DEA Réseaux et Systèmes Distribués
Université de Nice Sophia-Antipolis

Rapport de stage

Le Routage Multicast Hiérarchique

Présenté par
Chadi BARAKAT

Sous la direction de
Daniel KOFMAN

Laboratoire d'accueil
Département INF-RES

École Nationale Supérieure des Télécommunications - Paris

Soutenu le 25 Juin 1998

Résumé

Dans ce travail, on a traité le problème du routage multicast par des arbres partagés hiérarchiques. Les membres du groupe sont connectés par un backbone de cores organisé hiérarchiquement en plusieurs niveaux. Cette approche est considérée comme solution aux problèmes des autres types d'arbres multicast permettant au protocole d'évoluer à des grands réseaux. On a regroupé les travaux faits pour implémenter cette hiérarchie dans l'Internet et l'ATM. Certains problèmes d'implémentation ont été résolus. Ensuite, on a étudié par des simulations la variation de la performance du multicast en fonction des paramètres de l'arbre particulièrement le nombre de niveaux et la distribution de cores sur ces niveaux. Trois modèles ont été définis. Les deux premiers essaient de trouver le meilleur dimensionnement de l'arbre qui réduit le coût total de la bande passante consommée tandis que le troisième cherche l'effet de la hiérarchie sur la concentration de trafic aux cores. Cette concentration cause des blocages des demandes de connexion et une mauvaise utilisation des ressources. Dans le but de bien comprendre le problème, deux types de réseaux ont été considérés: les réseaux plats et les réseaux hiérarchiques. Les premiers servent à trouver la meilleure distribution des cores dans un domaine n'ayant aucune organisation hiérarchique et les derniers montrent l'effet de la topologie du réseau sur la performance du multicast. La combinaison des résultats obtenus pour chacun d'eux donne une idée claire sur le problème de dimensionnement.

Table des matières

Table des figures

Chapitre 1

Introduction

Le multicast ou bien la communication de groupe signifie la transmission d'informations d'une ou plusieurs sources à un certain nombre de destinations sans duplication à la source. Ces destinations n'ont qu'à former un groupe ayant une adresse connue. Une source désirant émettre au groupe se contente d'envoyer une seule copie de l'information à cette adresse. Cette méthode de communication devient de plus en plus répandue surtout avec l'apparition des nouvelles applications multimédia transmettant voix et vidéo entre les participants à une conférence. Ce genre d'applications est caractérisé par le grand trafic qu'il génère et les fortes contraintes temporelles qu'il impose (faible délai, synchronisation entre les récepteurs ...). De même, il connecte des membres éparpillés dans des vastes réseaux ce qui nous oblige de tenir compte des caractéristiques de ce nouveau milieu de transmission (rareté des ressources, longues distances, existence d'une variété de protocoles...).

L'acheminement du trafic multicast demande un support de la part du réseau sous forme des arbres joignant les sources et les destinations. Le trafic se propage le long de ces arbres et sera dupliqué en chacun de leurs noeuds. Puisqu'ils constituent le chemin suivi par les informations, la nature de ces arbres est le facteur le plus important qui affecte la performance du multicast. Les ressources qu'ils consomment déterminent le degré d'évolutivité du protocole. Un arbre très coûteux, en terme de bande passante ou bien en terme de volume d'informations à stocker dans les noeuds du réseau, ne permet pas l'existence d'un grand nombre de groupes en même temps. Aussi, le choix du chemin entre un nouveau membre et les sources affecte la qualité de la réception.

Dans un multicast local, l'effet de ces arbres sur la performance n'est pas très important mais dès qu'on commence à parler d'une communication entre des membres largement dispersés, il faut penser sérieusement à optimiser le volume des ressources consommées par ces arbres. Dans les grands réseaux formés d'une interconnexion de petits réseaux comme l'Internet, les ressources ne sont pas très abondantes d'une part et d'autre part les grandes distances qui séparent les membres rendent le choix des chemins répondant aux besoins des récepteurs en terme de QoS plus crucial.

Vue l'importance de cette optimisation, notre travail a traité une nouvelle approche présentée comme solution au problème de construction des arbres multicast. Après une brève comparaison entre les différents arbres proposés et utilisés par les protocoles existants, le routage par des arbres multicast partagés et hiérarchiques est expliqué. La performance de ce routage est fortement liée à la structure de la hiérarchie. Cette relation est étudiée par des simulations. Aussi, l'implémentation de cette nouvelle approche dans les deux types de réseaux les plus connus, Internet et ATM, est détaillée.

Dans le chapitre suivant, une comparaison entre les différents types d'arbres multicast est présentée. Le chapitre 3 traite en détails l'approche du routage multicast hiérarchique.

Dans ce chapitre, on parle de l'état de l'art dans ce domaine. On explique les propositions déjà faites et on présente des solutions à quelques problèmes non traités. Dans le chapitre 4, le sujet principal de notre simulation ainsi que les modèles conçus sont expliqués. Le chapitre 5 contient les résultats obtenus et les interprétations nécessaires. Le travail est conclu dans le chapitre 6.

Chapitre 2

Comparaison entre les arbres multicast

Plusieurs types d'arbres ont été proposés dans la littérature. Certains ont été utilisés par des protocoles dans l'Internet et l'ATM, d'autres sont pure théoriques et très difficiles à être mis en oeuvre. Chaque arbre possède ses propres caractéristiques qui le distinguent des autres. Parmi ces caractéristiques on peut citer:

- La complexité et la centralisation du calcul. En pratique, on a intérêt à distribuer le calcul et les opérations de gestion de l'arbre multicast. Un calcul centralisé requiert une connaissance de toute la topologie du réseau ce qui limite énormément l'évolutivité du protocole à des grands réseaux et engendre une grande charge de calcul à cause de la dynamicité des membres du groupe. Aussi, il rend la transmission très sensible au bon fonctionnement de ce centre.

- Le volume des ressources utilisées qui affecte aussi l'évolutivité du protocole. Par ressources on entend la bande passante demandée pour bâtir cet arbre et la quantité d'informations à stocker dans les noeuds du réseau pour le maintenir. On a toujours intérêt à réduire ce volume.
  A côté de la réduction, il faut bien choisir ces ressources; les informations sur l'arbre doivent être stockées seulement dans les noeuds concernés et la bande passante consommée doit être distribuée sur tout le réseau. Un arbre qui partage son trafic sur plusieurs liens est beaucoup plus mieux qu'un autre qui demande le même volume de ressources mais qui les concentre en quelques liens. La concentration réduit les revenus dans le cas d'un réseau fournissant des garanties de QoS comme l'ATM et dégrade la qualité de réception dans le cas d'un réseau Best Effort comme l'Internet.

- La performance des chemins entre les sources et les destinations. Il s'agit de la qualité de réception perçue par les membres du groupe (délai, variation du délai).

Une meilleure implémentation d'un arbre en pratique est celle qui respecte le service multicast de base. Avec ce service un nouveau membre est capable de joindre et de quitter le groupe sans informer les sources qui émettent donc sans connaître leur identité. Un tel multicast est dit **Receiver Driven**. Aussi, une nouvelle source doit être capable d'émettre au groupe sans connaître la liste de ses membres. Un tel service permet au protocole d'évoluer du point de vue des hôtes.

Dans la suite, on va récapituler les différentes propositions et les comparer suivant les critères ci-dessus. Les domaines d'application de chaque arbre seront aussi mentionnés.

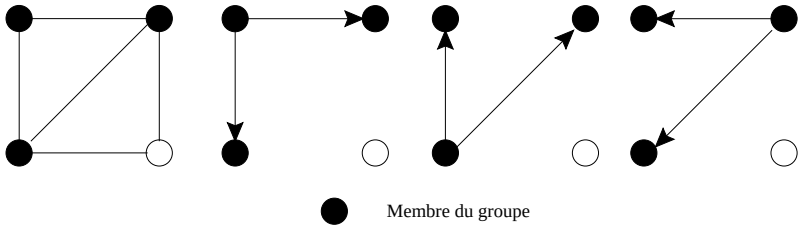

FIG. 2.1 – *Des arbres basés sur les sources*

2.1 Les différents types d'arbres multicast

2.1.1 Arbre basé sur chaque source (Source-Specific Tree)

Les données d'une source se propagent le long des plus courts chemins entre elle et les membres du groupe. Il s'agit d'un arbre par source et par groupe comme il est montré à la figure 2.1. L'avantage de cette approche, qui est utilisée dans un grand nombre de protocoles (DVMRP [11], MOSPF [12], PIM-DM [13], a mesh of VCs point-to-multipoint [1]), est qu'elle assure le plus faible délai de transmission entre sources et destinations ce qui est très convenable pour les applications temps réel interactives. De même, elle répartit le trafic des sources sur une grande partie du réseau (Load Balancing) ce qui assure une utilisation équitable des liens, donc une meilleure qualité de réception et un meilleur revenu pour l'opérateur.

L'implémentation de cet arbre se diffère un peu entre l'Internet et l'ATM, mais dans les deux cas il faut un certain mécanisme pour annoncer les sources aux récepteurs qui décident de joindre le groupe. En IP, puisqu'il n'y a pas de connexions, les paquets peuvent être émis sans l'existence des membres et donc ils constituent eux même le mécanisme qui annonce une source à son groupe. Une source diffuse son trafic dans tout le réseau et ce sont les réponses des membres qui créent les états dans les routeurs. L'ensemble des ces états forment les arbres des différentes sources.

En revanche, un réseau comme ATM demande l'établissement d'une connexion avant l'avancement du trafic et ne permet pas donc à une nouvelle source d'utiliser la diffusion pour annoncer sa présence. Ceci l'oblige de connaître les adresses des récepteurs pour établir une connexion point-to-multipoint avec eux. La solution était introduite par MARS [14] qui consiste en un serveur contenant la liste des membres et des sources. Pour qu'un nouveau membre puisse joindre les arbres existants, l'ATM Forum a définit le Leaf Initiated Join [3](Root LIJ and Network LIJ). Ceci lui permet, après avoir connu la liste des sources et les identificateurs de leur VC point-to-multipoint, de se rattacher à et de se détacher de ces VCs à sa volonté. Bien qu'une source et un membre sont obligés de connaître les autres, l'utilisation du MARS et du LIF rend le service multicast de l'ATM proche de celui de l'Internet.

Les inconvénients de ces arbres sont nombreux:

1. Un grand volume d'informations doit être stocké dans les routeurs pour maintenir l'état sur l'arbre (de l'ordre de $O(S \times G)$ où S est le nombre des sources émettant à un groupe et G le nombre des groupes). Le stockage d'une entrée par source et par groupe est inacceptable en cas des larges groupes ayant un grand nombre de sources. En IP, chaque entrée contient l'interface sur laquelle le trafic de cette source peut arriver et les interfaces de sortie. Un grand nombre d'entrées rend aussi la recherche plus coûteuse. En ATM, il y a une autre limitation; c'est le nombre des VC point-to-multipoint qu'il faut économiser.

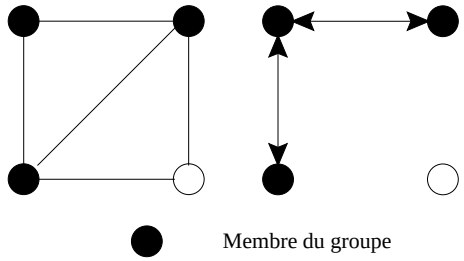

FIG. 2.2 – *Un arbre partagé*

2. Dans l'Internet, tous les routeurs du réseau stockent des informations même ceux qui ne sont pas sur l'arbre multicast.

3. Dans l'Internet aussi, la construction de ces arbres demandent une diffusion d'informations dans tout le réseau (DVMRP et PIM-DM diffusent les paquets de données lorsque les **Prunes** expirent et MOSPF diffuse des informations sur l'emplacement des membres du groupe).

4. En ATM, un mécanisme additionnel est ajouté pour annoncer les sources à un nouveau récepteur et les récepteurs à une nouvelle source. Le MARS constitue un point de concentration d'opérations ce qui limite la robustesse du multicast.

5. Ils ne tiennent pas compte de la bande passante consommée. Le coût total des arbres construits pour un groupe donné n'est pas limité. L'important pour eux est la minimisation du délai.

Ces inconvénients rendent cette approche non adéquate pour des grands réseaux ayant des groupes épars. Le grand volume d'état limite son application aux petits groupes et la diffusion le rend intéressant pour les groupes denses, donc pour les petits réseaux. Elle est le plus convenable pour faire du multicast dans des domaines riches en ressources.

2.1.2 Arbre partagé (Shared Tree)

Pour qu'un protocole soit capable d'évoluer, il faut réduire le volume d'informations à stocker dans un noeud et la bande passante consommée. Aussi, il faut éliminer le mécanisme d'annonce des sources aux récepteurs. Ce mécanisme a nécessité une inondation de l'Internet et un serveur d'adresses en ATM. Dans l'Internet, il faut aussi faire quelques choses pour que seulement les noeuds de l'arbre soient conscients de l'existence du groupe.

La solution est de disposer d'un même arbre bidirectionnel pour tous les membres du groupe que ce soient sources ou récepteurs (Figure 2.2). Pour maintenir un tel arbre, il suffit de garder une entrée par groupe au niveau de chaque noeud ce qui réduit énormément le volume d'informations surtout en cas de larges groupes (réduction de $O(S \times G)$ à $O(G)$). Cette nouvelle entrée contient les interfaces du noeud appartenant à l'arbre partagé et elle est indexée par l'adresse du groupe. Toutes ces interfaces peuvent recevoir du trafic multicast.

L'implémentation d'une telle entrée bidirectionnelle est facile dans l'Internet où le service fourni par le réseau est très simple et où les couches supérieures sont capables de corriger tous les problèmes de déséquencement et de synchronisation des paquets IP. En ATM, un message de la couche AAL5[1] est décomposé en des cellules de petite taille

1. ATM Adaptation Layer.

mais aucun identificateur de message n'est ajouté à l'entête ATM de ces cellules. Donc pour qu'un message AAL5 soit correctement interprété, une arrivée en séquence de ses cellules est demandée. Un commutateur ATM, recevant les messages de plusieurs membres et voulant les transmettre à la même destination sur un même VC de sortie, ne doit pas mélanger les cellules des différents messages.

Pour optimiser l'autre consommation de ressources (i.e. la bande passante) et puisqu'un seul arbre connecte les membres du groupe, il faut essayer de générer celui ayant un coût total minimal. Par coût total on désigne la somme des coûts de la bande passante sur les différentes branches de l'arbre. Il ne faut pas oublier qu'en pratique, à côté de la minimisation du coût, il faut chercher la solution qui s'implémente d'une façon distribuée, qui s'adapte à la dynamicité des membres et qui respecte les contraintes temporelles imposées par les applications.

L'arbre théorique qui s'intéresse à connecter un ensemble de noeuds d'un graphe par un minimum de coût est le **Minimum Steiner Tree** [20]. Ce problème a été prouvé NP-complet et il est impossible d'arriver à une solution en un temps non polynômial. Pour cela plusieurs heuristiques ont été définies pour calculer un arbre partagé ayant un coût proche de l'optimal mais en un temps polynômial. Une de ces heuristiques est l'algorithme **KMB** [21] qui a une complexité $O(|G|.|V|^2)$ où G est l'ensemble des membres du groupe et V l'ensemble des noeuds du réseau. Le coût de l'arbre obtenu est borné par

$$\frac{D(T_S)}{D(T_{opt})} \leq 2(1 - \frac{1}{|G|})$$

Ce type d'heuristiques requiert toujours la connaissance de toute la topologie du réseau par le point chargé du calcul de l'arbre[2] ce qui n'est pas de tout convenable en pratique. Aussi, le calcul s'intéresse seulement à la minimisation du coût et ne tient pas compte des contraintes temporelles. L'avantage de telles solutions est qu'elles constituent une référence théorique pour valider d'autres approches plus pratiques.

D'autres heuristiques ont été proposées pour résoudre le problème de concentration du calcul. Elles calculent l'arbre d'une manière distribuée et donc peuvent constituer des solutions pratiquement implémentables. Cependant la connaissance globale du réseau est toujours demandée. Les simulations faites ont montré qu'elles donnent un coût raisonnable par comparaison à la solution optimale. Doar & Leslie ont montré que leur **Naive Algorithm** [22] a un coût inférieur à 1.5 fois celui de KMB. Elles partent toutes d'un arbre optimal connectant quelques membres puis ajoutent les nouveaux membres suivant le plus court chemin à un des noeuds de l'arbre existant. Le choix du noeud de rattachement se fait suivant un des deux algorithmes de **Waxman** [23]:

Greedy qui consiste à choisir le noeud de l'arbre le plus proche du nouveau membre.

Weighted Greedy qui choisit le noeud **v** minimisant une fonction de la distance entre le nouveau membre **u** et **v** et de la distance entre **v** et un point **o** appelé propriétaire de la connexion. La fonction de coût est:

$$W(v) = (1 - w).d(u, v) + w.d(v, o)$$

où $0 \leq w \leq 0.5$ et d(x,y) est la distance entre deux noeuds x et y.

Un noeud qui désire quitter le groupe ne fait que se détacher de l'arbre s'il est une feuille. Il est clair qu'à un certain moment la performance de l'arbre se dégrade ce qui nécessite une restructuration périodique globale ou locale.

2. Ce noeud doit connaître la distribution des membres du groupe et la carte du réseau.

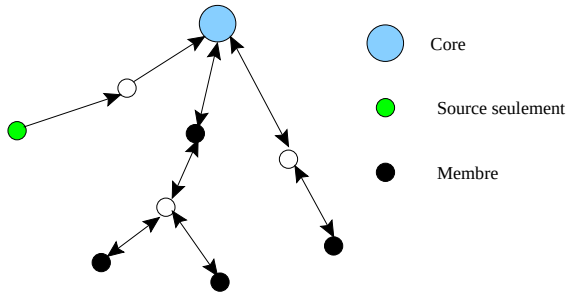

FIG. 2.3 – *Un arbre basé sur un centre*

Les simulations ont montré aussi que ces heuristiques ne sont pas si mauvaises en terme de délai et que ce délai est borné, en moyenne, par deux fois celui d'un arbre basé sur la source [24]. En plus, certaines heuristiques construisent des arbres partagés ayant des délais entre sources et récepteurs bornés et des variations de délai entre les différents récepteurs inférieures à une certaine tolérance [25]. Ces deux contraintes doivent être considérées lorsque la livraison des données aux récepteurs doit être accomplie simultanément et rapidement. Dans le cas d'un arbre par source, en cherchant les chemins les plus courts, on peut avoir des variations de délai très grandes entre les différents récepteurs.

Donc les problèmes d'implémentation du Steiner et de ses heuristiques limitent leur utilisation à des petits réseaux avec des groupes pas très dynamiques. Un autre problème des arbres partagés en général est la concentration de trafic sur les liens. Cette concentration est très mauvaise et limite le nombre des sources du groupe et le volume des données qu'elles peuvent émettre. Elle pose encore le problème de partage de la bande passante entre les différents membres.

Il faut donc chercher un certain compromis entre l'arbre théorique à coût optimal et les arbres basés sur les sources c.à.d un arbre partagé qui se calcule d'une façon distribuée et qui conserve le service multicast de base. Ce sont les arbres basés sur un centre.

