Numéro d'ordre : 3249

Thèse

présentée devant

l'Université de Rennes I

pour obtenir le grade de

Docteur de l'Université de Rennes I

Spécialité : Informatique

par

Alexandre GUITTON

préparée dans le projet Armor à l'Irisa
au sein de la composante universitaire IFSIC
de l'école doctorale MATISSE
sous la direction de

Raymond MARIE et Miklós MOLNÁR

Communications *multicast*

Contributions aux réseaux optiques et au passage à l'échelle

Soutenue le 7 octobre 2005 (mention très honorable)

Devant le jury suivant :

M.	Éric	FLEURY	Professeur	*Rapporteur*
M.	Gérard	HÉBUTERNE	Professeur	*Président du jury*
M.	Jean-Claude	KÖNIG	Professeur	*Rapporteur*
M.	Raymond	MARIE	Professeur	*Directeur de thèse*
M.	Miklós	MOLNÁR	Maître de conférences	*Codirecteur de thèse*

Remerciements

Je tiens tout d'abord à remercier Miklós MOLNÁR pour avoir encadré mon travail de DEA et pour m'avoir proposé une thèse dans ce domaine passionnant des réseaux de télécommunications. Je suis honoré que Raymond MARIE ait accepté d'être le directeur de cette thèse. Grâce à eux, j'ai appris ce qu'était la recherche et j'ai grandement mûrit sur le plan scientifique.

Je remercie très chaleureusement Gérard HÉBUTERNE d'avoir accepté d'être le président du jury, ainsi qu'Éric FLEURY et Jean-Claude KÖNIG, d'avoir lu et rapporté ma thèse. Leurs remarques, commentaires et questions m'ont permis d'améliorer mon manuscrit.

I would like to thank Anikó ZSIGRI for the few months we worked together; I learned a lot from her at the beginning of my thesis. Je tiens aussi à remercier Ali BOUDANI, pour avoir partagé avec moi certaines de ses idées, mais aussi pour sa bonne humeur communicative qui fait que travailler avec lui fut un plaisir. Je ne sais pas comment remercier Joanna MOULIERAC pour les heures passées dans son bureau à réfléchir sur l'agrégation, mais aussi pour avoir su être là quand j'en ai eu besoin et pour m'avoir fait découvrir la capoeira.

J'apprécie les discussions avec Bernard COUSIN, et nous avons plusieurs fois été confrontés à des sujets toujours très intéressants. J'aurais aimé travailler davantage avec Christian LAFOREST : sa rigueur et sa clarté nous ont permis d'avancer très rapidement sur un sujet à la fois riche et prometteur.

Bien sûr, je remercie mes amis à Rennes. J'espère n'oublier personne en les citant rapidement : Alban (pour sa blague sur le Pim's), Alexandra, Annie, Anthony, Ariel, Audrey (pour le week-end sur les plages bretonnes), Catherine, Cécile, Christophe, Claudine, Cyrille, David, Élisa, Francine, François, Frédéric, Gilles (pour avoir enduré la balle rouge), Guillaume (pour les fraises tagada), Gurvan, Hervé, Jean-Marie (pour Morrowind, c'est-à-dire TTT), Julien, Kamal, Ludovic (pour Antwar), Martin, Olivier, Sabine (pour les flamenkuches), Sonia (pour m'avoir proposé de dormir sur le plancher), Thierry (pour avoir enduré Gilles), Virginie, Yézékaël.

Enfin, je remercie mes proches. Leurs contributions sont discrètes mais essentielles.

Table des matières

Introduction

Les réseaux de communication ont connu un essor très important pendant les dernières années. En prenant l'exemple d'Internet, l'augmentation du nombre d'internautes est bien visible. Cette augmentation a conduit à une prolifération des types applications mais aussi à une diversification des besoins.

Il y a une dizaine d'années, Internet était principalement utilisé pour la recherche. Les applications étaient alors destinées au courrier électronique, au transfert de fichiers et anecdotiquement à la navigation sur le Web. Les protocoles sous-jacents étaient encore peu nombreux et clairement identifiables : SMTP/POP, FTP et HTTP.

À présent, beaucoup de nouveaux types d'application se développent pour le grand public. On peut citer les applications multimédia, les applications de téléphonie, les applications d'échanges de fichiers en pair-à-pair ou les jeux temps-réel massivement distribués. Ces applications ont toutes des besoins différents. Les besoins des opérateurs et de leurs clients ont eux aussi évolué. Ils s'expriment par des exigences plus fortes sur les performances, comme le débit atteint ou le délai perçu par le client. Il est très difficile de satisfaire ces besoins avec une architecture ouverte comme l'est Internet, d'autant plus que ces besoins sont souvent liés à de nombreux paramètres.

De plus, à l'heure actuelle, beaucoup d'applications réseaux mettent en relation un ensemble d'émetteurs, appelés sources, à un ensemble de récepteurs, appelés destinations. À titre d'exemples d'applications utilisant clairement de telles communications de groupe, on peut citer les vidéo-conférences, les systèmes d'échanges de fichiers en pair-à-pair ou les jeux massivement distribués. D'autres applications utilisent souvent des communications de groupe de manière moins évidente. On peut citer par exemple les applications mettant à jour des bases de données (antivirus, mises à jour du système d'exploitation, etc.). Pour gérer ces communications, les applications actuelles recourent souvent au multi-*unicast* : une connexion est établie entre chaque source et chaque destination. Cette méthode est très coûteuse et affecte grandement les performances du réseau.

Le *multicast* est une réalisation efficace d'une communication de groupe. Il requiert que certains routeurs du réseau copient les paquets et les envoient vers plusieurs directions. Il a été standardisé pour le réseau Internet grâce aux travaux de DEERING au début des années 90.

Cependant, le *multicast* reste peu développé pour plusieurs raisons. On peut citer le surdimensionnement des réseaux qui masque actuellement la dégradation des performances due au multi-*unicast* ou encore les problèmes de tarification et de sécurité qui sont inhérents au *multicast*. Toutefois, il est grandement probable qu'à moyen terme, le *multicast* connaisse un essor important. En effet, le surdimensionnement des réseaux croît moins vite que les besoins des utilisateurs ; aussi, des modèles de tarification et des mécanismes de sécurité sont développés.

Le modèle du *multicast* décrit par DEERING ne suffira pas à couvrir tous les besoins et tous les types d'applications. D'une part, ce modèle est basé sur les contraintes et spécificités des réseaux des années 90, ce qui le rend inadapté aux réseaux de nouvelle génération. D'autre part, il n'est pas en mesure de supporter l'explosion prévisible de l'utilisation du *multicast*.

Dans cette thèse, nous proposons des modèles complémentaires au modèle de Deering.

Plan de la thèse

Dans le chapitre 1, nous présentons l'intérêt du *multicast*. Puis, nous présentons le modèle utilisé actuellement sur Internet, le modèle de Deering, ainsi que les protocoles les plus souvent implémentés dans Internet. Puis, nous mettons en avant les limites de ce modèle et justifions ainsi l'utilisation de modèles complémentaires.

La suite de cette thèse est organisée en deux parties. Dans la première, constituée du chapitre 2, nous mettons en avant l'inadéquation du modèle de Deering pour les réseaux de nouvelle génération au travers du cas des réseaux tout optique. Dans la seconde partie de cette thèse, constituée des chapitres 3 et 4, nous proposons des solutions pour résoudre le problème de passage à l'échelle du *multicast*.

Dans le chapitre 2, nous nous concentrons sur les réseaux tout-optique, capables de transmettre très rapidement de grandes quantités d'information. Cette vitesse est obtenue par un examen sommaire de chaque paquet ne permettant pas la consultation de l'adresse des destinataires du paquet. Ce fait est en contradiction avec l'une des hypothèses du modèle de Deering. De plus, pour des raisons technologiques ou financières, les routeurs tout optique sont parfois incapables d'effectuer des opérations que les routeurs traditionnels réalisent. Ces spécificités conduisent à un changement de paradigme. Nous détaillons dans ce chapitre ces spécificités des réseaux tout optique et nous montrons en quoi le modèle de Deering n'est pas adapté pour obtenir les meilleures performances. Nous proposons plusieurs heuristiques permettant d'économiser les ressources critiques. De plus, nous identifions un problème nécessitant la construction de nouvelles structures de routage *multicast*, différentes des arbres traditionnels.

Ensuite, nous étudions la seconde problématique : le passage à l'échelle. Les problèmes liés au passage à l'échelle sont la complexité temporelle des algorithmes, la mémoire requise dans les routeurs et la gestion des informations distribuées lorsque le nombre de groupes devient très important. Pour faciliter la gestion de nombreux groupes *multicast*, nous analysons deux

propositions : l'élimination des états de routage *multicast* dans les routeurs et la diminution du nombre d'arbres à gérer. Ces deux méthodes ont l'avantage de réduire aussi la quantité de messages de contrôle nécessaires.

Dans le chapitre 3, nous nous axons sur le support d'un grand nombre de petits groupes *multicast* concurrents grâce au routage *multicast* explicite. De tels groupes de petite taille vont vraisemblablement occuper une place importante parmi les groupes *multicast*, car ils ont de nombreuses applications telles que la radio sur Internet ou les jeux distribués. On peut aussi mentionner que le *multicast* pour les petits groupes est prometteur pour gérer la mobilité ainsi que dans les réseaux sans infrastructure fixe. Nous proposons une analyse étendue du routage *multicast* explicite, qu'il soit plat ou arborescent, depuis sa justification jusqu'aux méthodes de déploiement incrémental.

Dans le chapitre 4, nous décrivons un mécanisme permettant de gérer un grand nombre de groupes *multicast* de taille quelconque. Pour cela, plusieurs groupes sont agrégés au même arbre *multicast* dans un domaine. Tout d'abord, nous montrons en quoi l'agrégation d'arbres permet au *multicast* de passer à l'échelle. Puis, nous étudions les compromis sur lesquels l'agrégation d'arbres repose. Nous montrons comment il est possible d'utiliser l'agrégation d'arbres pour gérer les besoins en qualité de service des groupes. Enfin, nous proposons un mécanisme de distribution très efficace, nécessitant très peu de messages de contrôle pour garder les informations à jour.

Finalement, nous concluons sur ce travail de thèse. Nous soulevons plusieurs questions restées en attente et donnons les perspectives du travail réalisé.

Démarche et moyens

Tout au long de cette thèse, nous tentons d'avoir la même démarche. Dans un premier temps, notre approche est algorithmique. Une fois les problèmes identifiés et formalisés, nous tentons de trouver des algorithmes permettant de le résoudre efficacement. Dans un second temps, notre approche est protocolaire. Nous tentons d'inclure les algorithmes développés au sein de protocoles performants.

Systématiquement, nous comparons les performances de nos propositions avec celles des propositions existantes par des simulations. Comme nos comparaisons sont faites sur des critères macroscopiques qui caractérisent les structures (chemins ou arbres) plutôt que microscopiques (nombre de paquets envoyés), nos simulations peuvent elles aussi être macroscopiques. Nous n'utilisons donc pas le logiciel NS (pour *Network Simulator*), mais des programmes travaillant directement sur la topologie. Les graphes choisis pour les simulations sont Abilène en tant que petit réseau et Renater ou Eurorings en tant que grands réseaux. Ils correspondent à des réseaux réels d'opérateurs. Les résultats obtenus par les simulations sont similaires sur des graphes aléatoires réalistes de même taille que nos graphes de tests.

Chapitre 1

Modèle de Deering

Le modèle de routage *multicast* proposé par Deering dans [Dee91] a été adopté il y a une quinzaine d'années par l'IETF (pour *Internet Engineering Task Force*), organisme de standardisation de l'Internet. Cependant, il reste peu déployé à l'heure actuelle. Ce faible déploiement peut s'expliquer en partie par des problèmes de passage à l'échelle de la gestion des groupes.

Dans ce chapitre, nous décrivons tout d'abord les objectifs du routage *multicast*, le modèle de Deering et finalement ses limites qui ont causé le retard de son déploiement.

1.1 Le *multicast*

Les réseaux point-à-point, comme Internet, sont basés sur des communications entre une source et une destination. Ces communications, dites « *unicast* », représentent actuellement la majeure partie du trafic Internet.

Il y a une quinzaine d'années, le besoin de nouveaux types de communications a conduit Deering à proposer un modèle de communications entre une source et plusieurs destinations, ou entre plusieurs sources et plusieurs destinations.

1.1.1 Les communications de groupe

Une communication de groupe met en jeu plusieurs participants. Les participants peuvent être des sources, c'est-à-dire des émetteurs de messages, ou des destinataires, c'est-à-dire des récepteurs de messages. Bien entendu, un participant peut être à la fois source et destinataire.

Le besoin de faire communiquer des groupes est très fréquent dans les applications. À titre d'exemple, on peut mentionner les applications de vidéo-conférence, les applications de diffusion de radio, de vidéo à la demande ou encore les jeux multi-joueurs. À l'heure actuelle,

les applications utilisent souvent une méthode appelée le « multi-*unicast* » : une communication *unicast* est créée pour chaque couple (source,destination) possible. La figure 1.1(a) présente cette méthode. Sur la figure, le nuage représente un réseau et les carrés des hôtes du réseau, c'est-à-dire des machines utilisateurs. Pour transmettre des données aux cinq destinataires, la source doit envoyer cinq messages, un par destinataire. Cela encombre à la fois la source et le réseau.

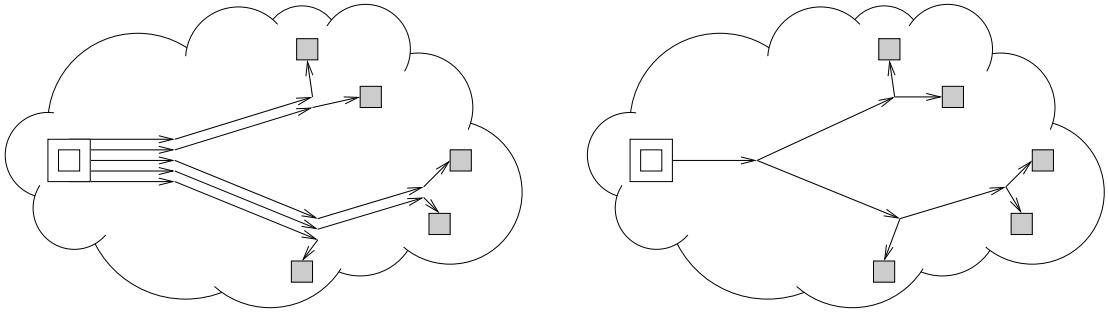


Figure 1.1 – *Le multi-unicast et le multicast.*

Une méthode plus efficace pour la source est d'envoyer un unique message, dupliqué au cours de son acheminement. Cette méthode est présentée sur la figure 1.1(b). Le réseau est beaucoup moins surchargé puisqu'il n'y a plus de messages redondants. Cette méthode de diffusion efficace d'un message à un groupe est appelé le « *multicast* ».

L'objectif principal du *multicast* est de minimiser la charge du réseau. D'un point de vue théorique, nous nous intéressons donc à la recherche d'une structure de coût minimum dans le graphe[1] représentant le réseau, et couvrant les participants de la communication de groupe. En remarquant que la structure de coût minimum est un arbre[2] (en l'absence de contraintes supplémentaires), la recherche d'un tel arbre est formulée par le problème de Steiner dans les réseaux.

1.1.2 Le problème de Steiner dans les réseaux

La recherche d'un arbre de coût minimum parmi ceux couvrant la source et les membres d'un groupe *multicast* est connue sous le nom de problème de Steiner dans les réseaux. Ce problème a été formulé par HAKIMI dans [Hak71].

Problème 1 (Problème de Steiner). *Soit $G = (V, E)$ un graphe et $Z \subset V$ un ensemble de nœuds. Le problème de Steiner dans les réseaux consiste à trouver un arbre T^* couvrant*

1. En théorie des graphes, un « réseau » est un graphe orienté valué. Dans cette thèse, nous nommerons réseau l'ensemble des dispositifs physiques permettant d'acheminer des messages. Comme les graphes que nous considérons sont toujours orientés symétriques, nous simplifions souvent la discussion ou les figures en supposant qu'ils sont non orientés (sauf mention du contraire).

