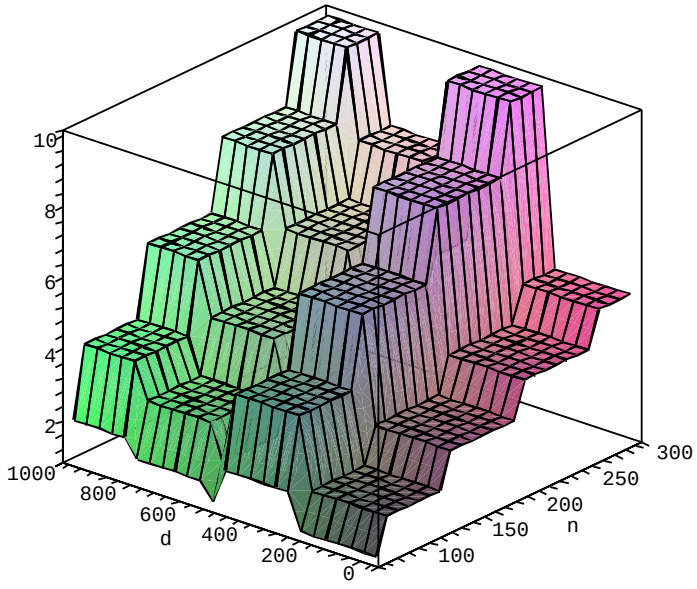


FIG. 2.6 – *Le taux de surcoût du protocole GXcast par rapport à un protocole de routage multicast traditionnel en terme de n et d pour $n > n_M$.*

600 paquets, soit respectivement 150 et 300 fois plus qu'un protocole de routage *multicast* traditionnel [15]. Il est important de rappeler que les protocoles de routage *multicast* traditionnel exigent la présence des états de routage *multicast* dans tous les routeurs appartenant aux arbres *multicast* des différents groupes et des paquets de contrôle spécifiques à chaque groupe circulent d'une manière permanente entre les routeurs des arbres pour maintenir ces états de routage.

Le protocole GXcast et l'*unicast* : Pour envoyer d octets de données à n membres d'un groupe, le nombre de paquets $p_{Unicast}(n, d)$ générés par la source en utilisant la transmission en mode *unicast* est défini par la formule suivante :

$$p_{Unicast}(n, d) = n * p(n, d) = n * \lceil \frac{d}{MTU - E} \rceil.$$

15. Rappelons que le protocole Xcast est incapable de gérer un groupe ayant plus de $n_{max} = 135$ membres.

La figure 2.7 présente le nombre de paquets générés par un protocole de routage *unicast* par rapport au nombre de paquets générés par le protocole GXcast. On remarque que la valeur du taux de surcoût est très élevé ce qui implique que le surcoût engendré par un protocole *unicast* par rapport à GXcast devient très important si le nombre de destinataires augmente.

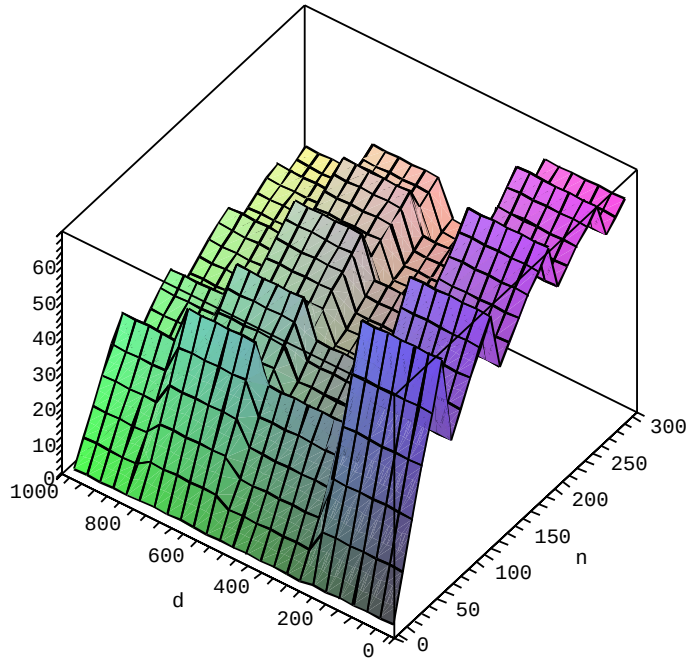


FIG. 2.7 – *Le taux de surcoût d'un protocole de routage unicast par rapport au protocole GXcast en terme de n et d.*

Le protocole GXcast et le protocole Xcast : Finalement nous étudions le taux de surcoût engendré par le protocole GXcast par rapport au protocole Xcast et nous considérerons deux cas seulement puisque le protocole Xcast est incapable de gérer un groupe ayant plus de n_{max} membres : $n \leq n_M = \frac{n_{max}}{2}$ et $\frac{n_{max}}{2} < n \leq n_{max}$.

Pour envoyer d octets de données à n membres d'un groupe, le nombre de paquets $p_{Xcast}(n, d)$ générés par le protocole Xcast est défini par la formule suivante dans les deux cas :

$$p_{Xcast}(n,d) = \lceil \frac{d}{MTU - E - IP * n} \rceil \approx \lceil \frac{d}{IP * (n_{max} - n)} \rceil.$$

Pour $n \leq \frac{n_{max}}{2}$, les deux protocoles GXcast et Xcast génèrent le même nombre de paquets, donc le taux de surcoût du protocole GXcast par rapport au protocole Xcast est égale à 1.

Pour $\frac{n_{max}}{2} < n \leq n_{max}$, nous avons :

$$p(n,d,\frac{n_{max}}{2}) = \lceil \frac{n}{\frac{n_{max}}{2}} \rceil \lceil \frac{d}{MTU - E - IP * \frac{n_{max}}{2}} \rceil \approx 2 * \lceil \frac{2 * d}{IP * n_{max}} \rceil.$$

Nous déduisons que : $\delta\ p_{GXcast/Xcast} = \frac{p(n,d,\frac{n_{max}}{2})}{p_{Xcast}(n,d)} \approx \frac{2 * \lceil \frac{2 * d}{IP * n_{max}} \rceil}{\lceil \frac{d}{IP * (n_{max} - n)} \rceil}$ est toujours ≤ 2.

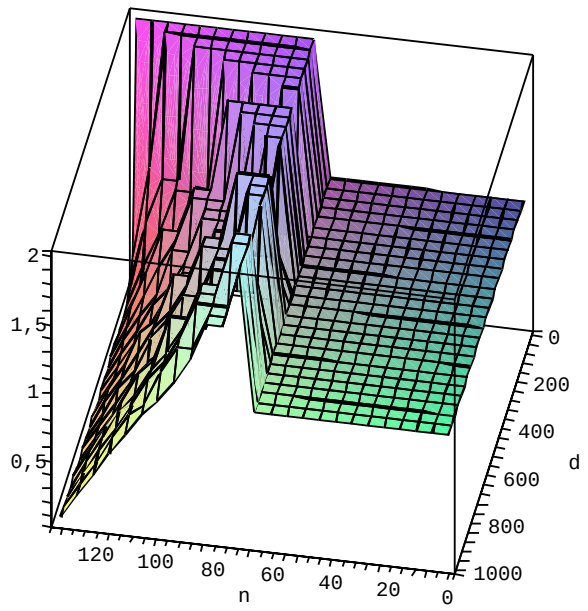


FIG. 2.8 – *Le taux de surcoût du protocole GXcast par rapport au protocole Xcast en terme de n et d.*

La figure 2.8 présente le nombre de paquets générés par le protocole GXcast par

rapport au nombre de paquets générés par le protocole Xcast. On remarque que la valeur du taux de surcoût ne dépasse jamais 2 et que dans beaucoup de cas cette valeur est largement inférieure[16] à 1 ce qui implique que $\delta\ p_{GXcast/Xcast}$ peut selon la taille de d prendre des valeurs $\leq\ 1$ et générer ainsi moins de paquets que le protocole Xcast.

2.3.2 Amélioration du protocole GXcast

L'effet du surcoût du protocole GXcast par rapport à un protocole de routage *multicast* traditionnel en terme de nombre de paquets circulant sur le réseau varie selon l'emplacement des destinataires et la topologie du réseau. Cet effet peut être atténué et limité aux liens aux alentours de la source. Considérons le réseau représenté sur la figure 2.9. Le groupe *multicast* G est composé de la source S et des six membres ayant joint le groupe dans l'ordre suivant : D_1, D_3, D_4, D_5, D_6 puis D_2. Observons le cas où n_M a été fixé à trois.

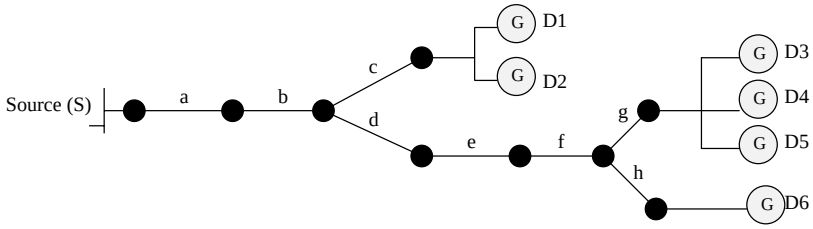


FIG. 2.9 – *Un exemple sur le tri des membres.*

Si la source S envoie un premier paquet GXcast à (D_1, D_3, D_4) et un second paquet GXcast à (D_5, D_6, D_2), en respectant l'ordre d'arrivée des membres dans le groupe, chacun des deux paquets empruntera les liens a, b, c, d, e et f. Un meilleur moyen d'atteindre les six membres en conservant la limitation $n_M = 3$ est d'envoyer, par exemple, le premier paquet GXcast à (D_1, D_2, D_6) et le second paquet GXcast à (D_3, D_4, D_5). Pour réaliser un regroupement optimal des membres, il faut cependant connaître la topologie entière du réseau.

16. Le nombre de paquets générés par le protocole GXcast peut être 16 fois plus petit que par le protocole Xcast.

Cependant, une certaine localité peut être déduite de la longueur du plus long préfixe commun de deux adresses IP. Traditionnellement, deux machines d'un même sous-réseau possèdent des adresses IP proches. Afin d'offrir un bon regroupement, nous proposons de trier la liste des membres à la source. La méthode naïve où les membres sont stockés dans une liste simple est moins performante que la solution que nous proposons. Pour s'en assurer, considérons les trois opérations essentielles à réaliser sur l'ensemble des membres :

- un nouveau membre adhère au groupe,
- un membre décide de quitter le groupe,
- la liste de membres doit être coupée en sous-listes de taille au plus égale à n_M.

L'algorithme naïf est très efficace lorsqu'il s'agit d'ajouter un nouveau membre au groupe ou lorsqu'il s'agit de découper la liste en sous-listes de taille n_M. Il se révèle assez mauvais lorsqu'il s'agit de supprimer un membre du groupe puisqu'il faut alors réaliser une recherche dans la liste des membres.

L'algorithme que nous proposons découpe la liste des membres de la même manière que l'algorithme naïf, à l'exception que la liste des membres est triée par adresse IP croissante. Nous proposons une structure d'arbre rouge-noir pour le stockage de la liste triée des membres [CLR90]. Un ajout dans un tel arbre est réalisé en temps logarithmique. Pour effectuer le retrait d'un membre de cet arbre, on procède en deux étapes : la première consiste à faire une recherche du membre, ce qui peut se faire en temps logarithmique et la seconde consiste à effacer le membre trouvé, ce qui peut se faire sans augmentation de la complexité. Enfin, la liste triée peut être obtenue par un parcours infixe de l'arbre en temps linéaire.

Le tableau 2.1 récapitule les complexités dans le pire des cas de l'algorithme naïf et de notre proposition. Nous pensons que l'utilisation des algorithmes proposés conduit à une diminution de la charge globale du réseau, avec un faible coût à la source.

Pour évaluer l'impact du tri, nous avons fait des simulations du protocole GXcast sur le réseau Abilene [Net]. La topologie Abilene représente l'épine dorsale expérimentale d'Internet2 aux Etats Unis. Elle se compose de 11 nœuds et de 14 liens (*cf.*

algorithme	arrivée d'un membre	départ d'un membre	découpage en sous-listes
naïf	$\mathcal{O}(1)$	$\mathcal{O}(n)$	$\mathcal{O}(n)$
arbre red-black	$\mathcal{O}(\log n)$	$\mathcal{O}(\log n)$	$\mathcal{O}(n)$

TAB. 2.1 – *La complexité de gestion de la liste des membres.*

figure 2.10).

FIG. 2.10 – *Le réseau Abilene.*

La figure 2.11 montre le coût moyen des arbres (à l'intérieur du réseau Abilene) pour $n = 210$ destinataires, divisés en paquets de $n_M = 70$ destinataires, obtenu après 1000 simulations. Le nombre de routeurs de bordure par nœud Abilene change de 1 à 20. Des sous-réseaux d'IP de classe C ont été aléatoirement attribués à chaque routeur de bordure, tout en assurant que deux adresses IP dans le sous-réseau du même routeur de bordure ont un préfixe en commun. Il n'y a aucune relation entre deux adresses IP dans deux sous-réseaux différents, même si ces deux routeurs de bordure sont reliés au même nœud Abilene. Lorsque les adresses IP sont aléatoirement choisies et uniformément distribuées, le coût moyen des arbres est égal à 30. En effet, pour chacun

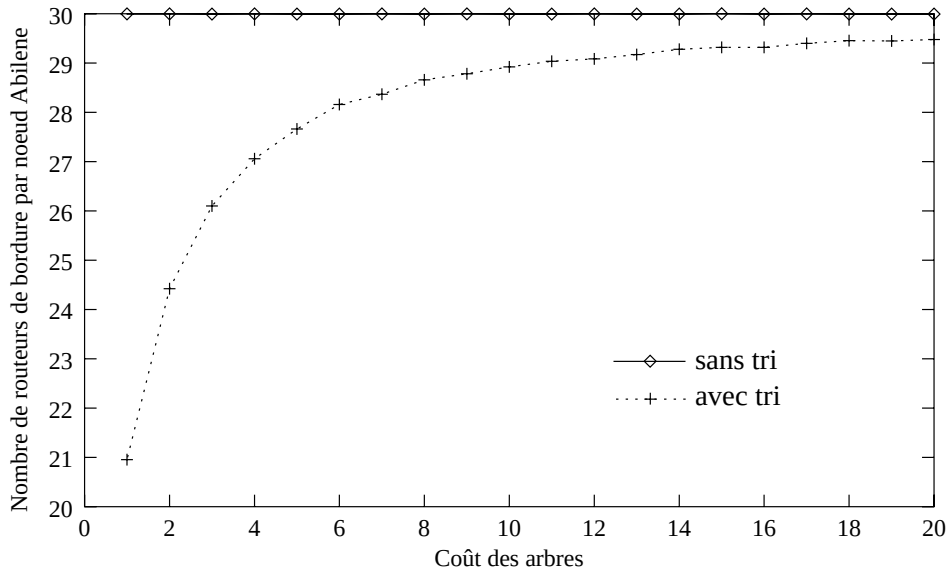


FIG. 2.11 – *L'influence du tri de la liste des destinataires sur le coût de l'arbre dans GXcast.*

des trois[17] arbres, chaque nœud Abilene est représenté au moins une fois dans la liste des destinataires et donc, 10 liens de la topologie d'Abilene sont présents pour chaque arbre. Quand il y a peu de routeurs de bordure par nœud Abilene, trier la liste des destinataires réduit considérablement le coût des arbres. En effet, une certaine localité peut être déduite : par exemple, un nœud Abilene qui contient un routeur de bordure ayant seulement les adresses IP les plus élevés n'est pas représenté dans la liste des destinataires pour le premier arbre.

La figure 2.12 montre le coût moyen des arbres sur la même topologie quand le nombre de paquets augmente et quand il y a 5 routeurs de bordure par nœud, obtenu après 1000 simulations. Le nombre de paquets varie de 1 à 20. À mesure que le nombre de paquets augmente, le coût des arbres sans tri augmente linéairement (chaque arbre couvre tous les nœuds). Plus le nombre de paquets générés est grand, plus le tri devient important. Pour 10 paquets générés, le gain dû au tri est proche de 50%. Par conséquent, trier la liste de destinations est très important dans le protocole GXcast.

17. Puisque $n/n_M = 3$, trois arbres sont construits. Sans tri, chaque arbre inclut tous les 11 nœuds de la topologie d'Abilene, couvrant 10 liens. Le coût total pour les trois arbres est donc 30.

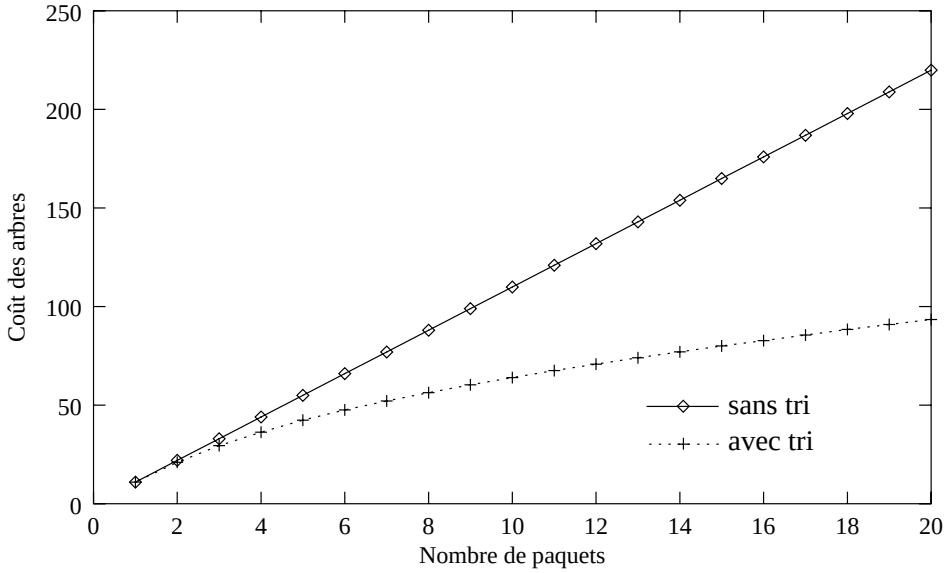


FIG. 2.12 – *L'influence du tri de la liste des destinataires sur le coût en terme de nombre de paquets générés dans GXcast.*

2.3.3 Analyse et simulation du coût du protocole GXcast

Le protocole GXcast peut être intéressant, utile et prometteur pour des applications comme les jeux en réseaux et les simulations interactives distribuées.

En effet, le trafic observé d'un jeu suit toujours le cycle de transmission suivant : le serveur envoie l'information d'état de jeu à chaque client (joueur) où des paquets sont lus et traités. Les clients synchronisent l'état de jeu de serveur avec leur état local de jeu, traitent les commandes des autres joueurs et retournent des paquets de mis à jour avec l'information de mouvement et de statut du joueur. Les paquets de mis à jour et d'information de serveur sont généralement de petite taille puisqu'ils contiennent seulement l'information de mouvement et de statut.

Ainsi, l'étude de [Far02] montre que typiquement la taille moyenne d'un paquet généré par un serveur est environ 127 octets avec un cœfficient de variation de 0.73. De plus, 99% des paquets est de taille inférieure à 250 octets et aucun paquet n'est plus grand que 1500 octets. Une grande partie de paquets de serveur atteignant une taille d'environ 1000 octets sont dûs à l'interruption du jeu (par exemple fin d'une partie

du jeu ou bien un changement de scénario où plus d'informations sont envoyés aux joueurs). Le trafic de client est caractérisé par une taille de paquet presque constante. La taille moyenne d'un paquet est de 82 octets avec un cœfficient de variation de 0.12. En plus, 99% de paquets ont une taille de 60 à 110 octets.

Une autre étude [FCFW02] sur les jeux en réseaux montrent que la taille moyenne d'un paquet généré par le serveur est environ 130 octets tandis que celui du client est de 40 octets. La taille moyenne des paquets reçus et émis par le serveur est de 80 octets. Cette étude a montré que sur une période d'une semaine, 16000 clients (moyenne de 95 joueurs par heure) ont établi une connection avec le serveur (ils ont participé au jeu) et 8000 ont été refusés (puisque le serveur a atteint sa limite de connections).

De même, dans le cadre de simulation interactive distribuée (DIS) [PMB99], des informations sur un environnement virtuel sont échangées entre différentes machines dans un système réparti. Cela permet de simuler le comportement des objets dans cet environnement. Les objets sont capables d'interactions physiques entre eux et peuvent détecter le autres objets par le visuel ou d'autres moyens (infrarouge, etc.). Le flux temps réel de DIS est formé de paquets de taille de 2000 bits (250 octets) et normalement transmis par le protocole de transport UDP.

2.3.3.1 Le simulateur NS

Pour mesurer le coût du protocole GXcast et le comparer avec celui du protocole Xcast nous avons implémenté les deux protocoles à l'aide du simulateur NS [FV01].

NS (*Network Simulator*) est un logiciel de simulation de réseaux informatiques : Il est principalement bâti avec les idées de la conception par objets, de réutilisabilité du code et de modularité. Il permet à l'utilisateur de définir un réseau et de simuler des communications entre les nœuds de ce réseau. NS utilise le langage OTcl (*Object Tools Command Language*) dérivé de TCL. A travers ce language, l'utilisateur décrit les conditions de la simulation : la topologie du réseau, les caractéristiques des liens physique, les protocoles utilisés, les communications qui ont lieu. La simulation doit d'abord être saisie sous forme de fichier que NS va utiliser pour produire un fichier

contenant les résultats. Ensuite, on utilise le programme NAM pour interpréter ces données et en donner un rendu graphique. Mais l'utilisation de l'OTcl permet aussi à l'utilisateur de créer ses propres procédures (par exemple s'il souhaite enregistrer dans un fichier l'évolution d'une variable caractéristique du réseau au cours du temps).

Il contient les fonctionnalités nécessaires à l'étude des algorithmes de routage *unicast* ou *multicast*, des protocoles de transport, de réservation, des services intégrés, des protocoles d'application comme HTTP. De plus le simulateur possède déjà une palette de systèmes de transmission, d'ordonnanceurs et de politiques de gestion de files d'attente pour effectuer des études de contrôle de congestion. La liste des principaux composants actuellement disponible dans NS par catégorie est:

- Application Web, ftp, telnet, générateur de trafic (CBR, ...)
- Transport TCP, UDP, RTP, SRM
- Routage statique ou dynamique (vecteur de distance)
- Routage *Multicast* (DVMRP, PIM)
- Gestion de file d'attente : RED, DropTail, Token bucket, etc
- Discipline de service : CBQ, SFQ, DRR, Fair Queueing
- Système de transmission : CSMA/CD, CSMA/CA, lien point à point

Prises ensembles, ces capacités ouvrent le champ à l'étude de nouveaux mécanismes au niveau des différentes couches de l'architecture du réseau. NS est devenu l'outil de référence pour les chercheurs du domaine. Ils peuvent ainsi partager leurs efforts et échanger leurs résultats de simulations. Cette façon de faire se concrétise aujourd'hui par l'envoi dans certaines listes de diffusion électronique de scripts de simulations NS pour illustrer les points de vue.

2.3.3.2 Le scénario de simulation

Nous avons utilisé NS pour développer notre simulateur des protocoles SGM. Ce simulateur peut servir aux chercheurs qui veulent tester les différents types de protocoles de routage *multicast* explicite. C'est le seul à notre connaissance qui existe dans le domaine.

Nous avons utilisé le réseau Abilene comme réseau d'essai et nous avons choisi 5 comme valeur moyenne du nombre de routeurs de bordure par nœud Abilene. A chaque routeurs de bordures sont connectées 5 destinataires possibles[18]. Notre réseau contient donc 341 nœuds au total, connectés par des liens bidirectionnels de 10 Gb/s avec un délai qui varie entre $2ms$ et $12ms$ selon la longueur du lien dans le réseau cœur et des liens bidirectionnels de 155 Mb/s avec un délai de $0.3ms$ à $0.5ms$ dans le reste de la topologie (entre les nœuds Abilene et les nœuds de bordure ainsi qu'entre les nœuds de bordure et les destinataires)(*cf.* figure 2.13).

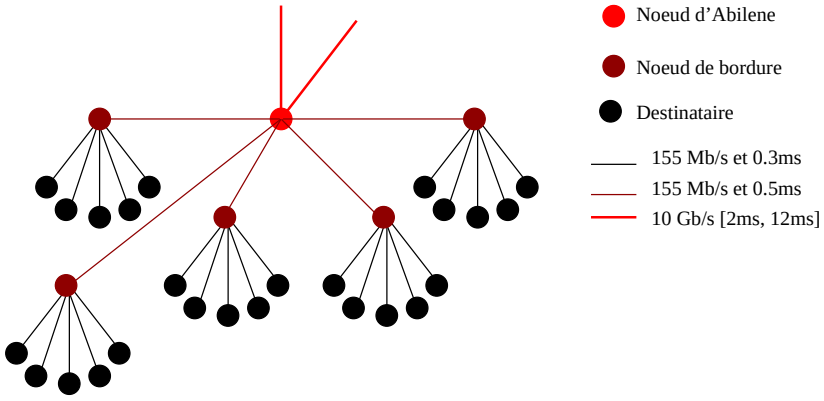


FIG. 2.13 – *Un nœud du réseau Abilene avec les nœuds de bordure et les destinataires.*

Nous considérons seulement un groupe *multicast* ayant une source appartenant à un sous-réseau et n destinataires (dans les autres sous-réseaux) qui joignent et quittent aléatoirement le groupe. Le nombre maximum de destinataires par paquet GXcast est de 70 (proche de $\frac{n_{max}}{2}$)[19]. Nous avons choisi nos paramètres de simulation en tenant compte des applications comme DIS et les jeux en réseaux. Le tableau 2.2 récapitule les paramètres utilisés dans la simulation.

La figure 2.14 représente le nombre estimé de paquets (selon la quantité de données (d) à transmettre) à générer par la source tandis que la figure 2.15 représente le nombre effectif de paquets générés par la source.

18. Nous avons essayé d'être conforme au mieux avec la topologie présentée dans Abilene.

19. Rappelons que le protocole Xcast est incapable de gérer un groupe ayant plus de $n_{max} = 135$ membres et que les deux protocoles GXcast et Xcast ont le même comportement si $n \leq \frac{n_{max}}{2}$.

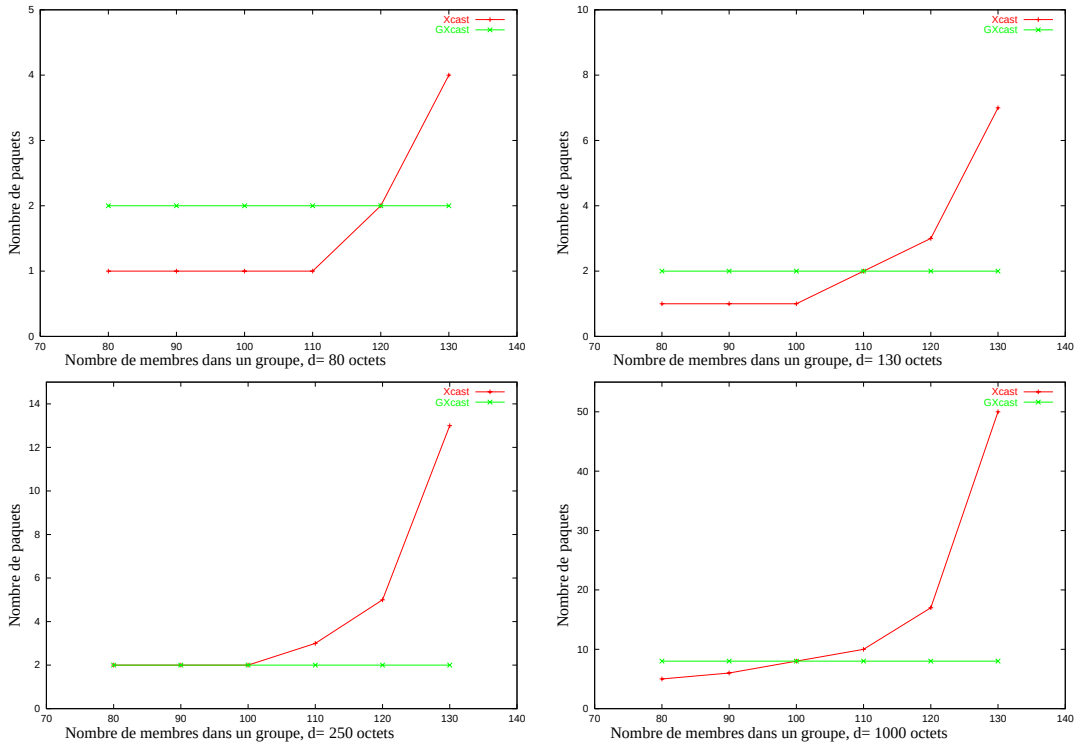


FIG. 2.14 – *Le nombre estimé de paquets à générer par la source avec les protocoles GXcast et Xcast.*

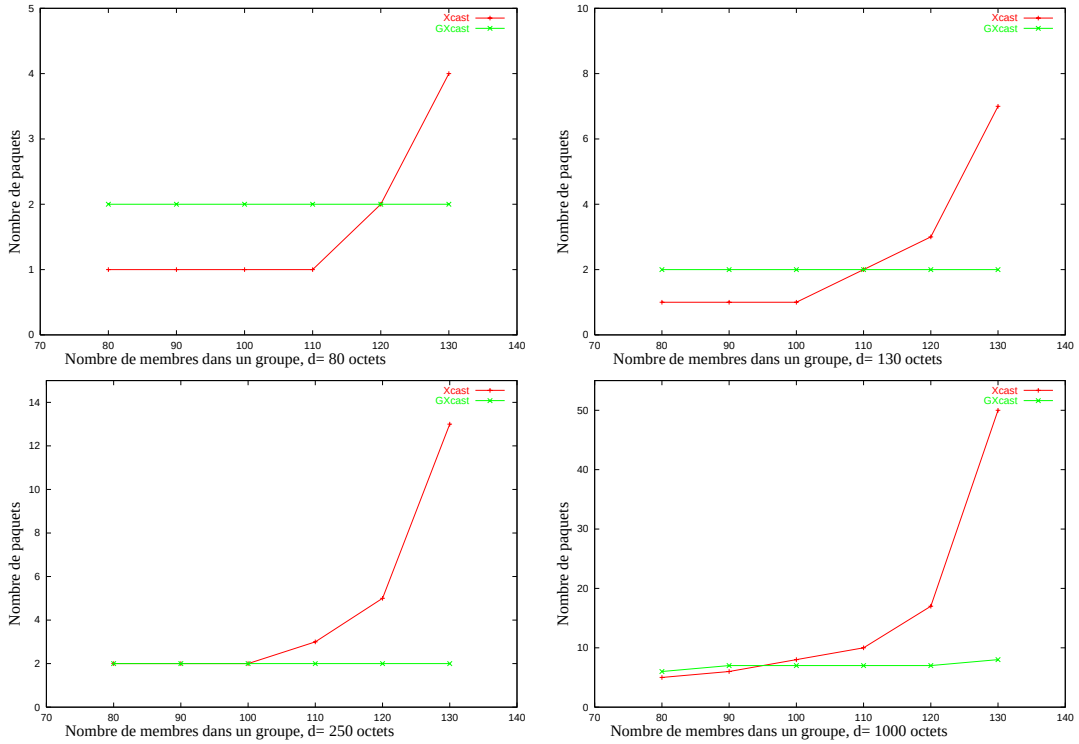


FIG. 2.15 – *Le nombre de paquets générés par la source avec les protocoles GXcast et Xcast.*

d	80, 130, 250, 1000	Taille de données à transmettre à chaque destinataire
n	80, 90, 100, 110, 120, 130	Nombre de destinataires par groupe
n_M	70	Nombre de destinataires maximum dans un paquet GXcast

TAB. 2.2 – *Les paramètres de simulation du protocole GXcast.*

L'axe horizontal représente le nombre de paquets générés par la source et l'axe vertical représente le nombre de destinataires (n qui varie de 80 à 130) appartenant à un groupe.

On remarque que le nombre effectif est égal au nombre estimé sauf dans le cas où $d = 1000$ octets. Ceci est dû du fait que la formule 2.1 suppose que les paquets générés par la source pour n destinataires dans GXcast contiennent forcément n_M destinataires. Prenons le cas où $n_M < n < 2 * n_M$: la source génère un paquet de n_M destinataires et un autre de $n - n_M$ destinataires et donc ce dernier peut contenir plus de données que le premier.

Pour mesurer le coût de paquets générés, nous définissons γ, l'estimateur de la charge moyenne du réseau par lien et σ, le taux de la charge moyenne du réseau par lien par les équations suivantes :

$$\gamma = \frac{\sum_{i=1}^{N_l} N_{paq}(l_i)}{t_{sim} * N_l}$$

et

$$\sigma = \frac{\sum_{i=1}^{N_l} (\frac{\sum_{j=1}^{N_{paq}(l_i)} (L_{pj})}{D_{l_i}})}{t_{sim} * N_l}$$

où N_l est le nombre de liens, $N_{paq}(l_i)$ est le nombre de paquets sur le lien l_i, L_{pj} est la taille d'un paquet pj, D_{l_i} est le débit du lien l_i et t_{sim} est le temps de simulation.

Nous faisons cette distinction puisque même si le nombre de paquets générés par la source avec le protocole GXcast est plus élevé que celui de paquets générés avec le

protocole Xcast, la taille d'un paquet GXcast est plus petite ou égale que celle d'un paquet Xcast. Nous considérons que la source génère un paquet Xcast chaque $2.5\ ms$ (cas d'un jeu en réseau) et nous remarquons que le nombre de paquets générés est linéaire en fonction du temps de simulation. Puisque les débits sont identiques sur tous les liens du réseau de la source et celui des destinataires et le sont aussi pour le réseau cœur, nous considérons seulement le nombre de paquets circulant et la quantité de données transmises dans chaque réseau pendant ce temps de simulation. Ainsi, les figures 2.16 et 2.17 représentent respectivement le nombre de paquets et la quantité de données transmises sur tous les liens du réseau de la source. Les figures 2.18 et 2.19 représentent respectivement le nombre de paquets et la quantité de données transmises sur tous les liens du réseau cœur et les figures 2.20 et 2.21 représentent respectivement le nombre de paquets et la quantité de données transmises sur tous les liens des réseaux des destinataires.

GXcast et Xcast dans le réseau de la source : Nous remarquons d'abord que le protocole Xcast est incapable de gérer des groupes ayant plus que 100 membres et surtout pour une charge utile de plus que 250 octets. Pour ce type de groupes, nous remarquons que non seulement le protocole GXcast génère moins de paquets que le protocole Xcast mais aussi que le volume transmis à travers le réseau source est beaucoup plus petit que celui de Xcast. Pour une charge utile de 80 à 130 octets, et bien que le protocole GXcast génère deux fois plus de paquets dans le réseau source, le surcoût engendré par le protocole GXcast en terme de volume transmis est faible. En effet, ce rapport diminue du fait que la taille de données transmises avec le protocole GXcast est inférieure à deux fois la taille de données transmises avec le protocole Xcast.

GXcast et Xcast dans le réseau cœur : Nous remarquons que malgré le fait que le nombre de paquets transmis par le réseau cœur avec le protocole GXcast est plus élevé que le nombre de paquets générés avec le protocole Xcast, le volume transmis à travers ce réseau cœur est plus petit que le volume généré avec le protocole Xcast et donc le protocole GXcast se comporte mieux que le protocole Xcast dans le réseau cœur. Ceci

constitue un avantage pour le protocole GXcast puisqu'avec le protocole GXcast on arrive à réduire le coût de l'arbre à l'intérieur

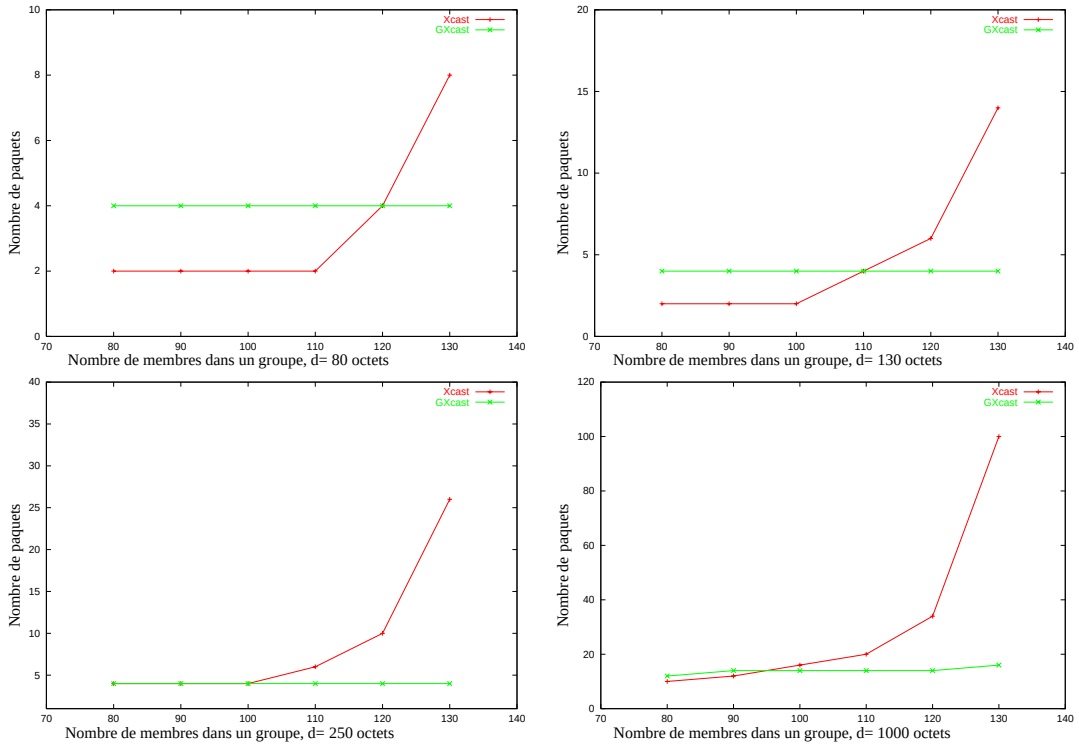


FIG. 2.16 – *Le nombre de paquets dans le réseau de la source avec les protocoles GXcast et Xcast.*

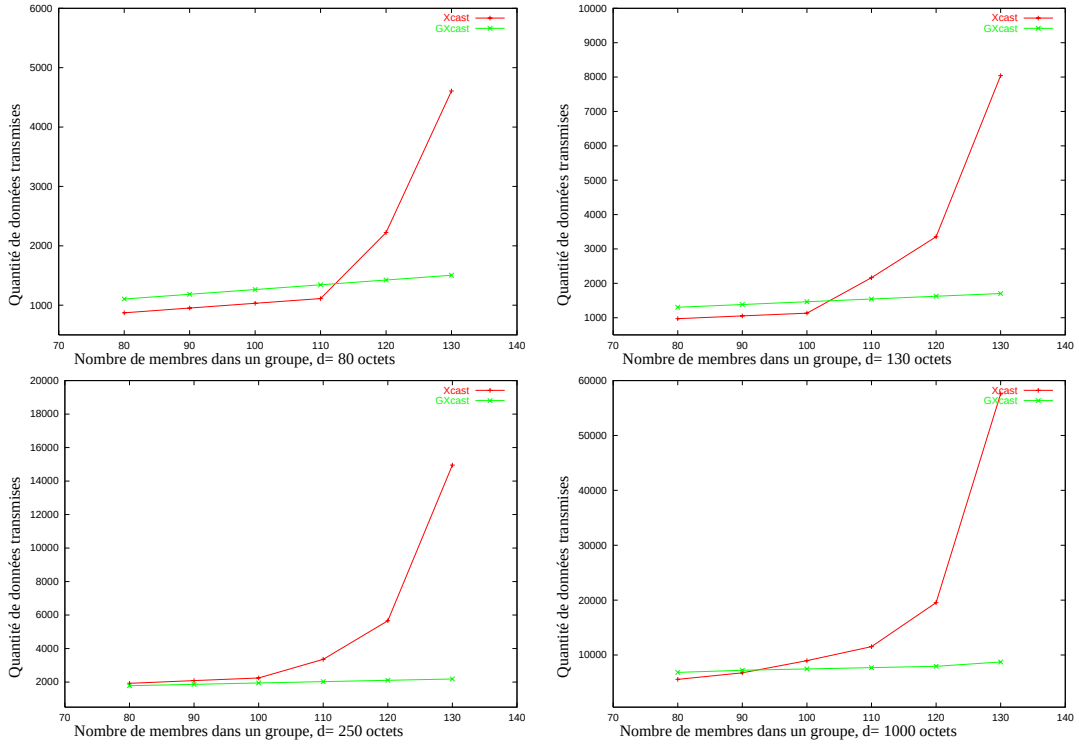


FIG. 2.17 – *Le volume transmis dans le réseau de la source avec les protocoles GXcast et Xcast.*

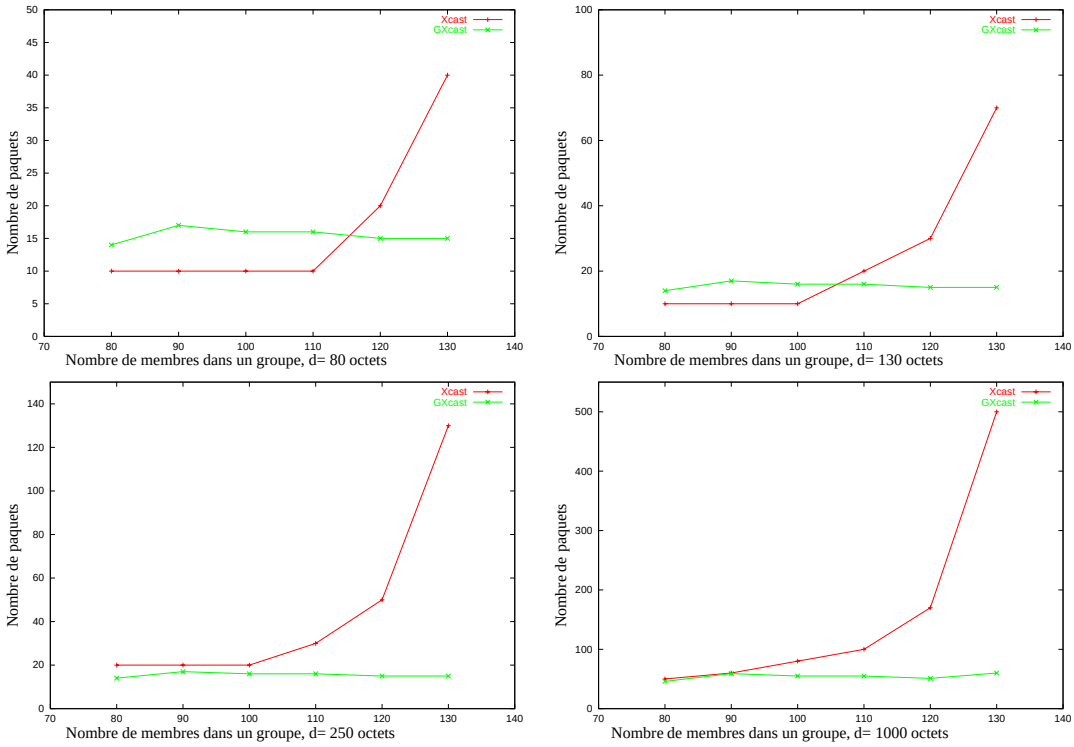


FIG. 2.18 – *Le nombre de paquets dans le réseau cœur avec les protocoles GXcast et Xcast.*

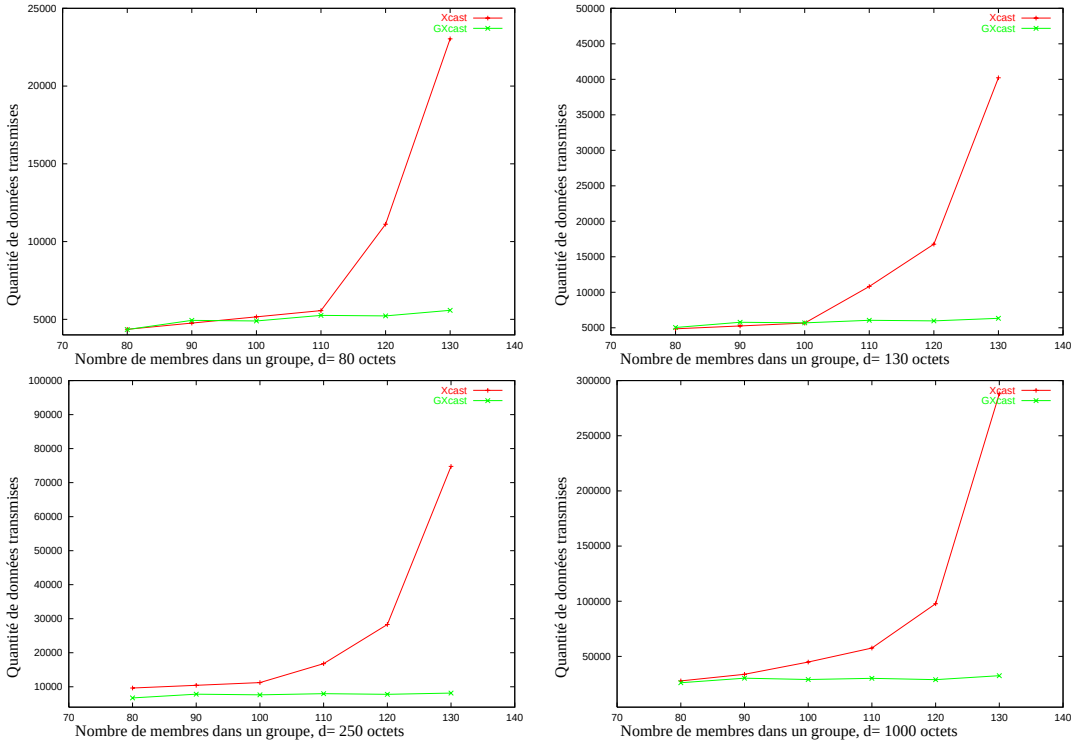


FIG. 2.19 – *Le volume transmis dans le réseau cœur avec les protocoles GXcast et Xcast.*

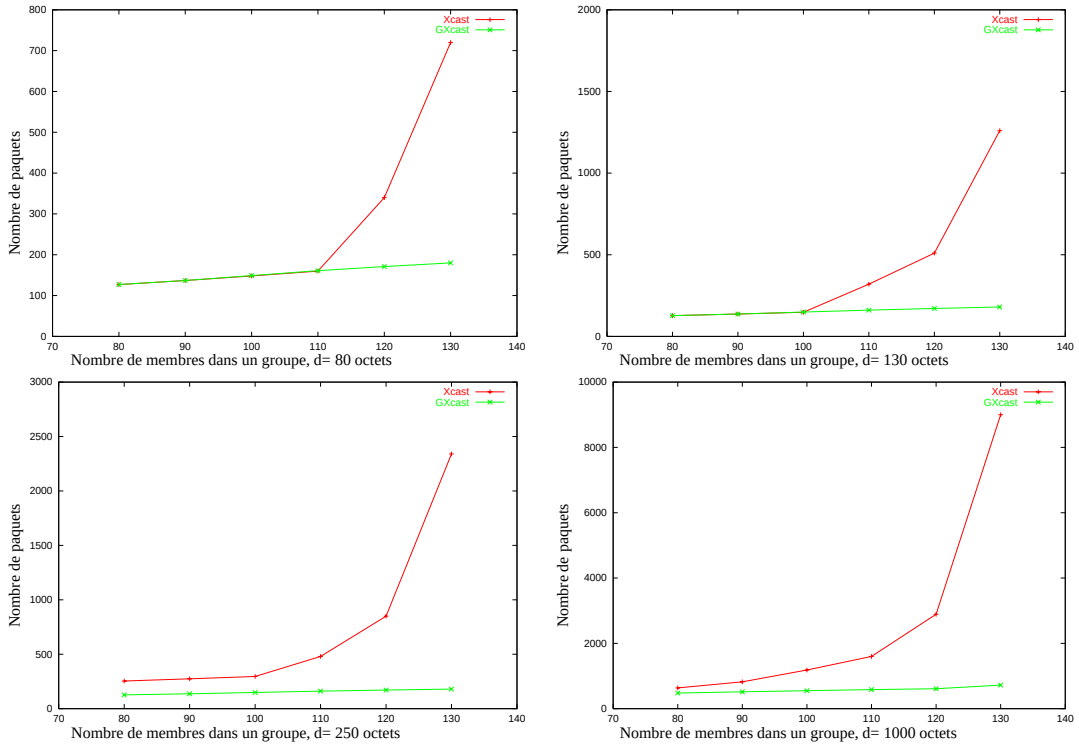


FIG. 2.20 – *Le nombre de paquets dans les réseaux des destinataires avec les protocoles GXcast et Xcast.*

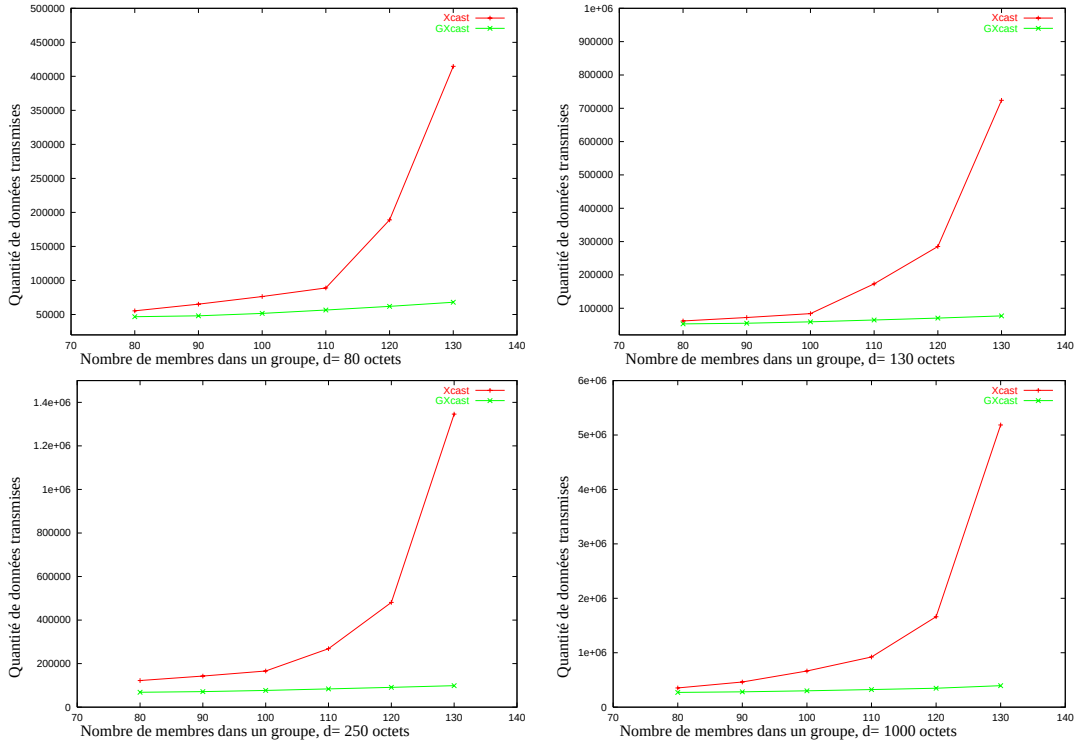


FIG. 2.21 – *Le volume transmis dans les réseaux des destinataires avec les protocoles GXcast et Xcast.*

du réseau cœur. Pour les groupes de moins que 100 membres et pour une charge utile de moins que 130 octets, le protocole GXcast a un comportement similaire au protocole Xcast et parfois génère un léger surcoût.

GXcast et Xcast dans les réseaux des destinataires : Finalement, nous remarquons une nette réduction du nombre de paquets et du volume transmis dans les réseaux des destinataires avec le protocole GXcast par rapport au protocole Xcast. Ceci est dû du fait que chaque destinataire recevra un paquet Xcast de taille fixe tandis que le paquet GXcast destiné au même destinataire est de taille moins élevé ce qui réduit le volume transmis à travers les réseaux des destinataires.