2.1.3 Arbre basé sur un centre (Center Based Tree)

Il s'agit d'un arbre partagé basé sur un point appelé centre de l'arbre (Figure 2.3). Un noeud qui désire joindre le groupe envoie sa demande au centre suivant le chemin le plus court. Les noeuds traversés par cette demande créent les entrées nécessaires pour marquer ce chemin comme branche de l'arbre partagé. Cette nouvelle branche établie s'arrête au centre ou bien au premier noeud rencontré sur l'arbre. Lorsqu'une source veut seulement émettre au groupe, elle envoie ses données au centre qui se charge de les diffuser sur l'arbre partagé à tous les membres du groupe.

Cet arbre est la seule implémentation des arbres partagés en pratique. Elle réduit le volume d'informations à stocker dans les noeuds et conserve le service multicast de base. Il suffit qu'un nouveau membre ou bien une nouvelle source connaisse l'adresse du centre pour pouvoir communiquer avec les autres membres. Ses problèmes majeurs sont le choix du centre et la concentration de trafic en ce point et sur les liens qui l'entourent. Le choix du centre affecte la performance de la transmission comme le délai et la bande passante consommée. Aussi, si les centres de plusieurs groupes sont placés l'un à côté de l'autre, la concentration de trafic sera très grave. Il faut essayer de les disperser dans le réseau. Un choix performant est un problème NP-complet qui requiert une connaissance de tout le réseau et de la position des membres du groupe. De même ici des heuristiques ont été proposées pour choisir le centre en fonction des paramètres qu'on cherche à optimiser. Toujours il faut faire un compromis entre la quantité d'informations demandées pour choisir

le centre et la non performance de l'arbre obtenu.

Malgré ces problèmes et pour les avantages qu'on a cités, cet arbre a été adopté pour le routage multicast dans les grands réseaux surtout dans les backbones pas très riches en ressources. Dans la section suivante on va parler des différents protocoles implémentant cet arbre. Ensuite le routage multicast hiérarchique sera introduit comme solution aux problèmes du Center Based Tree permettant l'évolutivité du protocole.

Chapitre 3

Le Routage Multicast Hiérarchique

La nécessité d'avoir un arbre partagé qui conserve le service multicast de base et qui réduit la charge de calcul et la quantité d'informations, rend le Center Based Tree le plus convenable pour un protocole de routage multicast à grande échelle. Par un bon placement du centre, on est capable d'optimiser le coût total de l'arbre et de respecter certaines contraintes temporelles comme le délai moyen et la variation de délai. Un membre est capable de joindre le groupe et de le quitter sans demander à un point central de calculer le nouvel arbre. Une branche s'ajoute pour connecter un nouveau membre au groupe sans affecter les branches qui sont déjà sur l'arbre et donc la qualité de réception perçue par les autres. Les sources sont capables d'émettre au groupe sans qu'elles connaissent les identités de ses membres.

Deux protocoles multicast de l'Internet ont utilisé cette approche. On trouve la version Sparse Mode du Protocol Independent Multicast (PIM-SM) [10] et le Core Based Tree (CBT) [9]. Le multicast par serveur en ATM [1] utilise aussi un centre pour diffuser le trafic au groupe. Une brève explication de ces protocoles est donnée par la suite.

3.1 Les protocoles utilisant le Center Based Tree

En Internet, un membre désirant joindre le groupe envoie un message **Join** au centre de l'arbre (Rendez-vous Point dans PIM, Core dans CBT). Ce message se propage jusqu'à trouver un routeur qui est déjà sur l'arbre du groupe désiré, sinon jusqu'au centre. Il sera alors acquitté et une entrée pour le groupe sera réservée au niveau de chaque routeur traversé. À partir de cet instant, le trafic multicast arrivant à un de ces routeurs et destiné à ce groupe sera dupliqué vers les autres. Ce nouveau membre peut alors jouer le rôle d'un émetteur au groupe ainsi qu'un récepteur. Maintenant, pour maintenir cet état créé dans les routeurs, une signalisation est demandée. Elle sert à informer l'arbre de l'existence d'un membre et de le corriger lors de tout changement dans l'état du réseau. Cette signalisation peut être de bout en bout entre membre et centre (PIM-SM) ou bien entre routeurs de l'arbre (CBT).

En PIM-SM, les membres envoient périodiquement leur Join au RP qui les acquitte. Ces messages suivent les chemins fournis par les tables de routage unicast. Un routeur garde une interface dans une entrée tant qu'il reçoit des Join ou des Ack. Les branches de l'arbre changent alors dynamiquement lors de tout changement dans l'état des liens et des routeurs du réseau. On dit que l'état dans PIM est du **Soft-State**.

Au contraire dans CBT, où l'état est appelé **Hard-State**, les routeurs CBT de l'arbre dialoguent entre eux par des **Hello**. Si un routeur trouve que son père[1] ne répond pas

1. Le routeur CBT par lequel il passe pour aller au core.

suite à un problème au niveau du père ou des liens entre eux[2], il essaie de chercher un autre chemin vers le core. S'il trouve que ce nouveau chemin ne passe pas par un de ses fils[3], il envoie un message Join au core; rien ne change au niveau des branches de l'arbre situées au dessous de lui. Maintenant, s'il trouve que ce nouveau chemin passe par un de ses fils, il envoie un message **Flush** vers le bas. Ce message se propage jusqu'à détruire tout l'arbre et chaque membre fait ensuite joindre le core suivant son chemin optimal. Ce dialogue entre les routeurs réduit beaucoup la tâche du core qui ne répond qu'aux messages envoyés par ses voisins. Cependant, il complique les routeurs et peut induire des chemins non optimaux entre les membres et le core surtout s'il y a des routeurs dans le réseau qui ne comprennent pas CBT.

Si une source désire émettre au groupe sans être un membre, donc sans recevoir du trafic, elle encapsule ses données dans des paquets et les envoie dans un message **Register** en point à point au centre qui les diffuse sur l'arbre partagé. Si le centre trouve que le trafic généré par cette source dépasse un certain seuil, il lui envoie un Join. L'acquittement de ce Join par la source établit une branche entre eux. La source cesse alors d'encapsuler ses données et les envoie à l'adresse du groupe. Dans ce cas, les membres situés entre elle et le centre les reçoivent avant qu'elles atteignent le centre.

Le multicast par serveur en ATM utilise des VC point-to-point entre chaque émetteur au groupe et le serveur et un VC point-to-multipoint basé sur le serveur et ayant comme feuilles les récepteurs. Un nouveau récepteur utilise le LIJ pour se rattacher au VC point-to-multipoint et une nouvelle source établit un VC avec le serveur. Les deux n'ont qu'à connaître l'adresse ATM du serveur associé au groupe multicast.

Bien qu'un seul arbre est utilisé, cette approche possède plusieurs problèmes, en plus de ceux du Center Based Tree:

1. Ce n'est pas un vrai Shared Tree. Un VC et donc une entrée sont demandés pour chaque source ce qui provoque presque le même problème que le VC-mesh[4].

2. Les cellules des messages des différentes sources ne peuvent pas être mélangées au serveur ce qui limite l'efficacité du multiplexage statistique sur le VC point-to-multipoint.

3. Dans le cas d'un membre émetteur et récepteur, les messages qu'il émet sur le VC point-to-point sont reçus sur le VC point-to-multipoint. Une solution est de séparer un récepteur de l'arbre lorsqu'il devient une source et de lui envoyer le trafic multicast des autres sources sur son VC point-to-point. Une telle solution consomme plus de bande passante.

Le Centre Based Tree résout les problèmes d'un arbre par source mais comme toute architecture concentrant les opérations et le trafic en un seul point, il a plusieurs inconvénients qui limitent sa robustesse et son évolutivité. Ces problèmes sont détaillés dans la section suivante.

3.2 Les limitations d'un seul centre

La défaillance du centre unique détruit tout l'arbre et oblige les membres à joindre un autre, ce qui fait arrêter la session et créer des états transitoires. Le choix du centre affecte

2. Le père répond tant qu'il y a une route unicast entre les deux routeurs.
3. Les routeurs CBT qui se trouvent au dessous de lui sur l'arbre.
4. Le VC-mesh consomme plus de ressources parce que ses VC point-to-multipoint traversent plus de commutateurs.

beaucoup la performance de la session surtout lorsque la taille du réseau augmente. Si le groupe est localisé dans une certaine zone et si le centre est situé loin de ses membres, le délai rencontré sera très grand ainsi que la bande passante réservée. C'est difficile de trouver un emplacement du centre qui convient toujours à un vaste réseau et à des groupes dynamiques.

L'état stocké dans le centre et le travail qu'il est demandé d'accomplir deviennent importants dans le cas des grands groupes. Dans PIM-SM par exemple, ce centre doit répondre à tous les Join des membres et garder un état pour chaque source pour pouvoir contrôler son trafic.

Dans un réseau comme l'Internet formé de plusieurs domaines et plusieurs niveaux hiérarchiques, il est important de mettre en place un protocole qui assure un multicast à plusieurs portées et qui connecte les membres par un arbre partagé tout en limitant les informations de contrôle à la zone dans laquelle se trouvent ces membres. Il doit être aussi compatible avec les protocoles multicast qui existent déjà dans les domaines (DVMRP, MOSPF, PIM-DM). Un protocole à un seul centre pose aussi un problème administratif dans le choix du centre. Un domaine donné est obligé de se connecter au centre pour joindre le groupe et une fois qu'il le joint, il sera capable de recevoir tout le trafic envoyé partout dans le monde. Aussi, l'annonce des informations sur ce centre dans tout le réseau est un peu difficile vue la diversité des protocoles de routage adoptés dans les différents domaines.

Il faut faire quelque chose pour rendre le Center Based Tree valable pour un grand réseau et capable de répondre à tous les besoins comme la robustesse, l'évolutivité, la limitation de la portée d'un groupe, l'optimisation de l'utilisation des ressources et le respect des contraintes temporelles.

3.3 L'adoption d'une hiérarchie de centres

L'approche était de placer dans le réseau plusieurs centres au lieu d'un seul. Un membre se connecte au centre le plus proche ce qui lui permet de communiquer avec les autres membres à proximité avec un faible délai. Si ce centre ne connaît pas le groupe, il le cherche dans les autres et donc, un backbone sera établi pour lier les arbres des différents centres.

Cette duplication de centres résout les problèmes déjà cités mais elle crée d'autres. Elle réduit l'impact sur la performance d'un mauvais placement du centre unique. Elle distribue la concentration du trafic et des opérations sur plusieurs points ce qui optimise l'utilisation des ressources et allège la tâche d'un centre. Aussi, elle permet à des groupes d'exister localement.

En revanche, elle crée le problème de construction du backbone. La solution est simple pour un membre qui n'a qu'à connaître le centre le plus proche de lui. Mais un centre qui décide de chercher le groupe dans le réseau aura plusieurs choix. Si on lui demande de joindre le centre le plus proche, on risque de créer des chemins pas de tout optimaux. Il faut lui dire explicitement de joindre un certain noeud du réseau qui se charge de connecter son arbre à ceux des autres centres.

Le choix de ce noeud a été résolu par CBT [16] à l'aide d'un mécanisme d'élection entre les cores. Le core élu sera promu à un niveau supérieur. Tous les cores qui ont des membres se connectent au chef (Figure 3.1). L'initiateur du groupe joint ce chef pour annoncer son groupe à tout le réseau.

Cette organisation des cores en deux niveaux améliore la performance de l'arbre entre les membres connectés par un core donné mais ne suffit pas pour un vaste réseau où le chef pourrait être loin de deux cores désirant se connecter. En fait, dans un large réseau, le problème de connexion des cores est identique à celui de connexion des membres dans le

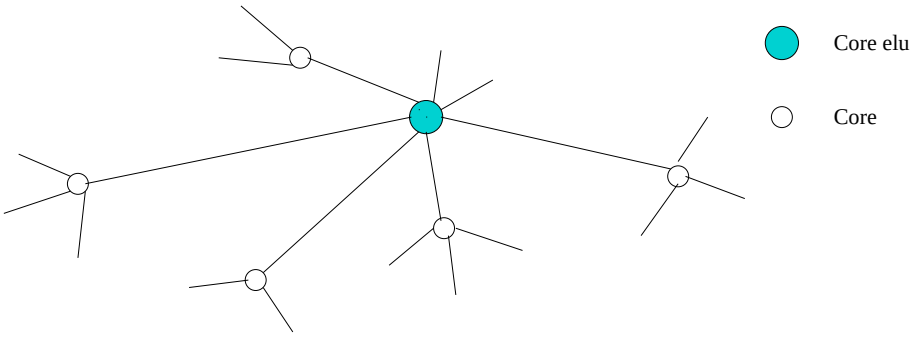

FIG. 3.1 – *Construction du backbone en CBT*

cas d'un seul core. Il faut de nouveau un arbre partagé à plusieurs centres pour connecter les arbres donc une promotion de quelques centres à un niveau plus haut puis une autre élection d'un chef. Si on répète cette élection plusieurs fois, on aura à la fin un backbone de centres sous forme d'un arbre hiérarchique à plusieurs niveaux. Le nombre des centres à placer dans un réseau, le nombre des niveaux de cet arbre et la distribution des centres sur chaque niveau, sont liés à la taille du réseau et à l'emplacement des membres du groupe.

L'assignation des niveaux aux centres peut être faite d'une façon statique ou dynamique par élection. Un centre annonce son identité aux noeuds qui l'entourent. Si on suppose que notre arbre est formé de **N** niveaux avec n_i centres par niveau i, la relation suivante doit être satisfaite:

$$n_1 > n_2 > \cdots > n_N = 1$$

Étudions maintenant le mécanisme de construction du backbone qui est capable de répondre aux exigences d'un protocole évolutif. Tout d'abord on prend le cas d'un réseau n'ayant aucune organisation hiérarchique qu'on appelle réseau **plat**. Un tel réseau représente bien les domaines de l'Internet [5] et les Peer Group d'un réseau ATM. Ensuite, on prend le cas d'un grand réseau organisé hiérarchiquement par le routage unicast. Les différentes propositions faites pour implémenter cet arbre hiérarchique seront étudiées dans les sections suivantes.

3.4 Construction du backbone dans un réseau plat

Les centres des différents niveaux peuvent se trouver partout dans le réseau. Le mécanisme de construction doit donner naissance à un arbre multicast parfait qui satisfait aux exigences déjà mentionnées. La performance de l'arbre doit être fonction de l'emplacement des membres du groupe. Un groupe local doit consommer moins de bande passante et rencontrer moins de délai qu'un groupe épars. Aussi, ce mécanisme doit donner à un groupe la possibilité d'avoir plusieurs portées. La limitation de la portée permet une bonne utilisation de l'espace d'adressage puisque la même adresse pourra être utilisée en même temps dans plusieurs zones du réseau. Elle est aussi un besoin de sécurité et d'authentification.

Ceci sera possible si les zones desservies par les centres d'un niveau X ne se chevauchent pas. Un centre sera donc le point d'accès des membres de sa zone au reste du groupe et ces membres pourrons communiquer entre eux par un arbre bâti complètement à l'intérieur de cette zone. On sera alors capable de créer un groupe à l'intérieur d'une zone et d'interdire

5. Le réseau d'un Internet Service Provider par exemple.

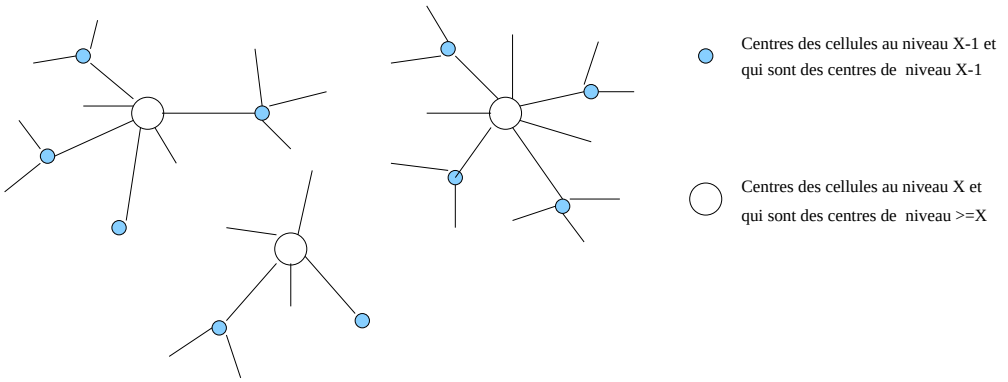

FIG. 3.2 – *Décomposition d'un réseau en des cellules au niveau X*

aux autres régions du réseau de le joindre si on demande à ce point d'accès d'arrêter tout le trafic multicast sortant. Puisque le nombre de centres à un niveau X est plus grand que celui à un niveau $Y > X$, le nombre de cellules à X est alors plus grand que celui à Y et la performance de l'arbre (taille et donc délai moyen) est mieux pour des petits groupes localement placés.

Pour répondre à ces questions, on demande à un membre de joindre le centre le plus proche de lui quelque soit son niveau. Un centre d'un niveau X joint celui le plus proche à condition que son niveau soit plus grand que X. À un niveau X, le réseau plat sera divisé en des cellules séparées centrées sur les centres de niveau $\geq X$ et chacune d'elles (A par exemple) est composée des cellules du niveau $X - 1$ dont les centres sont plus proches du centre de cette cellule (A) que des centres des autres (Figure 3.2). Au niveau le plus bas, on trouve les cellules de base centrées sur tous les centres de tous les niveaux. Chacune d'elles est composée des noeuds du réseau les plus proches de son centre.

Le mécanisme précèdent n'est pas suffisant pour avoir un arbre parfait. Certains problèmes peuvent avoir lieu et des fonctions additionnelles doivent être ajoutées.

3.5 Problèmes d'interconnexion des centres

Deux problèmes peuvent exister lors de la connexion des centres d'un arbre partagé: les boucles et les blocages. Une boucle en multicast est catastrophique. Les routeurs qui en font partie dupliquent le paquet tournant sur leurs sorties chaque fois qu'il traverse la boucle ce qui inonde le réseau par la même information. Cependant, le blocage risque de partager l'arbre multicast et séparer le groupe en plusieurs parties.

Normalement (ce qui est le cas de CBT), lorsqu'un membre ou un centre désire joindre un autre centre, son Join s'arrête au premier noeud rencontré sur l'arbre où il sera acquitté. Un chemin s'établit entre ce noeud et l'appelant.

Ceci marche bien en cas d'un seul centre mais lorsque la racine d'un arbre décide de se connecter à un centre, son Join peut rencontrer une des branches de son arbre. S'il s'arrête en ce point, une boucle se créera (Figure 3.3).

Maintenant, si le chemin entre les deux centres passe par une des branches de l'arbre, le Join se bloquera au premier noeud. Il n'y aura pas de boucles dans ce cas mais l'arbre basé sur le centre appelant sera séparé du reste du groupe (Figure 3.4).