2. Le terme le plus approprié est en fait « arborescence », puisque la source du groupe est généralement identifiée comme racine. Cependant, le terme « arbre » est le plus utilisé dans la littérature. C'est ce terme que nous emploierons tout au long de cette thèse.

Z, tel que, pour tout arbre T couvrant Z, on ait :

$$w(T^*) \leq w(T),$$

$w(T)$ étant le nombre d'arêtes d'un arbre T.

Lorsque $Z = V$, des algorithmes polynomiaux existent pour résoudre ce problème : l'algorithme de Prim [Pri57] et l'algorithme de Kruskal [Kru56]. Dans le cas plus général où $Z \neq V$, KARP a montré que le problème de Steiner était NP-complet (voir [Kar72]).

Algorithmes exacts

Il existe plusieurs algorithmes exacts au problème de Steiner dans les réseaux, détaillés dans [Win87]. Nous n'allons présenter que l'algorithme STEA et l'algorithme TEA.

L'algorithme STEA. L'algorithme STEA (pour *Spanning Tree Enumeration Algorithm*) a été proposé dans [Hak71]. Il se base sur la détermination des nœuds de branchement de l'arbre. Pour cela, il énumère tous les sous-ensembles de $V \backslash Z$. Pour chaque sous-ensemble S, un arbre couvrant T_S de poids minimum est construit dans le graphe des distances de $Z \cup S$. Parmi tous ces arbres, celui de coût minimum est l'arbre T^* recherché.

La complexité temporelle dans le pire des cas de l'algorithme STEA est $\mathcal{O}(|Z|^2 \cdot 2^{|V|-|Z|} + |V|^3)$.

L'algorithme TEA. L'algorithme TEA (pour *Topology Enumeration Algorithm*) a lui aussi été proposé dans [Hak71]. Cet algorithme utilise le fait qu'un arbre couvrant $|Z|$ feuilles a au plus $|Z|-2$ nœuds de branchement. Il énumère toutes les topologies d'arbres couvrants.

La complexité temporelle dans le pire des cas de l'algorithme TEA est $\mathcal{O}(|V|^3 \cdot 2^{|Z|})$.

Heuristiques

Utiliser des algorithmes exacts pour le problème de Steiner dans les réseaux n'est pas réaliste dans le contexte de communications réseaux, étant donnée la complexité du problème. En effet, les calculs doivent généralement être rapides.

Cependant, il existe des heuristiques permettant d'obtenir en temps polynomial des arbres dont le coût est garanti faible. Cette garantie de performance s'obtient mathématiquement en prouvant que dans le pire des cas, le coût de l'arbre obtenu par l'heuristique reste proche du coût de l'arbre optimal.

Définition 1 (r-approximation). *Une heuristique $\mathcal{H}$ est une r-approximation d'un problème de minimisation $\mathcal{P}$ si pour toute instance $\mathcal{I}$ de $\mathcal{P}$ on a :*

$$w(\mathcal{S}_{\mathcal{I}}^{\mathcal{H}}) \leq r \cdot w(\mathcal{S}_{\mathcal{I}}^{*}),$$

où $\mathcal{S}_{\mathcal{I}}^{\mathcal{H}}$ est la solution au problème $\mathcal{P}$ obtenue par l'heuristique $\mathcal{H}$ sur l'instance $\mathcal{I}$, $\mathcal{S}_{\mathcal{I}}^{*}$ est la solution optimale du problème sur l'instance $\mathcal{I}$, et $w(\mathcal{S})$ est le coût d'une solution $\mathcal{S}$.

L'heuristique TM et l'heuristique KMB, que nous présentons dans la suite, sont toutes deux des 2-approximations.

L'heuristique TM. L'heuristique TM (pour TAKAHASHI et MATSUYAMA, les auteurs) a été proposée dans [TM80]. C'est une heuristique qui construit un arbre T itérativement. L'arbre T est initialisé en un nœud quelconque[3] de Z. À chaque itération, un nœud de Z non encore connecté à T est ajouté à T au moyen d'un plus court chemin. Le nœud ajouté à T est celui qui en est le plus proche.

La complexité dans le pire des cas de l'heuristique TM est $\mathcal{O}(|Z| \cdot |E| + |Z| \cdot |V| \log |V|)$.

L'heuristique KMB. L'heuristique KMB (pour KOU, MARKOWSKY et BERMAN, les auteurs) a été proposée dans [KMB81]. Elle se déroule en trois phases. À la première phase, elle construit la fermeture métrique $\bar{G}$ des nœuds de Z. À la deuxième phase, elle construit $\bar{T}$, un arbre couvrant de poids minimum sur $\bar{G}$. À la troisième phase, elle projette $\bar{T}$ dans G afin d'obtenir un arbre T en supprimant les arêtes redondantes.

La complexité dans le pire des cas de l'heuristique KMB est $\mathcal{O}(|Z|^2 + |E| + |V| \log |V|)$.

1.1.3 La réalisation du *multicast*

Nous venons de voir le *multicast* d'un point de vue théorique. Dans ce paragraphe, nous étudions quelques aspects liés à la réalisation du *multicast* dans un réseau de communication réel. Nous distinguons la réalisation dans le plan de contrôle et dans le plan de données.

Plan de contrôle

Afin d'acheminer les paquets à leurs destinations, il est nécessaire de prendre des décisions de routage, c'est-à-dire déterminer le trajet que les paquets suivront. Le plan de contrôle régit l'ensemble des mécanismes impliqués par cette détermination.

Il est aussi nécessaire d'intégrer au réseau la fonctionnalité de routage *multicast*. Cela passe par la constitution de protocoles *multicast* capables d'échanger des informations sur les groupes du réseau.

Comme une machine peut être source de plusieurs groupes, il est important de pouvoir nommer un groupe afin de déterminer à quels destinataires les paquets sont destinés. De plus, il est important de pouvoir gérer dynamiquement les adhésions et départs des membres d'un groupe. La notion purement logique de groupe augmente la complexité des protocoles à déployer.

Plan de données

La réalisation du *multicast* au niveau du plan de données inclus entre autres la manière dont les paquets sont traités dans les routeurs, la manière dont les paquets sont copiés dans les routeurs et le temps nécessaire aux destinataires pour recevoir les messages émis par la source.

Le traitement des paquets dans les routeurs est soumis à plusieurs contraintes : il doit être rapide et pouvoir être implémenté efficacement en matériel.

3. Quand l'heuristique TM est utilisée pour le *multicast*, le nœud initial sélectionné est souvent la source du groupe.

La copie des paquets dans les routeurs est un élément clé du *multicast*. Cette copie doit être rapide afin de ne pas surcharger le routeur.

Le temps nécessaire aux destinataires pour recevoir les messages émis par la source est souvent appelé le délai. Le délai réel est très difficile à calculer, puisqu'il est dépendant de tous les autres paquets du réseau. Nous ferons souvent l'hypothèse que le délai ne dépend que du temps de transmission sur les arêtes, en négligeant donc le temps de traitement dans les routeurs. Ainsi, le délai pour atteindre un destinataire sera dit minimal quand le paquet suit un plus court chemin de la source à ce destinataire. Il existe deux grands types d'arbres basés sur des plus courts chemins : les arbres des plus courts chemins et les arbres des plus courts chemins inverses.

Arbre des plus courts chemins. Un arbre des plus courts chemins est une arborescence dont la racine est la source du groupe, telle que le chemin de la source à chaque destinataire est un plus court chemin. Si la métrique considérée est le délai de propagation sur les liens, un arbre des plus courts chemins induit un délai minimal pour tous les destinataires.

Arbre des plus courts chemins inverses. Un arbre des plus courts chemins inverses est une arborescence dont la racine est la source du groupe, telle que le chemin de chaque destinataire à la source est un plus court chemin. Bien qu'un arbre des plus courts chemins inverses ne soit pas en général (lorsque la valuation des deux arcs d'une même arête est différente, ce qui est souvent le cas sur Internet) un arbre des plus courts chemins, il est souvent construit par les protocoles.

1.2 Le modèle de Deering

Le modèle de Deering est basé sur le modèle OSI (pour *Open Systems Interconnect*) de l'ISO (pour *International Standards Organization*) qui définit une norme à sept couches. Ces sept couches sont la couche physique, la couche liaison de données, la couche réseau, la couche transport, la couche session, la couche présentation et la couche application. Chaque couche a une tâche bien précise :

La couche physique est responsable de la transmission élémentaire sur un canal de communication. L'unité d'information manipulée par cette couche est le bit.

La couche liaison de données est responsable de l'encodage et du décodage des paquets de données en flux de bits appelés trames. Cette couche est séparée en deux sous-couches : la sous-couche LLC (pour *Logical Link Control*) et la sous-couche MAC (pour *Media Access Control*). La sous-couche LLC contrôle le flux de bits, la synchronisation des trames et effectue un contrôle d'erreur. La sous-couche MAC fournit à la couche supérieure les primitives pour injecter ou extraire des données.

La couche réseau est responsable de l'acheminement et du routage. L'acheminement est la transmission des paquets de proche en proche. Le routage est la détermination du chemin qu'un paquet donné doit suivre. Les responsabilités qui incombent à la couche réseau incluent donc l'adressage (afin d'identifier un routeur destination donné), le contrôle de congestion et le séquencement des paquets.

La couche transport est responsable de la transmission transparente de messages entre deux hôtes. Cette couche gère le contrôle de flux et la récupération d'erreurs de bout en bout. Elle assure que les messages transmis sont complets.

La couche session est responsable de la création, du maintien et de la destruction des connexions entre applications.

La couche présentation est responsable de l'indépendance des données par rapport au format utilisé par le réseau.

La couche application est responsable des services fournis par les applications. Les protocoles HTTP, POP3, SMTP et FTP cités en introduction font partie de cette couche.

Selon cette norme à sept couches, les seules interactions possibles sont entre couches adjacentes.

Le modèle de Deering est basé sur la couche réseau. Il s'agit d'un modèle ASM (pour *Any-Source Multicast*) : tout hôte peut émettre des messages pour un groupe quelconque. Ce modèle ne permet donc pas de contrôler les sources d'un groupe. Il repose sur une gestion distribuée des groupes, un adressage global des groupes, un acheminement unidirectionnel ou bidirectionnel et un routage à trois niveaux.

Dans la suite de ce paragraphe, nous présentons le modèle de Deering. Une description détaillée se trouve dans [Ram00].

1.2.1 Adressage

L'adressage dans le modèle de Deering est global. Un groupe g est identifié en IPv4 par une adresse de classe D [RFC 1112], c'est-à-dire comprise entre `224.0.0.0` et `239.255.255.255`. En IPv6, un groupe g est identifié par une adresse dont le préfixe est `FF::/8` [RFC 3513].

Les adresses *multicast* sont attribuées par le protocole MADCAP, le protocole GLOP (seulement en IPv4) ou l'utilisation d'adresses *multicast* basées sur un préfixe *unicast* (seulement en IPv6). Ces protocoles permettent d'éviter que les adresses de groupes entrent en collision, c'est-à-dire que deux groupes se voient attribuer la même adresse.

Le protocole MADCAP

Le protocole MADCAP (pour *Multicast Address Dynamic Client Allocation Protocol*) est décrit dans [RFC 2730]. Le protocole MADCAP est compatible avec IPv4 et IPv6. Fonctionnellement, il est très similaire au protocole DHCP (pour *Dynamic Host Configuration Protocol*) [RFC 2131][4].

Les clients MADCAP sont les hôtes qui désirent créer un groupe *multicast*. Ils font des requêtes à des serveurs MADCAP, appelés serveurs MAAS (pour *Multicast Address Allocation Server*), en *unicast* par une adresse configurée dynamiquement en utilisant DHCP par exemple, ou en *multicast*, par une adresse attribuée statiquement par l'ICANN (pour *Internet Corporation for Assigned Names and Numbers*)[5]. Chaque serveur MAAS étant responsable d'une plage d'adresses qui lui a été attribué (statiquement ou dynamiquement), il

4. D'ailleurs, MADCAP se nommaient initialement MDHCP (pour *Multicast Dynamic Host Configuration Protocol*).

5. L'ICANN est le nouveau nom de l'organisme IANA (pour *Internet Assigned Numbers Authority*).

n'y a pas de collisions entre les adresses. L'attribution des plages d'adresses *multicast* aux serveurs MAAS est détaillée plus loin.

Le protocole GLOP

Le protocole GLOP (pour *Global Allocation Protocol*) est décrit dans [RFC 3180]. Il est spécifique à IPv4. GLOP n'alloue que des adresses *multicast* qui sont comprises entre `233.0.0.0` et `233.255.255.255`. Il fait correspondre à chaque système autonome une plage de 256 adresses. Pour cela, GLOP utilise le numéro d'un système autonome qui est de la forme `x.y`. Lorsqu'une source d'un système autonome `x.y` demande une adresse *multicast*, un serveur GLOP lui fournit une adresse comprise entre `233.x.y.0` et `233.x.y.255`.

Chaque serveur GLOP étant responsable d'une plage d'adresse, il est à même d'éviter les collisions.

Adresses *multicast* basées sur un préfixe *unicast*

Dans IPv6, les adresses *multicast* peuvent être basées sur un préfixe *unicast* selon la méthode décrite dans [RFC 3306]. Une partie de l'adresse *unicast* de la source est incluse dans l'adresse *multicast* pour éviter les collisions. En reprenant l'exemple de [RFC 3306], un réseau dont le préfixe *unicast* est `3FFE:FFFF:0001::/48` aura un préfixe *multicast* basé sur un préfixe *unicast* de la forme `FF3x:0030:3FFE:FFFF:0001::/96`, où `x` peut prendre une valeur quelconque.

Cette technique est très prometteuse puisqu'elle interdit les collisions et le nombre de protocoles d'adressage à déployer. Cependant, il semble qu'elle soit plus adaptée au modèle SSM (pour *Source Specific Multicast*) qu'au modèle ASM (pour *Any Source Multicast*), sur lequel est basé le modèle de Deering.

1.2.2 Acheminement des paquets

L'acheminement est le procédé qui consiste à transmettre un paquet reçu sur une ou plusieurs interfaces de sortie jusqu'à sa destination. Dans le modèle de Deering, cet acheminement n'a lieu que si le paquet en question provient de certaines interfaces d'entrée. Cela permet d'éviter d'une part les paquets redondants et d'autre par que des personnes malveillantes ne se fassent passer pour des sources pour causer un déni de service. Il est en effet important de réduire les possibilités de déni de service *multicast*, qui sont par nature plus dangereuses qu'en *unicast*.

Dans le modèle de Deering, l'acheminement des paquets sur un arbre *multicast* préalablement construit peut être unidirectionnel ou bidirectionnel.

Acheminement unidirectionnel

Considérons un paquet p reçu par un routeur r depuis une interface i, à destination d'un groupe *multicast* dont l'adresse est g. Le routeur r peut extraire du paquet p l'adresse s de la source.

Pour acheminer p de manière unidirectionnelle, r consulte sa table d'acheminement à la recherche d'une entrée (s, g). S'il n'existe pas de telle entrée, p est détruit. Si une telle entrée existe, elle fait correspondre à (s, g) un ensemble I d'interfaces d'entrées et un ensemble O d'interfaces de sortie. Si $i \notin I$, p est détruit. Sinon, une copie de p est envoyée sur chaque interface de sortie de O. Nous avons toujours $I \cap O = \emptyset$ et $|I| \leq 1$.

La recherche de l'entrée (s, g) dans la table d'acheminement *multicast* est une recherche exacte en *multicast* alors que c'est une recherche du plus long préfixe commun en *unicast*. Cela vient du fait que les adresses *unicast*, qui ont une signification géographique ou topologique, peuvent être agrégées facilement. Les adresses *multicast*, qui n'ont qu'une signification logique, ne peuvent pas être agrégées. Elles ont donc un masque de longueur maximale.

L'acheminement unidirectionnel est associé aux groupes ayant peu de sources. L'avantage est que le routage est dépendant de la source, il peut donc être optimal pour chaque source. Par contre, le nombre d'états total dans les routeurs dépend du nombre de couples (s, g).