Après que nous avons remarqué que le surcoût provient de la taille des paquets et que le nombre de paquets ne peut pas être considéré comme le seul facteur de surcoût, nous avons fait une comparaison avec un protocole de routage *unicast*. En effet, avec un protocole de routage *unicast*, la source envoie un paquet séparé à chaque destinataire mais la taille de ce paquet est plus petite que la taille d'un paquet GXcast dès que le nombre de membres du groupe dépasse quelques unités. On rappelle que le nombre de paquets GXcast générés est plus petit que celui en *unicast*. Puisque le protocole GXcast et la transmission en mode *unicast* peuvent supporter plus que 130 membres limités par le protocole Xcast, nous prenons comme paramètre les 3 valeurs suivantes : 90, 140 et 190 destinataires dans un groupe. En effet, ces valeurs ont été choisi en supposant que le nombre de destinataires dans un jeu en réseau va doubler : de 95 par heure à 190 par heure. Nous prenons les mêmes paramètres de simulation. Ainsi, les figures 2.22 et 2.23 représentent respectivement le nombre de paquets et la quantité de données transmises sur tous les liens du réseau de la source. Les figures 2.24 et 2.25 représentent respectivement le nombre de paquets et la quantité de données transmises sur tous les liens du réseau cœur et les figures 2.26 et 2.27 représentent respectivement le nombre de paquets et la quantité de données transmises sur tous les liens des réseaux des destinataires. L'axe horizontal représente le nombre de paquets ou la

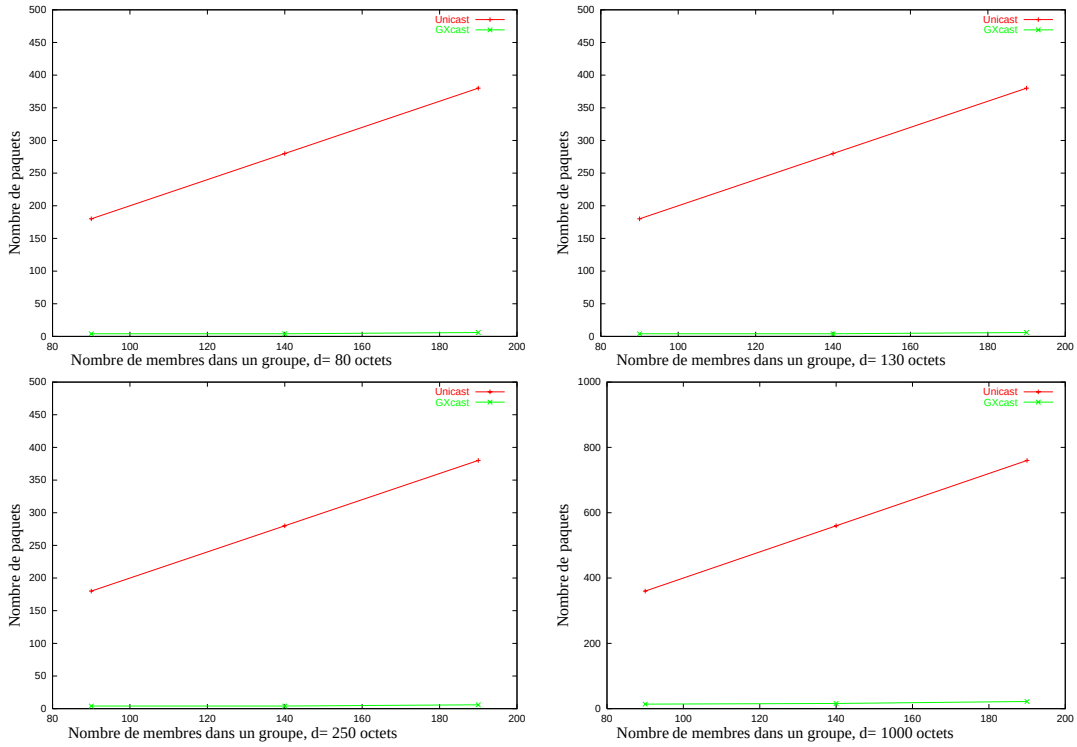


FIG. 2.22 – *Le nombre de paquets dans le réseau de la source avec les protocoles GXcast et Unicast.*

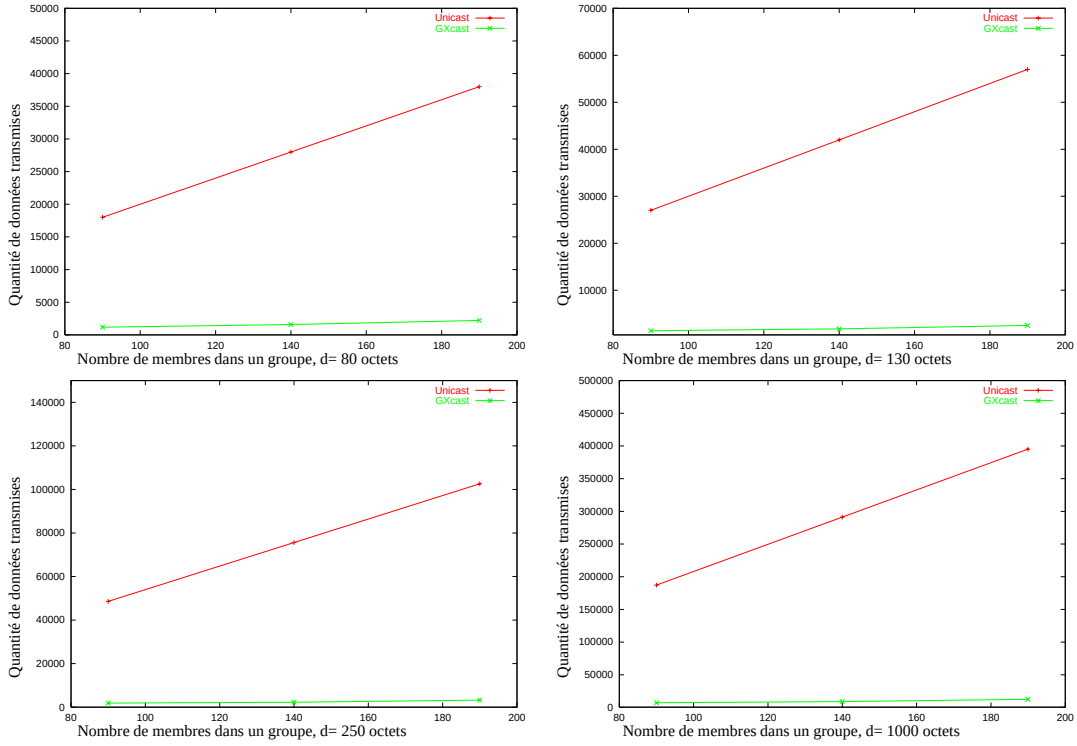


FIG. 2.23 – *Le volume transmis dans le réseau de la source avec les protocoles GXcast et Unicast.*

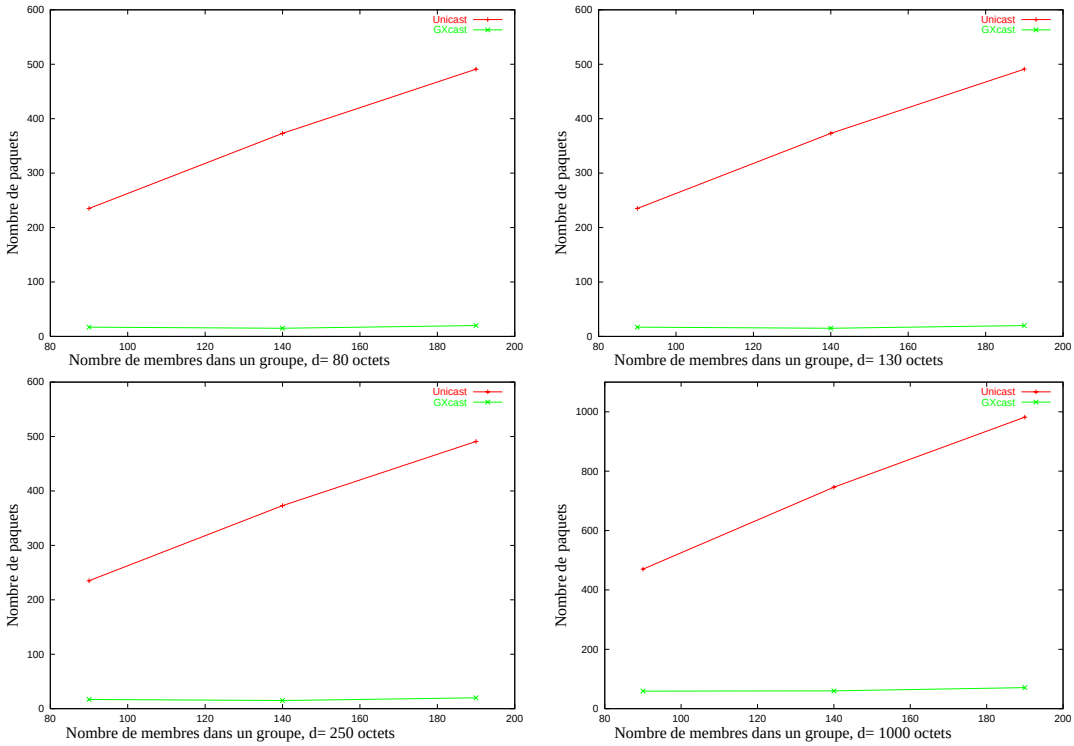


FIG. 2.24 – *Le nombre de paquets dans le réseau cœur avec les protocoles GXcast et Unicast.*

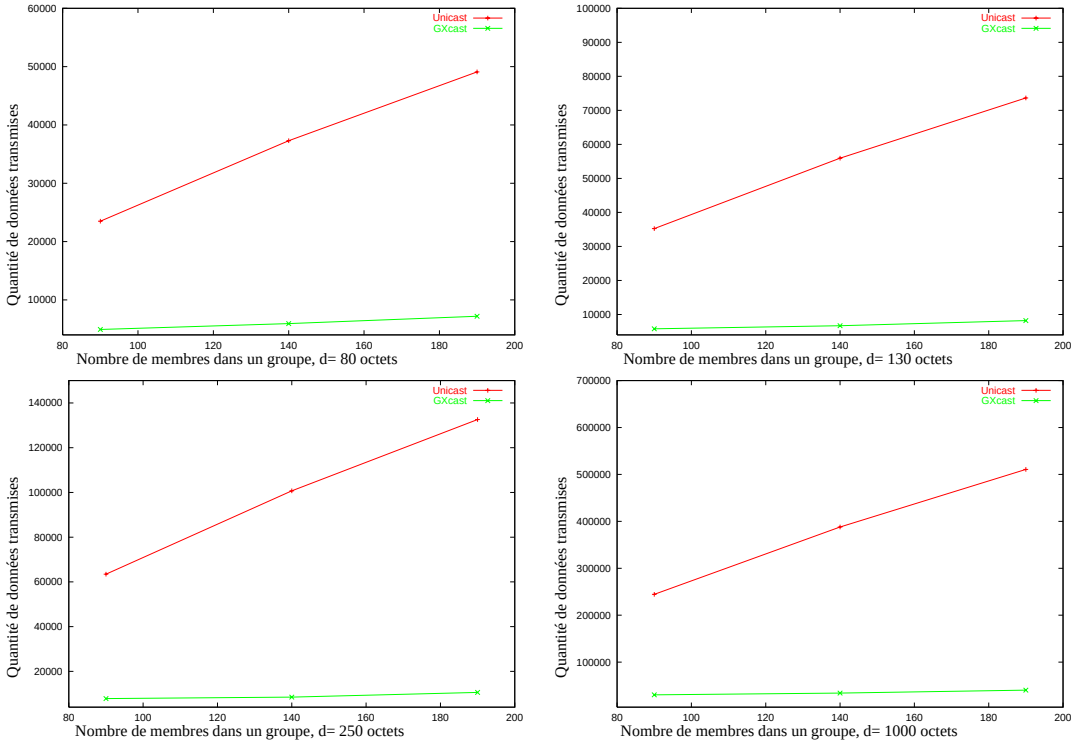


FIG. 2.25 – *Le volume transmis dans le réseau cœur avec les protocoles GXcast et Unicast.*

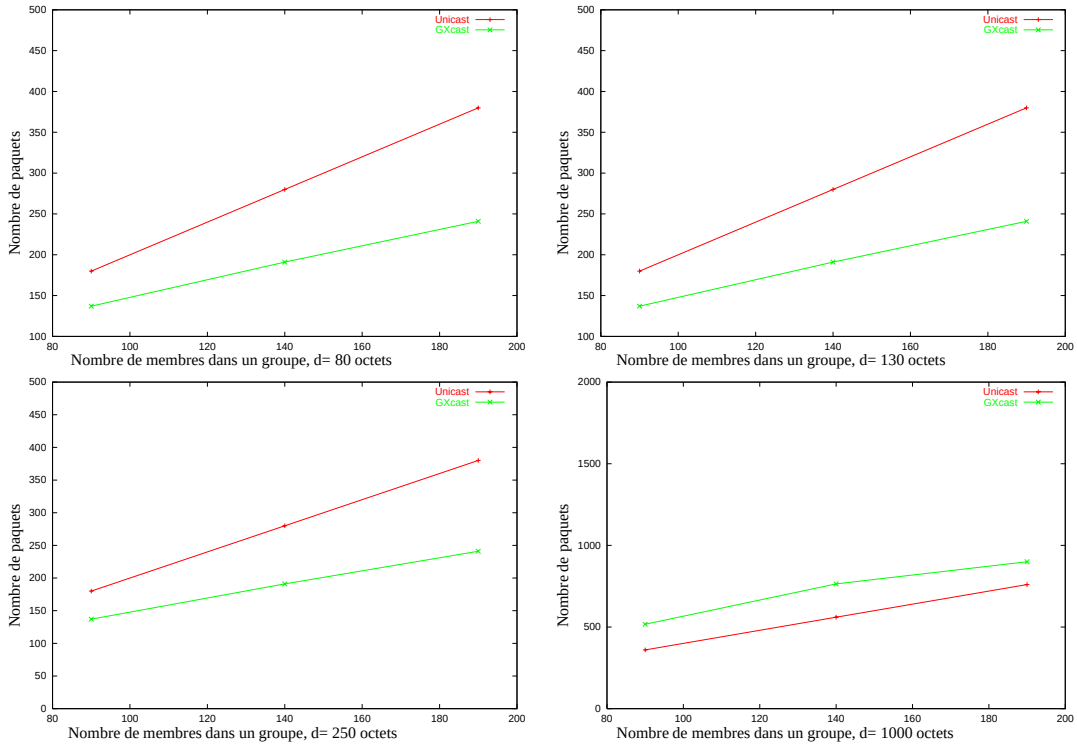


FIG. 2.26 – *Le nombre de paquets dans les réseaux des destinataires avec les protocoles GXcast et Unicast.*

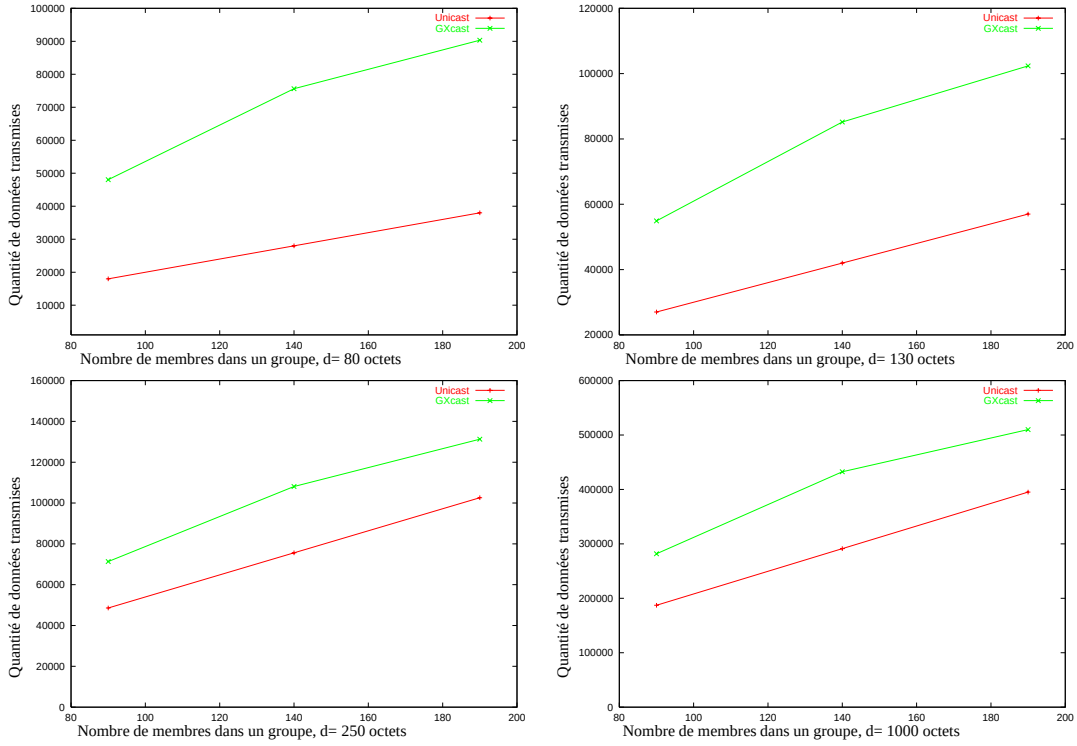


FIG. 2.27 – *Le volume transmis dans les réseaux des destinataires avec les protocoles GXcast et Unicast.*

quantité de données transmises et l'axe vertical représente le nombre de destinataires (n prend les 3 valeurs suivantes : 90, 140 et 190) appartenant à un groupe. Les axes portants le label "Unicast" représentent les valeurs obtenues en utilisant la transmission en mode *unicast*.

GXcast et Unicast dans le réseau de la source : Nous remarquons tout d'abord que le nombre de paquets générés est beaucoup plus élevé avec le protocole Unicast qu'avec le protocole GXcast. Ce surcoût se confirme avec le volume transmis à travers le réseau de la source. Avec les paramètres choisis nous remarquons qu'utiliser GXcast réduit de 20 fois le coût des liens du réseau source. Ceci constitue un avantage majeur en tenant compte du fait que les liens au voisinage des sources ne sont pas toujours capables de supporter une charge importante d'où l'intérêt d'utiliser la transmission en mode *multicast* au lieu d'utiliser la transmission en mode *unicast*.

GXcast et Unicast dans le réseau cœur : L'avantage de l'utilisation du protocole GXcast au lieu d'un protocole utilisant la transmission en mode *unicast* se manifeste encore une fois dans le réseau cœur. En effet, le nombre de paquets qui traverse le réseau cœur est de 15 fois plus important avec le protocole Unicast qu'avec le protocole GXcast. Le volume transmis est de l'ordre de 7 fois moins élevé et peut arriver à moins 12 fois avec le protocole GXcast qu'avec le protocole Unicast selon la charge utile de données.

GXcast et Unicast dans les réseaux des destinataires : Finalement, dans les réseaux des destinataires, le nombre de paquets transmis avec le protocole GXcast est moins élevé qu'avec le protocole Unicast. Ceci est un résultat attendu puisque GXcast utilise un seul paquet pour n_M destinataires tandis que Unicast utilise un paquet pour chaque destinataire. Mais ce nombre élevé de paquets ne se traduit pas en volume transmis, puisque la taille d'un paquet *unicast* est plus petite que la taille d'un paquet GXcast. Nous déduisons que le protocole Unicast présente plus d'avantages sur le protocole GXcast dans les réseaux des destinataires.

2.3.4 L'étude du délai du protocole GXcast

Le délai est le temps écoulé entre l'envoi d'un paquet par la source et sa réception par le destinataire. Pour illustrer l'impact du coût de traitement d'un paquet GXcast, nous discutons brièvement les divers délais ajoutés à un paquet lors de son voyage d'un nœud à un autre dans un réseau. Le délai tient compte du délai de traitement d'un paquet dans un nœud, du délai de propagation le long du chemin (le temps nécessaire pour envoyer un paquet par l'intermédiaire d'un lien), du délai de transmission (le temps nécessaire pour envoyer (injecter) un paquet sur un lien) et du délai de file d'attente (le temps que le paquet passe dans la file d'attente avant qu'il puisse être envoyé) induit par la mise en file d'attente des paquets dans les systèmes intermédiaires (*cf.* figure 2.28).

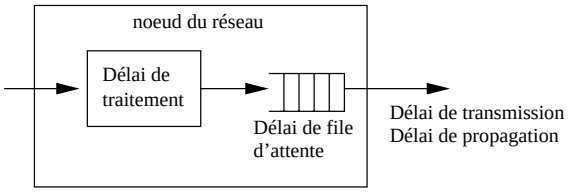


FIG. 2.28 – *Les différents délais dans un nœud du réseau.*

Le tableau 2.3 montre un calcul simple de ces délais pour des liens de 200 km et de 1 Gb/s, et des paquets de taille 1250 octets. Dans la plupart des cas, les principaux délais sont les délais de propagation et de file d'attente et sont donc considérés dans les simulations et les mesures. Le délai de transmission est généralement négligemment petit pour des liens rapides et de petits paquets et n'est pas donc considéré. Traditionnellement, (la colonne "Routage simple" du tableau 2.3) le délai de traitement est également négligeable. Cependant, le traitement de paquet peut prendre des valeurs considérables dans les cas où la modification de la charge utile est nécessaire (Comme dans IPsec, où on a besoin d'environ 100 instructions d'usage de processeur par octets de charge utile) [RNW03]. En conséquence, le délai de traitement peut contribuer pas moins que 50% du délai total de traitement d'un paquet (la colonne "Modification complexe de la charge utile" du tableau 2.3).

Délai	Routage simple	Modification complexe de la charge utile
Délai de traitement	10 μs	1000 μs
Délai de propagation	1000 μs	1000 μs
Délai de transmission	10 μs	10 μs
Délai de file d'attente	0..∞	0..∞

TAB. 2.3 – *Les différents délais pour un lien de* 1*Gb/s et de* 200 *km, un paquet de taille* 10*kb et un processeur de* 100*MIPS.*

2.3.4.1 Le temps de traitement global

Nous définissons le temps de traitement global d'un protocole comme étant la somme des temps de traitement des paquets qu'il envoie. Pour un paquet GXcast comportant n_M destinataires, le temps de traitement de ce paquet est approché par $t_{n_M} = \tau_1 + \tau_2 * n_M$, où τ_1 représente le temps de traitement des en-têtes IP et GXcast et τ_2 représente le temps de traitement associé à une entrée dans la liste des destinataires (recherche dans la table de routage, création des paquets par interface de sortie, etc.). On a donc le temps de traitement global :

$$t_G(n_M) = \lceil \frac{n}{n_M} \rceil t_{n_M} \approx \lceil \frac{n}{n_M} \rceil * (n_M + 1) * \tau_1 \approx n * (1 + \frac{1}{n_M}) * \tau_1.$$

La fonction $t_G(n_M)$ est une fonction strictement décroissante. Son minimum est donc en $t_G(n_{max})$. Cependant, choisir $n_M = n_{max}$ est irréaliste comme nous l'avons démontré dans les paragraphes 2.2.4.1 et 2.2.4.3 puisque cette valeur ne tient pas compte de la quantité de données utile effectivement transportée. Cependant, le choix d'une valeur petite pour n_M augmenterait rapidement le coût suite au nombre élevé de paquets générés ainsi que le temps de traitement global. La valeur par défaut de $n_M = \frac{n_{max}}{2}$, nous conduit à un temps de traitement global qui semble proche de l'optimal et c'est un bon compromis. Cependant, dans notre étude sur le délai, t_{n_M} représente le temps de traitement (dans le processeur du nœud) d'un paquet GXcast tandis que la valeur de $t_G(n_M)$ est incluse d'une façon indirecte dans la mesure du temps d'attente dans la file d'attente du nœud. Il est évidement clair que $t_{n_M} = t_n$ si $n \leq n_M$ et que

$t_{n_M} < t_n$ si $n > n_M$.

2.3.4.2 Le délai supplémentaire par rapport aux protocoles Xcast et Unicast

Le délai total d'acheminement d'un paquet est le temps qu'il lui faut pour atteindre sa destination à partir de la source. Il dépend en partie du nombre de paquets dans la file d'attente des routeurs et du temps de traitement de chacun d'eux. La différence de délai perçue par l'utilisateur final (c'est-à-dire le destinataire) joue un rôle important dans le choix des protocoles.

Prenons T comme l'arbre *multicast* (dans notre cas c'est l'ensemble de chemins suivis par les paquets GXcast) d'une source S vers tous les destinataires. La liste des destinataires est appelée L. Soit un réseau représenté par un graphe orienté $G = (V, A, \delta)$ où V représente l'ensemble de nœuds, A est l'ensemble des liens et δ la fonction délai[20] $\delta : A \rightarrow I\!R^+$.

Prenons $P_T(S, v)$ le chemin unique de la source S vers la destination $v \in L$[21], sur l'arbre T, de sorte que :

$$\delta_{T,v} = \sum_{l \in P_T(S,v)} \delta(l) \ \text{ pour } \ v \in L,$$

où $\delta_{T,v}$ est le délai pour une destination v sur l'arbre T.

Nous définissons le délai moyen global pour un protocole P par :

$$\delta_g(P) = \frac{\sum_{i=1}^{n} \delta_{T,v_i}}{n}, v_i \in L.$$

Nous définissons le délai supplémentaire moyen $\delta_+(P)$ d'un protocole P en comparaison avec le protocole GXcast par l'équation suivante :

$$\delta_+(P) = \delta_g(P) - \delta_g(\text{GXcast}).$$

20. δ inclut le temps de transmission, de propagation, de traitement et d'attente.
21. $|L| = n$.

Le délai supplémentaire moyen mesure le surcoût de délai en comparaison avec le protocole GXcast.

Afin de mesurer les délais supplémentaires moyen, minimum et maximum introduits au niveau du récepteur par la variation de la charge utile, le protocole GXcast a été implémenté sous la plate-forme de simulation NS avec le même réseau décrit précédemment. Ainsi, les figures 2.29 et 2.30 représentent respectivement le délai supplémentaire (moyen, minimum et maximum) du protocole Xcast par rapport au protocole GXcast et celui du protocole Unicast par rapport au protocole GXcast.

Le délai supplémentaire du protocole Xcast par rapport au protocole GXcast : Nous remarquons tout d'abord que le délai supplémentaire moyen, minimum et maximum sont positifs dans la majorité des cas. Ce qui établit l'avantage du protocole GXcast sur le protocole Xcast. Nous remarquons encore que la courbe de délai croît rapidement avec la croissance du nombre de membres dans un groupe. Nous remarquons que dans les cas où GXcast est moins performant que Xcast, le délai supplémentaire de GXcast par rapport à Xcast ne dépasse pas $10\mu s$ tandis que dans la plupart des autres cas Xcast dépasse largement ces valeurs.

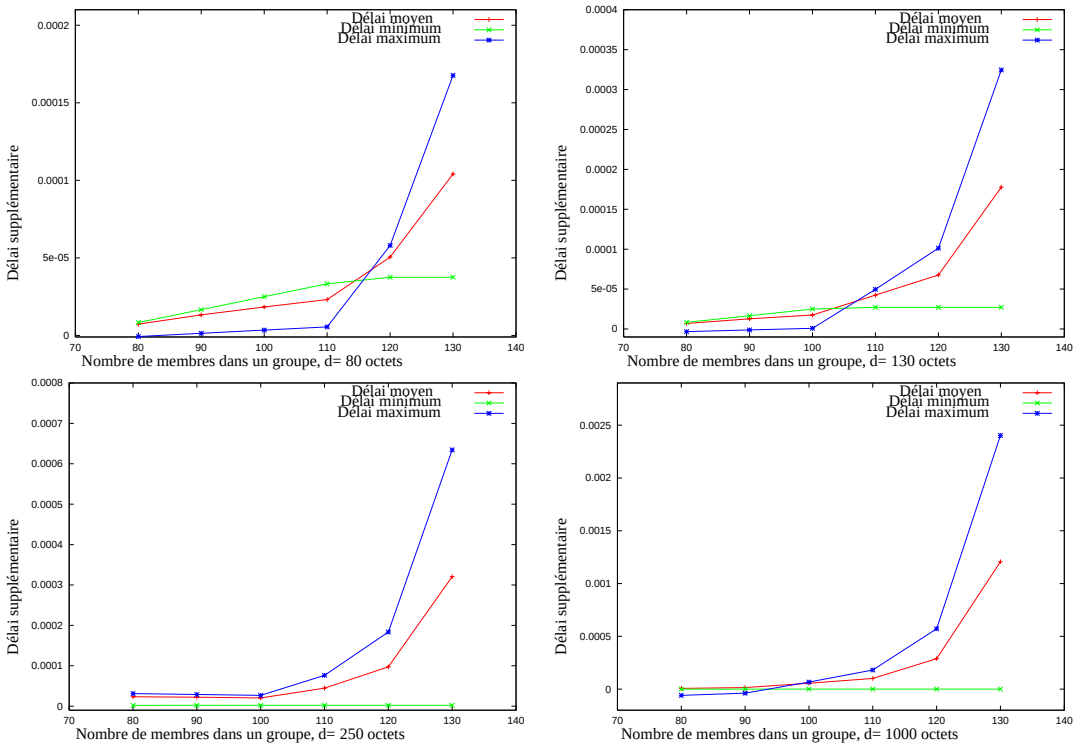


FIG. 2.29 – *Le délai supplémentaire du protocole Xcast par rapport au protocole GXcast.*

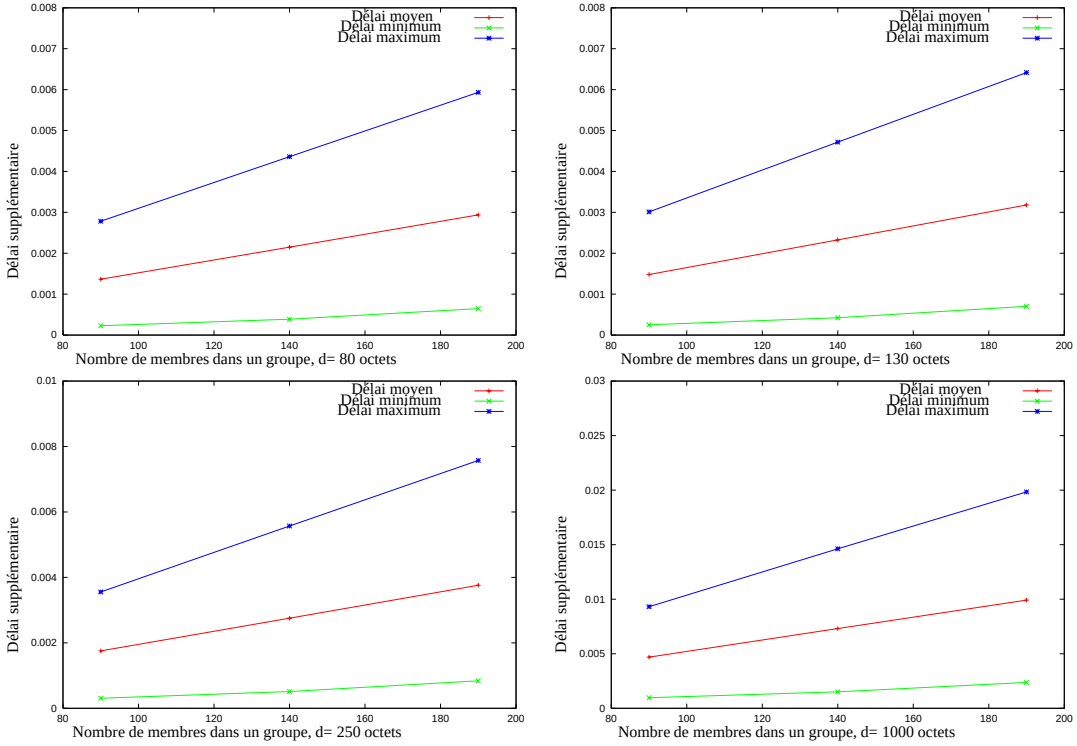


FIG. 2.30 – *Le délai supplémentaire du protocole Unicast par rapport au protocole GXcast.*

Le délai supplémentaire du protocole Unicast par rapport au protocole GXcast : Nous remarquons que le protocole GXcast est plus rapide que le protocole Unicast partout.

2.3.5 Utilisation du PMTU au lieu du MTU

Dans tout ce chapitre, nous avons utilisé pour valeur de MTU le MTU minimal garanti par le protocole IP. Cependant, le MTU d'un chemin (PMTU) peut aussi être utilisé. L'utilisation d'une valeur plus grande pour le MTU par défaut améliore notablement le comportement de GXcast et diminue la charge du réseau. Le MTU de l'arbre *multicast* que les paquets GXcast vont emprunter est la plus petite valeur des MTU des différents liens, de la source aux destinataires. À titre informatif, le protocole IPv6 prévoit de stocker le PMTU pour les différentes destinations *unicast* [MDM], ce qui facilite le calcul du MTU de l'arbre pour GXcast. On notera cependant qu'il faut envisager de mettre à jour la valeur de n_M pour des changements notables de la valeur de MTU.

2.4 Conclusion

Les protocoles Xcast et Xcast+ permettent de gérer efficacement un grand nombre de petits groupes *multicast*. Une partie de leurs inconvénients vient de leur incapacité à gérer la fragmentation des paquets. De plus, une limite forte existe sur le nombre de membres du groupe *multicast*. Dans ce chapitre, nous avons proposé une extension à ces protocoles, nommée GXcast. GXcast permet de résoudre le problème de la fragmentation et optimise des critères comme le nombre de paquets envoyés ou le temps de traitement des paquets au sein des routeurs. À l'issue de l'étude de ce protocole, nous avons montré que GXcast gère facilement avec une réduction du coût et de délai, un grand nombre de groupes de taille moyenne, même lorsque les membres sont disséminés sur quelques centaines de sous-réseaux.

Chapitre 3

Le protocole SEM

L'acheminement efficace de paquets vers de multiples destinations est un compromis entre consommation de bande passante, temps de traitement d'un paquet, volume occupé par les états de routage, et messages de signalisation nécessaire pour maintenir l'arbre *multicast*. Nous avons vu que l'utilisation de l'*unicast* conduit à un gaspillage de bande passante, que le *multicast* traditionnel nécessite la mémorisation d'un grand nombre d'états de routage pour les différents groupes et un grand nombre de messages de signalisation par groupe pour maintenir ces états de routage et que le *multicast* explicite nécessite un temps de traitement croissant.

Dans le chapitre précédent, afin de limiter l'impact de la fragmentation, le protocole GXcast produit plusieurs paquets comportant chacun un nombre réduit de destinataires dans l'en-tête. Mais le protocole GXcast n'est pas adapté pour des groupes dont la taille dépasse plusieurs centaines de membres et il génère un traitement important dans chacun des routeurs. En effet, on a vu que l'optimum est atteint lorsque les paquets GXcast occupent autant d'espaces pour les adresses destinataires que pour les données ce qui a pour conséquence à diviser le débit utile par 2. Évidemment, il est préférable qu'un paquet contient plus de données que d'adresses destinataires.

Pour résoudre ces problèmes, nous décrivons dans ce chapitre, le protocole de routage *multicast* SEM (*Simple Explicit Multicast*), qui utilise une méthode efficace pour construire les arbres *multicast* et pour acheminer les paquets *multicast*. Afin de

construire un arbre *multicast*, la source encode la liste d'adresses de destination (appartenant à un groupe) dans un message *branch* de type GXcast[1]. Ce message a pour rôle de découvrir les routeurs de branchement de l'arbre *multicast*. Seuls, les routeurs de branchement de l'arbre mémorisent les états de routage *multicast* pour chaque groupe qui le traversent. Un plan de contrôle est utilisé pour permettre à chaque routeur de branchement de reconnaître ses prochains routeurs de branchement sur l'arbre pour un groupe.

Nous proposons que la source utilise le procédé de transmission *unicast* pour ses paquets *multicast* et les envoie vers ses prochains routeurs de branchement. Chaque routeur de branchement agit en tant qu'une source et les paquets sont acheminés d'un routeur de branchement à un autre suivant l'arbre construit par les messages *branch*.

Le protocole SEM est original. En effet, pour simplifier l'allocation d'une adresse *multicast*, SEM utilise la notion de canal source-spécifique (S, G) où S est l'adresse *unicast* de la source et G une adresse de groupe IP classe D. SEM réduit aussi les états de routage dans les routeurs et construit à l'aide d'un message *branch* de type GXcast un arbre des plus courts chemins basé à la source, et pas un arbre partagé comme la plupart des protocoles *multicast* traditionnel.

Nous étudions d'abord quelques propositions importantes pour résoudre le problème de la résistance au facteur d'échelle en *multicast*. Nous présentons les deux protocoles HBH [HCFCD01] et REUNITE [SEZ00] et l'impact de la présence des nœuds de branchement sur l'arbre. Par la suite nous présentons les mécanismes du protocole SEM, les algorithmes utilisés pour la construction de l'arbre ainsi que la structure des différents messages. Nous comparons ensuite le protocole SEM au protocole HBH. Finalement nous évaluons notre protocole par simulation en terme de : la diminution en taille des tables de routage par rapport à PIM-SM (source spécifique) et par rapport à HBH, le surcoût dû aux messages de contrôle par rapport à HBH et le coût de l'arbre et le temps de traitement d'un paquet dans un routeur en comparaison avec le protocole GXcast.

1. Un message *branch* ne contient pas de données.

3.1 Propositions pour résoudre le problème de la résistance au facteur d'échelle

Pour résoudre le problème de la résistance au facteur d'échelle *multicast*, plusieurs propositions sont apparues. Certaines propositions réduisent le nombre d'états de routage *multicast* par l'utilisation de tunnels [TN98] ou par l'agrégation des états de routage [REG99].

Outre les protocoles explicites étudiés dans le chapitre précédent qui éliminent les états de routage *multicast* complètement en codant explicitement la liste de destinataires dans les paquets au lieu d'utiliser une adresse de groupe *multicast* (Xcast et GXcast) certains protocoles éliminent ces états partiellement en utilisant les routeurs de branchement de l'arbre *multicast* : REUNITE [SEZ00] et HBH [HCFCD01]. Nous étudions dans le chapitre 4 une solution pour résoudre le problème de la résistance au facteur d'échelle et d'agrégation *multicast*. Cette solution réduit le nombre d'états de routage *multicast* dans un réseau en utilisant des mécanismes d'ingénierie de trafic à travers des tunnels MPLS entre les routeurs de branchement de l'arbre *multicast*.

3.1.1 Tunnellisation et agrégation d'états

Dans [TN98], un protocole de routage *multicast*, DTM (*Dynamic Tunnel Multicast*), est utilisé. DTM construit d'une façon dynamique des tunnels en se servant des états de routage d'un autre protocole de routage *multicast* traditionnel. DTM élimine les états de routage pour un groupe *multicast* dans les routeurs qui ne sont pas routeurs de branchement pour ce groupe[2]. Le mécanisme est décrit par la figure 3.1.

La destination établit la construction du tunnel en envoyant à la source une requête d'établissement du tunnel[3](étape 1 sur la figure). En recevant cette requête, un routeur intermédiaire entre la destination et la source transmet la requête inchangée vers la source (étape 2 sur la figure). Si ce routeur est un point d'extrémité potentiel du tunnel

2. Les routeurs de branchement sont ainsi considérés comme les extrémités des tunnels.
3. Ce message est similaire à un message d'adhésion classique.

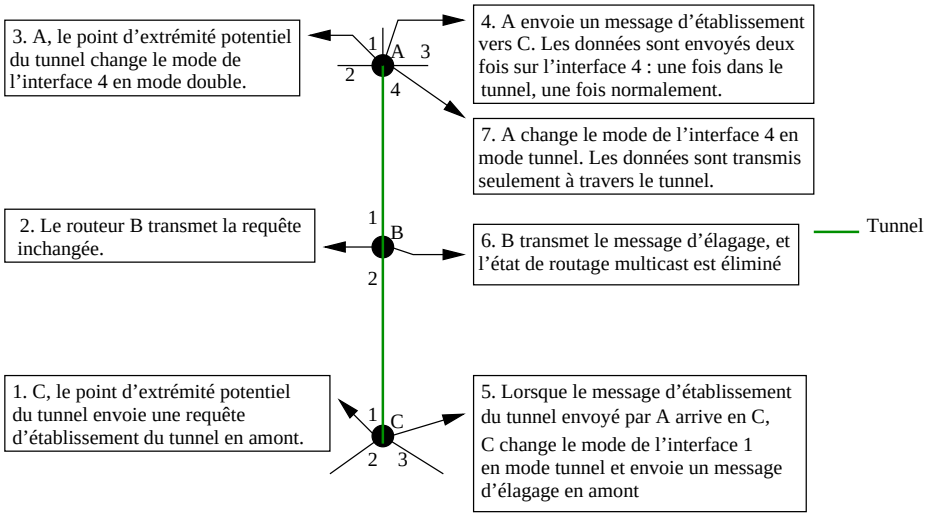


FIG. 3.1 – *L'établissement d'un tunnel dans le protocole DTM.*

(le routeur appartient déjà à l'arbre *multicast*), il change le mode de l'interface à travers laquelle il a reçu la requête (l'interface 4 sur l'exemple de la figure) en mode double (mode natif et mode tunnel) et envoie un message d'établissement du tunnel vers la destination (étapes 3 et 4 sur la figure). Une fois ce message d'établissement du tunnel arrive à la destination, celle-ci change le mode de l'interface à travers laquelle elle a reçu le message en mode tunnel et envoie un message d'élagage de l'arbre *multicast* en amont (étape 5 sur la figure). Ce message d'élagage est par la suite transmis vers le point d'extrémité du tunnel pour changer le mode de transmission de l'interface 4 du mode double en mode tunnel tout en éliminant les états de routage *multicast* de tous les routeurs sur son chemin. Pour la transmission de paquets *multicast*, la source utilise l'encapsulation des paquets *multicast* dans des paquets IP *unicast* [Per96] qui auront pour destination l'extrémité du tunnel. L'inconvénient de cette proposition est qu'une interface d'un routeur peut fonctionner en mode double où deux copies du même paquet seront envoyées l'une en mode natif et l'autre en mode tunnel.

Dans [REG99], l'agrégation des états de routage est étudiée. En effet, un état de routage est défini par une adresse IP *multicast* (adresse du groupe), une interface d'entrée et des interfaces de sorties. Si les interfaces d'entrées et les interfaces de sorties sont identiques pour deux états de routage, l'agrégation se fait en remplaçant les pré-

fixes des deux adresses IP *multicast* par le plus long préfixe commun aux deux adresses. Le nouveau état de routage contient le préfixe résultant, l'interface d'entrée et les interfaces de sorties. Il existe plusieurs types d'agrégation possible : l'agrégation stricte, l'agrégation pseudo-stricte et l'agrégation *leaky* (*cf.* figure 3.2).

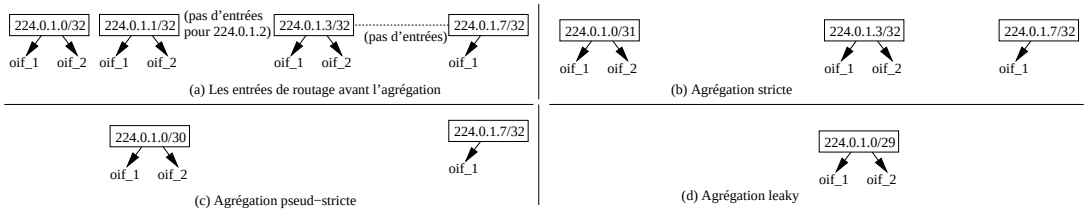


FIG. 3.2 – *Un exemple des différents types d'agrégation d'adresses.*

Dans l'agrégation stricte, les deux états de routage 224.0.1.0/32 et 224.0.1.1/32 ont les mêmes interfaces d'entrées et de sorties et ils seront remplacés par l'état de routage 224.0.1.0/31. Suite à l'absence de l'état de routage 224.0.1.2/32, l'état de routage 224.0.1.3/32 n'est pas agrégé avec les deux premiers états de routage. Dans l'agrégation pseudo-stricte, les états de routage 224.0.1.0/32, 224.0.1.1/32 et 224.0.1.3/32 seront agrégés et remplacés par l'état de routage 224.0.1.0/30. Dans l'agrégation *leaky*, un paquet peut être envoyé vers des interfaces où il peut ne pas avoir de membres pour le groupe. Si les interfaces de sorties ne sont pas identiques pour deux états de routage ayant les mêmes interfaces d'entrées, les deux adresses IP *multicast* sont aussi remplacées par le plus long préfixe commun aux deux adresses mais les interfaces de sorties sont l'union des interfaces de sorties pour tous les états de routage. Ainsi, dans notre exemple les états de routage sont remplacés par un seul état de routage ayant pour préfixe 224.0.1.0/29.

Ces deux propositions (DTM et agrégation d'adresses) dépendent de l'existence préalable d'un protocole de routage *multicast* traditionnel, comme PIM-SM [EFH+98] ou CBT [Bal97a, Bal97b]. De plus, en utilisant ces deux techniques, plusieurs copies du même paquet sont envoyées sur le même chemin.

3.1.2 Les protocoles REUNITE et HBH

REUNITE [SEZ00] et HBH [HCFCD01], utilisent les arbres *unicast* récursifs (utilisation des adresses *unicast* d'un routeur de branchement à un autre) pour assurer le service *multicast*.

REUNITE n'utilise pas donc les adresses IP de la classe D et un groupe est identifié par le couple (S, P), où S est l'adresse *unicast* de la source et P est un numéro de port alloué par la source[4]. REUNITE abandonne ainsi presque complètement le modèle d'IP *multicast* traditionnel proposé par Deering [Dee91]. REUNITE se base sur l'idée de la présence de nœuds de branchement sur l'arbre *multicast* (cette idée sera étudiée dans le sous-chapitre 3.2).

Chaque routeur REUNITE possède une table de contrôle (MCT, *Multicast Control Table*) et une table de routage (MFT, *Multicast Forwarding Table*). Seuls les routeurs de branchement pour un canal mémorisent l'état de routage *multicast* dans la table MFT. Tout autre routeur de l'arbre du canal mémorise seulement une entrée de contrôle *multicast* (dans la table MCT) et envoie simplement les paquets de données sur le chemin *unicast*.

Les paquets de données sont transmis en *unicast* vers le premier destinataire à s'abonner au groupe. Appelons $R1$ ce destinataire. Dans un routeur de branchement, les paquets de données reçus sont transmis vers chaque destinataire présent dans l'entrée MFT correspondante à (S, P). Les copies de paquets ont comme adresse de destination l'adresse des destinataires dans la MFT. La figure 3.3 illustre le fonctionnement de l'arbre *multicast* REUNITE. Une partie du chemin des données y est représentée : S envoie des paquets en *unicast* vers $R1$[5]. Lorsque $H1$ reçoit un des paquets à destination $R1$ avant de retransmettre normalement le paquet vers $H2$ comme l'indique la table de routage normale, $H1$ crée une copie du paquet et l'envoie vers $R4$[6]. $H3$ transmet, en *unicast*, les paquets puisqu'il ne possède pas d'entrée pour (S, P) dans sa

4. Le terme canal est utilisé pour représenter le couple (S, P).
5. Ces paquets arriveront à $R1$ en *unicast*.
6. Le paquet initial va continuer son chemin vers $R1$.

MFT. $H5$ crée une copie pour $R8$, et finalement $H7$ crée une copie des paquets pour $R5$ et $R6$. Le paquet à destination de $R4$ arrive en *unicast* à $R4$. Il en est de même pour $H2$, $H4$ et $H6$ avec les paquets d'adresse de destination initiale $R1$.

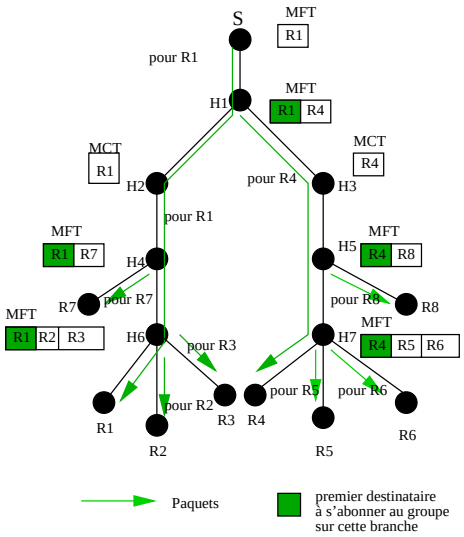


FIG. 3.3 – *Arbre REUNITE.*

Le protocole de routage *multicast* HBH essaye de résoudre quelques problèmes relatifs à la construction et la maintenance de l'arbre *multicast* REUNITE. Premièrement, HBH utilise la notion de canal source-spécifique et un groupe *multicast* est identifié par le canal (S, G) où S est l'adresse *unicast* de la source et G une adresse de groupe IP classe D. L'utilisation du modèle d'adressage *multicast* IP préserve la compatibilité avec les protocoles traditionnels de routage *multicast*. Deuxièmement, dans REUNITE, les routes pour certains destinataires peuvent changer lors du départ d'un autre membre du groupe, un comportement non souhaitable du point de vue de la qualité de service. En particulier, si le premier routeur qui a adhéré à un groupe quitte ce groupe, l'entretien de l'arbre devient très compliqué. Troisièmement, HBH résout le problème de routage asymétrique présent dans REUNITE. En effet, pour certains destinataires, les abonnements au groupe peuvent se réaliser selon des routes non optimales, c'est-à-dire, le chemin le plus court n'est pas toujours garanti. Enfin, l'arbre construit par REUNITE peut conduire à la duplication de paquets sur certains liens

[HCFCD01].

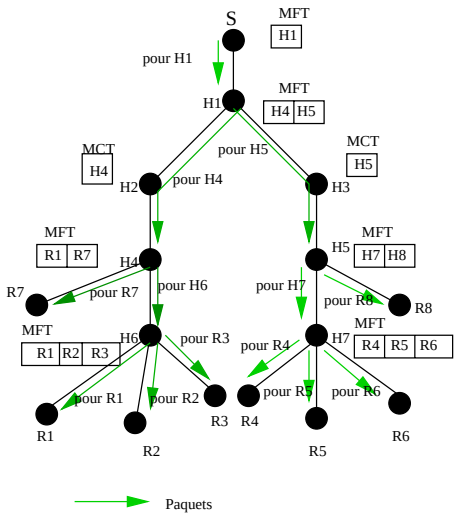


FIG. 3.4 – *Arbre HBH.*

Dans HBH les tables de routage (MFT) contiennent des adresses des prochains routeurs de branchement au lieu des adresses des destinataires. La figure 3.4 montre un arbre HBH. S envoie les paquets en *unicast* vers $H1$. $H1$ crée deux copies des paquets, qui seront envoyées respectivement à $H4$ et $H5$ (les prochains routeurs de branchement). $H3$, ayant seulement une table de contrôle et aucun état de routage dans sa MFT pour le canal, transmet les paquets en *unicast* vers $H5$. $H5$ reçoit les paquets et envoie une copie à $H7$ et une copie à $R8$. Finalement, $H7$ envoie une copie des paquets à $R4$, $R5$ et $R6$. HBH a essayé d'éliminer les états de routage MFT de tous les routeurs de l'arbre qui ne sont pas de branchement[7] en préservant seulement les états de contrôle MCT dans ces routeurs.

7. HBH garde toujours des états de routage MFT dans les routeurs de branchement.

3.2 L'impact de la présence des nœuds de branchement

3.2.1 Observation

Dans le cas général des protocoles de routage *multicast*, pour chaque groupe *multicast*, tous les routeurs appartenant à une branche de l'arbre entre la source et les destinataires mémorisent un état de routage pour cette groupe. Considérons le réseau illustré dans la figure3.5 (canal *multicast* formé d'une source S et d'un groupe *multicast* G).

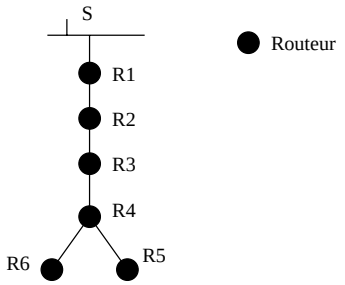


FIG. 3.5 – *Un exemple d'un arbre multicast pour le canal* (S, G).

Supposons qu'il existe des destinataires membres du groupe G attachés au routeur $R5$, alors lorsqu'on utilise un protocole de routage *multicast* traditionnel tous les routeurs intermédiaires entre la source et $R5$ ($R1$, $R2$, $R3$, $R4$) mémorisent un état de routage *multicast* correspondant à ce groupe G même s'ils n'ont aucun membre directement lié et même s'ils ne prennent aucune décision de duplication. Prenons le même réseau avec plusieurs groupes *multicast* (une source S et deux groupes $G1$ et $G2$). Supposons qu'il existe des destinataires membres du groupe $G1$ attachés à $R5$ et des destinataires membres du groupe $G2$ attachés à R6. $R1$, $R2$, $R3$, $R4$ doivent mémoriser ainsi des états de routage *multicast* correspondants aux deux groupes $G1$ et $G2$ même s'ils n'ont aucun membre directement lié. Avec la croissance du nombre de groupes, le nombre d'état de routage *multicast* dans les routeurs croît ce qui implique que le taux d'occupation des ressources des routeurs croît : la durée de recherche d'une entrée dans la table de routage et donc la vitesse de commutation est fortement influencé par la taille de la table de routage *multicast*.

La figure 3.6, extraite de [SEZ00], représente une topologie *multicast* résultante de plusieurs essais de *traceroute*[8] entre une source située à l'université Carnegie Mellon vers 15 destinataires différents répartis sur l'Internet. Dans cette topologie, nous remarquons qu'il existe seulement 8 routeurs de branchement parmi les 97 routeurs faisant partie de l'arbre, soit environ 8%.

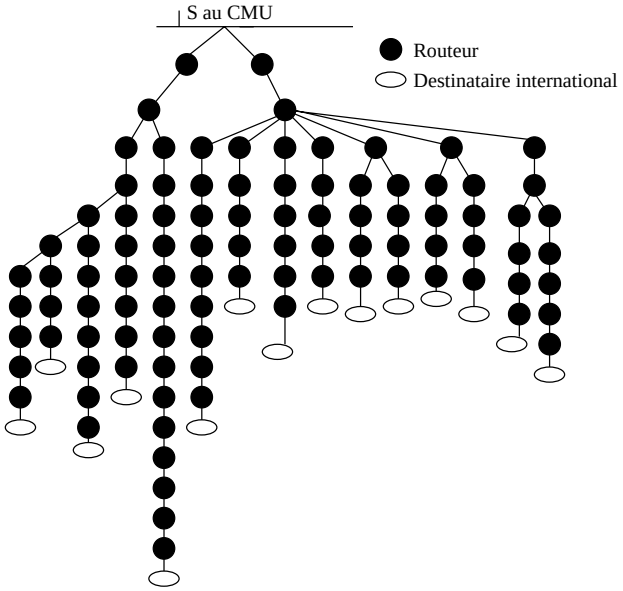


FIG. 3.6 – *Traceroutre entre une source située au CMU vers* 15 *destinataires différents.*

Ce faible ratio semble être vérifié même dans un plan national, la figure 3.7 représente aussi une topologie *multicast* résultante de plusieurs essais de *traceroute* entre une source située à l'université de Rennes1 vers 5 destinataires français différents. Dans cette topologie, nous remarquons qu'il existe 4 routeurs de branchement parmi les 30 routeurs faisant partie de l'arbre, soit environ 13%.

Nous proposons que seuls les routeurs de branchement sur un arbre *multicast* devraient mémoriser les états de routage. L'existence des états de routage *multicast* dans les autres routeurs consomme inutilement les ressources du réseau.

8. *Traceroute* est un utilitaire permettant de déterminer le trajet emprunté par un paquet IP sur l'Internet.

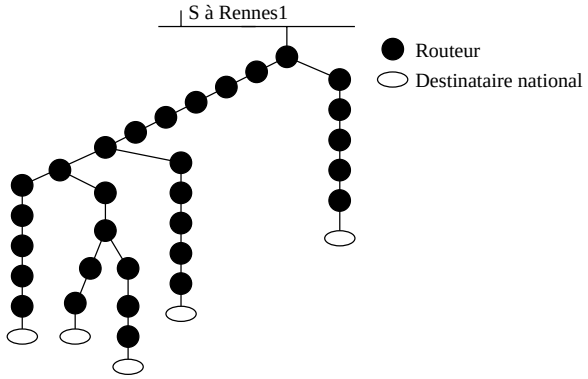


FIG. 3.7 – *Traceroutre entre une source située à l'IRISA vers* 5 *destinataires français différents.*

3.2.2 La diminution en taille des tables de routage *multicast*

Pour un arbre source-spécifique basé à la source, un état de routage *multicast* est identifié par le canal (S, G) où S est l'adresse de la source et G est l'adresse du groupe.

Pour un arbre *multicast* T associé à un canal *mulicast* (S, G), nous considérons le paramètre α comme étant le nombre moyen d'états dans la table de routage *multicast* par routeur :

$$\alpha(T) = \frac{N_e}{N_T} \tag{3.1}$$

où N_e est la somme du nombre d'états *multicast* dans tous les routeurs sur l'arbre (c'est-à-dire le nombre total d'états correspondant à (S, G), et N_T le nombre de routeurs sur l'arbre. Une analyse similaire est faite dans [TN98].

Pour un arbre source-spécifique, chaque routeur sur l'arbre mémorise un état (S, G) dans sa table de routage. Dans ce cas $N_e = N_T$ et la valeur du paramètre α atteint son maximum 1. La valeur minimale de α, α_{min}, pour n'importe quel arbre T est définie par l'équation suivante :

$$\alpha_{min}(T) = \frac{N_b + N_l + N_s}{N_T} \tag{3.2}$$

où N_b étant le nombre de routeurs de branchement sur l'arbre, N_l le nombre de routeurs d'extrémité (absence de routeurs en aval pour ce type de routeur) sur l'arbre,

N_s le nombre de routeurs source sur l'arbre (toujours égal à 1 dans le cas d'unique canal *multicast* source-spécifique[9]), et N_T le nombre total de routeurs sur l'arbre.

Nous pouvons déduire que la valeur du paramètre α pour la topologie de la figure 3.7 est inférieure à 34% lorsque des tunnels entre les routeurs de branchement sont utilisés. Le protocole SEM que nous présentons ci-dessous utilise des tunnels entre les routeurs de branchement et se propose donc de réduire considérablement la taille globale des tables de routage *multicast*.

3.3 Simple Explicit Multicast Protocol

Pour simplifier l'allocation d'une adresse *multicast*, SEM utilise la notion de canal source-spécifique, noté (S, G) où S est l'adresse *unicast* de la source et G est une adresse *multicast* standard. Une plage d'adresses *multicast* est utilisée pour identifier facilement les groupes SEM des autres groupes *multicast* traditionnels.