Ces deux problèmes sont dus au fait qu'on n'a pas distingué entre les Join des centres des différents niveaux et qu'on n'a pas modifié les branches déjà établies. Supposons qu'on distingue entre les Join mais on laisse les branches inchangées. Une solution pourrait être

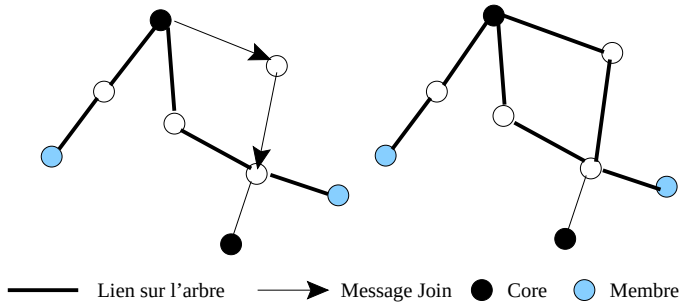

FIG. 3.3 – *Création d'une boucle*

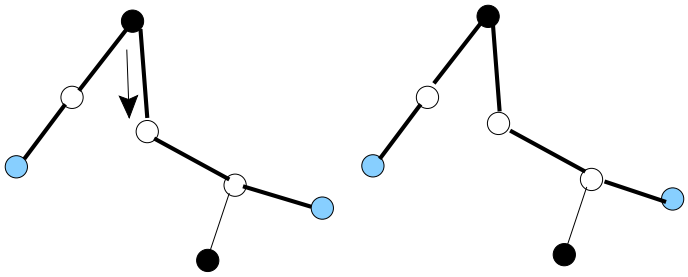

FIG. 3.4 – *Création d'un blocage*

de dire à un noeud déjà traversé par un Join de renvoyer au centre son Join et de lui demander de trouver un autre chemin. Mais ce noeud rencontré peut être sur l'arbre d'un autre centre ce qui ne cause pas de boucles et il faut mieux acquitter le Join. Aussi, il peut se passer qu'il n'y a pas d'autres chemins ou bien il faut un longtemps au centre pour qu'il le trouve ce qui provoque une séparation des membres qu'il sert du reste du groupe.

Pour assurer toujours la connectivité de l'arbre, le noeud qui reçoit un Join d'un centre et trouve qu'il est sur une branche de niveau plus faible que celui de la branche à établir, fait avancer ce message vers la destination mais se déconnecte du centre auquel il est lié et s'accroche à la nouvelle branche. Le résultat est que les membres qui se trouvent au dessous de ce noeud seront connectés directement à cette nouvelle branche (Figure 3.5). Ce mécanisme assure une connectivité des membres du groupe avec absence totale des boucles comme il est prouvé dans [15].

Pour marquer les branches de l'arbre, on utilise le niveau du centre qui a créé la branche. Un Join porte toujours le niveau du noeud qui l'a émis (0 en cas d'un membre) et marque la branche qu'il établit par ce niveau. C'est cette priorité donnée à certaines branches sur d'autres qui sert pour la résolution des problèmes. Une branche coupée par une autre plus

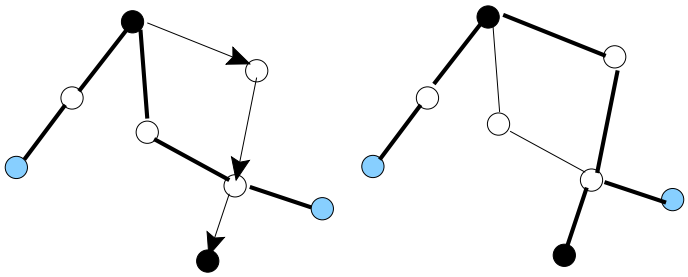

FIG. 3.5 – *Coupure d'une branche*

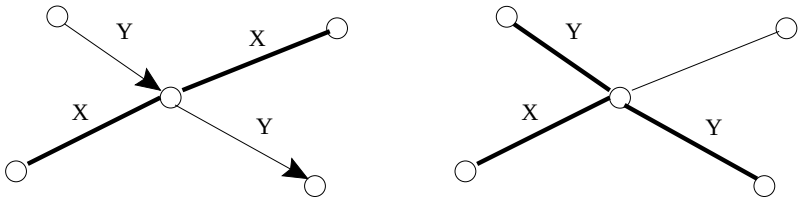

FIG. 3.6 – *Les niveaux des branches (Y>X) au point de rencontre*

prioritaire conserve son niveau initial (Figure 3.6).

3.6 Placement des centres dans un réseau hiérarchique

Pour réduire la taille des tables de routage et la complexité du calcul des routes, un grand réseau est organisé d'une façon hiérarchique. On trouve des domaines interconnectés par un backbone pour former un autre domaine de niveau plus haut et ainsi de suite. Un noeud du réseau a une vision complète de la topologie de son domaine[6] mais une vision agrégée des autres. Dans l'Internet, il s'agit des Systèmes Autonomes (AS) interconnectés par un backbone de liens entre les routeurs de bord. Chacun de ces AS est composé de plusieurs niveaux de réseaux et de sous réseaux jusqu'à arriver aux hôtes. En ATM, la hiérarchie est logique et elle est bâtie automatiquement par le protocole de routage **PNNI**[7] [2]. L'ensemble des commutateurs est divisé en des Peer Group (PG). Chaque PG élit un de ces noeuds comme Leader pour le représenter au PG du niveau supérieur. Ceci continue jusqu'à ce qu'on arrive à un seul PG au niveau le plus haut.

Un certain arbre multicast est demandé pour interconnecter les membres d'un groupe étalé sur plusieurs domaines. L'arbre partagé est la meilleure car dans les backbones et les niveaux supérieurs de la hiérarchie, les ressources sont rares[8] et les deux arbres partagé et spécifique à la source produisent presque la même concentration du trafic sur les liens [24]. Puisque cette concentration est l'inconvénient majeur des arbres partagés, sa disparition les rendent les optimums. Plusieurs centres de plusieurs niveaux sont nécessaires pour construire cet arbre et on peut profiter de la hiérarchie du réseau pour les placer. À un niveau donné du réseau, on associe à chaque domaine un centre. Ce centre agrège le trafic multicast envoyé et reçu par les membres du domaine et constitue leur point d'accès au groupe. Au niveau plus haut, on aura plusieurs centres qu'on peut les interconnecter par un autre arbre partagé à plusieurs centres si c'est nécessaire. La racine de cet arbre constitue le point d'accès au niveau plus haut et ainsi de suite.

Dans le cas d'un domaine à un seul routeur externe, le placement d'un centre n'apporte pas des grandes choses puisque le trafic multicast est naturellement agrégé en ce routeur. Au contraire, le placement d'un centre est bénéficiant dans le cas où plusieurs points d'accès des membres d'un domaine au monde extérieur existent. On est sûr qu'avec ce centre le trafic rentre une seule fois dans le domaine et par la même route.

Pour avoir une bonne agrégation du trafic, il suffit donc de respecter la hiérarchie du réseau pour placer les centres. Avec ces centres comme points d'accès au reste du groupe, on peut tourner à l'intérieur des domaines les protocoles de routage multicast existants ce qui est le but du Routage Multicast Hiérarchique.

6. Les distances à tous les noeuds dans un routage Distance Vector et les informations sur tous les liens et les noeuds dans un routage Link State.

7. Private Network-Network Interface.

8. Les liens peuvent avoir une grande capacité mais c'est le degré de connectivité des domaines qui n'est pas important.

Les protocoles de l'Internet qui utilisent le Center Based Tree (PIM-SM, CBT) sont parfaitement compatibles avec l'approche hiérarchique. Le core ou le RP agrège le trafic du domaine et le passe au centre de l'arbre. Le problème est avec ceux qui utilisent un arbre par source comme DVMRP, MOSPF et PIM-DM.

Tels protocoles sont dits **Sender-Initiated** car ce sont les sources qui prennent l'initiative et annoncent, par diffusion de leur trafic multicast, leur existence au groupe. Un récepteur réagit seulement à cette diffusion et choisit les sources qui l'intéressent. En plus, un routeur utilise le Reverse Path Forwarding (RPF) pour avancer le trafic reçu sur une interface. Un paquet ne sera pas dupliqué que s'il arrive sur le chemin le plus court vers sa source. Donc le centre placé dans un domaine doit tout d'abord s'assurer que le test RPF ne va pas empêcher les paquets d'une source située à l'extérieur d'atteindre tous les noeuds. Aussi, il faut qu'il annonce par diffusion cette source aux récepteurs du domaine. Mais la difficulté est que le Center Based Tree est une approche **Receiver-Initiated**. Un centre doit recevoir au moins une demande d'un membre pour qu'il joigne le niveau plus haut et donc pour qu'il reçoive le trafic des sources externes qui émettent au groupe. Telles sources ne sont pas connues par le centre dans l'absence des demandes explicites des membres. Il faut mettre en place un mécanisme pour annoncer une nouvelle source aux domaines adoptant la diffusion. Puisque les routeurs de bord constituent les points d'accès au reste du réseau, le test RPF ne peut pas réussir que si un d'eux diffuse le trafic. Donc, c'est le rôle du centre de chercher ce routeur et de lui envoyer en unicast les informations à diffuser. Pour faciliter la résolution de ces deux problèmes, il faut mieux choisir un des routeurs de bord comme centre d'un domaine utilisant un protocole Sender-Initiated.

En donnant des niveaux logiques aux centres, on peut associer à une adresse multicast un certain niveau de la hiérarchie et par suite limiter la portée du groupe utilisant cette adresse à la zone desservie par un centre de ce niveau. Un centre d'un niveau X doit refuser toutes les demandes de connexion à un groupe ayant une adresse associée à un niveau $< X$.

La question qui se pose maintenant est comment interconnecter un grand nombre de noeuds à un niveau donné par un arbre partagé hiérarchique. Autrement dit, combien de centres et de niveaux il faut placer dans un domaine donné pour avoir un arbre multicast de meilleures performances et un seul point d'accès au monde extérieur. Aussi, on a dit qu'il faut mieux suivre la hiérarchie du réseau pour placer les centres. Dans ce cas, il faut trouver la meilleure organisation hiérarchique de l'ensemble des noeuds du réseau qui améliore la performance du multicast. Pour un réseau déjà organisé, si on trouve que le multicast requiert une modification de l'existant, il faut essayer de trouver un compromis entre le gain apporté par le multicast et le coût à payer pour la modification. La solution à ces deux problèmes constitue le but des simulations faites dans ce travail et détaillées dans le chapitre suivant.

Avant de répondre à ces questions, on va présenter, dans la suite de ce chapitre, les propositions faites pour implémenter cet arbre multicast hiérarchique dans les deux types de réseaux les plus connus, Internet et ATM. L'étude de ces propositions nous permet d'expliquer l'intérêt de nos modèles de simulation. Des solutions à certains problèmes sont aussi proposées. À la fin, le problème de dimensionnement des arbres multicast est présenté.

3.7 Routage Multicast Hiérarchique dans l'ATM

L'utilisation des arbres partagés hiérarchiques pour faire du multicast dans un large réseau ATM a été proposée dans [4]. Cette proposition utilise la hiérarchie du PNNI pour placer les différents cores. Dans tous les PG et à tous les niveaux, un des noeuds est choisi comme **core** selon un certain critère, par exemple le noeud de bord ayant le degré le plus élevé et qui est donc proche d'un grand nombre de PG (Figure 3.7).

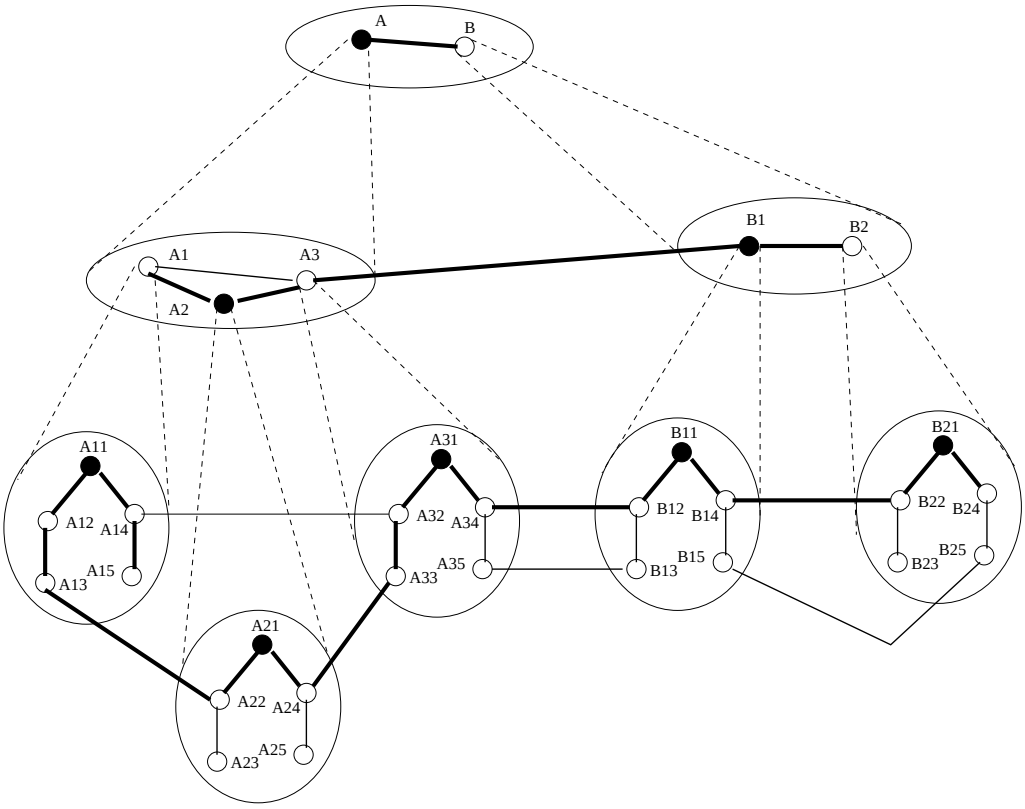

FIG. 3.7 – *Placement des cores dans un réseau ATM*

Dans PNNI, chaque PG de n'importe quel niveau est représenté au niveau plus haut par un noeud logique. Ce noeud agrège les informations sur le PG qu'il représente et les passe à ses voisins. Il récupère aussi les informations sur ses voisins et les niveaux supérieurs et les passe vers le bas. Un noeud logique peut profiter de cette diffusion pour annoncer à ses fils les identités des cores choisis dans les PG en haut. À tout instant, un noeud du réseau possède dans sa table de routage les identités des cores choisis. Le mécanisme utilisé dans PNNI pour élire le Leader d'un PG peut aussi servir dans le choix du core.

Un nouveau membre qui désire joindre le groupe envoie son Join au core de son PG et une connexion s'établit entre eux. À son tour, ce core joint celui du niveau plus haut en cas où il ne connaît pas le groupe. Ceci se répète jusqu'à atteindre le core du plus haut PG. Le résultat est un arbre multicast à chaque niveau de la hiérarchie. Lorsqu'un core d'un niveau X joint celui du niveau $X+1$, une branche s'ajoute à l'arbre au niveau $X+1$. Elle se traduit par de nouvelles branches aux niveaux plus bas. Un lien entre deux noeuds logiques qui s'ajoute à l'arbre à un certain niveau, donne naissance au niveau plus bas à une branche entre les deux cores des PG représentés par ces deux noeuds comme le montre la figure 3.8.

À la fin, les membres dans chaque PG au niveau physique seront connectés par un arbre partagé basé sur son core. La connexion de ces différents arbres se fait par un autre arbre partagé basé sur les cores mêmes. Ce dernier arbre constitue le backbone et il est formé par les liens externes entre les PG et quelques liens internes. En réalité tous les cores du plan physique sont au même niveau mais c'est la hiérarchie logique qui donne des priorités à certains cores sur d'autres.

L'utilisation de la hiérarchie du PNNI pour dimensionner l'arbre multicast rend le protocole évolutif avec la taille du réseau. De même, l'utilisation des cores le rend capable

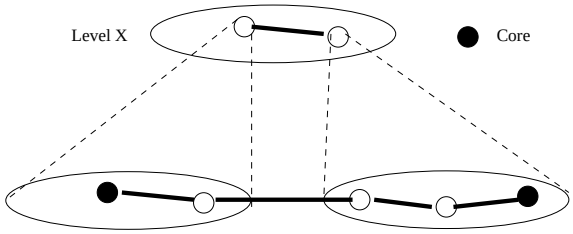

FIG. 3.8 – *Le mappage entre les branches de deux niveaux adjacents*

de supporter des groupes dynamiques. Un membre joint le groupe et le quitte sans connaître les autres. Il ne s'adresse qu'à une seule destination pour chercher tout le trafic multicast. Un tel schéma a bien respecté le service multicast de base.

Cependant, cette proposition a des problèmes. Tout d'abord, l'implémentation d'un arbre partagé est difficile en ATM. Il nécessite un VC multipoint-to-multipoint sur lequel les différents membres peuvent émettre et recevoir. La difficulté est dans le mélange du trafic de plusieurs sources sur une même connexion de sortie (VC Merging). Pour qu'un message puisse être interprété correctement à la destination, un noeud intermédiaire ne doit pas mélanger ses cellules avec les celles des autres messages. Cette condition limite beaucoup l'efficacité du multiplexage statistique en ATM et rend difficile le respect de la QoS[9] demandée par les membres. Une autre cause qui complique la garantie de QoS est le partage même de cette ressource commune entre les membres. Cependant, la version 2 du protocole PNNI [3] a introduit la notion du VC multipoint-to-point. Ce VC permet à plusieurs destinations d'envoyer des informations à la racine sur un même circuit. Le mécanisme de mélange des messages adopté par ce VC, combiné à celui de duplication des messages utilisé par un VC point-to-multipoint, rendent possible l'implémentation d'un arbre partagé en pratique.

Un autre problème est qu'elle considère les noeuds et les liens logiques comme étant des entités physiques. Des Join sont envoyés entre un core logique et celui du niveau plus haut et des arbres sont bâtis à chaque niveau. Mais en PNNI, les noeuds logiques servent seulement pour l'échange des informations de routage pas pour le trafic usager. Aussi, il est impossible pour un noeud de garder dans sa table de routage les informations nécessaires sur un arbre par niveau et par groupe. Un noeud physique peut connaître seulement s'il est sur l'arbre ou non. Tout ce qu'un noeud peut faire lorsqu'il décide de joindre le groupe est d'envoyer sa demande au core où il est sûr de joindre les autres membres. Le message s'arrête au premier noeud rencontré sur l'arbre. Il faut donc une signalisation qui, en utilisant les informations sur les cores, construit l'arbre suivant le schéma proposé. Le problème de boucle n'a pas été aussi traité. Pour le résoudre, cette signalisation doit utiliser le mécanisme de coupure des branches déjà étudié.

Le choix d'un des noeuds logiques comme core veut dire que l'ensemble des PG au dessous de ce noeud est un core du réseau au niveau considéré. À un niveau donné, le réseau est donc divisé en des zones dont une est choisie comme core. Toutes les demandes de connexion au groupe doivent être envoyées à ce core. Cette zone core est à son tour divisée en plusieurs sous-zones dont une est choisie comme core (Figure 3.9). La demande qui entre dans une zone doit être acheminée vers sa sous-zone core. Ceci continue jusqu'à arriver à la racine de l'arbre. Cette racine (A.2.1 dans le réseau de la figure 3.7) est trouvée en descendant à partir du core du PG le plus haut vers le plan physique tout en passant par les cores logiques. Donc un nouveau membre doit envoyer sa requête à ce noeud mais

9. Quality of Service.

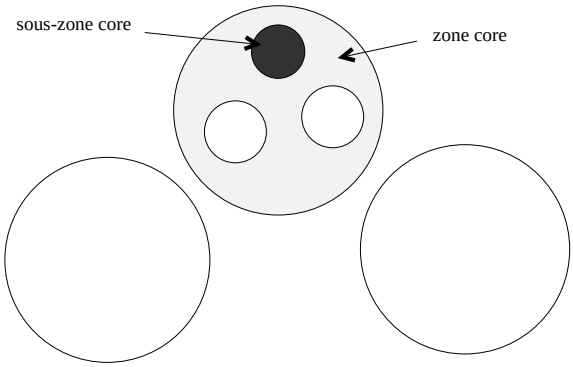

FIG. 3.9 – *Décomposition d'un réseau en zones à un certain niveau*

en passant par les zones et les sous-zones cores.