Acheminement bidirectionnel

Considérons un paquet p reçu par un routeur r depuis une interface i, à destination d'un groupe *multicast* dont l'adresse est g.

Pour acheminer p de manière bidirectionnelle, r consulte sa table d'acheminement en cherchant une entrée $(*, g)$. S'il n'existe pas de telle entrée, p est détruit. Si une telle entrée existe, elle fait correspondre à $(*, g)$ un ensemble B d'interfaces. Si $i \notin B$, p est détruit. Sinon, une copie de p est envoyée sur chaque interface de $B \backslash \{i\}$.

La recherche de l'entrée $(*, g)$ dans la table d'acheminement *multicast* est une recherche exacte.

L'acheminement bidirectionnel est associé aux groupes ayant un grand nombre de sources. L'avantage est que le nombre d'états ne dépend pas du nombre de sources. Par contre, le routage est indépendant de la source, donc il ne peut pas respecter les besoins particuliers de chaque source.

Exemple d'acheminement

Un exemple d'acheminement unidirectionnel sur un arbre est présenté sur la figure 1.2(a). La source s envoie ses messages en *unicast* à la racine c de l'arbre. Puis, le message de la source est acheminé en *multicast* le long de l'arbre. Les messages *multicast* suivent toujours l'arbre dans le même sens, quelque soit la source qui envoie les données.

Un exemple d'acheminement bidirectionnel sur un arbre est présenté sur la figure 1.2(b). La source s envoie ses messages en *unicast* vers la racine c de l'arbre. Dès qu'un routeur de l'arbre intercepte ces messages, il les achemine en *multicast* comme s'il était la racine de l'arbre. Les messages *multicast* peuvent suivre l'arbre dans les deux sens.

1.2.3 Routage

Le routage est le procédé qui consiste à déterminer les interfaces de sortie d'un paquet.

Dans le modèle de Deering, le routage *multicast* est basé sur une architecture à trois niveaux : local, intra-domaine et inter-domaines. Les hôtes informent leurs routeurs désignés

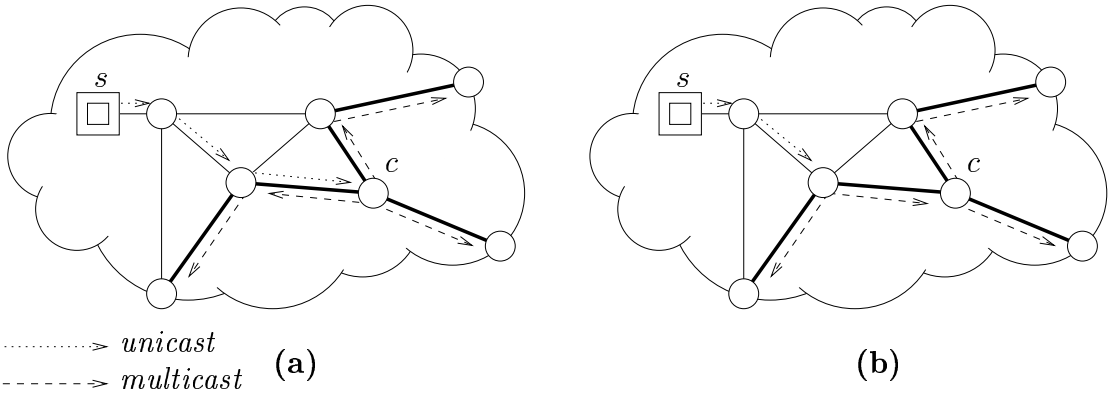


Figure 1.2 – *Acheminement unidirectionnel et bidirectionnel sur un arbre.*

de leurs appartenances aux groupes (routage local). Les routeurs d'un même domaine collaborent pour réaliser le routage jusqu'aux extrémités de ce domaine (routage intra-domaine). Finalement, les routeurs de bordure des domaines collaborent pour décider du routage global (routage inter-domaines).

Le routage local

Il existe deux protocoles réalisant le routage local : IGMP pour IPv4 et MLD pour IPv6.

Le protocole IGMP. Le protocole IGMP (pour *Internet Group Management Protocol*) est conçu pour IPv4. Son champ d'action se limite aux réseaux locaux d'un routeur *multicast*. IGMPv1 est décrit dans l'annexe 1 de [RFC 1112], IGMPv2 est décrit dans [RFC 2236] et IGMPv3 est décrit dans [RFC 3376].

Le mécanisme du protocole IGMP est illustré par la figure 1.3. Un routeur IGMP r associe à chaque groupe *multicast* l'ensemble de ses sous-réseaux contenant au moins un membre de ce groupe. Pour maintenir cette information, il envoie périodiquement un message `Query` en *multicast* (à l'adresse `224.0.0.1` sous IPv4, qui est l'adresse des membres d'au moins un groupe) sur tous ses sous-réseaux (message 1). Les hôtes membres d'au moins un groupe *multicast*, h_1 et h_3 sur l'exemple, répondent à ce message par un `Report` (message 2). Pour éviter une explosion du nombre de réponses envoyées, deux méthodes sont utilisées :

- plutôt que d'envoyer le `Report` instantanément après avoir reçu le `Query`, chaque hôte attend un temps aléatoire ;
- les messages `Report` sont envoyés à l'adresse du groupe afin que les autres membres du groupe puissent le recevoir.

Un hôte (h_3 sur l'exemple) qui reçoit un `Report` d'un autre hôte (h_1 sur l'exemple) pour un groupe auquel il appartient ne génère pas de `Report`. Si le routeur IGMP n'a pas reçu de `Report` après avoir envoyé un certain nombre de `Query`, il considère que le groupe n'est plus actif sur ce sous-réseau et ne transmet plus de paquets *multicast* pour ce groupe, sur ce lien.

IGMPv2 ajoute un mécanisme d'élection du routeur en charge d'envoyer les messages `Query` et de collecter les messages `Report`. De plus, IGMPv2 introduit les messages `Leave` et `Group-Specific-Query`. Ces messages permettent à IGMPv2 de détecter rapidement s'il

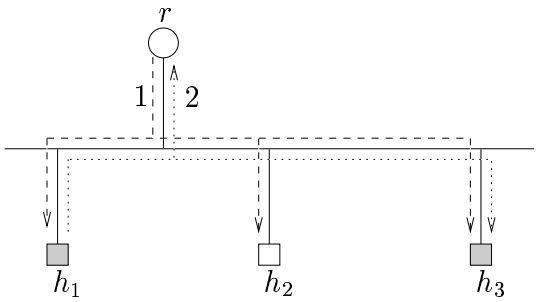


Figure 1.3 – *Le protocole IGMP.*

reste des membres dans un groupe : le message `Leave` est envoyé par un hôte qui quitte le groupe s'il est celui qui a émis le dernier `Report` ; le routeur IGMP interroge alors le groupe avec le message `Group-Specific-Query` et si aucun hôte ne répond, le routeur considère qu'il n'y a plus de membres actifs dans ce sous-réseau. IGMPv2 permet donc une plus grande dynamique des groupes.

IGMPv3 intègre des messages de changements d'état, l'agrégation des messages `Report` et le filtrage des sources par les hôtes.

Le protocole MLD. Le protocole MLD (pour *Multicast Listener Discovery*) est conçu pour IPv6. MLDv1 possède les fonctionnalités d'IGMPv2 et MLDv2 possède les fonctionnalités d'IGMPv3.

Le routage intra-domaine

Les protocoles intra-domaine sont en charge d'acheminer les paquets *multicast* au sein d'un domaine de routage. Les hypothèses du routage intra-domaine incluent souvent :

- une topologie de taille petite à moyenne,
- l'unicité de la politique de routage,
- la disponibilité de la bande passante.

Il existe beaucoup de protocoles intra-domaine. Les protocoles standardisés sont les protocoles DVMRP, MOSPF, PIM-SM, PIM-SSM et CBT.

Le protocole DVMRP. Le protocole DVMRP (pour *Distance-Vector Multicast Routing Protocol*) est décrit dans [RFC 1075]. Il s'agit d'un protocole à vecteur de distances, comme le protocole RIP [RFC 2453] pour l'*unicast*. Dans un protocole à vecteur de distances, chaque routeur dispose d'un vecteur de distances supposées qu'il transmet périodiquement à ses voisins. Lorsqu'un routeur reçoit un vecteur de distances d'un voisin, il met à jour son propre vecteur de distances et le transmet à son tour à ses voisins. Au bout de plusieurs itérations (parfois nombreuses), chaque routeur possède une vision stable des distances aux autres routeurs du domaine (si la topologie est stable).

Le protocole DVMRP utilise une table de routage et une table d'acheminement. La table d'acheminement sert à acheminer les paquets. Elle est construite au moyen des informa-

tions contenues dans la table de routage. L'acheminement dans le protocole DVMRP est unidirectionnel.

La table de routage contient pour chaque groupe g de source s un arbre des plus courts chemins inverses, basé en s, et couvrant tous les routeurs du domaine. Pour cela, le mécanisme RPM (pour *Reverse Path Multicasting*) et le vecteur de distances sont utilisés. L'arbre est construit au premier paquet *multicast* reçu par le routeur.

La table d'acheminement contient une version élaguée de l'arbre contenu dans la table de routage. L'élagage se fait comme suit : les feuilles d'un arbre qui ne sont pas concernées par le trafic *multicast* envoient un message `Prune` à leur routeur père dans l'arbre. Le routeur père supprime de la table d'acheminement l'interface de sortie amenant à ce fils pour le groupe (s, g) pour un temps donné. Périodiquement, le routeur père rattache les routeurs fils à l'arbre en ré-émettant sur toutes les interfaces de l'arbre.

Considérons l'exemple de la figure 1.4 représentant un domaine DVMRP. La figure 1.4(a) présente la construction initiale de l'arbre *multicast*. L'hôte h_1 prévient son routeur désigné d qu'il désire adhérer au groupe g par un message IGMP `Report` (message 1). Des messages (non représentés) de création de l'arbre sont inondés dans le réseau. Le premier paquet envoyé par s est envoyé sur un arbre total des plus courts chemins inverses, basé en s (message 2). Les routeurs feuilles de l'arbre qui n'ont pas de membres rattachés (b et e) envoient un message d'élagage à leur père (message 3). L'arbre construit à cette étape est représenté en traits épais. La figure 1.4(b) présente la situation lorsque l'hôte h_2 joint le groupe *multicast*. h_2 envoie un message IGMP `Report` à son routeur désigné e (message 4). e est effectivement rattaché à l'arbre quand son père c ré-émet sur toutes ses interfaces (message 5).

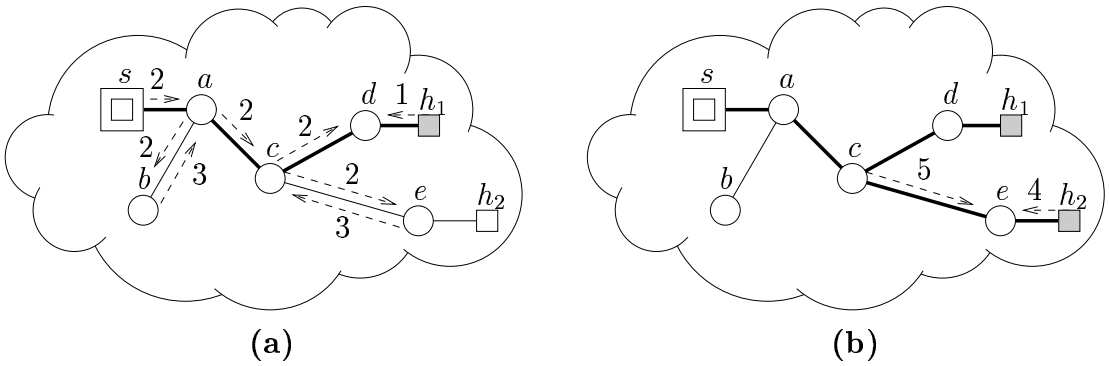


Figure 1.4 – *Le protocole DVMRP.*

Une critique du protocole DVMRP est qu'il suppose que le routage est symétrique : le chemin d'un nœud a à un nœud b doit être l'opposé du chemin de b à a. Ce n'est pas le cas sur Internet. Cette hypothèse conduit à faire des arbres sur lesquels le délai de la source jusqu'aux destinataires n'est pas minimal. Une autre critique que l'on peut faire au protocole DVMRP est qu'il a tendance à utiliser beaucoup de bande passante pendant les phases d'inondations et d'élagages périodiques.

Le protocole MOSPF. Le protocole MOSPF (pour *Multicast Extensions to OSPF*) est décrit dans [RFC 1584]. Il s'agit d'un protocole à état de liens basé sur le protocole

OSPF [RFC 2328] pour l'*unicast*. Le protocole MOSPF utilise le protocole OSPF pour connaître la topologie et les informations sur les membres des groupes.

Dès qu'un groupe change, le routeur qui détecte ce changement diffuse à tous les routeurs du réseau un message de notification. Ainsi, tous les routeurs sont rapidement informés de l'appartenance des membres à un groupe.

Le protocole MOSPF utilise un cache de routage et une table d'acheminement. Le cache de routage sert à construire la table d'acheminement. L'acheminement dans le protocole MOSPF est unidirectionnel.

Lorsqu'un routeur MOSPF reçoit un paquet *multicast* pour un groupe g avec une source s, il construit un arbre basé sur s des plus courts chemins et couvrant les membres du groupe, grâce à sa connaissance de la topologie. L'arbre est placé dans le cache de routage pour éviter de le recalculer à chaque paquet. La table d'acheminement est remplie avec les interfaces menant aux fils du routeur dans l'arbre actuel. Le routage est consistant car tous les routeurs MOSPF calculent le même arbre des plus courts chemins, le calcul étant toujours initié à la source s. Les routeurs MOSPF qui ne sont pas sur l'arbre d'un groupe n'ont pas à maintenir d'état ou à calculer d'arbre pour ce groupe.

Un routeur MOSPF est obligatoirement un routeur OSPF. Toutefois, la réciproque n'est pas vraie. Les différences principales entre les protocoles OSPF et MOSPF suivent :

- dans le protocole MOSPF, l'arbre est calculé à partir de la source tandis que dans le protocole OSPF, l'arbre est calculé à partir du routeur qui calcule ;
- dans le protocole MOSPF, les sous-réseaux des routeurs sont ignorés ;
- dans le protocole MOSPF, les interfaces d'entrées sont aussi stockées dans la table d'acheminement.

MOSPF est considéré lent lorsque la dynamicité des groupes est importante (*cf* [Ram00]). En effet, quand les groupes sont très dynamiques, une grande partie de la bande passante est utilisée pour la distribution des informations concernant les groupes. De plus, les routeurs MOSPF vident l'entrée du cache de routage et recalculent un nouvel arbre pour un groupe donné à chaque fois que ce groupe change.

Le protocole PIM-SM. Le protocole PIM-SM (pour *Protocol Independent Multicast - Sparse Mode*) est décrit dans [RFC 2362]. Il se base sur la table d'acheminement *unicast*, et non pas sur le protocole *unicast* lui-même, ce qui le rend indépendant au protocole *unicast* sous-jacent [6].

Le protocole PIM-SM construit un arbre partagé par toutes les sources d'un groupe g. La racine de cet arbre est appelé le point de rendez-vous. L'arbre construit est souvent un arbre des plus courts chemins inverses dont la racine est le point de rendez-vous. Sur cet arbre, l'acheminement est unidirectionnel. Les sources envoient leurs messages en *unicast* au point de rendez-vous qui les retransmet en *multicast* aux membres.

En recevant un message `join` pour un groupe g, un routeur qui n'est pas sur l'arbre *multicast* de ce groupe construit un état dans sa table d'acheminement puis propage le message vers le point de rendez-vous. Lorsqu'un routeur étant déjà sur l'arbre reçoit ce message `join`, il met à jour sa table d'acheminement mais ne propage pas le message. Les

6. C'est de cette particularité que le protocole PIM-SM tire son nom.

messages `join` sont envoyés de manière périodique pour maintenir les entrées. Le départ d'un membre se fait en cessant l'envoi des messages `join`.

Lorsque le débit d'une source devient très important ou si un membre en fait la demande explicite, les rattachements à la source se font par des plus courts chemins (ne passant pas obligatoirement par le point de rendez-vous).