Comme nous allons le voir dans les sous-chapitres suivants, pour construire l'arbre *multicast*[10], le protocole SEM utilise les deux messages *branch* et *previous_branch*. De plus, le protocole SEM utilise un message *alive* pour maintenir l'arbre. L'abonnement à l'arbre et le désabonnement de l'arbre se font avec les deux messages *join* et *leave* source spécifique qui atteignent toujours la source. Ces 2 messages sont identiques à ceux utilisés pour le service SSM [Bha03, HC03]. Seuls les routeurs de branchement de l'arbre gardent un état de routage pour le canal (S, G) dans leur table de routage *multicast* (appelée ci-après TRM). $TRM(S, G)$ (*cf.* figure 3.8) représente l'entrée associée au canal (S, G) dans TRM. Les entrées dans chaque $TRM(S, G)$ sont S, l'adresse de la source, G, l'adresse du groupe, p_B l'adresse du routeur de branchement précédent, et les adresses des prochains routeurs de branchement sur l'arbre.

9. Une source peut être un nœud de branchement aussi et il est compté une seule fois.
10. L'arbre *multicast* est formé uniquement des routeurs de branchement.

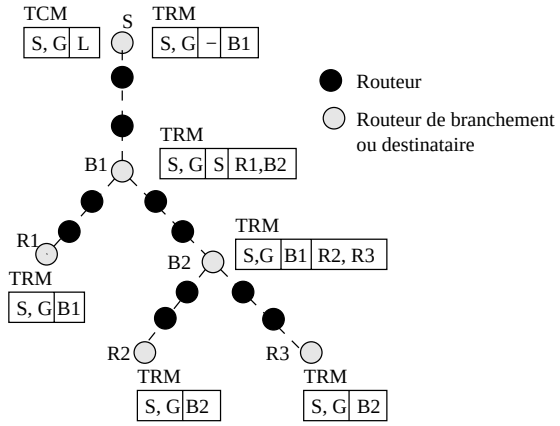


FIG. 3.8 – *Les tables de routage multicast (TRM).*

3.3.1 Les mécanismes du protocole SEM dans les destinataires et dans les routeurs

Un destinataire désirant s'abonner au canal (S, G) émet un message d'abonnement IGMP à destination du canal (S, G). Quand le routeur désigné (DR : *designated router*) associé au destinataire reçoit ce message, il envoie à la source S un message d'adhésion source-spécifique *join(S,G)*. Lorsque la source reçoit ce message, il maintient dans une table de contrôle *multicast* (TCM) la liste L des adresses de tous les routeurs DR ayant des destinataires appartenant au canal (S, G). $TCM(S, G)$ (*cf.* figure 3.8) représente l'entrée associée au canal (S, G) dans TCM. Les entrées dans chaque $TCM(S, G)$ sont S, l'adresse de la source, G, l'adresse du groupe, et la liste L.

Les routeurs intermédiaires n'ont pas besoin de garder un état de routage ou un état de contrôle correspondant au canal (S, G) mais il est nécessaire que les DR associés aux destinataires connaissent l'adresse de la source et l'adresse du routeur de branchement précédent pour le canal. La version actuelle d'IGMP, IGMPv3 [CDT02], permet la découverte de la source ainsi que l'appartenance à un groupe source spécifique d'une station située sur le sous-réseau dont est responsable le DR. Les destinataires SEM n'utilisent pas de messages de contrôle additionnels pour s'abonner aux groupes SEM : ils utilisent IGMP. La figure 3.9 décrit le processus d'abonnement d'un destinataire à un canal (S, G).

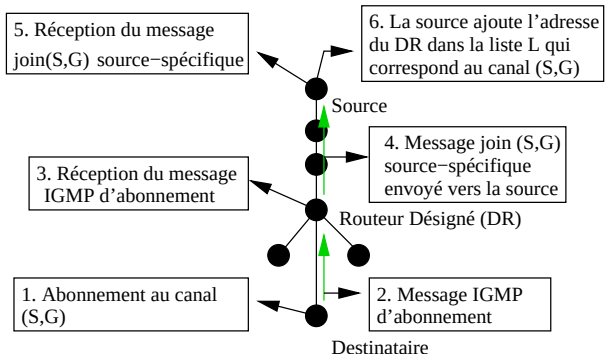


FIG. 3.9 – *L'abonnement à un canal* (S, G) *dans le protocole SEM.*

Le destinataire s'abonne au canal (S, G) en envoyant un message d'adhésion IGMP au DR de son réseau local (étape 1 et 2 sur la figure 3.9). Le DR reçoit ce message d'adhésion IGMP et envoie à la source un message *join(S,G)* source spécifique (étape 3 et 4 sur la figure 3.9). À la réception de ce message, la source ajoute l'adresse du DR à la liste L pour le canal (S, G) [11] (étape 5 et 6 sur la figure 3.9).

Le protocole SEM utilise des messages *alive* entre les routeurs de branchement pour maintenir l'arbre. Les messages *alive* sont périodiquement envoyés en *unicast* par les destinataires (ou bien par le DR du réseau local des destinataires) vers le routeur de branchement précédent sur l'arbre, servant ainsi à rafraîchir l'état de routage dans ce routeur auquel le destinataire s'est connecté. Un routeur B qui reçoit du routeur R un message *alive(S,G,R)* [12] qui lui est destiné rafraîchit l'entrée qui correspond à R dans sa table de routage *multicast* $TRM(S, G)$ (cette entrée est appelée ci-après $TRM(S, G).R$). Le message *alive(S,G,R)* est rejeté et un nouveau message *alive(S,G,B)* est envoyé au routeur de branchement précédent. Le message *alive(S,G,B)* est périodique, utilisé pour la maintenance de l'arbre et ne se produit pas directement après le rejet du message *alive(S,G,R)*.

11. Si aucune liste n'existe pour le canal (S, G), une nouvelle liste est créée.

12. $Alive(S, G, R)$ désigne le message *alive* envoyé en *unicast* par le routeur R vers le routeur de branchement précédent pour le canal (S, G).

3.3.2 La construction de l'arbre dans SEM

Dans SEM, la source maintient les adresses de tous les DR qui ont envoyé des messages d'adhésion source-spécifique pour un canal (S, G) et encode explicitement, dans l'en-tête SEM d'un message *branch*, la liste L d'adresses de ces DR. Chaque routeur, le premier et ceux tout le long du chemin, analyse l'en-tête du message *branch*, classe les destinations en sous-listes L_i selon leur prochain routeur, génère un message *branch* avec l'en-tête SEM appropriée et l'envoie vers chacun des prochains routeurs [13]. Le rôle du message *branch* est de découvrir les routeurs de branchement de l'arbre *multicast*. La liste L_i des destinations dans un message pour un routeur donné inclut seulement les destinations que ce routeur permet d'atteindre. La figure 3.10 décrit le traitement d'un message *branch* [14] dans un routeur SEM.

Lorsque ce message *branch* atteint un routeur qui n'est pas de branchement pour le canal (S, G), il est envoyé sans changement à l'unique prochain routeur pour toutes les destinations. Si le routeur est un routeur de branchement pour la liste, il vérifie la présence d'une entrée dans la table de routage qui correspond au canal (S, G). Si $TRM(S, G)$ existe, elle est mise à jour [15]. Si cette entrée n'existe pas, une entrée de routage est créée dans la table de routage TRM du routeur de branchement. L'entrée contient l'adresse source, l'adresse *multicast* pour le groupe, l'adresse du routeur de branchement précédent et la liste des adresses des prochains routeurs de branchement (la liste est au début vide). Le routeur de branchement remplace la valeur du routeur de branchement précédent dans l'en-tête SEM du nouveau message *branch* par sa propre adresse. Dans les deux cas, le routeur renvoie des messages *branch* pour toutes les listes L_i.

Le routeur de branchement envoie également un message *previous_branch* vers le routeur de branchement précédent (l'adresse est déduite du message *branch*). La figure

13. Ce mécanisme est similaire à celui du protocole GXcast qui peut être utilisé comme nous le verrons ultérieurement.

14. *Branch(S,G,p_B,L)* désigne le message *branch* de la liste L pour un canal (S, G) envoyé par le routeur p_B.

15. $TRM(S, G)$ est initialisée à vide.

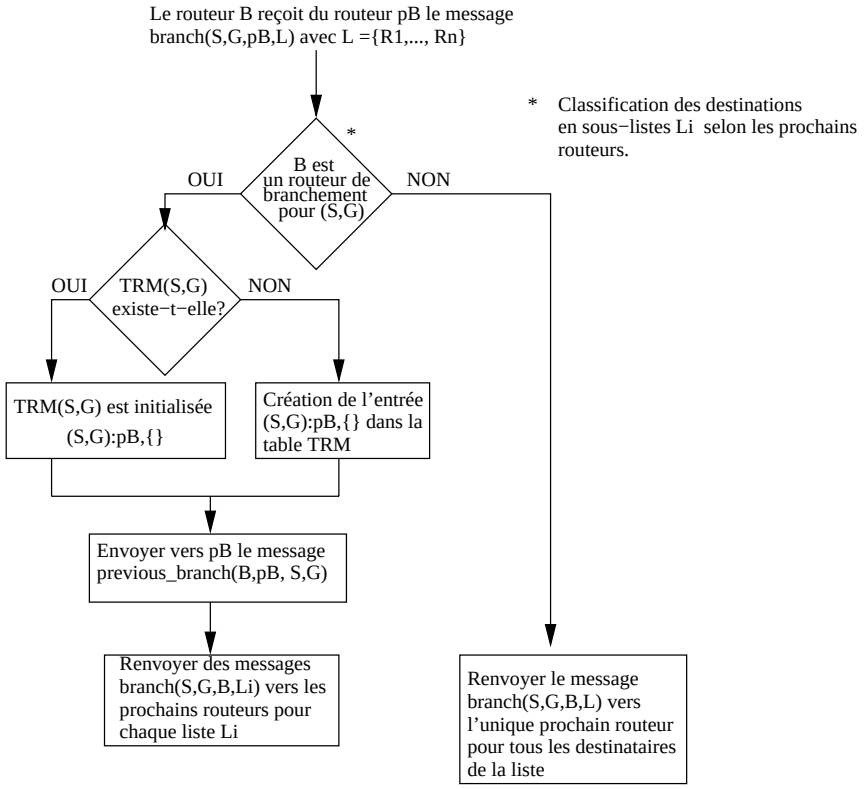


FIG. 3.10 – *Le traitement du message branch dans un routeur SEM.*

3.11 décrit le traitement d'un message *previous_branch*[16] dans un routeur SEM.

Le message *previous_branch* reçu par le routeur de branchement précédent met à jour l'état correspondant au canal (S, G). Ce message est donc employé pour informer le routeur de branchement précédent de la découverte de ses prochains routeurs de branchement. À la fin de cette opération, nous obtenons un chemin de la source vers chaque routeur de destination DR en utilisant les adresses des prochains routeurs de branchement. La source peut ainsi envoyer des paquets de données au différents destinataires du canal (S, G).

Exemple : considérons le groupe représenté sur la figure 3.12, comportant une source et six destinataires A, B, C, D, E, F. Dans ce cas A, B, C, D, E et F génèrent des messages d'adhésion IGMP aux DR associés à leurs sous-réseaux.

16. *Previous_branch(B,p_B,S,G)* désigne le message *previous_branch* pour le canal (S, G) envoyé par

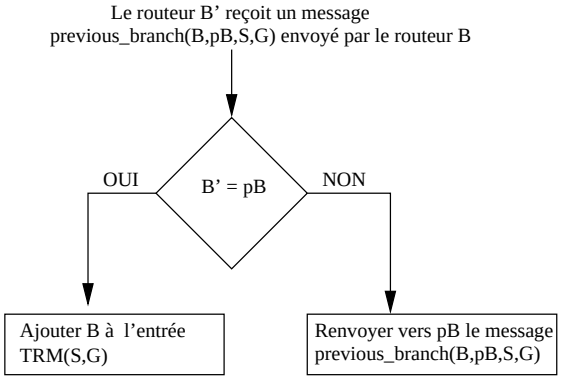


FIG. 3.11 – *Le traitement du message previous_branch dans un routeur SEM.*

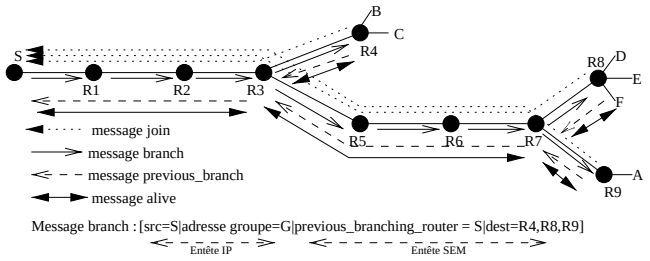


FIG. 3.12 – *La construction de l'arbre SEM.*

En recevant les messages IGMP, les routeurs $R4$, $R8$ et $R9$ envoient chacun un message *join* source-spécifique vers la source S. S à son tour envoie un message *branch* vers le premier routeur $R1$ avec la liste des routeurs *multicast* ($R4$, $R8$ et $R9$) dans son en-tête SEM. L'en-tête IP du message *branch* envoyé par S à $R1$ contient l'adresse S de la source et l'adresse G du groupe. L'en-tête SEM du message *branch* contient l'adresse du routeur de branchement précédent et la liste de tous les routeurs de destination (*cf.* figure 3.12). La valeur initiale du routeur de branchement précédent est l'adresse de la source S elle-même.

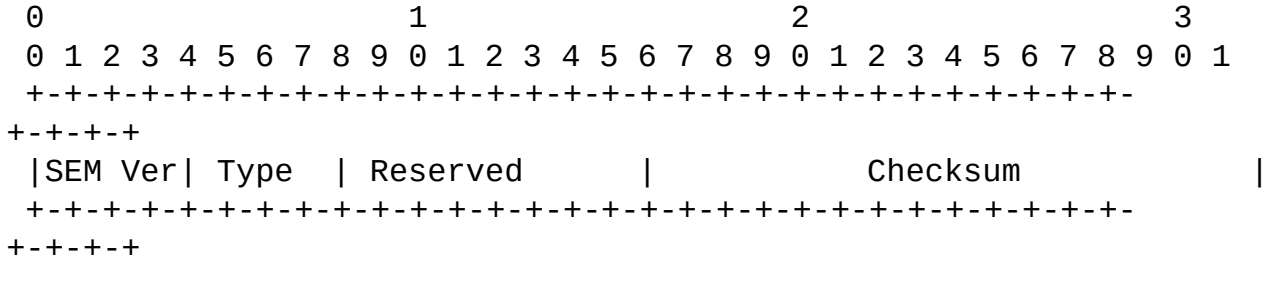
Dans notre exemple, aucun état de routage est crée dans $R1$ et $R2$ pour le canal (S, G). Un état de routage *multicast* est créé dans $R3$ (une entrée est insérée dans la table de routage *multicast* (TRM) de $R3$). Cet état de routage *multicast* contient l'adresse S de la source, l'adresse G du groupe, une liste vide pour les prochains

le routeur B au routeur p_B.

routeurs de branchement et la valeur de l'adresse S de la source du message pour le champ *previous_branching_router*. Les nouveaux messages *branch* envoyés par $R3$ contiennent S, G, la liste appropriée de destinations (un message *branch* contient $R4$ et le deuxième message *branch* contient $R8$ et $R9$), et $R3$ dans le champ *previous_branching_router*. En appliquant le même processus, aucun état ne sera créé en $R5$ et en $R6$. À la fin de cette opération, les états (S, G) dans $S, R3, R7$ contiendra respectivement $R3$, $(R4, R7)$ et $(R8, R9)$ en tant que prochains routeurs de branchement pour le groupe.

3.3.3 La structure des messages *branch* et *previous_branch*

Aux messages de contrôle de SEM est associé un numéro de protocole spécifique. Le type de protocole de l'en-tête IP est `SEM_PROTO`. Les messages de contrôle SEM sont de type *unicast* (*previous_branch* et *alive*) et *multicast* (*branch*). L'entête SEM commence toujours par les 4 octets suivants :



Le numéro de la version de SEM (`SEM Ver`) est 1. Le `Checksum` est calculé de manière similaire à TCP. Les types d'un message SEM sont :

0 = *branch*

1 = *previous_branch*

2 = *join*

3 = *leave*

4 = *alive*

5 = données

Le message *branch* est toujours placé dans un paquet IP *multicast*. L'en-tête IP du datagramme contenant le message *branch* portera l'adresse S de la source et l'adresse G du groupe (comme destination). L'en-tête SEM contient le champ *previous_branching_router* qui représente l'adresse du routeur de branchement précédent [17] et la liste L des adresses de tous les DR qui ont envoyés à la source un message d'adhésion au canal (S, G) (*cf.* annexe C.1).

Le message *previous_branch* est toujours placé dans un paquet IP *unicast*. L'en-tête IP du datagramme contenant le message *previous_branch* contient le routeur lui-même comme adresse source et le routeur de branchement précédent comme adresse de destination. L'en-tête SEM du message *previous_branch* contient les adresses de la source et du groupe (le canal (S, G)) (*cf.* annexe C.2).

Le message *alive* est toujours placé dans un paquet IP *unicast*. L'en-tête IP du datagramme contenant le message *alive* envoyé par un routeur B vers p_B, le routeur de branchement en amont sur l'arbre, portera l'adresse B comme adresse source et p_B comme adresse de destination. L'en-tête SEM contient les deux adresses S et G qui représente le canal (S, G) (*cf.* annexe C.5).

Un message *join* ou un message *leave* est toujours placé dans un paquet IP *unicast*. L'en-tête IP du datagramme contenant ce message envoyé par un routeur R vers la source S, portera l'adresse R comme adresse source et S comme adresse de destination. L'en-tête SEM contient les deux adresses S et G qui représente le canal (S, G) (*cf.* annexes C.3 et C.4) [18].

Les paquets de données envoyé en mode SEM sont des paquets IP *unicast*. L'en-tête IP du datagramme contenant les données en mode SEM contient l'adresse de la source et l'adresse du prochain routeur de branchement. L'en-tête SEM contient seulement l'adresse du groupe (*cf.* annexe C.6).

17. Le message initial contient l'adresse de la source S comme valeur de ce champ.

18. Une autre façon d'implémenter ces deux messages consistent à utiliser le même format des messages *join/prune* comme cela se fait dans PIM-SM.

3.3.4 La livraison des paquets de données en mode SEM

Quand la source envoie un paquet vers un groupe, l'état du canal correspondant est examiné [19]. Le paquet est envoyé directement en mode SEM (en *unicast*) aux prochains routeurs de branchement. Quand les prochains routeurs de branchement reçoivent le paquet, la même opération est répétée. Ainsi si le routeur recevant le paquet n'est pas le prochain routeur de branchement pour ce paquet, il le renvoie en *unicast* vers le prochain routeur de branchement. Quand le paquet arrive à un DR, le champ destination du paquet est remplacé par l'adresse G du groupe pour atteindre ainsi tous les destinataires dans les sous-réseaux du DR.

3.3.5 La maintenance de l'arbre SEM

Lorsqu'un DR n'a plus de destinataires pour un canal (S, G) dans son sous-réseau, il cesse d'envoyer les messages *alive(S,G)* vers le routeur de branchement précédent. Ce dernier (routeur de branchement précédent) élimine l'état de routage correspondant après un certain temps et génère un message *leave* source-spécifique (envoyé directement à la source). En recevant ce message, la source élimine l'état correspondant et envoie un nouveau message *branch* pour reconstruire l'arbre (*cf.* figure 3.13).

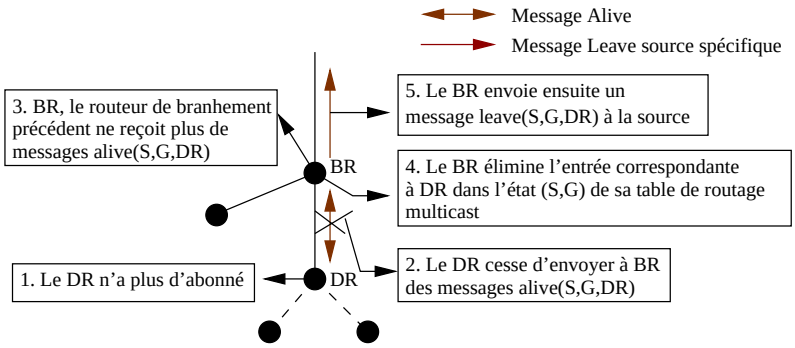


FIG. 3.13 – *Le cas d'un DR qui n'a plus d'abonné.*

En outre, quand un DR ou un routeur de branchement tombe en panne, le routeur de branchement précédent ne recevra plus les messages *alive* et élimine ainsi l'état de

19. La livraison des paquets de données en mode GXcast sera étudiée dans le sous-chapitre 3.4.5

routage. Il envoie aussi un message *leave* source spécifique à la source qui envoie à son tour un nouveau message *branch* pour reconstruire l'arbre.

Un temporisateur à la source est associé à l'état de contrôle pour chaque canal (S, G). Un message *branch* n'est pas envoyé directement après la réception d'un message *join* ou d'un message *leave*. La réception d'un message *join* ou *leave* arme le temporisateur associé au canal (S, G). La source attend que ce temporisateur expire pour envoyer un message *branch*.

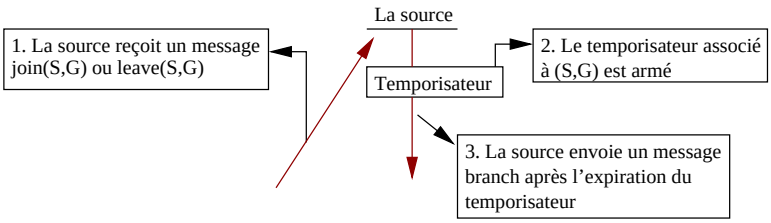


FIG. 3.14 – *Le temporisateur associé à un canal et les messages join et leave.*

La figure 3.14 décrit le comportement de la source à la réception d'un message *join(S,G)* ou *leave(S,G)* en tenant compte de l'expiration du temporisateur associé au canal (S, G).

3.4 La comparaison entre le protocole SEM et le protocole HBH

Il y a beaucoup de similitudes entre le protocole SEM et le protocole HBH mais également quelques différences notamment vis-à-vis de l'acheminement des paquets et la gestion du plan de contrôle. Nous décrivons brièvement les mécanismes de gestion de l'arbre dans les deux protocoles REUNITE et HBH avant de faire la comparaison avec le protocole SEM.

3.4.1 Le mécanisme de gestion de l'arbre REUNITE

Dans REUNITE, des messages *join* sont générés périodiquement par chaque destinataire et sont envoyés en *unicast* vers la source du groupe. Des messages *tree* sont générés périodiquement par la source du groupe et sont envoyés [20] en direction de tous les destinataires de la table MFT de la source. Lorsqu'un nouveau destinataire s'abonne au groupe, il envoie un message *join* vers la source. Si le message *join* est intercepté par un routeur ayant un MCT non vide, ce routeur devient un routeur de branchement. Le routeur élimine alors la table de contrôle MCT et crée une table de routage MFT à la place.

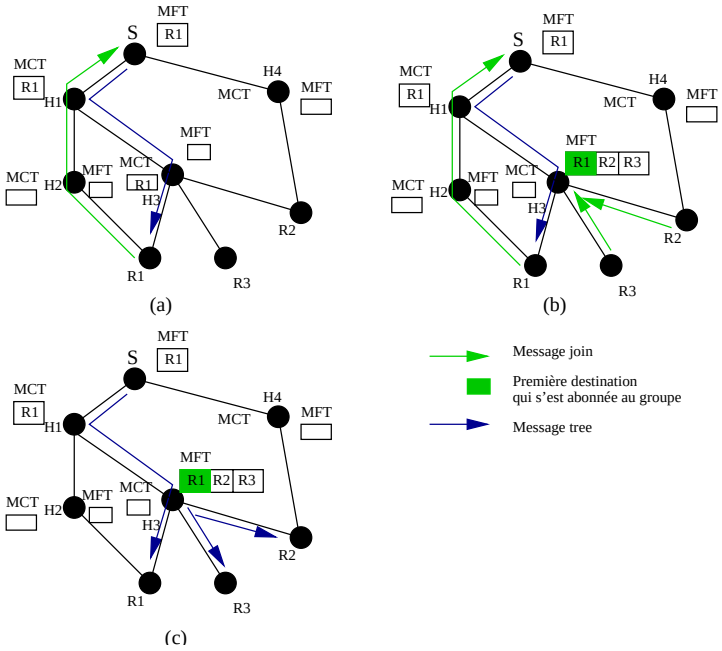


FIG. 3.15 – *Le mécanisme de gestion de l'arbre REUNITE.*

Pour mieux comprendre le mécanisme de gestion de l'arbre REUNITE, prenons l'exemple de la figure 3.15. Supposons les routes *unicast* suivantes : $R1 \rightarrow H2 \rightarrow H1 \rightarrow S$; $S \rightarrow H1 \rightarrow H3 \rightarrow R1$; $R2 \rightarrow H3 \rightarrow H1 \rightarrow S$; $S \rightarrow H4 \rightarrow R2$; $S \rightarrow H1 \rightarrow H3 \rightarrow R3$ et $R3 \rightarrow H3 \rightarrow H1 \rightarrow S$.

20. Les messages *tree* sont des messages *unicast* mais on les considère comme *multicast* puisqu'ils vont générer d'autres *tree* qui atteindront tous les destinataires du groupe.

Supposons maintenant que $R1$ s'abonne au groupe, ensuite $R2$ et finalement $R3$. $R1$ s'abonne au groupe en envoyant un message ***join(S,G,R1)*** vers S. Ce message arrive à S qui ajoute $R1$ à sa MFT. S commence à envoyer ***tree(S,G,R1)*** sur l'arbre. Ces messages créent des entrées $(S, G, R1)$ dans la MCT de $H1$ et $H3$. Les données suivent le même chemin vers $R1$. Ensuite, quand $R2$ s'abonne en envoyant un ***join(S,G,R2)*** vers S, ce message est intercepté par $H3$ car ce routeur a déjà installé l'état pour ce canal. $H3$ crée une MFT avec $R1$ le prochain routeur[21] à suivre par les paquets et rajoute $R2$ comme destinataire. Également, $R3$ est ajouté dans la MFT comme destinataire. Quand des paquets sont reçues de S vers $R1$, deux copies sont créées et envoyées à $R2$ et $R3$. Remarquons que $R1$ et $R3$ reçoivent les paquets émis par S à travers le chemin le plus court. Ce n'est pas pourtant le cas pour $R2$.

Quand un routeur REUNITE reçoit un paquet, il extrait du paquet les adresses de source et destination et le numéro du port de la source et consulte ensuite la table MFT. Si l'entrée n'est pas trouvée, alors une deuxième consultation est faite dans la table de routage *unicast*. Ainsi, quand un paquet *unicast* est reçu, deux consultations sont exigées. REUNITE ne propose aucune solution pour ce problème important mais il suppose que puisque la consultation de la table MFT est basée sur une correspondance exacte de l'adresse et non pas sur une correspondance du plus long préfixe, cette double consultation est rapide.

3.4.2 Le mécanisme de gestion de l'arbre HBH

HBH utilise trois types de messages dans la construction de l'arbre *multicast* : *join*, *tree* et *fusion*. Les messages *join* sont périodiquement envoyés en *unicast* par les destinataires vers la source, et servent à rafraîchir l'état de routage (l'entrée dans la MFT) dans le routeur auquel le destinataire s'est branché. La source envoie périodiquement en *multicast* un message *tree*[22] pour chaque canal (S, G) qui sert à rafraîchir la structure de l'arbre. Des messages *fusion* sont envoyés en *unicast* par des routeurs de bran-

21. Ce champ est appelé dst dans [SEZ00].
22. Nous étudions le mode de transmission du message *tree* ultérieurement.

chement potentiels afin de construire la structure de distribution de l'arbre, à l'aide des messages *tree* reçus depuis la source. La figure 3.16 présente les algorithmes de traitement des trois types de messages utilisés dans HBH.

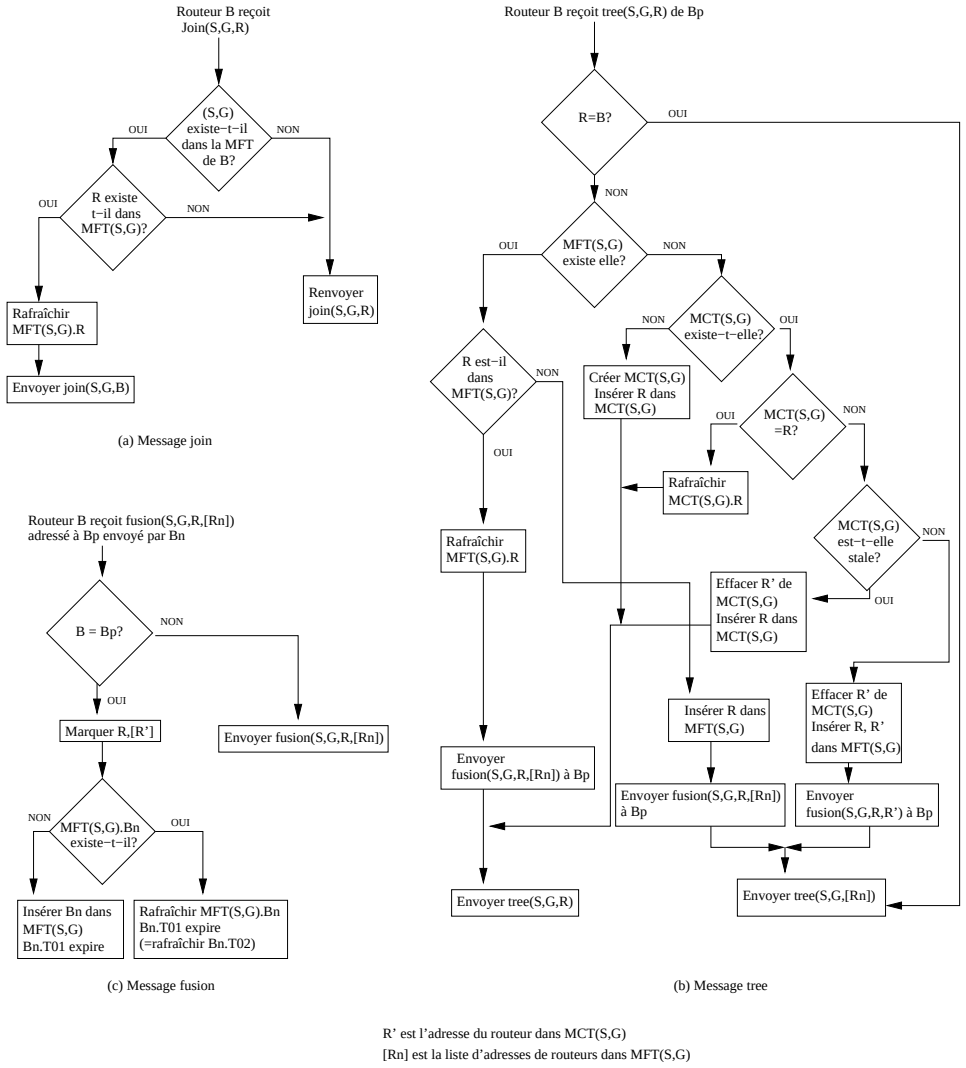


FIG. 3.16 – *Le traitement des messages dans HBH.*

Chaque routeur HBH dans l'arbre (S, G) a soit une MCT s'il n'est pas nœud de branchement soit une MFT pour (S, G) s'il l'est. La MCT pour un canal (S, G) ne peut avoir qu'une entrée, à laquelle sont associés deux timers, $t1$ et $t2$. Quand $t1$ ex-

pire, l'entrée devient *stale*[23]. À l'échéance de $t2$, l'entrée (la MCT par conséquent) est détruite. Un nœud qui est sur l'arbre du canal (S, G) mais qui ne fait pas de branchement aura une MCT pour (S, G). L'entrée dans la MFT peut aussi être marquée[24].

Prenons le même exemple du paragraphe précédent. $R1$ s'abonne au canal (S, G) et S commence à envoyer des messages *tree(S,G,R1)*. Ces messages créent une MCT pour (S, G) (contenant $R1$) dans $H1$ et $H3$. Quand $R2$ commence à envoyer un *join(S,G,R2)* en s'abonnant au groupe, ce message n'est pas intercepté et atteint la source qui commence à envoyer des *tree(S,G,R2)* (le premier *join* émis par le destinataire n'est jamais intercepté et arrive toujours à la source). Les messages *tree(S,G,R2)* émis par la source créent l'état pour le canal (S, G) dans la MCT de $H4$ (*cf. figure 3.17b*).

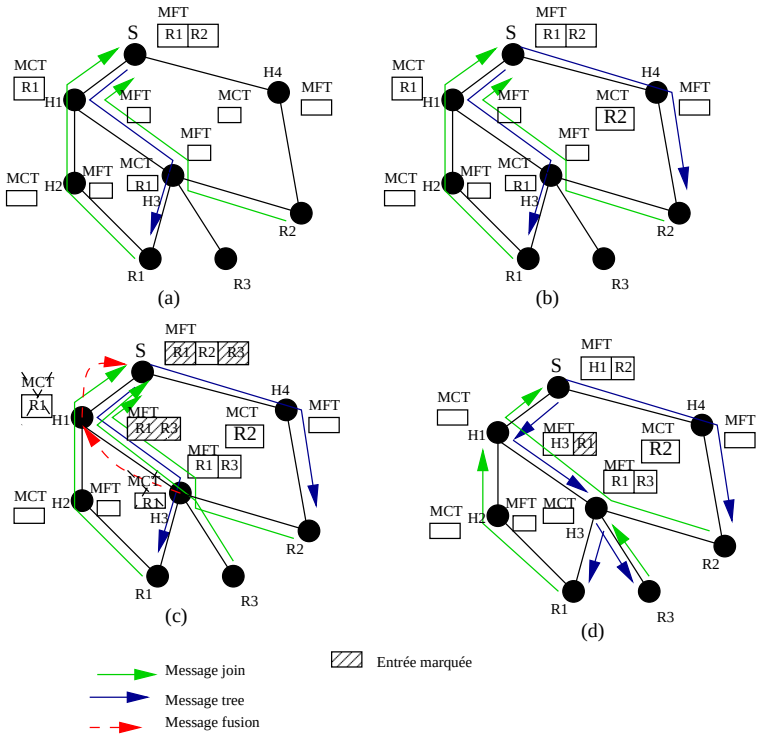


FIG. 3.17 – *Le mécanisme de gestion de l'arbre HBH.*

$R3$ à son tour envoie un *join(S,G,R3)* vers S qui commence à envoyer des *tree(S,G,R3)*.

23. Une entrée *stale* est utilisée pour le routage de paquets *multicast* sur l'arbre, mais ne produit pas de message *tree* en aval de l'arbre.

24. Une entrée marquée produira un message *tree* en sortie mais ne traitera pas les paquets de données.

Dès que $H1$ commence à recevoir deux messages *tree* différents, il envoie un message *fusion(S,G,R1,R3)* vers la source. La réception du *fusion* par S entraîne l'ajout de $H1$ dans la table MFT de S, et le marquage de $R1$ et $R3$. De la même façon que $H1$, $H3$ reçoit des messages *tree(S,G,R1)* et *tree(S,G,R3)* et envoie par conséquent un *fusion(S,G,R1,R3)* vers $H1$. À la réception du message *fusion(S,G,R1,R3)*, $H1$ ajoute $H3$ dans sa table MFT et marque les deux entrées $R1$ et $R3$. La MFT de $H3$ contient maintenant $R1$ et $R3$. Les *join(S,G,R1)* subséquents seront interceptés par $H1$ (et rafraîchiront l'entrée marquée pour $R1$ dans la MFT de $H1$). Les *join(S,G,R3)* rafraîchiront l'entrée $R3$ dans la MFT de $H3$. Les paquets sont adressés par S à $H1$ puis à $H3$, qui les envoie à $R1$ et envoie une copie à $R3$. Comme S ne recevra plus de *join* issus de $R1$ et $R3$, les entrées marquées de $R1$, $R3$ disparaissent à l'échéance des temporisateurs correspondants. La structure finale est celle de la figure 3.17d.

3.4.3 Le mécanisme de gestion de l'arbre SEM

SEM utilise trois messages dans la construction de l'arbre *multicast* : *join*, *branch* et *previous_branch*. De plus, des messages *alive* remplacent les messages *join* après l'envoi du premier *join* vers la source et ils sont périodiquement envoyés en *unicast* par les destinataires vers le routeur de branchement précédent sur l'arbre, servant ainsi à rafraîchir l'état de routage dans ce routeur auquel le destinataire s'est connecté.

La source envoie un message *branch* de type GXcast (*cf.* sous-chapitre 3.4.4) qui sert à découvrir les routeurs de branchement sur l'arbre. Des messages *previous_branch* sont envoyés en *unicast* par des routeurs de branchement potentiels afin de construire la structure de distribution de l'arbre, à l'aide des messages *branch* reçus depuis la source.

Un destinataire envoie le premier message *join* en *unicast* vers la source. La source envoie un message *branch* de type GXcast sur chaque canal (S, G). Contrairement à REUNITE et à HBH, SEM élimine toute présence d'un état de routage (équivalent d'une MFT) ou d'un état de contrôle (équivalent d'une MCT) dans les routeurs intermédiaires sur l'arbre *multicast* qui ne sont pas considérés comme routeurs de branche-

ment. Un routeur de branchement SEM dans l'arbre de distribution de (S, G) a un seul état de routage pour chaque canal (S, G) ($TRM(S, G)$). SEM simplifie l'algorithme de routage en éliminant aussi la présence des entrées marquées ou des entrées *stale*. À chaque entrée dans l'état de routage est associé un temporisateur, $t1$. À l'échéance de $t1$, l'entrée est détruite.

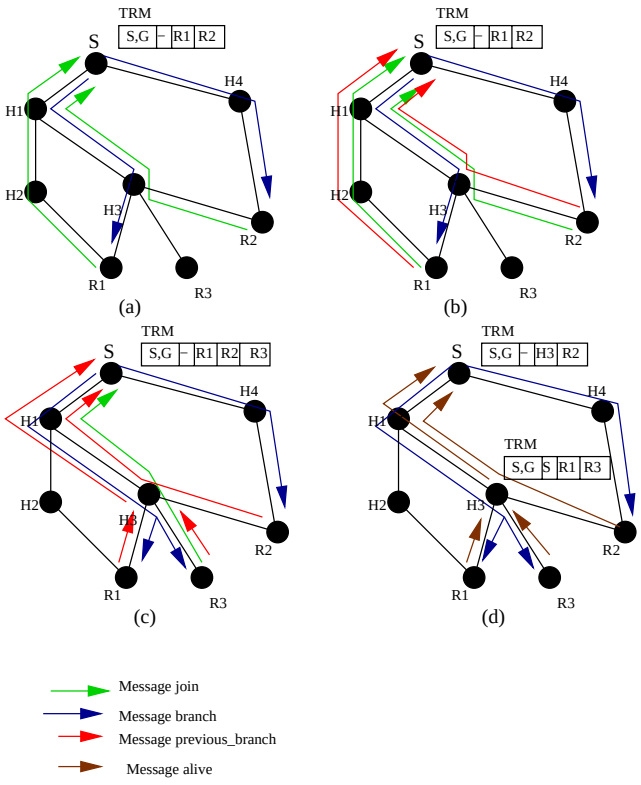


FIG. 3.18 – *Le mécanisme de gestion de l'arbre SEM.*

Considérons l'exemple de la figure 3.18. $R1$ s'abonne au canal (S, G) et S commence à envoyer des messages *branch(S,G,*p_B*=S,R1)*. Comme réponse à ces messages *branch*, $R1$ envoie un message *previous_branch(R1,*p_B*=S,S,G)*. Ces messages créent un état de routage pour (S, G) (contenant $R1$) dans S. Quand $R2$ commence à envoyer un *join(S,G,R2)* en s'abonnant au groupe, ce message n'est pas intercepté et atteint la source[25] qui commence à envoyer un message *branch(S,G,*p_B*=S,R1,R2)*. Par

25. Le premier *join* émis par un destinataire n'est jamais intercepté et arrive toujours à la source.

la suite, des messages *alive* remplacent les messages *join* pour assurer la maintenance de l'arbre.

Le message *branch(S,G,p_B=S,R1,R2)* émis par la source permet la génération d'un message *previous_branch(R2,p_B=S,S,G)* envoyé par $R2$ vers S et créant ainsi l'état pour le canal (S, G) dans S. Cet état contient $R1$, $R2$ (puisque les chemins entre la source S et les deux destinataires $R1$ et $R2$ sont différents) (*cf.* figure 3.18b).

$R3$ à son tour, envoie un *join(S,G,R3)* vers S qui commence à envoyer des *branch(S,G,p_B=S,R1,R2* Le mécanisme GXcast du message *branch* découvre les routeurs de branchement pour le canal (S, G) (S et $H3$ sont deux routeurs de branchement pour l'arbre). S intercepte le message *branch(S,G,p_B=S,R1,R2,R3)* et génère deux messages : *branch(S,G,p_B=S,R1,R3)* et *branch(S,G,p_B=S,R2)*, le premier sur l'interface connectant S à $H1$ et le second sur l'interface connectant S à $H4$. De même, $H3$ génère aussi deux messages *branch* : *branch(S,G,p_B=H3,R1)* et *branch(S,G,p_B=H3,R3)*. Les messages *previous_branch* générés par $R1$, $R2$ et $R3$ comme réponse aux messages *branch* créent finalement des états de routage dans S (contenant $H3$[26] et $R2$ comme entrées) et dans $H3$ (contenant $R1$ et $R3$ comme entrées). Les données adressées par S à $H3$ sont envoyées à $R1$ et $R3$. De plus, les destinataires envoient périodiquement des messages *alive* vers les routeurs de branchement précédents pour maintenir l'arbre. Les messages *alive(S,G,R1)* et *alive(S,G,R3)* envoyés par $R1$ et $R3$ sont interceptés par $H3$ qui envoie à son tour un message *alive(S,G,H3)* vers la source S. La structure finale est celle de la figure 3.18d.

3.4.4 Le message *branch* de type GXcast

Nous avons vu dans le chapitre 2 que nous pouvons encoder au plus 135 adresses IP dans un paquet Xcast et il en est de même pour un message *branch*[27]. Un message *branch* pour un groupe ayant plus que 135 destinataires peut être diviser en plusieurs messages *branch* de type GXcast. À l'arrivée d'un message *branch* à un routeur, un état de contrôle temporaire ($TCM(S, G)$) est créé dans le routeur contenant la liste (L_i)

26. Comme prochain routeur de branchement.
27. $MTU = 576$ octets, 12 octets pour l'en-tête SEM (*cf.* annexe C.1)

des adresses des destinataires dans l'en-tête SEM du message *branch* ($TCM(S,G)$: $L = L_i$) et un temporisateur $t2$ est associé à cet état. À l'arrivée des prochains *branch* pour le même canal (S,G) et si le temporisateur n'a pas expiré, la nouvelle liste des adresses des destinataires dans l'en-tête SEM du message *branch* est ajouté à l'état correspondant au canal (S,G) ($TCM(S,G) : L = L + L_i$) et le temporisateur est armé à nouveau. Cette opération se répète pour chaque message *branch* pour toutes les sous-listes L_i de la liste principale L. Si le temporisateur $t2$ expire le routeur traite la liste L dans l'état de contrôle, classe les destinations en sous-listes L_i selon leur prochain routeur, génère un message *branch* avec l'en-tête SEM appropriée et l'envoie vers chacun des prochains routeurs. La figure 3.19 représente le traitement d'un message *branch* de type GXcast dans un routeur SEM. À l'échéance de $t2$, l'état de contrôle temporaire est détruit[28].

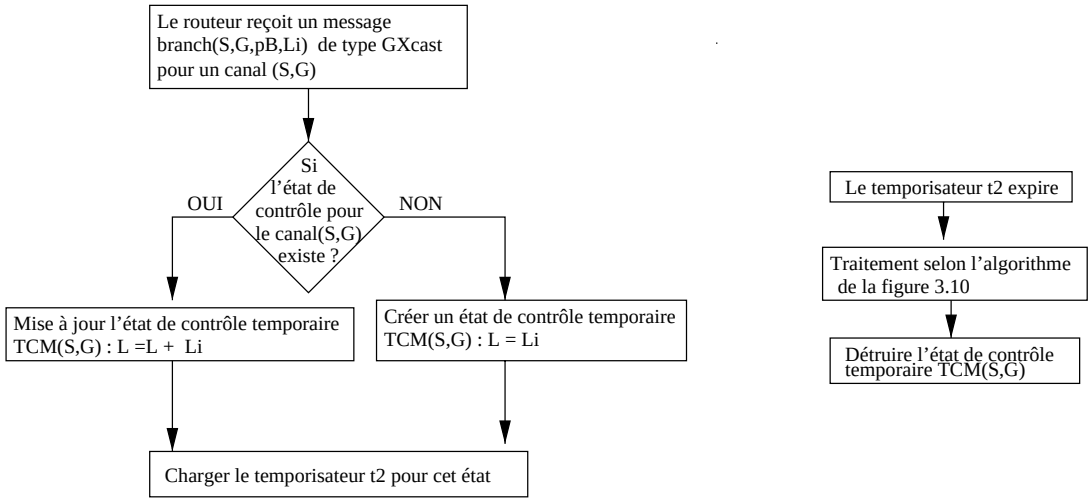


FIG. 3.19 – *Le traitement d'un message branch de type GXcast dans un routeur SEM.*

28. Notons que l'état de contrôle $TCM(S,G)$ pour un canal (S,G) à la source n'est jamais détruit. La source aura besoin toujours de la liste des destinataires.

3.4.5 La transmission des messages *branch* de type GXcast et la rapidité de la construction de l'arbre

Il est clair que le temps de construction de l'arbre dans les trois protocoles REUNITE, HBH et SEM est plus élevé que le temps de construction de l'arbre par un protocole de routage *multicast* traditionnel (Par exemple PIM-SM). Ceci est dû au fait que la construction de l'arbre ne dépend plus avec ces trois protocoles uniquement des messages *join* traditionnels qui construisent l'arbre d'une manière efficace et rapide. En effet, les premiers messages *join* de HBH et SEM doivent atteindre toujours la source. La construction de l'arbre dans REUNITE est réalisée en utilisant des messages *join* et des messages *tree* ensemble. De plus, à ces messages s'ajoutent des messages *fusion* dans le cas de HBH. Ces messages sont identiques aux messages utilisés dans SEM.

SEM élimine toute présence d'états de routage ou de contrôle dans les routeurs intermédiaires qui ne sont pas des routeurs de branchement pour l'arbre. Ceci amène à ce que l'entretien de l'arbre lors d'un départ d'un membre soit plus facile dans HBH.

SEM utilise un mécanisme très efficace pour éviter cet inconvénient. En effet, lors de construction de l'arbre SEM, le protocole SEM utilise la transmission en mode GXcast pour livrer les paquets de données. Une fois l'arbre construit, la transmission en mode SEM est à nouveau utilisée. Si les groupes sont très dynamiques, on tombe dans le cas d'un protocole GXcast pur. Si les groupes sont moyennement dynamiques ou quasi-stables l'arbre construit par le protocole SEM est quasi-stable aussi. À la source d'un arbre SEM, l'algorithme de la figure 3.20 est utilisé.

3.4.6 Une comparaison entre SEM et HBH

Premièrement, selon HBH, une table (MCT ou MFT) existe dans tous les routeurs entre la source et la destination et ces tables sont utilisées pour contrôler ou acheminer les paquets *multicast*. HBH a essayé de réduire la taille des tables de routage MFT dans les routeurs intermédiaires qui ne sont pas de branchement pour l'arbre HBH. Mais comme le démontre la figure 3.17d pour SEM (correspondant à la figure 5 dans

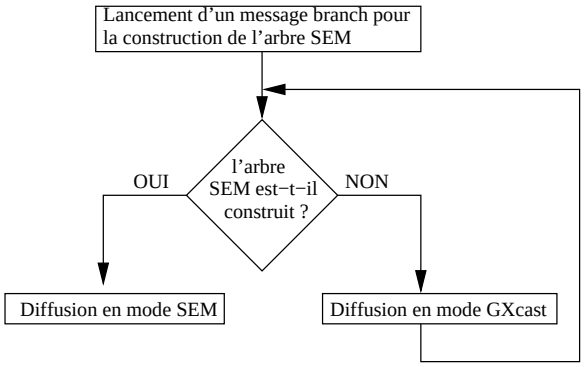


FIG. 3.20 – *L'algorithme de diffusion des paquets en mode GXcast lors de la construction de l'arbre SEM.*

[HCFCD01]), nous remarquons que le routeur $H1$ n'est pas un routeur de branchement pour l'arbre HBH mais contient toujours une table de routage MFT. Dans SEM, contrairement à HBH, on n'a ni table MFT ni table MCT dans les routeurs qui ne sont pas routeurs de branchement.

Prenons l'exemple de la figure 3.21. Supposons les routes *unicast* suivantes : $R1 \rightarrow H2 \rightarrow H1 \rightarrow H0 \rightarrow S$; $S \rightarrow H0 \rightarrow H1 \rightarrow H3 \rightarrow R1$; $R2 \rightarrow H4 \rightarrow H0 \rightarrow S$; $S \rightarrow H0 \rightarrow H1 \rightarrow H3 \rightarrow R2$; $R3 \rightarrow H3 \rightarrow H1 \rightarrow H0 \rightarrow S$; $S \rightarrow H0 \rightarrow H1 \rightarrow H3 \rightarrow R3$;. Ce réseau est asymétrique puisque certains de ses routes sont asymétriques.

Appliquons le mécanisme du protocole HBH. $R1$ s'abonne au canal (S, G) et S commence à envoyer des messages *tree(S,G,R1)*. Ces messages créent une MCT pour (S, G) (contenant $R1$) dans $H0$, $H1$ et $H3$. $R3$ envoie un *join(S,G,R3)* vers S qui commence à envoyer des *tree(S,G,R3)*. En suivant le même raisonnement que celui du paragraphe 3.4.2, la structure de l'arbre à la fin de cette phase est celle de la figure 3.21a.

Supposons maintenant que $R2$ commence à envoyer des *join(S,G,R2)* en s'abonnant au groupe, ces messages ne sont pas interceptés et atteignent la source qui commence à envoyer des *tree(S,G,R2)* (*cf.* figure 3.21b). Dès que $H0$ commence à recevoir deux messages *tree* différents (*tree(S,G,H1)* et *tree(S,G,R2)*), il détruit la MCT et crée une entrée pour (S, G) dans sa MFT, contenant $H1$ et $R2$, et il envoie un message

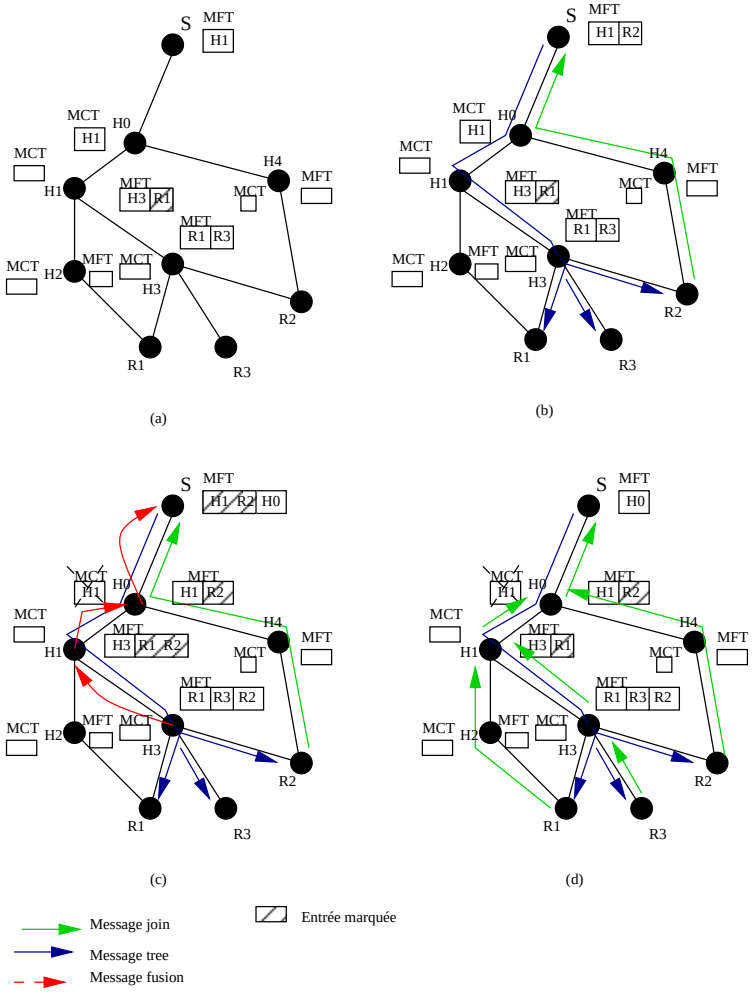


FIG. 3.21 – *La comparaison de la réduction des états de routage entre SEM et HBH.*

fusion(S,G,H1,R2) vers la source. La réception du ***fusion*** par S entraîne l'ajout de $H0$ dans la table MFT de S, et le marquage de $H1$ et $R2$. De la même façon que $H0$, $H1$ reçoit le message ***tree(S,G,R2)*** et envoie par conséquent un ***fusion(S,G,H3,R1,R2)*** vers $H0$. La réception du ***fusion*** par $H0$ entraîne l'ajout de $H1$ dans sa table MFT, et le marquage de $R2$. $H3$ reçoit des messages ***tree(S,G,R2)*** et envoie par conséquent un ***fusion(S,G,R1,R3,R2)*** vers $H1$. La réception du ***fusion*** par $H1$ entraîne le rafraîchissement de $H3$ dans la table MFT, et le marquage de $R2$ (*cf.* figure 3.21c). La structure finale de l'arbre est celle de la figure 3.21d.

Nous déduisons que si le réseau est asymétrique (ce qui le cas dans Internet), la réduction d'états de routage avec HBH n'est pas suffisante. Rappelons que lorsque le nombre de groupes augmente dans le réseau, le nombre d'états de routage croît aussi. Il suffit dans notre exemple que les destinataires $R1$, $R2$ et $R3$ appartiennent à n groupes *multicast* différents pour avoir n états de routage sur chaque routeur appartenant à l'arbre *multicast*.

Deuxièmement, afin de construire et maintenir l'arbre *multicast* HBH, un message *tree* est envoyé. Le mode de diffusion du message n'est pas détaillé dans la proposition HBH. Nous envisageons trois modes de diffusion : *multicast, unicast* et *unicast* récursif. Il semble que ces modes ne soient pas adaptés.

HBH n'utilise pas le routage *multicast* traditionnel. Aucun état de routage *multicast* traditionnel n'existe dans les routeurs intermédiaires et donc un message *tree* ne peut pas être envoyé en mode *multicast*. En plus, un message *tree* doit suivre le plus court chemin *unicast* entre la source et le destinataire et non pas un chemin RPF entre la source et le destinataire comme celui emprunté par le message *join*.

Un message *tree* peut être envoyé en mode *unicast* récursif si l'arbre est déjà construit. Donc il y a un message qui atteint tous les destinataires, mais dont l'adresse de destination est modifiée au cours de sa progression dans l'arbre, selon l'*unicast* récursif. Le problème se pose alors lors de la construction de l'arbre. En effet, le premier message *join* est envoyé directement à la source et ne laisse aucune information dans les routeurs intermédiaires concernant le destinataire qui a envoyé le message

join. Donc il n'y a aucune information concernant les destinataires dans les routeurs intermédiaires qui peut être utilisée par le message *tree* pour l'acheminement en mode *unicast* récursif. En plus, il est indiqué dans [HCFCD01] (figure 3.16 et 3.17) qu'un état de contrôle MCT n'est transformé en état de routage MFT qu'à la réception d'un nouveau message *tree* pour un nouveau destinataire.