PNNI [2] utilise le routage à la source (Source Routing). Un noeud qui désire établir une connexion avec une destination calcule, suivant sa vision de la topologie du réseau, la route complète et l'ajoute à son message SETUP. Le plus court chemin pour traverser un PG à un certain niveau est calculé et la liste des noeuds obtenus constitue un élément dit DTL[10]. Un noeud à l'intérieur d'un PG fait avancer le message suivant la route portée. Si un nouveau PG est rencontré, le premier noeud ajoute à la route les DTL nécessaires pour traverser ce PG et ceux qui ne sont pas visibles par les derniers noeuds qui ont fait un calcul de route. Ceci continue jusqu'à ce que la destination du message soit atteinte.

En implémentant ce nouveau schéma pour le multicast, il faut essayer d'être compatible avec la procédure d'établissement d'une connexion classique. Cette compatibilité permettra aux noeuds qui ne supportent pas cette nouvelle capacité multicast de fonctionner correctement.

3.7.1 Implémentation du schéma

Dans notre implémentation, tous les nouveaux membres envoient leur demande au plus haut core logique. La différence avec l'appel classique est dans le calcul du chemin à l'intérieur d'un PG. Ici, ce n'est pas le plus court chemin pour traverser un PG au suivant qui est calculé. Deux chemins sont cherchés: un qui va au core du PG et un autre qui va du core à la destination. Les deux chemins sont ensuite concaténés dans le même élément DTL.

Un noeud intermédiaire ainsi qu'un core fait avancer le message suivant la route portée s'il trouve qu'il n'est pas sur l'arbre du groupe. Sinon, il l'arrête et envoie la réponse CONNECT à la source pour établir la connexion entre eux. En s'adressant à la même destination, on est sûr que tous les messages SETUP vont converger à la racine de l'arbre où ils s'arrêtent. Le groupe reste toujours connecté. Pour comprendre ce mécanisme, illustrons le par un exemple sur la figure 3.7.

En partant d'un groupe vide, le noeud A.1.5 envoie sa demande d'adhésion au core logique A ce qui assure une annonce du groupe à tout le réseau. Cette demande doit s'arrêter à la racine de l'arbre A.2.1. La route calculée par A.1.5 renferme trois DTLs, un pour chaque niveau. Le DTL de niveau 1 donne le chemin à suivre pour aller au PG(A.2) et qui passe par le core du PG(A.1). Son contenu est A.1.5 - A.1.4 - A.1.1 - A.1.2 - A.1.3. Le DTL de niveau 2 contient le chemin A.1 - A.2 et le DTL de niveau 3 contient seulement le noeud A.

10. Designated Transit List.

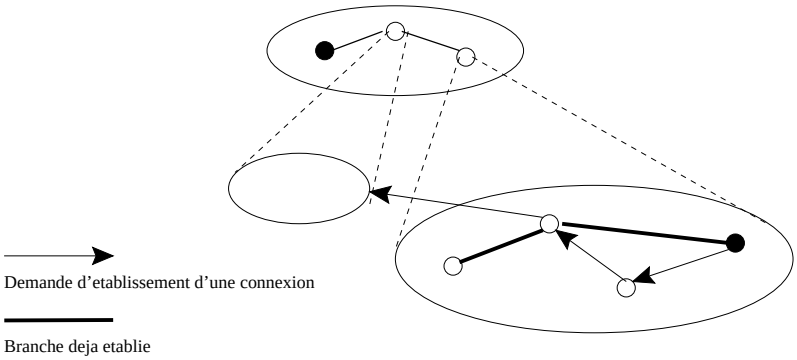

FIG. 3.10 – *Problème de boucle en ATM*

En arrivant à A.2.2, le DTL de niveau 1 est remplacé par un autre contenant le chemin vers le PG suivant. Mais puisque on est dans le PG de destination, seulement le chemin vers le core est calculé. Le message arrive à A.2.1 où il s'arrête et le message CONNECT est renvoyé à la source.

Maintenant, un nouveau membre B.2.4 décide de joindre le groupe. Il prépare la même demande. Son DTL de niveau 1 porte le chemin B.2.4 - B.2.1 - B.2.2. Le DTL 2 porte le chemin B.2 - B.1 et le DTL 3 le chemin B - A. En B.1.4, le DTL 1 est remplacé par B.1.4 - B.1.1 - B.1.2. Au point A.3.4, le DTL 2 expire sans avoir trouvé l'arbre. Donc le message continue son chemin vers la racine. Les nouvelles valeurs des DTL au point A.3.4 seront:

DTL 1 : A.3.4 - A.3.1 - A.3.2
DTL 2 : A.3 - A.2
DTL 3 : B - A

Au point A.2.4, le DTL 1 est modifié pour qu'on puisse aller vers la racine A.2.1 où on rencontre l'autre membre qui a déjà joint le groupe.

Cette implémentation est avantageuse puisqu'elle minimise le trafic de contrôle et respecte la demande d'établissement d'une connexion classique. Un noeud ne demande que les identités des cores choisis au dessus de lui. L'arbre multicast, comme il est proposé, est établi automatiquement sans aucun dialogue entre les cores des niveaux adjacents et sans avoir besoin de garder des informations sur un arbre par niveau.

3.7.2 Détection des boucles

Comme dans le cas d'un réseau plat, cette construction peut créer des boucles. Le problème se pose lorsqu'un core ayant déjà des membres connectés à lui décide de joindre le groupe et lorsque son message rencontre un des noeuds de son arbre (Figure 3.10). Mais puisque le message SETUP porte la route déjà traversée, un noeud peut détecter facilement, par un simple test des DTL, l'existence d'une boucle. Ceci aura lieu si le message est passé par un des cores qui se trouvent entre ce noeud et le niveau le plus haut. En cas de boucle, la solution est d'avancer le message suivant la route qu'il porte et d'envoyer un RELEASE au core du PG pour libérer la connexion. Le sous arbre basé sur ce noeud sera connecté directement à la nouvelle branche. Le RELEASE s'arrête au premier noeud ayant plus que deux liens incidents sur l'arbre.

Puisque la route dans un PG est calculée par un seul noeud et puisque le core du PG n'introduit aucune modification, le cas d'une boucle créée par une même connexion disparaît. Le noeud qui fait le calcul des DTL détecte l'existence de cette boucle et prend

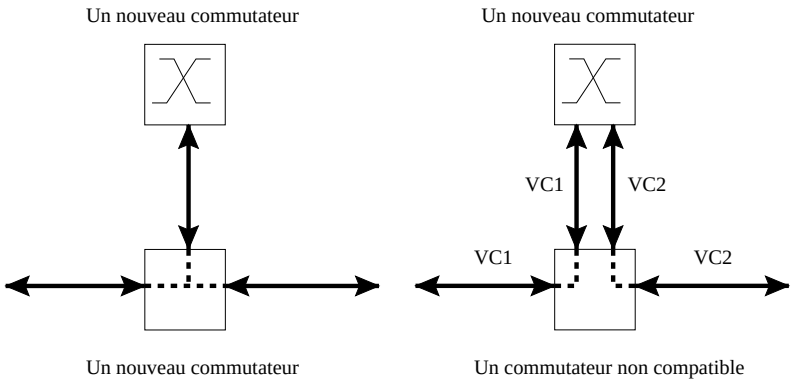

FIG. 3.11 – *Cas des commutateurs non compatibles*

les précautions nécessaires. Par exemple, il peut prévoir les conséquences d'un RELEASE et mettre la route résultante directement dans le message. Aussi, cette prévision peut résoudre le problème de blocage. Par exemple, dans le réseau de la figure 3.7, lorsque le noeud B.2.3 décide de joindre le groupe, il trouve que le chemin au PG(B.1) passe à B.2.1 puis retourne à B.2.2. Il peut dans ce cas mettre directement comme DTL 1 le chemin B.2.3 - B.2.2.

3.7.3 Cas des commutateurs non compatibles

L'implémentation de cette nouvelle capacité multicast dans les commutateurs ne peut pas se faire d'une façon instantanée. Pendant la phase transitoire, certains commutateurs non compatibles peuvent exister. Si un tel commutateur reçoit le message SETUP d'une connexion multicast, il va voir en lui une demande d'établissement d'un VC classique. Il le laisse passer après avoir ajouté une entrée à sa table de commutation comme le montre la figure 3.11(cette entrée contient les couples VCI/VPI d'entrée et de sortie du VC1). Lorsque ce commutateur reçoit une nouvelle demande, il la considère de nouveau comme une demande classique. Il ajoute une nouvelle entrée à sa table (création de VC2). Ces deux entrées peuvent pointer vers un même lien physique.

Dans ce cas, lorsqu'une cellule envoyée par un membre au groupe arrive au commutateur non compatible sur un des VC établis (par exemple VC1), il la commute sur la sortie de ce VC sans tenir compte des autres entrées correspondantes au groupe (VC2). C'est le rôle du premier commutateur supportant le nouveau protocole de corriger cet effet. Un tel commutateur duplique sur tous les VC du groupe une cellule envoyée à un voisin non compatible (sur VC1 et VC2) et il duplique la cellule reçue sur un de ces VC (soit VC1) sur les autres (VC2).

Cette solution, similaire au Tunneling en IP, fait augmenter le délai et consomme plus de bande passante mais elle est nécessaire pour assurer une modification lente des commutateurs.

3.8 Routage Multicast Hiérarchique dans l'Internet

Dans l'Internet, le service Best Effort non connecté rend l'implémentation d'un arbre partagé très facile. Chaque paquet contient, dans l'entête IP, les adresses source et destination et un identificateur de paquet, et dans l'entête TCP un numéro de séquence. Ces informations permettent à un hôte d'interpréter les paquets de plusieurs sources quelque soit leur ordre d'arrivée.

Plusieurs protocoles ont été proposés pour connecter un groupe avec un arbre partagé hiérarchique. On trouve les arbres à deux niveaux utilisés par GUM (BGMP) pour faire du multicast entre les domaines ainsi que les arbres à plusieurs niveaux proposés par HPIM et OCBT. Certains s'intéressent à un niveau du réseau comme GUM et OCBT, d'autres ont essayé de faire quelque chose de global comme HIP et HPIM.

3.8.1 GUM (Grand Unified Multicast)

GUM[11] [18] est un protocole désigné pour connecter les systèmes autonomes (AS) de l'Internet par un arbre partagé avec la possibilité de passer à un arbre par source pour les applications sensibles au délai. Le core de l'arbre est un des domaines et son choix est fonction de l'adresse du groupe. En effet, l'espace des adresses IP multicast est divisé entre les différents domaines et le créateur d'un groupe choisit son adresse parmi celles de son domaine qui sera alors racine de l'arbre. Ceci assure un faible délai pour les domaines qui se trouvent à proximité.

A l'intérieur des domaines, on peut tourner n'importe quel protocole multicast. C'est la responsabilité des routeurs de bord de faire la traduction des protocoles. Lorsqu'un membre cherche un groupe à l'extérieur (son adresse n'appartient pas à celles de son domaine), le routeur de bord est appelé et une table de routage dite Multicast Routing Information Base est consultée pour trouver une route vers le domaine core. Un Join sera envoyé à ce core ce qui créera une entrée bidirectionnelle (*,G) dans les routeurs parcourus. Si un membre désire recevoir le trafic d'une source suivant le plus court chemin, il envoie un Join directement au domaine de cette source ce qui crée une entrée spécifique à cette source (S,G) dans les routeurs. L'entrée (*,G) est plus prioritaire que (S,G) pour ne pas provoquer une duplication des données émises par une source.

Puisque les membres du groupe n'envoient pas leur demande directement au core mais passent par les routeurs de bord, on peut parler d'une hiérarchie à deux niveaux. Le premier niveau de l'arbre est formé par les routeurs de bord et au deuxième niveau, on trouve le domaine core. Ici, la racine de l'arbre est un domaine qui se comporte, grâce au dialogue entre les routeurs, comme un seul noeud ce qui ressemble à l'approche des cores logiques en ATM.

3.8.2 OCBT (Ordered Core Based Tree)

OCBT [16] est une extension du protocole CBT. Il a introduit des solutions aux problèmes d'interconnexion des cores de CBT en les organisant en des niveaux et en utilisant le mécanisme déjà cité pour construire l'arbre multicast. Il s'intéresse à un seul domaine du réseau (par exemple le backbone) et son objectif était la construction d'un arbre parfait sans boucles et sans blocages avec une réparation rapide de l'arbre en cas de défaillance des liens ou des routeurs. Ce dernier point est assuré grâce à une modification du mécanisme de reconstruction de l'arbre de CBT. Le message Flush ne détruit pas tout l'arbre mais s'arrête au premier core rencontré.

Pour éliminer les boucles dues aux régimes transitoires dans les tables de routage unicast, ce protocole met un routeur dans un état de suspension après l'envoi de son Join jusqu'à la réception d'un acquittement de la destination ou du premier routeur CBT rencontré sur l'arbre. Pendant cet état, ce routeur ignore tous les Join qu'il reçoit. Donc il ignore son propre Join lorsqu'il retourne à lui suite à un problème de boucle. Ce message sera retransmis plusieurs fois jusqu'à la disparition du régime transitoire.

11. Il est appelé aussi BGMP (Border Gateway Multicast Protocol).

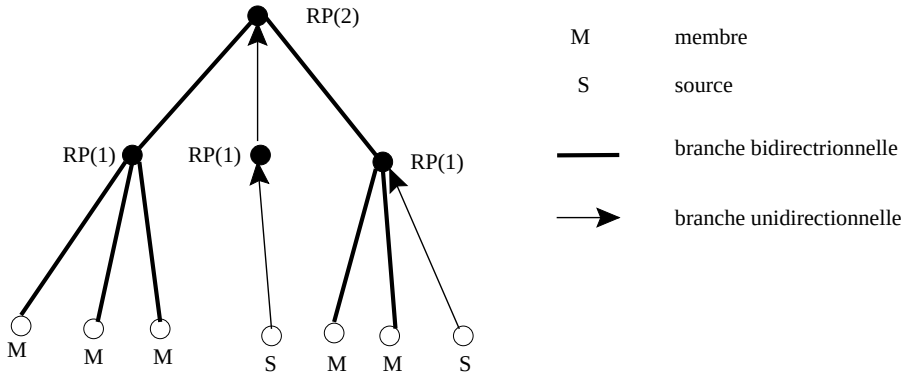

FIG. 3.12 – *Construction de l'arbre multicast dans HPIM*

Puisqu'il considère seulement un réseau plat dans lequel il distribue des cores de différents niveaux, ce protocole n'est pas suffisant pour faire du multicast dans un vaste réseau organisé hiérarchiquement.

3.8.3 Hierarchical PIM

HPIM [17], qui est la version hiérarchique du protocole PIM-SM, s'intéresse à tout le réseau. Il a proposé l'utilisation de plusieurs RPs à plusieurs niveaux et il a adopté PIM-SM entre les membres du groupe et un RP de niveau 1 et entre les RPs d'un niveau X et celui de niveau $X+1$. Cependant, il n'a pas bien expliqué le placement des RPs ni les messages demandés pour construire un arbre parfait.

Les RPs sont placés aux différents niveaux de l'Internet en allant des hôtes vers le backbone entre les routeurs de border. Les RPs de niveau 1 sont placés à proximité des hôtes. Le routeur désigné du réseau local auquel le hôte est connecté cherche dans une liste de RPs de niveau 1 celui qui correspond à l'adresse du groupe désiré. Ceci se fait en appliquant une fonction à cette adresse qui retourne le RP convenable. Il se connecte à ce RP en envoyant périodiquement des Join comme en PIM-SM. Ce RP à son tour se connecte au RP de niveau plus haut par des messages Join (en appliquant toujours une fonction à l'adresse pour trouver le RP de niveau plus haut) jusqu'à atteindre un RP qui est sur l'arbre.

Si une source désire seulement émettre au groupe, elle envoie son message Register qui crée une entrée unidirectionnelle entre elle et le premier RP qui a des entrées bidirectionnelles (Figure 3.12). Sur un chemin unidirectionnel, le trafic passe seulement dans le sens source—->RP.

Lorsqu'un membre crée un groupe, le chemin jusqu'au RP le plus haut sera réservé. Si un RP trouve qu'il n'a qu'une seule interface bidirectionnelle, ceci veut dire qu'il est ou bien le RP le plus élevé et il n'a pas de membres mais il reçoit des Join d'un RP qui se trouve au niveau plus bas ou bien qu'il est un RP feuille. Dans ces deux cas, il n'a pas besoin du trafic multicast. La solution était d'ajouter un nouveau message **Prune** pour arrêter l'arrivée de ce trafic et pour économiser la bande passante (Figure 3.13). Ce Prune se propage jusqu'au premier noeud ayant plus que deux interfaces marquées sur l'arbre où il sera acquitté. Les interfaces des routeurs traversés par ce Prune seront marquées inactives.

Pour le problème de boucle, la solution est un peu différente de celle utilisée dans OCBT. Un Join est acquitté par un RP et non par un routeur et son passage est nécessaire pour rafraîchir les entrées dans les routeurs d'une branche de l'arbre. Ceci pose un problème lorsqu'un routeur sur l'arbre est rencontré. Il faut faire quelque chose pour arrêter la

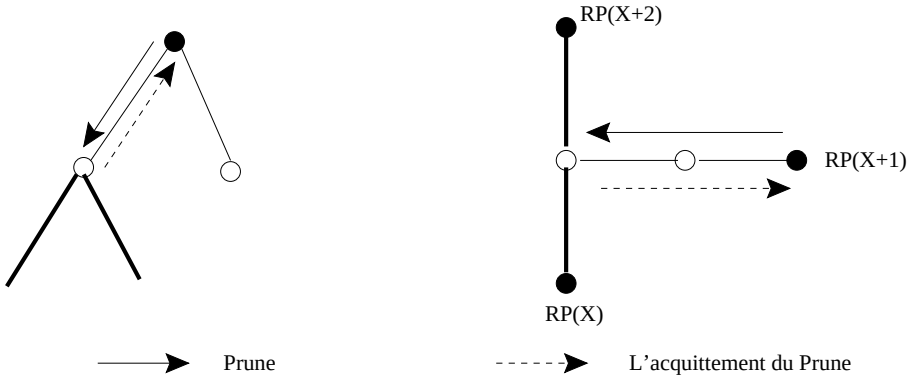

FIG. 3.13 – *Envoi du message Prune par un RP*

nouvelle branche au point de rencontre avec une branche de niveau supérieure ou bien pour couper une branche de niveau plus bas.

Une solution pourrait être l'acquittement du Join par le point de rencontre comme dans CBT (Figure 3.14). Donc, le Join n'arrive pas au RP de destination et ne rafraîchit pas les entrées qui existent sur la partie de la branche à supprimer. Les entrées expireront plus tard et le RP considérera que le noeud qui a envoyé ce Join est maintenant hors du groupe.

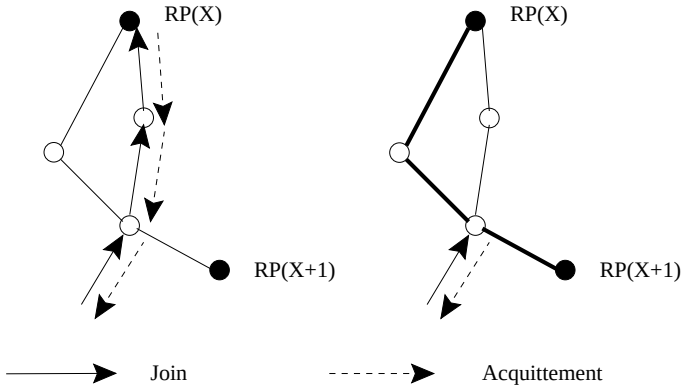

FIG. 3.14 – *Acquittement d'un Join par le point de rencontre*

Mais, puisque tout routeur est susceptible d'être un point de rencontre de deux branches de niveaux différents, cette solution est coûteuse. Elle oblige tous les routeurs du réseau d'être puissants pour pouvoir émuler un RP et répondre à un grand nombre de Join.