Le protocole PIM-SM est illustré par la figure 1.5. Sur la figure 1.5(a), les hôtes h_1 et h_2 envoient un message de rattachement au point de rendez-vous rp du groupe. L'arbre *multicast* résultant est présenté en traits épais sur la figure 1.5(b). Sur la figure 1.5(c), l'hôte h_1 informe le point de rendez-vous rp qu'il désire se connecter par un plus court chemin à la source (message 1) et envoie un message de notification vers la source s (message 2). Les routeurs construisent la branche de l'arbre par laquelle h_1 va être rattachée en acheminant ce second message. L'arbre résultant est présenté sur la figure 1.5(d). Notons que l'envoi des données de la source peut être dupliqué. Les entrées de routage correspondant à cette branche étant du type (s, g), il n'y a pas de conflit avec d'éventuelles entrées $(*, g)$ pour d'autres hôtes.

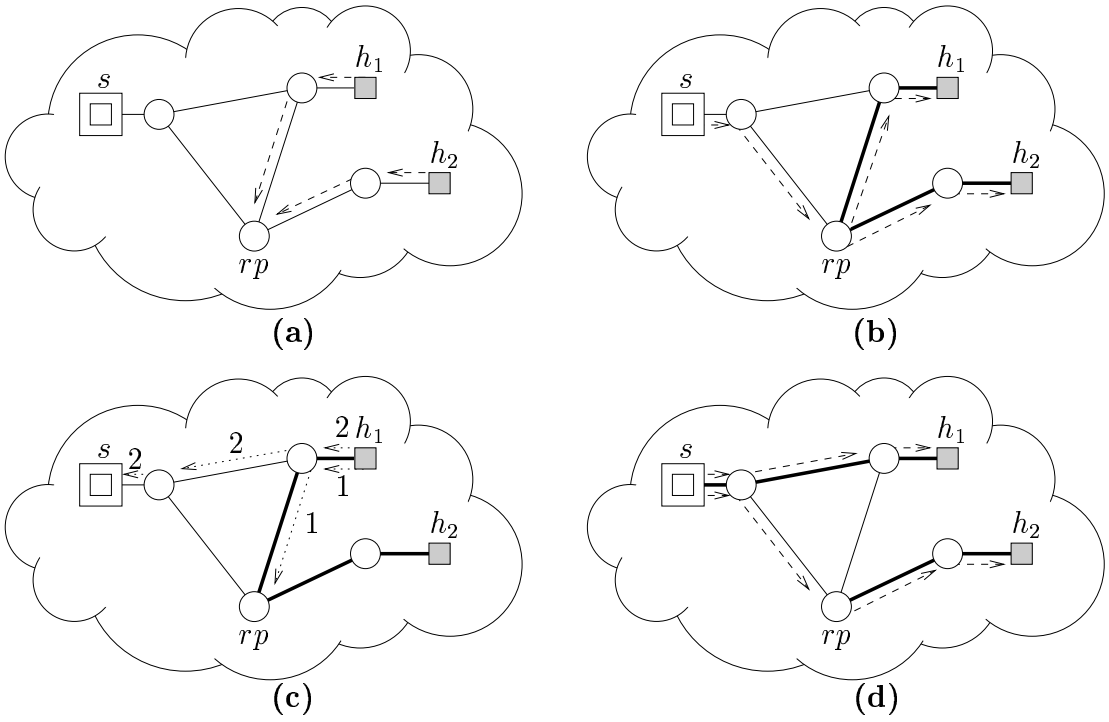


Figure 1.5 – *Le protocole PIM-SM.*

Le choix du point de rendez-vous est important pour évaluer les performances du protocole PIM-SM. Actuellement, il existe dans chaque domaine un routeur appelé BSR (pour *Bootstrap Router*) qui collecte périodiquement les adresses des routeurs candidats pour être point de rendez-vous, et les diffuse dans tout le domaine. Les candidats pour être point de rendez-vous envoient régulièrement des messages de notification au BSR. C'est le BSR qui a la charge d'associer les adresses de groupes aux points de rendez-vous et de diffuser ces associations. Le choix du point de rendez-vous peut se faire à partir de la connaissance

des sources du groupe (puisqu'elles doivent s'enregistrer), mais pas à partir de la connaissance des membres (puisqu'ils ne sont pas encore connus). Le choix optimal d'un point de rendez-vous dans PIM-SM est un problème encore ouvert. Il existe toutefois des mécanismes permettant de changer dynamiquement de point de rendez-vous.

Le protocole PIM-SSM. Le protocole PIM-SSM (pour Protocol Independent Multicast - Source Specific Multicast) est la modification du protocole PIM-SM vis-à-vis des spécifications de [RFC 3569]. Il est très similaire au protocole PIM-SM.

Les deux différences principales entre PIM-SSM et PIM-SM sont les suivantes :

- les adresses des groupes PIM-SSM sont restreintes à la plage `232.0.0.0/8` en IPv4 et à la plage `FF3x::/96` en IPv6.
- PIM-SSM ne crée que des arbres basés à la source et il n'y a donc pas de routeurs de rendez-vous.

Le protocole CBT. Le protocole CBT (pour *Core Based Trees*) est très similaire au protocole PIM-SM. Il est décrit dans [RFC 2189] et l'architecture qu'il nécessite dans [RFC 2201].

Le protocole CBT construit un arbre enraciné en un routeur noyau. Le routeur noyau est similaire au point de rendez-vous du protocole PIM-SM. L'arbre construit est partagé par toutes les sources du groupe. À la différence du protocole PIM-SM, le routage sur cet arbre est bidirectionnel. Le protocole CBT est lui aussi indépendant du protocole *unicast* sous-jacent.

Le protocole CBT (tout comme le protocole PIM-SM) concentre le trafic sur les liens de l'arbre partagé, contrairement aux protocoles qui construisent un arbre par source. Un autre désavantage du protocole CBT est le choix difficile du routeur noyau. Notons enfin que le protocole CBT ne permet pas de basculer des branches de l'arbre en plus courts chemins depuis la source.

Le routage inter-domaines

Plusieurs propositions concernent le routage *multicast* inter-domaines. On peut citer la combinaison PIM-SM/MBGP/MSDP pour le court terme, la combinaison MASC/BGMP pour le moyen terme ou encore l'architecture RAMA.

La combinaison PIM-SM/MBGP/MSDP. La combinaison PIM-SM/MBGP/MSDP utilise les trois protocoles PIM-SM, MBGP et MSDP. Le protocole MBGP (pour *Multiprotocol Extensions for BGP-4*) est décrit dans [RFC 2858]. Le protocole MSDP (pour *Multicast Source Discovery Protocol*) est décrit dans [RFC 3618].

L'architecture PIM-SM/MBGP/MSDP est représentée sur la figure 1.6. Quatre intra-domaines sont présentés, utilisant tous PIM-SM pour le routage *multicast* intra-domaine. Le protocole MBGP permet aux routeurs RP (pour *Rendez-vous Point*, c'est-à-dire les routeurs points de rendez-vous) d'obtenir une vision de la topologie *multicast* inter-domaines, représentée par les liens de *peering* entre routeurs RP. Cette topologie *multicast* inter-domaines n'est pas nécessairement congruente avec la topologie *unicast* inter-domaines, représentée par les liens inter-domaines. Des différences entre ces deux visions de la topologie peuvent

apparaître à cause de politiques restrictives de certains domaines. Dans l'exemple représenté, il n'y a pas de lien *multicast* entre les domaines 2 et 3, alors qu'il y a un lien inter-domaines *unicast* entre eux.

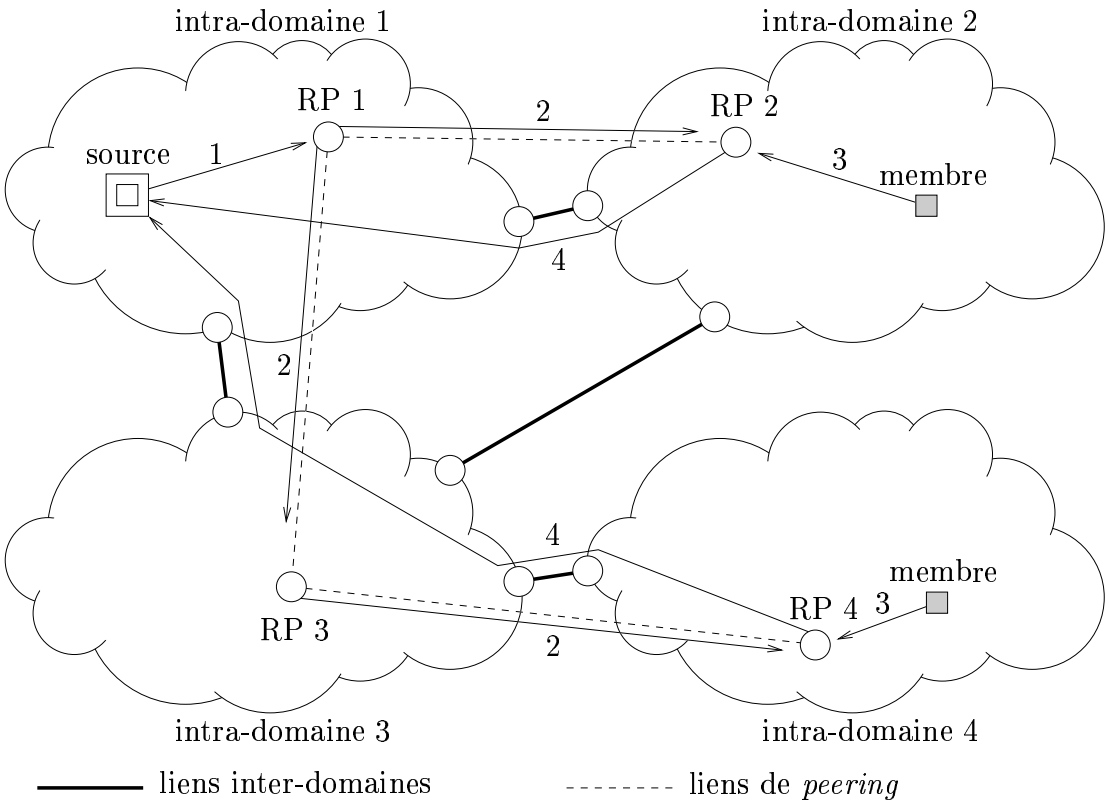


Figure 1.6 – *L'architecture PIM-SM/MBGP/MSDP.*

Lorsque la source d'un groupe émet des messages, ils sont acheminés jusqu'au point de rendez-vous du domaine, RP 1 sur la figure (message 1). Ce point de rendez-vous prévient par un message MSDP `source-active` (message 2) tous les autres point de rendez-vous, en utilisant la topologie *multicast*. Ce message permet aux points de rendez-vous des autres domaines de connaître l'adresse de la source associée à une adresse de groupe donnée.

Quand un hôte désire joindre le groupe *multicast*, il envoie un message PIM-SM `join` (message 3) à son point de rendez-vous (sur la figure, RP 2 ou RP 4). Ce point de rendez-vous achemine alors le message `join` (message 4) vers la véritable source du groupe (connue par MSDP). L'acheminement du message suit le mécanisme du protocole PIM-SM.

En résumé, le protocole MBGP permet de relier les points de rendez-vous des intra-domaine par des liens de *peering*. Le protocole MSDP sert à prévenir les points de rendez-vous lorsqu'une source est active, en utilisant les liens de *peering*. Finalement, PIM-SM est utilisé pour construire chaque sous-arbre intra-domaine et le sous-arbre inter-domaines.

Le problème principal de cette approche est la maintenance d'états inter-domaines dans tous les points de rendez-vous. Dans la figure, RP 3 maintient des états alors que l'intra-domaine 3 ne contient ni sources ni membres pour le groupe.

La combinaison MASC/BGMP. La combinaison MASC/BGMP utilise le protocole BGMP (pour *Border Gateway Multicast Protocol*), décrit dans [RFC 3913], l'architecture MALLOC (pour *Multicast Address Allocation*), décrite dans [RFC 2908], dont le protocole MASC (pour *Multicast Address-Set Claim*) fait partie, et le protocole MBGP. Le protocole MASC est décrit dans [RFC 2909].

L'architecture MALLOC est composée de trois protocoles : un protocole d'adressage inter-domaines (en général le protocole MASC), un protocole d'adressage intra-domaine (en général le protocole AAP, pour *Address Allocation Protocol*, décrit dans [HH00]) et un protocole client (en général MADCAP). L'architecture générale est présentée sur la figure 1.7.

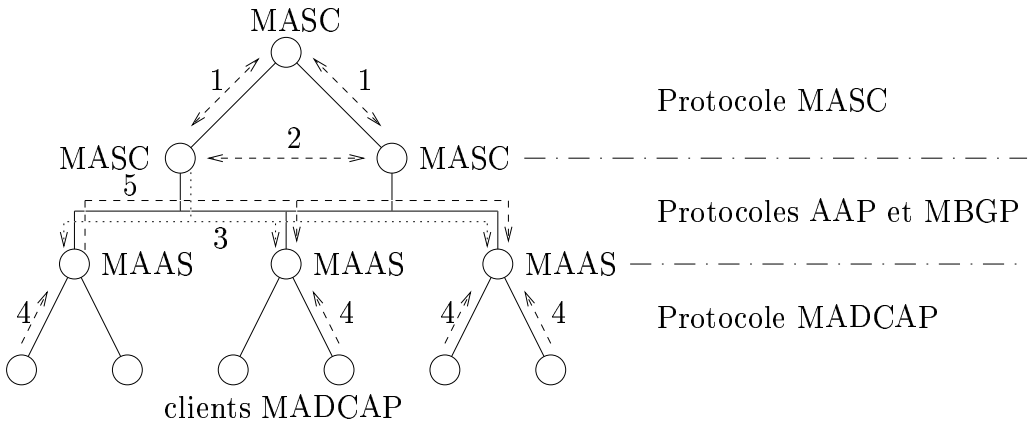


Figure 1.7 – *L'architecture MALLOC.*

Le protocole MASC régit l'espace d'adressage *multicast* au moyen d'une architecture décentralisée, organisée sous forme de forêt. Les routeurs MASC ont tous un père, à l'exception des routeurs MASC de haut niveau. La distribution de plages d'adresses est effectuée au moyen d'un mécanisme « d'écoute, d'annonce et de détection de collisions » : les routeurs MASC fils écoutent les plages d'adresses de leurs parents (message 1), annoncent ces plages à leurs frères dans l'arbre (message 2), et attendent pour s'assurer qu'il n'y a pas de collisions.

Le protocole AAP est utilisé pour transmettre en *multicast* ces plages d'adresses d'un routeur MASC aux routeurs MAAS étant dans le même domaine (message 3). Le protocole MBGP permet de transmettre ces plages d'adresses aux routeurs MAAS qui ne sont pas dans le même domaine que le routeur MASC (également message 3). Les serveurs MAAS ont pour rôle de réduire la probabilité de collisions d'adresses.

Finalement, les clients MADCAP interrogent les serveurs MAAS pour leur demander dynamiquement des adresses. Lorsqu'un serveur MAAS reçoit une requête pour une adresse *multicast* (message 4), il annonce à tous les autres routeurs MAAS qu'il envisage d'utiliser une adresse donnée (message 5). S'il ne reçoit aucune notification de collision au bout d'un certain temps, il attribue l'adresse au groupe en la retournant au client MADCAP. Lorsqu'un serveur MAAS est proche d'épuiser sa plage d'adresses, il prévient un serveur MASC.

Le protocole BGMP permet à un routeur de bordure de domaine de construire un arbre inter-domaines partagé. Le point de rendez-vous de l'arbre partagé BGMP n'est pas un routeur, comme dans CBT, mais un domaine. C'est le rôle du protocole BGMP de décider du domaine qui agira comme point de rendez-vous. Pour cela, le protocole BGMP a besoin

de connaître le domaine qui est responsable de l'adresse du groupe, et c'est en cela qu'il est nécessairement couplé au protocole MASC. L'arbre construit est bidirectionnel.