Prenons l'exemple de la figure 3.22. $R1$ s'abonne au canal (S, G) et S commence à envoyer des messages *tree(S,G,R1)*. Ces messages créent une MCT pour (S, G) (contenant $R1$) dans $H1$ et $H3$. $R3$ à son tour, envoie un *join(S,G,R3)* vers S qui commence à envoyer des *tree(S,G,R3)*. Il n'y a aucun moyen pour S pour envoyer des messages *tree* en *unicast* récursif. En effet, dans le cas de l'*unicast* récursif, S va envoyer un message *tree* à $R1$ et ensuite dans $H3$ il y aura une duplication du message *tree* : le message initial pour $R1$ et une copie pour $R3$. Mais puisque il n'y a aucun état dans $H3$ lors de la construction de l'arbre, l'*unicast* récursif ne peut pas être utilisé dans ce cas.

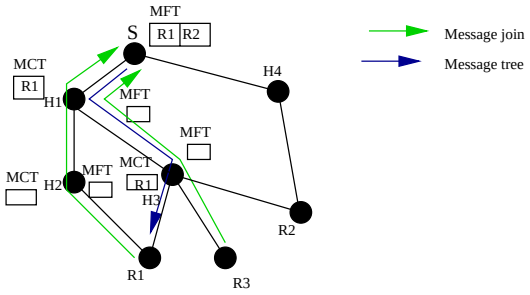


FIG. 3.22 – *Le message tree en mode unicast récursif.*

Donc il nous semble que le message *tree* décrit dans HBH ne peut pas être qu'*unicast* et un message *tree* est envoyé pour chaque destinataire. Une fois l'arbre est stable, un message *tree* en mode *unicast* récursif peut être envoyé[29].

Il ne reste donc que le mode *unicast*. Pour chaque destinataire qui s'abonne au canal (S, G), un message *tree* est envoyé de la source vers ce destinataire. Ceci nous

29. En fait, il n'y a pas grande différence entre un message *unicast* récursif et *unicast* quand l'arbre est stable, puisque la destination du message *tree* est le prochain routeur de branchement sur l'arbre *multicast* déjà construit.

paraît le mieux adapté avec la description faite dans [HCFCD01] (*cf.* figure 3.16). Le nombre de messages *tree* envoyés périodiquement dans HBH est donc proportionnel au nombre de destinataires.

Par conséquent, en envoyant le message *tree* afin de construire l'arbre HBH, ce message *tree* passant par un routeur produit toujours de nouveaux messages *tree* pour toutes les entrées de la table MFT. La figure 3.23 représente un arbre HBH en phase de construction. Les deux routeurs $R1$, $R2$ envoient leurs messages *join* à la source. Suite au message *join* de $R1$, la source envoie un message *tree* vers $R1$ qui crée un table MCT dans chaque routeur entre S et $R1$. Suite au message *join* de $R2$, la source envoie un message *tree* vers $R2$. Ce message *tree* détruit la table MCT dans le prochain routeur ($H1$) et crée une table MFT à sa place avec $R1$ et $R2$ comme entrées de cette table. Un message *tree* est généré ensuite pour chaque entrée dans la nouvelle table MFT (soit $R1$ et $R2$). Cette opération est répétée pour chaque routeur entre la source et les destinations. Nous déduisons que pendant la phase de construction de l'arbre et avant que les entrées marquées aient expiré, 5 messages *tree* seront produits pour le routeur $R1$ et un seul message *tree* sera produit pour le routeur $R2$. Chaque nouveau message *join* pour une nouvelle destination aura le même effet sur l'arbre jusqu'au moment où les entrées marquées expieront. Une fois l'arbre construit, les entrées marquées vont expirer et HBH résout le problème automatiquement puisque les entrées de prochains routeurs de branchement sont toujours *stale* (entrées qui permettent l'acheminement des paquets de données mais ne produisent aucun message *tree*). C'est vrai seulement si les messages *tree* ne sont pas envoyés immédiatement après les messages *join* périodiques qui rafraîchissent les entrées *stale*. Sinon le problème d'inondation persiste. HBH peut limiter l'effet de ce problème en utilisant un temporisateur comme celui utilisé dans 3.4.4 pour que les messages *branch* de type GXcast.

Pour construire l'arbre *multicast*, nous considérons dans notre proposition du protocole SEM que l'utilisation d'un message *branch* de type GXcast (qui a un rôle équivalent aux messages *tree* de HBH) est le mieux adapté pour la construction de l'arbre *multicast*.

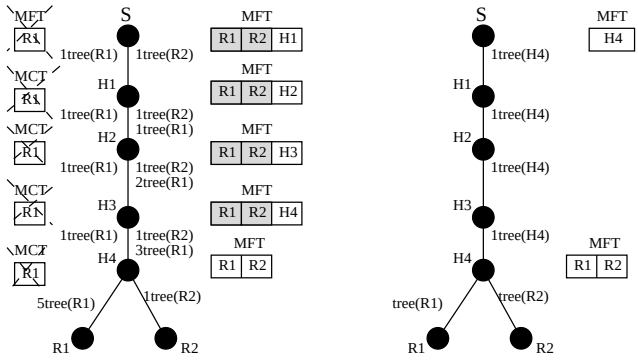


FIG. 3.23 – *Le message tree en mode unicast.*

Comme conclusion, nous pouvons déduire que la construction de l'arbre est plus simple dans SEM que dans HBH. La présence de MCT et de MFT dans les routeurs, le traitement des messages *tree* et *fusion* et le grand nombre de ces messages pendant la phase de construction de l'arbre ajoutent une certaine complexité au protocole HBH.

3.5 Évaluation et simulation

Notre protocole est évalué en terme de résistance au facteur d'échelle (taille des tables de routage *multicast*, surcoût dû aux messages de contrôle) et d'efficacité (coût de l'arbre *multicast*, temps de traitement des entêtes *multicast* dans les routeurs).

3.5.1 Scénario de simulation

Nous simulons SEM dans NS (simulateur de réseau) [FV01] pour valider le comportement du protocole et son efficacité pour la réduction du nombre d'états de routage et pour la construction de l'arbre. La performance de SEM est comparée à celles de PIM, GXcast et HBH. PIM dans nos simulations se réfère à la simulation dans NS de PIM-SM qui construit exclusivement des arbres source-spécifique. En plus de SEM, nous simulons quelques mécanismes du protocole HBH selon [HCFCD01] et nous utilisons le simulateur du protocole GXcast. Nous présentons deux modèles de simulation

générés à l'aide du générateur de scénarios de simulation GT-ITM [ZCB96] : chacun avec un graphe de 100 nœuds dont tous les liens sont bidirectionnels avec une bande passante de 20 Mbit/s. La topologie du premier modèle (considérée comme un réseau dense) est basée sur le premier algorithme de Waxman [Wax88] avec 0.3 comme degré de distribution des nœuds. La topologie du deuxième modèle (considérée comme un réseau clairsemé) est basée sur l'algorithme aléatoire pur et divisée en 5 domaines. Quatre domaines contiennent à la fois des destinations et des sources, alors que le cinquième domaine est considéré comme domaine du cœur. Le nombre P (le pourcentage de sources parmi les nœuds du réseau) et le nombre N_{DR} (nombre de DR destinataires pour chaque source) sont aléatoirement déployés dans le réseau[30]. Une destination adhère aléatoirement à l'arbre et il n'y a aucun désabonnement explicite (message *leave*)[31]. Le tableau 3.1 récapitule les paramètres utilisés dans la simulation.

N	100	Nombre de nœuds dans le réseau
P	10, 20, 30, 40, 50, 60	Pourcentage de sources parmi les nœuds du réseau (nombre d'arbres)
N_{DR}	3, 6, 9, 12, 15, 18	Nombre de DR destinataires pour chaque source

TAB. 3.1 – *Les paramètres de simulation du portocole SEM*

3.5.2 La diminution en taille des tables de routage *multicast*

La taille des tables de routage dans tous les routeurs du réseau du premier modèle de la topologie (mode dense et algorithme de Waxman) est montré dans la figure 3.24 et celui du deuxième modèle de la topologie (mode clairsemé et algorithme aléatoire pur) est montré dans la figure 3.25.

L'axe horizontal est le pourcentage des sources (parmi les nœuds) actives dans le réseau, et l'axe vertical est la taille globale des tables de routage dans le réseau. Les

30. Le nombre de destinataires derrière les DR (dans leur sous-réseaux) peut être beaucoup plus élevé.
31. Notre but est d'obtenir un maximum d'états de routage dans les routeurs pour bien montrer la réduction obtenue par SEM par rapport à PIM.

poly-lignes marquées PIM-x et SEM-x montrent la taille globale des tables de routage pour les protocoles PIM et SEM respectivement quand le nombre de destinations par groupe est x.

La taille de la table de routage s'accroît avec le nombre de groupes actifs et le nombre de destinations, comme prévu dans le sous-chapitre 3.2. Nous déduisons de la figure 3.24 et la figure 3.25 que la réduction du nombre d'états de routage dans SEM est approximativement 40% pour le mode dense et 80% pour le mode clairsemé respectivement en comparaison avec PIM et ce quelque soit le pourcentage de sources. Ceci est un résultat attendu puisque dans un réseau dense le nombre de routeurs de branchement est plus élevé que celui dans un réseau clairsemé. Nous concluons que notre protocole est plus approprié aux réseaux clairsemés.

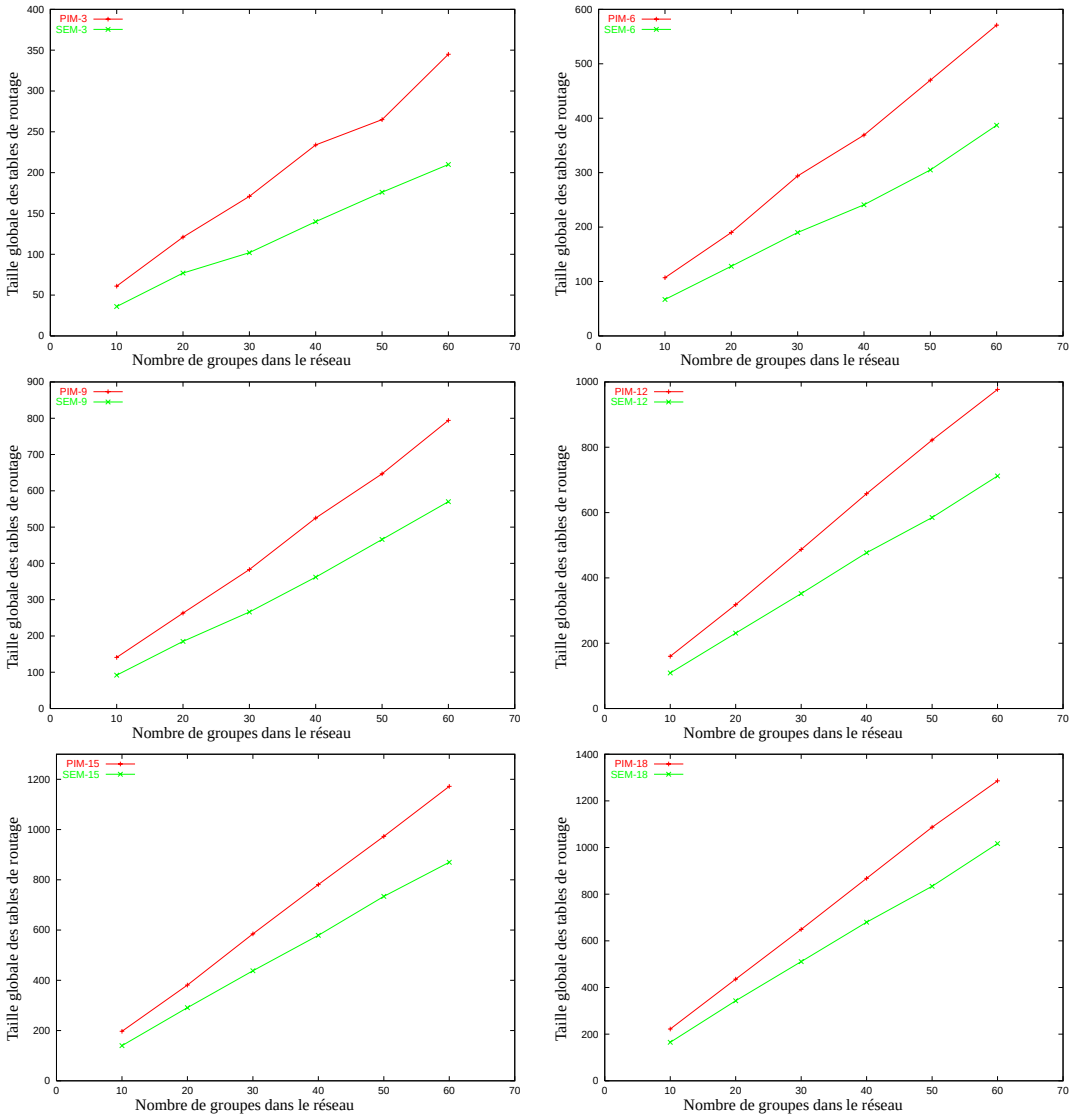


FIG. 3.24 – *La taille globale des tables de routage en fonction du nombre de groupes dans le réseau - mode dense et algorithme de Waxman.*

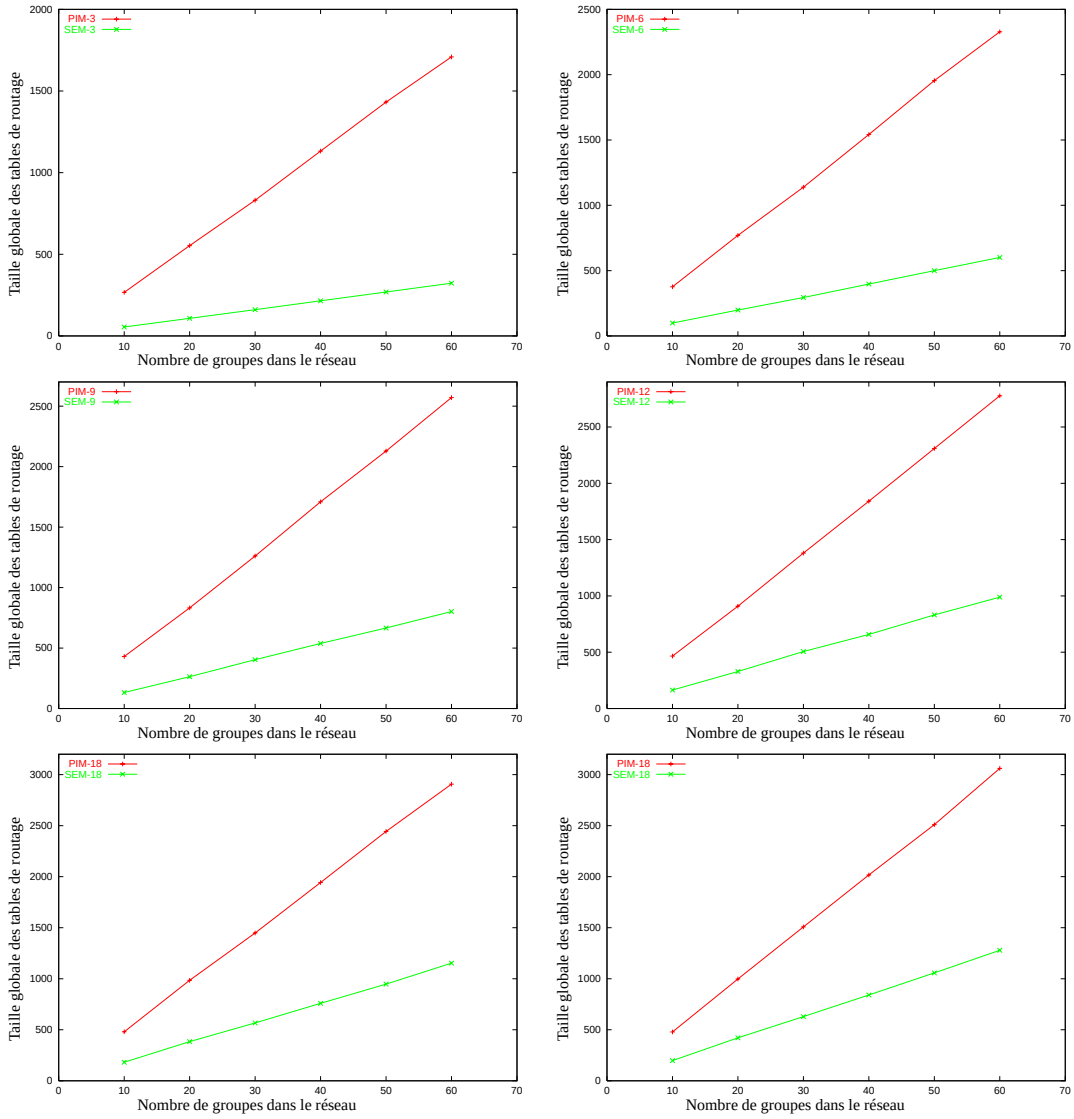


FIG. 3.25 – *La taille globale des tables de routage en fonction du nombre de groupes dans le réseau - mode clairsemé et algorithme aléatoire pur.*

La réduction par rapport au protocole HBH Nous avons présenté dans le sous-chapitre 3.4.6 que le protocole SEM réduit mieux que le protocole HBH le nombre d'états de routage dans un réseau asymétrique. Pour les simulations nous avons utilisé la topologie proposée pour HBH dans [HCFCD01]. Cette topologie 3.26 est typique d'un grand réseau d'ISP (MCI) [AGKT98].

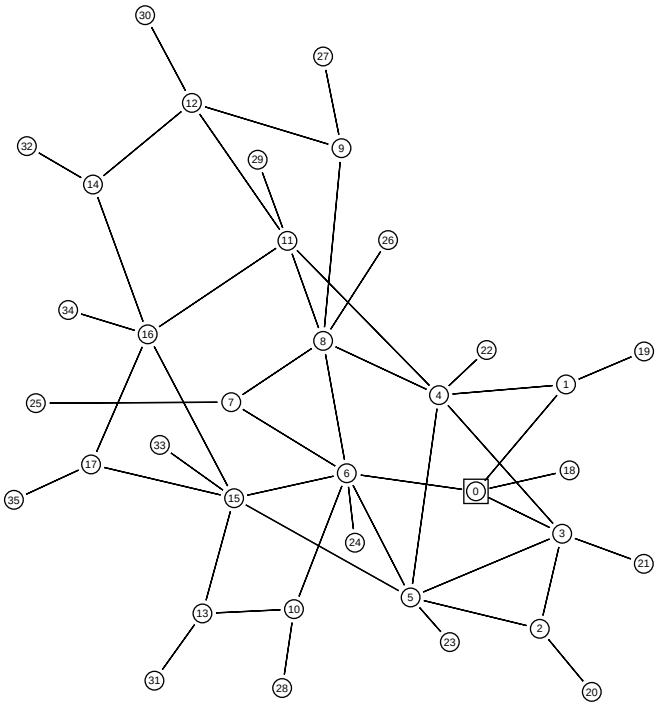


FIG. 3.26 – *Le réseau MCI.*

Un seul destinataire est connecté à chaque nœud de la topologie. La présence de un ou plusieurs destinataires attachés à un même nœud du réseau ne change pas le coût de l'arbre *multicast*. Les nœuds de 0 à 17 sont des routeurs (du cœur du réseau) tandis que les nœuds de 18 à 35 sont des membres potentiels du canal *multicast*. Au lien $< n1, n2 >$ qui connecte les nœuds $n1$ et $n2$ sont associés deux coûts, $n1$-$n2$ et $n2$-$n1$ choisis aléatoirement dans l'intervalle $[1, 10]$. Les simulations considèrent un groupe *multicast* de 1 vers N. Le nœud 18 est fixé comme source. Un nombre variable de destinataires est choisi aléatoirement parmi les nœuds 19 à 35. Pour chaque nombre de destinataires, nous avons réalisé, comme dans HBH, 500 simulations par algorithme.

La figure 3.27 présente le nombre moyen d'états de routage pour un canal dans le réseau pour SEM et HBH tandis que la figure 3.28 présente le surcoût du protocole HBH par rapport au protocole SEM en terme de ce nombre moyen d'états de routage[32]. Nous remarquons (HBH % SEM) que le protocole SEM réduit d'au moins 34% ce nombre moyen d'états de routage. Cette réduction varie selon le nombre de destinataires et peut atteindre environ 100%[33] pour les canaux ayant 2 destinataires.

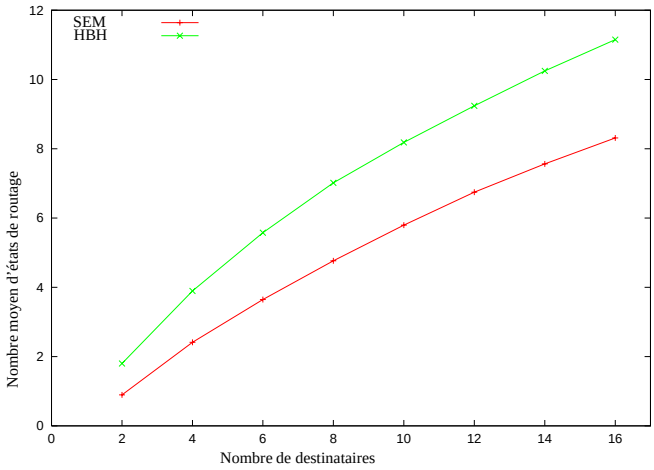


FIG. 3.27 – *Le nombre moyen d'états de routage pour SEM et HBH.*

Le tableau 3.2 présente le nombre total d'entrées dans les tables de routage pour tous les routeurs intermédiaires entre la source et les destinataires dans le réseau. On considère 4000 canaux dans le réseau et on compare le cas du protocole SEM par rapport au protocole HBH. Nous remarquons que le protocole SEM utilise des tables de routage ayant un plus petit nombre d'entrées que celles utilisées avec le protocole HBH. Cette diminution du nombre total d'entrées dans les tables de routage devient de plus en plus importante si le nombre de groupes actifs dans le réseau augmente.

32. Nous ne considérons que les états de routage présents dans les nœuds intermédiaires des arbres (essentiellement des nœuds du cœur de réseau : ni source, ni destinataire).

33. Nous avons déjà vu qu'il y 0 réduction d'états de routage dans HBH par rapport à un protocole de routage *multicast* pour un canal comme celui du réseau présenté dans la figure 3.21.

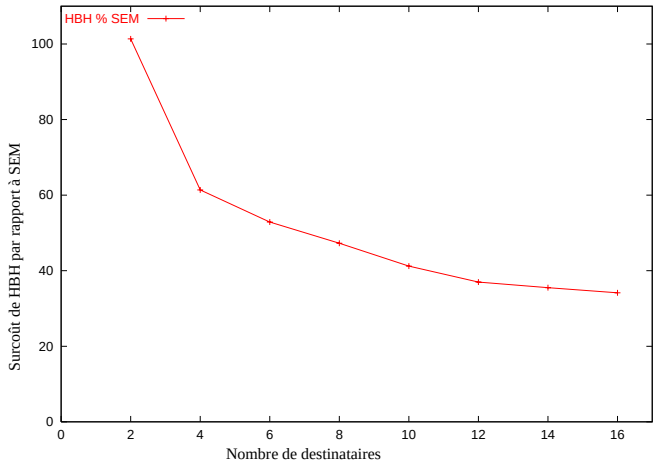


FIG. 3.28 – *Le surcoût de HBH par rapport à SEM en terme de nombre moyen d'états de routage pour un canal.*

Protocole	Le nombre total d'entrées dans les tables de routage pour tous les routeurs de branchement du réseau pour 4000 canaux
SEM	20054
HBH	28529
Surcoût	8475

TAB. 3.2 – *La réduction globale des entrées dans les tables de routage de SEM par rapport à HBH dans le réseau.*

3.5.3 Le coût de l'arbre et le surcoût dû aux messages de contrôle

Les paquets SEM suivent l'arbre des plus courts chemins entre la source et les destinataires. Ceci constitue un avantage sur les protocoles de routage *multicast* traditionnel comme PIM-SM (avec un arbre partagé et un routeur de rendez-vous) et CBT qui envoient les paquets *multicast* vers un point de rendez-vous qui à son tour les renvoient aux destinataires. Donc le coût de la transmission de données dans l'arbre *multicast* construit par SEM est inférieur à celui des protocoles *multicast* traditionnel.

Dans HBH, les messages *join* périodiques, les messages *tree* et les messages *fusion* générés durant la phase de construction de l'arbre provoquent un surcoût important. Le surcoût dû aux messages de contrôle de SEM peut être mesuré en utilisant le nombre

total de paquets de contrôle envoyés par lien ou bien le pourcentage de la bande passante utilisée par ces messages de contrôle. Rappelons que SEM utilise des messages *branch*, des messages *previous_branch* pour construire l'arbre et des messages *alive* périodiques pour assurer l'entretien de cet arbre.

Dans SEM, le premier message *join* atteint la source. Entre deux routeurs de branchement il y a des messages *alive* périodiques. Dans PIM, il y a aussi des messages *join* périodiques entre deux routeurs sur l'arbre *multicast*. Si la même période de rafraîchissement est choisie, le nombre de paquets de contrôle est presque le même dans PIM et dans SEM, s'il n'y a de changement de membres du groupe *multicast* (l'absence d'un nouveau message *join* ou *leave*).

Le tableau 3.3 récapitule les messages de contrôle envoyés par les trois protocoles : PIM, HBH et SEM[34]. Le surcoût dans SEM par rapport à PIM résulte des messages *join* envoyés directement à la source au moment d'adhésion d'un nouveau destinataire et des messages *branch* envoyés pour construire l'arbre. Dans le cas des groupes moyennement statiques le surcoût dû à ces messages n'est pas important. En revanche, avec le protocole HBH le nombre de paquets de contrôle croît avec le temps puisque les messages *tree* de HBH sont envoyés périodiquement vers tous les destinataires du groupe.

La figure 3.29 présente le nombre global de paquets de contrôle *branch* et *tree* pour SEM et HBH[35] pour la topologie MCI représentée sur la figure 3.26. Les poly-lignes marquées $tree$-src, $branch$-src montrent le nombre des messages *tree* et *branch* générés à la source tandis que les poly-lignes marquées $tree$-$core$, $branch$-$core$ montrent le nombre de messages *tree* et *branch* qui traversent le réseau cœur de cette topologie. Nous déduisons de la figure 3.29 que le nombre de messages de contrôle pour l'entretien de l'arbre dans HBH est beaucoup plus élevé que le nombre dans SEM, ce qui représente un avantage pour SEM par rapport à HBH.

34. Pour simplifier, nous supposons que dans cette topologie les liens du réseau sont symétriques.

35. Nous considérons que les arbres sont stables et nous ne tenons pas compte de la duplication des messages *tree* présentée dans le sous-chapitre 3.4.6. De plus, nous ne considérons pas les messages *tree* périodiques de HBH.

Protocole	Messages identiques	Messages additionnels par rapport à PIM
PIM	1 message *join multicast* 1 message *join* périodique	
SEM	1 message *previous_branch* 1 message *alive* périodique	1 message *join* vers la source 1 message *branch*
HBH	1 message *join* périodique 1 message *fusion*	1 message *join* vers la source 1 message *tree* vers tous les destinataires 1 message *tree* périodique

TAB. 3.3 – *Les messages de contrôle envoyés par les trois protocoles PIM, SEM et HBH.*

3.5.4 La comparaison entre le protocole SEM et le protocole GXcast

Lors de construction de l'arbre SEM, le protocole SEM utilise la transmission en mode GXcast pour livrer les paquets. Une fois l'arbre construit, les mécanismes du protocole SEM sont à nouveau utilisés. Si les groupes sont très dynamiques, on tombe dans le cas du protocole GXcast. Si les groupes sont moyennement dynamiques l'arbre construit par le protocole SEM est quasi-stable aussi.

Nous avons fait une comparaison entre plusieurs variantes du protocole SEM et le protocole GXcast. En effet, nous avons fait varier le taux d'utilisation du protocole GXcast au sein du protocole SEM. Si l'arbre n'est pas stable (groupes très dynamiques) le volume transmis dans le réseau s'approche du volume transmis par le protocole GXcast. Si l'arbre est stable (groupes statiques) le volume de données s'approche d'un protocole de routage *multicast* traditionnel. Comme nous l'avons vu dans le chapitre 2, le surcoût du protocole GXcast provient de la taille des paquets.

Nous prenons la même topologie du réseau Abilene et nous choisissons comme paramètres les 3 valeurs suivantes : 90, 140 et 190 destinataires dans un groupe. Ainsi,

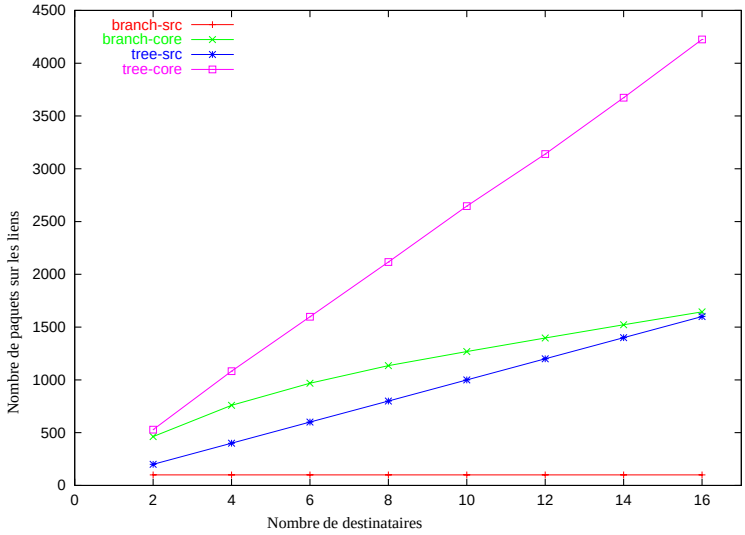


FIG. 3.29 – *Le surcoût de HBH par rapport à SEM en terme de paquets de contrôle.*

les figures 3.30 et 3.31 représentent respectivement le nombre de paquets et la quantité de données transmises sur tous les liens du réseau de la source. Les figures 3.32 et 3.33 représentent respectivement le nombre de paquets et la quantité de données transmises sur tous les liens du réseau cœur et les figures 3.34 et 3.35 représentent respectivement le nombre de paquets et la quantité de données transmises sur tous les liens des réseaux des destinataires. L'axe horizontal représente le nombre de paquets ou la quantité de données transmises et l'axe vertical représente le nombre de destinataires (n prend les 3 valeurs suivantes : 90, 140 et 190) appartenant à un groupe. Les poly-lignes marquées P-x, représentent les valeurs obtenues en utilisant la transmission de paquets avec x% du trafic en mode SEM et $(100 - x)$% du trafic en mode GXcast. D'après les figures, nous déduisons qu'utiliser GXcast pendant la phase de construction de l'arbre n'introduit pas un surcoût important par rapport à SEM si le pourcentage d'utilisation du mode GXcast est faible (P-100 et P-95 par exemple). Par contre, ce surcoût devient important si le pourcentage d'utilisation du mode GXcast est élevé (P-40 et P-0). Ceci est un résultat normal et attendu vue la nature du protocole GXcast. En revanche, l'utilisation du mode GXcast est un avantage pour le protocole SEM lui permettant de réduire le problème de la non réception des paquets de données par les destinataires

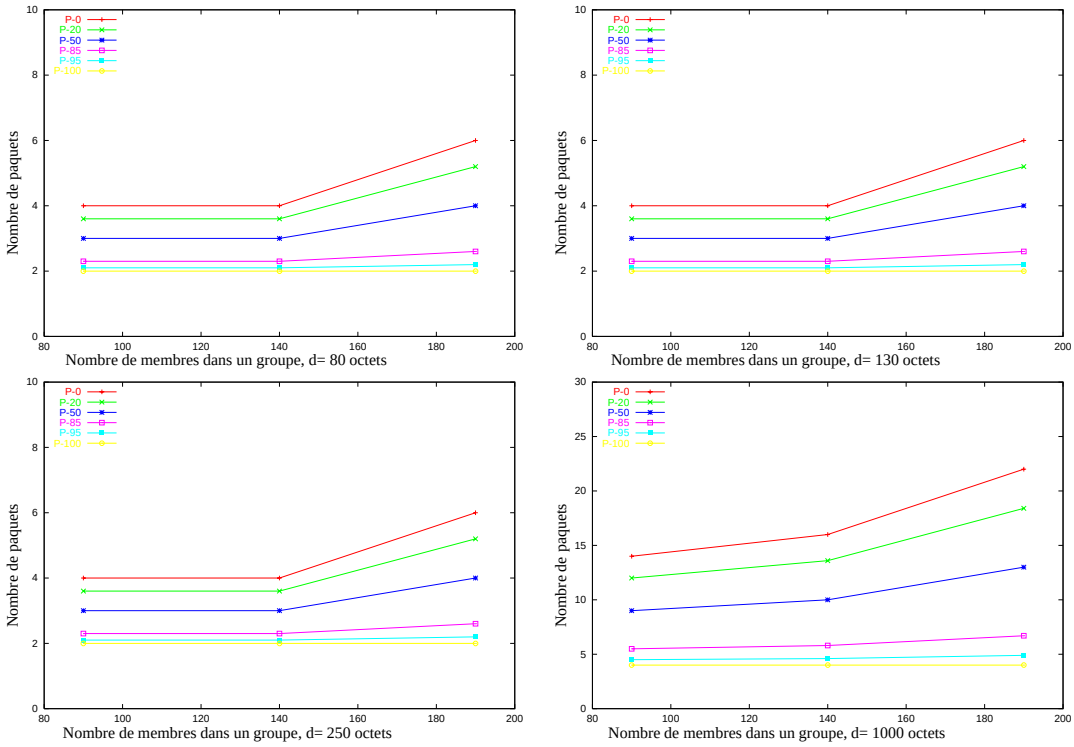


FIG. 3.30 – *Le nombre de paquets dans le réseau de la source avec les protocoles GXcast et SEM.*

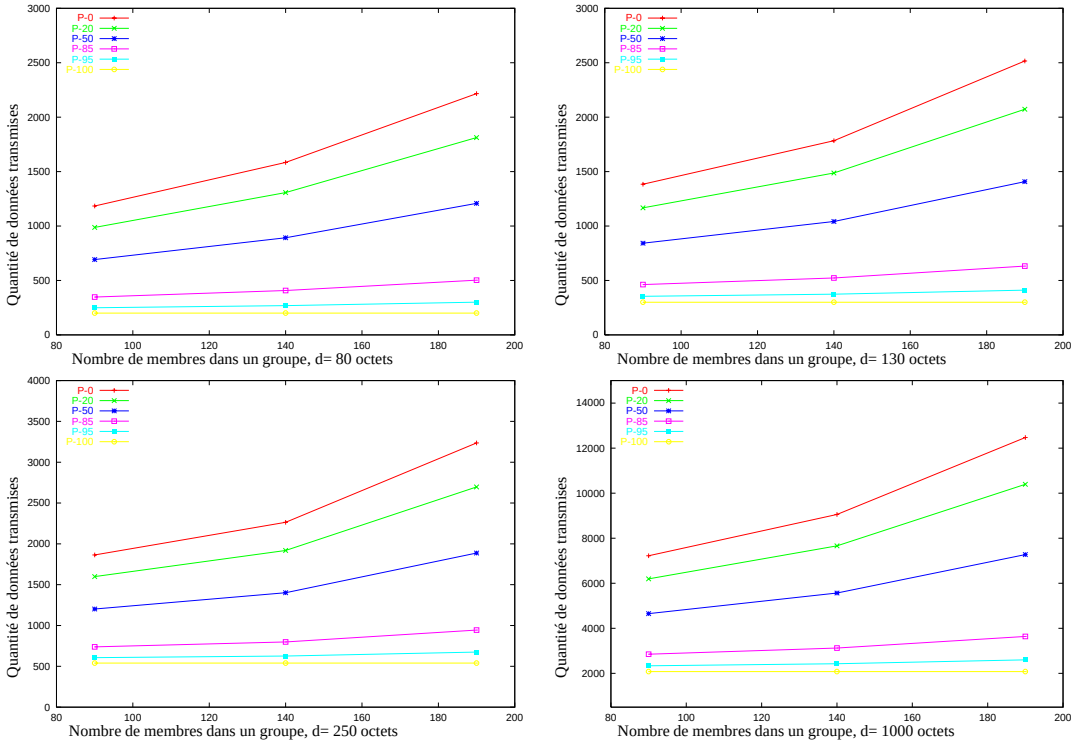


FIG. 3.31 – *Le volume transmis dans le réseau de la source avec les protocoles GXcast et SEM.*

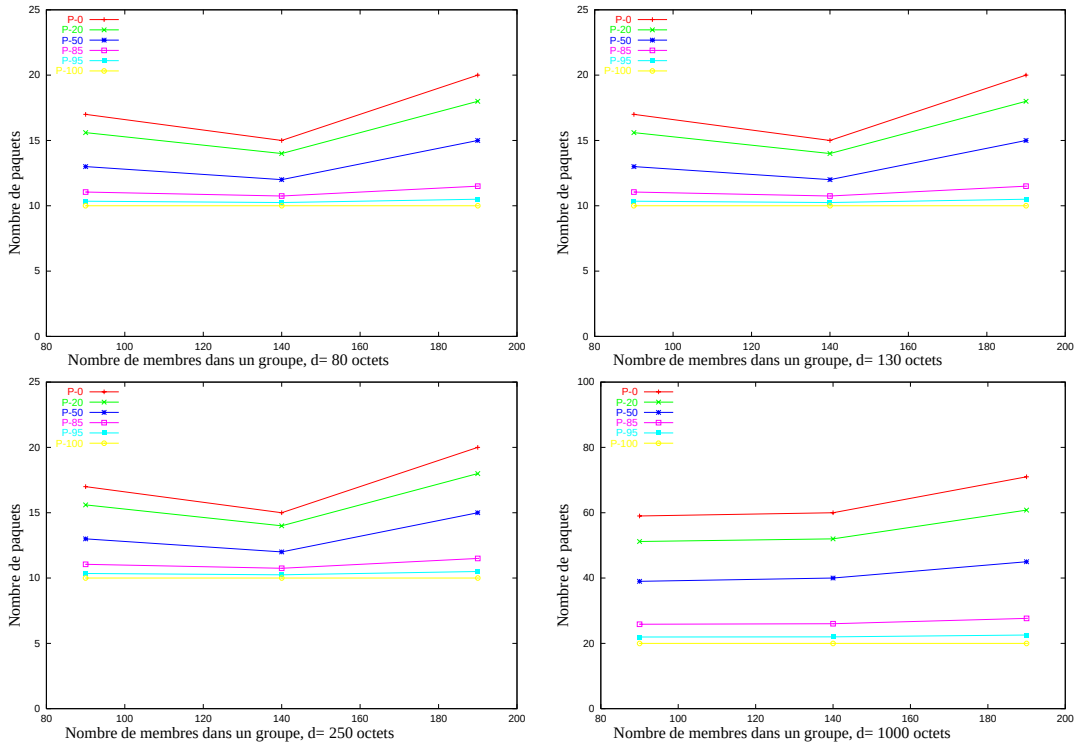


FIG. 3.32 – *Le nombre de paquets dans le réseau cœur avec les protocoles GXcast et SEM.*

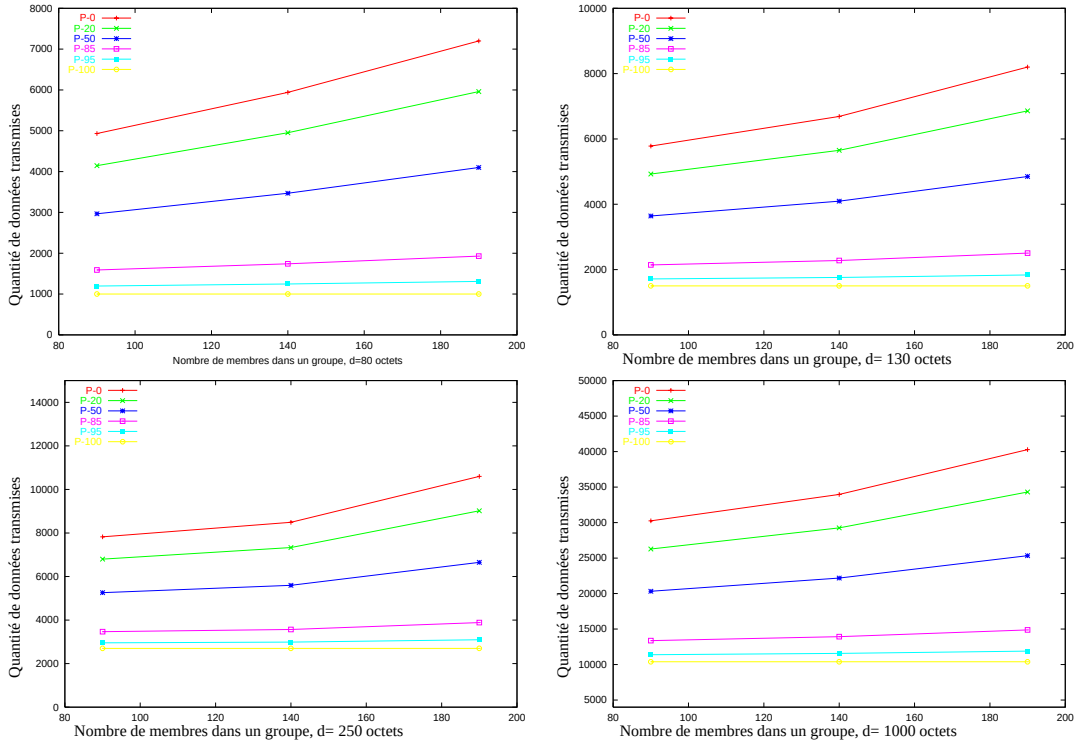


FIG. 3.33 – *Le volume transmis dans le réseau cœur avec les protocoles GXcast et SEM.*

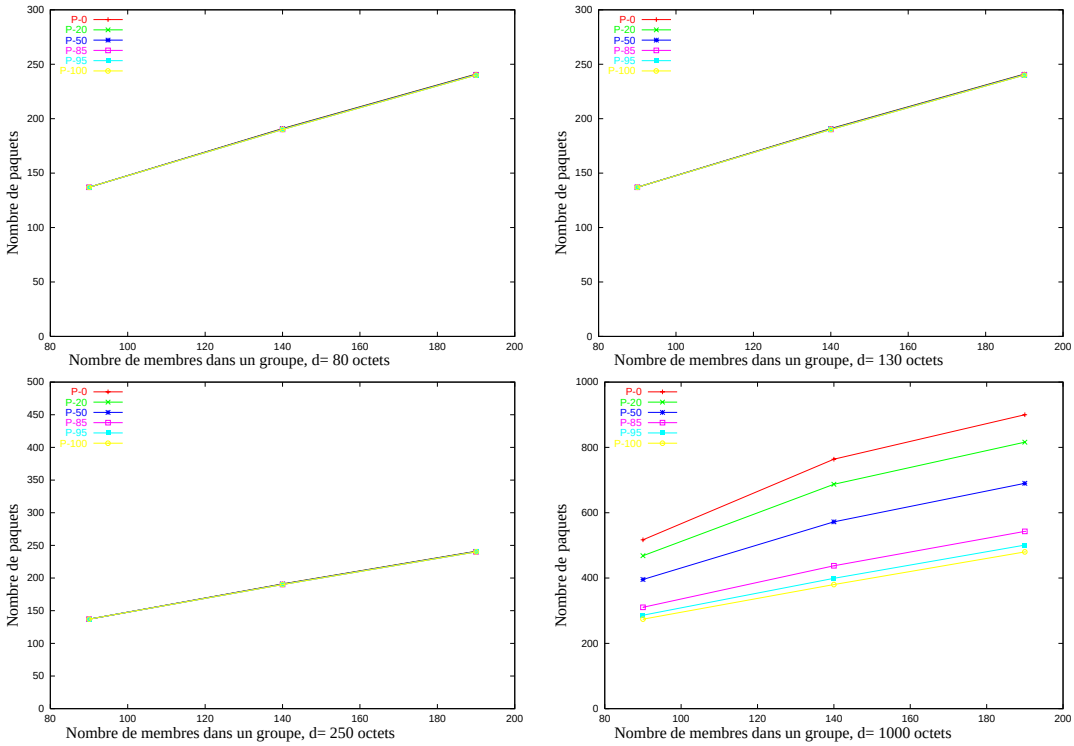


FIG. 3.34 – *Le nombre de paquets dans les réseaux des destinataires avec les protocoles GXcast et SEM.*

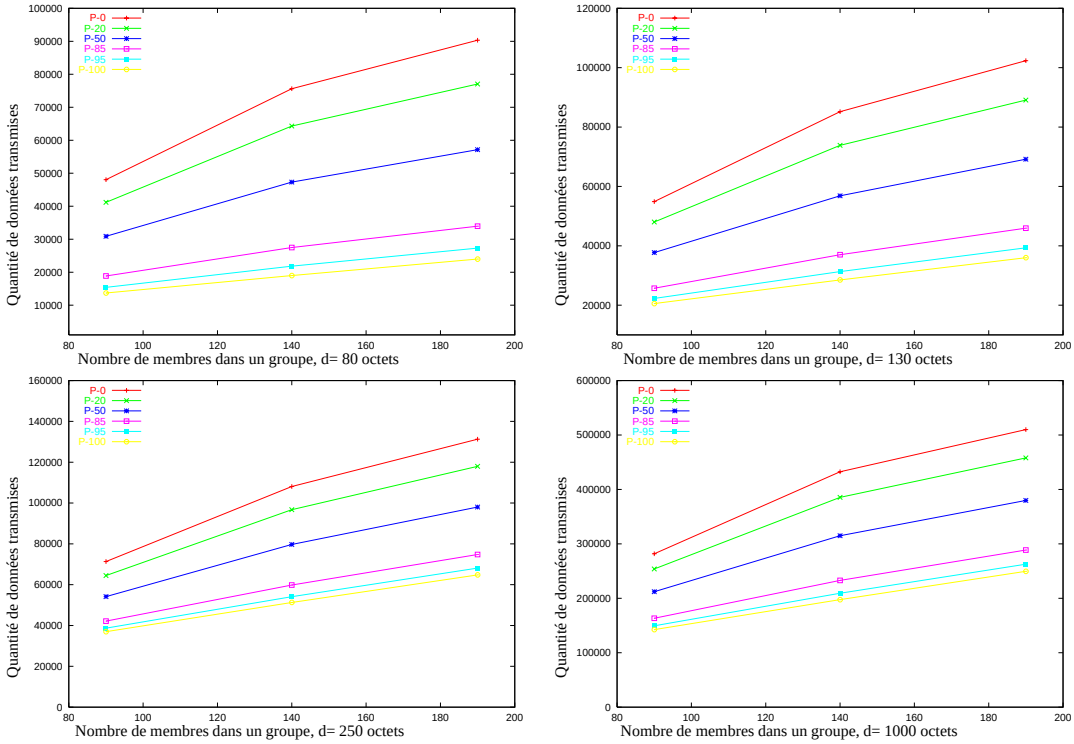


FIG. 3.35 – *Le volume transmis dans les réseaux des destinataires avec les protocoles GXcast et SEM.*

pendant la phase de construction de l'arbre. Ce problème est crucial dans le cas des groupes dynamiques.

3.5.5 Le temps de traitement et le délai

Le temps de traitement de l'en-tête GXcast dans chaque routeur croît avec la taille du groupe *multicast*. Les routeurs GXcast traitent tous les paquets GXcast de la même façon. Dans SEM, seuls les messages *branch* ont besoin d'un traitement supplémentaire qui dépend de la taille du groupe *multicast*. En comparaison avec GXcast, le temps de traitement global des paquets et par la suite le délai sont réduits au minimum dans SEM. SEM permet un plus grand nombre de membres et consomme moins de ressources que GXcast.

3.6 Conclusion

Dans ce chapitre, nous avons proposé le protocole SEM (*Simple Explicit Multicast*), un protocole de routage *multicast* qui utilise une méthode efficace pour construire les arbres *multicast* et pour acheminer les paquets *multicast*. Ce protocole est basé sur les nœuds de branchement. Afin de construire l'arbre *multicast*, la source encode la liste des adresses IP des destinations dans un message *branch* de type GXcast. Ce message a pour rôle de découvrir les routeurs de branchement de l'arbre *multicast*. Ainsi, seuls, les routeurs de branchement de l'arbre mémorisent les états de routage pour un canal *multicast*. Pour acheminer les paquets *multicast*, SEM utilise les arbres *unicast* récursifs, à l'origine proposé dans REUNITE. Les paquets sont acheminés d'un routeur de branchement à un autre suivant l'arbre construit par le message *branch*.

Le protocole SEM est original. En effet, pour simplifier l'allocation d'une adresse *multicast*, SEM utilise la notion de canal source-spécifique (S, G) où S est l'adresse *unicast* de la source et G est une adresse *multicast* standard. SEM réduit aussi le nombre d'états de routage dans les routeurs et construit un arbre des plus courts chemins, et pas un arbre partagé comme beaucoup de protocoles *multicast* traditionnels.

SEM peut aussi utiliser la transmission de paquets de données en mode GXcast pendant la phase de construction de l'arbre ce qui lui permet de réduire le problème de la non réception des paquets de données par les destinataires pendant cette phase de construction de l'arbre

Nous avons décrit les mécanismes du protocole SEM, les algorithmes utilisés pour la construction de l'arbre ainsi que la structure des différents messages. Nous avons comparé par la suite, le protocole SEM au protocole HBH. Finalement nous avons évalué notre protocole avec des simulations en terme de : diminution en taille des tables de routage par rapport à un protocole de routage *multicast* traditionnel et par rapport à HBH, surcoût dû aux messages de contrôle par rapport à HBH, coût de l'arbre et temps de traitement d'un paquet dans un routeur en comparaison avec le protocole GXcast.

Nos travaux montrent que le protocole SEM présente beaucoup d'avantages par rapport au protocole HBH en ce qui concerne le nombre de messages de contrôle pendant la phase de construction de l'arbre et la réduction des états de routage dans les routeurs qui ne sont pas de branchement. SEM réduit aussi le temps de traitement d'un paquet dans un routeur en comparaison avec le protocole GXcast.

Chapitre 4

Le protocole MMT

Avec l'augmentation du nombre d'utilisateurs et l'apparition de nouvelles applications multimédia, le modèle de service *best-effort* d'Internet rencontre de sérieux problèmes pour garantir une qualité de service (QoS) acceptable par ce genre d'application. Les fournisseurs d'accès à Internet (ISP, *Internet Service Provider*) relèvent le défi de concevoir leurs réseaux de manière à satisfaire les demandes des clients pour des services rapides, fiables et différenciés.

Après avoir rappelé les limitations du modèle de service *best effort* en matière de garantie de la QoS et avoir défini l'ingénierie de trafic et la commutation de labels avec MPLS (*Multi-Protocol Label Switching*), nous traitons de l'ingénierie de trafic *multicast*. Nous définissons d'abord l'ingénierie de trafic *multicast* (ITM) et sa particularité par rapport à l'ingénierie de trafic *unicast* (IT). Nous étudions la difficulté de combiner le *multicast* et MPLS dans un réseau. Nous décrivons les propositions MPLS pour l'ingénierie de trafic *multicast* et nous justifions la nécessité de définir un nouveau protocole. Par la suite nous proposons le protocole MMT (*MPLS Multicast Tree*), qui utilise les chemins MPLS entre les nœuds de branchement de l'arbre *multicast* afin de réduire les états de routage et d'augmenter la résistance au facteur d'échelle. Nous présentons des améliorations au protocole MMT et nous l'évaluons en terme de résistance au facteur d'échelle et d'efficacité. Nous présentons un simulateur pour le trafic *multicast* dans un domaine MPLS et nous discutons de la mise en application de MMT

dans ce simulateur. Nous présentons quelques résultats de simulation pour valider notre évaluation et nous concluons finalement que le protocole MMT semble prometteur et bien adapté à une éventuelle implémentation de l'ingénierie de trafic *multicast* dans le réseau.

4.1 Les limitations du modèle de service *best effort*

Un réseau devra fournir différents types de service pour faire cohabiter les applications habituelles et des applications sensibles à la congestion, au délai, à la gigue et à la perte de paquets, telles que par exemple la vidéo-conférence, où la qualité visuelle et la qualité audio sont essentielles. Un mécanisme de routage rapide est nécessaire pour répondre à ce besoin.

Afin de fournir une QoS suffisante aux utilisateurs d'Internet, une première solution consiste à augmenter la bande passante disponible de sorte que la capacité supplémentaire du réseau permette à tous les utilisateurs d'obtenir la QoS appropriée, évitant ainsi la congestion du réseau. Une autre solution consiste à supposer que la bande passante ne peut pas être considérée comme illimitée et donc que les ressources devraient être convenablement allouées aux utilisateurs pour leur assurer la QoS demandée. À mesure que la bande passante disponible aux utilisateurs augmente, de nouvelles applications sont développées, ce qui érode l'augmentation de la capacité du réseau. Ainsi, une gestion adéquate de ressources est nécessaire pour fournir une QoS suffisante à travers Internet.

L'un des modèles les plus prometteurs pour fournir la QoS à travers Internet est l'ingénierie de trafic qui essaye d'allouer convenablement les ressources limitées du réseau afin de fournir la QoS demandée. Plus spécifiquement, l'IT traite l'association efficace des demandes du trafic aux ressources du réseau, et reconfigure d'une manière adaptative cette association. Ainsi, l'IT et la QoS sont fortement liées puisque le rôle de l'IT est de contrôler le trafic dans un réseau et vise typiquement à maximiser l'efficacité opérationnelle de ce réseau tout en respectant certaines contraintes, tandis

que le rôle de la QoS est de spécifier[1] les besoins des applications et de réserver par la suite les ressources nécessaires pour garantir le service. Les facteurs conduisant le développement de meilleurs outils d'ingénierie de trafic incluent, en plus de la QoS, l'existence de paramètres réglables interdépendants, la croissance du réseau et la variabilité du trafic. Dans le modèle *best-effort*, les ressources disponibles du réseau ne sont pas bien employées. Cela a pour résultat d'augmenter les délais tandis qu'il y a une possibilité d'une part de fournir une meilleure QoS par la réduction du délai et la diminution du taux de perte de paquets et d'autre part d'accroître le débit des utilisateurs en utilisant la même infrastructure de réseau. Cette meilleure gestion a pour conséquence une réduction de la vulnérabilité du réseau.

Indépendamment des assurances de QoS, un autre aspect important d'Internet est l'utilisation de la bande passante. Plusieurs applications comme les services d'audio, de video à la demande et les téléconférences consomment beaucoup de bande passante. En réduisant le nombre de paquets transmis à travers le réseau, le service *multicast* augmente essentiellement la QoS donnée aux utilisateurs dû à la bande passante supplémentaire disponible dans le réseau, ce qui augmente les performances du réseau.

La commutation de labels avec MPLS, étudiée et standardisée au sein de l'IETF [RVC01], a été présentée comme une solution très intéressante pour gérer la bande passante d'un réseau. Elle consiste à commuter le trafic au niveau de la couche "Liaison de données" (couche 2 du modèle OSI) [TBCT99].