Il faut donc ajouter de nouveaux messages au protocole PIM pour rendre inactives les interfaces qui forment la branche à supprimer et ceci malgré le passage des Join. Un routeur ne duplique pas les paquets sur une interface inactive. Une fois que la cause de la boucle disparaît, ces interfaces reviennent à l'état actif.

HPIM marque les messages Join et Register par le niveau du RP qui les a envoyés et il a expliqué ce qui se passe lorsqu'un message Register envoyé par une source rencontre un noeud traversé par un autre Register. Puisque le Register va de bas en haut, un noeud qui n'a pas de membres ne peut pas envoyer le trafic de la source vers deux interfaces de sortie sinon une boucle (si les deux Registers correspondent à la même source) ou bien une duplication du trafic (si les deux sorties amènent à des noeuds différents de l'arbre) aura lieu. Ce noeud cherche si les interfaces de sortie de ces deux Register sont différentes. Si

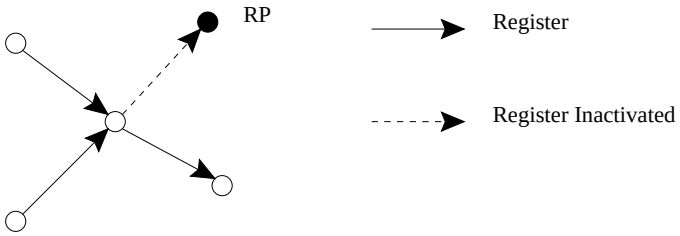

FIG. 3.15 – *Inactivation d'un message Register*

elles sont les mêmes, il avance les deux messages sans rien faire. Sinon, il prend le message qui a le niveau le plus élevé, l'avance et envoie un message **Register Inactivated** sur l'interface de sortie de l'autre comme le montre la figure 3.15. Ce message marque inactive la branche de l'arbre entre ce noeud et le RP suivant.

Cependant, HPIM n'a pas parlé des autres problèmes qui peuvent exister, comme par exemple la rencontre entre un Register et un Join ou bien la rencontre entre deux Join.

Le premier cas peut être résolu en donnant la priorité à une entrée bidirectionnelle sur celle créée par un Register. Le routeur de l'arbre qui reçoit un Register et qui trouve que son interface de sortie n'est pas marquée sur l'arbre, arrête ce message et envoie à sa place un Register Inactivated. Le trafic arrivant de cette source à ce routeur sera dupliqué sur les branches de l'arbre et non pas vers le RP auquel le Register a été destiné.

Le deuxième cas nécessite un autre type de messages pour inactiver une branche de l'arbre créée par un Join. On peut le nommer **Join Inactivated**. Le routeur applique la même fonction de détection des boucles de OCBT pour trouver la branche à supprimer. Les entrées correspondantes au groupe dans les routeurs de cette branche seront rafraîchies par un Join Inactivated qui informe les routeurs de ne pas avancer du trafic multicast sur les interfaces d'envoi et de réception de ce message et de les marquer inactives. Ce nouveau Join s'arrête au premier noeud du réseau (routeur ou RP) qui a des interfaces marquées sur l'arbre. En ce point, un test sera fait pour savoir si on l'avance comme il est ou bien si on le convertit en un Join ordinaire.

On voit bien que HPIM et OCBT adoptent presque le même mécanisme pour construire l'arbre mais HPIM est plus global puisqu'il a parlé de tout le réseau et il est rentré plus dans les détails du protocole en parlant de la signalisation requise. Dans OCBT, un membre peut se trouver proche d'un core de n'importe quel niveau pour cela il lui a donné la possibilité de le joindre directement. HPIM n'a pas parlé de ça puisque les RPs sont placés suivant la hiérarchie du réseau.

Il reste à savoir comment HPIM peut limiter la portée d'un groupe. L'espace d'adressage est réparti entre les différents niveaux. Les RPs d'un niveau donné connaissent qu'ils sont les routeurs de bord d'une bande d'adresses et arrêtent les messages Join et Register portant une de ces adresses. Un RP est donc la racine de l'arbre correspondant à une des adresses de son niveau et limite la portée du groupe à la zone qu'il sert. Un membre qui n'appartient pas à cette zone et qui désire joindre le groupe monte avec son message Join jusqu'au niveau associé à cette adresse. Le RP qui prend son message trouve qu'il est le routeur de bord de ce groupe. Il l'arrête et ne connecte pas ce membre aux autres qui sont dans l'autre zone. Ceci est possible parce que les zones servies par deux RPs d'un même niveau sont géographiquement disjointes. Pour allouer une adresse multicast à un groupe, il suffit de connaître la zone de couverture voulue pour choisir une des adresses du niveau qui limite cette zone.

3.8.4 HIP

C'est une nouvelle proposition [19] pour faire du multicast dans l'Internet. Elle utilise OCBT pour lier, avec un arbre partagé, plusieurs domaines de l'Internet. L'inconvénient de HPIM et OCBT est qu'ils n'ont pas parlé d'autres protocoles et ils ont imposé à tous les routeurs du réseau de parler une seule langue PIM ou CBT. HIP laisse ces domaines mais les émule au niveau plus haut par un **Routeur Virtuel** qui parle OCBT. On aura alors à chaque niveau du réseau, plusieurs Routeurs Virtuels à interconnecter par un arbre OCBT.

Les routeurs de bord d'un domaine élisent un d'eux comme **Coordinateur** qui synchronise leurs opérations pour que le domaine apparaisse au niveau plus haut comme un seul routeur parlant OCBT. Il traite tous les messages externes reçus et prend la décision en fonction des états des différents routeurs de bord. Lorsqu'un membre du domaine décide de joindre le groupe, ce coordinateur choisit un des routeurs de bord comme routeur externe et le charge d'envoyer le Join aux autres routeurs virtuels du niveau plus haut en utilisant OCBT. Lorsqu'un des routeurs de bord reçoit un Join et le coordinateur décide que ce Join doit s'arrêter, des actions sont prises pour connecter ce routeur à l'arbre interne au routeur virtuel. Ceci doit être accompli suivant le protocole multicast qui tourne dans le domaine. Si ce Join doit continuer son chemin, le coordinateur choisit le routeur de bord convenable pour avancer le Join à un autre routeur virtuel.

3.9 Dimensionnement de l'arbre Multicast Hiérarchique

Ayant parlé de l'avantage de la hiérarchie de centres, il faut passer maintenant à l'étude de son dimensionnement. Cette étude doit se baser sur les différentes propositions déjà expliquées. Le but est de trouver, pour un réseau de topologie donnée et pour un groupe de distribution donnée, la variation de la performance de l'arbre en fonction de sa structure. Un arbre hiérarchique possède comme paramètres le nombre des centres, le nombre des niveaux et la distribution des centres sur chaque niveau.

Vue l'importance du volume du trafic généré par les nouvelles applications multicast, une optimisation de l'utilisation des ressources du réseau est demandée. Le fait d'utiliser un arbre partagé optimise le volume d'informations dans les noeuds. Il reste à optimiser la bande passante totale consommée. Cette dernière optimisation, qui est considérée dans notre travail comme le critère pour évaluer la performance du multicast, demande un calcul du meilleur dimensionnement de l'arbre qui minimise la somme des coûts de la bande passante consommée sur ses liens.

D'après ce qu'on a vu, le problème de dimensionnement peut être divisé en deux parties: le cas des réseaux plats et celui des réseaux hiérarchiques. Un réseau plat peut être trouvé dans et entre les domaines de l'Internet et dans les PG de l'ATM. Dans un tel réseau, on a intérêt a trouver le meilleur arbre qui donne un seul point d'accès au monde extérieur. Une fois que ce problème est résolu, l'effet de la hiérarchie du réseau pourra être étudié.

On a dit qu'il faut mieux, comme il est proposé dans [4], placer les centres de l'arbre sur les différents niveaux du réseau. Pour un réseau existant, cette solution est l'optimum mais pour un nouveau réseau à dimensionner, il faut voir l'effet de cette organisation hiérarchique du réseau sur la performance du multicast.

C'est un problème compliqué et les solutions sont particulières à chaque cas. Des modélisations mathématiques peuvent être faites pour trouver des résultats empiriques mais les calculs nécessaires sont très difficiles et imposent beaucoup de simplifications qui risquent de rendre le problème un peu loin de la réalité.

Le but principal de ce travail est de procéder par simulations pour étudier les différents

aspects de ce problème. Avec des simulations, nous sommes capables de voir un grand nombre de cas possibles et d'essayer de trouver, à partir des valeurs moyennes et des variances, des relations générales pour résoudre un tel problème en pratique. Dans la suite, on expliquera les modèles proposés et on parlera des outils utilisés. Les interprétations des résultats obtenus seront élaborées à la fin.

Chapitre 4

Modèles de simulation

Notre but était l'étude de performance des arbres partagés hiérarchiques. On cherche à trouver, par simulations, la meilleure distribution des centres sur les niveaux qui réduit le coût total de la bande passante consommée. Pour que le résultat soit général, le problème a été divisé en deux parties et un modèle a été conçu pour étudier chacune d'elles.

Le premier modèle considère des réseaux plats dans lesquels les centres de l'arbre sont placés sans aucune contrainte de la part du réseau. Un réseau plat peut se trouver tout seul où bien il peut constituer un domaine d'un grand réseau organisé hiérarchiquement.

L'autre modèle prend un grand réseau hiérarchique formé de plusieurs domaines placés sur plusieurs niveaux et distribue les centres suivant cette hiérarchie. Le trafic d'un domaine et de ceux situés au dessous de lui sont agrégés au centre qui constitue le point d'accès de l'ensemble au reste du groupe. Puisque les paramètres de l'arbre sont imposés par la hiérarchie du réseau, ce modèle considère les différentes organisations possibles d'un ensemble de noeuds pour trouver celle qui améliore la performance du multicast.

En combinant les résultats de ces deux modèles, on sera capable de trouver toujours le bon dimensionnement. Le premier est suffisant pour un réseau déjà existant tandis que le deuxième requiert un changement d'organisation mais il donne une idée sur l'effet de ce respect de la hiérarchie du réseau sur la performance de l'arbre. Ce changement est plus simple en ATM où la hiérarchie est bâtie automatiquement grâce au PNNI.

Comme hypothèses de ces deux modèles, on trouve la taille du réseau M et celle du groupe à connecter G. Puisque les réseaux en pratique ont des topologies très différentes (même dans un réseau donné, la topologie change suite aux changements dans l'état des liens et des noeuds) et puisque la distribution des membres varie dynamiquement (ces membres peuvent exister partout), il est difficile de trouver une solution optimale pour chaque cas particulier. Il faut prendre dans les simulations un grand nombre de cas possibles pour les mêmes hypothèses puis calculer les moyennes et les variances des résultats obtenus. A partir de ces moyennes, on essaie de trouver toute cohérence qui permet de définir des résultats généraux. La variance d'un résultat donné montre à quel point la performance de l'arbre sera affectée par un changement dans l'état d'un des composants du problème.

Les résultats qui nous permettent de bien étudier cette hiérarchie sont le coût total de l'arbre et son rapport au coût minimal qu'il suffit de dépenser pour lier les membres. En variant les hypothèses du problème (M et G) et les paramètres de l'arbre hiérarchique (N et n_i)[1], et par comparaison des résultats, on pourra étudier:

- La variation de la performance d'un arbre ayant une structure donnée en fonction des hypothèses du problème. Ceci sert à étudier l'effet d'un changement dans la topologie du réseau ou dans la distribution du groupe sur le coût du multicast.

1. Il s'agit des paramètres du réseau hiérarchique dans le deuxième modèle.

– La variation de la performance de l'arbre en fonction de ses paramètres et ceci pour des hypothèses données du problème. Cette étude sert à trouver le meilleur dimensionnement de l'arbre hiérarchique [2].

Ensuite, un troisième modèle est présenté pour étudier l'effet de la hiérarchie sur la concentration de trafic aux centres. Cette concentration limite le nombre de groupes que le réseau peut supporter simultanément et donc les revenus de l'opérateur. Les meilleurs paramètres de l'arbre qui réduisent cette concentration sont cherchés.

Avant de détailler les modèles de simulation, une brève présentation des méthodes de modélisation d'un réseau du monde réel est faite. Ensuite, les modèles de réseau choisis ainsi que l'outil de simulation utilisé sont expliqués. Les résultats obtenus et leurs interprétations viennent dans le chapitre suivant.

4.1 Modélisation d'un réseau

L'outil utilisé pour modéliser un réseau est le **graphe**. Les noeuds du graphe représentent les routeurs ou les commutateurs et les arcs représentent les liens de transmission. La longueur d'un arc indique le temps de transmission entre les deux noeuds et son poids désigne le coût à payer lors de la transmission sur ce lien.

Plusieurs modèles ont été définis dans la littérature pour générer un graphe représentant le mieux un réseau. Le modèle **Pure Random** est le plus simple. Il distribuent les noeuds en des points arbitraires du plan puis il prend chaque paire de noeuds et ajoute un lien entre eux avec une probabilité **p**. Le résultat est un graphe plat n'ayant aucune organisation hiérarchique. Bien qu'il ne reflète pas la structure des grands réseaux réels, ce modèle est beaucoup utilisé dans l'étude des problèmes des réseaux.

D'autres modèles Random existent aussi. Ils distribuent toujours les sommets arbitrairement dans le plan. La différence est qu'ils changent la probabilité d'un lien dans le but de mieux représenter les réseaux réels. Le plus utilisé est celui proposé par **Waxman** [23] qui prend comme probabilité d'un lien entre deux points **u** et **v**:

$$P(u,v) = \alpha.e^{-\frac{d}{\beta.L}} \tag{4.1}$$

où $\alpha > 0, \beta \leq 1$ sont les paramètres du modèle, d est la distance Euclidienne entre u et v et L la distance maximale entre n'importe quels deux noeuds. Une augmentation de α augmente le nombre des liens dans le graphe tandis qu'une augmentation de β augmente le rapport des liens longs sur les liens courts.

Plusieurs variances du modèle de Waxman ont été proposées. Une appelée modèle de **Doar et Leslie** [22] multiplie $P(u,v)$ par un facteur $\frac{k.\varepsilon}{n}$ où ε est le degré moyen désiré, n le nombre des noeuds et k une constante dépendante de α et β. L'ajout de ce facteur permet un contrôle direct sur le nombre des liens du graphe généré.

Deux autres modèles ont été aussi proposés dans [7] pour relier la probabilité d'un lien à la distance entre les noeuds comme dans Waxman mais avec des fonctions plus directes. Le modèle **Exponentiel** adopte une probabilité qui décroît exponentiellement avec la distance:

$$P(u,v) = \alpha.e^{-\frac{d}{L-d}} \tag{4.2}$$

et le modèle de **Localité** divise les liens en deux catégories suivant leur longueur et assigne une probabilité à chaque catégorie:

$$P(u,v) = \alpha \ si \ d < r \ et \ \beta \ si \ d \geq r \tag{4.3}$$

2. La meilleure organisation du réseau dans le deuxième modèle.

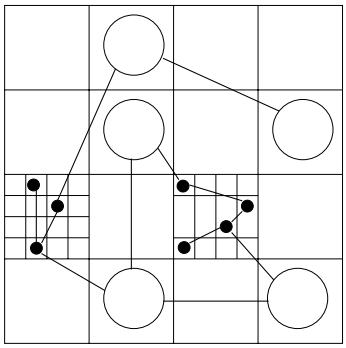

FIG. 4.1 – *Un graphe N-niveaux avec* $S_N = S_{N-1} = 4$

Une des caractéristiques de ce dernier modèle est qu'on est capable d'y étendre plusieurs résultats du modèle Random.

Tous ces modèles ne tiennent pas compte de la hiérarchie existante dans les réseaux réels bien qu'ils sont capables de refléter quelques notions de localité si certains noeuds sont plus connectés que d'autres. Leur problème est qu'ils donnent à un noeud la possibilité de joindre tous les autres et le graphe résultant n'est pas formé d'une interconnexion de petits graphes comme c'est le cas dans un Internetwork. Ceci limite la taille du réseau qu'ils peuvent modéliser puisque, avec un faible degré et un grand nombre de noeuds, le calcul d'un graphe plat **connecté** devient très difficile.

Les deux méthodes suivantes définies dans [7] créent des topologies hiérarchiques en connectant des petits composants suivant une structure particulière. Dans un modèle hiérarchique, il est possible de définir une politique de routage pour que les routes entre les noeuds d'un domaine ne sortent pas à l'extérieur. Ceci peut se faire en ajoutant aux longueurs des côtés, un poids de routage politique qui sera utilisé dans le calcul des plus courts chemins.

Le modèle de **N-niveaux** construit la topologie d'une façon récursive. Il part d'un graphe connecté. En chaque étape de la récursion, chaque noeud de la topologie courante est remplacé par un nouveau graphe connecté. Le raccordement d'un lien au nouveau graphe est résolu de plusieurs manières. Par exemple, on peut choisir arbitrairement un des noeuds du nouveau graphe comme extrémité d'un lien qui était connecté à son père.

Ce modèle définit pour chaque niveau i ($i = 1 \ldots N$) de la hiérarchie une échelle S_i et un nombre moyen de noeuds dans un domaine m_i. Il part au niveau N (le niveau le plus haut) d'un carré et le divise en $S_N \times S_N$ petits carrés. Il distribue ensuite arbitrairement les m_N points sur ces petits carrés et les connecte par un graphe (Figure 4.1). Au niveau i, il prend chaque carré contenant un noeud du niveau $i+1$ et le divise aussi en $S_i \times S_i$ petits carrés. Ici, il ne distribue pas m_i points sur chaque carré d'un noeud du niveau $i+1$. Ce qu'il fait c'est calculer le nombre total des noeuds à ce niveau puis les distribuer arbitrairement sur tous les carrés du niveau $i+1$ divisés. Le nombre total des noeuds dans le graphe résultant vaut alors

$$M = m_N \times m_{N-1} \times \cdots \times m_1 \tag{4.4}$$

En prenant 1 comme surface d'un carré contenant un noeud au niveau 1, la longueur d'un côté du carré de départ devient

$$L = S_N \times S_{N-1} \times \cdots \times S_1 \tag{4.5}$$

L est considérée alors comme étant la valeur maximale des coordonnés d'un point du graphe et $\sqrt{2}.L$ la longueur maximale d'un arc du graphe.

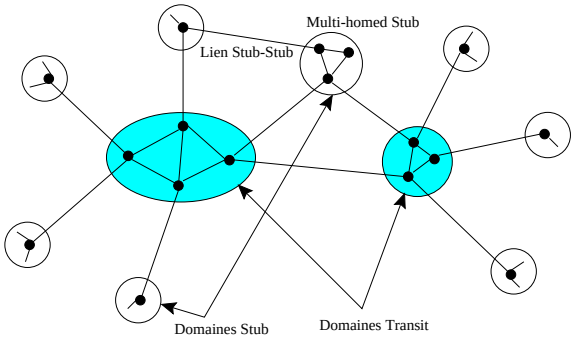

FIG. 4.2 – *Un graphe Transit-Stub*

Le modèle **Transit-Stub** produit des graphes hiérarchiques en interconnectant des domaines Transit et Stub. Tout d'abord, il construit un graphe Random connecté en utilisant une des méthodes déjà vues. Chaque noeud de ce graphe représente un domaine Transit et sera remplacé par un autre graphe Random connecté représentant le backbone du domaine. Ensuite, pour chaque noeud de chaque domaine Transit, il génère un nombre de graphes Random connectés qui représentent les domaines Stub liés à ce noeud. À la fin, il ajoute quelques côtés supplémentaires entre des paires de noeuds, un d'un domaine Transit et un autre d'un Stub ou bien un de deux Stub différents (Figure 4.2).