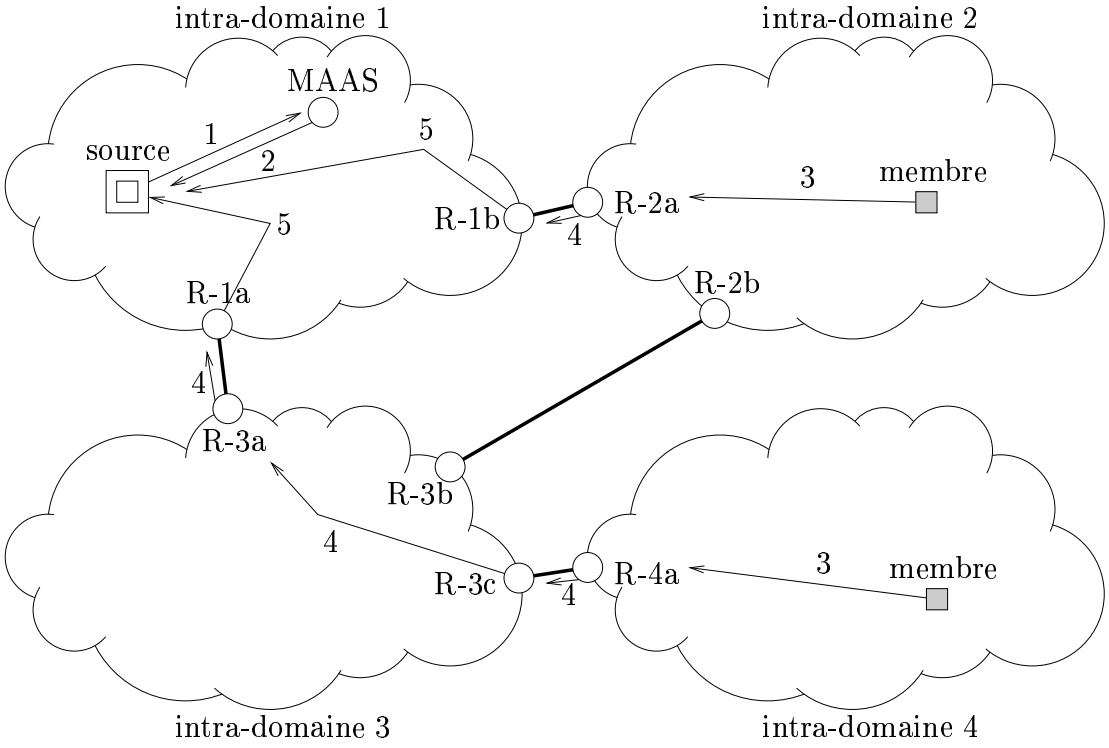


Figure 1.8 – *Le protocole BGMP.*

La figure 1.8 présente la manière dont les routeurs BGMP construisent l'arbre partagé. L'adresse du groupe g acquise par la source (messages 1 et 2) est une adresse du domaine 1. Cela signifie que le point de rendez-vous BGMP pour ce groupe est le domaine 1. Quand un membre du domaine 4 décide de joindre le groupe g, il envoie un message d'adhésion IGMP. Le routeur désigné du membre prévient un routeur de sortie BGMP du domaine, R-4a dans notre exemple, au moyen d'un protocole de routage *multicast* intra-domaine (message 3). Selon R-4a, le prochain routeur BGMP vers le domaine 1 étant R-3c, R-4a crée l'entrée $(*, g)$ correspondant à R-3c (message 4). Rappelons que l'information sur la topologie est obtenue par MBGP. À son tour, R-3c crée une entrée $(*, g)$ pour le prochain routeur BGMP vers le domaine 1, R-3a (message 4). Similairement, R-3a crée une entrée $(*, g)$ pour R-1a (message 4) et R-1a s'enregistre au protocole intra-domaine (message 5).

La combinaison MASC/BGMP souffre de la complexité de son déploiement (car beaucoup de protocoles sont en jeu) et du problème de passage à l'échelle des annonces des adresses *multicast*.

L'architecture RAMA. L'architecture RAMA (pour *Root Addressed Multicast Architecture*) est décrite initialement dans [BCD+99] et dans [Alm99]. Elle est beaucoup plus simple que les combinaisons PIM-SM/MBGP/MSDP et MASC/BGMP et permet de s'intéresser à quelques problèmes de sécurité du *multicast*. Cette architecture tire profit de la

présence fréquente d'une source unique ou clairement identifiable dans les les groupes *multicast*. Cependant, l'architecture est toujours à l'étude. Les deux protocoles de référence de l'architecture RAMA sont les protocoles EXPRESS et SM.

Le protocole EXPRESS (pour *Explicitely Requested Single-Source*) est décrit dans [HC99]. Le protocole EXPRESS est souvent associé au protocole ECMP (pour *EXPRESS Count Message Protocol*) qui permet d'avoir des informations sur le groupe, notamment sur le nombre de membres. Dans le protocole EXPRESS, chaque groupe est identifié par le couple (s, id) où s est la source du groupe et id un canal choisi par s. L'arbre construit par le protocole EXPRESS est un arbre unidirectionnel des plus courts chemins inverses, basé en s. Grâce à la notion de canal introduite par EXPRESS, le contrôle d'accès peut être fait à la source, ce qui permet notamment de facturer les communications (par le biais du protocole ECMP par exemple). De plus, EXPRESS peut fournir des groupes fermés (à l'opposé du modèle ouvert habituel, où tous les hôtes peuvent joindre un groupe quelconque).

Le protocole SM (pour *Simple Multicast*) est décrit dans [PLB+99]. Dans ce protocole, chaque groupe est identifié par le couple (c, g), où c est un routeur noyau. Le routeur c correspond au routeur désigné de la source primaire du groupe. À la différence du protocole EXPRESS, l'arbre construit par le protocole SM est un arbre bidirectionnel.

1.3 Les limites du modèle

Le modèle de Deering est basé sur les spécificités des réseaux des années 90 et sur les hypothèses de l'époque concernant le *multicast*. Dans cette partie, nous montrons que le modèle de Deering n'est adapté ni aux réseaux de nouvelle génération, ni à l'explosion prévisible du *multicast*.

1.3.1 Le modèle de Deering et les réseaux de nouvelle génération

Beaucoup de réseaux sont dits de nouvelle génération. La plupart du temps, cette dénomination est commerciale et indique que les débits sont importants. Cependant, certains réseaux apparaissent vraiment comme nouveaux : leurs caractéristiques et leurs fonctionnements sont radicalement changés. On peut mentionner les réseaux sans fils à infrastructure fixe, les réseaux sans fils sans infrastructure fixe ou les réseaux tout optique.

Dans ces réseaux de nouvelle génération, les hypothèses du modèle de Deering ne sont plus valides. Dans le cas des réseaux sans fils ou sans infrastructure fixe, le cadre n'est plus point-à-point. De plus, les protocoles existants (y compris les protocoles *unicast*) ne sont pas adaptés à la mobilité des hôtes ou des routeurs. Cette mobilité induit une dynamique de la topologie qui est très délicate à gérer.

Dans le cas des réseaux tout optique que nous étudierons en détail dans cette thèse, il est nécessaire de changer de paradigme : les communications ne sont plus orientées paquets (comme c'est le cas pour les réseaux IP) mais orientés connexions (comme c'est le cas pour les réseaux MPLS). Le faible traitement des paquets dans les routeurs conduit à proposer de nouvelles méthodes spécifiques à ces réseaux. Les réseaux tout optique étant principalement des réseaux de cœur, il est probable que la dynamique des groupes *multicast* à l'intérieur de

ces réseaux soit limitée. Nous avons choisi de ne pas nous concentrer sur ce point dans cette thèse.

Le modèle *multicast* doit être modifié afin de tirer profit des nouvelles caractéristiques.

1.3.2 Le modèle de Deering et le passage à l'échelle du *multicast*

L'explosion de l'utilisation d'Internet n'était pas prévisible à l'époque où sa standardisation a été faite. Le protocole IP a bien supporté le passage à l'échelle, grâce à de nombreux facteurs, dont une allocation judicieuse des adresses *unicast*.

Contrairement à l'*unicast*, le *multicast* actuel n'est pas en mesure de supporter un passage à l'échelle. Les routeurs de cœur de réseau devraient en effet stocker les informations nécessaires à la gestion de tous les groupes *multicast* traités. En plus de ces informations, il faut prendre en compte les messages de contrôles nécessaires à leur maintenance et à leur mise à jour. La difficulté s'accentue lorsque l'on intègre les modifications dynamiques de l'ensemble des destinataires des membres. Il est donc important de trouver un nouveau mécanisme permettant au *multicast* de gérer l'augmentation du nombre de groupes ou de la taille des groupes.

1.4 Plan de la thèse

Dans cette thèse, nous nous attaquons principalement à deux aspects du modèle de Deering qui limitent le développement du *multicast* : son inadaptabilité aux réseaux de nouvelle génération tout optique et le passage à l'échelle en nombre de groupes.

Tout d'abord, nous verrons dans le chapitre 2 en quoi les hypothèses du modèle de Deering ne s'appliquent pas aux réseaux tout optique. Nous nous concentrons sur le cas des réseaux tout optique à routage en longueurs d'onde et nous montrons que le modèle de Deering n'est pas performant. Nous proposons alors des solutions performantes dans ce contexte.

Puis, nous considérons le passage à l'échelle des groupes. Il est probable que le nombre de groupes concurrents devienne un goulot d'étranglement pour le *multicast*, puisque le nombre d'entrées *multicast* et d'arbres *multicast* à maintenir croissent proportionnellement avec le nombre de groupes. En étudiant d'une part les petits groupes dans le chapitre 3 et d'autre part les groupes de taille quelconque dans le chapitre 4, nous apportons des techniques efficaces pour améliorer le passage à l'échelle du *multicast*.

Références

[Alm99] K. Almeroth. « The Evolution of Multicast: From the Mbone to Inter-Domain Multicast to Internet2 Deployment ». *IEEE Network*, septembre 1999.

[BCD+99] T. Ballardie, J. Crowcroft, C. Diot, C.-Y. Lee, R. Perlman, et Z. Wang. « On extending the standard IP multicast architecture ». Research Report RN/99/21, University College London, octobre 1999.

[Dee91] S. E. Deering. « *Multicast Routing in a Datagram Internetwork* ». Thèse de doctorat, Stanford University, décembre 1991.

[HC99] H. W. Holbrook et D. R. Cheriton. « IP Multicast Channels: EXPRESS Support for Large-scale Single-source Applications ». Dans *ACM SIGCOMM*, pages 65–78, septembre 1999.

[HH00] M. Handley et S. Hanna. « Multicast Address Allocation Protocol (AAP) ». draft draft-ietf-malloc-aap-03.txt, IETF, juin 2000.

[Hak71] S. L. Hakimi. « Steiner's problem in graphs and its implications ». *Networks*, 1:113–133, 1971.

[KMB81] L. Kou, G. Markowsky, et L. Berman. « A fast algorithm for Steiner trees in graphs ». *Acta Informatica*, 15:141–145, 1981.

[Kar72] R. M. Karp. « Reducibility Among Combinatorial Problems ». Dans *Complexity of Computer Computations*, pages 85–103. Plenum Press, 1972.

[Kru56] J. Kruskal. « On the Shortest Spanning Tree of a Graph and the Traveling Salesman Problem ». *American Mathematical Society*, 7(1):48–50, février 1956.

[PLB+99] R. Perlman, C. Lee, A. Ballardie, J. Crowcroft, Z. Wang, T. Maufer, C. Diot, J. Thoo, et M. Green. « Simple Multicast: A Design for Simple, Low-Overhead Multicast ». Draft draft-perlman-simple-multicast-03.txt, IETF, octobre 1999.

[Pri57] R. C. Prim. « Shortest connection networks and some generalizations ». *The Bell Systems Technical Journal*, 36(11):1389–1401, novembre 1957.

[RFC 1075] D. Waitzman, C. Partridge, et S. E. Deering. « Distance Vector Multicast Routing Protocol ». RFC 1075, IETF, novembre 1988.

[RFC 1112] S. E. Deering. « Host Extensions for IP Multicasting ». RFC 1112, IETF, août 1989.

[RFC 1584] J. Moy. « Multicast Extensions to OSPF ». RFC 1584, IETF, mars 1994.

[RFC 2131] R. Droms. « Dynamic Host Configuration Protocol ». RFC 2131, IETF, mars 1997.

[RFC 2189] A. Ballardie. « Core Based Trees (CBT version 2) Multicast Routing – Protocol Specification ». RFC 2189, IETF, septembre 1997.

[RFC 2201] A. Ballardie. « Core Based Trees (CBT) Multicast Routing Architecture ». RFC 2201, IETF, septembre 1997.

[RFC 2236] W. Fenner. « Internet Group Management Protocol, Version 2 ». RFC 2236, IETF, novembre 1997.

[RFC 2328] J. Moy. « OSPF Version 2 ». RFC 2328, IETF, avril 1998.

[RFC 2362] D. Estrin, D. Farinacci, A. Helmy, D. Thaler, S. E. Deering, M. Handley, V. Jacobson, C. Liu, P. Sharma, et L. Wei. « Protocol Independent Multicast-Sparse Mode (PIM-SM): Protocol Specification ». RFC 2362, IETF, juin 1998.

[RFC 2453] G. Malkin. « RIP Version 2 ». RFC 2453, IETF, novembre 1998.

[RFC 2730] S. Hanna, B. Patel, et M. Shah. « Multicast Address Dynamic Client Allocation Protocol (MADCAP) ». RFC 2730, IETF, décembre 1999.

[RFC 2858] T. Bates, Y. Rekhter, R. Chandra, et D. Katz. « Multiprotocol Extensions for BGP-4 ». RFC 2858, IETF, juin 2000.

[RFC 2908] D. Thaler, M. Handley, et D. Estrin. « The Internet Multicast Address Allocation Architecture ». RFC 2908, IETF, septembre 2000.

[RFC 2909] P. Radoslavov, D. Estrin, R. Govindan, M. Handley, S. Kumar, et D. Thaler. « The Multicast Address-Set Claim (MASC) Protocol ». RFC 2909, IETF, septembre 2000.

[RFC 3180] D. Meyer et P. Lothberg. « GLOP Addressing in 233/8 ». RFC 3180, IETF, septembre 2001.

[RFC 3306] B. Haberman et D. Thaler. « Unicast-Prefix-based IPv6 Multicast Addresses ». RFC 3306, IETF, août 2002.

[RFC 3376] B. Cain, S. E. Deering, I. Kouvelas, B. Fenner, et A. Thyagarajan. « Internet Group Management Protocol, Version 3 ». RFC 3376, IETF, octobre 2002.

[RFC 3513] R. Hinden et S. Deering. « Internet Protocol Version 6 (IPv6) Addressing Architecture ». RFC 3513, IETF, avril 2003.

[RFC 3569] S. Bhattacharyya. « An Overview of Source-Specific Multicast (SSM) ». RFC 3569, IETF, juillet 2003.

[RFC 3618] B. Fenner et D. Meyer. « Multicast Source Discovery Protocol (MSDP) ». RFC 3618, IETF, octobre 2003.

[RFC 3913] D. Thaler. « Border Gateway Multicast Protocol (BGMP): Protocol Specification ». RFC 3913, IETF, septembre 2004.

[Ram00] M. Ramalho. « Intra- and Inter-Domain Multicast Routing Protocols: A Survey and Taxonomy ». *IEEE Communications Surveys & Tutorials*, 3(1), 2000.

[TM80] H. Takahashi et A. Matsuyama. « An approximate solution for the Steiner problem in graphs ». *Mathematica Japonica*, 24(6):573–577, 1980.

[Win87] P. Winter. « Steiner Problem in Networks: A Survey ». *Networks*, 17:129–167, 1987.

Chapitre 2

Réseaux à routage en longueur d'onde

L'utilisation de fibres optiques à la place de câbles en cuivre a permis une importante augmentation de la capacité des réseaux, notamment grâce à la technique de multiplexage WDM (pour *Wavelength Dense Multiplexing*) décrite plus loin dans ce chapitre. Cependant, la capacité des réseaux actuels est limitée par le traitement des routeurs électroniques. Pour gérer une capacité de plusieurs Tbps (pour *Terabits par secondes*), ce qui est théoriquement possible en utilisant WDM, il est nécessaire d'utiliser des routeurs entièrement optiques. De récentes avancées dans la technologie optique permettent la réalisation de tels routeurs tout optique basés sur le routage en longueur d'onde et étant capables de gérer de tels volumes d'information. À moyen terme, ces routeurs tout optique vont certainement remplacer la plupart des routeurs électroniques dans les réseaux à haut débit.