MPLS et *multicast* sont deux technologies complémentaires. La combinaison de ces deux technologies où les arbres *multicast* sont construits dans des réseaux MPLS augmente la performance du réseau et présente une solution efficace pour la résistance au facteur d'échelle *multicast*. Le service *multicast* essaye de conserver la bande passante du réseau, alors que MPLS utilisé comme outil d'IT essaye d'approvisionner les utilisateurs en bande passante d'une façon appropriée.

1. Il faut spécifier les besoins pour pouvoir y répondre.

4.2 L'ingénierie de trafic

On entend sous le terme IT toutes les méthodes permettant de gérer au mieux le trafic dans les équipements d'un réseau [XHBN00].

Les différents réseaux se développent rapidement en taille et en débit. Les fonctions de gestion du réseau qui pouvaient autrefois être gérées par un petit groupe d'administrateurs, utilisant des méthodes basées sur l'intuition et l'expérimentation, doivent maintenant être gérées par des outils d'ingénierie de trafic efficaces qui unissent les informations de configuration et d'utilisations venant de nombreuses sources. Le niveau d'intervention manuelle impliqué dans le processus d'ingénierie de trafic doit être réduit autant que possible.

Par ailleurs, l'IT est nécessaire dans l'Internet principalement parce que les protocoles de routage interne (IGP, *Interior Gateway Protocol*) utilisent toujours le plus court chemin pour expédier le trafic. Malgré le fait que l'approche du plus court chemin conserve les ressources du réseau et bien qu'il soit très simple à appliquer aux grands réseaux, il ne fait pas toujours la meilleure utilisation de ces ressources et peut également introduire les problèmes suivants :

- Les plus courts chemins de différentes sources peuvent se superposer sur certains liens provoquant ainsi la congestion de ces liens.
- Le trafic d'une source à un destinataire peut dépasser la capacité d'un lien alors qu'un autre chemin légèrement plus long reste sous-utilisé.

Le premier problème, la superposition entraînant une congestion, peut être résolu par l'augmentation de la capacité des liens, ou par application des techniques classiques de contrôle du congestion, voire les deux. Les techniques classiques pour le contrôle de congestion incluent : la limitation des débits (*rate limiting*), le contrôle de flux par fenêtre coulissante (*window flow control*), la gestion de la file d'attente des routeurs, le contrôle de l'ordonnancement (*schedule-based control*), et bien d'autres [YR95]. Ces techniques essayent de régler la demande de sorte que le trafic s'adapte aux ressources disponibles. Le deuxième problème, la congestion résultant de l'allocation insuffisante de ressources, peut être résolu par les outils d'ingénierie de trafic.

Plusieurs chercheurs ont proposés d'ajouter des capacités d'ingénierie de trafic dans les réseaux traditionnels en utilisant toujours des algorithmes du plus court chemin ([FT00, RR94]). Bien que ces propositions aient été présentées pour améliorer l'efficacité du réseau, elles souffrent de plusieurs limitations [EJLW01] :

- Les contraintes sur le trafic (par exemple, éviter certains liens pour un trafic particulier d'une source à une destination) ne peuvent pas être gérées.
- Les modifications des métriques de liens pour permettre l'association explicite du trafic à un chemin[2] tend à avoir des effets difficilement contrôlable sur le reste du réseau.
- Le partage de charge ne peut pas être fait entre les chemins de coûts différents.

L'utilisation de MPLS, de sa capacité d'acheminer le trafic *unicast* à travers des chemins explicites et de sa flexibilité de gestion du trafic qu'il apporte, permet d'éviter ces limitations.

4.3 La commutation de labels avec MPLS

Dans un réseau IP classique, chaque routeur décide, en fonction de l'adresse de destination contenue dans l'en-tête d'un paquet, si celui-ci est destiné à un des sous-réseaux directement connectés ou, dans le cas contraire, vers quel routeur voisin il doit faire suivre le paquet. Pour prendre cette décision, il utilise le contenu de sa table de routage, laquelle est construite par les protocoles de routage. Cette table associe à des adresses de sous-réseaux (ou plus généralement à des préfixes d'adresses IP) le prochain routeur sur le chemin menant vers le sous-réseau de destination. Ce préfixe pouvant être de longueur variable et l'ordre n'étant pas imposé dans la table de routage, le routeur doit examiner l'ensemble de la table de routage pour décider quelle est l'entrée de la table qui correspond le mieux à l'adresse de destination du paquet. Ce traitement est relativement coûteux du fait de la taille sans cesse croissante des tables de routage du cœur de l'Internet (environ 100 000 entrées en 2001) [Bon02].

2. Le chemin est appelé ainsi : chemin explicite.

4.3.1 L'évolution de IP à MPLS

L'évolution vers MPLS est née de la confrontation des différentes solutions élaborées pour mieux intégrer l'ATM [MS94] et l'IP. Pour ce qui est de la comparaison avec des technologies issues de l'ATM, l'essentiel des problèmes réside :

- dans le choix des protocoles de recherche de chemins, et dans la différence d'approche mode connecté/mode non connecté,
- dans la complexité à assembler/segmenter des paquets de longueur variable en cellules de longueur courte à très haut débit (difficulté pour l'ATM à supporter des interfaces à 2.5 Gbit/s et au-delà) [CSFT00].

L'idée a germé de ne garder de l'ATM que sa capacité de transfert et remplacer les protocoles de commandes jusque là associés (Q.2931, PNNI) par un contrôle direct des matrices de commutation par les protocoles de routage utilisés dans les réseaux IP (OSPF [Moy91], IS-IS [Cal90]) [CSFT00].

Les motivations pour l'élaboration d'une telle technique vont bien au-delà de la facilité d'intégration de l'IP et l'ATM. Il s'agit en effet d'enrichir les capacités de la technologie IP :

- La capacité à mettre en œuvre une ingénierie de trafic et par là même permettre la définition de contrats de qualité de service.
- La capacité à isoler des trafics autorisant ainsi la création de réseaux privés virtuels (VPN [RR99]) directement à partir du réseau et non plus seulement sur la base de techniques de tunnels. Il s'agit ici d'intégrer dans le service IP les vertus des technologies relais de trame ou ATM qui travaillent en mode connecté.
- La capacité à croître : un élément fort de l'architecture MPLS est de permettre les encapsulations successives de label. Il s'agit en fait d'une extension, cette fois non limitée, du concept établi dans l'ATM avec l'empilement VP, VC. Chaque niveau de commutation ne travaille que sur son propre niveau de label, indépendamment de l'agrégation des flux qui seront transportés. On obtient ainsi un réseau en pelures d'oignon indépendantes les unes des autres, y compris du point

de vue des protocoles de routage. Cette indépendance permet plus facilement d'envisager la conception de grands réseaux.

– Le chemin basé sur les mécanismes MPLS peut être établi en traversant plusieurs types de réseaux de transport de niveau 2 tels que ATM ou l'Ethernet. Une des vraies promesses de MPLS est la capacité de créer des circuits de bout en bout, avec des caractéristiques de performances spécifiques, à travers n'importe quel type de réseaux de transport, éliminant le besoin de réseaux de recouvrement ou de mécanismes de contrôle de niveau 2 seulement.

– L'utilisation de MPLS sur un ensemble de réseaux hétérogènes permet de remonter certains contrôle de la couche "Liaison de données" à la couche "Réseau" et amène ainsi une simplification de la gestion des réseaux pour les opérateurs.

4.3.2 Le principe de MPLS

L'idée fondamentale de la commutation de labels avec MPLS consiste à remplacer les mécanismes traditionnels de routage par des mécanismes plus rapides. Un domaine MPLS est un ensemble de routeurs et de liens (généralement un système autonome) où le routage des paquets n'est plus basé sur l'analyse de l'adresse de destination mais sur celle d'un label. En effet, un label de taille fixe ajouté[3] au paquet sert comme index à une table de commutation MPLS pour déterminer l'interface de sortie pour le paquet et la nouvelle valeur du label. L'analyse de l'adresse de destination n'est faite qu'une seule fois à l'entrée du domaine MPLS (routeur *ingress*). Cette analyse permet de choisir le premier label à appliquer au paquet. Le choix de ce label dépend aussi de l'information locale de routage. À l'intérieur d'un domaine MPLS, un routeur MPLS (appelé ci-après LSR, *Label Switching Router*) examine le label entrant (*incoming label*) du paquet MPLS, examine la table de commutation MPLS et remplace[4] ce label par un label sortant (*outgoing label*). Quand le paquet quitte le domaine MPLS, le label

3. Cette opération est appelée opération de labelisation ou opération *push*.
4. Cette opération est appelée opération d'échange ou opération *swap*.

est enlevé[5] au dernier routeur (routeur *egress*) (*cf.* figure 4.1) [XHBN00].

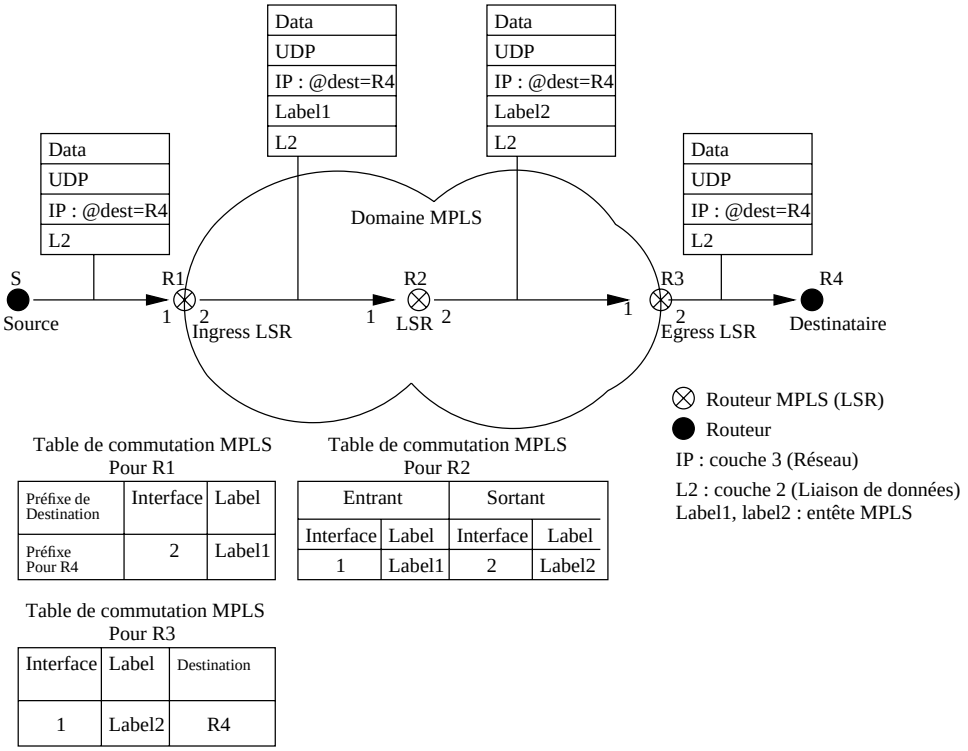


FIG. 4.1 – *Le mécanisme de routage dans un domaine MPLS.*

Pour configurer[6] MPLS sur un réseau IP, un protocole de routage *unicast* à état des liens, tel que OSPF [Moy91] ou IS-IS [Cal90], doit être utilisé. Un LSR est donc un routeur IP qui contient deux tables : une table de routage IP et une table de commutation MPLS.

MPLS enrichit le routage classique en permettant aux paquets de suivre un chemin explicite au lieu de suivre le plus court chemin traditionnel [AMAO99]. Les paquets qui possèdent le même label au niveau d'un LSR donné appartiennent à une même classe d'équivalence appelée FEC (*Forwarding Equivalence Class*) et reçoivent le même traitement au sein du domaine. Le chemin pris par tous les paquets d'une

5. Cette opération est appelée opération de dépilage ou opération *pop*.

6. C'est-à-dire distribuer les labels MPLS et construire les tables de commutation MPLS dans les routeurs.

même FEC est appelé chemin commuté ou LSP (*Label Switched Path*). Il est constitué de l'ensemble des LSR traversés par un paquet appartenant à cette même FEC. Les informations collectées par les protocoles de routage sont utilisées pour attribuer puis distribuer les labels aux LSR voisins. Ceci se fait en utilisant des protocoles de signalisation comme RSVP [ABG+01] ou LDP [ADF+01].

Il existe deux méthodes d'établissement des LSP dans un réseau : établissement classique et établissement explicite. Dans la première méthode, les chemins par défaut à l'intérieur d'un réseau MPLS sont découverts de façon classique par un protocole de routage *unicast* par échange de tables de routage. Le protocole de signalisation exploite les tables de routage ainsi établies pour déterminer et échanger des valeurs de label qui seront utilisées par les LSR pour commuter l'ensemble des paquets caractéristiques d'une FEC le long des LSP ainsi établis. La deuxième méthode consiste à définir explicitement, en plus de chaque LSP nominal établi par les protocoles de routage classique, des chemins supplémentaires ou alternatifs (à ces LSP nominaux) permettant d'atteindre une même destination.

Cette capacité de MPLS d'acheminer le trafic *unicast* à travers des chemins explicites et la flexibilité de gestion du trafic qu'il apporte, améliorent le rapport prix/performance des différents équipements de routage et l'efficacité des routeurs tout particulièrement pour les grands réseaux et enrichit les services sans que l'on ait à modifier forcément tout le réseau.

4.4 L'ingénierie de trafic *multicast*

Le trafic *multicast* possède des caractéristiques spécifiques dues à la nature des protocoles de routage *multicast* [OSL+02]. L'ITM consiste à gérer au mieux le trafic, et en particulier le trafic *multicast*, dans les équipements d'un réseau. Pour utiliser efficacement les ressources du réseau, un arbre (appelé ci-après arbre explicite) peut être construit à partir de plusieurs chemins explicites. En effet, la plupart des protocoles de routage *multicast* traditionnel sont basés sur l'arbre des plus courts chemins inverses

[Gra97] puisqu'il utilise le chemin menant à la source d'un paquet (et non pas venant de la source). Dans le cas d'un réseau asymétrique, l'arbre peut ne pas être optimal pour une source donnée. De même un arbre partagé utilise le chemin menant au point de rendez-vous (et non pas venant du point de rendez-vous). L'utilisation de l'interface RPF[7] provoquera un routage sur un chemin sous-optimal ou pourrait même empêcher des destinataires de recevoir le trafic de certaines sources. Un protocole de routage *multicast* doit être capable de proposer un chemin optimal et donc d'obtenir une interface RPF en respectant les contraintes (capacité des liens, destinataires avec des demandes de services différents, etc.). Ceci demandera des modifications importantes aux protocoles traditionnels de routage *multicast*.

De plus, comme on l'a déjà vu dans le chapitre 1, le routage *multicast* se base sur l'adresse IP *multicast* et c'est pourquoi il est très difficile d'agréger le trafic *multicast* puisque les destinataires appartenant au même groupe peuvent être situés à des multiples localisations (*cf.* sous-chapitre 1.1). En revanche, le rétablissement rapide (*fast recovery*) lors de la rupture d'un chemin est très important dans l'ITM puisque cette rupture peut isoler un sous-arbre, soit plusieurs destinataires, et n'implique pas seulement le lien fautif.

4.4.1 Les difficultés de servir de l'IP *multicast* dans un domaine MPLS

MPLS peut être employé dans un réseau pour expédier le trafic *unicast* à travers les chemins explicites et le trafic *multicast* en utilisant des arbres explicites.

MPLS présente plusieurs avantages sur les protocoles traditionnels de routage [XHBN00, RVC01]. Prenons à son compte les avantages des protocoles de commutations de la couche "Liaison de données", le service *multicast* dans les réseaux MPLS peut tirer profit de la réduction de trafic avec le service *multicast* d'une part, de la flexibilité,

7. Rappelons que nous appelons "interface RPF" (*Reverse Path Forwarding Interface*) l'interface menant à la source ou au point de rendez-vous. Cette notion fait l'hypothèse de l'existence d'un protocole de routage *unicast* et est donc dépendante de lui.

de la vitesse et de la possibilité de fournir de la QoS de MPLS d'autre part. Puisque MPLS est déjà standardisé par l'IETF, il est inévitable d'aborder les questions de l'introduction et de l'étude de l'implémentation d'IP *multicast* dans un domaine MPLS[8].

Dans [OSL+02, OL99], un cadre pour le déploiement du service *multicast* dans un domaine MPLS a été proposé. Une vue d'ensemble sur l'application des techniques MPLS sur les protocoles de routage *multicast* est étudiée et quelques caractéristiques de l'ingénierie de trafic *multicast* en utilisant MPLS ont été présentées. Ainsi, la combinaison des différents protocoles *multicast* et de MPLS a été étudiée. Les caractéristiques des protocoles *multicast* suivantes sont considérées : l'agrégation de trafic, l'inondation et l'élagage, la coexistence d'arbres basés à une source et d'arbres partagés, les arbres uni-directionnels et bi-directionnels, les données *multicast* transmis en mode encapsulé (à travers un tunnel) ou en mode natif, le contrôle selon l'interface RPF, la résistance au facteur d'échelle, la complexité du calcul pour le routage de paquets, la latence d'adhésion à un groupe et le surcoût de messages de contrôle et d'entretien de l'arbre. Les avantages et les inconvénients des protocoles de routage IP *multicast* existants dans le contexte de MPLS sont décrits et la relation avec les différents méthodes de distribution de labels sont discutées. L'étude n'a pas mené au choix d'un seul protocole de routage *multicast* mais l'auteur conclut que différents protocoles de routage *multicast* pourraient être déployés simultanément dans Internet.

Tandis que MPLS offre une grande flexibilité dans le routage de paquets, il n'enrichit pas la fonctionnalité intrinsèque du routage du *multicast* IP. Au contraire, des nouveaux problèmes surgissent en associant les arbres *multicast* de la couche "Réseau" aux LSP de la couche "Liaison de données". Plus particulièrement, lors de l'étude de l'ingénierie de trafic avec MPLS pour assurer la QoS du trafic *multicast*, plusieurs difficultés apparaissent :

La construction du LSP. La structure arborescente du *multicast* exige d'établir des LSP point-à-multipoint ou même des LSP multipoint-à-multipoint. Dans l'architec-

8. Un draft de standardisation [YPV+03] a été présenté en novembre 2003 au sein du groupe MPLS à l'IETF.

ture actuelle de MPLS, seuls les LSP point-à-point ont été étudiés. MPLS n'exclut pas d'autres types de LSP, mais aucun mécanisme n'a été standardisé pour le moment. D'ailleurs, l'adhésion dynamique au groupe *multicast* indique que les LSP *multicast* sont volatiles. Les conséquences sont un surcoût de signalisation important, et un surcoût de labels consommés.

L'initiateur de la construction de l'arbre ***multicast*.** Les mécanismes de construction des arbres *multicast* peuvent être classés selon l'initiateur de la construction de l'arbre :

- La construction de l'arbre est initiée par la source (AIS) : ce type d'arbre, calculé par la source, est dédié aux groupes ayant un nombre limité de destinataires avec peu de messages d'adhésion et de messages de désabonnement.
- La construction de l'arbre est initiée par le destinataire (AID) : ce type d'arbre est dédié aux groupes ayant un grand nombre de destinataires avec des messages d'adhésion et des messages de désabonnement fréquents. Chaque destinataire calcule indépendamment un chemin adapté à la QoS vers la source. Le calcul du chemin *multicast* est ainsi divisé en plusieurs calculs de chemin *unicast*.

Pour les deux types de construction de l'arbre, le construction peut se faire aussi par une entité centrale (qui peut être la source ou bien un routeur dédié à cette mission) : pour l'ITM, il est préférable que les informations sur le routage dans le réseau soient centralisées. En effet, les arbres *multicast* devraient être construits en prenant en compte les variations de l'ensemble formés par les destinataires et leur hétérogénéité, c'est-à-dire des destinataires avec des demandes de services différents. L'entité centrale joue un rôle important en collectant les différents changements d'adhésion aux groupes et la bande passante disponible sur les différents chemins, information importante pour la construction des LSP.

L'agrégation de trafic. Dans le contexte MPLS, le trafic est agrégé et associé aux LSP à l'entrée du domaine MPLS, ce qui favorise la résistance au facteur d'échelle puisque plusieurs trafics peuvent être associés à un chemin explicite. Un même label

peut être associé à des paquets ayant une adresse de destination correspondant à un préfixe d'adresse différent dans la table de routage IP; il suffit qu'ils empruntent le même chemin dans le domaine MPLS. Cela permet de réduire considérablement la taille des tables de commutation MPLS et la signalisation nécessaire à l'établissement des chemins. Ainsi, les trafics peuvent être agrégés beaucoup plus efficacement qu'avec CIDR [FLYV93]. En effet, pour pouvoir agréger des trafics avec IP, il faut que leurs adresses soient en quelque sorte contiguës dans l'espace d'adressage, c'est-à-dire qu'elles aient un certain nombre de bits en commun. De plus, lorsque le maillage augmente dans le réseau, l'agrégation perd de son efficacité car les tables de routage contiennent de plus en plus d'exceptions. Un des intérêts de la commutation de labels est justement de permettre l'agrégation des paquets en fonction du nœud de sortie du cœur de réseau, et non plus en fonction du sous-réseau de destination du paquet, comme c'est le cas avec IP. Bien qu'agrégés dans le domaine MPLS, dès leur sortie du domaine, les paquets sont à nouveau traités comme des paquets IP indépendants et leurs routes peuvent à nouveau diverger. L'agrégation se fait donc localement sur la base d'informations locales au cœur de réseau.

En revanche, il est difficile d'agréger le trafic *multicast* avec d'autres trafics puisque ce trafic doit suivre un arbre explicite avec des labels *multicast*. Ceci vient du fait que l'adresse *multicast* peut représenter des destinataires se trouvant dans différentes localisations physiques. Le routage *multicast* est basé sur les adresses IP *multicast*. L'objectif final est d'acheminer le paquet à tous les destinataires du groupe *multicast* associé à cette adresse IP *multicast*. Puisque les destinataires peuvent être situés n'importe où dans l'Internet, il n'y a aucune autre alternative qu'une entrée par adresse IP *multicast* dans la table de routage *multicast*. Cela veut dire qu'il est très difficile d'agréger des adresses IP *multicast*. Les labels favorisent l'agrégation d'arbre mais ne résout pas complètement le problème. En effet, on doit concevoir des algorithmes qui peuvent agréger des flux *unicast* avec des flux *multicast* ainsi que d'agréger de multiples flux *multicast* ensemble. Malheureusement, les études courantes sur l'agrégation *multicast* (*cf.* sous-chapitre 4.4.2) sont limitées à l'agrégation des états de routage dans chaque

routeur plutôt qu'à l'agrégation des LSP.

Le passage du routage de la couche "Liaison de données" au routage de la couche "Réseau" dans les LSR. Pour le trafic *unicast*, un seul label entrant sur une interface entrante correspond à un seul label sortant sur une interface sortante tandis que pour le trafic *multicast*, une seule interface entrante peut correspondre à plusieurs interfaces sortantes (aux nœuds de branchement de l'arbre). Dans ce cas, il est probable d'avoir dans ces nœuds un routage mixte de la couche "Liaison de données" sur certaines interfaces et un routage de la couche "Réseau" sur d'autres interfaces. Malgré ce routage mixte, le coût de traitement des paquets reste plus faible que celui d'un routage au niveau de la couche "Réseau".

Lors du basculement d'un arbre partagé à un arbre basé à la source (comme dans le protocole PIM-SM), certains routeurs pourraient être sur les deux arbres, et avoir les deux types d'états de routage $(*, G)$ et (S, G) pour la même adresse de destination G. Un passage du routage de la couche "Liaison de données" au routage de la couche "Réseau" peut être nécessaire pour éviter la duplication de paquets sur les deux arbres pendant la période de transition.

Cette condition est en désaccord avec la norme courante de MPLS, où il est préférable que seuls les LSR de bordure du domaine MPLS aient la capacité de routage de la couche "Réseau". Le routage mixte présente un avantage dans les LSR de bordure du domaine MPLS puisqu'il existe des branches de l'arbre menant aux destinataires à l'extérieur du domaine MPLS et ces branches n'influent pas sur les LSP des différentes branches de l'arbre à l'intérieur du domaine MPLS.

4.4.2 Les propositions MPLS pour l'ITM

Bien qu'il y ait encore des problèmes à résoudre, l'ingénierie de trafic avec MPLS offre des possibilités intéressantes pour fournir la QoS pour des communications en mode *multicast*. Il s'avère nécessaire de rappeler les différentes solutions proposées pour l'ingénierie de trafic *multicast* dans un domaine MPLS avant de présenter notre

solution.

Nous présentons d'abord les principales contributions de standardisation au sein de l'IETF du service *multicast* dans un domaine MPLS : *Engineering Paths for Multicast Traffic* [LACA99], *IP Multicast Support in MPLS Networks* [AGA99], *Using PIM to distribute MPLS Labels for Multicast Routes* [FRR00], *LDP and RSVP Extentions* [CBC+02a, CBC+02b, YK03, YI03, Che01, HL99] et *Establishing Point to Multipoint MPLS TE Tunnels* [AWK03]. Par la suite nous présentons succinctement d'autres recherches où la combinaison de MPLS et du *multicast* a été étudiée : *Aggregated Multicast* [FCGF01], *Link Failure Recovery for MPLS Networks with Multicasting* [Poi02] et *Scalable MPLS Multicast Using Label Aggregation in Internet Broadcasting Systems* [YKDKHJ+03]. Nous étudions ces propositions selon différents critères : l'agrégation, l'initiateur de la construction de l'arbre *multicast* et l'existence d'une entité centrale pour calculer l'arbre.

4.4.2.1 Les principales contributions de standardisation au sein de l'IETF

ENGINEERING-PATHS. *Engineering Paths for Multicast Traffic* [LACA99] suppose que les messages d'adhésion à un groupe *multicast* émis par les destinataires soient traités par des entités d'IT présentes dans les routeurs du chemin entre les destinataires et la source. Ainsi, l'entité d'IT détermine le chemin explicite que doit suivre un message d'adhésion et ajoute à ce message un label MPLS utilisé comme identificateur du chemin dans le domaine MPLS. Les ressources nécessaires pour les communications sont allouées en même temps que les chemins sont choisis.

Les figures 4.2 et 4.3 décrivent la procédure de découverte du chemin explicite (et le traitement du message d'adhésion) dans un routeur *egress* et dans un routeur intermédiaire, c'est-à-dire entre l'*egress* et la source. À l'arrivée d'un message d'adhésion, l'entité de routage *multicast* dans le routeur *egress* crée un état de routage *multicast* pour le groupe. L'entité de routage *multicast* détermine en utilisant l'entité d'IT (MCTE, *Multicast Control Traffic Engineering*) le prochain saut pour ce message. Si le message d'adhésion correspond à une FEC déjà définie, ce message d'adhésion est en-

voyé vers le MCTE qui alloue les ressources nécessaires et lui ajoute un en-tête MCTE (label MPLS) adéquat qui correspond au chemin explicite (*cf.* figure 4.2).

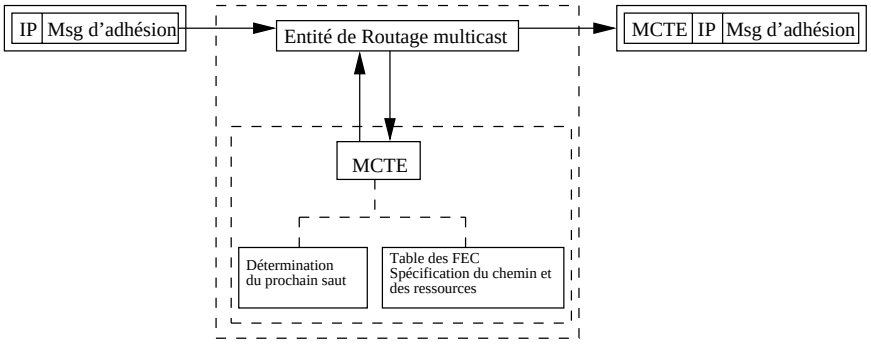


FIG. 4.2 – *Procédure de découverte du chemin explicite dans le routeur egress.*

À l'arrivée du message d'adhésion dans un routeur intermédiaire, un état de routage *multicast* est créé dans le routeur. L'entité d'IT traite le message et un nouveau chemin explicite est calculé. Un nouvel en-tête MCTE est ajouté et le message d'adhésion est envoyé de nouveau vers la source selon le chemin explicite calculé (*cf.* figure 4.3).

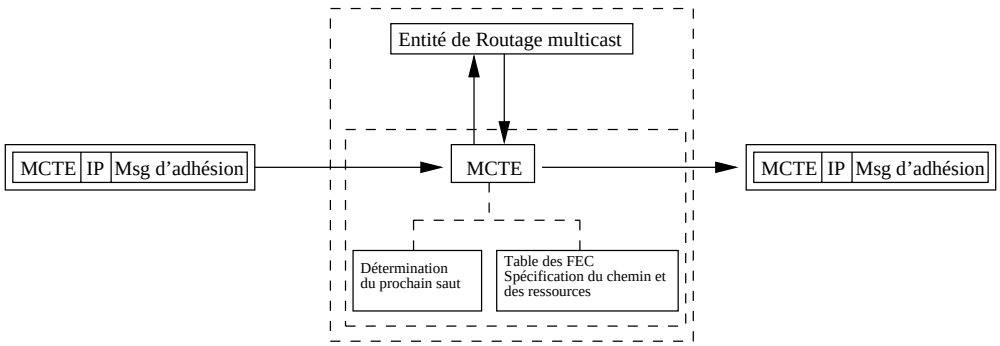


FIG. 4.3 – *Procédure de découverte du chemin explicite dans un routeur intermédiaire.*

L'idée principale de cette proposition est qu'aucune modification des protocoles de routage *multicast* traditionnel, ou des protocoles d'allocation de ressources tels que RSVP ou CR-LDP n'est nécessaire. De plus, l'entité d'IT prend en considération l'agrégation des chemins explicites choisis pour différents groupes.

Les entités d'IT de tous les routeurs doivent être informées de tout changement de topologie et de tout changement de disponibilité des ressources réservées par un

destinataire, ce qui augmente le nombre de messages de contrôle échangés entre ces entités d'IT dans le réseau. De plus, les paquets seront transmis en utilisant un protocole de routage *multicast* traditionnel selon l'adresse de groupe IP classe D de la couche "Réseau" et non pas selon le routage de la couche "Liaison de données". Les routeurs garderont des états de routage pour chaque groupe dont l'arbre passe par eux, ce qui ne simplifie pas le problème de résistance au facteur d'échelle dans le réseau.

MULTICAST-SUPPORT-MPLS. *IP Multicast Support in MPLS Networks* [AGA99] présente les deux modes dense et épars de l'IP *multicast* dans le contexte d'un réseau MPLS.

Les arbres *multicast* en mode dense sont construits par les paquets de données : les états de routage (S, G) n'existent pas avant les paquets de données. On dit que la construction de l'arbre est dirigée par les données. Les arbres construits sont basés à la source et peuvent être enracinés sur des sources différentes. L'agrégation des différents états de routage dans le même routeur est difficile : les interfaces entrantes et sortantes peuvent être différentes pour chaque état de routage. Ceci conduit à conclure que l'allocation des labels doivent être dirigée par les données et par flux (S, G).

Les arbres *multicast* en mode épars sont construits par des messages d'adhésion de la couche "Réseau" et peuvent co-exister avec un arbre partagé. Une nouvelle FEC (nommée (G, S)) est introduite. Cette FEC représente les paquets IP envoyés par la source S sur l'arbre partagé représenté par $(*, G)$. En revanche, la FEC (S, G) représente les paquets IP envoyés par la source S sur l'arbre basé à la source représenté par (S, G). Il est donc difficile d'attribuer un label unique au trafic de plusieurs sources sur une branche d'un arbre partagé.

La solution proposée est d'utiliser une allocation des labels dirigée par les données pour tous les arbres *multicast* (mode dense, mode épars avec un arbre partagé, mode épars avec un arbre basé à la source) dans un domaine MPLS. Lorsqu'un LSR reçoit un paquet *multicast* avec un label qui n'est pas déjà associé à l'interface entrante, le traitement du routage de la couche "Réseau" est utilisé. En effet, l'interface sortante est déterminée selon le routage de la couche "Réseau" et un label non utilisé, c'est-à-

dire un label *multicast* libre est donc alloué. Cette association entre label et FEC est envoyée vers les prochains routeurs et ainsi de suite.

L'avantage de cette proposition est qu'elle ne dispose pas de changements dans les protocoles *multicast* traditionnels. Mais cette solution présente plusieurs inconvénients :

- Les arbres ne s'agrègent pas tandis qu'un des buts de l'utilisation de MPLS est justement d'assurer une certaine agrégation.
- Les arbres sont construits d'une façon dirigée par les données sans tenir compte de l'IT. Dans le cas des arbres en mode épars, l'arbre est déjà construit avant même l'utilisation de MPLS.
- Il y a deux tables de labels dans chaque routeur : une table de labels pour le trafic *multicast* et une table de labels pour le trafic *unicast*.

PIM-MPLS. *Using PIM to distribute MPLS Labels for Multicast Routes* [FRR00] (appelée ci-après PIM-MPLS) est une proposition qui utilise les messages d'adhésion du protocole *multicast* PIM-SM [EFH$^+$98] pour distribuer des labels MPLS et construire ainsi l'arbre *multicast* MPLS (arbre *multicast* dont le routage est effectué au niveau de la couche "Liaison de données"). Les messages d'adhésion de PIM-SM sont modifiés pour porter un label MPLS alloué par un LSR en aval.

L'implémentation de PIM-SM dans MPLS est faite d'une façon très simple. Prenons l'exemple de la figure 4.4 où $S1$ est la source, $G1$ est l'adresse du groupe et les deux destinataires $R6$ et $R9$ veulent adhérer au canal $(S1, G1)$. Quand $R6$ adhère au canal, l'association entre label et adresse IP ressemble à celle utilisée dans l'*unicast* MPLS avec seulement une différence : les labels sont associés à une entrée $(S1, G1)$ au lieu d'être associés à une FEC[9]. L'association est faite du destinataire en amont jusqu'à la source. Quand le message d'association atteint un nœud de l'arbre (un nœud contenant une entrée $(S1, G1)$), ce nœud devient un nœud de branchement. Un nœud de branchement contient un <label entrant, interface entrante> associé à plusieurs

9. On parle dans ce cas là d'une FEC $(S1, G1)$.

<label sortant, interface sortante>.

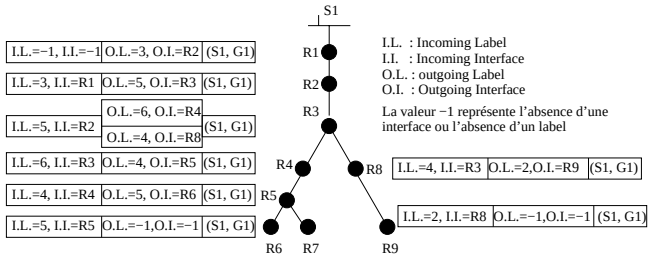


FIG. 4.4 – *Un exemple de l'implémentation de PIM-SM dans MPLS.*

Quand $R9$ adhère au même canal, l'association entre label et adresse IP est faite de la même manière de $R9$ jusqu'à la source. L'association en amont est répétée dans $R8$ jusqu'à ce que le message d'association atteigne le routeur $R3$. Le routeur $R3$ est déjà un nœud de l'arbre pour le canal $(S1, G1)$ et n'envoie donc pas le message d'association en amont. Le routeur $R3$ devient alors un nœud de branchement pour le canal $(S1, G1)$. L'acheminement de données est fait de la même manière qu'en mode *unicast*. Quand la source $S1$ souhaite transmettre des données au groupe $G1$, elle consulte sa table de commutation MPLS, trouve le label approprié au canal $(S1, G1)$, marque[10] les paquets de données et les transmet vers l'interface sortante. Tous les routeurs intermédiaires commutent le label et acheminent les paquets, exactement comme dans l'*unicast*. Dans le cas d'un nœud de branchement, le paquet est dupliqué autant de fois qu'il y a de labels sortants pour le label entrant dans la table de commutation MPLS, et pour chaque copie de paquet, le label est permuté et le paquet est transmis sur l'interface sortante.

Le désabonnement est traité de la même manière : du membre qui quitte le groupe vers la source. À chaque nœud, une désassociation du label est faite jusqu'à atteindre la source ou un nœud de branchement. Si elle atteint un nœud de branchement, seule l'interface qui correspond au <label sortant, interface sortante> est enlevée et le nœud de branchement peut redevenir un simple nœud de l'arbre.

Avec PIM-MPLS, MPLS n'est pas employé avec toute son efficacité comme un

10. C'est-à-dire ajoute un label au paquet.

outil d'IT puisque l'arbre *multicast* est construit selon le protocole PIM-SM sans tenir compte de l'IT : les chemins explicites ne sont pas utilisés. La table de commutation de chacun des routeurs d'un arbre contient un label *multicast* pour cet arbre. Ainsi le nombre d'états croît avec le nombre de groupes.

LDP-RSVP-EXTENSIONS. Plusieurs drafts IETF ont été proposés pour étendre les protocoles LDP [ADF+01] et RSVP-TE [ABG+01] afin de les adapter au service *multicast* dans un réseau MPLS.

Les deux drafts IETF [CBC+02a] et [CBC+02b] proposent d'intégrer le service *multicast* au sein même des protocoles LDP et RSVP. Les arbres *multicast* sont calculés par une entité centrale et aucun protocole de routage *multicast* n'est utilisé. Pour permettre la construction de l'arbre *multicast* MPLS, les messages de contrôle d'un arbre *multicast* (*join, leave, destroy*) ainsi que l'algorithme de la découverte de l'interface RPF sont directement implémentés dans une version modifiée de LDP ou RSVP. De nouveaux algorithmes (la demande de label (*label request*) et l'association de label (*label mapping*)) ainsi que des extensions aux messages existants dans LDP et RSVP (*hello, notification, path*) sont introduits dans LDP et RSVP. L'information complète de l'arbre (calculé par l'entité centrale) doit être stockée dans tous les nœuds de branchement de l'arbre (appelés LSR-RP). Cependant, les auteurs n'ont pas défini les critères pour choisir ces LSR-RP. De plus, des messages *hello* sont employés pour informer les LSR des adresses IP de la source et du groupe de l'arbre *multicast* : un message de *notification* est envoyé pour chaque source et groupe. Lorsque le nombre de groupes croît, le nombre de messages de contrôle (*hello* et *notification*) croît aussi et diminue la résistance au facteur d'échelle.

De même, les drafts IETF [YK03],[YI03] et [Che01, HL99] supposent que l'arbre *multicast* est calculé par une entité centrale et qu'une modification doit être faite sur les protocoles de signalisation (LDP et RSVP) en créant un nouvel objet (*Explicit Tree Object*) qui contient toutes les informations concernant l'arbre. Les drafts [YK03] et [Che01] proposent que ce nouvel objet soit intégré dans RSVP-TE.

Toutes ces propositions, d'une part, combinent MPLS et *multicast* et gèrent la QoS

en s'intéressant seulement à la réservation de ressources mais, d'autre part, augmentent la complexité de l'ingénierie de trafic *multicast*. Une nouvelle table de labels *multicast* est créée indépendamment de la table de labels *unicast* existante sans fournir une possibilité d'agrégation d'adresses *multicast* ou de labels *multicast*.

P2MP-MPLS-TE-LSP. *Establishing Point to Multipoint MPLS TE LSPs* [AWK03] utilise le protocole RSVP-TE pour construire des tunnels *multicast*. Tous les destinataires sont supposés connus par la source qui établit des tunnels p2p (point-à-point) entre la source et chaque destinataire. Un tunnel p2p est identifié par la source et un identificateur du groupe (appelé pid). Les tunnels p2p appartenant au même groupe *multicast* (ayant le même pid) sont combinés pour former ainsi un tunnel p2mp (point-à-multipoint).

Les avantages de cette proposition est qu'aucun protocole *multicast* n'est utilisé et qu'il n'y aura pas de grandes modifications des procédures du protocole RSVP-TE pour construire les tunnels p2mp. Mais il est important de distinguer les p2p LSP pour le trafic *unicast* des p2p LSP pour le trafic *multicast*. En effet, RSVP-TE est basé sur l'idée qu'un message PATH est envoyé par la source pour construire un chemin explicite entre la source et un destinataire et que le destinataire répond avec un message RESV pour la réservation de ressources sur le chemin explicite. Dans cette proposition, l'arbre *multicast* MPLS calculé par la source est ajouté au message PATH ce qui permet au destinataire de répondre et de déterminer les routeurs de branchement de l'arbre. Le traitement d'un paquet *unicast* est différent de celui d'un paquet *multicast* à la source de l'arbre. Il est donc nécessaire de faire la différence entre un p2p *unicast* et un autre *multicast* pour la même source et le même destinataire.

Conclusion Vu la nature dynamique des arbres *multicast* et le nombre croissant de messages de contrôle à échanger entre les routeurs dans un réseau, nous pensons [BC01] que la création de tunnels MPLS uniquement pour le trafic *multicast* n'est pas une solution qui résiste au facteur d'échelle. Les travaux de l'IETF sont orientés vers la création d'une entité centrale qui sera chargée de recevoir les messages d'adhésion

et de désabonnement des différents destinataires appartenant à un même groupe. L'entité centrale calcule par la suite l'arbre *multicast* et transmet l'arbre *multicast* MPLS calculé vers le réseau. Cela évite d'utiliser un protocole de routage *multicast* traditionnel. En revanche, nous étions parmi les premiers à proposer au sein de l'IETF une solution élégante [BC01] qui utilise une entité centrale pour calculer l'arbre *multicast* dans un domaine MPLS. Cette proposition est cohérente avec les idées proposées dans les travaux de standardisation [YPV+03] au sein du groupe MPLS à l'IETF du service *multicast* dans un domaine MPLS. En revanche, ces travaux [YPV+03] fixent les extensions nécessaires à RSVP-TE dans le contexte de *multicast* MPLS mais ne proposent aucune solution pour le calcul des tunnels MPLS *multicast*, ni pour la mise en œuvre de ces tunnels dans un réseau.

4.4.2.2 Les principaux travaux traitant de la combinaison de MPLS et de *multicast*

AGGREGATED-MULTICAST. *Aggregated Multicast* [FCGF01] suppose que plusieurs groupes *multicast* peuvent partager un même arbre *multicast* au lieu de construire un arbre *multicast* pour chaque groupe *multicast*. Ceci réduit le nombre d'états de routage dans les routeurs et, également, l'entretien de l'arbre au cœur de réseau. Chaque groupe *multicast* est associé à un arbre agrégé. Pour la gestion de l'arbre agrégé et l'association entre les groupes *multicast* et les arbres agrégés, une entité centrale de gestion appelé *tree manager* est introduite.

Exemple : considérons le réseau représenté sur la figure 4.5 où le domaine A est un réseau de cœur et les domaines B, C, D, X, Y sont les réseaux clients du domaine A. Considérons le groupe *multicast* $G1$ originaire du domaine D et ayant des destinataires dans les domaines B et C. Les routeurs $A1, Aa, Ab, A2, A3$ forment la partie de l'arbre *multicast* dans le domaine A qui correspond au groupe $G1$. Considérons un autre groupe *multicast* $G2$ originaire du domaine X et ayant des destinataires dans les domaines B et C. À l'intérieur du domaine A, les deux groupes $G1$ et $G2$ partagent exactement les mêmes branches de l'arbre *multicast*. Considérons maintenant

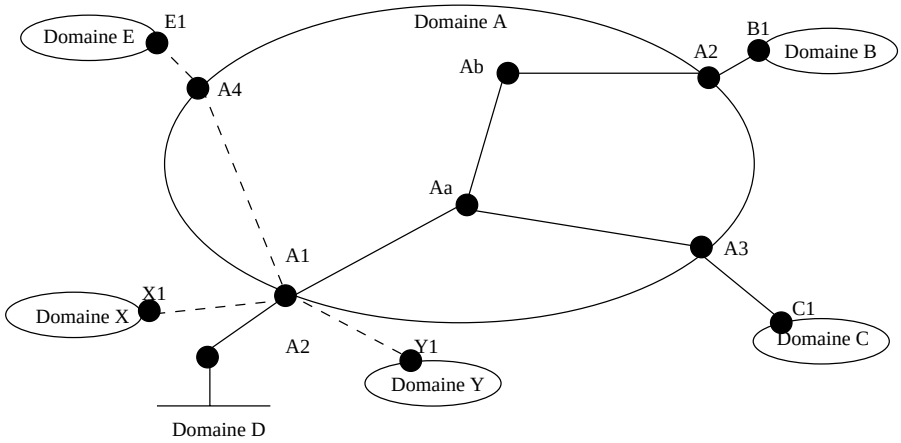


FIG. 4.5 – *L'arbre aggregated multicast.*

un troisième groupe $G3$ originaire du domaine X et ayant des destinataires dans le domaine B seulement. Les routeurs $A1, Aa, Ab, A2$ forment la partie de l'arbre *multicast* dans le domaine A qui correspond au groupe $G3$. Avec *aggregated multicast,* un arbre couvrant $A1, A2, A3$ (appelé arbre agrégé) est prédéfini avec une seule adresse *multicast*[11]. Cet arbre est partagé par tous les groupes associés par la suite à cet arbre.

Dans l'association entre groupes *multicast* et arbres agrégés, les complications surgissent quand il n'y a pas une association parfaite : aucun arbre agrégé existant ne couvre un groupe (association perméable). L'inconvénient de l'association perméable est qu'une certaine portion de la bande passante du réseau est gaspillée pour fournir des données à des nœuds qui ne sont pas impliqués dans un groupe. Dans notre exemple, l'association de $G3$ à l'arbre agrégé signifie que $A2$ recevra des paquets *multicast* destinés aux membres du groupe $G3$ bien qu'il n'y ait aucun destinataire du groupe $G3$ relié à $A2$. Bien entendu, $A2$ filtrera les paquets à destination de $G3$ et donc aucun de ses paquets ne circulera dans le domaine B.

L'inconvénient de *Aggregated multicast* est que le nombre de labels est lié au nombre d'arbres agrégés à construire dans le réseau. Si le nombre d'arbres continue à augmenter, le nombre de labels reste élevé malgré la réduction faite par rapport à PIM-MPLS. Ce phénomène est accentué en présence des contraintes de l'ingénierie

11. Les auteurs ne spécifient pas s'il s'agit d'une nouvelle adresse *multicast* ou bien d'un simple label.

de trafic puisque il faudra construire plus d'arbres agrégés pour équilibrer la charge du réseau.

SCALABLE-AGGREGATION. *Scalable MPLS Multicast Using Label Aggregation in Internet Broadcasting Systems* [YKDKHJ+03] identifie un arbre *multicast* par ses routeurs de bordure. Deux arbres ayant les mêmes routeurs de bordure doivent être agrégés. En effet, prenons l'exemple du paragraphe précédent. Les deux groupes $G1$ et $G2$ ont les mêmes routeurs de bordure ($A1$ comme *ingress* et $A2, A3$ comme *egress*). Un seul arbre agrégé représente les deux groupes $G1$ et $G2$ et cet arbre est associé à un label *multicast* unique. À l'entrée du domaine A, le label associé aux deux groupes $G1$ et $G2$ est ajouté au paquet destiné aux groupes $G1$ ou $G2$. Ceci dans le but d'assurer la résistance au facteur d'échelle en utilisant moins de labels pour les arbres *multicast* dans un réseau MPLS. Une table est créée dans chaque routeur *ingress* dont les entrées sont tous les routeurs représentant un arbre *multicast* et les labels correspondants. Un nouveau label qui correspond à l'arbre associé à $G3$ est créé dans le routeur *ingress*.

Il est considéré que les arbres ayant les mêmes routeurs de bordure sont identiques, ceci est contradictoire avec le fait que plusieurs arbres ayant les mêmes routeurs de bordures n'ont pas forcément les mêmes branches entre l'*ingress* et les différents *egress* selon la topologie et la bande passante disponible dans le réseau. Un problème se pose aussi sur la façon d'associer un nouveau groupe à un arbre et le temps que prend cette association. Un paquet pour un nouveau groupe arrivant à l'*ingress* du domaine déclenche la recherche de la FEC correspondante dans la table de labels. Si la FEC n'existe pas, l'*ingress* doit découvrir l'arbre *multicast* pour le groupe, c-à-d. les routeurs de bordure de sortie pour ce groupe, les sauvegarder dans une autre table de l'*ingress* (appelée table des nœuds des arbres *multicast*) et par la suite associer cet arbre à un label *multicast* non utilisé (*cf.* figure 4.6).

LINK-FAILURE. *Link Failure Recovery for MPLS Networks with Multicasting* [Poi02] présente une méthode pour la récupération de la panne d'un lien *multicast* dans un réseau MPLS. Une implémentation dans Linux du *multicast* dans un réseau MPLS en se

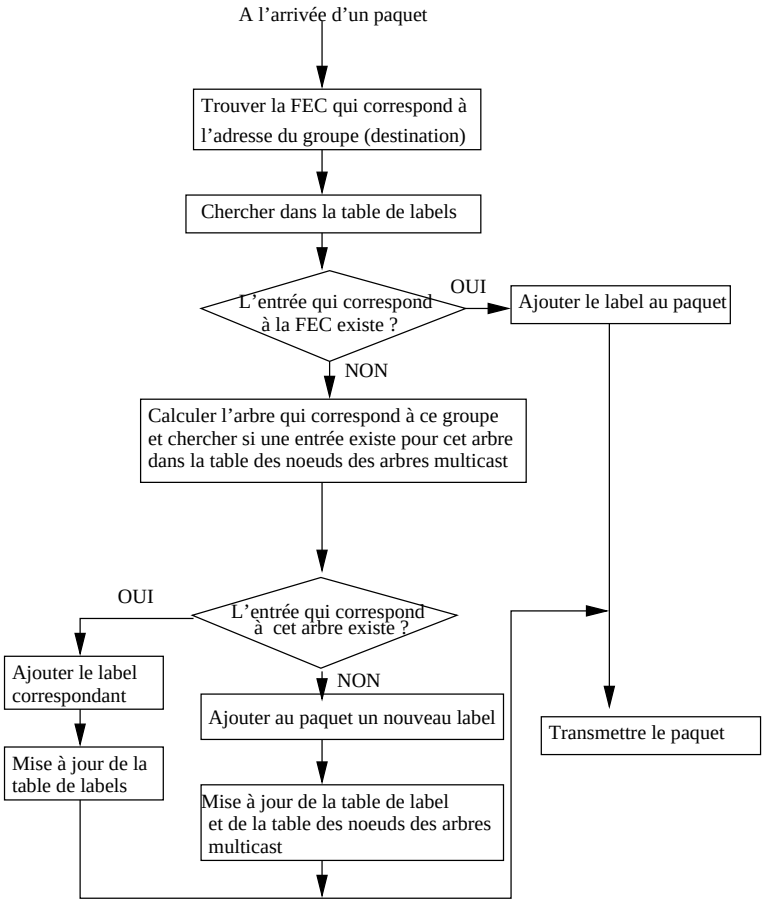


FIG. 4.6 – *L'algorithme de routage de scalable-aggregation.*

basant sur le routage explicite a été développé. Il est supposé que l'arbre est déjà calculé par une entité centrale. L'arbre est représenté dans un fichier et par la suite le protocole MulTreeLDP (qui est le nouveau protocole de signalisation *multicast* proposé par l'auteur) est chargé de construire l'arbre *multicast* MPLS. L'algorithme présenté trouve un chemin de secours qui sera utilisé en cas de panne d'un lien. Un mécanisme de re-routage rapide MPLS *multicast* est présenté et implémenté, mécanisme qui est une extension du re-routage rapide avec MPLS.

Conclusion. Les deux propositions [FCGF01] et [YKDKHJ+03] tendent à réduire la taille des tables de routage *multicast* dans les routeurs des arbres *multicast* en utilisant l'agrégation de labels dans un domaine MPLS. La proposition de Pointurier [Poi02]

est importante pour l'implémentation de re-routage rapide d'un arbre *multicast* dans un réseau MPLS. Ces propositions supposent que l'arbre *multicast* explicite est déjà calculé par une entité centrale qui possède les informations sur la topologie et sur la capacité disponible sur tous les liens. On suppose que la topologie est connue d'une façon administrative ou bien qu'un protocole de routage à état des liens est opérationnel et la base de données d'état des liens est accessible. Ces propositions traitent de la construction d'un arbre *multicast* dans un réseau MPLS tout en respectant les exigences du réseau en matière de QoS. Mais dans la proposition [FCGF01], l'association parfaite entre un groupe *multicast* et arbre agrégé en présence des contraintes d'ingénierie de trafic conduit à des tables de labels de taille importante dans les routeurs de cœur. Pour la proposition [YKDKHJ$^+$03], comme dit précédemment, la nature dynamique des arbres *multicast* et le nombre croissant de messages de contrôle à échanger entre les routeurs dans un réseau rendent la création de tunnels MPLS pour le trafic *multicast* peu résistante au facteur d'échelle. Dans les sous-chapitres suivants, nous nous servons des idées présentées pour expliquer notre motivation à définir un nouveau protocole pour assurer le service *multicast* dans un réseau MPLS où MPLS est utilisé comme outil d'ingénierie de trafic.

4.5 Le protocole MMT

Suite à l'étude de l'état de l'art à travers celles des drafts IETF et des différentes propositions précédentes, nous proposons le protocole MMT (*MPLS multicast tree*). Le protocole MMT construit un arbre *multicast* dans un réseau MPLS en considérant seulement les routeurs de branchement de cet arbre. En limitant la présence d'états de routage *multicast* aux routeurs de branchement, le protocole MMT convertit les flux *multicast* en multiple flux quasi-*unicast*.

Le protocole MMT facilite la gestion de réservation des ressources pour le trafic *multicast* dans un domaine MPLS. Il peut être employé par un ISP pour fournir le service *multicast*, pour ses clients et les réseaux voisins pairs, dans son réseau WAN

(*Wide Area Network*) et surtout dans son réseau cœur. Dans MMT, au lieu de construire un arbre pour chaque canal [12] *multicast* individuel dans le réseau cœur, on peut avoir plusieurs canaux *multicast* qui partagent des branches de leurs arbres. MMT permet la diminution de la taille des tables de routage et ainsi permet d'obtenir un gain considérable de performances car seuls les routeurs de branchement auront besoin de mémoriser les informations de routage associées à un canal *multicast*. Les LSP *unicast* sont utilisés entre les routeurs de branchement de l'arbre *multicast*. En utilisant cette méthode, nous réduisons la quantité d'informations à mémoriser dans les routeurs et nous assurons la résistance au facteur d'échelle. Notre approche est facile à déployer puisque nous utilisons pour le trafic *multicast* la technique de routage MPLS utilisée pour le trafic *unicast*.

En comparaison avec l'IP *multicast*, le protocole MMT présente plusieurs avantages qui sont détaillés comme suit :

– Il simplifie l'installation des LSP : puisque les nœuds de duplication de l'arbre *multicast* sont situés aux LSR de branchement, il n'y a aucun besoin de créer et de maintenir des LSP point-à-multipoint ou des LSP multipoint-à-multipoint. Au lieu de cela, un arbre peut être décomposé et ses branches associées à des LSP point-à-point. Les LSP point-à-point sont utilisés pour la transmission du trafic *multicast*.

– Il rend les flux *multicast* plus faciles à agréger : chaque branche d'un flux *multicast* peut être agrégée avec d'autres flux *unicast* qui partagent les mêmes LSR d'entrée et de sortie. Ainsi la résistance au facteur d'échelle de l'ingénierie de trafic MPLS ne sera pas compromise.