Il évident que le modèle Transit-Stub modélise bien un Internet formé d'un backbone entre les routeurs de bord des domaines. Cependant, son utilisation dans notre simulation n'est pas très utile puisqu'il fixe le nombre des niveaux et donc ne nous permet pas de bien étudier l'effet de la hiérarchie sur la performance du multicast.

En revanche, le modèle N-niveaux, par son nombre de niveaux variable, constitue le bon choix pour considérer toutes les organisations possibles d'un réseau. La manière suivant laquelle il génère les graphes ressemble beaucoup à la construction de la hiérarchie en PNNI. Les domaines à un niveau i représente les PG et les liens entre eux ne sont que les liens logiques. Le seul problème de ce modèle est qu'il représente un lien logique entre deux PG par un seul lien physique tandis qu'en ATM, un tel lien est le résultat d'une agrégation de plusieurs liens du plan physique. Ce problème n'a pas d'effet dans notre cas puisque le calcul des routes entre les PG se fait suivant la vision d'un seul point qui est le core choisi et donc même s'il y a plusieurs liens physiques entre deux PG voisins, un d'eux peut être sur l'arbre à un moment donné. En plus de l'ATM, un tel modèle peut répondre aussi au problème de dimensionnement d'un nouveau réseau Internet.

Pour modéliser les PG et les domaines de l'Internet, le modèle Pure Random est adopté. Des liens peuvent exister entre tous les noeuds sans tenir compte de la distance. La probabilité du placement d'un lien p entre deux noeuds peut être exprimée en fonction du degré moyen d'un noeud du réseau D et de la taille du graphe M par:

$$p = \frac{D}{M-1} \tag{4.6}$$

4.2 Outil utilisé

Aucun des simulateurs existants n'est capable de fournir des réponses aux questions qu'on a posées. Il est plus facile de développer un outil de simulation que de se fatiguer à ajouter de nouvelles fonctions à un outil déjà fait. Tout ce qu'on demande dans notre travail c'est un générateur de graphes plats et hiérarchiques utilisant les deux modèles déjà expliqués. Ayant ces graphes, on est capable de placer des centres et des membres puis de

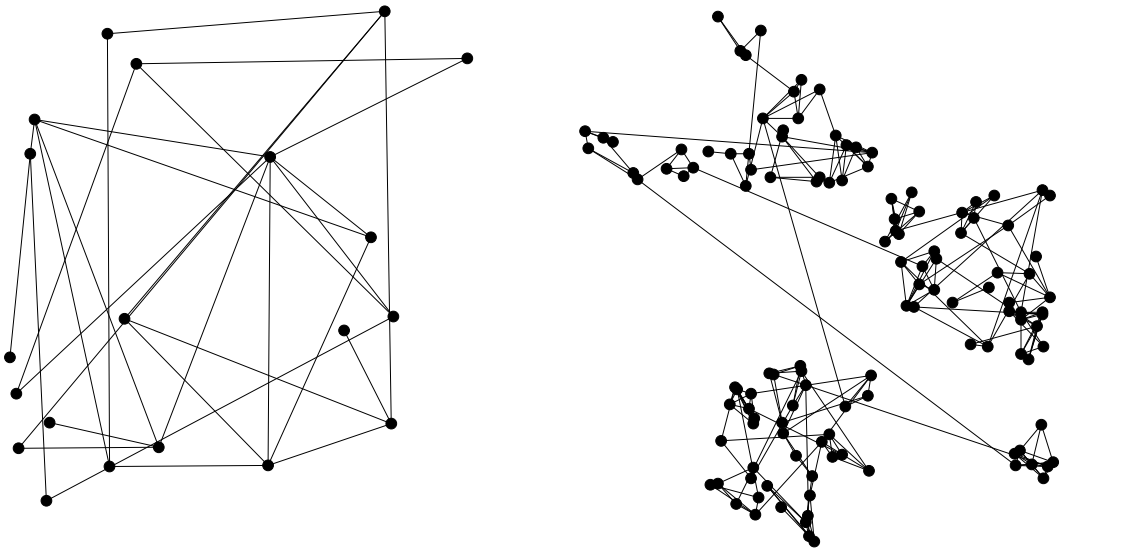

FIG. 4.3 – *Les deux types de graphes utilisés*

les connecter par un arbre partagé suivant les mécanismes étudiés. Aussi, pour évaluer la performance d'un dimensionnement donné, on peut implémenter dans ces graphes et pour la même distribution des membres une des heuristiques du Steiner. Les coûts des deux arbres peuvent être facilement calculés et par suite comparés.

Le générateur choisi est le **GT-ITM** (The Georgia Tech Internetwork Topology Models) [6] développé à Georgia Tech. Il donne des graphes Pure Random et hiérarchiques N-niveaux ce qui répond parfaitement à nos besoins (Figure 4.3). Il est basé sur une plate-forme de structures de données et de routines pour la représentation et la manipulation des graphes appelée Stanford Graphbase (SGB) [8].

Les paramètres à passer à ce générateur dépendent du type du graphe à produire. Pour un Pure Random, il demande la taille du graphe M, le degré moyen d'un noeud D et la longueur du carré dans lequel le graphe va être placé S. Pour un graphe hiérarchique, il demande le nombre des niveaux N et pour chaque niveau i, les paramètres m_i, S_i et D_i.

Le programme qu'on a développé appelle le générateur avec les paramètres désirés. Puis il convertit le graphe généré du format SGB en un autre format plus facile à être manipulé. Ensuite, il place les membres d'une façon arbitraire et les connecte par un arbre Steiner puis par un arbre partagé. Les coûts des deux arbres sont calculés comme étant la somme des poids des liens qui les composent.

4.3 Calcul du Steiner

L'heuristique choisie pour trouver le coût minimal suffisant pour lier les membres du groupe est proposée dans [5]. C'est une amélioration de l'algorithme KMB qui a les mêmes caractéristiques du point de vue coût de l'arbre mais une complexité plus petite. La borne supérieure du rapport du coût de l'arbre de cette heuristique à celui de l'arbre Steiner optimal est $2(1 - \frac{1}{L})$ où L est le nombre des feuilles de l'arbre optimal.

Cet algorithme consiste à marquer les G noeuds du réseau où se trouvent les membres du groupe comme **sources**. Ensuite, il connecte un noeud quelconque du graphe à la source la plus proche suivant le plus court chemin. On aura alors G arbres disjoints basés sur les

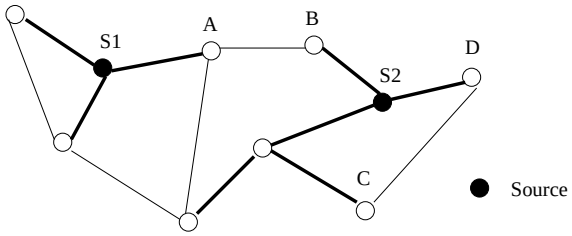

FIG. 4.4 – *Connexion des noeuds aux sources*

membres du groupe (Figure 4.4).

Maintenant, il prend les liens qui n'appartiennent pas à un de ces G arbres et qui connectent deux arbres différents (par exemple $A - B$). Ceux qui connectent deux noeuds d'un même arbre (par exemple $C - D$) ne sont pas considérés. En ajoutant au poids de chacun de ces liens les distances entre ses extrémités et les sources des arbres auxquels elles appartiennent ($d(S_1, A)$ et $d(S_2, B)$) puis en comparant les résultats, on obtient la longueur du plus court chemin, s'il existe, entre deux membres quelconques et le lien n'appartenant pas aux G arbres qui a servi pour le construire. Ensuite, on forme un autre graphe ayant comme noeuds les G membres du groupe et comme poids des arêtes les longueurs trouvées.

Ayant ce graphe qui ne contient que les membres du groupe, on cherche l'arbre recouvrant minimal (Minimal Spanning Tree) suffisant pour lier tous ses sommets. Dans l'arbre résultant, chaque arête est représentée dans le graphe initial par un lien n'appartenant pas aux arbres des sources. Pour chaque arête, on marque dans le graphe initial le lien qui le représente avec les plus courts chemins entre ses deux extrémités et les sources de leurs arbres. L'ensemble des liens marqués constituent un arbre connecté sans boucles liant les membres du groupe. C'est l'arbre proche de l'optimal.

4.4 Calcul du coût d'un arbre multicast

Pour bien utiliser les ressources du réseau, un protocole de routage multicast doit optimiser la bande passante totale consommée par l'arbre. Puisque toutes les branches d'un arbre partagé sont traversées par le même trafic, l'optimisation consiste à minimiser leur nombre. Mais en pratique certains liens du réseau ont des capacités plus grandes que les autres et on a intérêt à les choisir pendant l'établissement d'une nouvelle branche. Aussi, l'installation de la même capacité peut être plus coûteuse sur un lien que sur un autre par exemple à cause de la distance. Une certaine bande passante ne coûte pas la même chose sur tous les liens du réseau et il faut essayer de choisir ceux qui minimisent le coût total de l'arbre. Ce coût, qui est le poids à associer à un lien du graphe, est inversement proportionnel à sa capacité et proportiennel au coût d'installation. Dans notre travail, on a supposé que tous les liens sont également dimensionnés et on a pris la longueur comme représentant du coût d'installation. En plus de ces deux composantes du coût, l'administrateur peut ajouter un poids politique pour empêcher l'utilisation d'un certain chemin lorsque d'autres existent. Un tel poids politique est donné aux liens externes pour que le trafic interne ne sort pas des domaines.

Dans le premier modèle, les liens possèdent la même importance puisqu'ils sont placés dans le domaine sans tenir compte de la distance. Le facteur " longueur " disparaît et le même coût de la bande passante, donc le même poids, est assigné aux liens d'un graphe plat. Pour simplifier le calcul, on a pris " 1 " comme poids. Le coût total d'un arbre sera alors le nombre de ses branches et le critère du choix d'un chemin sera le nombre de sauts pas la longueur.

Dans le deuxième modèle, on a quitté le domaine pour travailler dans tout le réseau. Dans les backbones, et d'après notre générateur, les liens sont généralement plus longs et pas très nombreux. Ils constituent une ressource rare qu'il faut économiser bien qu'ils sont supposés avoir la même capacité. Un chemin interne doit être choisi même s'il possède un plus grand nombre de sauts qu'un chemin externe. Pour augmenter le coût de la bande passante sur ces liens, la longueur est choisie comme poids. Le coût total de l'arbre sera la somme des longueurs de ses branches et le plus court chemin entre deux noeuds sera choisi en comparant la longueur des chemins possibles.

4.5 Premier modèle: cas d'un réseau plat

Ayant un graphe Pure Random de taille M (le degré D est fixé à 3), les G membres du groupe sont placés arbitrairement dans le graphe de telle sorte que deux membres n'occupent pas le même noeud. Ensuite, le coût du Steiner C_S liant ces membres est calculé.

Maintenant, on place arbitrairement les cores de l'arbre partagé. Deux cores ne peuvent pas occuper le même noeud tandis qu'un membre et un core peuvent exister ensemble. Dans le deuxième cas, le core représente le routeur désigné du réseau local sur lequel se trouve le membre ou bien le commutateur ATM auquel la machine de ce membre est connectée. Les paramètres de l'arbre hiérarchique sont le nombre de niveaux N et le nombre de cores par niveau n_i. Le nombre total des cores distribués vaut

$$n = n_1 + n_2 + \cdots + n_N \tag{4.7}$$

On commence ensuite à ajouter les membres au groupe l'un après l'autre en utilisant cette hiérarchie de cores. Un membre joint le core le plus proche de lui et un core, à son tour, joint celui ayant un niveau plus élevé. En cas de boucle, les branches sont coupées suivant le mécanisme expliqué. Les cores qui trouvent qu'ils n'ont pas besoin du trafic multicast, envoient des Prune pour se détacher de l'arbre. À la fin, le coût total C_H est calculé, comme dans le cas du Steiner, comme étant le nombre des liens marqués sur l'arbre.

Ayant C_S et C_H, le rapport de coût, qui donne une idée sur la performance d'un dimensionnement donné de la hiérarchie, est calculé $Ratio = \frac{C_H}{C_S}$.

Dans les simulations, plusieurs valeurs sont prises pour chacun des paramètres du modèle. Les résultats obtenus sont groupés ensemble pour étudier l'effet d'un certain paramètre sur la performance. Pour chaque combinaison, plusieurs itérations sont faites (100 résultats sont trouvés pour les mêmes valeurs des paramètres) puis le résultat moyen et la variance sont calculés.

4.6 Deuxième modèle: cas d'un réseau hiérarchique

Notre but ici est d'étudier l'effet de la hiérarchie du réseau sur le coût du multicast. Puisque l'approche de l'ATM est la seule qui a bien expliqué le placement des cores et la construction de l'arbre partagé dans un réseau hiérarchique, et puisque la hiérarchie du graphe généré par notre outil ressemble beaucoup à celle de PNNI, ce modèle a utilisé les mêmes fonctions de cette approche. Cependant, les résultats trouvés peuvent être appliqués à n'importe quel réseau.

Suivant cette approche, les cores de l'arbre multicast sont placés dans les différents domaines du réseau. Tous les paramètres du modèle, sauf G, sont ceux du graphe N-niveaux. Le nombre de cores et leur distribution sur les niveaux sont imposés par la structure du

graphe. Comme paramètres, on trouve N, m_i, D_i et S_i qui nous donnent M et L d'après les équations 4.4 et 4.5.

Dans le graphe généré, un core est choisi arbitrairement dans chaque domaine du niveau 1 du graphe. Ces domaines sont au nombre de

$$m_2 \times m_3 \times \cdots \times m_N = \frac{M}{m_1} \tag{4.8}$$

Les cores choisis sont ensuite marqués avec le niveau 1. Au niveau 2 du graphe et comme en PNNI, chaque domaine du niveau 1 est représenté par un noeud et ces noeuds sont groupés en un ensemble de domaines. De nouveau, un core est choisi dans chacun de ces domaines. D'après le principe de mappage illustré à la figure 3.8, un core choisi au niveau 2 du graphe correspond au core du domaine qu'il représente et qui porte le niveau 1. D'une manière générale, un core de niveau X de l'arbre est choisi arbitrairement parmi les cores de niveau $X-1$ qui appartiennent au même domaine au niveau X du graphe et qui eux même sont choisis parmi ceux de niveau $X-2$ et ainsi de suite. Donc tous les cores de l'arbre se trouvent sur ceux de niveau 1 et leur distribution sur les niveaux est fonction des paramètres du graphe N et m_i.

Le nombre total de cores n vaut tout simplement le nombre de domaines au niveau 1 du graphe donné en 4.8. La relation entre les n_i et les m_i est donnée par:

$$n_N = 1$$

$$n_i = (m_{i+1} \times m_{i+2} \times \cdots \times m_N) - n_N - n_{N-1} - \cdots - n_{i+1} \quad pour \quad 1 \leq i \leq N-1 \tag{4.9}$$

Les membres du groupe, comme dans le cas précédent, sont placés arbitrairement sans tenir compte de la hiérarchie du réseau. Ensuite, on commence à les connecter suivant l'approche d'ATM. Un membre joint le core de niveau 1 de son domaine. Un core de niveau X cherche le core de niveau $X+1$ dans le domaine du niveau $X+1$ du graphe auquel il appartient. Pour traverser un certain domaine, on est toujours obligé de passer par son core. Les problèmes de boucle et des cores feuilles sont résolus comme il a été expliqué.

À la fin, le coût total de l'arbre C_H est calculé. Le Steiner n'est pas cherché puisque la comparaison entre les coûts des deux arbres a été faite dans [4]. Comme dans le premier modèle, les paramètres sont modifiés et les résultats sont groupés pour étudier l'influence d'un paramètre sur la performance du multicast.

4.7 Troisième modèle: étude de la concentration

La nécessité de joindre un core pour recevoir le trafic multicast rend la plupart des liens autour du core marqués sur l'arbre du groupe. Donc une bande passante sera réservée sur ces liens ce qui empêchera son utilisation par d'autres applications. Supposons que le volume du trafic circulant sur l'arbre partagé est indépendant de la taille du groupe. Alors la concentration au core ne posera pas des problèmes de saturation et de blocage en cas d'existence d'un seul groupe. Un nouveau membre de ce groupe cause une réservation de la bande passante seulement entre lui et le premier noeud de l'arbre et ne change pas celle réservée sur les autres branches.

Le problème se pose lorsque d'autres groupes utilisant le même core viennent se créer. Un nouveau groupe ne peut pas exister que s'il y a suffisement de la bande passante sur les liens qui entourent le core. Une saturation de ces liens causée par un grand nombre de groupes fait bloquer les demandes de connexions. En effet, l'information sur cette saturation parviennent aux membres qui veulent successivement joindre un certain groupe. Puisqu'il

ne possède pas des informations sur l'arbre, le nouveau membre essaie de trouver un chemin non saturé vers le core. Une fois qu'il le trouve, il envoie sa demande qui pourra être arrêtée par un noeud de l'arbre avant d'atteindre sa destination. Mais si un tel chemin n'est pas trouvé, la demande sera refusée bien que l'arbre du groupe cherché peut être atteignable. Parfois le chemin ne pourra pas être trouvé suite à une saturation des liens autour du nouveau membre pas autour du core.

Dans un réseau fournissant des garanties de QoS comme ATM, il faut essayer d'utiliser au maximum les ressources disponibles et de réduire la probabilité de blocage d'une demande de connexion au groupe. Notre but ici est d'étudier, suivant le schéma proposé dans le deuxième modèle, l'effet de la hiérarchie du réseau sur cette probabilité.

Puisque c'est le nombre de groupes qui compte, le modèle consiste à placer dans un réseau hiérarchique, ayant N et m_i comme paramètres, des groupes de taille tirée aléatoirement. Comme il est déjà mentionné, les liens du réseau sont supposés avoir la même capacité C. Les groupes génèrent tous le même volume de trafic pris comme unité, donc C n'est autre que le nombre maximum des groupes qui peuvent envoyés leurs trafics sur un lien sans qu'ils le saturent.

Lorsqu'un lien s'ajoute à l'arbre d'un groupe, sa capacité est réduite d'une unité. Le calcul du plus court chemin doit considérer seulement les liens du réseau non saturés. Le même algorithme de **Dijkstra** est utilisé mais un test additionnel est effectué pour voir si la bande passante résiduelle sur un lien est plus grande que zéro. Le poids n'est pas modifié en fonction de l'utilisation et il est toujours égal au longueur du lien.

Une modification est introduite au mécanisme de construction de l'arbre utilisé dans le deuxième modèle. Un membre, qui n'a pas pu trouver un chemin non saturé vers le core, annule sa demande, enregistre le numéro de son essai et incrémente un compteur $Blockage$ indiquant le nombre cumulatif des essais bloqués. Une fois que ce nombre atteint un seuil $Threshold$, les essais seront arrêtés. La variation de $Blockage$ en fonction des numéros des essais est ensuite étudiée. Ce qui nous intéresse c'est le point P à partir duquel le blocage des demandes sera dominant. Ce point est le début d'une variation presque linéaire de $Blockage$.