Comme nous l'avons dit en introduction, le modèle de Deering [Dee91] est basé sur la norme OSI à sept couches. Selon cette norme, le protocole IP se trouvant dans la couche réseau doit interagir avec un protocole de la couche liaison de données, souvent SONET/SDH (pour *Synchronous Optical Networks / Synchronous Digital Hierarchy*, voir dans l'annexe C de [SRMG01]), qui à son tour interagit avec un protocole de la couche physique auquel WDM appartient. Cet empilement est souvent appelé IP sur SONET/SDH sur WDM. Il est présenté sur la figure 2.1(a). Cependant, l'utilisation d'IP sur SONET/SDH sur WDM ne permet pas d'utiliser au mieux les fonctionnalités des routeurs tout optique. Une approche de plus en plus utilisée consiste à supprimer la couche liaison de données en utilisant directement IP sur WDM [Liu02, Dix03], comme présenté sur la figure 2.1(b). En effet, les services apportés par la couche liaison de données, à savoir le contrôle des flux de bits, la synchronisation des trames et le contrôle d'erreur, sont inutiles ou redondants : WDM contrôle les flux, ne nécessite pas de synchronisation et IP contrôle les erreurs au moyen d'une somme de contrôle. En outre, les avantages de l'utilisation d'IP sur WDM sont nombreux :

– la charge d'acheminement qui incombe au protocole IP est diminuée (car le trafic n'a

pas à être découpé en tranches de 10 Gbps (pour *Gigabits par secondes*) comme le fait SONET/SDH),
- la suppression de la perte de bande passante due au découpage temporel strict de SONET/SDH,
- la gestion de trafics de plusieurs Tbps,
- la réduction (voire la suppression) du besoin de conversions du signal optique pour un traitement électronique,
- la possibilité de réaliser une optimisation multi-couches, au détriment de la transparence.

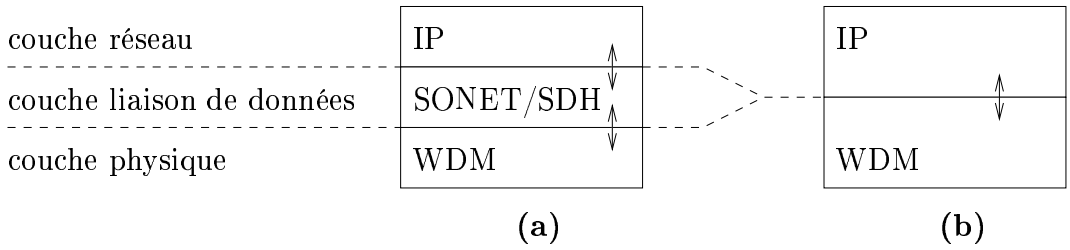


Figure 2.1 – *La pile IP sur SONET/SDH sur WDM et la pile IP sur WDM.*

Toutefois, l'utilisation d'IP sur WDM nécessite de prendre en compte l'hétérogénéité des fonctionnalités des routeurs optiques. Il faut donc que les algorithmes de routage intègrent les particularités physiques des routeurs.

Dans le début de ce chapitre, nous présentons la technologie optique, certaines de ses contraintes et les problèmes qui se posent pour le *multicast*. Puis, nous comparons à l'état de l'art des algorithmes de routage *multicast* que nous avons conçu pour gérer ces contraintes. Finalement, nous étudions des problèmes théoriques liés au routage tout optique.

2.1 Les réseaux tout optique

Les avancées récentes dans la technologie optique ont abouti à la réalisation d'équipements optiques dont les performances dépassent largement celles des équipements électroniques. Dans ce paragraphe, nous décrivons brièvement la technologie optique et l'architecture sur laquelle nous nous basons. Nous présentons ensuite les avantages des réseaux tout optique sur les réseaux électroniques. Enfin, nous étudions les contraintes et la modélisation des réseaux tout optique qui nous servira dans la suite du chapitre.

2.1.1 La technologie optique

La technologie optique a rapidement progressé ces dernière années, passant de la conception de fibres optiques plus performantes à l'élaboration de routeurs tout optique. Dans ce paragraphe, nous présentons ces deux aspects en nous appuyant sur le livre de Siva Ram Murthy et Gurusamy [SRMG01].

Les fibres optiques

Les fibres optiques sont généralement basées sur le phénomène physique de réflexion totale. À la frontière entre deux milieux, un rayon lumineux incident subit une réflexion partielle ou une réflexion totale (dans des cas limites). Lorsque l'indice n_{int} du milieu interne est inférieur à l'indice n_{ext} du milieu externe, le rayon subit toujours une réflexion partielle. Une partie du rayon est réfléchie et reste dans le milieu interne, tandis qu'une autre partie du rayon lumineux (appelée le rayon réfracté) sort du milieu interne, ce qui réduit d'autant l'intensité du rayon réfléchi.

Lorsque l'indice n_{int} du milieu interne est supérieur à l'indice n_{ext} du milieu externe, le rayon lumineux subit une réflexion totale si son angle d'incidence i est plus grand qu'un angle critique i_c. Dans le cas contraire, le rayon subit une réflexion partielle. L'angle critique i_c est défini par :

$$i_c = \sin^{-1} \frac{n_{\text{ext}}}{n_{\text{int}}}.$$

La figure 2.2 présente la réflexion partielle d'un rayon lumineux dans le cas où $n_{\text{int}} \geq n_{\text{ext}}$ mais l'angle du rayon incident est plus petit que l'angle critique i_c.

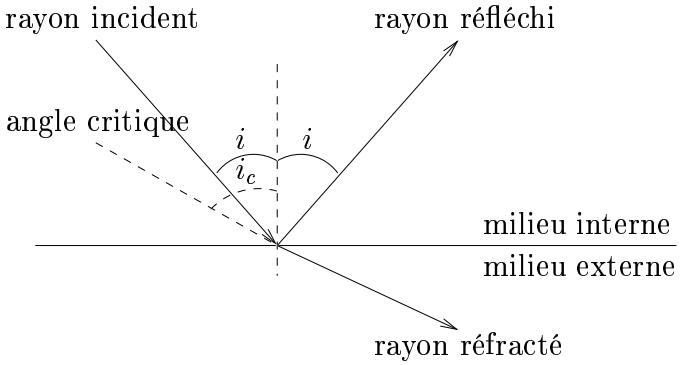


Figure 2.2 – *La réflexion partielle (cas où $n_{int} \geq n_{ext}$ et $i < i_c$).*

La figure 2.3 présente la réflexion totale d'un rayon lumineux. Dans ce cas, l'intensité du rayon réfléchi est égale à celle du rayon incident. Le rayon lumineux est ainsi capturé dans le milieu interne. Ce principe de propagation cloisonnée dans un milieu sans perte d'énergie (dans le cas idéal) est le principe de base d'une fibre optique.

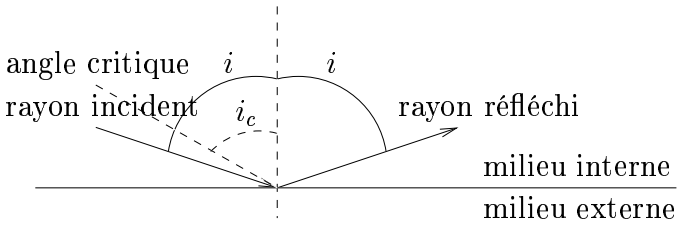


Figure 2.3 – *La réflexion totale (cas où $n_{int} \geq n_{ext}$ et $i \geq i_c$).*

Comme le montre la figure 2.4, une fibre optique est composée de trois parties :

– un verre de cœur d'indice de réfraction absolu n_{int},

– un verre de revêtement d'indice de réfraction absolu n_{ext} légèrement inférieur à n_{int},
– une gaine isolante.

La faible différence entre les indices n_{int} et n_{ext} garantit que le rapport $n_{\text{ext}}/n_{\text{int}}$ va être proche de 1 et donc que l'angle critique i_c va être relativement petit. Ainsi, des rayons d'un large spectre d'incidence subiront une réflexion totale.

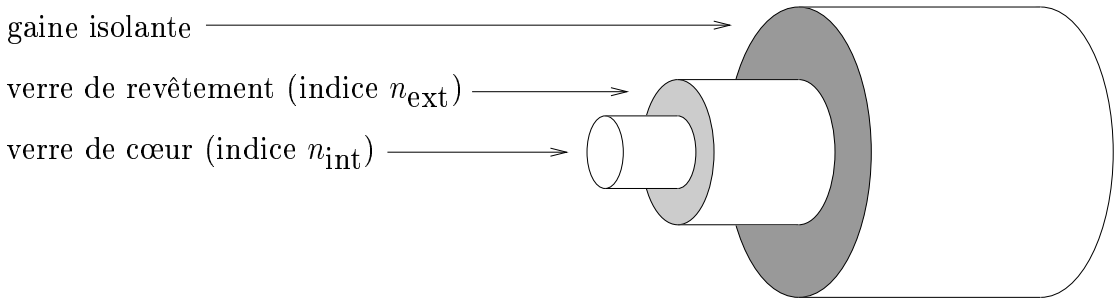


Figure 2.4 – *Coupe d'une fibre optique.*

Certaines fibres optiques permettent l'émission simultanée de plusieurs rayons lumineux d'angles d'incidence différents. Ces fibres sont appelées fibres multimodes. Ce procédé de multiplexage provoque cependant des interférences entre les rayons. Un autre procédé de multiplexage est le WDM, qui est un multiplexage de longueur d'ondes (nous l'avons déjà mentionné en introduction). Plusieurs rayons lumineux de longueurs d'onde différentes sont émis simultanément. En raisons de phénomènes d'interférences complexes, le nombre de longueurs d'onde multiplexables dans une fibre optique est limité. Le tableau 2.1 indique (entre autres) l'ordre de grandeur du nombre de longueurs d'onde multiplexables dans les fibres actuelles.

Débit	Nombre de longueur d'ondes	Modulation	Distance	Taux d'erreur	Référence
3 Tbps	300	11.6 Gbps	7380 km	inconnu	[VPM01]
3.2 Tbps	80	42.7 Gbps	5200 km	inconnu	[ZLG+02]
6.4 Tbps	160	42.7 Gbps	100 km	inconnu	[ZNS+03]
10.2 Tbps	256	42.7 Gbps	100 km	10^{-15}	Alcatel [BFC+01]
10.92 Tbps	273	40 Gbps	117 km	10^{-9}	NEC [FKM+01]

Tab. 2.1 – *Quelques caractéristiques des fibres optiques actuelles.*

Sur le tableau 2.1, nous pouvons voir aussi que la distance parcourue par les signaux dans une fibre optique est limitée. En effet, un rayon lumineux est atténué à mesure qu'il se propage dans une fibre optique. Cette atténuation est due à la dispersion de Rayleigh, qui provient de la présence d'impuretés dans le verre de cœur. C'est pourquoi les signaux transitant dans une fibre optique ont besoin d'être ré-amplifiés régulièrement. Cette ré-amplification est appelée la régénération 1R. L'utilisation d'amplificateurs introduit un compromis : le signal émis mais aussi le bruit sont amplifiés simultanément.

En outre, la dispersion de Rayleigh déforme les rayons lumineux. La figure 2.5 présente la forme des signaux optiques à leur émission et à leur réception. La remodélisation des signaux est nécessaire régulièrement. La combinaison de la remodélisation et de la ré-amplification des signaux est appelée la régénération 2R.

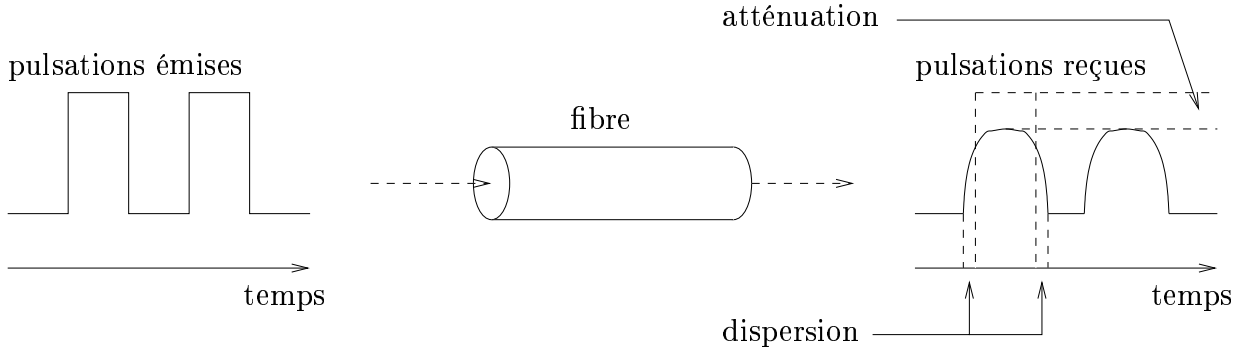


Figure 2.5 – *Les phénomènes d'atténuation et de dispersion déforment les signaux transmis.*

Finalement, deux signaux émis simultanément peuvent arriver l'un après l'autre avec un délai significatif, car la vitesse de propagation d'un rayon lumineux dépend à la fois de sa longueur d'onde et de son angle d'incidence. Pour palier à ce problème, il faut re-synchroniser régulièrement les signaux. La combinaison de la re-synchronisation, de la remodélisation et de la ré-amplification est appelée la régénération 3R.

Un équipement réalisant la régénération 1R est appelé un amplificateur. Un équipement réalisant la régénération 2R ou 3R est appelé un répéteur. Pour pouvoir régénérer correctement le signal, la distance entre amplificateurs ou répéteurs est limitée. Le tableau 2.2 indique les distances usuelles entre ces différents équipements de régénération.

Équipements de régénération	Distance maximale
Amplificateurs (1R)	généralement entre 40 km et 50 km, jusqu'à 100 km
Répéteurs 2R	généralement entre 80 km et 120 km, jusqu'à 500 km
Répéteurs 3R	généralement entre 80 km et 120 km, jusqu'à 500 km

TAB. 2.2 – *Distance maximale entre équipements de régénération.*

Les équipements optiques

Après la réalisation de fibres optiques permettant des débits de plusieurs Tbps, l'avancée de la technologie optique a permis la réalisation de routeurs tout optique. Nous décrivons ici les propriétés et contraintes de ces routeurs.

Conception d'un WADM. Techniquement, un WADM (pour *Wavelength Add/Drop Multiplexer*) est un équipement tout optique permettant d'insérer ou de retirer des longueurs d'onde traversant une fibre optique. Un WADM peut être réalisé avec un démultiplexeur, autant de (2×2)-commutateurs que de longueurs d'onde, et un multiplexeur. La figure 2.6

présente un WADM pour deux longueurs d'onde. Le commutateur c_0, dans la position *bar*, laisse passer le signal correspondant à la longueur d'onde λ_0. Le commutateur c_1, dans la position *cross*, retire la longueur d'onde λ_1 et insère un nouveau signal sur cette même longueur d'onde.

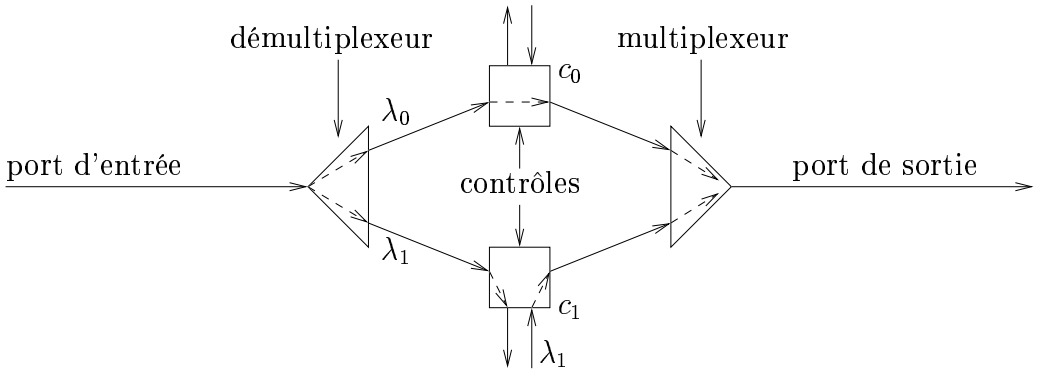


Figure 2.6 – *Schéma de conception d'un WADM.*

Au niveau de l'architecture, les WADM permettent aux routeurs d'injecter ou d'extraire du trafic. Ils sont donc placés en bordure de réseau et sont connectés aux routeurs utilisateurs.