– Il utilise une entité centrale [13] pour assurer l'ITM dans le réseau : une entité centrale garde toute l'information nécessaire sur les liens point-à-point ou point-

12. Rappelons qu'un canal est un groupe identifié par le couple (S, G) où S est l'adresse de la source et G l'adresse du groupe.

13. L'entité centrale est un point de défaillance critique. Une certaine redondance de l'entité centrale peut assurer la survivabilité du service. Une certaine distribution de l'entité centrale est envisageable. Nous ne le traitons pas ici car: (1) elle complexifierait inutilement le discours;(2) idéalement la distribution est indépendante de l'ITM.

à-multipoint. Toutes les sources et tous les destinataires des différents groupes *multicast* ainsi que la bande passante associée sont connus et l'entité centrale est informée directement de tout changement de topologie du réseau (pannes de liens ou de routeurs) et de tout changement d'appartenance d'un destinataire à un groupe. Un arbre est calculé à l'aide de cette entité centrale et transmis par la suite sur le réseau (*cf.* paragraphe 4.5.3).

– Il est inter-opérable avec d'autres protocoles *multicast* : ce protocole peut être limité à un seul domaine (typiquement aux domaines de cœur). Dans les autres domaines, les protocoles de routage *multicast* traditionnel peuvent être utilisés et une fois transmis dans le domaine MPLS, les paquets *multicast* seront traités facilement par les mécanismes du protocole MMT.

Dans les paragraphes suivants, nous présentons le concept de MMT, la transmission de paquets *multicast*, le rôle de l'entité centrale chargée de calculer l'arbre et de faire une collecte des états de liens et des adhésions aux groupes ainsi que d'exécuter l'algorithme de l'association groupe-arbre, et enfin nous présentons la construction de l'arbre MPLS ainsi que les nouveaux LSP. Nous évaluons l'approche proposée et en concluons qu'elle est adaptée à nos besoins et prometteuse.

4.5.1 Le concept du protocole MMT

La figure 4.7 représente une topologie hiérarchique inter-domaine. Le domaine A est l'épine dorsale du réseau national ou régional d'un ISP, et les domaines C, D et E sont des réseaux de clients du domaine A. Les domaines G, F et B sont d'autres réseaux de clients du domaine A. Soit un canal originaire du domaine H et ayant des membres dans les domaines B, C et D. Sans tenir compte des détails du protocole *multicast* intra-domaine existant, supposons que l'arbre *multicast* pour ce canal est formé par les chemins suivants ($A1-A3$, $A3-A4$, $A3-A5$). Considérons un deuxième canal originaire du domaine H et ayant des membres dans les domaines B, C et E. De même, l'arbre *multicast* pour ce canal est formé par les chemins suivants ($A1 - A3$, $A3 - A4$, $A3 - A6$). Considérons maintenant un troisième canal originaire du domaine

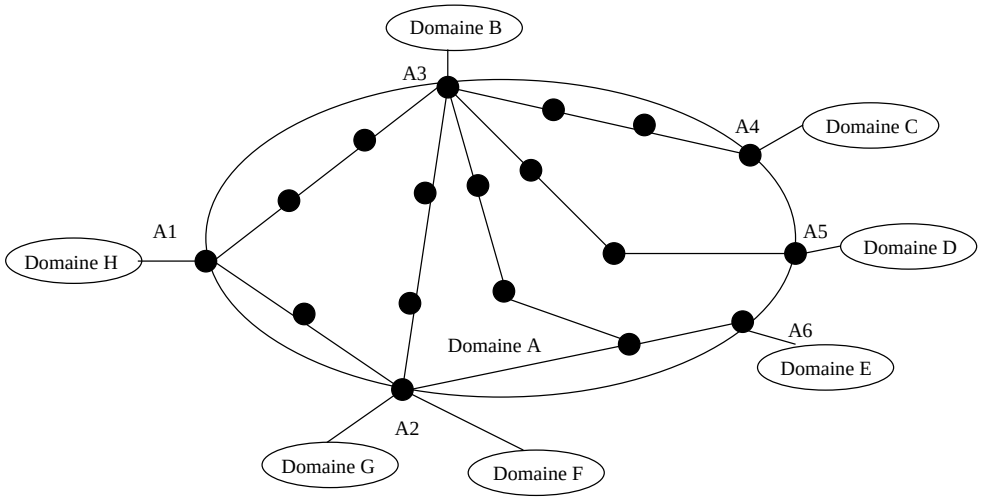


FIG. 4.7 – *Le concept de l'arbre multicast MPLS.*

G et ayant des membres dans les domaines B, C et E. De même, l'arbre *multicast* pour ce canal est formé par les chemins suivants ($A2 - A3$, $A3 - A4$, $A3 - A6$).

Dans le routage *multicast* traditionnel, un état de routage séparé existe dans tous les routeurs appartenant à l'arbre correspondant à un canal pour chacun des trois canaux de l'exemple précédent malgré le fait que les arbres *multicast* soient pratiquement identiques ou bien qu'ils aient beaucoup de routeurs en commun. De plus, il est impossible de faire l'agrégation de différents flux malgré qu'ils partagent une grande partie de leur chemins. En effet, avec les protocoles *multicast* traditionnels, il n'est pas possible d'agréger les adresses IP *multicast* : l'adresse IP *multicast* n'est pas liée à une localisation physique donc les différents destinataires appartenant au même canal *multicast* peuvent être très dispersés. Les trois canaux partagent $A3 - A4$, les canaux 1 et 2 partagent $A1 - A3$ et $A3 - A4$ et finalement les canaux 2 et 3 partagent $A3 - A6$.

Dans MMT, les LSP de MPLS sont utilisés entre les routeurs de branchement. Dans notre exemple du paragraphe précédent, les LSP $A1 - A3$, $A3 - A4$, $A3 - A5$, $A3 - A6$, $A2 - A3$ sont utilisés si on applique le protocole MMT. Ainsi, les routeurs de bordure (qui constituent les routeurs d'extrémités des LSP (*ingress et egress*)) et les routeurs qui sont des routeurs de branchement appartenant à l'arbre *multicast*, maintiennent les états de routage pour l'arbre. Tous les autres routeurs intermédiaires n'ont pas à

maintenir les états de routage *multicast*. Les labels *unicast* sont utilisés et l'agrégation est possible puisque nous utilisons pour les paquets *multicast* l'agrégation des labels au lieu de l'agrégation des adresses IP *multicast*.

4.5.2 La transmission de paquets *multicast*

Nous distinguons deux niveaux de routage : le routage externe au domaine A où le routage IP est appliqué à un paquet *multicast* et le routage interne au domaine A où un paquet est acheminé selon le routage MPLS. Pour le routage externe, un paquet *multicast* peut être acheminé suivant l'adresse de destination IP *multicast*. À l'entrée du domaine A, les paquets *multicast* sont encapsulés et diffusés entre les routeurs de branchement. L'encapsulation des paquets *multicast* dans des paquets *unicast* a été étudiée dans [TN98], [BC03], [SEZ00] et [HCFCD01], mais cette encapsulation augmente la complexité du traitement. Le problème consiste à réduire l'effet de l'encapsulation des paquets IP *multicast* dans les paquets IP *unicast*. Pour cela le paquet est analysé selon son en-tête IP *multicast*. Le routeur détermine quels sont les prochains routeurs de branchement pour ce paquet. Selon cette information, les copies multiples des paquets sont produites et un label MPLS est ajouté au paquet *multicast* selon le prochain routeur de branchement. En arrivant au prochain routeur de branchement, le label est enlevé et le même processus est répété, jusqu'à ce que le paquet arrive à sa destination. Si le paquet *multicast* passe d'un domaine MPLS à un autre domaine non MPLS, alors le paquet est acheminé par les protocoles *multicast* existants dans ce nouveau domaine.

4.5.3 Le rôle de l'entité centrale NIMS

Un routeur de branchement doit découvrir ses prochains routeurs de branchement sur l'arbre *multicast*. Pour ceci, nous proposons d'utiliser un routeur de gestion dans chaque domaine, appelé NIMS (*Network Information Management System*), chargé de recevoir les messages d'adhésion et les messages de désabonnement de tous les membres des canaux dans ce domaine. Il est informé de la topologie du réseau et de

l'existence des LSP *unicast* (*cf.* figure 4.8). Ceci est conforme avec les orientations de l'IETF au sujet du *multicast* MPLS. Le NIMS peut par exemple être élu par un mécanisme semblable à celui du routeur rendez-vous dans PIM-SM [EFH+98][14].

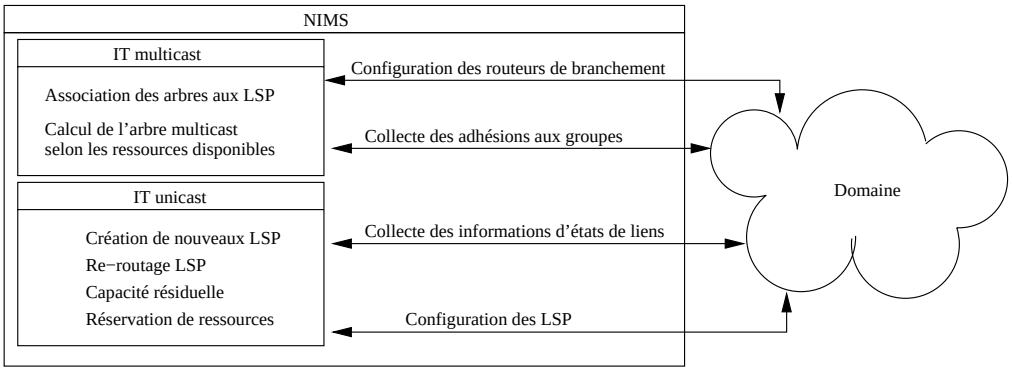


FIG. 4.8 – *Les composants du routeur de gestion des informations du réseau (NIMS).*

Le NIMS calcule l'arbre *multicast*, détermine les routeurs de branchement et installe les états de routage *multicast* dans ces routeurs (les adresses de leurs prochains routeurs de branchement) sur l'arbre *multicast*. MPLS est utilisé entre ces routeurs de branchement : les paquets *multicast* voyagent d'un routeur de branchement à un autre en utilisant les LSP préexistants dans le réseau pour l'*unicast* (*cf.* figure 4.9).

Comme nous pouvons le remarquer sur la figure 4.9, le routeur $R4$ garde un état de routage *multicast* pour le canal (S, G). Mais il y a un passage du routage de la couche "Liaison de données" au routage de la couche "Réseau" dans le routeur $R4$. Nous discutons dans le sous-chapitre 4.6.3 l'utilisation d'un label *multicast* pour identifier l'arbre et par la suite les labels *unicast* sont empilés au dessus de l'ancien label dans le paquet.

Le calcul de l'arbre *multicast* consiste à découvrir tous les routeurs de branchement pour ce groupe. Le NIMS envoie alors des messages *branch* à tous les routeurs de branchement pour les informer de leurs prochains routeurs de branchement. À la réception de ce message, un routeur de branchement crée un état de routage *multicast*

14. Le NIMS peut être différent au sein du même domaine pour chaque canal (S, G). On a ainsi naturellement une équilibrage de la charge, une répartition du service NIMS et une survivabilité du système accrue.

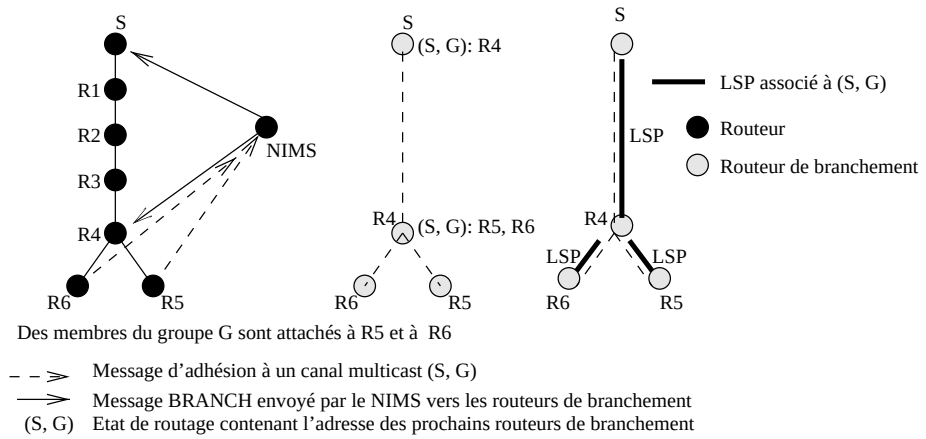


FIG. 4.9 – *La construction de l'arbre multicast MPLS.*

pour le canal *multicast*. Une fois les routeurs de branchement et leurs prochains routeurs identifiés, les paquets de données seront envoyés d'un routeur de branchement aux autres jusqu'à atteindre leurs destinations. Les LSP *unicast* déjà établis sont employés entre les routeurs de branchement d'arbre *multicast* afin de réduire la taille des états de routage et afin d'augmenter la résistance au facteur d'échelle.

4.5.3.1 Le NIMS et le routage inter-domaine

Si le NIMS découvre qu'il y a des sources appartenant à d'autres domaines, il considère alors les routeurs de bordure amenant vers ces domaines comme des sources virtuelles. Par la suite, ces routeurs envoient des messages d'adhésion (S, G) vers les routeurs de bordure des autres domaines.

Deux cas peuvent se présenter :

– Le domaine qui reçoit ces messages d'adhésion (S, G) applique notre approche et possède donc un NIMS. Dans ce cas, les routeurs de bordure de ce nouveau domaine reçoivent les messages d'adhésion (S, G) et informent à leur tour le NIMS.

– Le domaine qui reçoit ces messages d'adhésion (S, G) applique une autre approche. Dans ce cas, le protocole de routage *multicast* présent dans le domaine construit l'arbre de routage *multicast* en se servant des messages d'adhésion

(S, G) et crée des états de routage sur tous les routeurs de l'arbre.

Lorsqu'un paquet *multicast* arrive à un routeur de bordure dans un domaine qui implémente notre approche, un label est ajouté et le paquet est ainsi envoyé vers tous les destinataires.

4.5.3.2 La collecte des informations d'états de liens

Le NIMS doit obtenir les informations sur les états de liens de tous les routeurs dans le domaine afin de calculer un arbre approprié pour chaque canal *multicast*. Si le réseau est petit et très stable, les administrateurs de réseau peuvent configurer le NIMS manuellement. Naturellement, ce n'est pas le cas typique. D'une façon générale, la charge du réseau est sujette à des changements fréquents. Dans le cas dynamique, chaque routeur dans le domaine doit informer le NIMS des changements, tels que des liens ou routeurs tombant en panne, ou des liens ou routeurs apparaissant. Le routage *unicast* à états de liens (par exemple OSPF) est utilisé pour le routage *unicast*. Dans ce cas le NIMS[15] tirera bénéfice de l'inondation des paquets à états de liens de tous les routeurs, et obtiendra ainsi la topologie du réseau facilement.

L'annonce d'états de liens est très semblable à MOSPF [Moy94b] sauf qu'ici, seul le NIMS calcule les arbres *multicast*. Puisque tous les routeurs sont informés de la topologie du réseau et des membres des groupes *multicast*, chaque routeur a la capacité d'agir en tant que NIMS. Si le NIMS original échoue, un nouveau NIMS peut être élu parmi les routeurs du cœur de réseau.

4.5.3.3 La collecte des adhésions aux groupes

Pour trouver un arbre approprié ou pour construire un nouvel arbre pour un groupe *multicast*, le NIMS doit connaître rapidement les adhésions au groupe. On dit qu'un routeur (de bordure d'un domaine) devient membre d'un groupe lorsque la route choisie entre la source et un des membres passe par ce routeur. Pour la collecte d'adhésion

15. Le NIMS est un routeur OSPF.

aux groupes, la manière la plus simple est que chaque routeur de bordure envoie son adhésion au groupe directement au NIMS. D'ailleurs, si le routage *unicast* utilise l'approche à états de liens, l'adhésion peut être inclue dans le paquet d'état du lien. À savoir, les routeurs de bordure ajoutent des informations aux paquets d'états de liens pour indiquer à quels groupes ils veulent adhérer. Ainsi, après que le NIMS ait reçu les paquets à états de liens de tous les routeurs, il peut calculer un arbre *multicast* pour un groupe *multicast*.

4.5.3.4 L'algorithme d'association d'un groupe à un arbre dans le NIMS

Afin de construire un arbre pour un nouveau groupe *multicast* ou bien une branche pour accéder à un nouveau membre d'un groupe *multicast*, le NIMS doit maintenir la liste et la localisation des membres de différents groupes *multicast*, la bande passante nécessaire pour chaque groupe et une table associant arbre et groupe (ici on entend par arbre l'ensemble des branches, et par la suite celui des LSP constituant l'arbre).

Une organisation possible de cette information est montrée sur la figure 4.10. Nous supposons que la bande passante est le seul critère de QoS pour chaque groupe *multicast*. Cela peut être facilement étendu à n'importe quel autre critère de QoS. Les données des tables dans la figure 4.10 sont celles qui peuvent résulter de la topologie présentée sur la figure 4.7. La table des associations contient chaque arbre pour chaque groupe. La virgule dans "$A1(A3(A4, A5))$" (*cf.* la table du routeur de branchement dans la figure 4.10) indiquent que $A4$ et $A5$ sont deux routeurs de branchement frères, et les parenthèses indiquent que $A3$ et $A4, A5$ sont respectivement des routeurs de branchement fils de $A1$ et $A3$.

Le NIMS calcule les arbres en se basant sur l'adhésion au groupe *multicast* et il découvre les branches que les paquets doivent suivre tout en tenant compte de la bande passante sur les différentes branches de l'arbre. Si le chemin est utilisable (c'est-à-dire qu'il y a une bande passante suffisante), l'arbre est construit, sinon l'algorithme cherche un autre arbre avec un faible coût. Il faut noter que la bande passante résiduelle est calculée en fonction du trafic *unicast* aussi, puisque le LSP utilisé doit satisfaire les

GID	Membres	Bande passante demandée (Mbit/s)
G0	s: A1 d: A4, A5	1
G1	s: A1 d: A4, A6	1.5
G2	s: A2 d: A4, A6	0.2

s : source d : destinataire

LSPID	Groupes participants	Bande passante résiduelle
A1−A3	G0, G1	...
A2−A3	G2	···
A3−A4	G0, G1, G2	···
A3−A5	G0	···
A3−A6	G1, G2	···

GID	Routeurs de branchement
G0	A1(A3(A4,A5))
G1	A1(A3(A4,A6))
G2	A2(A3(A4,A6))

FIG. 4.10 – *Les données des différents arbres dans le routeur NIMS.*

conditions de l'*unicast* et du *multicast* en même temps.

Le NIMS doit respecter la formule suivante :

$$BP_{LSP}(l) < BP_{Res}(l),$$

où $BP_{LSP}(l)$ est la bande passante pour un lien l appartenant à un LSP d'un certain arbre *multicast* calculé et $BP_{Res}(l)$ est la bande passante disponible pour ce lien l.

Considérons maintenant la figure 4.11 qui représente une topologie intra-domaine. Un canal (groupe $G0$[16]) est originaire de A et a comme membres D et G. L'arbre *multicast* pour ce canal est formé par les chemins suivants ($A-B$, $B-C$, $C-D$, $C-F$, $F-G$). Considérons un deuxième canal (groupe G1) originaire de H et ayant des membres dans les domaines D et G. L'arbre *multicast* pour ce canal est formé par les chemins suivants ($H-E$, $E-B$, $B-C$, $C-D$, $C-F$, $F-G$). Considérons maintenant un troisième canal (groupe G2) originaire de A et ayant des membres dans les domaines D et G. Normalement, l'arbre *multicast* pour ce canal doit être formé par les chemins suivants ($A-B$, $B-C$, $C-D$, $C-F$, $F-G$). Supposons qu'il n'y a pas de bande passante disponible pour ce groupe sur le LSP $B-C$. Le NIMS, ne

16. Pour simplifier les figures nous utilisons Gi comme identificateur du canal mais normalement le canal est identifié par un couple (S,G) où S est l'adresse de la source qui peut être dans un autre domaine et G est l'adresse du groupe.

peut plus de construire l'arbre *multicast* sur le plus court chemin, il va alors choisir un chemin plus long mais moins chargé. Le NIMS dans notre exemple choisit le chemin ($A-B$, $B-E$, $E-F$, $F-C$, $C-D$, $F-G$).

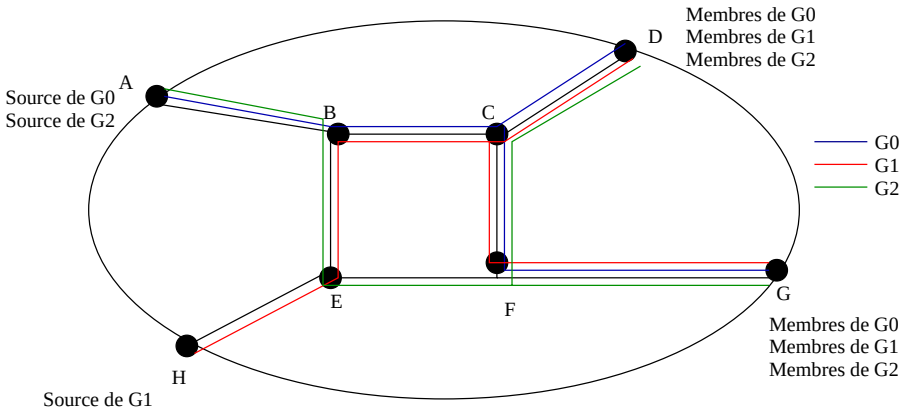


FIG. 4.11 – *Un exemple de réseau avec des arbres multicast dans MMT.*

Nous remarquons dans notre exemple que seulement les chemins $C-D$ et $C-G$ sont à agréger directement et qu'il n'y a pas d'autres chemins à agréger directement. Ceci est dû du fait que C est un routeur de branchement pour les deux canaux *multicast* (groupes $G0$ et $G1$). Notons que selon MPLS, il est possible d'assurer une agrégation partielle de la partie commune des chemins qu'empruntent les paquets. Un nouveau label est empilé au-dessus de l'ancien label dans le paquet. À l'extrémité de la partie commune des chemins, le label est enlevé et l'ancien label est utilisé pour acheminer le paquet.

4.5.4 La construction de l'arbre MPLS ainsi que des nouveaux LSP

En première analyse, on pourrait penser que les paquets *multicast* doivent suivre un chemin indépendant de celui suivi par les paquets *unicast*. Dans ce cas les techniques de routage décrites dans [RTF+01] sont utilisées. Un routeur doit mémoriser un label MPLS pour chaque destination *unicast* et un autre (parmi les labels *multicast*) pour chaque groupe *multicast* même si au bout du compte les deux types de paquets

GID	Membres	Bande passante demandée(Mbit/s)
G0	s : A d : D, G	1
G1	s : H d : D, G	1
G2	s : A d : D, G	1

s : source d : destinataire

GID	Routeurs de branchement
G0	A(C(D,G))
G1	H(C(D,G))
G2	A(F(D,G))

LSPID	Groupes participants	Bande pasante résiduelle(Mbit/s)
A−B	G0, G2	...
B−C	G0, G1	0
H−E	G1	···
B−E	G2	···
E−B	G1	···
E−F	G2	···
C−D	G0, G1, G2	···
C−F	G1	···
F−C	G2	···
F−G	G0, G1, G2	···

FIG. 4.12 – *Les données des différents arbres dans le routeur NIMS pour le réseau de la figure 4.11.*

ont la même destination. Ceci amène à une complication du traitement et une charge supplémentaire pour les routeurs.

4.5.4.1 Les LSP *unicast* et les LSP *multicast*

Dans notre approche, nous supposons que les paquets *multicast* utilisent les LSP qui existent déjà pour l'*unicast* et par la suite nous n'avons pas besoin de construire de nouveaux LSP pour le trafic *multicast* (sauf dans le cas où aucun LSP n'existe pour atteindre une destination). Aucune modification des protocoles de construction des chemins (comme LDP) n'est nécessaire et les mêmes labels sont utilisés pour le trafic *multicast* et pour le trafic *unicast* entre deux routeurs de branchement. Puisque nous utilisons les techniques MPLS, la répartition de charge *multicast* entre deux routeurs de branchement peut être réalisée du fait que le paquet *multicast* choisit le LSP le moins chargé.

Pour l'ITM, le NIMS sera chargé aussi de trouver les meilleurs chemins pour les paquets. Le NIMS peut utiliser, en plus des informations dynamiques sur la topologie et les membres de groupes, les informations statiques acquises pendant une certaine période sur la charge de chaque lien et la dynamique des adhésions et des départs des membres pour chaque canal *multicast*. Ainsi, lorsqu'un paquet est envoyé en mode

multicast, les LSP de l'*unicast* sont déjà créés en tenant compte de la charge induite par la présence des paquets *multicast* sur ces LSP.

4.5.4.2 Calcul de l'arbre MPLS dans le NIMS

La source S annonce sa volonté de diffuser des informations en utilisant un certain groupe G. Par exemple, cette annonce sera affichée sur une page web. Tous les destinataires intéressés par ces informations envoient leur message d'adhésion vers la source S. Lorsque ce message d'adhésion arrive au domaine qui applique notre approche, un état de routage correspondant au canal (S, G) est créé dans le routeur de bordure recevant ce message d'adhésion. Par la suite, le message d'adhésion est envoyé au NIMS. Le NIMS calcule ainsi l'arbre et les différents LSP *unicast* à utiliser par cet arbre. Il n'est pas nécessaire de construire l'arbre *multicast* MPLS. Il suffit d'informer les routeurs de branchement et le routeur *ingress* de leurs prochains routeurs de branchement, donc des prochains LSP à utiliser. Pour ceci, le NIMS envoie un message *branch* vers l'*ingress* et les routeurs de branchement. Ce message *branch* contient l'adresse du groupe (S, G) et les prochains routeurs de branchement.

La figure 4.13 représente les états de routage dans les routeurs de branchement et dans les routeurs intermédiaires pour le canal $(S, G0)$ de l'exemple de la figure 4.11. Notons que les tables de routage contiennent l'interface menant au prochain routeur et non pas l'adresse du prochain routeur utilisé dans la figure pour des raisons de simplification de la présentation.

4.6 Une extension du protocole MMT : le protocole MMT2

Si un routeur du cœur est un routeur de branchement pour un arbre *multicast*, et qu'il y ait un passage obligatoire de la couche "Réseau" à la couche "Liaison de données" pour chaque paquet *multicast*, cela peut être considéré comme un surcoût inacceptable. Nous présentons dans ce sous-chapitre comment le protocole MMT évite ce problème, nous présentons le protocole ERM qui propose une approche similaire,

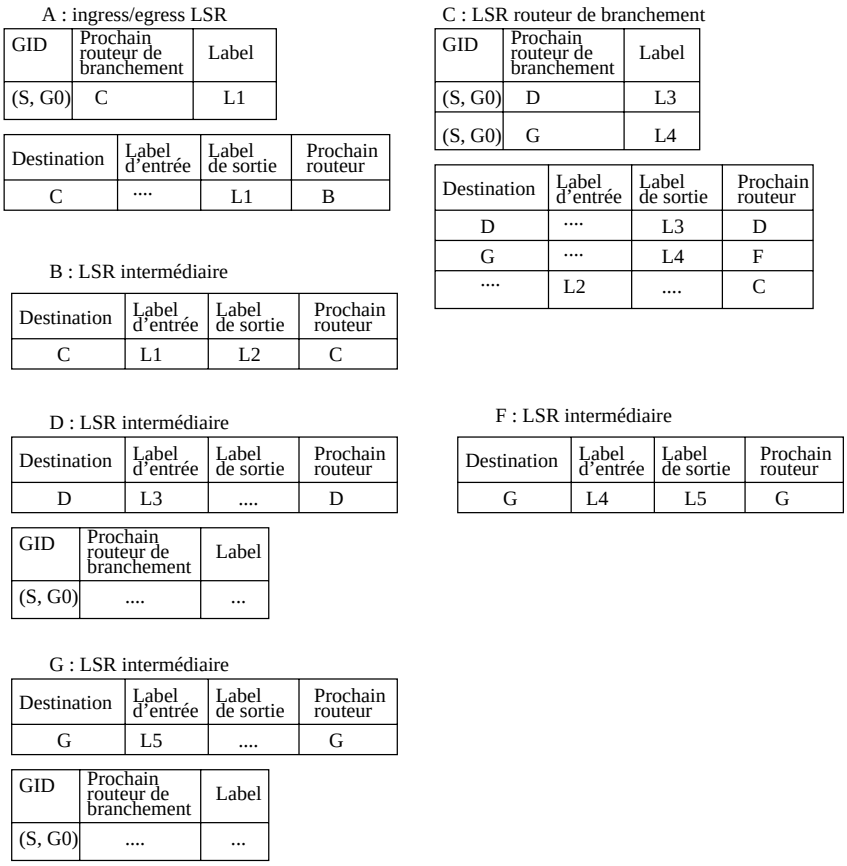


FIG. 4.13 – *Les états de routage dans les routeurs.*

et nous proposons enfin le protocole MMT2 : une extension du protocole MMT pour éviter le passage de la couche "Réseau" à la couche "Liaison de données" et pour simplifier le routage *multicast* MPLS.

En général, les réseaux des ISP sont constitués de points de présence (POP, *Point of Presence*) liés entre eux. La figure 4.14 représente une partie du réseau d'un ISP décrit dans [XHBN00]. Un POP est une combinaison d'un ou plusieurs routeurs d'accès (RA) connectés aux clients, routeurs de bordure (RB) connectés à d'autres ISP, routeurs de *hosting* (RH) connectés aux serveurs WEB des compagnies comme Yahoo, et les routeurs de cœur (RC) connectés à d'autres POP. Les paquets d'un RA, RB, RH adressés à d'autres POP doivent être envoyés vers le RC. Le RC transmet ainsi les paquets vers les autres RC dans d'autres POP.

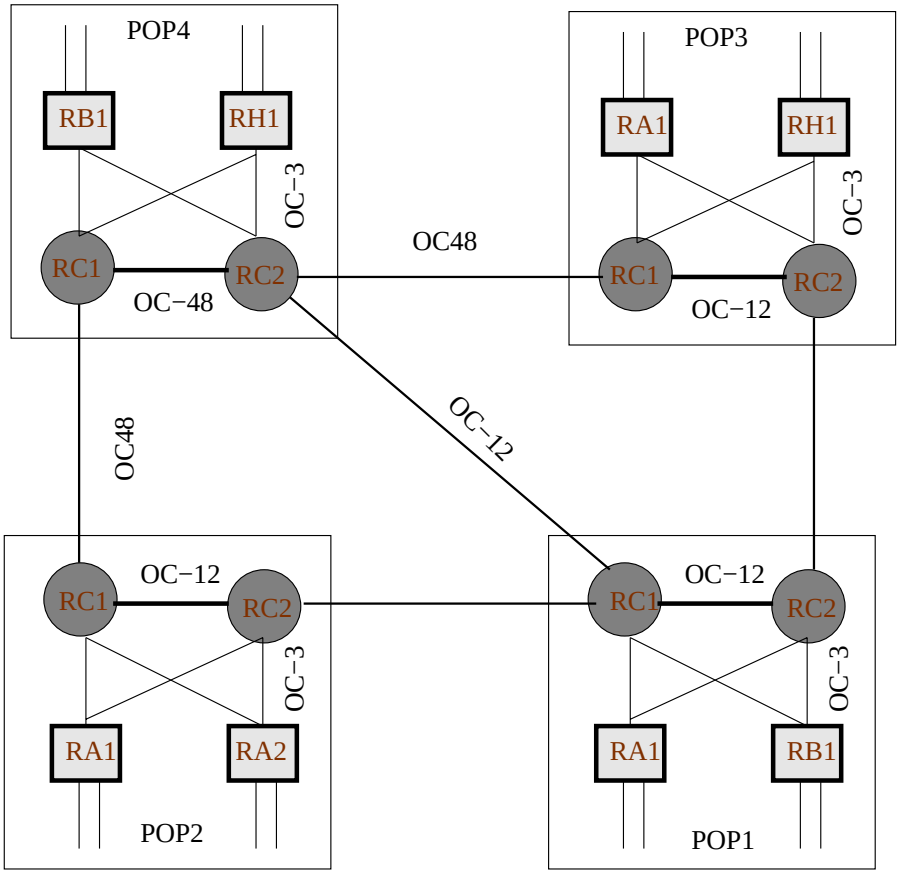


FIG. 4.14 – *Une partie du réseau d'un ISP.*

Il y a deux façons majeures de concevoir le transport de paquets IP sur un réseau à commutation de niveau 2. La première consiste à reporter toute la complexité des décisions de routage à la frontière du réseau et à faire établir les chemins par des entités spécialisées qui utiliseront ensuite la signalisation propre au réseau commuté (par exemple, Q.2931 pour un réseau ATM) pour établir des circuits de niveau 2 permanents ou semi-permanents. À l'extrême, un maillage complet de circuits de niveau 2 relie chaque équipement frontière à tous les autres et le routeur d'entrée dans le réseau se contente de déterminer quel est le routeur de sortie.

La seconde façon d'opérer est plus innovante; elle consiste à utiliser les protocoles de routage IP pour effectuer la signalisation du réseau commuté de niveau 2. Cette approche permet une mise en place automatique des circuits sur la base des informa-

tions de routage, sans imposer de mécanisme de traduction d'adresses et de localisation propre aux solutions classiques de transport d'IP sur infrastructure ATM. Cette solution permet de laisser les décisions de routage aux protocoles de routage IP qui fonctionnent déjà et qui sont bien connus. De plus, elle n'est pas incompatible avec des mécanismes de gestion de trafic plus élaborés qui construiront des chemins en fonction des besoins, ceux-ci cohabitant avec les chemins construits classiquement.

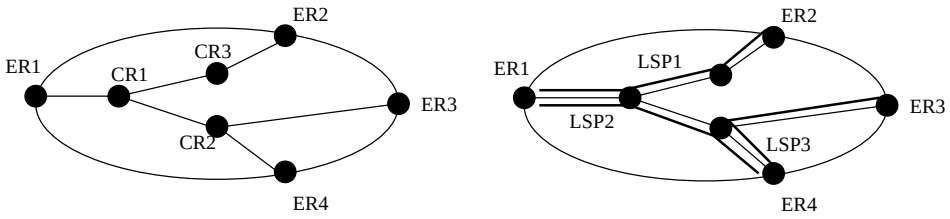
4.6.1 Le protocole ERM

Edge Router Multicasting [YM02] est aussi une proposition pour l'ITM en utilisant MPLS. Cette proposition est basée sur les mêmes principes que le protocole MMT. ERM limite les points de branchement de l'arbre *multicast* aux routeurs de bordure des domaines MPLS. Les paquets sont envoyés sur les branches à l'aide des tunnels MPLS établis entre les routeurs de bordure à travers les routeurs du cœur. En conséquence, comme dans MMT, les installations de LSP *multicast*, l'association des flux *multicast* et l'agrégation du trafic *multicast* sont transformées en simples problèmes *unicast*.

Considérons les figures 4.15(a) et 4.15(b) avec les routeurs LSR de bordure $ER1$, $ER2$, $ER3$ et $ER4$ comme membres d'un groupe *multicast*. La figure 4.15(a) représente l'arbre *multicast* construit en utilisant les protocoles de routage IP *multicast* traditionnel. Les nœuds de branchement sont les LSR du cœur $CR1$ et $CR2$. Dans ERM, un arbre *multicast* s'embranche au LSR de bordures ER1 et ER4. Il est relié par les LSP déjà existants dans le réseau : LSP1, LSP2, et LSP3 respectivement, comme indiqué sur la figure 4.15(b). En limitant les points de branchement seulement aux bords, conceptuellement, ERM transforme un flux *multicast* en multiples flux quasi-*unicast*.

Les auteurs ont étudiés deux versions d'ERM. La première version (ERM1) est basée sur des modifications des protocoles *multicast* existants :

- Les routeurs de bordure sont sélectionnés comme les routeurs RP dans le cas de PIM-SM ou CBT.
- Un message d'adhésion n'est traité que dans un routeur de bordure (pour éviter qu'il y ait des routeurs de branchement parmi les routeurs du cœur).



(a) Exemple du protocole ERM - arbre *multicast*.

(b) Exemple du protocole ERM - arbre *multicast* MPLS.

FIG. 4.15 – *Le protocole ERM.*

ERM1 exige toujours que les routeurs du cœur participent au processus de routage *multicast* et à la construction de l'arbre. La deuxième version (ERM2) est conçue pour éviter cette contrainte. En effet, un routeur est élu comme étant un contrôleur *multicast* (MM, *Multicast Manager*) et il est chargé de gérer les appartenances aux groupes dans un domaine MPLS (il a un rôle similaire au rôle de NIMS dans MMT). À la réception d'un message d'adhésion à un groupe donné, un routeur de bordure envoie un message d'adhésion au MM dans son réseau. En recevant ce message d'adhésion, le MM répond avec une liste de routeurs de bordure. Le MM calcule l'arbre (comme dans le cas de NIMS) en évitant que des routeurs du cœur soient considérés comme routeurs de branchement de l'arbre.

4.6.2 Discussion sur ERM et MMT

Nous remarquons d'abord que la proposition du protocole ERM est inspirée de la proposition du protocole MMT. Mais il n'est pas clair comment et sur quel critère le MM choisit la liste de routeurs de bordure envoyée vers le nouveau routeur de bordure membre de l'arbre. De plus, le processus de découverte du routeur de bordure que le routeur de bordure initial doit choisir comme point d'adhésion à l'arbre, n'est pas évident.

Dans MMT, nous considérons qu'un nœud de cœur[17] peut jouer le rôle d'un nœud

17. Un POP est considéré comme un nœud de cœur.

de branchement. Le NIMS peut choisir ainsi un routeur (RH, RB, ou RA) dans le POP comme étant le routeur chargé de garder les états de routage (la table de commutation) pour l'arbre *multicast* MPLS. On se retrouve dans le cas du protocole ERM.

Dans ERM, contrairement à MMT, la réservation de la bande passante pour les flux *multicast* n'est pas traitée. De plus, le stress des liens augmente puisque la duplication d'un paquet n'est permise que dans les routeurs qui n'appartiennent pas au réseau cœur. L'algorithme de construction de l'arbre *multicast* interdit les routeurs de branchement dans le réseau du cœur. Ceci amène à une augmentation du nombre de paquets envoyés par le réseau cœur. Par conséquence, les liens peuvent être vite saturés, ce qui amène le contrôleur *multicast* à refuser des demandes de construction d'arbre *multicast* à travers les liens saturés. Les caractéristiques de la solution de ERM rendent la solution non recommandée [RCT+03].

Pour les réseaux qui appliquent MMT, nous supposons que le réseau ne contient que des routeurs IP[18]. C'est le cas actuellement dans les réseaux : les réseaux Abilene ou vBNS+ changent leur routeur de cœur vers des routeurs IP. De plus, nous supposons qu'il y a un *full mesh* entre tous les routeurs de cœur du même réseau. Ce n'est pas loin d'être le cas dans les grands ISP comme GlobalCenter [XHBN00]. En effet, dans GlobalCenter, le réseau MPLS est formé de 200 routeurs. Le réseau est divisé en 9 régions. Il y a un *full mesh* entre les routeurs de chaque régions ce qui permet d'acheminer tout le trafic intra-domaine à l'intérieur d'un région. Un autre *full mesh* est fait entre les routeurs cœur de toutes les régions. Il est important de noter qu'un *full mesh* entre tous les routeurs participant au réseau MPLS est possible mais il n'était pas choisi pour des questions de performance et de facilité de gestion.

Enfin, beaucoup de routeurs de cœur déjà déployés dans l'épine dorsale, avec le temps, sont poussés vers les points d'accès aux clients. Il sera donc nécessaire pour le trafic *multicast* d'être traité au niveau des routeurs du cœur.

Les réseaux deviennent de plus en plus avancés. Dans le cas d'un réseau Abilene,

18. Cette hypothèse est relâchée dans le sous-chapitre suivant (4.6.3) avec MMT2.

un nœud du cœur contient les équipements suivants :

- Un routeur du cœur $JuniperT640$.
- Un routeur d'accès $Cisco2651$.
- Un sous-réseau $GigEvlan$ pour connecter ces routeurs.

Les nouveaux routeurs du cœur comme le $JuniperT640$ permettent même le routage *multicast* traditionnel et peuvent gérer facilement des états de routage *multicast* pour l'arbre *multicast* MPLS de MMT.

4.6.3 Le protocole MMT2

Dans ce sous-chapitre, nous supposons que quelques routeurs dans le réseau ne supportent pas le routage mixte. Nous essayons de résoudre le problème du routage mixte dans un routeur de cœur en utilisant un double niveau de labels tout en préservant le principe du protocole MMT. Le label du niveau inférieur est un label unique représentant un canal (S, G). Un label (appartenant à un intervalle de labels réservé au protocole MMT2) est attribué au canal (S, G) lors de la réception par le NIMS des messages d'adhésion pour ce canal. Ce label est le label identificateur du canal dans le domaine géré par le NIMS. Ce label peut être différent d'un domaine à un autre. Le NIMS informe tous les routeurs de branchement à propos de ce label ainsi que les labels qui correspondent aux prochains routeurs de branchement pour ce canal. Une extension du message *branch* est nécessaire pour porter les nouvelles informations. Le label qui correspond au canal (S, G) est ajouté au paquet *multicast* à l'entrée du domaine, le LSR *ingress* du domaine ajoute aussi les labels du niveau supérieur qui correspondent aux prochains routeurs de branchement pour le canal. Dans les routeurs intermédiaires qui ne sont pas routeurs de branchement, le paquet est analysé suivant le label entrant placé en haut de pile, label qui sera remplacé par un label sortant comme dans le cas du MPLS *unicast*. Lorsque le paquet arrive dans le routeur intermédiaire de branchement, le label du niveau supérieur est enlevé, le label indentificateur du canal est traité et les nouveaux labels qui correspondent aux prochains routeurs de branchement sont ajoutés (*cf.* figure 4.16). Cette opération est répétée jusqu'à l'arrivée

aux routeurs destinataires. Tous les labels sont ainsi dépilés et de nouveau le paquet est envoyé vers les routeurs *ingress* des autres domaines ou bien directement vers les destinations appartenant aux sous réseaux des routeurs *egress* destinataires.

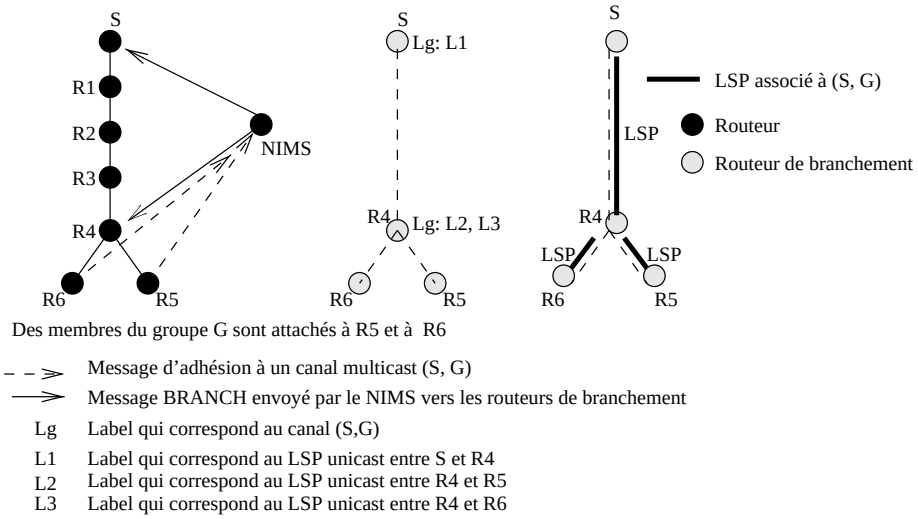


FIG. 4.16 – *La construction de l'arbre multicast MPLS avec le protocole MMT2.*

Les figures 4.17 et 4.18 représentent respectivement les nouvelles données des différents groupes dans le NIMS et les nouveaux états de routage dans les routeurs avec le protocole MMT2, pour l'exemple décrit par la figure 4.11.

GID	Membres	Bande passante demandée (Mbit/s)
G0	s : A d : D, G	1
G1	s : H d : D, G	1.5
G2	s : A d : D, G	0.2

s: source d: destinataire

GID	Routeurs de branchement	Label de canal
G0	A(C(D,G))	Lg0
G1	H(C(D,G))	Lg1
G2	A(F(D,G))	Lg2

LSPID	Groupes participants	Bande pasante résiduelle (Mbit/s)
A−B	G0, G2	...
B−C	G0, G1	0
H−E	G1	···
B−E	G2	···
E−B	G1	···
E−F	G2	···
C−D	G0, G1, G2	···
C−F	G1	···
F−C	G2	···
F−G	G0, G1, G2	···

Lg0, Lg1, Lg2 : labels correspondant aux groupes G0, G1 et G2.

FIG. 4.17 – *Les données des différents arbres dans le NIMS par le protocole MMT2.*

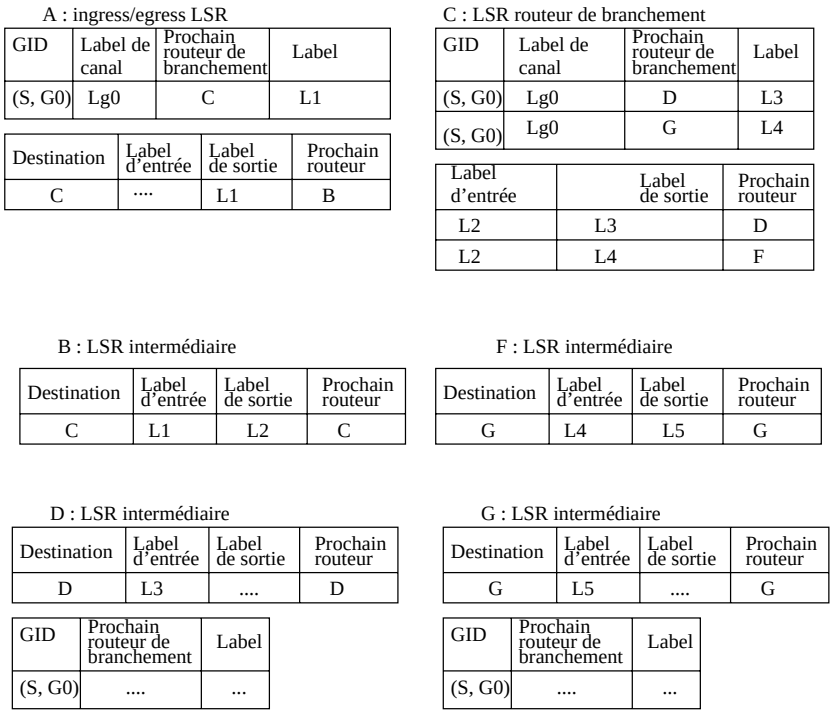


FIG. 4.18 – *Les états de routage dans les routeurs dans le protocole MMT2.*

4.6.4 Le protocole MMT2 et les arbres agrégés

Puisque le nombre de labels est limité pour un domaine, le NIMS peut refuser d'attribuer un label *multicast* à un canal si le nombre de groupes représentés par les canaux (S, G) est plus grand que le nombre de labels disponibles. Le NIMS est chargé de choisir s'il veut accepter l'attribution d'un label à un canal et de libérer dans ce cas un label déjà attribué à un autre canal. Les canaux peuvent être classés selon un ordre de priorité et ceci selon une classification de service associée à chaque canal à l'entrée du domaine.

Si le domaine est simplement un domaine de transit sans branchement, cette attribution d'un label *multicast* à un groupe *multicast* n'est pas nécessaire puisque nous n'aurons pas besoin d'un passage de la couche "Liaison de données" à la couche "Réseau" dans les routeurs du cœur.

Pour résoudre le problème de pénurie de labels, MMT2 permet de calculer uniquement les arbres agrégés. On choisit (comme *Aggregated multicast*) que deux canaux

sont associés à un même arbre agrégé dans un domaine si l'arbre calculé pour le premier canal a exactement les mêmes branches que l'arbre calculé pour le deuxième canal dans le domaine. Ceci permet au NIMS d'associer plusieurs canaux à un même arbre agrégé, d'économiser l'utilisation de labels dans le domaine et de réduire même les états de routage à conserver dans les routeurs de branchement.

4.7 Le simulateur pour le *multicast* MPLS

MNS (*MPLS Network Simulator*) [AC00] est une implémentation dans le simulateur NS des différents mécanismes de MPLS. Il s'agit essentiellement d'un module qui permet de relayer les paquets en se basant sur un label. MNS est maintenant distribué avec NS. Nous décrivons dans le paragraphe suivant l'implémentation de MPLS dans le simulateur NS.

4.7.1 L'implémentation de MPLS dans le simulateur NS

Pour simuler MPLS dans un réseau, MNS [AC00] fournit deux fonctions principales : la distribution de labels par LDP et la commutation de labels MPLS. NS est un simulateur où un nœud est représenté par un agent et par un classificateur. L'agent est l'objet chargé d'émettre et de recevoir les paquets, alors que le classificateur est l'objet responsable de classifier ces paquets. Le classificateur est chargé de déterminer si le paquet est à transmettre au prochain nœud en direction de la destination ou bien de fournir ce paquet à l'agent local si le nœud de réception est la destination de ce paquet. Par conséquent, afin de construire un nœud MPLS, un nouveau objet, appelé le classificateur MPLS, devrait être créé afin de pouvoir classifier les paquets reçus, déterminer s'ils ont un label, et les traiter également. En outre, un nouvel agent, l'agent LDP, doit être également inséré dans le nœud afin de distribuer des labels à d'autres nœuds MPLS et de construire les LSP (*cf.* figure 4.19).

Un nœud MPLS dans NS possède trois tables pour gérer les informations liées aux LSP et à la distribution de labels : PFT (*Partial Forwarding Tale*), LIB (*Label*

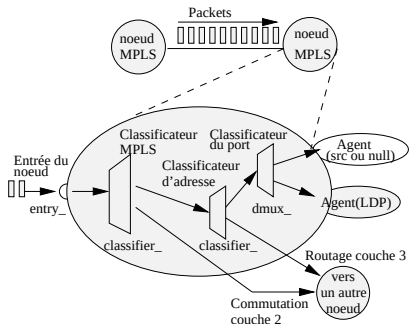


FIG. 4.19 – *L'architecture d'un nœud MPLS dans NS.*

Information Base) et ERB (*Explicit Routing information Base*). La table PFT est un sous-ensemble de la table de routage et est formée des champs FEC, PHB (*Peer-Hop Behaviour*) et LIBptr[19]. La table LIB contient l'information sur les LSP. La table ERB contient les informations sur les LSP explicites. La figure 4.20 montre la structure de ces tables et le processus de routage d'un paquet.

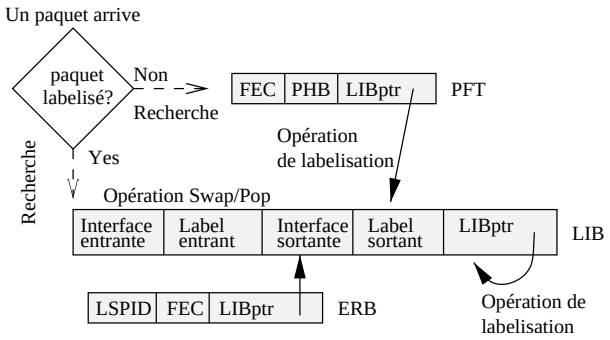


FIG. 4.20 – *La structure des tables nécessaires pour le routage d'un paquet MPLS.*

La table LIB est utilisée pour associer le couple <label entrant, interface entrante> au couple <label sortant, interface sortante>. Elle est employée quand l'opération de routage de la couche "Liaison de données" doit être exécutée : quand un paquet labelisé[20] arrive, une opération d'échange/dépilage (*swap/pop*) est exécutée et la table LIB est examinée. Si le paquet n'est pas labelisé, un label correspondant est ajouté au paquet, au moyen de la table PFT. Le nœud MPLS recherchera dans cette table une

19. Le champ LIBptr dans chaque table est un pointeur vers une entrée de LIB.
20. Le paquet possède déjà un label.

entrée où la FEC est l'adresse de destination du paquet. L'entrée résultante pourrait pointer sur une entrée dans la table LIB pour ajouter un label au paquet ou bien pointer sur NULL ce qui indique la nécessité d'utilisation du routage niveau "Réseau". La table ERB est seulement utilisée pour garder l'information sur les LSP explicites (ER-LSP). Ainsi, elle ne participe pas au routage de paquets. Dans le cas où un flux est à associer à un ER-LSP déjà établi, une nouvelle entrée qui a le même champ LIBptr que celui de l'entrée ERB correspondante devrait être ajoutée dans la table PFT.

4.7.2 La distribution de Labels

Dans MNS, la distribution de labels et la construction d'un LSP sont faites suite à un échange de messages LDP entre les agents LDP des nœuds LSR. MNS offre trois modes de distribution de labels : dirigé par les messages de contrôle, dirigé par les données et routage explicite.

Le mode dirigé par les messages de contrôle est basé sur la distribution de messages LDP entre tous les agents LDP même s'il n'y a aucune donnée à transmettre. Les LSP sont construits pour chaque FEC [21] en envoyant des messages d'association de chaque agent LDP à tous les autres, contenant la FEC avec le label correspondant qui devrait être utilisé pour la transmission de données. À la fin de cette opération, toutes les tables LIB de tous les nœuds MPLS sont remplies et les LSP sont assignés pour toutes les FEC.

Le mode dirigé par les données consiste à distribuer les messages LDP et à construire les LSP seulement pour les FEC impliquées dans la transmission de données. Par conséquent, quand un nœud souhaite transmettre des données à une FEC, il envoie une demande de construction d'un LSP à la FEC. Les premiers paquets transmis sont acheminés selon le routage de la couche "Réseau" jusqu'à ce que le LSP soit construit, alors le routage niveau "Liaison de données" est utilisé. Quand la FEC reçoit la demande, elle répond avec un message d'association en amont vers la source. Chaque

21. Dans MNS, chaque nœud dans le domaine est considéré comme étant une FEC différente.

routeur sur le chemin reçoit ce message d'association, le traite, crée un nouveau message LDP et le transmet au prochain routeur vers la source. De cette façon un LSP est construit de la source vers la destination.

Dans le mode de routage explicite, les LSP sont construits d'une manière explicite. L'utilisateur choisit explicitement les nœuds successifs du chemin que les paquets de données doivent suivre. Les messages d'association sont distribués seulement le long de ce chemin et construisent ainsi le LSP pour une FEC.

4.7.3 La commutation de labels MPLS

Quand un paquet arrive à un certain nœud, il est traité par le classificateur MPLS et acheminé à un agent local ou à un autre nœud.

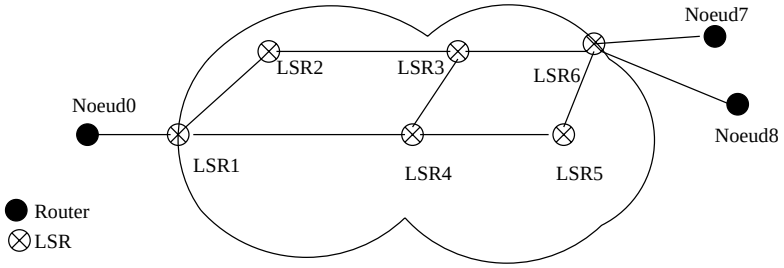


FIG. 4.21 – *L'exemple de simulation de MPLS dans NS.*

Le paquet est traité selon l'algorithme suivant :

- Le LSR *ingress* ajoute le label qui correspond à la FEC du paquet et achemine ce paquet vers le prochain LSR.
- Chaque LSR intermédiaire, examine le label dans le paquet reçu, le remplace par le label sortant et par la suite achemine le paquet selon la table LIB.
- Lorsque le paquet atteint le LSR *egress* (à la sortie du domaine MPLS), le label est enlevé et le paquet est acheminé vers sa destination selon le routage de la couche "Réseau".