En variant les paramètres du réseau, on peut chercher la meilleure organisation qui donne une grande valeur à P. Un grand nombre de groupes pourra alors exister simultanément et les ressources seront efficacement utilisées.

Chapitre 5

Résultats des simulations

Pour chacun des modèles définis et pour chaque combinaison des paramètres, on a obtenu autant de résultats que d'itérations effectuées (100). La moyenne et la variance de ces résultats ont été ensuite calculées. Ceci a donné un ensemble de points de la fonction de coût pour les deux premiers modèles (*Ratio* pour le premier et C_H pour le deuxième) et de la fonction qui donne les numéros des essais qui ont subi un blocage pour le troisième modèle. En moyennant sur toutes les combinaisons ayant la même valeur d'une certaine variable, on trouve la variation de la performance de l'arbre en fonction d'un des paramètres du modèle. Ces variations et leurs interprétations sont présentées dans les paragraphes suivants.

5.1 Premier modèle: cas d'un réseau plat

Les variations du rapport de coût $Ratio(M, G, N, n_i)$ sont étudiées et les résultats sont illustrés graphiquement pour un réseau Pure Random de 100 noeuds.

Variation de *Ratio* en fonction de G
La figure 5.1 montre la variation de la performance de l'arbre partagé en fonction de la taille du groupe. On remarque, comme c'est le cas de tous les arbres multicast, un rapprochement du Steiner lorsque la taille du groupe augmente. Ceci est le résultat d'une amélioration d'un emplacement donné des cores en cas des grands groupes. En moyenne, l'arbre hiérarchique est au plus deux fois plus coûteux que le Steiner et pour des petits groupes, on peut même avoir un coût plus faible.

La variance décroît avec l'augmentation de la taille du groupe. Cette décroissance est causée par une diminution de l'effet du placement des cores sur le coût. En effet, pour les petits groupes, il y a une grande probabilité que les cores soient placés loin des membres ce qui donne naissance à un arbre coûteux. Mais aussi et à cause de la hiérarchie, il y a une grande probabilité que les cores se trouvent en un bon emplacement par rapport aux membres ce qui réduit considérablement le coût. Cependant, les grands groupes sont le plus souvent éparpillés dans le réseau. Donc, un mauvais placement des cores par rapport à certains membres constitue un bon placement par rapport à d'autres ce qui cause une compensation des coûts. Pour cela, la plupart des distributions des cores et des membres donnent presque la même performance.

Cette grande variance en cas des petits groupes et qui est due à la grande probabilité d'avoir un core à proximité d'un groupe local, ne se voit pas en cas d'un seul core. La racine de l'arbre est presque toujours loin des membres.

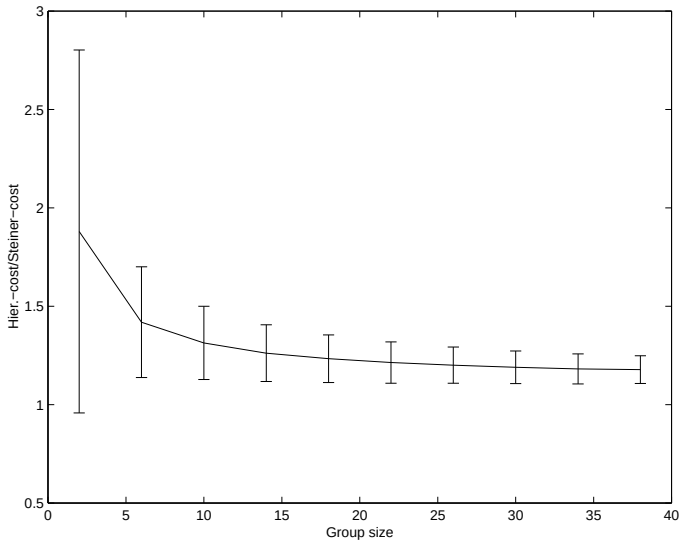

FIG. 5.1 – *Variation de Ratio en fonction de la taille du groupe*

Variation de *Ratio* en fonction de N

Maintenant, l'effet du nombre de niveaux est étudié. Puisque les simulations ont montré les mêmes résultats pour les différentes tailles du groupe, la moyenne de *Ratio* est calculée pour tous les G et les n_i. Contrairement à l'intuition, la figure 5.2 montre une dégradation de la performance de l'arbre pour toute augmentation du nombre de niveaux. Ce résultat veut dire tout simplement qu'on a intérêt à placer un seul core dans un réseau plat pour lier les membres distribués sur ses noeuds bien qu'on a expliqué dans le chapitre 3 que les cores divisent le réseau en des cellules ce qui agrège le trafic et réduit le coût du multicast.

L'interprétation de ce résultat est liée à la nature même du réseau modélisé. Un lien peut exister entre n'importe quels deux noeuds et tous les liens ont la même importance. Donc des cellules séparées ne peuvent pas exister à un niveau de la hiérarchie. Un membre peut se trouver géographiquement dans une cellule tout en étant connecté au centre d'une autre cellule. Donc la longueur du chemin entre un membre et le core le plus proche de lui est comparable à celle du chemin entre deux cores. Ceci augmente considérablement le coût du chemin entre un membre et un core ou bien entre deux cores lorsqu'on insère un niveau entre eux. Dans le cas des cellules disjointes, les deux chemins ne sont pas comparables et la faible augmentation du coût qui résulte de l'ajout d'un niveau sera compensée directement lorsqu'un nouveau membre se connecte au core ajouté.

Un autre phénomène qui contribue à la réduction de la performance est la dispersion du trafic multicast. Le mécanisme de construction adopté donne à un noeud la possibilité de choisir le core le plus proche quelque soit son identité. Donc, il y a une grande probabilité que deux membres voisins, qui utilisaient le même chemin pour se connecter à la racine avant l'insertion d'un niveau, choisissent deux cores différents du niveau ajouté. Cette séparation des chemins initiaux augmente le coût de l'arbre et annule l'agrégation naturelle qui avait lieu aux noeuds du réseau. Il faut mieux forcer ces deux noeuds voisins à joindre le même core pour garder l'agrégation. Une telle solution nécessite une organisation hiérarchique du réseau pour pouvoir appliquer le routage politique et interdire la connexion à un core à l'extérieur si le core du domaine est disponible.

Ce résultat, bien qu'il est trouvé dans le contexte d'un réseau plat, nous permet de conclure le meilleur placement des cores dans un réseau hiérarchique. En partant des hôtes vers le haut dans la hiérarchie du réseau, un seul core doit être choisi dans chaque domaine

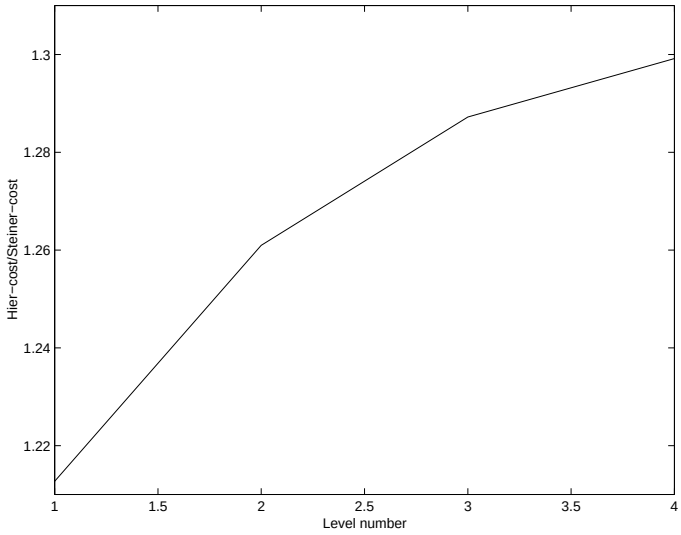

FIG. 5.2 – *Variation de Ratio en fonction du nombre de niveaux*

rencontré. Dans l'Internet, ce core est un des routeurs externes connectant plusieurs zones d'un certain domaine ou bien un des cores des domaines lorsqu'il s'agit seulement d'un backbone de liens entre les routeurs de bord. En ATM, le backbone connectant les PG est logique et il est représenté au plan physique par les liens externes. Donc, comme il est défini dans [4], un core d'un niveau X vient se placer sur un des cores du niveau $X-1$.

Le choix d'un seul core par domaine dans notre deuxième modèle garantit une meilleure performance pour une organisation donnée du réseau. Cependant, le calcul de la meilleure organisation constitue l'autre problème à résoudre.

Variation de *Ratio* en fonction de n_i
Le but de cette étude est de trouver la meilleure distribution des cores sur les N niveaux. Mais, puisque le coût minimum a été obtenu pour $N = n_1 = 1$, une telle étude devient non intéressante.

Cependant, pour un N donné, on a trouvé que la performance de l'arbre se dégrade en augmentant le nombre des cores à un niveau. Un grand nombre de cores aggrave le phénomène de dispersion du trafic multicast qui a causé l'augmentation du coût lors de l'insertion d'un niveau dans l'arbre.

Si pour une raison quelconque, par exemple pour respecter certaines contraintes temporelles, on trouve qu'il faut disposer d'un certain N, le meilleur dimensionnement sera pour $n_i = 2 \times n_{i+1}$. Une valeur de n_i plus petite rend la hiérarchie sans sens. Le nombre total de cores à placer vaut alors $n = 2^N - 1$.

5.2 Deuxième modèle: cas d'un réseau hiérarchique

Comme dans le cas précédent, les simulations faites dans [4] ont montré un rapprochement du coût C_H de celui du Steiner lorsque la taille du groupe augmente. La variation de $C_H(N, m_i, G)$ en fonction de l'organisation du réseau a été étudiée en fixant la taille M et en prenant des paramètres m_i et N vérifiant l'équation 4.4. Les courbes présentées

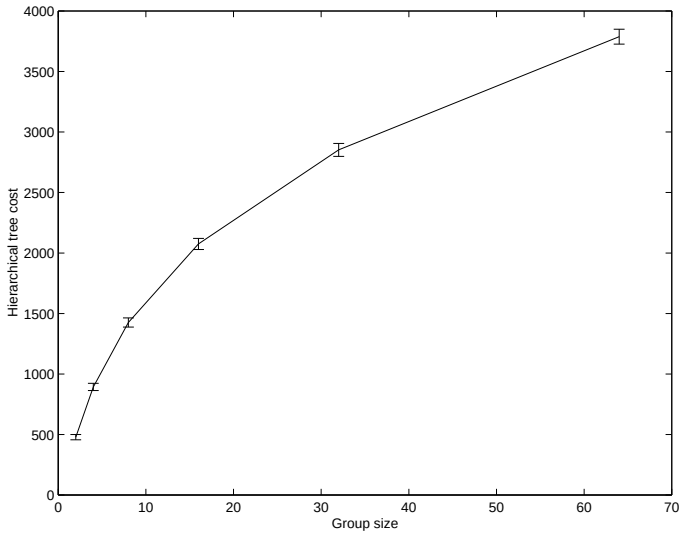

FIG. 5.3 – *Variation de C_H en fonction de la taille du groupe*

dans la suite correspondent à un $M = 100$ et à une valeur maximale des coordonnées des noeuds $L = 200$.

Un des problèmes du générateur était la nécessité d'assigner des valeurs entières aux m_i ce qui limite beaucoup le nombre des points possibles et rend les courbes non significatives. Une modification a été introduite au générateur pour qu'il puisse accepter des valeurs réelles au nombre moyen des noeuds par domaine.

Variation de C_H en fonction de G

Cette étude montre l'importance de l'agrégation du trafic multicast ayant lieu dans le réseau. Ce phénomène, qui est dû à la hiérarchie des cores et au maillage du réseau, précise le coût de l'arbre partagé. Plus on agrège, plus on gagne. Le degré des noeuds, qui détermine le nombre de liens et par suite le nombre des chemins possibles, affecte l'agrégation naturelle hors des cores. Un degré élevé donne à un membre un grand nombre de choix pour aller au core ce qui réduit la probabilité de rencontrer l'arbre multicast sur son chemin et empêche l'agrégation.

La figure 5.3 montre toujours une augmentation du coût chaque fois qu'un nouveau membre joint le groupe. Ceci est logique puisque toujours une branche s'ajoute à l'arbre pour connecter un nouveau membre au reste du groupe. Mais, suite à l'agrégation, le taux de cette augmentation décroît lorsque G augmente (la pente de la courbe devient de plus en plus petite). Cette décroissance s'explique par le fait que la probabilité de trouver un noeud de l'arbre proche du nouveau membre devient plus grande en cas des grands groupes. Donc le coût de la branche à ajouter diminue. Pour optimiser l'utilisation de la bande passante, on a intérêt à augmenter la probabilité d'occurrence de l'agrégation donc à avoir une décroissance rapide de la pente de cette courbe.

Variation de C_H en fonction de N

Pour un nombre de niveaux donné, on a fait la moyenne sur toutes les valeurs de m_i et de G considérées. Contrairement au premier modèle, les résultats (Figure 5.4) ont montré une amélioration de la performance de l'arbre pour toute augmentation de N. Ceci veut dire que pour avoir un coût minimal, il faut prendre le plus grand nombre possible de niveaux.

La cause de cette différence entre les deux modèles est l'organisation hiérarchique du

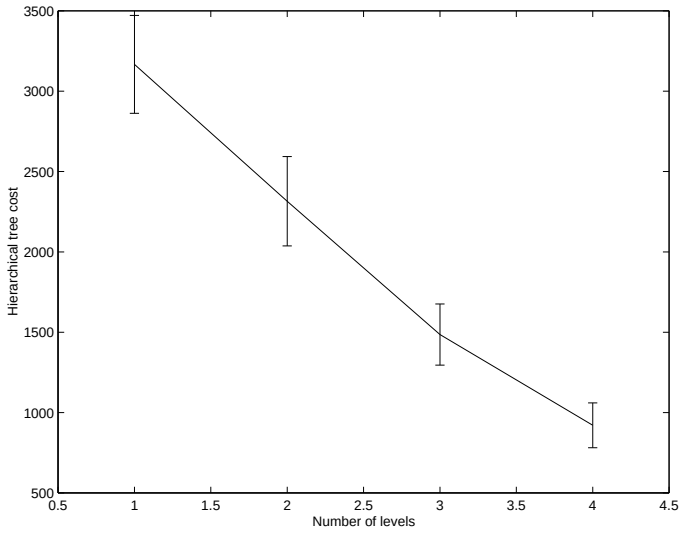

FIG. 5.4 – *Variation de C_H en fonction du nombre de niveaux*

réseau. À un niveau donné, on trouve maintenant des cellules complètement séparées et contenant des réseaux totalement connectés. Les membres de chaque cellule parlent entre eux sans utiliser les liens externes et une fois qu'une connexion aux autres cellules est demandée, c'est le core de la cellule qui choisit un seul chemin externe même s'il y en a plusieurs. Tout le trafic multicast entrant et sortant de cette cellule sera agrégé sur ce chemin.

Aussi, puisque les cellules sont séparées, le surcoût qui résulte de l'insertion d'un niveau n'est pas très important. Le coût du chemin entre un membre et le core de sa cellule est faible par rapport à celui du chemin externe.

La dispersion du trafic, qui a causé la dégradation de la performance dans le premier cas, disparaît ici puisque deux membres voisins, qui sont le plus souvent situés dans une même cellule, sont forcés à joindre le même core ce qui conserve l'agrégation naturelle du trafic.

Ce qui a été dit explique la non augmentation du coût lorsqu'on augmente N mais ne donne pas une interprétation pour cette amélioration de la performance. En fait, la réduction du coût est le résultat d'une nouvelle agrégation du trafic aux cores du niveau inséré. À ce niveau, le réseau est aussi divisé en des cellules séparées ce qui agrège le trafic des cellules en bas en quelques points avant de les passer au niveau supérieur.

Soit un grand domaine A avec des membres connectés par un seul core de niveau 1 (Figure 5.5). Deux membres voisins peuvent choisir deux chemins différents pour joindre ce core bien qu'il y en a un commun. Maintenant, insérons un niveau dans la hiérarchie du réseau et mettons les deux membres dans un même domaine, A.1 par exemple. Ceci les force à joindre tout d'abord un core de niveau 1 avant de joindre le core de niveau 2 ce qui agrège le trafic qui était dispersé et réduit le coût du multicast.

Variation de C_H en fonction de m_i

L'étude précédente a montré qu'il faut augmenter autant que possible le nombre de niveaux dans un réseau pour avoir une bonne performance de l'arbre. Mais la question qui reste sans réponse est comment varie cette performance en fonction du nombre de domaines (cellules, PGs) à un certain niveau. Pour bien comprendre cette variation, on considère tout d'abord le cas d'un réseau à deux niveaux puis on généralise à une valeur quelconque

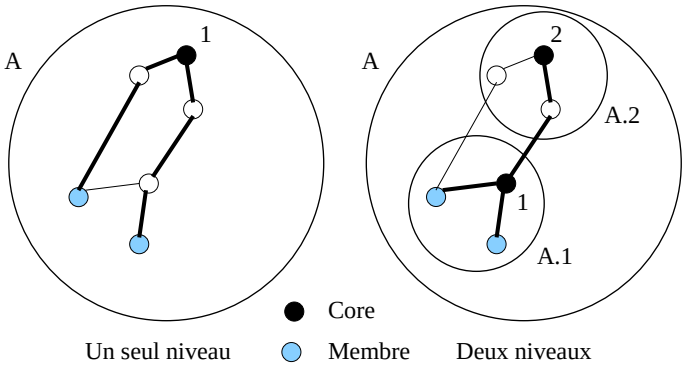

FIG. 5.5 – *Agrégation du trafic au niveau inséré*

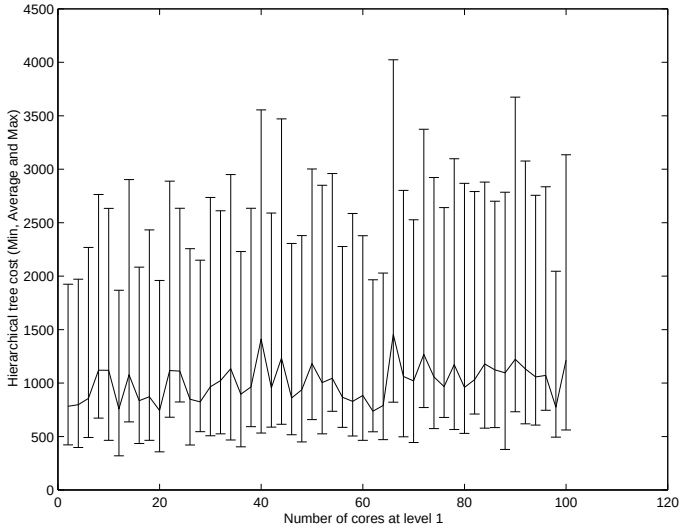

FIG. 5.6 – *Variation de C_H en fonction de m_2 pour un petit groupe*

de N.

Pour $N = 2$, m_2 désigne le nombre de cores au niveau 1 dont un est choisi comme racine de l'arbre. La variation de C_H en fonction de m_2 est étudiée. Les courbes obtenues ont montré, comme il était le cas dans un réseau plat, une augmentation du coût lorsque m_2 croit. Mais on a trouvé que le taux de cette augmentation n'a pas la même importance en cas des grands et des petits groupes. Le coût reste presque uniforme lorsque G est faible (Figure 5.6) et croit rapidement pour les grandes valeurs de G (Figure 5.7).