Conception d'un WXC. Techniquement, un WXC (pour *Wavelength Selective Crossconnect*) est un équipement chargé de réaliser la commutation optique, c'est-à-dire d'associer des ports de sortie aux ports d'entrée. La commutation peut être effectuée de deux façons : par commutation de fibres ou par commutation de longueurs d'onde.

Un WXC à commutation de fibres transfère toutes les longueurs d'onde d'un port d'entrée quelconque à un port de sortie quelconque. Un (2×2)-WXC à commutation de fibres est soit dans l'état *bar* s'il associe au port d'entrée 0 le port de sortie 0, et au port d'entrée 1 le port de sortie 1, soit dans l'état *cross* s'il associe au port d'entrée 0 le port de sortie 1, et au port d'entrée 1 le port de sortie 0. Un $(n \times n)$-WXC à commutation de fibres peut être réalisé avec n^2 (2×2)-WXC à commutation de fibres. La figure 2.7 présente un (4×4)-WXC à commutation de fibres. Sur cet exemple, le port d'entrée 0 est associé au port de sortie 2, le port d'entrée 1 est associé au port de sortie 3, le port d'entrée 2 est associé au port de sortie 1 et le port d'entrée 3 est associé au port de sortie 0.

Un WXC à commutation de longueurs d'onde permet de transférer une longueur d'onde quelconque d'un port d'entrée quelconque à un port de sortie quelconque. Un $(n \times n)$-WXC à commutation de longueurs d'onde peut être réalisé avec n démultiplexeurs, autant de $(n \times n)$-commutateurs que de longueurs d'onde, et n multiplexeurs. La figure 2.8 présente un (2×2)-WXC à commutation de longueurs d'onde pour deux longueurs d'onde.

Au niveau de l'architecture, les WXC sont utilisés comme routeurs internes du réseau. Il n'est pas nécessaire que tous les WXC d'un réseau utilisent le même mécanisme de commutation (à fibres ou à longueurs d'onde). Cependant, c'est souvent le cas pour des raisons de gestion. Dans la suite de ce chapitre, nous nous concentrons sur les WXC à commutation de longueurs d'onde, plus souples que les WXC à commutation de fibres. Nous utiliserons

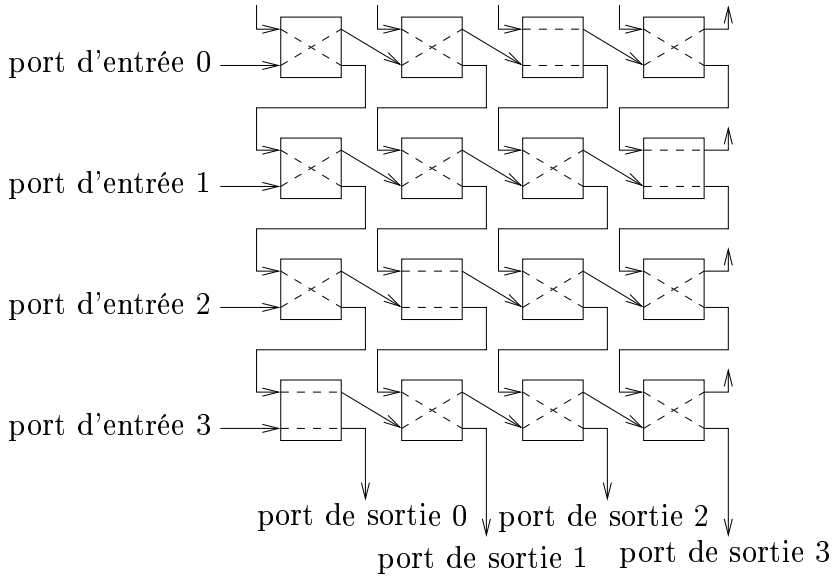


Figure 2.7 – *Schéma de conception d'un WXC à commutation de fibres.*

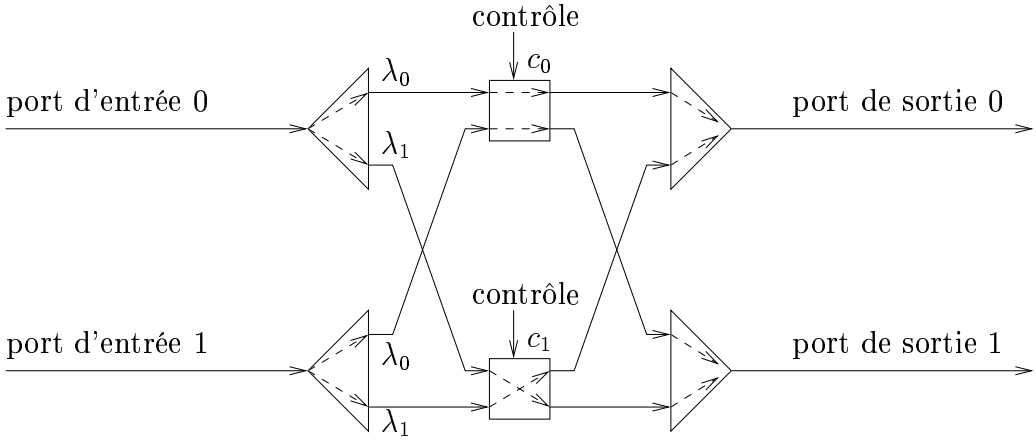


Figure 2.8 – *Schéma de conception d'un WXC à commutation de longueurs d'onde.*

le terme WXC pour désigner un WXC à commutation de longueurs d'onde et le terme de routeur tout optique pour désigner un WXC intégrant les fonctionnalités d'un WADM.

La conversion de longueur d'ondes. Il est possible d'insérer dans l'architecture d'un WXC des convertisseurs de longueurs d'onde paramétrables. Ces convertisseurs permettent de convertir un signal de longueur d'onde λ en un signal de longueur d'onde λ'. Cependant, des limites physiques font que les longueurs d'onde disponibles pour le signal sortant dépendent généralement de la longueur d'onde du signal entrant λ.

La duplication des signaux. Certains routeurs tout optique peuvent dupliquer les signaux. Pour dupliquer un signal d'entrée en au plus f signaux de sortie identiques, ces routeurs nécessitent $f \cdot (1 + NB.\lambda)$ prismes et $f \cdot NB.\lambda$ amplificateurs optiques, où $NB.\lambda$ est le nombre de longueurs d'onde disponibles. Sans ces amplificateurs, les signaux dupliqués auraient une intensité beaucoup plus faible que le signal initial et deviendraient inutilisables. La figure 2.9 présente une architecture réalisant la duplication d'un signal en 2 signaux identiques.

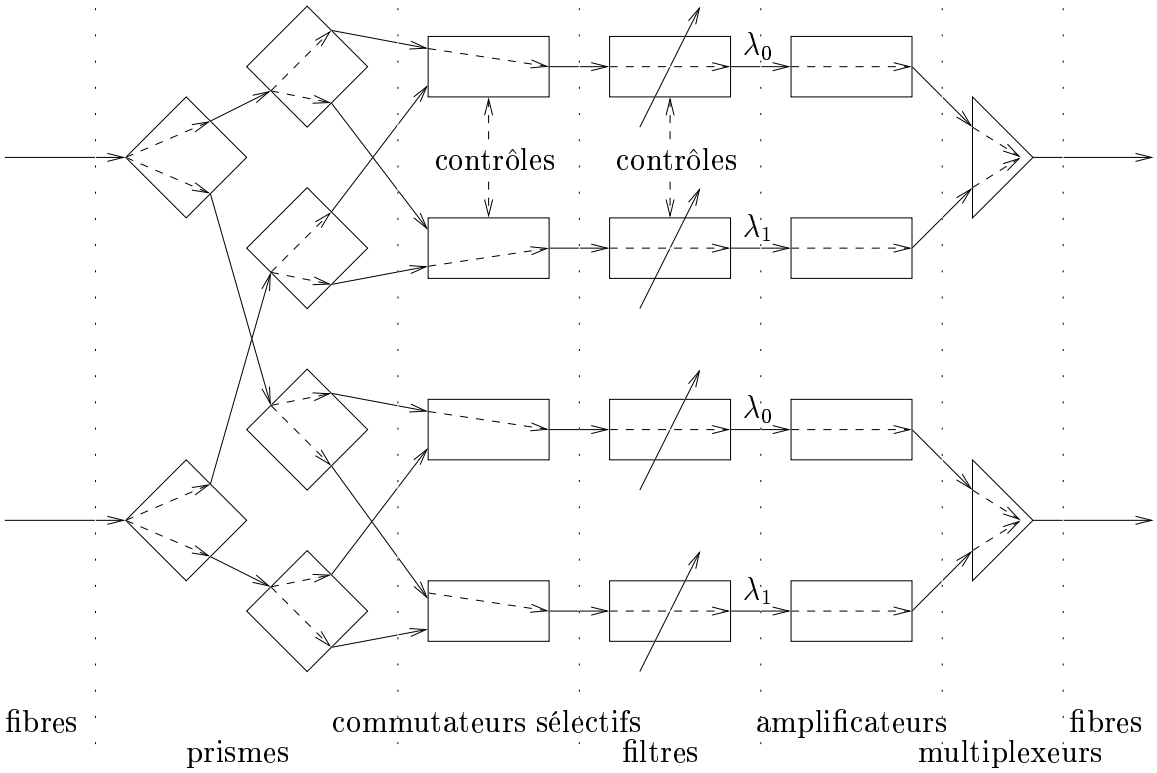


Figure 2.9 – *Schéma d'un WXC pouvant dupliquer chaque signal en* 2 *signaux identiques.*

La présence d'un grand nombre d'amplificateurs optiques augmente le coût des routeurs tout optique pouvant dupliquer les signaux. Par conséquent, il semble que peu de routeurs tout optique seront capables de dupliquer les signaux dans les futurs réseaux tout optiques. De plus, le nombre de signaux de sortie provenant d'un même signal d'entrée est limité à une valeur souvent inférieure au nombre de ports de sortie du routeur.

Un routeur pouvant dupliquer un signal en f signaux de sortie est aussi capable d'en extraire une copie pour lui (en utilisant la fonctionnalité du WADM). L'extraction d'une copie se fait en utilisant un duplicateur optique particulier qui crée une copie du signal de très faible intensité (de l'ordre de 5 %) et une copie du signal de très grande intensité (de l'ordre de 95 %). Le signal de faible intensité est extrait, tandis que le signal de grande intensité est acheminé normalement. Cette technique ne nécessite pas l'utilisation d'amplificateurs. En d'autres termes, nous considérons qu'extraire le signal pour en garder une copie locale ne réduit pas la capacité de duplication d'un routeur.

La conversion O/E/O. Certains WXC sont incapables, au niveau optique, de dupliquer les signaux ou de convertir les longueurs d'onde. Pour dupliquer ou convertir, ces WXC doivent convertir le signal optique en signal électronique, éventuellement le dupliquer au niveau électronique, puis le reconvertir en signal optique en utilisant une longueur d'onde quelconque. Cette conversion est nommée « conversion O/E/O » (pour optique/électronique/optique). L'inconvénient principal de la conversion O/E/O est que le signal est traité au niveau électronique, ce qui réduit considérablement le débit. De plus, le niveau électronique traitant les paquets moins vite qu'ils arrivent, de nombreuses pertes peuvent se produire si la conversion O/E/O est beaucoup utilisée. L'utilisation de la conversion O/E/O doit être évitée autant que possible dans les réseaux tout optique.

2.1.2 Des réseaux électroniques aux réseaux tout optique

Dans les réseaux à haut débit, l'évolution des réseaux électroniques vers les réseaux tout optique est due aux avantages des fibres optiques et des routeurs tout optique.

Avantages des fibres optiques sur les câbles en cuivre

Les fibres optiques présentent beaucoup d'avantages par rapport aux câbles de cuivre :

- le débit sur une fibre optique est très important (en théorie, il peut atteindre 25 Tbps par fibre),
- l'atténuation du signal est suffisamment faible pour que la transmission puisse être réalisée sur de grandes distances (typiquement transocéaniques),
- les transmissions ne sont pas soumises aux perturbations électromagnétiques ou radiofréquences,
- les transmissions sur des fibres distinctes ne se parasitent pas mutuellement,
- le taux d'erreur est faible (de l'ordre de 10^{-12} pour les fibres optiques alors qu'il est de l'ordre de 10^{-6} pour les câbles en cuivre),
- les transmissions sur une fibre optique sont difficiles à écouter en dehors des extrémités,
- le coût d'une fibre optique est assez faible (ce qui explique que plusieurs fibres soient souvent installées simultanément),
- la puissance nécessitée est faible,
- la liaison résiste à la corrosion,
- l'installation peut être réalisée dans un milieu déflagrant, puisqu'il n'y a pas d'étincelles.

Avantages des routeurs tout optique sur les routeurs électroniques

Les routeurs optiques présentent eux aussi beaucoup d'avantages par rapport aux routeurs électroniques :

- les routeurs tout optique peuvent gérer les capacités de plusieurs Tbps des fibres optiques,
- le temps de traitement des paquets est très faible.

Architecture

Nous considérerons l'architecture suivante, représentée sur la figure 2.10 :

- les routeurs de bordure du domaine considéré sont des routeurs utilisateurs utilisant IP,
- tous les autres routeurs du domaine sont des WXC,
- les WXC connectés à des routeurs utilisateurs intègrent les fonctionnalités des WADM,
- il existe un réseau de contrôle des routeurs WXC et une entité (que nous supposerons centralisée) qui a la charge de gérer leur configuration.

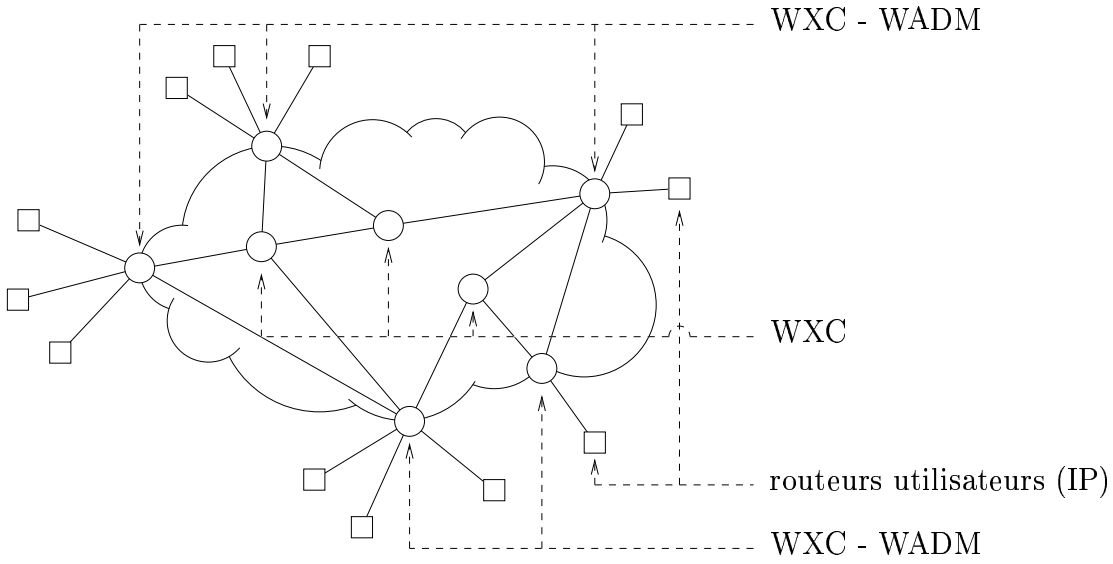


Figure 2.10 – *L'architecture des réseaux tout optique que nous étudions.*

Notons qu'aucun routeur optique ne dispose de files d'attente (optique). En effet, les débits étant très élevés, la quantité d'informations à stocker serait gigantesque. Le délai de transmission des informations sur un chemin est donc majoritairement dû aux délais de propagations sur les liens de ce chemin.