Dans l'exemple présenté sur la figure 4.21, les nœuds MPLS sont $LSR1$ à $LSR6$ et forment le domaine MPLS. Les nœuds $Noeud0$, $Noeud7$, et $Noeud8$ sont des nœuds

non MPLS. Les tables de routage MPLS dans $LSR5$ sont les suivantes :

```
     ___PFT dump___ [LSR: 5]
-------------------------------------
FEC   PHB   LIBptr   AlternativePath
 4     -1     0         -1
 0     -1     1         -1
 1     -1     2         -1
 6     -1     3         -1
 7     -1     4         -1
 8     -1     5         -1
 2     -1     6         -1
 3     -1     7         -1

    ___LIB dump___ [LSR: 5]
-------------------------------------
 # iIface iLabel oIface oLabel LIBptr
 0: -1     1       4      0     -1
 1: -1     2       4      1     -1
 2: -1     3       4      2     -1
 3: -1     4       6      0     -1
 4: -1     5       6      0     -1
 5: -1     6       6      0     -1
 6: -1     7       4      5     -1
 7: -1     8       6      1     -1

     ___ERB dump___ [LSR: 5]
-------------------------------------
    FEC     LSPid      LIBptr
```

La valeur -1 pour $iIface$, $iLabel$, $oIface$ ou $oLabel$ signifie que l'interface ou le label est indéterminé. À l'entrée du domaine MPLS, la table contient souvent $iIface = -1$. Dans MNS, le test si un paquet est labelisé est fait uniquement sur $iIface$ et on ne s'intéresse pas à $iLabel$.

4.7.4 Le code MPLS dans NS et le *multicast*

Le code MPLS dans NS ne fonctionne pas avec les protocoles de routage *multicast*. En effet :

- Aucun mécanisme de distribution de labels n'existe pour les groupes *multicast*.
- Aucun duplicateur *multicast* (dans le code MPLS de NS) n'existe pour coopérer avec le classificateur MPLS.
- L'en-tête MPLS contient des pointeurs qui ne fonctionnent pas avec un réplicateur *multicast*[22].

Dans le sous-chapitre suivant, nous décrivons les modifications requises au simulateur MNS pour permettre la transmission de paquets *multicast* dans les réseaux MPLS sans mettre en application un nouveau protocole. Deux points principaux doivent être considérés : les tables d'information des nœuds MPLS et la transmission de paquet *multicast*.

4.8 L'implémentation d'un simulateur pour PIM-SM

Dans [BCJD03], nous avons présenté une simulation du protocole PIM-SM dans un domaine MPLS. PIM-SM est le protocole *multicast* le plus largement mis en application. Une plage d'adresses IP de groupe a été réservée pour des applications spécifiques et des protocoles *multicast* spécifiques (SSM) et devrait soutenir les arbres basés à la source, excluant la présence d'un point de rendez-vous et d'un arbre partagé. Notre simulateur a été conçu pour le protocole PIM-SM (source spécifique) dans les réseaux MPLS mais il peut être aussi bien adapté à d'autres protocoles. Ce simulateur est basé sur la proposition [FRR00] où un label est distribué avec les messages d'adhésion du protocole PIM-SM.

Notre objectif principal était de mettre en application la simulation avec NS de PIM-SM dans des réseaux MPLS sans modifications importantes du code de l'*unicast*

22. Un réplicateur *multicast* existe pour chaque simulation d'un protocole *multicast* dans NS.

MPLS dans NS, assurant la compatibilité entre les nœuds.

4.8.1 Les tables d'information des nœuds MPLS

Comme déjà mentionné dans la section 4.7.1, un nœud MPLS contient trois tables d'information : LIB, PFT et ERB. Pour appliquer la proposition de PIM-MPLS, une association entre le canal (S, G) et <label entrant, interface entrante> d'une part, et une association entre <label entrant, interface entrante> et plusieurs <label sortant, interface sortante> d'autre part, sont nécessaires. Les bases d'information des nœuds MPLS doivent être modifiées pour prendre cela en compte.

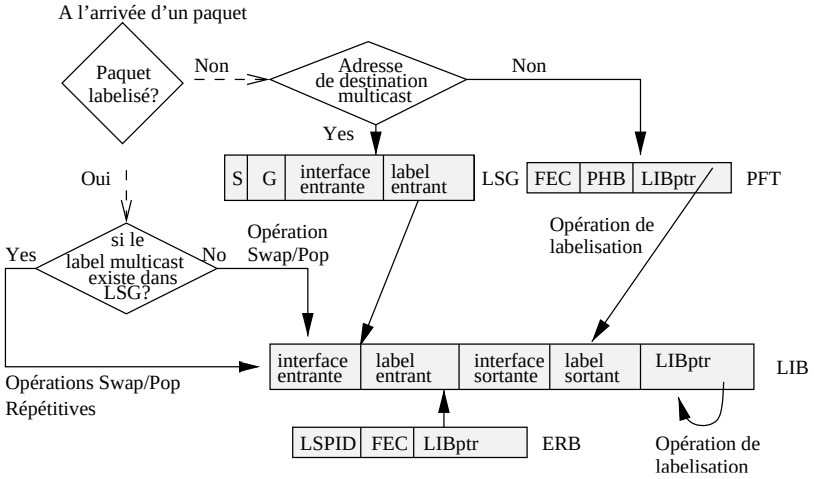


FIG. 4.22 – *La structure des tables pour la transmission de paquets dans PIM-MPLS.*

Pour la première association, la table LSG (*Label for Source and Group*) est définie. Cette table inclut quatre champs : le label entrant, l'interface entrante, la source et le groupe. Quand un nouveau membre adhère à un canal (S, G) et quand de nouveaux labels sont alloués en amont, cette table est remplie à chaque nœud. Quant à la deuxième association où un <label entrant, interface entrante> est associé à plusieurs <label sortant, interface sortante> dans un nœud de branchement, il n'y a pas besoin de créer une nouvelle table. La table LIB peut être remplie plus d'une fois du même <label entrant, interface entrante> mais avec plusieurs <label sortant, interface sortante> différents.

4.8.2 La transmission de paquets *multicast*

Les données sont transmises exactement comme des paquets *unicast* MPLS avec une seule différence aux nœuds de branchement. La procédure est la suivante : quand un paquet labelisé arrive, une recherche est faite dans la table LSG pour <label entrant, interface entrante>. Si le résultat est positif, alors le paquet labelisé est un paquet *multicast*. Notez que ceci peut être évité, mais dans ce cas le code *unicast* de MPLS devait être changé. Par conséquent, le nœud peut être de branchement et la table LIB peut contenir plus d'une entrée sortante. Dans ce cas, au lieu d'accéder à la table LIB une seule fois, une recherche répétitive pour plus d'une entrée est faite. Pour chaque entrée, une copie du paquet est créée, puis le label est permuté avec le label sortant correspondant, pour être finalement transmis à l'interface sortante. Il faut noter qu'il n'y a pas une entité spécifique qui joue le rôle d'un réplicateur de paquets dans les nœuds. Au lieu de cela, pour chaque entrée sortante dans la table LIB (pour le même label et pour la même interface entrants), une permutation de label est faite pour une copie de ce paquet, et cette copie est envoyée sur l'interface sortante. Donc la duplication d'un paquet est faite en se basant sur la table de commutation et non pas sur une information présente dans un réplicateur comme c'est le cas des simulations des protocoles *multicast* traditionnels.

4.8.3 La distribution de labels par les messages d'adhésions et de désabonnement

Les fonctions *join-group* et *prune-group* sont exécutées aux nœuds qui souhaitent adhérer à un canal (S, G) ou le quitter. L'attribution de labels est faite du nœud qui vient de joindre le canal vers la source. L'algorithme vérifie d'abord si le nœud appartient déjà à l'arbre (S, G). Si c'est le cas, alors il n'y a aucun besoin de continuer le processus d'adhésion. Dans le cas contraire, l'algorithme génère un nouveau <label entrant, interface entrante> pour ce nœud et cherche l'adresse du prochain nœud en amont vers la source. Il crée une entrée dans la table LIB avec un <label entrant, inter-

face entrante> et l'associe avec <label sortant = -1, interface sortante = -1> [23] puisque c'est le nœud destinataire. Il crée aussi une entrée dans la table LSG avec un <label entrant, interface entrante> et l'associe à (S, G). Le couple <label entrant, interface entrante> pour ce nœud est égal au couple <label sortant, interface sortante> pour le prochain nœud vers la source.

Si ce prochain nœud appartient à l'arbre (S, G), alors l'algorithme recherche l'entrée <label entrant, interface entrante> dans la table LSG (associée avec le canal (S, G)) pour ce nœud et ajoute une entrée dans la table LIB avec l'entrée <label entrant, interface entrante> associée au couple <label sortant, interface sortante> déjà calculé. Ce processus est répété à chaque nœud vers la source. Il faut noter qu'à la source il n'y a aucun besoin du couple <label entrant, interface entrante>.

Quand un nœud quitte le canal, des labels doivent être désalloués. La désallocation de labels est appliquée sur tous les nœuds sur le chemin menant à la source. Elle s'arrête dans un des deux cas suivants : quand elle atteint la source ou quand elle atteint un nœud de branchement pour le canal (S, G). Désallouer un label veut dire supprimer l'entrée (S, G) correspondante dans les tables LSG et LIB. Dans un nœud de branchement, l'algorithme supprime seulement l'entrée correspondante de la table LIB puisque le nœud de branchement a besoin de l'entrée dans LSG pour pouvoir envoyer des paquets de données aux autres branches.

4.8.4 Un exemple simple de simulation

La figure 4.23 illustre un exemple simple de réseau de simulation pour le protocole PIM-SM (le fichier MPLSPIMSMexample.tcl en TCL pour cet exemple est présenté dans l'annexe de [BJCD02]).

Prenons $LSR5$ la source, l'adresse de groupe est $G = 8$ [24] et supposons que $LSR0$ et $LSR1$ sont destinataires du canal (S, G) avant que la source ne commence sa trans-

23. Rappelons que la valeur -1 pour $iIface$, $iLabel$, $oIface$ ou $oLabel$ signifie que l'interface ou le label est indéterminé.

24. Cette valeur est choisie au hasard. Le simulateur NS le permet.

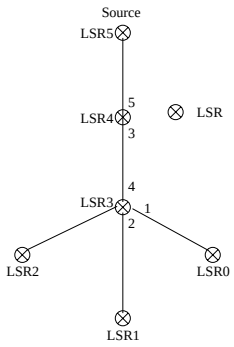


FIG. 4.23 – *Un exemple d'un réseau pour le simulateur de PIM-MPLS.*

mission. L'allocation de labels MPLS est faite automatiquement par MNS, et donc toutes les tables d'information sont remplies. Notons que la table LIB de l'exemple est employée pour l'*unicast* et le *multicast*. Afin de voir comment les labels sont alloués à chaque nœud, nous considérerons les deux nœuds $LSR4$ et $LSR3$ et nous vérifions comment les tables LIB et LSG sont remplies pour chaque nœud. Au nœud $LSR4$, la table LSG contient l'entrée $(S = 5, G = 8)$ qui est associée au couple <label entrant=6, interface entrante=5>. Dans la table LIB, l'entrée <label entrant=6, interface entrante=5> est associée au couple <label sortant=6, interface sortante=3>. Ceci est montré dans les deux tables suivantes :

```
     ___LIB dump___ [LSR: 4]
--------------------------------------
#  iIface iLabel oIface oLabel LIBptr
0:  -1      1      3      0      -1
1:  -1      2      5      0      -1
2:  -1      3      3      2      -1
3:  -1      4      3      3      -1
4:  -1      5      3      4      -1
5:   5      6      3      6      -1

    ___LSG dump___ [LSR: 4]
--------------------------------------
#  iIface iLabel Source Group
0:   5      6      5      8
```

Tandis que pour $LSR3$, qui est un nœud de branchement, la table LSG contient l'entrée $(S = 5, G = 8)$ qui est associée au couple <label entrant=6, interface entrante=4>, et la table LIB associe l'entrée <label entrant=6, interface entrante=4> aux deux sorties : <label sortant=1, interface sortante=1> et <label sortant=1, interface sortante=2>. Ceci est montré dans les deux tables suivantes :

```
     ___LIB dump___ [LSR: 3]
-------------------------------------
#  iIface iLabel oIface oLabel LIBptr
0:   -1     1      4      0     -1
1:   -1     2      0      0     -1
2:   -1     3      1      0     -1
3:   -1     4      2      0     -1
4:   -1     5      4      2     -1
6:    4     6      1      1     -1
7:    4     6      2      1     -1

    ___LSG dump___ [LSR: 3]
-------------------------------------
#  iIface iLabel Source Group
0:   4      6      5      8
```

4.9 La simulation du protocole MMT

L'implémentation de MMT dans NS diffère de PIM-MPLS puisque les mécanismes des deux protocoles sont différents. Notre objectif principal était de mettre en application MMT sans modifications importantes du code effectuant le réglage *unicast* MPLS existant dans NS. Nous avons utilisé le simulateur pour PIM-MPLS en introduisant des changements pour que le code s'adapte au protocole MMT.

4.9.1 Les tables d'information des nœuds MPLS

Rappelons qu'un nœud MPLS contient trois tables d'information : LIB, PFT, et ERB. Pour appliquer la proposition MMT, une association entre un canal (S, G) et

plus d'une FEC d'une part, et l'association de chaque FEC à un <label entrant, interface entrante> et donc à un <label sortant, interface sortante> d'autre part, sont nécessaires. La base d'information des nœuds MPLS doit être modifiée comme indiqué sur la figure 4.24.

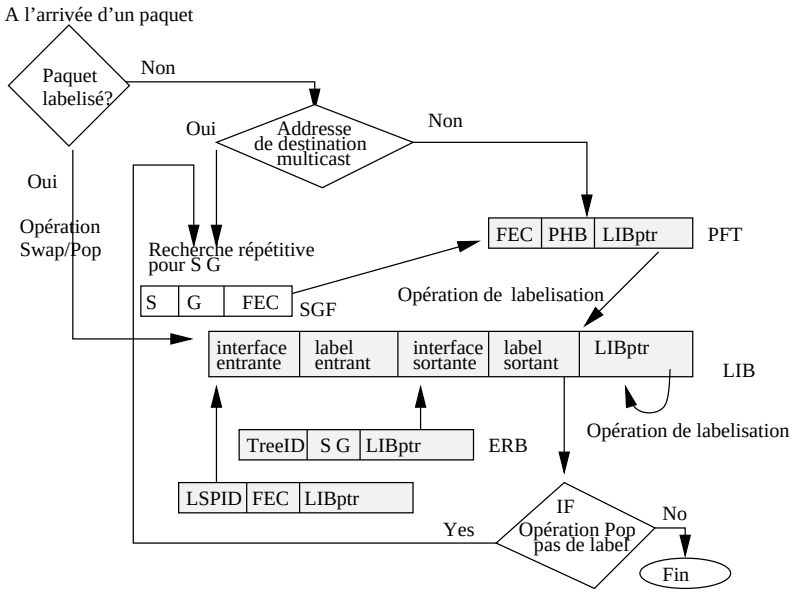


FIG. 4.24 – *La structure des tables pour la transmission de paquets dans MMT.*

Pour la première association, la table SGF (*Source Group FEC*) est définie. Cette table existe seulement dans les routeurs de branchement et inclut trois champs : source, groupe et FEC. Un canal (S, G) peut être associé à plusieurs FEC dans les routeurs de branchement. Dans la table LIB, chaque FEC est associée exactement à un <label entrant, interface entrante>. Quant à la deuxième association, chaque <label entrant, interface entrante> est associé à exactement un <label sortant, interface sortante>. La table LIB reste inchangée avec un <label entrant, interface entrante> pour un <label sortant, interface sortante>.

4.9.2 La transmission des paquets *multicast*

Les données sont transmises exactement comme dans le relayage *unicast* MPLS avec seulement une différence aux nœuds de branchement. La procédure est la sui-

vante : quand un paquet labelisé arrive, une opération *swap/pop* est exécutée et la table LIB est examinée. Si en raison d'une opération de dépilage (*pop*) le paquet reste sans label, une recherche est faite dans la table SGF pour attribuer le paquet à une ou plusieurs FEC. Pour chaque FEC, une copie du paquet est créée, et le label est échangé avec le label sortant correspondant, puis transmis sur l'interface sortante.

Nous allons évaluer le protocole MMT en utilisant ce simulateur. Ce simulateur est disponible sur Internet et peut être utilisé par tous les chercheurs pour vérifier des mécanismes de routage *multicast* dans les domaines MPLS.

4.10 Évaluation du protocole MMT

Dans cette section, le protocole MMT est évalué en terme de résistance au facteur d'échelle (la diminution en taille des tables de routage *multicast*) et d'efficacité (le coût de l'arbre *multicast* et le temps de traitement des en-têtes *multicast* dans les routeurs).

Notre approche présente une certaine efficacité vis-à-vis des autres propositions *multicast* (DVMRP, PIM-SM, MOSPF) et *multicast* MPLS (PIM-MPLS, *Aggregated multicast*) puisque d'une part nous utilisons l'arbre des meilleurs chemins pour acheminer les paquets et d'autre part nous utilisons la technique de commutation rapide de MPLS dans les routeurs.

Nous allons comparer MMT et son extention le protocole MMT2 [25] avec les principaux protocoles de routage MPLS *multicast*, notamment PIM-MPLS [FRR00] et *Aggregated Multicast* [FCGF01]. Dans nos simulations PIM-MPLS se réfère au simulateur décrit dans [BJCD02]. Rappelons que dans [BJCD02] nous avons présenté un simulateur pour le routage *multicast* dans un réseau MPLS où nous avons choisi PIM-SM (avec un arbre spécifique enraciné à la source) comme protocole de routage *multicast*. Nous simulons le protocole MMT avec NS pour valider le comportement de base de l'approche et son efficacité à réduire le nombre d'états de routage, à diminuer le temps de traitement d'un paquet et à baisser le coût des arbres.

25. Nous ne considérons que les arbres agrégés.

4.10.1 La diminution en taille des tables de routage *multicast*

Dans notre approche, le NIMS calcule les arbres *multicast* dans le domaine et informe avec des messages *branch* les différents routeurs de branchement à propos de leurs prochains routeurs de branchement (ou bien de leurs prochains labels avec les interfaces de sortie) sur chacun des arbres *multicast*. À la réception du message *branch*, le routeur de branchement crée un état de routage *multicast* pour le canal *multicast*. Puisque notre approche ne stocke des états de routage que dans les routeurs de branchement, une réduction de taille des tables de routage est évidente : cette réduction de taille des tables de routage a été démontrée pour le protocole SEM dans le chapitre 3 et peut être également démontrée pour le protocole MMT.

Un domaine MPLS peut servir uniquement comme domaine de transit pour un canal. Aucune source ni destinataire ne sont présents dans le domaine pour ce canal. Un arbre ayant un ou plusieurs nœuds de branchement dans le domaine est appelé BT (*Branched Tree*). Un arbre avec un seul chemin dans le domaine où aucun nœud de branchement ne figure dans l'arbre est appelé OPT (*One Path Tree*). Le tableau 4.1 représente le nombre moyen d'états de routage dans les routeurs dans les deux cas : arbres BT de transit avec nœuds de branchement et arbres OPT de transit sans nœud de branchement.

Protocole/ Arbre	PIM-MPLS	*Aggregated multicast*	MMT	MMT2
BT	$\bar{n_T} * T$	$\bar{n_{T_{aggr}}} * T_{aggr}$	$\bar{n_{MMT}} * T$	$\bar{n_{MMT-aggr}} * T_{aggr}$
OPT	$\bar{n_T} * T$	$\bar{n_{T_{aggr}}} * T_{aggr}$	$2 * T$	$2 * T_{aggr}$

TAB. 4.1 – *Le nombre moyen d'états de routage dans les routeurs.*

T représente le nombre d'arbres *multicast* (ou bien le nombre de canaux *multicast*) présents dans le réseau, $\bar{n_{MMT}}$ le nombre moyen de routeurs de branchement sur les arbres en utilisant le protocole MMT, $\bar{n_{MMT-aggr}}$ [26] le nombre moyen de routeurs de branchement sur les arbres agrégés en utilisant le protocole MMT2, $\bar{n_T}$ le nombre

26. Dans le reste de cette évaluation, nous considérons que $\bar{n_{MMT}}$ et $\bar{n_{MMT-aggr}}$ incluent aussi les états présents dans les sources et les destinations.

moyen de routeurs sur les arbres *multicast* en utilisant un protocole de routage *multicast* traditionnel, T_{aggr} est le nombre d'arbres agrégés de *Aggregated Multicast*, $n_{\bar{T_{aggr}}}$ est le nombre moyen de routeurs sur l'arbre agrégé. Ces valeurs satisfont la relation suivante :

$$T \geq T_{aggr},$$

$$\bar{n_T} \geq \bar{n_{MMT}}, \bar{n_{MMT-aggr}},$$

$$\bar{n_{T_{aggr}}} \geq \bar{n_{MMT-aggr}},$$

et

$$\bar{n_T}, \bar{n_{T_{aggr}}} \geq 2.$$

Il est donc évident d'après le tableau 4.1 que MMT présente de meilleures performances que PIM-MPLS. Dans le cas d'arbres OPT, le nombre d'états de routage pour MMT dans les routeurs intermédiaires dans le réseau est égal à 0[27] et donc présente des avantages par rapport à *aggregated multicast*. Par contre, dans le cas d'arbres BT, le nombre d'états de routage pour le protocole MMT n'est pas toujours inférieur à celui d'*Aggregated Multicast*. En effet, le protocole MMT présente de meilleures performances sur *Aggregated Multicast* seulement dans le cas où $\bar{n_{MMT}} * T < \bar{n_{T_{aggr}}} * T_{aggr}$ donc lorsque la formule suivante est respectée :

$$\bar{n_{T_{aggr}}} > \bar{n_{MMT}} * \frac{T}{T_{aggr}}. \tag{4.1}$$

Or, ceci est possible dans beaucoup de cas. Prenons l'exemple suivant :

27. Si on ne prend pas en compte les états de routage dans les deux routeurs de bordure source et destination dans le réseau.

D'après [CKM+03, CKM+02], le réseau vBNS (*cf.* figure 4.25) est formé de 43 routeurs dont 16 sont des routeurs cœurs. Les 43 routeurs du réseau participent à la diffusion du trafic *multicast*.

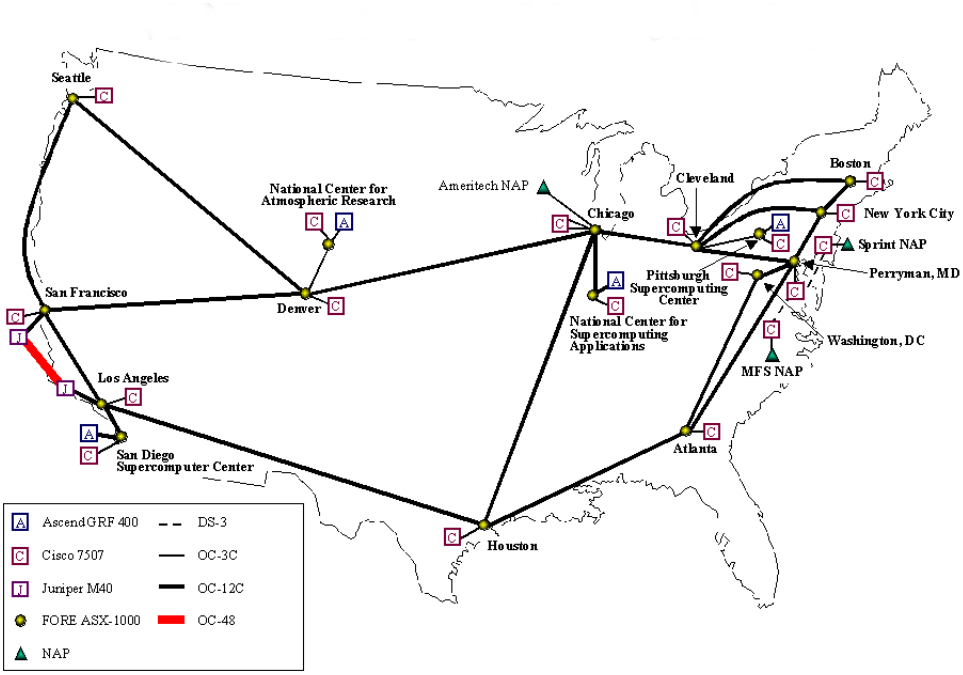


FIG. 4.25 – *Le réseau vBNS.*

Dans l'exemple présenté [CKM+03], un ensemble de 2500 canaux *multicast* sont présents dans le réseau. Ces 2500 arbres sont regroupés en 1150 arbres agrégés. Donc d'après l'équation 4.1, $n_{\bar{T_{aggr}}}$ doit être plus grand que $n_{\bar{MMT}} * \frac{2500}{1150} \approx 2.2 * n_{\bar{MMT}}$ pour que MMT soit meilleur. Comme nous l'avons présenté dans le chapitre 3, le nombre de routeurs de branchement sur un arbre est très petit (de l'ordre de 8% du nombre de routeurs de l'arbre *multicast*). Nous déduisons que $n_{\bar{MMT}} \approx 4$. Si la valeur de $n_{\bar{T_{aggr}}}$ dépasse $n_{\bar{MMT}} * \frac{T}{Taggr} \approx 9$, MMT a de meilleurs performances. Il est donc possible que MMT réduise mieux que *Aggregated Multicast* la taille des tables de routage *multicast* dans les routeurs. Finalement, d'après le tableau 4.1, MMT2 présente des meilleurs performances par rapport à tous les autres protocoles.

Pour valider notre évaluation, nous considérons les 3 réseaux : MCI[28] (18 nœuds dans le cœur du réseau), Abilene (11 nœuds dans le cœur du réseau) et NSFNET (14 nœuds dans le cœur du réseau), et nous calculons le nombre d'arbres agrégés pour 5000 arbres (*cf.* figure 4.26). Nous considérons qu'à chaque nœud du cœur du réseau est attaché un seul nœud qui joue le rôle d'une source ou d'un destinataire. Le tableau 4.2 représente le nombre de membres pour chaque chaque groupe.

Réseau	Nombre de destinataires
Abilene	entre 2 à 10
NSFNET	entre 2 à 12
MCI	entre 2 à 16

TAB. 4.2 – *Le nombre de membres pour chaque arbre dans les réseaux : Abilene, NSFNET et MCI.*

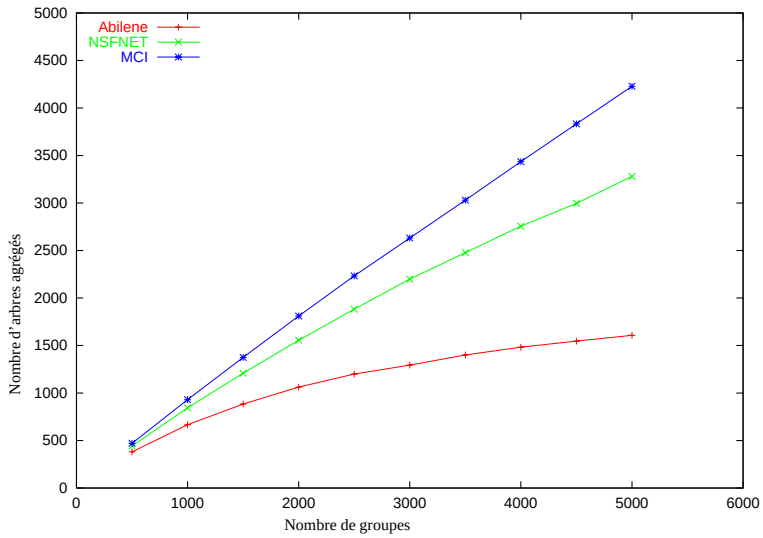


FIG. 4.26 – *Le nombre d'arbres agrégés pour 5000 groupes dans les 3 réseaux : Abilene, NSFNET et MCI.*

Les figures 4.27, 4.28 et 4.29 représentent respectivement le nombre moyen d'états de routage dans un routeur pour les réseaux Abilene, MCI et NSFNET.

28. Rappelons que MCI a développé le réseau vBNS+.

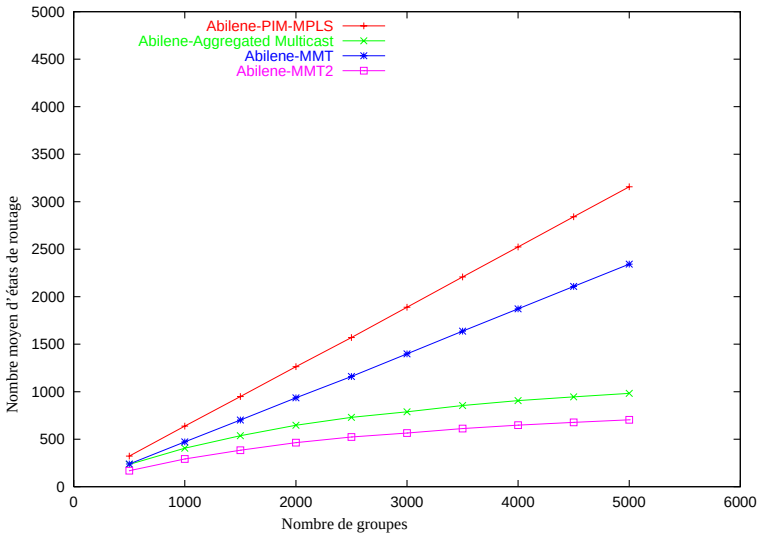


FIG. 4.27 – *Le nombre moyen d'états de routage dans un routeur dans le réseau Abilene pour les différents protocoles.*

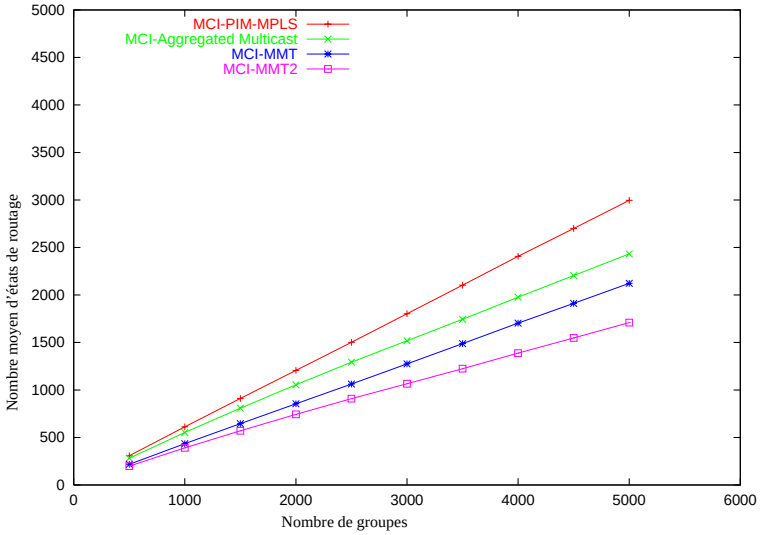


FIG. 4.28 – *Le nombre moyen d'états de routage dans un routeur dans le réseau MCI pour les différents protocoles.*

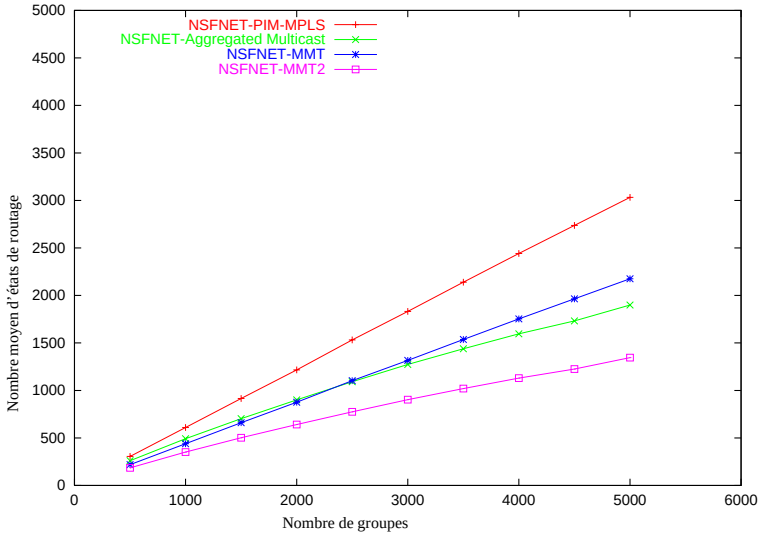


FIG. 4.29 – *Le nombre moyen d'états de routage dans un routeur dans le réseau NSFNET pour les différents protocoles.*

Nous remarquons que le protocole MMT2 présente des avantages sur tous les autres protocoles. Nous remarquons aussi que PIM-MPLS présente les plus mauvais résultats. Pour MMT et *Aggregated multicast*, nous remarquons que MMT présente des avantages sur *Aggregated multicast* dans le réseau MCI mais il est très mauvais avec le réseau Abilene. À partir d'un certain nombre d'arbres, *Aggregated multicast* a plus d'avantages sur MMT dans le réseau NSFNET. En effet, le réseau Abilene contient seulement 11 nœud : si le nombre de membres dans un groupe est grand, alors tous les routeurs du cœur sont considérés comme routeurs de branchement. Notons que nous considérons les états de routage dans la source et dans tous les destinataires. Si le nombre de membres dans les groupes est petit, le rapport $\frac{T}{T_{aggr}}$ devient grand et donc le protocole MMT ne convient pas à ce genre de topologie. Dans tous les cas, le protocole MMT2 réduit la taille des tables de routage et présente de nombreux avantages.

4.10.2 Le temps de traitement des en-têtes *multicast* dans les routeurs

Le délai d'établissement de l'arbre et le temps de transmission d'un paquet sont deux paramètres importants à étudier.

Dans les trois protocoles MMT, MMT2 et *Aggregated multicast*, une entité centrale reçoit les messages d'adhésion aux canaux, calcule l'arbre pour chaque canal et déclenche l'association entre labels et canaux. Le délai d'établissement de l'arbre est donc le même pour les trois protocoles. Mais un avantage pour MMT et MMT2 est que les LSP *unicast* sont utilisés pour le trafic *multicast*. Notons que le calcul d'un million d'arbres dans une entité centrale peut prendre un temps de calcul de l'ordre de 43 secondes[29].

En revanche, dans le cas d'un protocole de routage *multicast* traditionnel, chaque routeur entre l'*ingress* et l'*egress* doit contenir un état de routage pour chaque arbre *multicast*. Considérons $t_{Trait}(Ri)$, qu'on appelle temps de traitement, le temps nécessaire pour traiter un paquet dans un routeur Ri et par la suite le retransmettre vers l'interface de sortie. Nous allons comparer le temps de traitement global $t_{Traitg} = \Sigma t_{Trait}(Ri)$ d'un paquet pour les protocoles PIM-MPLS, *Aggregated multicast*, MMT et MMT2.

Le temps de traitement d'un paquet, t_{Trait}, peut être calculé par la formule suivante :

$$t_{Traitg} \quad = \quad \Sigma t_{Trait}(Ri) = t_L + t_{FA} + t_R,$$

où t_L est le temps de transmission d'un paquet sur les liens entre la source et le destinataire sur l'arbre *multicast*, t_{FA} est le temps d'attente du paquet dans la file d'attente des routeurs, et t_R est le temps de traitement du paquet dans les routeurs. Posons $A = t_L + t_{FA}$, c'est une constante qui ne change pas avec les différents protocoles[30].

29. L'algorithme d'association d'un canal à un arbre a été exécuté sur un Linux $P4\ 2.4Ghz$.
30. Réseau avec des liens symétriques.

PIM-MPLS. Nous pouvons constater que $t_R = \bar{n_T} * t_{MPLS\ \bar{}\ multicast}$ d'où :

$$t_{Traitg} = A + \bar{n_T} * t_{MPLS\ \bar{}\ multicast},$$

où $\bar{n_T}$ est le nombre moyen de routeurs sur un arbre *multicast* et $t_{MPLS\ \bar{}\ multicast}$ est le temps moyen pour parcourir la table de labels. $t_{MPLS\ \bar{}\ multicast}$ dépend du nombre d'arbres passants par un routeur.

MMT. Dans MMT, seuls les routeurs de branchement gardent les états de routage *multicast*. Dans ces routeurs, le protocole MMT cherche à trouver la FEC correspondante dans la table de labels. Tous les autres routeurs sur l'arbre utilisent les tables de routage MPLS *unicast* pour acheminer les paquets. D'où :

$$t_{Traitg} = A + (\bar{n_T} - n_{\bar{M}MT}) * t_{MPLS\ \bar{}\ unicast} + n_{\bar{M}MT} * (t_{MPLS\ \bar{}\ multicast}).$$

$t_{MPLS\ \bar{}\ unicast}$ est le temps moyen pour parcourir la table de labels pour l'*unicast*.

$$\begin{aligned}
\delta_{MMT,PIM-MPLS} &= \bar{n_T} * (t_{MPLS\ \bar{}\ unicast} \\
&- t_{MPLS\ \bar{}\ multicast}) + n_{\bar{M}MT} * (t_{MPLS\ \bar{}\ multicast} - t_{MPLS\ \bar{}\ unicast}) \\
&= (\bar{n_T} - n_{\bar{M}MT}) * (t_{MPLS\ \bar{}\ unicast} - t_{MPLS\ \bar{}\ multicast}).
\end{aligned}$$

La valeur de $n_{\bar{M}MT}$ est plus petite en général que la valeur de $\bar{n_T}$. De plus, avec l'augmentation du nombre de canaux, la valeur de $t_{MPLS\ \bar{}\ multicast}$ croît aussi. La valeur de $t_{MPLS\ \bar{}\ unicast}$ devient aussi plus petite que la valeur de $t_{MPLS\ \bar{}\ multicast}$ et $\delta_{MMT,PIM-MPLS}$ prend alors des valeurs négatives. Ce qui nous permet de conclure que le protocole MMT présente des avantages sur le protocole PIM-MPLS et tous les autres protocoles utilisant le même type de construction d'arbres *multicast* MPLS.

Rappelons que d'après [JNTPTR03], un routeur Juniper T640 peut traiter un paquet en $10^{-9}s$ et le gain en temps de traitement d'un paquet se traduit en un débit plus élevé

et une capacité à acheminer un nombre plus élevé de paquets.

Aggregated multicast. Comme dans le cas de PIM-MPLS nous pouvons constater que :

$$t_R = n_{\bar{T_{aggr}}} * t_{\bar{MPLS}\ aggr},$$

d'où :

$$t_{Traitg} = A + n_{\bar{T_{aggr}}} * t_{\bar{MPLS}\ aggr},$$

où $t_{\bar{MPLS}\ aggr}$ est le temps moyen pour parcourir la table de labels présente dans les routeurs après utilisation du protocole *Aggregated multicast*. Nous obtenons ainsi :

$$\begin{aligned} \delta_{MMT,Aggregated\ multicast} &= (\bar{n_T} - n_{\bar{MMT}}) * t_{\bar{MPLS}\ unicast} \\ &+ n_{\bar{MMT}} * t_{\bar{MPLS}\ multicast} \\ &- n_{\bar{T_{aggr}}} * t_{\bar{MPLS}\ aggr}. \end{aligned}$$

Il n'est pas facile de faire des approximations avec cette formule. En effet, la recherche dans une table de routage n'est pas linéaire : elle peut être parfois logarithmique avec l'utilisation des différentes techniques de recherche [WVTP01]. La valeur $t_{\bar{MPLS}\ aggr}$ dépend aussi du taux de réduction des arbres *multicast*.

Par contre, en comparaison avec le protocole MMT2, il est facile de remarquer[31] que $\delta_{MMT2,Aggregated\ multicast}$ prend souvent des valeurs négatives, ce qui nous permet de conclure que le protocole MMT2 présente des avantages sur le protocole *Aggregated multicast* en terme de temps de traitement d'un paquet.

31. Puisque $(n_{\bar{T_{aggr}}} - n_{\bar{MMT}-aggr})$ est toujours ≥ 0 et $(t_{\bar{MPLS}\ unicast} - t_{\bar{MPLS}\ aggr})$ est souvent ≤ 0 (avec la croissance de nombre de canaux *multicast* dans le réseau).

$$\begin{aligned}\delta_{MMT2,Aggregated\ multicast} &= (n_{\bar{T_{aggr}}} - n_{\bar{MMT-aggr}}) * t_{\bar{MPLS}\ unicast} \\ &+ n_{\bar{MMT_{aggr}}} * t_{\bar{MPLS}\ aggr} \\ &- n_{\bar{T_{aggr}}} * t_{\bar{MPLS}\ aggr} \\ &= (n_{\bar{T_{aggr}}} - n_{\bar{MMT-aggr}}) * (t_{\bar{MPLS}\ unicast} - t_{\bar{MPLS}\ aggr}).\end{aligned}$$

Remarquons qu'il est très difficile de simuler les valeurs exactes du temps de traitement global d'un paquet *multicast*. En effet, ce traitement dépend de la taille de la table de routage et de la technique de recherche utilisée pour trouver une entrée dans cette table . Nous contentons donc de faire la comparaison simple entre PIM-SM et PIM-MPLS. En effet, d'après [ETM], MPLS peut réduire 25% le temps de traitement d'un paquet et puisque les deux protocoles PIM-SM et PIM-MPLS ont exactement le même nombre d'états de routage, la comparaison devient facile en supposant que le temps de traitement d'un paquet *unicast* et *multicast* est le même[32].

La figure 4.30 représente respectivement les délais de bout en bout minimum, moyen et maximum des arbres lorsque le nombre de destinataires varie de 2 à 16 pour les deux protocoles PIM-SM et PIM-MPLS dans le réseau MCI. Nous avons réalisé 100 simulations pour chaque valeur de 2 à 16. La valeur présentée dans les graphes est la valeur moyenne pour les 100 simulations. nous déduisons que MPLS permet de réduire le délai de bout en bout pour les différentes destinations.

4.10.3 Le coût de l'arbre

Les protocoles MMT, MMT2 utilisent l'arbre des meilleurs chemins. En l'absence de contraintes, l'arbre des meilleurs chemins est l'arbre des plus courts chemins. *Aggregated Multicast* utilise l'arbre des plus courts chemins tandis que le protocole PIM-MPLS utilise l'arbre des plus courts chemins inverses mais dans les deux cas une seule

32. Nous prenons la valeur de $10^{-9}s$ pour un Juniper T640. Prendre une autre valeur ne change pas la qualité des résultats.

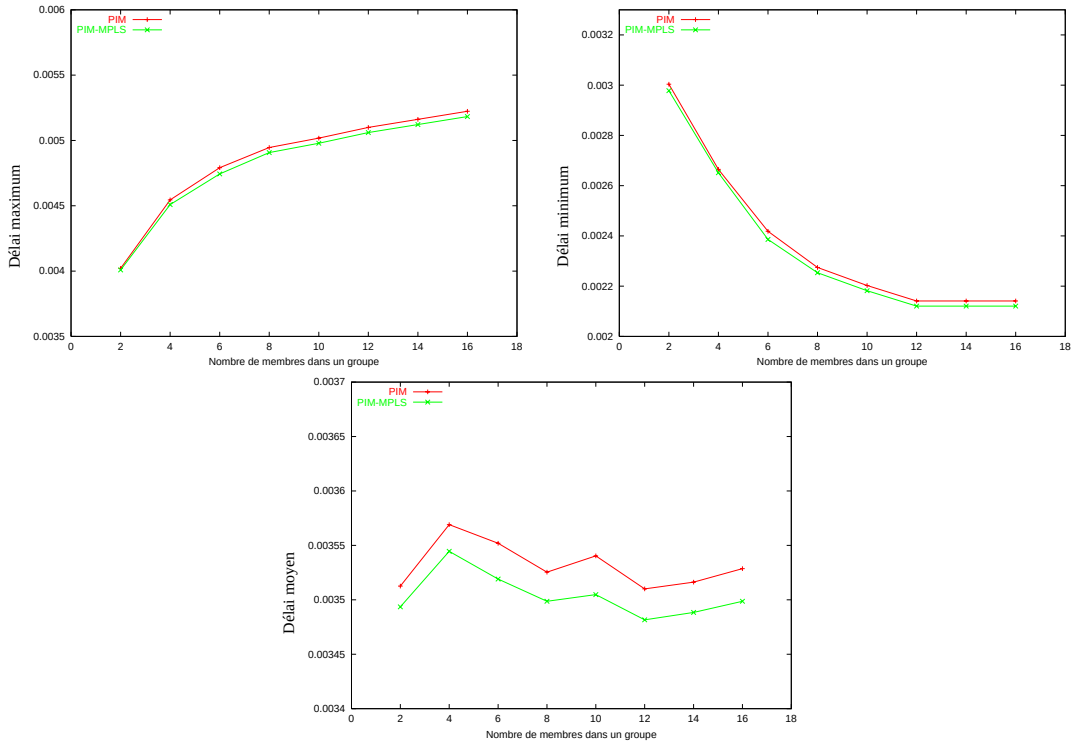


FIG. 4.30 – *Le délai global (minimum, maximun et moyen) d'un paquet pour les protocoles PIM-SM et PIM-MPLS dans le réseau MCI.*

copie d'un paquet est transmise sur un lien. Remarquons que l'arbre des plus courts chemins est identique à l'arbre des plus courts chemins inverses si on considère que le réseau est symétrique (ce n'est pas le cas comme nous l'avons déjà vu dans le chapitre 3).

Nous considérons la topologie de MCI[33] et nous utilisons le même scénario utilisé dans le sous-chapitre 3.5.2. Ayant identifié le coût de l'arbre comme étant le coût moyen des chemins de la source vers tous les destinataires, la figure 4.31 représente le coût moyen de l'arbre construit par les protocoles MMT[34] et PIM-MPLS pour le réseau MCI. Nous remarquons que le coût de l'arbre est moins élevé avec le protocole MMT qu'avec le protocole PIM-MPLS qui utilise l'arbre des plus courts chemins

33. Au lien $< n1, n2 >$ qui connecte les nœuds $n1$ et $n2$ sont associés deux coûts, $n1$-$n2$ et $n2$-$n1$ choisis aléatoirement dans l'intervalle $[1, 10]$.

34. Les protocoles MMT2 et *Aggregated Multicast* offrent des résultats identiques au protocole MMT.

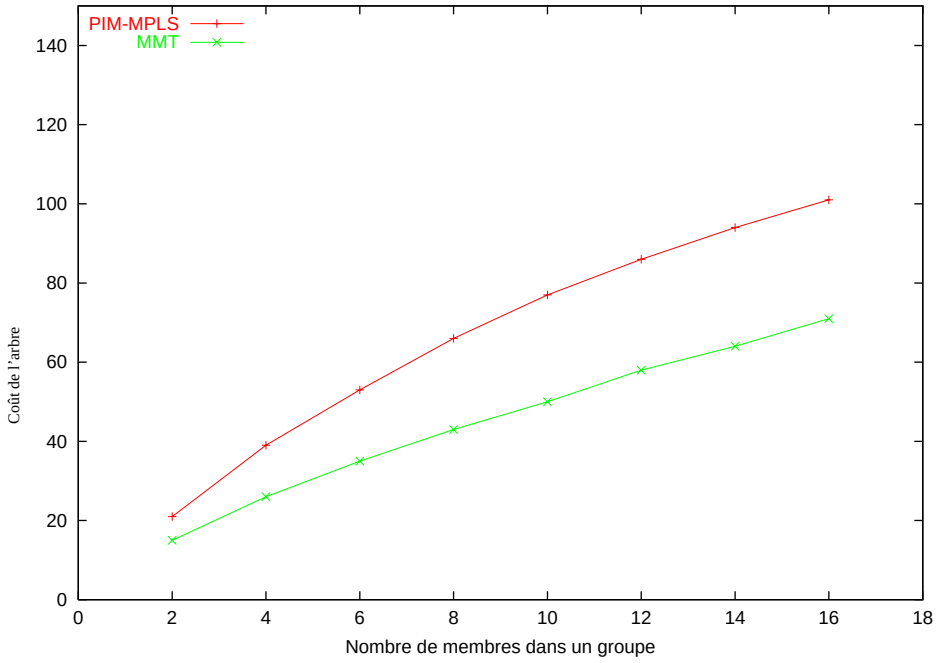


FIG. 4.31 – *Le coût de l'arbre construit par les protocoles MMT et PIM-MPLS pour le réseau MCI.*

inverses.

4.11 Conclusion

Dans ce chapitre, nous avons proposé une nouvelle approche pour construire des arbres *multicast* dans les réseaux MPLS. D'abord, nous avons présenté l'ingénierie de trafic et les avantages de la commutation de labels avec MPLS. Nous avons défini l'ingénierie de trafic *multicast* et sa particularité par rapport à l'ingénierie de trafic *unicast*. Nous avons étudié la difficulté de combiner le *multicast* et MPLS dans un réseau. Nous avons décrit les propositions MPLS pour l'ingénierie de trafic *multicast* et nous avons justifié la nécessité de définir un nouveau protocole. Par la suite nous avons proposé le protocole MMT (*MPLS Multicast Tree*), qui utilise les LSP MPLS entre les nœuds de branchement de l'arbre *multicast* afin d'enlever les états de routage dans les routeurs intermédiaires et d'augmenter la résistance au facteur d'échelle.

Dans notre approche, seuls les routeurs de branchement ont besoin de mémoriser les états de routage pour un canal *multicast*. Des tunnels *unicast* MPLS sont utilisés entre les routeurs de branchement de l'arbre *multicast*. En utilisant cette méthode, nous réduisons la quantité d'information à mémoriser dans les routeurs et une certaine résistance au facteur d'échelle est ainsi assurée. Notre approche présente une certaine efficacité vis-à-vis des autres propositions *multicast* (DVMRP, PIM-SM, MOSPF) et *multicast* MPLS (PIM-MPLS, *Aggregated multicast*) puisque d'une part nous utilisons l'arbre des meilleurs chemins (qui coïncide avec l'arbre des plus courts chemins en absence des contraintes d'ingénierie de trafic) pour acheminer les paquets et d'autre part nous utilisons la technique de commutation rapide de MPLS dans les routeurs. C'est une entité centrale (le NIMS) qui calcule les chemins entre les routeurs de branchement. De plus, le problème d'agrégation des adresses IP *multicast* peut être transformé en une simple agrégation des labels.

Nous avons présenté une extension du protocole MMT : le protocole MMT2. MMT2 ajoute des améliorations au protocole MMT et a été proposé essentiellement pour résoudre le problème du routage mixte de la couche "Réseau" et la couche "Liaison des données" dans un routeur de cœur du réseau. MMT2 utilise un double niveau de labels tout en préservant le principe du protocole MMT. Le label du niveau inférieur est un label unique représentant un canal (S, G). Le label qui correspond au canal (S, G) est ajouté au paquet *multicast* à l'entrée du domaine, le LSR *ingress* du domaine ajoute aussi les labels du niveau supérieur qui correspondent aux prochains routeurs de branchement pour le canal. Dans les routeurs intermédiaires qui ne sont pas routeurs de branchement, le paquet est analysé suivant le label entrant placé en haut de pile, label qui sera remplacé par un label sortant comme dans le cas du MPLS *unicast*.

MMT2 permet d'utiliser des arbres agrégés. Deux canaux sont associés à un même arbre agrégé dans un domaine si l'arbre calculé pour le premier canal a exactement les mêmes branches que l'arbre calculé pour le deuxième canal dans le domaine. Ceci permet à MMT2 d'associer plusieurs canaux à un même arbre agrégé, d'économiser l'utilisation de labels dans le domaine et de réduire même les états de routage à conser-

ver dans les routeurs de branchement.

Nous avons évalué MMT et MMT2 en terme de résistance au facteur d'échelle (la diminution en taille des tables de routage) et d'efficacité. Nous avons présenté un simulateur pour le trafic *multicast* dans un domaine MPLS et nous avons discuté de la mise en application de MMT dans ce simulateur. Ce simulateur de *multicast* MPLS développé au cours de cette thèse est à notre connaissance le seul de ce type et peut être un bon outil pour de futures recherches sur le *multicast* MPLS. Les résultats de simulation valident notre évaluation : nous remarquons une diminution en taille des tables de routage *multicast* par rapport aux autres approches *multicast* MPLS. Nous remarquons aussi un temps de traitement rapide d'un paquet dû à l'utilisation de la technique de commutation de labels MPLS dans les routeurs. Nous validons le coût faible réalisé par MMT par rapport aux protocoles qui utilisent l'arbre des plus courts chemins inverses. Nous concluons finalement que le protocole MMT semble prometteur et bien adapté à une éventuelle implémentation de l'ingénierie de trafic *multicast* dans le réseau.

Conclusion

Contribution

Nous avons présenté dans ce mémoire deux aspects du routage *multicast* : la gestion des petits groupes et l'ingénierie de trafic. L'étude des propositions actuelles de protocoles, leurs problèmes de résistance au facteur d'échelle et la surchage globale induite dans le réseau, ont mis en évidence la complexité du routage *multicast* dans Internet et le fait qu'un unique protocole de routage *multicast* est incapable de servir les différents types d'applications *multicast*. Le nombre d'applications *multicast* ayant souvent des exigences multiples voire contradictoires ne cesse de croître. Les protocoles de routage *multicast* doivent présenter une certaine flexibilité selon les besoins des applications. Dans un réseau où il y a un très grand nombre de groupes *multicast* de petite taille (*SGM : Small Group Multicast*) dont les destinataires sont largement dispersés, le modèle de routage *multicast* traditionnel ne convient pas. Par ailleurs, dans un réseau où les protocoles de routage *multicast* conviennent bien aux applications, l'ingénierie de trafic est nécessaire.

Nous avons mis en relation ces deux aspects du routage *multicast* et nous définissons de nouveaux protocoles plus adaptés.

Dans la première partie de cette thèse, nous avons proposé le protocole GXcast, protocole de routage explicite dédié aux petits groupes. Le protocole GXcast gère de manière optimale la fragmentation de paquets Xcast en limitant le nombre d'adresses de destination encodées dans l'en-tête du paquet Xcast. À l'issue de l'étude de ce protocole, nous avons montré que GXcast gère facilement avec une réduction du coût et de

délai, un grand nombre de groupes de taille moyenne, même lorsque les membres sont disséminés sur quelques centaines de sous-réseaux. Les sources de notre simulateur de groupes *multicast* explicites sont disponibles librement sur Internet.

Ensuite, nous avons décrit le protocole SEM, aussi particulièrement adapté aux petits groupes. Le protocole SEM utilise le service Xcast pour construire un arbre réduit entre la source et les destinataires. Une fois l'arbre réduit construit, les paquets *multicast* sont acheminés en utilisant les adresses *unicast* des différents nœuds de branchement. SEM réduit considérablement la taille des tables de routage *multicast*. Nos travaux montrent que le protocole SEM présente de nombreux avantages par rapport à la plupart des protocoles *multicast* concurrents. Notamment, lorsque nous le comparons au protocole HBH, nous constatons une diminution de la taille des tables de routage ainsi qu'une diminution du surcoût dû aux messages de contrôle. De plus, SEM réduit le temps de traitement d'un paquet dans un routeur en comparaison avec le protocole GXcast.