L'augmentation du coût est due à une dispersion du trafic dans le backbone de l'arbre. Cette dispersion fait remplacer un chemin commun utilisé par plusieurs membres par des chemins séparés. En plaçant plus de domaines à un certain niveau du réseau, on augmente la probabilité de séparation des membres voisins. Par exemple, deux membres, qui étaient connectés au même core de niveau 1 comme le montre la figure 5.8, joignent deux cores différents après la séparation. Bien que la surface des domaines et par suite le coût des liens internes deviennent plus petits, le choix de deux cores augmente le nombre des points qui calculent des chemins externes et annule l'agrégation qui a eu lieu dans le domaine initial. Il y a une grande probabilité que ces deux cores choisissent deux chemins différents vers la racine de l'arbre ce qui disperse davantage le trafic dans le backbone.

En cas des petits groupes, il n'y a pas beaucoup de chemins communs qui risquent

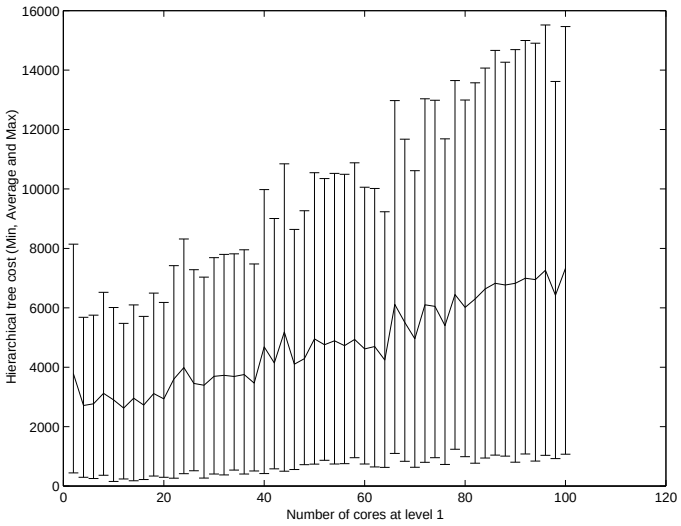

FIG. 5.7 – *Variation de C_H en fonction de m_2 pour un grand groupe*

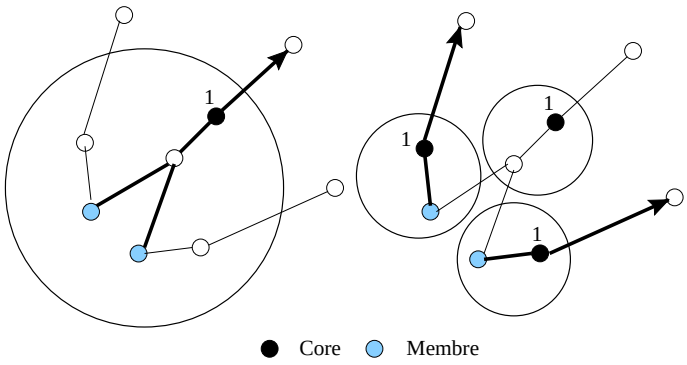

FIG. 5.8 – *Dispersion du trafic multicast dans le backbone*

d'être séparés ce qui rend l'augmentation du nombre de domaines m_2 sans grand effet. Mais dès que G commence à croître, la probabilité de séparation des chemins communs devient plus grande et la performance commence à se dégrader plus rapidement.

Donc, pour avoir un coût minimum, il faut donner à m_2 la plus petite valeur possible. Puisqu'une valeur de 1 n'a pas de sens, la meilleure organisation du réseau en deux niveaux est celle qui le divise en deux domaines au niveau 1. Pour les autres valeurs de N, on a trouvé le même résultat. À un niveau i de la hiérarchie, le coût atteint toujours son minimum pour la plus petite valeur de m_i qui est bornée inférieurement par 2.

Ayant connu l'effet de N et de m_i sur la performance, on peut conclure le meilleur dimensionnement du réseau. Il faut essayer de l'organiser en un grand nombre de niveaux avec des domaines très petits de taille 2. Si on est limité par un N donné et si la taille voulue est M, les domaines au niveau 1 (les PGs au plan physique en ATM) doivent avoir une taille moyenne

$$m_1 = \frac{M}{2^{N-1}} \tag{5.1}$$

Dans ce cas, l'arbre partagé sera formé, en total, de 2^{N-1} cores placés sur les N niveaux de telle sorte que 2^{N-i} cores se trouvent à un niveau i.

5.3 Décomposition d'un réseau en niveaux et domaines

Ayant les résultats déjà trouvés, il faut voir comment peut on organiser hiérarchiquement un réseau existant pour avoir un multicast de meilleure performance. La topologie d'un tel réseau limite le nombre maximum de niveaux qu'on peut considérer et donc le gain qu'on peut avoir.

D'après ce qu'on a vu, un domaine d'un réseau à un certain niveau est un ensemble de noeuds connectés entre eux. Il faut que tous les membres qui se trouvent dedans soient capables de joindre son core. Si ce domaine peut être divisé en des cellules disjointes qui forme chacune un petit réseau connecté, il sera mieux de placer un core dans chacune de ces cellules et de choisir un de ces cores comme racine de l'arbre. Les membres d'une cellule seront forcés à joindre son core qui à son tour joint la racine de l'arbre. Ceci agrège le trafic et optimise la bande passante.

Mais, il faut essayer toujours de trouver des domaines de petite taille aux niveaux allant de N à 2. Donc, on commence l'organisation par la division du réseau en un petit nombre de domaines connectés. Ensuite, une autre division de chaque domaine est cherchée et de nouveau, on essaie de trouver le plus petit nombre de domaines. Cette division du réseau continue jusqu'à ce qu'on arrive à des réseaux plats indivisibles qui ressemblent aux graphes de notre premier modèle. Dans chacun de ces réseaux plats et d'après le résultat trouvé, un seul noeud est choisi comme core pour avoir un arbre interne de faible coût. La promotion des cores aux niveaux plus hauts et la construction de l'arbre se font suivant le mécanisme décrit dans le deuxième modèle. Cette organisation, qui allie les résultats trouvés, assure un meilleur coût de l'arbre résultant.

5.4 Troisième modèle: étude de la concentration

Comme dans les deux simulations précédentes, il faut trouver des valeurs moyennes au point de début de blocage P et aux numéros des essais refusés. Pour cela, un grand nombre d'itérations (500) a été effectué. Chaque itération donne pour chaque valeur de $Blockage$ située entre 0 et $Threshold$, le numéro de la demande de connexion $Number$ qui a échouée. Ensuite, la moyenne de tous les $Number$ des différentes itérations qui correspondent au même $Blockage$ est calculée. Ayant la variation de $Blockage$ en fonction des numéros des essais pour chaque N et m_i, l'effet de la hiérarchie sur la concentration pourra être étudié.

Tout d'abord, on a fixé M à 100, C aussi à 100 et $Threshold$ à 50. Ensuite, on a commencé à varier N et m_i et à tracer les variations de $Blockage$.

Effet de N sur la concentration

En moyennant sur les différentes valeurs de m_i qui correspondent à un même N, cet effet est étudié. La figure 5.9 montre la variation de $Blockage$ pour trois valeurs de N. On remarque clairement une amélioration de la performance lorsque le nombre de niveaux augmente. La hiérarchie raccourcit le chemin non saturé qu'un membre doit trouver pour recevoir le trafic. Il suffit qu'il se rattache au core le plus proche de lui pour que sa demande soit acceptée ce qui réduit la probabilité de blocage. Maintenant, si ce core n'est pas capable de le connecter au reste du groupe suite à une saturation aux niveaux plus hauts, il lui permet au moins de communiquer avec ses voisins. Le partage du groupe qui en résulte reste transparent aux membres. C'est aux cores d'observer l'état des liens et d'essayer toujours de trouver un chemin pour connecter leurs arbres au reste du groupe.

Mais l'ajout de cores au niveau 1 n'est pas suffisant pour améliorer la performance. Si ces cores ne sont pas organisés d'une façon hiérarchique, on aura un long chemin entre

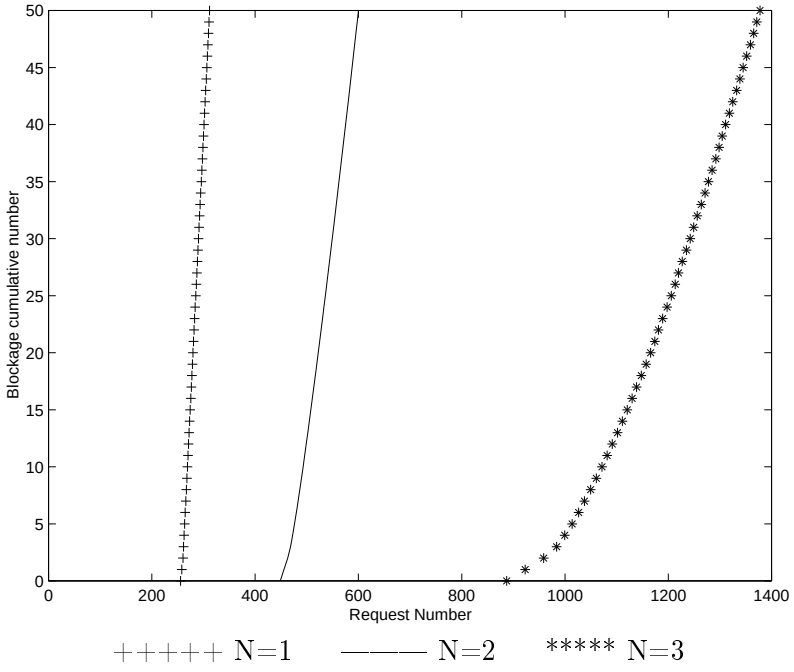

FIG. 5.9 – *La variation des blocages pour différentes valeurs de N*

un core et son père et donc une consommation d'une importante bande passante qui fait bloquer les demandes ultérieures.

On remarque aussi que la pente de la courbe décroît avec N. Cette décroissance veut dire qu'il faut plus de demandes de connexions pour aller du point P au seuil. Lorsque le blocage commence à apparaître, il ne sera pas totale et certains membres arrivent à joindre le groupe bien que d'autres non. C'est la hiérarchie qui, en divisant le réseau en des domaines séparés sur les différents niveaux, rend la saturation d'une zone du réseau sans influence sur les autres. Les membres de la zone saturée pâtissent seuls du blocage et ne privent pas les autres du service multicast.

Chapitre 6

Conclusion

Bien que plusieurs propositions ont été faites, le problème du routage multicast avec des arbres partagés hiérarchiques n'a pas été traité en détails. Les propositions dans le monde de l'Internet ont étendu les protocoles utilisant un seul centre pour qu'ils puissent évoluer à des vastes réseaux et elles ont montré l'intérêt de la hiérarchie pour résoudre certains problèmes comme les boucles en OCBT et les portées des groupes en HPIM. Cependant, elles n'ont pas parlé de la performance de cet arbre, du point de vue consommation de ressources, ni du placement des différents centres dans le réseau.

Au contraire, l'approche de l'ATM a bien expliqué la relation entre la structure de l'arbre et celle de la hiérarchie de PNNI et elle a aussi étudié, à cause de la nécessité de l'optimisation des ressources en ATM, l'efficacité de l'arbre généré par comparaison au Steiner. Mais elle n'est pas beaucoup entrée dans les détails de l'implémentation.

Pour toutes ces propositions, le but était de fournir un service multicast très simple permettant à n'importe quel membre de joindre et de quitter le groupe sans connaître les identités des autres. Aussi, l'utilisation d'un seul arbre par groupe réduit le volume d'informations qu'il faut stocker dans les noeuds du réseau.

Donc, un travail était demandé pour regrouper les résultats des recherches dans ce domaine et pour fournir des solutions aux problèmes non traités. Il fallait essayer de bien caractériser cette nouvelle approche de routage multicast. Le principal problème non résolu était la variation de la performance de l'arbre en fonction de ses paramètres. Cette étude permet de trouver le meilleur placement des centres dans un réseau quelconque, que ce soit ATM ou Internet, qui optimise la consommation des ressources.

Après avoir expliqué les avantages de la hiérarchie, on a traité les différentes propositions tout en essayant de combler les lacunes qui y existent. Ensuite, on est sorti du contexte de l'ATM et de l'Internet pour étudier, par des simulations, la performance de l'arbre hiérarchique d'une façon générale. Toujours les résultats obtenus sont appliqués à chacun de ces deux réseaux.

Trois modèles ont été conçus et un outil de simulation a été développé pour implémenter chacun d'eux.

Le premier modèle considère des réseaux plats n'ayant aucune organisation hiérarchique et cherche le meilleur dimensionnement de l'arbre. On a trouvé que la meilleure performance est obtenue pour un seul core. C'est le cas des domaines de l'Internet et des PG de l'ATM.

Le deuxième modèle considère un réseau hiérarchique et place un core dans chaque domaine de chaque niveau. Le résultat était une amélioration de la performance pour toute augmentation du nombre de niveaux. À un niveau donné, des domaines de petite taille doivent être considérés.

Le troisième modèle a traité le problème de concentration de trafic aux cores et son effet sur le blocage des demandes de connexion. Il a montré que l'insertion de niveaux dans

la hiérarchie éloigne le point de début du blocage et le rend moins total. Seulement les membres des zones saturées subissent une mauvaise qualité de service.

Cependant, cette étude a concentré seulement sur l'amélioration du routage multicast. Ceci a nécessité une modification de la hiérarchie du réseau dans le deuxième modèle. Puisque la performance du routage unicast, en ce qui concerne le volume des tables de routage et le choix des chemins, est fortement liée à cette hiérarchie, toute organisation du réseau doit être un compromis entre les gains apportés par chacun des deux routages.

Le partage d'une branche de l'arbre entre les différents membres du groupe pose des problèmes de QoS. Des mécanismes de contrôle de trafic doivent être mis en place pour utiliser d'une façon efficace et équitable cette ressource commune. Un noeud du réseau, qui concentre les trafics de plusieurs sources sur un même lien de sortie, doit contrôler cette équité. Mais ceci fait apparaître de nouveau le facteur nombre de sources qui a disparu grâce à l'utilisation d'un arbre partagé. En plus, en ATM, un commutateur ne connaît pas les sources des cellules qu'il commute. L'adresse de la source se trouve dans l'entête AAL5 qui est transparent à la couche ATM. Donc, un tel contrôle est difficile.

Un mécanisme d'annonce des centres aux membres du groupe et aux centres des niveaux plus bas doit être aussi défini. En ATM, puisqu'un seul protocole de routage est utilisé, la proposition était de profiter de la diffusion de PNNI. Mais dans l'Internet, l'existence de plusieurs protocoles de routage nécessite l'adoption d'un mécanisme qui ne requiert aucune modification de l'existant. Des solutions comme le Session Directory ou bien l'utilisation des protocoles multicast de type Sender-Intiated peuvent être utilisées.

Enfin, il faut étudier l'effet de la hiérarchie sur la qualité de réception comme le délai, la gigue et les pertes. Cette étude devient importante dans les vastes réseaux. L'amélioration de cette qualité est une exigence de l'utilisateur qui s'oppose à l'intérêt de l'opérateur qui cherche à minimiser la bande passante consommée pour maximiser les revenus.

Bibliographie

[1] ATM User Network Interface (UNI) Specification Version 3.1. ISBN 0-13-393828-X, Prentice Hall, Englewood Cliffs, NJ, June 1995.

[2] The ATM Forum Technical Committee. Private Network-Network Interface Specification Version 1.0. March 1996.

[3] The ATM Forum Technical Committee. Private Network-Network Interface Specification Version 2.0. September 1997.

[4] R. Venkateswaran, C. S. Raghavendra, X. Chen, and V. Kumar. Hierarchical Multicast Routing in ATM Networks. *In IEEE Intl. Conf. on Communications*, volume 3, pages 1690-1694, June 1996.

[5] L.T. KOU, K. MAKKI. An even faster approximation algorithm for the steiner tree problem in graphs. *Congressus Numerantium*, 59 (1987), pp. 147-154.

[6] GT-ITM: Georgia Tech Internetwork Topology Models. http://www.cc.gatech.edu/fac/Ellen.Zegura/graphs.html.

[7] E. Zegura, K. Calvert, and S. Bhattacharjee. How to model an Internetwork. *In Proceedings of IEEE INFOCOM*, April 1996.

[8] D. Knuth. The Stanford GraphBase: A platform for combinatorial computing. *Addison-Wesley*, 1994.

[9] A. Ballardie, B. Cain, and Z. Zhang. Core Based Trees (CBT version 3) Multicast Routing Protocol Specification. *Internet Draft*, March 1998, work in Progress.

[10] D. Estrin, D. Farinacci, D. Thaler, S. Deering, M. Handley, V. Jacobson, C. Liu, P. Sharma, and L. Wei. Protocol Independent Multicast-Sparse Mode (PIM-SM) Protocol Specification. *Internet Draft*, September 1997, work in Progress.

[11] T. Pusateri. Distance Vector Multicast Routing Protocol. *Internet Draft*, March 1998, work in Progress.

[12] J. Moy. Multicast extensions to OSPF. *RFC 1584*, Proteon, March 1994.

[13] D. Estrin, D. Farinacci, V. Jacobson, C. Liu, L. Wei, P. Sharma, and A. Helmy. Protocol Independent Multicast-Dense Mode (PIM-DM) Protocol Specification. *Internet Draft*, November 1995, work in Progress.

[14] G. Armitage. Support for multicast over UNI 3.0/3.1 based ATM networks. *RFC 2022*, November 1996.

[15] C. Shields. Oredered core based trees. *Master's thesis*. University of California, Santa Cruz, California, June 1996.

[16] C. Shields and J. J. Garcia-Luna-Aceves. The Ordered Core Based Tree Protocol. *Proceedings of the IEEE INFOCOM*, Kobe, Japan, April 1997.

[17] M. Handley, J. Crowcroft, and I. Wakeman. Hierarchical protocol independent multicast (HPIM). *University College London*, November 1995.

[18] D. Thaler, D. Estrin, and D. Meyer. Border Gateway Multicast Protocol (BGMP) Protocol Specification. *Internet Draft*, March 1998, work in Progress.

[19] C. Shields and J.J. Garcia-Luna-Aceve. The HIP Protocol for Hierarchical Multicast Routing. *Proc. Seventeenth Annual ACM SIGACT-SIGOPS Symposium on PRINCIPLES OF DISTRIBUTED COMPUTING (PODC 98)*, Puerto Vallarta, Mexico, 1998.

[20] E. N. Gilbert and H. O. Pollak. Steiner Minimal Trees. *SIAM Journal on Applied Mathematics*, Vol. 16, No. 1, pp. 1-29, January 1968.

[21] L. Kou, G. Markowsky, and L. Berman. A fast algorithm for Steiner trees. *Acta Informatica*, Vol. 15, pp. 141-145, 1981.

[22] M. Doar and I. Leslie. How bad is naive mulitcast routing. *In Proccedings of the IEEE Infocom'93*, 1993.

[23] B. M. Waxman. Routing of multipoint connections. *IEEE Journal on Selected Areas in Communications*, Vol. 6, No. 9, December 1988.

[24] L. Wei and D. Estrin. The trade-offs of multicast trees and algorithms. *In Proceedings of ICCCN'94*, pp. 17-24, 1994.

[25] G. N. Rouskas and I. Baldine. Multicast routing with end-to-end delay and delay variation constraints. *IEEE Journal on Selected Areas in communications*, Vol. 15, No. 3, pp. 346-356, April 1997.