Finalement, nous avons discuté de la particularité de l'ingénierie de trafic *multicast* par rapport à l'ingénierie de trafic *unicast* afin de clarifier et de justifier notre choix de MPLS comme outil d'ingénierie de trafic. Nous avons étudié la combinaison du *multicast* et de MPLS en tant qu'outil d'ingénierie de trafic dédié à économiser les ressources du réseau.

Nous avons proposé le protocole MMT, protocole de routage *multicast* dans un réseau MPLS. Le protocole MMT utilise les chemins MPLS entre les nœuds de branchement de l'arbre *multicast* afin de rendre la communication en mode *multicast* simple, d'augmenter la résistance au facteur d'échelle et de réduire le nombre d'états de routage *multicast*. Notre approche présente une certaine efficacité par rapport aux autres propositions *multicast* (DVMRP, PIM-SM, MOSPF) et aux autres propositions *multicast* MPLS (PIM-MPLS, *Aggregated multicast*) puisque d'une part nous utilisons l'arbre des meilleurs chemins (qui coïncide avec l'arbre des plus courts chemins en absence des contraintes d'ingénierie de trafic) pour acheminer les paquets et d'autre part nous utilisons la technique de commutation rapide de MPLS dans les routeurs.

L'ensemble de nos résultats montrent que l'ingénierie de trafic *multicast* bénéficie du développement de protocoles spécialisés. Nous avons présenté une extension du protocole MMT : le protocole MMT2. MMT2 ajoute des améliorations au protocole MMT et a été proposé essentiellement pour résoudre le problème du routage mixte de la couche "Réseau" et la couche "Liaison des données" dans un routeur de cœur du réseau. MMT2 utilise un double niveau de labels tout en préservant le principe du protocole MMT. L'avantage de MMT2 est qu'il permet de calculer uniquement les arbres agrégés. Il permet donc d'associer plusieurs canaux à un même arbre agrégé, d'économiser l'utilisation de labels dans le domaine et de réduire même les états de routage à conserver dans les routeurs de branchement. Nous avons évalué MMT et MMT2 en terme de résistance au facteur d'échelle (la diminution en taille des tables de routage) et d'efficacité.

Nous avons développé un simulateur pour le trafic *multicast* dans un domaine MPLS permettant de futures recherches de la communauté sur le *multicast* MPLS. Nous avons discuté de la mise en application de MMT dans ce simulateur. Les résultats de simulation valident notre évaluation : nous remarquons une diminution en taille des tables de routage *multicast* par rapport aux autres approches *multicast* MPLS. Nous remarquons aussi un temps de traitement rapide d'un paquet dû à l'utilisation de la technique de commutation de labels MPLS dans les routeurs. Nous validons le coût faible réalisé par MMT par rapport aux protocoles qui utilisent l'arbre des plus courts chemins inverses.

Nous concluons finalement que le protocole MMT semble prometteur et bien adapté à une éventuelle implémentation de l'ingénierie de trafic *multicast* dans le réseau.

Perspectives

Notre contribution ne se limite pas à un réseau particulier. La concurrence acharnée entre fournisseurs d'accès à Internet ne peut qu'amener de nouvelles solutions de routage *multicast*. Ces nouvelles solutions porteront certainement à la fois sur le plan

de données et sur le plan de contrôle.

L'introduction de la commutation optique va jouer un rôle. On peut combiner la commutation de longueur d'onde avec MPLS pour faciliter la gestion de réseaux optiques (*cf.* GMPLS). Toutefois, l'élément essentiel pour la commutation de paquets est basé sur les informations contenues dans l'en-tête des paquets. C'est pourquoi on peut se poser la question de la faisabilité du routage explicite tout optique. De plus, dans les réseaux tout optique certains routeurs ne sont pas capables d'être des routeurs de branchement d'un arbre *multicast*. L'adaptation du principe de SEM à cette particularité serait certainement très intéressante.

Par ailleurs, la multiplication des groupes *multicast*, et notamment des petits groupes *multicast*, devraient apporter son lot de contraintes sur les réseaux : gestion optimale d'un grand nombre de labels, technique d'agrégation d'arbre *multicast*, etc..

Le plan de contrôle est susceptible de faire l'objet de nombreuses évolutions, en particulier dans le cadre des efforts pour définir une nouvelle génération de réseau à hauts débits qui permette l'ingénierie de trafic *multicast*. Le contrôle des communications *multicast* devra s'appuyer sur des besoins en QoS des applications.

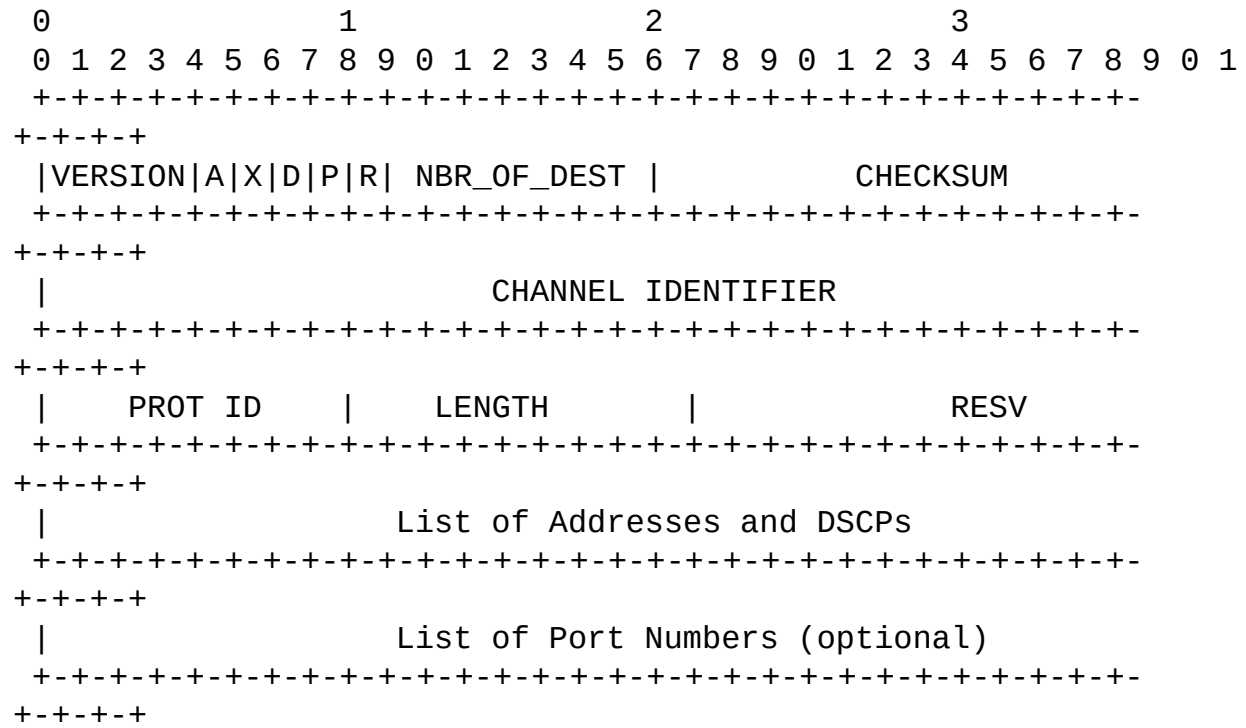
L'étude des performances de nos protocoles, en les implémentant sur différents architectures de réseaux (par ex. IPv6, ETHERNET), et la poursuite de l'étude comparative du routage explicite, surtout dans le cas de petits groupes, et de routage *multicast* classique serait très utile.

Dans cette thèse, nous nous sommes intéressés au réseau Internet, car c'est dans cet environnement qu'ont été concrétisées les propositions les plus significatives de routage *multicast*. L'étude de l'intégration des mécanismes de routage explicite dans des différents architectures de réseaux pose des problèmes encore très ouverts, et les propositions actuelles offrent des solutions partielles qui pourraient être enrichies ou optimisées, notamment en s'inspirant de nos propositions.

Annexe A

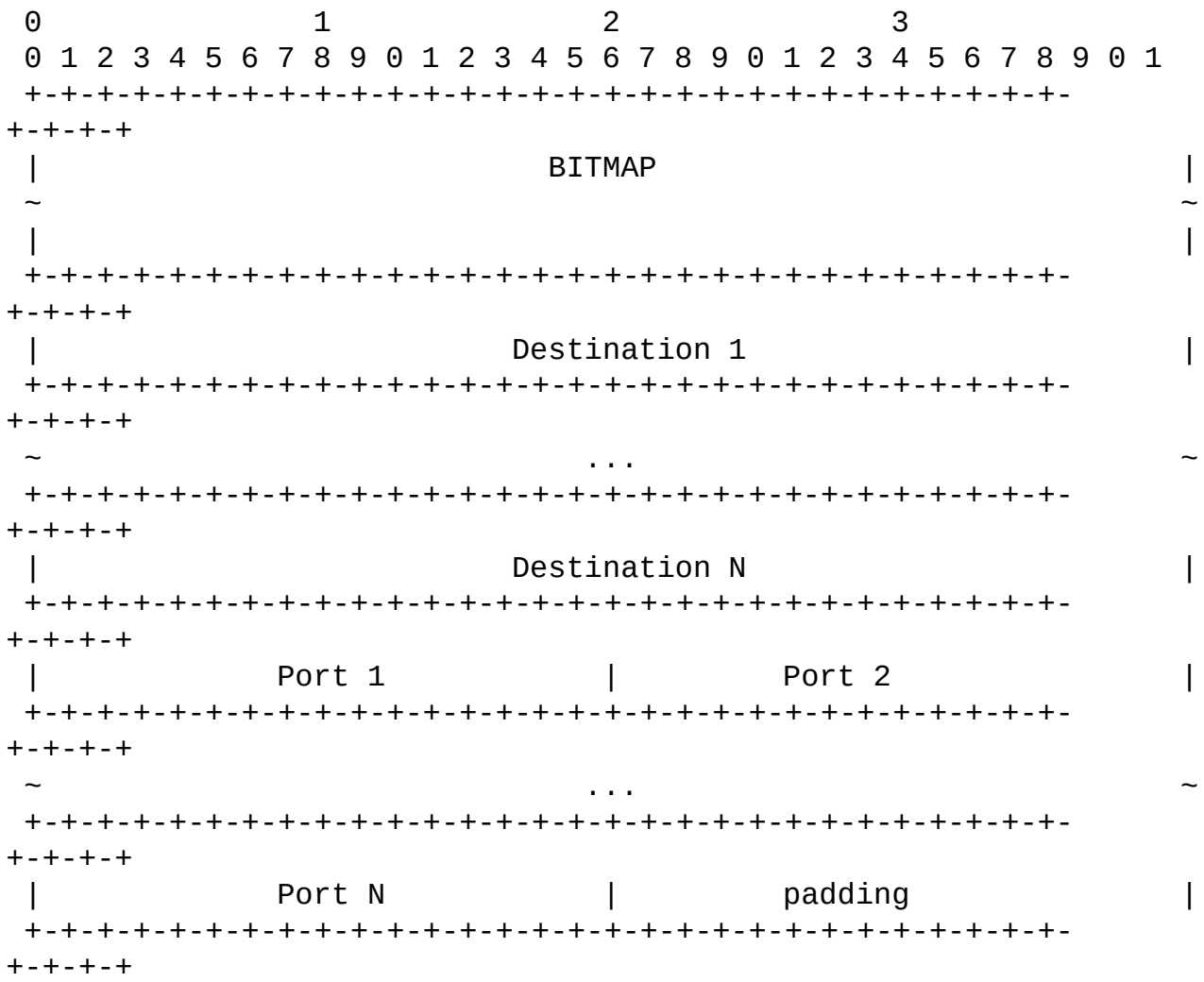
L’en-tête Xcast

L’en-tête Xcast (pour IPv4) est décrit sur le schéma suivant :



L’en-tête Xcast se compose de deux parties: une partie fixe (12 premiers octets) (voir [BFI+03] pour plus de détails) et deux champs de longueur variable qui sont spécifiées par la partie fixe. Si le bit D est positionné à 1, le paquet contient un octet

$DSCP$ (*Diffserv Codepoint*) pour chaque destination. Si le bit P est positionné à 1, le paquet contient un numéro de port pour chaque destination. Les deux champs de longueur variable sont décrits par le schéma suivant :

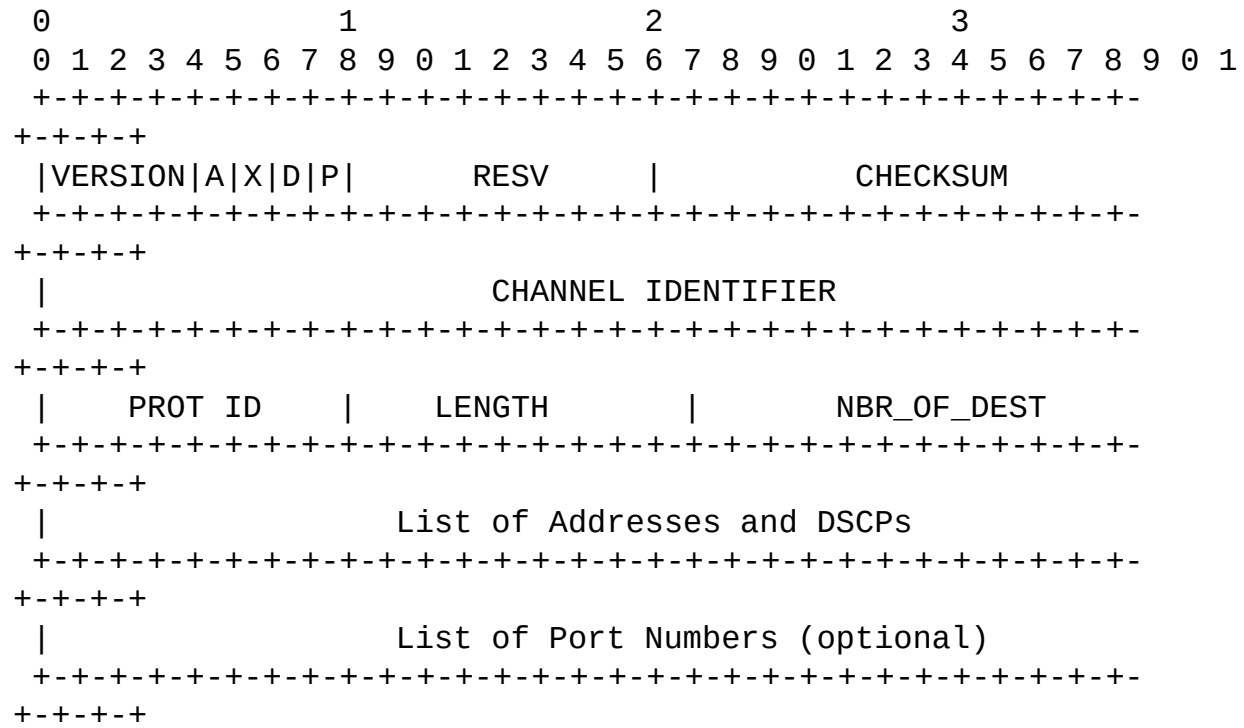


Chaque destination a un *bit* correspondant dans le *BITMAP* (*cf.* sous-chapitre 2.1.3.2) pour indiquer si la destination est encore valide sur cette branche de l'arbre. Le premier *bit* correspond à la première destination dans la liste et chaque $Port\ i$ est associé naturellement à la $Destinataire\ i$. Notons que si le bit A est positionnée à 1, la destination (les 32 de l'adresse destination) qui correspond à un bit = 0 dans le BITMAP doit être remplacée par des bits zéros.

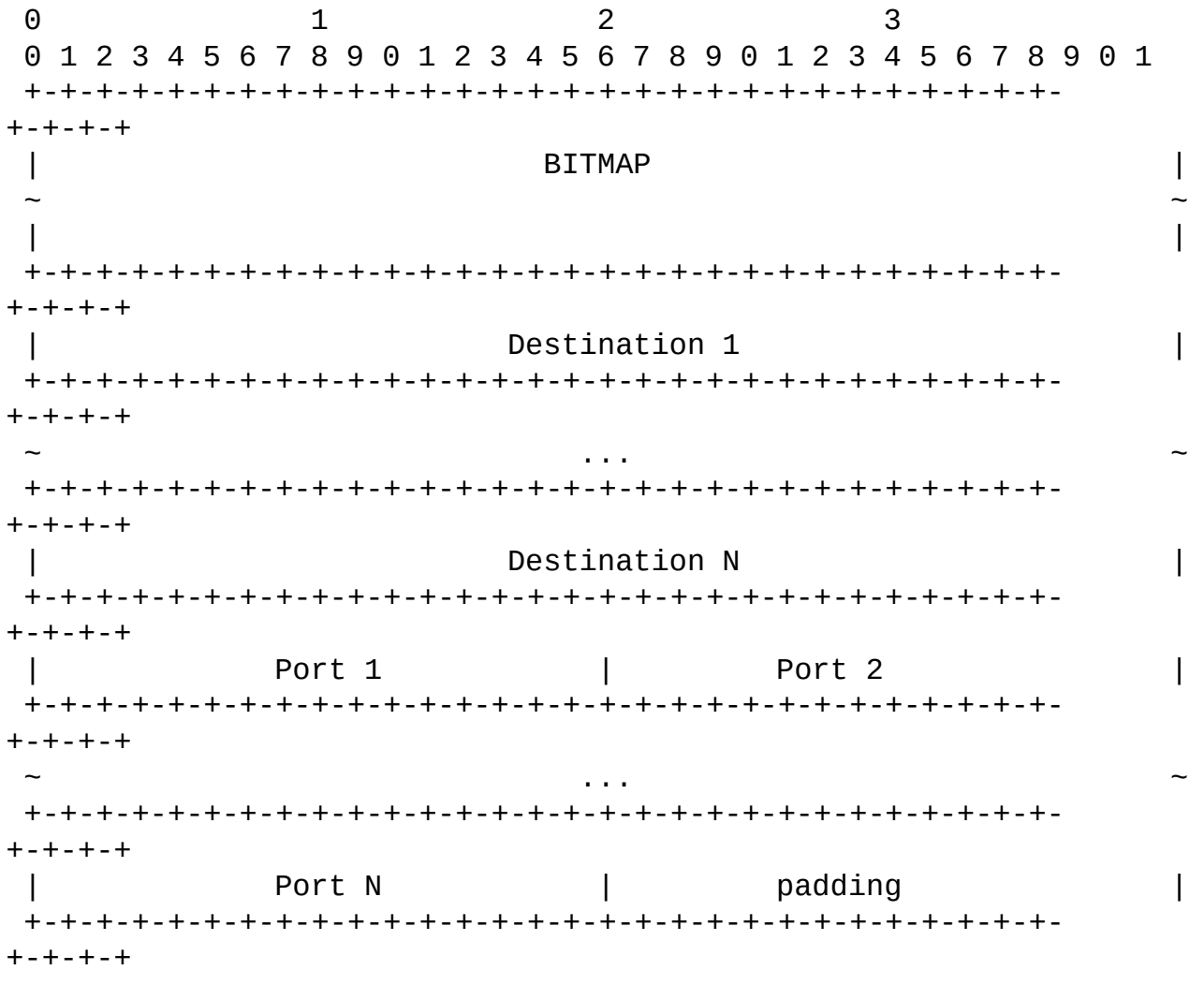
Annexe B

L’en-tête GXcast

L’en-tête GXcast (pour IPv4) est décrit sur le schéma suivant :



Il est similaire à l’en-tête Xcast et se compose de deux parties: une partie fixe (12 premiers octets) et deux champs de longueur variable qui sont spécifiées par la partie fixe. Ces deux champs (de longueur variable) sont décrits par le schéma suivant :



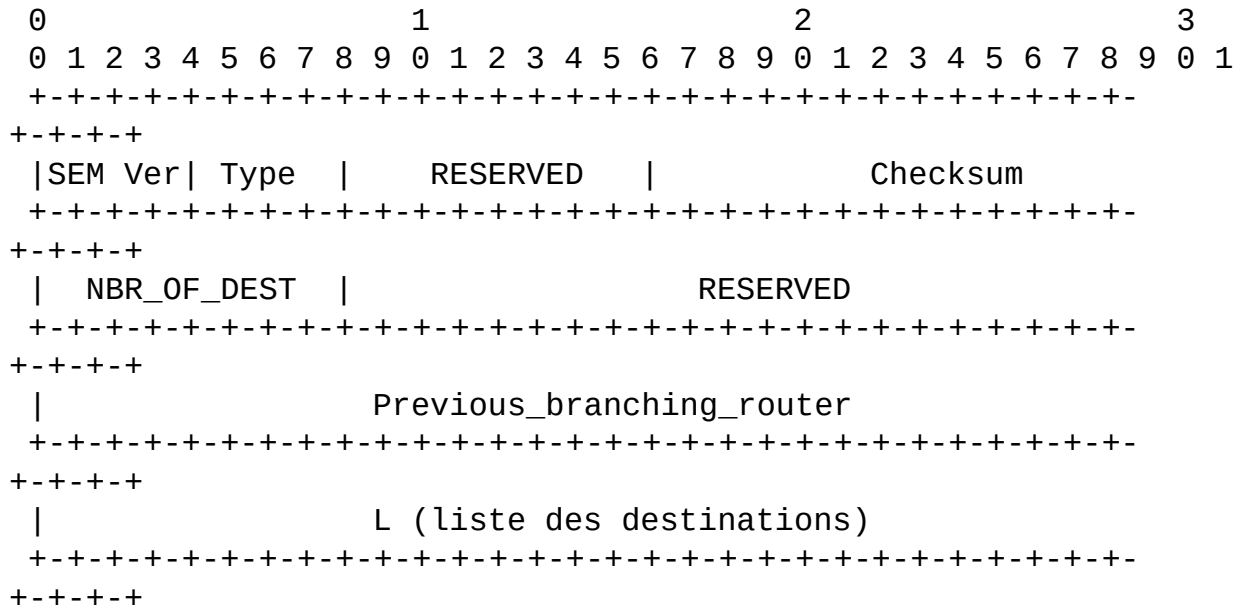
Chaque destination a un *bit* correspondant dans le *BITMAP* pour indiquer si la destination est encore valide sur cette branche de l'arbre. Le premier *bit* correspond à la première destination dans la liste et chaque $Port\ i$ est associé naturellement à la $Destinataire\ i$.

Annexe C

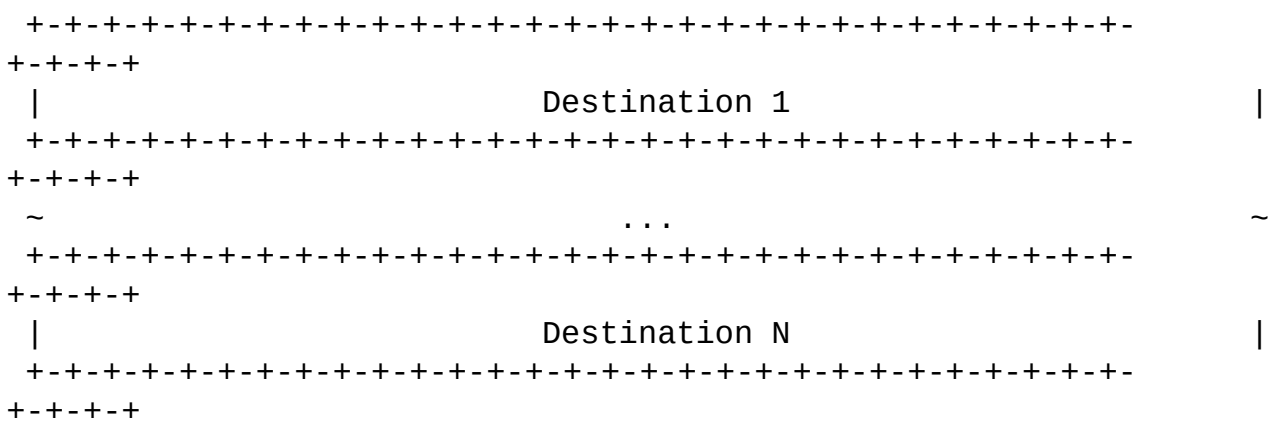
Les entêtes des messages SEM

C.1 Le message *branch*

L'en-tête SEM du message *branch* est décrit sur le schéma suivant :

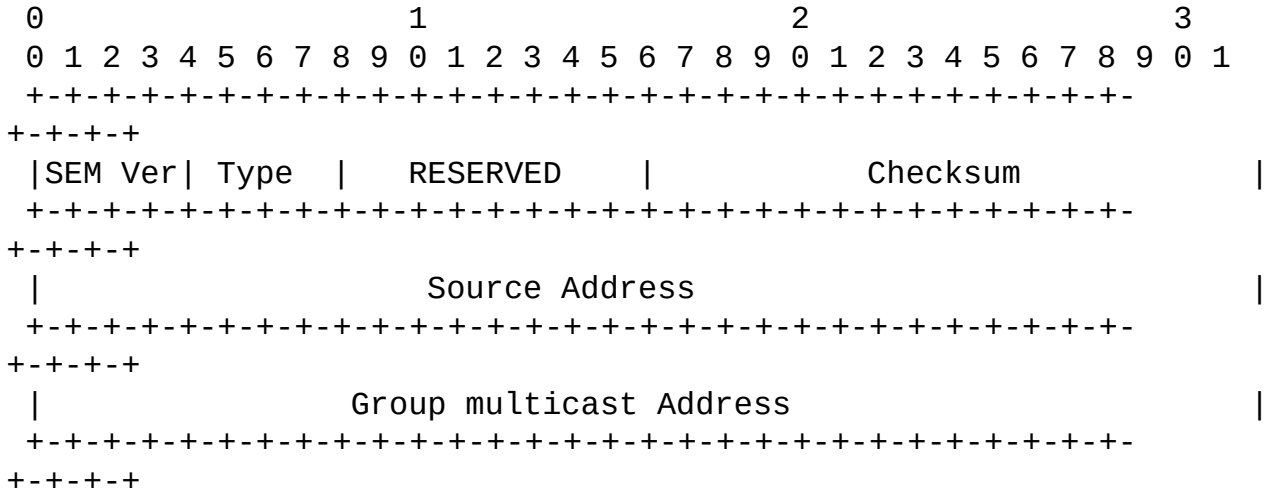


L représente la liste des destinations (champ de longueur variable) décrit par le schéma suivant :



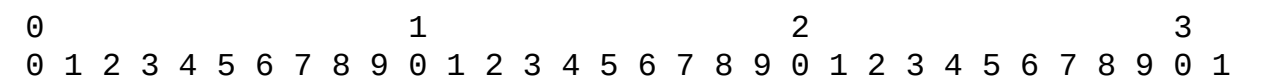
C.2 Le message *previous_branch*

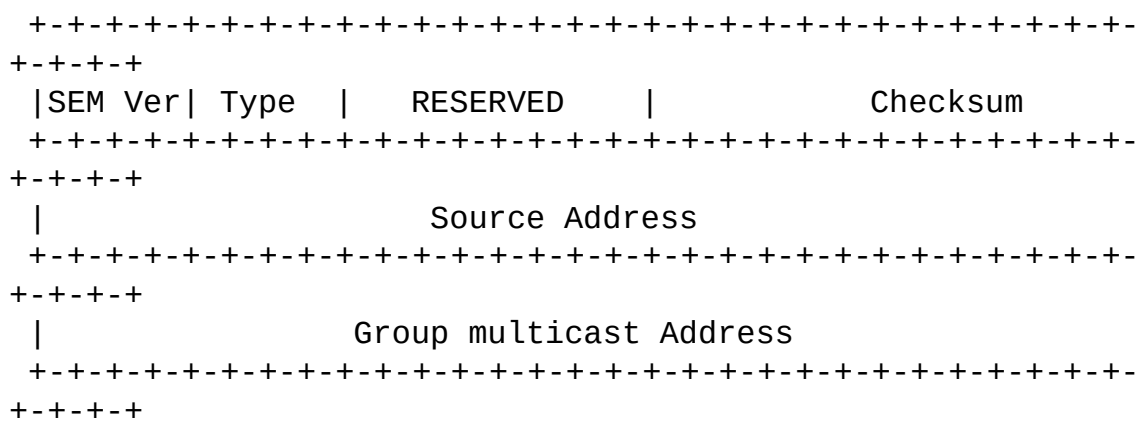
L'en-tête SEM du message *previous_branch* est décrit sur le schéma suivant :



C.3 Le message *join*

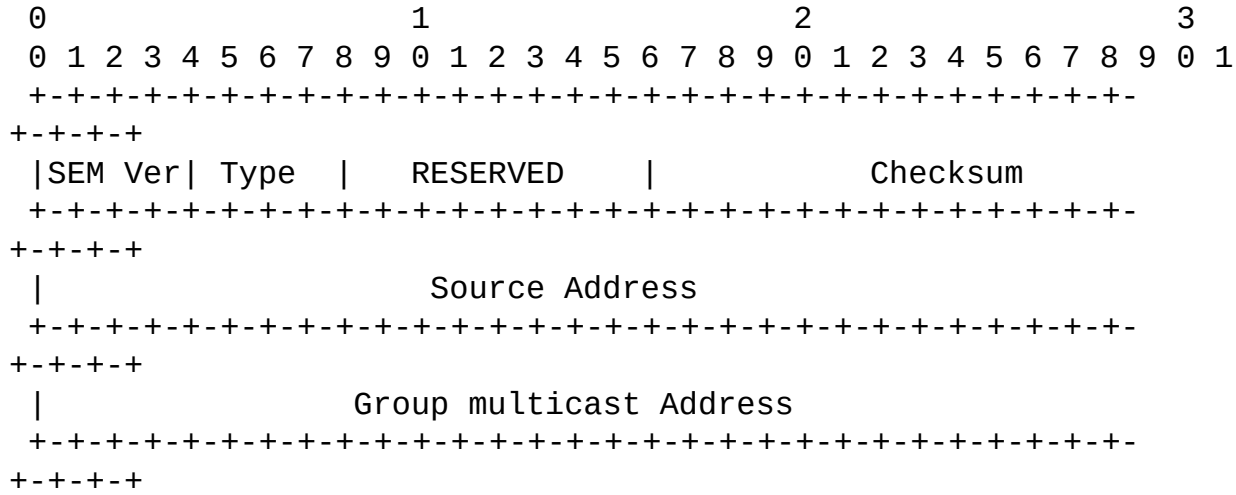
L'en-tête SEM du message *join* est décrit sur le schéma suivant :





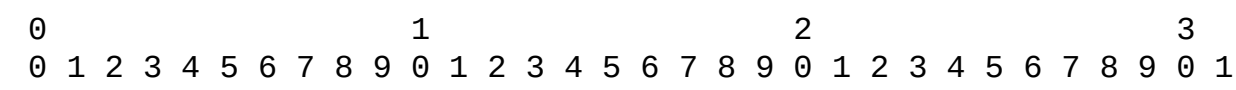
C.4 Le message *leave*

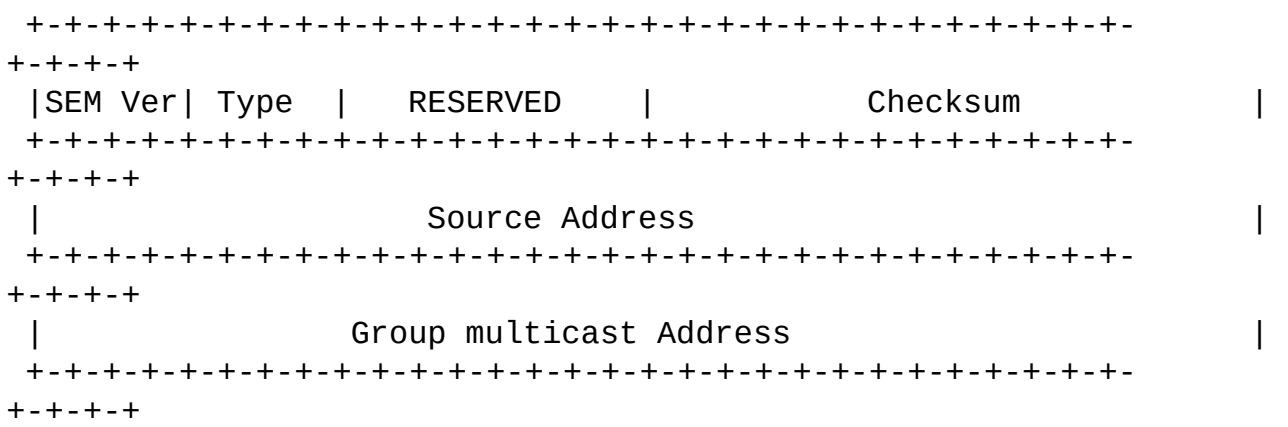
L'en-tête SEM du message *leave* est décrit sur le schéma suivant :



C.5 Le message *alive*

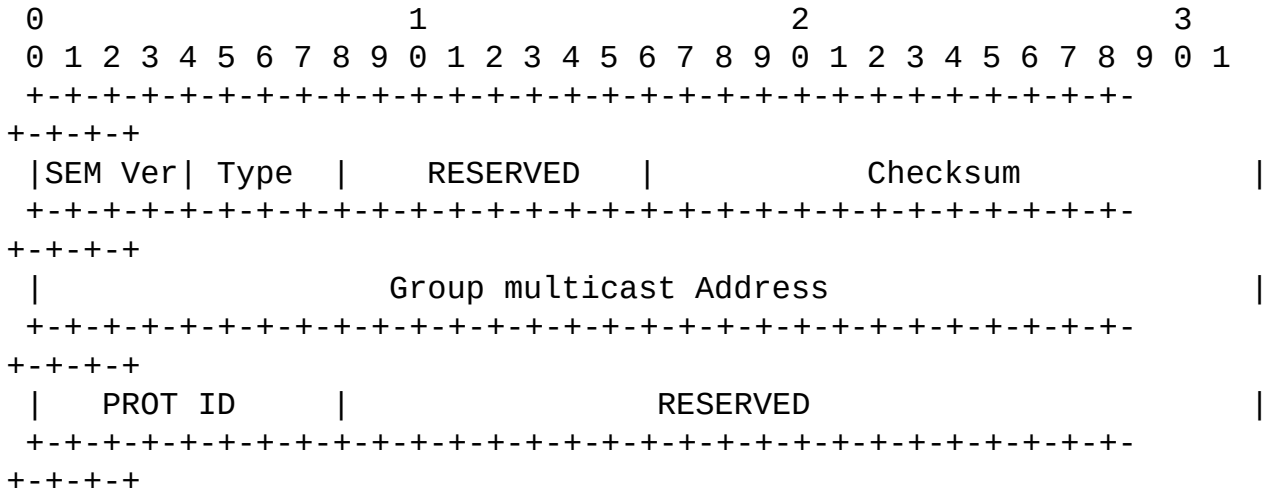
L'en-tête SEM du message *alive* est décrit sur le schéma suivant :





C.6 Le paquet de données en mode SEM

L'en-tête SEM du message de données envoyé en mode SEM est décrit sur le schéma suivant :



Liste des publications

Articles de Journaux :

1 A. Boudani, A. Guitton, B. Cousin. GXcast : une généralisation du protocole Xcast. Informations, Savoirs, Décisions et Médiations (ISDM), 2004.

Conférences Internationales :

1 A. Boudani, Guitton, B. Cousin. GXcast: Generalized Explicit Multicast Routing Protocols. The 9th IEEE Symposium on Computer and Communications, Alexandria, Egypt, 2004.

2 A. Boudani, B. Cousin. Sem: A new small group multicast routing protocol. The 10th International Conference on Telecommunications (ICT), Tahiti,Papeete, France, 2003.

3 A. Boudani, B. Cousin. Multicast Routing Simulator over MPLS Networks. The 36th Annual Simulation Symposium, Orlando, Florida, USA, 2003.

4 A. Boudani, B. Cousin, J. Bonnin. MPLS Multicast Traffic Engineering. IEEE ROC&C'2003, Mexique, 2003.

5 A. Boudani, B. Cousin. A New Approach to Construct Multicast Trees in MPLS Networks. The 7th IEEE Symposium on Computers and Communications (ISCC), Taormina, Italy, Juillet 2002.

Conférences Nationales :

1 A. Boudani, A. Guitton, B. Cousin. GXcast : une généralisation du protocole Xcast. Manifestation des Jeunes Chercheurs STIC (Majecstic), Marseille, France,

2003.

Rapports de recherche :

1 A. Boudani, B. Cousin. Using MPLS for Multicast Traffic Engineering. Rapport de Recherche IRISA, No1548, 2003.

2 A. Boudani, C. Jawhar, B. Cousin, M. Doughan. A Simulator for Multicast Routing over an MPLS Network. Rapport de Recherche IRISA, No1493, Octobre 2002.

Divers :

1 A. Boudani, A. Guitton, B. Cousin. GXcast: Generalized Explicit Multicast Routing Protocol. draft IETF: draft-boudani-gxcast-00.txt, 2003.

2 A. Boudani, B. Cousin. The MPLS Multicast Tree (MMT). draft IETF: draft-boudani-mpls-multicast-tree-00.txt, Novembre 2001.

3 A. Boudani, B. Cousin. Simple Explicit Multicast (SEM). draft IETF: draft-boudani-simple-xcast-00.txt, Juin 2001.

Table des figures

Liste des tableaux

Bibliographie

[ABG+01] D. Awduche, L. Berger, D. Gan, T. Li, V. Srinivasan, and G. Swallow. RSVP-TE: Extensions to RSVP for LSP tunnels. IETF RFC 3209, December 2001.

[AC00] G. Ahn and W. Chun. Design and Implementation of MPLS Network Simulator Supporting LDP and CR-LDP. In *IEEE International Conference on Networks (ICON'00)*, 2000.

[ADF+01] L. Andersson, P. Doolan, N. Feldman, A. Fredette, and B. Thomas. LDP Specification. IETF RFC 3036, January 2001.

[AGA99] A. Acharya, F. Griffoul, and F. Ansari. IP Multicast Support in MPLS Networks. IETF Internet draft, February 1999.

[AGKT98] G. Apostolopoulos, R. Guerin, S. Kamat, and S. Tripathi. Qaulity of Service Based Routing : A Performance Perspective. In *ACM SIGCOMM*, pages 913–919, July 1998.

[Alm00] K. Almeroth. The Evolution of Multicast: From the Mbone to Interdomain Multicast to Internet2 Deployment. *IEEE Network*, pages 10–20, February 2000.

[AMAO99] D. Awduche, J. Malcom, J. Agogbua, and M. O'Dell. Requirements for Traffic Engineering over MPLS. IETF RFC 2702, September 1999.

[AWK03] R. Aggarwal, L. Wei, and K. Kompella. Establishing Point to Multipoint MPLS TE LSPs. IETF Internet draft, May 2003.

[Bal97a] A. Ballardie. Core Based Trees (CBT) Multicast Routing Architecture. IETF RFC 2201, 1997.

[Bal97b] A. Ballardie. Core Based Trees (CBT version 2) Multicast Routing (Protocol Specification). IETF RFC 2189, 1997.

[Ban97] M. Banikazemi. IP Multicasting: Concepts, Algorithms, and Protocols. www.cis.ohio-state.edu/jain/cis788-97/, 1997.

[BC01] A. Boudani and B. Cousin. An Effective Solution for Multicast Scalability: The MPLS Multicast Tree (MMT). IETF Internet draft, October 2001.

[BC02] A. Boudani and B. Cousin. A New Approach to Construct Multicast Trees in MPLS Networks. In *The Seventh IEEE Symposium on Computers and Communications*, July 2002.

[BC03] A. Boudani and B. Cousin. SEM: A New Small Group Multicast Routing Protocol. In *The 10th International Conference on Telecommunications*, February 2003.

[BCJD03] A. Boudani, B. Cousin, C. Jawhar, and M. Doughan. Multicast Routing Simulator over MPLS Networks. In *The 36th Annual Simulation Symposium part of the ASTC 2003 Conference*, March 2003.

[BCKR98] T. Bates, R. Chandra, D. Katz, and Y. Rekhter. Multiprotocol Extensions for BGP-4. IETF RFC 2283, 1998.

[BFI+03] R. Boivie, N. Feldman, Y. Imai, W. Livens, D. Ooms, and O. Paridaens. Explicit multicast (Xcast) basic specification. IETF Internet draft, January 2003.

[BGC04] A. Boudani, A. Guitton, and B. Cousin. GXcast: Une généralisation du protocole Xcast. *Informations, Savoirs, Décisions et Médiations (ISDM)*, 13, February 2004.

[Bha03] S. Bhattacharyya. An Overview of Source-Specific Multicast (SSM). IETF RFC 3569, 2003.

[BJCD02] A. Boudani, C. Jawhar, B. Cousin, and M. Doughan. A Simulator for Multicast Routing over an MPLS Network. Technical report 1493, IRISA, October 2002.

[Bon02] J.-M. Bonnin. MPLS. *Techniques de l'ingénieur, Réseaux(IP)*, page 2905, May 2002.

[Cal90] R. Callon. Using of OSI IS-IS for routing in TCP/IP and Dual Environments. IETF RFC 1195, 1990.

[CBC+02a] J. Chung, M. Benito, H. Chhabra, G. Cho, and P. Rasiah. LDP Extensions for MPLS Multicasting Services. IETF Internet draft, February 2002.

[CBC+02b] J. Chung, M. Benito, H. Chhabra, G. Cho, and P. Rasiah. RSVP-TE extensions for MPLS Multicasting services. IETF Internet draft, February 2002.

[CDT02] B. Cain, S. Deering, and A. Thyagarajan. Internet Group Management Protocol, version 3. IETF RFC 3376, 2002.

[Che99] S. Chen. *Routing Support For Providing Guaranteed End-To-End Quality-Of-Service*. PhD thesis, University of Illinois, 1999.

[Che01] D. Cheng. RSVP-TE: Extensions to RSVP for Multicast LSP Tunnels. IETF Internet draft, October 2001.

[CKM+02] J. Cui, J. Kim, D. Maggiorini, K. Boussetta, and M. Gerla. Aggregated Multicast: A Comparative Study. In *Networking 2002*, May 2002.

[CKM+03] J. Cui, J. Kim, D. Maggiorini, K. Boussetta, and M. Gerla. Aggregated Multicast: A Comparative Study. *Special Issue of Cluster Computing: The Journal of Networks, Software and Applications*, 2003.

[CLR90] T. H. Cormen, C. E. Leiserson, and R. L. Rivest. *Introduction to Algorithms*. MIT Press/McGraw-Hill, 1990.

[CSFT00] Conseil Scientifique France Télécom. L'évolution des Réseaux. Forum France Télécom recherche, Mémento technique N°15, ISSN 1250-5447, March 2000.

[Dee91] S. Deering. *Multicast Routing in a Datagram Internetwork*. PhD thesis, Stanford University, 1991.

[DH98] S. Deering and R. Hinden. Internet Protocol, version 6 (IPv6) Specification. IETF RFC 2460, 1998.

[DHPP00] S. Deering, S. Hares, C. Perkins, and R. Perlman. Overview of the 1998 IAB Routing Workshop. IETF RFC 2902, August 2000.

[EFH+98] D. Estrin, D. Farinacci, A. Helmy, D. Thaler, S. Deering, M. Handley, V. Jacobson, C. Liu, P. Sharma, and L. Wei. Protocol Independent Multicast-Sparse Mode (PIM-SM): Protocol Specification. IETF RFC 2362, 1998.

[EJLW01] A. Elwalid, C. Jin, S. Low, and I. Widjaja. MATE: MPLS Adaptive Traffic Engineering. In *INFOCOM*, pages 1300–1309, 2001.

[ETM] Enabling Technologies MPLS. www.juniper.net/solutions.

[Far02] J. Farber. Network Game Traffic Modelling. In *The First Workshop on Network and System Support for Games (Netgames)*, April 2002.

[FCFW02] W. Feng, F. Chang, W. Feng, and J. Walpole. Provisioning Online Games: A Traffic Analysis of a Busy Counter-Strike Server. In *The Second ACM SIGCOMM Workshop on Internet Measurement*, November 2002.

[FCGF01] A. Fei, J. Cui, M. Gerla, and M. Faloutsos. Aggregated Multicast: An Approach to Reduce Multicast State. In *Proceedings of the Third International COST264 Workshop (NGC 2001) UCL. London*, number 2233 in LNCS, pages 172–188, November 2001.

[Fen97] W. Fenner. Internet Group Management Protocol, version 2. IETF RFC 2236, 1997.

[FLYV93] V. Fuller, T. Li, J. Yu, and K. Varadhan. Classless Inter-Domain Routing (CIDR): an Address Assignment and Aggregation Strategy. IETF RFC 1519, 1993.

[FM03] D. Fenner and D. Meyer. Multicast Source Discovery Protocol (MSDP). IETF RFC 3618, 2003.

[FRR00] D. Farinacci, Y. Rekhter, and E. Rosen. Using PIM to distribute MPLS labels for multicast routes. IETF Internet draft, November 2000.

[FT00] B. Fortz and M. Thorup. Internet Traffic Engineering by Optimizing OSPF Weights. In *INFOCOM (2)*, pages 519–528, 2000.

[FV01] K. Fall and K. Varadhan. The NS Manual. UC Berkeley, LBL, USC/ISI, and Xerox PARC, January 2001.

[Gra97] D. Grad. *Algorithme et Architecture de Routage Logique pour Les Communications de groupe*. PhD thesis, Université Louis Pasteur, 1997.

[HC99] H. Holbrook and D. Cheriton. IP Multicast Channels: Express Support for Large-scale Single-source Applications. In *ACM SIGCOMM*, 1999.

[HC03] H. Holbrook and B. Cain. Source-Specific Multicast for IP. IETF Internet draft, 2003.

[HCFCD01] L. HMK Costa, S. Fdida, and O. CMB Duarte. Hop-by-hop Multicast Routing Protocol. In *ACM SIGCOMM*, pages 249–259, August 2001.

[HL99] H. Hummel and S. Loke. Explicit Tree Routing. IETF Internet draft, June 1999.

[IANA01] Internet Assigned Numbers Authority. http://www.isi.edu/in-notes/iana/assignments/multicast-addresses, 2001.

[JNTPTR03] Juniper Networks T640 Performance Test Report. Technical Report TR0112, March 2003.

[LACA99] C-Y. LEE, L. Andersson, K. Carlberg, and B. Akyol. Engineering Paths for Multicast Traffic. IETF Internet draft, October 1999.

[LB00] H. S. Laxman and M. Biswanath. Multicast Routing Algorithms and Protocols : A Tutorial. *IEEE Network*, pages 90–102, February 2000.

[Mal98] G. Malkin. RIP Version 2. IETF RFC 2453, November 1998.

[MDM] J. McCann, S. Deering, and J. Mogul. Path MTU Discovery for IP version 6. IETF RFC 1981.

[Moy91] J. Moy. OSPF Version 2. IETF RFC 1247, July 1991.

[Moy94a] J. Moy. MOSPF: Analysis and Experience. IETF RFC 1585, 1994.

[Moy94b] J. Moy. Multicast Extensions to OSPF. IETF RFC 1584, 1994.

[MS94] D. McDysan and L. Spohn. *ATM Theory and Application*. McGraw Hill, inc, 1994.

[MYKS01] S. MyungKI, K. YongJin, P. KiShik, and K. SangHa. Explicit Multicast Extension (Xcast+) for Efficient Multicast Packet Delivery. *ETRI Journal*, 23(4), December 2001.

[Net] Abilene Network. http://abilene.internet2.edu/.

[OL99] D. Ooms and W. Livens. IP Multicast in MPLS Networks. Technical leaflet, Alcatel Corporate Research Center, 1999.

[Oom00] D. Ooms. Taxonomy of Xcast/SGM proposals. IETF Internet draft, July 2000.

[OSL+02] D. Ooms, B. Sales, W. Livens, A. Acharya, F. Griffoul, and F. Ansari. Overview of IP Multicast in a Multi-Protocol Label Switching MPLS Environment. IETF RFC 3353, August 2002.

[Per96] C. Perkins. Minimal Encapsulation within IP. IETF RFC 2004, October 1996.

[PMB99] M. Pullen, M. Myjak, and C. Bouwens. Limitations of Internet Protocol Suite for Distributed simulation in the large Multicast Environment. IETF RFC 2502, February 1999.

[Poi02] Y. Pointurier. *Link Failure Recovery for MPLS Networks with Multicasting*. PhD thesis, Master Thesis, University of Virginia, August 2002.

[Pos81] J. Postel. Internet Protocol. IETF RFC 791, 1981.

[Ram00] M. Ramalho. Intra- and Inter-domain Multicast Routing Protocols: A Survey and Taxonomy. *IEEE Communications Surveys and Tutorials*, 3(1):2–25, February 2000.

[RCT+03] E. Rosen, Y. Cai, D. Tappan, Y. Rekhter, and D. Farinacci. Multicast in MPLS/BGP VPNs. IETF Internet draft, October 2003.

[REG99] P. Radoslavov, D. Estrin, and R. Govindan. Exploiting the Bandwidth-Memory Tradeoff in Multicast State Aggregation. Technical report 99-697, University of Southern California, Dept. of CS, July 1999.

[REG+00] P. Radoslavov, D. Estrin, R. Govindan, M. Handley, S. Kumar, and D. Thaler. The Multicast Adress-Set Claim (MASC) Protocol. IETF RFC 2909, 2000.

[RL95] Y. Rekhter and T. Li. A Border Gateway Protocol 4 (BGP-4). IETF RFC 1771, 1995.

[RNW03] R. Ramaswamy and T. Wolf N. Weng. Considering Processing Cost in Network Simulations. In *Workshop on Models, Methods and Tools for Reproducible Network Research (MoMeTools) in conjunction with ACM SIGCOMM*, August 2003.

[RR94] M.A. Rodrigues and K.G. Ramakrishnan. Optimal Routing in Shortest Path Networks. In *ITS'94*, 1994.

[RR99] E. Rosen and Y. Rekhter. BGP/MPLS VPNs. IETF RFC 2547, March 1999.

[RTF+01] E. Rosen, D. Tappan, G. Fedorkow, Y. Rekhter, D. Farinacci, T. Li, and A. Conta. MPLS label stack encoding. IETF RFC 3032, January 2001.

[RVC01] E. Rosen, A. Viswanathan, and R. Callon. Multiprotocol label switching architecture. IETF RFC 3031, January 2001.

[SEZ00] I. Stoica, T. Eugene, and H. Zhang. REUNITE: A Recursive Unicast Approach to Multicast. In *INFOCOM (3)*, pages 1644–1653, 2000.

[Sia03] A. Adams J. Nicholas W. Siadak. Protocol Independent Multicast-Dense Mode (PIM-DM): Protocol Specification (Revised). IETF Internet Draft, 2003.

[Sit03] Reliable Multicast Transport IETF Working Group Web Site. http://www.ietf.org/html.charters/rmt-charter.html, February 2003.

[SKTA+98] P. Radoslavov S. Kumar, D. Thaler, G. Alaettino, D. Estrin, and M. Handley. The MASC/BGMP Architecture for Inter-domain Multicast Routing. In *ACM SIGCOMM*, September 1998.

[SM97] C. Semeria and T. Maufer. Introduction to IP Multicast Routing. IETF Internet Draft, 1997.

[TBCT99] M. Tezeghdanti, J-M. Bonnin, B. Cousin, and L. Toutain. Agrégation dans les réseaux MPLS. Conférence sur de nouvelles architectures pour la communication (DNAC'99), December 1999.

[TD95] A. Thyagarajan and S. Deering. Hierarchical Distance-Vector Multicast Routing Protocol. In *ACM SIGCOMM*, September 1995.

[Tha03] D. Thaler. Border Gateway Multicast Protocol (BGMP): Protocol Specification. IETF Internet Draft, 2003.

[TN98] J. Tian and G. Neufeld. Forwarding State Reduction For Sparse Mode Multicast Communication. In *INFOCOM (2)*, pages 711–719, March 1998.

[Wax88] B. Waxman. Routing of Multipoint Connections. *IEEE Journal on Selected Areas in Communications*, 6(9):1617–1622, December 1988.

[WPD88] D. Waitzman, C. Partridge, and S. Deering. Distance Vector Multicast Routing Protocol. IETF RFC 1075, 1988.

[WVTP01] M. Waldvogel, G. Varghese, J. Turner, and B. Plattner. Scalable high-speed prefix matching. *ACM Transactions on Computer Systems*, 19(4):440–482, November 2001.

[XHBN00] X. Xiao, A. Hannan, B. Bailey, and L. Ni. Traffic Engineering with MPLS in the Internet. *IEEE Network*, 14(2):28–33, March 2000.

[YI03] S. Yasukawa and I. Inoue. Requirements for Point-to-Multipoint capability extension to MPLS. IETF Internet draft, February 2003.

[YK03] S. Yasukawa and A. Kullberg. Extended RSVP-TE for Point-to-Multipoint LSP Tunnels. IETF Internet draft, June 2003.

[YKDKHJ+03] O. Young-Kyu, K. Dong-Keun, Y. Hun-Je, D. Mi-Sun, and L. Jaiyong. Scalable MPLS Multicast using Label Aggregation in Internet Broadcasting System. In *The 10th International Conference on Telecommunications*, February 2003.

[YM02] B. Yang and P. Mohapatra. Edge Router Multicasting With MPLS Traffic Engineering. In *IEEE International Conference on Networks (ICON'02)*, 2002.

[YPV+03] S. Yasukawa, D. Papadimitriou, J-P. Vasseur, A. Farrel, Y. Kamite, M. Jork, R. Aggarwal, Andrew G. Malis, and A. Kullberg. Requirements for Point to Multipoint Extension to RSVP-TE. IETF Internet draft, October 2003.

[YR95] C. Yang and A. Reddy. A Taxonomy for Congestion Control Algorithms in Packet Switching Networks. *IEEE Network*, 9(5), July 1995.

[ZCB96] E. Zegura, K. Calvert, and S. Bhattacharjee. How to Model an Internetwork. In *INFOCOM*, 1996.

Résumé

Cette thèse s'intitule "Routage *multicast* : gestion des petits groupes et ingénierie de trafic". Les deux problèmes étudiés dans cette thèse concernent le passage à l'échelle et l'ingénierie de trafic pour le *multicast*. Nous mettons en relation ces deux aspects et nous définissons de nouveaux protocoles plus adaptés. Dans la première partie de cette thèse, nous proposons le protocole GXcast, protocole de routage explicite dédié aux petits groupes. Le protocole GXcast gère de manière optimale la fragmentation de paquets Xcast en limitant le nombre d'adresses des destinations encodées dans l'entête du paquet Xcast. Ensuite, nous décrivons le protocole SEM, aussi particulièrement adapté aux petits groupes. Le protocole SEM utilise le service Xcast pour construire un arbre réduit entre la source et les destinataires. Une fois l'arbre réduit construit, les paquets *multicast* sont acheminés en utilisant les adresses *unicast* des différents nœuds de branchement.

Finalement, nous étudions la combinaison du *multicast* et de MPLS en tant qu'outil d'ingénierie de trafic dédié à économiser les ressources du réseau. Nous proposons le protocole MMT, protocole de routage *multicast* dans un réseau MPLS. Le protocole MMT utilise les chemins MPLS entre les nœuds de branchement de l'arbre *multicast* afin de rendre la communication en mode *multicast* simple, d'augmenter la résistance au facteur d'échelle et de réduire le nombre d'états de routage *multicast*. L'ensemble de nos résultats montrent que l'ingénierie de trafic *multicast* passe par le développement de protocoles spécialisés.

Mots clés : routage *multicast*, petits groupes, ingénierie de trafic, Xcast, Internet, MPLS.