2.1.3 Contraintes et modélisation des réseaux tout optique

Dans ce chapitre, nous nous focalisons sur les réseaux tout optique à routage en longueurs d'onde. Ces réseaux sont prometteurs et sont conformes aux spécifications de GMPLS

(pour *Generalized Multi-Protocol Label Switching*) [RFC 3471]. Un tel réseau tout optique est constitué d'un ensemble de WXC interconnectés par des fibres optiques. En pratique, plusieurs routeurs utilisateurs sont connectés à chaque WXC. Ces routeurs utilisateurs émettent ou reçoivent les informations du WXC. Nous identifierons les routeurs utilisateurs au WXC auquel ils sont associés.

Dans la suite de ce chapitre, nous décrivons le routage en longueurs d'onde et nous modélisons les contraintes des WXC.

Le routage en longueurs d'onde

Dans un réseau à routage en longueurs d'onde, un message est acheminé d'un routeur source à un routeur destination en suivant un chemin optique. Un tel chemin ne nécessite pas de conversion O/E/O. Les données sont émises sur un chemin optique par un transmetteur optique et elles sont extraites du chemin optique par un récepteur optique.

Les caractéristiques primordiales d'un chemin optique sont qu'il est soumis à la contrainte de continuité de la longueur d'onde (*cf* la contrainte 1), à la contrainte d'assignation de longueurs d'onde distinctes (*cf* la contrainte 2) et qu'il possède la propriété de réutilisation des longueurs d'onde (*cf* la propriété 1).

Contrainte 1 (Continuité de la longueur d'onde). *La même longueur d'onde doit être utilisée sur tous les liens d'un chemin optique.*

Contrainte 2 (Assignation de longueurs d'onde distinctes). *Deux chemins optiques ne peuvent pas partager la même longueur d'onde sur une même fibre optique.*

Propriété 1 (Réutilisation des longueurs d'ondes). *Deux chemins optiques empruntant des fibres optiques différentes peuvent utiliser la même longueur d'onde.*

Pour réaliser la commutation de paquets dans un tel réseau, on utilise une approche à un saut ou une approche à plusieurs sauts.

L'approche à un saut nécessite qu'un chemin optique soit établi entre la source et la destination.

Pour l'approche à plusieurs sauts, les paquets sont acheminés de routeur en routeur par des chemins optiques. Au niveau des routeurs intermédiaires, qui reçoivent le message d'un chemin optique l_1 et qui doivent le transmettre sur un autre chemin optique l_2, trois cas peuvent se produire :

- les longueurs d'onde associées à l_1 et l_2 sont les mêmes, le routeur transmet simplement le message,
- les longueurs d'onde associées à l_1 et l_2 sont différentes et le routeur peut convertir la longueur d'onde ; il transmet le message après avoir converti la longueur d'onde,
- les longueurs d'onde associées à l_1 et l_2 sont différentes et le routeur ne peut pas convertir la longueur d'onde ; il réalise alors une conversion O/E/O du message avant de le retransmettre sur l_2.

Modélisation des contraintes

Un réseau tout optique est souvent modélisé comme un multigraphe orienté symétrique $G = (V, E)$, V étant l'ensemble des nœuds de G et E l'ensemble de ses arêtes. L'orientation

est due au fait qu'une fibre optique est unidirectionnelle. La symétrie vient du fait que les fibres optiques sont toujours appariées, une dans chaque direction. Enfin, plusieurs paires de fibres sont souvent installées entre deux routeurs car une fibre coûte beaucoup moins cher que son installation.

Comme indiqué précédemment, certains routeurs ne sont pas en mesure de dupliquer les signaux au niveau optique. De plus, certains routeurs ne peuvent dupliquer un signal qu'en un nombre limité de signaux. Ainsi, à chaque routeur $n \in V$ est associé sa capacité de duplication $s(n) \geq 1$. Un réseau est dit à capacité de duplication totale si chaque routeur $n \in V$ est capable de dupliquer un signal en un nombre quelconque de signaux ($s(n) = +\infty$). Un réseau est dit à capacité de duplication clairsemée si chaque routeur $n \in V$ est soit incapable de dupliquer un signal au niveau optique ($s(n) = 1$), soit capable de dupliquer un signal en un nombre quelconque de signaux ($s(n) = +\infty$). Dans les autres cas, le réseau est dit à capacité de duplication limitée.

Certains routeurs ne sont pas en mesure de convertir la longueur d'onde d'un signal au niveau optique. Ainsi, à chaque routeur $n \in V$ est associé sa capacité de conversion $c(n) \in \{0, 1\}$. Un réseau est dit à capacité de conversion totale si chaque routeur $n \in V$ est capable de convertir la longueur d'onde ($c(n) = 1$) et s'il n'y a pas de restrictions pour la conversion des longueurs d'onde. Un réseau est dit à capacité de conversion clairsemée si les routeurs sont soit capables de convertir la longueur d'onde ($c(n) = 1$) sans restriction, soit incapables de la convertir ($c(n) = 0$). Dans les autres cas, le réseau est dit à capacité de conversion limitée.

Dans ce chapitre, nous ne considérons que des réseaux à capacité de duplication clairsemée et sans capacité de conversion (sauf mention du contraire). La figure 2.11 présente les légendes associées à toutes les figures de ce chapitre. À chaque fois, nous représenterons la source d'un groupe *multicast* comme un nœud pouvant convertir pour ce groupe. En effet, la source recevant toujours les informations au niveau électronique, elle peut générer des signaux optiques de longueurs d'onde quelconques. Il faut noter que la capacité de conversion d'une source n'est valable que pour le groupe dont elle est la source.

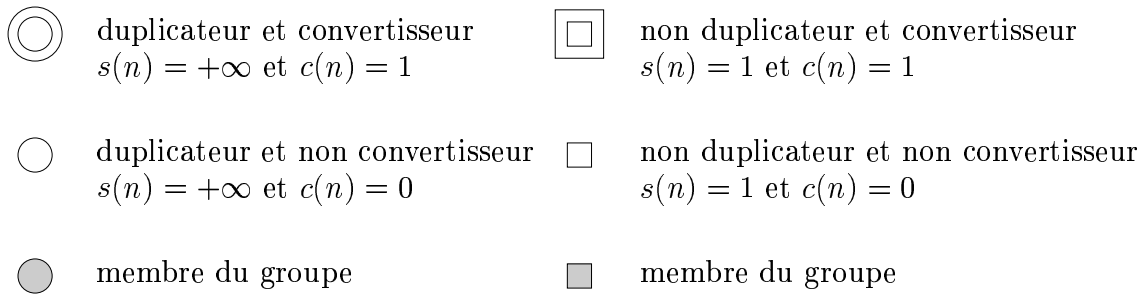


Figure 2.11 – *Légende des figures du chapitre 2.*

2.2 Le problème du routage *multicast*

L'étude de Malli, Zhang et Qiao [MZQ98] est la première à justifier l'utilisation du *multicast* dans les réseaux tout optique. Notamment, elle montre que le *multicast* réduit

considérablement le nombre de canaux de longueurs d'onde et le nombre de longueurs d'onde utilisés par rapport au multi-*unicast* dans les réseaux réalistes et dans plusieurs réseaux réels.

Le problème du routage *multicast* dans les réseaux tout optique consiste à satisfaire des requêtes de communications de groupes entre une source s et un ensemble M de destinataires en établissant des routes optiques. Remarquons que ce problème fait souvent abstraction de l'assignation des longueurs d'onde à ces routes optiques.

Généralement, plusieurs hypothèses sont faites sur le réseau :

- le réseau est sans capacité de conversion : $\forall n \in V,\ c(n) = 0$,
- le nombre de longueurs d'onde utilisables dans le réseau est arbitrairement grand.

L'hypothèse du grand nombre de longueurs d'onde utilisables vient de la présence de multiples fibres entre deux points.

La structure de chemin optique sert à réaliser des communications *unicast*. Pour réaliser des communications *multicast*, les structures d'arbres optiques et de forêts optiques sont utilisées. Nous allons décrire la structure d'arbre optique, la structure de forêt optique, puis les métriques permettant d'évaluer la qualité des solutions au problème du routage *multicast*.

2.2.1 L'arbre optique

L'arbre optique, initialement décrit dans [SM99], est l'équivalent point-à-multipoint du chemin optique. L'arbre optique est lui aussi soumis à la contrainte de continuité de la longueur d'onde (*cf* la contrainte 1) et à la contrainte d'assignation de longueurs d'onde distinctes (*cf* la contrainte 2). La duplication aux nœuds de branchement de l'arbre optique est réalisée au moyen de routeurs pouvant dupliquer les signaux. La contrainte 3, ci-dessous, indique qu'un nœud de branchement n d'un arbre optique ne peut pas dupliquer un signal en plus de signaux que ne lui permet sa capacité de duplication $s(n)$.

Contrainte 3 (Capacité de duplication). *Soit T un arbre optique. Pour tout nœud n de T, $d_T^+(n) \leq s(n)$, où $d_T^+(n)$ est le degré sortant de n dans T.*

À cause de cette contrainte de duplication, il n'est pas toujours possible de couvrir un groupe avec un seul arbre optique[1]. La figure 2.12 présente un réseau dans lequel aucun routeur n'est en mesure de dupliquer. Dans cette situation limite, deux arbres optiques t_1 et t_2 sont nécessaires pour couvrir les deux membres du groupe, r_5 et r_8, en raison de l'incapacité à dupliquer du routeur r_4.

Rappelons que nous supposons que l'extraction locale d'un signal pour les routeurs membres n'affecte pas leur capacité de duplication.

2.2.2 La forêt optique

Un seul arbre optique ne suffit pas toujours à couvrir un groupe. Il est parfois nécessaire de construire un ensemble d'arbres optiques qui ne sont pas disjoints (puisqu'ils sont tous enracinés en la source du groupe). Nous appellerons, par abus de langage, « forêt optique » un ensemble d'arbres optiques non nécessairement disjoints. Une forêt optique permet toujours

1. Nous verrons ultérieurement que c'est possible en utilisant une structure particulière présentée initialement dans [AD00].

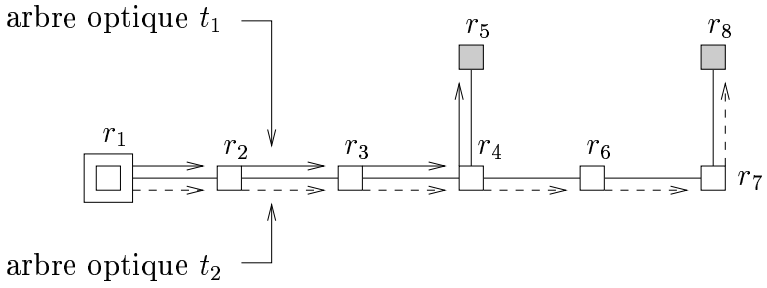


Figure 2.12 – *Deux arbres optiques sont nécessaires pour couvrir le groupe.*

de couvrir un groupe *multicast* (dans ce cas, tous les arbres de la forêt optique sont enracinés en la source du groupe) ou un ensemble de groupes. C'est pourquoi toutes les métriques annoncées par la suite seront définies sur une forêt optique. Notons que sur la figure 2.12, l'ensemble $\{t_1, t_2\}$ est déjà une forêt optique.

2.2.3 Les métriques

Dans un réseau électronique, chaque communication utilisant une arête réduit la bande passante disponible sur cette arête. Dans un réseau tout optique, la contrainte d'assignation de longueurs d'ondes distinctes (*cf* la contrainte 2) fait qu'un seul chemin optique est utilisé par longueur d'onde. La totalité des 40 Gbps disponibles sur un canal de longueur d'onde sont utilisés dès lors qu'un chemin optique est établi pour cette longueur d'onde, même si le chemin optique n'est utilisé que pour une communication de faible débit[2]. La ressource critique n'est plus la bande passante utilisée par la communication, mais le nombre de canaux de longueurs d'onde utilisés.

Métrique 1 (Nombre de canaux de longueurs d'onde). *Le nombre de canaux de longueurs d'onde utilisés par une forêt optique F est :*

$$NB.canaux(F) = \sum_{T \in F} NB.arcs(T),$$

où $NB.arcs(T)$ correspond au nombre d'arcs de l'arbre T.

La contrainte d'assignation de longueurs d'onde distinctes (*cf* la contrainte 2) impose que deux chemins ou arbres partageant la même fibre optique soient couverts par des longueurs d'onde différentes. Minimiser le nombre de longueurs d'onde nécessaires est très important, mais ne révèle pas du problème du routage *multicast*. Toutefois, l'algorithme de routage peut essayer de faciliter l'assignation des longueurs d'onde ultérieure.

Il semble que la minimisation de la superposition de chemins optiques ne facilite pas obligatoirement l'assignation des longueurs d'onde mais puisse même parfois la complexifier. À notre connaissance, ce résultat n'a pas été généralisé pour les arbres. La différence fondamentale entre un ensemble de chemins et un ensemble d'arbres enracinés en une même source et que la superposition des arbres autour de la source semble être le point critique.

2. C'est pourquoi plusieurs communications de faible débit sont agrégées dans les routeurs utilisateurs, afin de réduire cet inconvénient.

Plus il y a d'arbres dans la forêt à partager une même fibre, plus la recherche de longueurs d'onde convenables risque d'être difficile. Pour caractériser ce phénomène, nous définissons le nombre de longueurs d'onde minimal nécessaires pour réaliser une forêt optique. Nous utilisons le mot « couleurs » car l'association entre la longueur d'onde et la couleur est souvent faite.

Métrique 2 (Nombre de longueurs d'onde). *Le nombre minimal de longueurs d'onde nécessaires pour couvrir une forêt optique F est :*

$$NB.couleurs(F) = \max_{a \in A} card\{T \in F | T \text{ utilise l'arc } a\}.$$

Cette métrique permet aussi de comparer les performances des algorithmes de la littérature à celles de nos algorithmes. On peut retrouver des comparaisons basées sur cette métrique, entre autres, dans [ZWQ00].

Un transmetteur optique étant un composant relativement coûteux, il est important de réaliser des forêts optiques en nécessitant peu, c'est-à-dire ayant peu d'arbres. Il en va de même pour le nombre de récepteurs optiques. Cependant, le nombre de récepteurs optiques dans un réseau est égal au nombre de transmetteurs optiques puisqu'ils sont couplés.

Métrique 3 (Nombre de transmetteurs). *Le nombre de transmetteurs nécessaires pour réaliser une forêt optique F est :*

$$NB.transmetteurs(F) = \sum_{T \in F} d^+(racine(T)),$$

où $d^+(racine(T))$ désigne le degré sortant de la racine d'un arbre T.

Enfin, comme nous l'avons vu auparavant, le nombre de conversions O/E/O nécessaires pour réaliser une forêt est une métrique importante car elle influe sur les performances du routage.

Métrique 4 (Nombre de conversions O/E/O). *Le nombre $NB.oeo(F)$ de conversions O/E/O nécessaires pour réaliser une forêt optique F est le nombre d'arbres T de F tels que la racine de T n'est pas la source du groupe et tel que la racine de T ne peut pas convertir les longueurs d'onde.*

2.2.4 Le problème de Steiner contraint sur les degrés

Tel que nous l'avons décrit, le routage *multicast* consiste à trouver une forêt optique pouvant couvrir un groupe de source s et dont les membres constituent l'ensemble M. Puisque chaque arbre optique de la forêt optique est soumis à la contrainte de duplication, le problème de routage *multicast* est souvent formulé comme un problème de Steiner contraint sur les degrés [Vos92]. Ce problème consiste à trouver un arbre de poids minimum couvrant le sous-ensemble de nœuds $\{s\} \cup M$ tel que le degré de tout nœud dans l'arbre est inférieur ou égal à sa capacité de duplication.

Le problème de Steiner contraint sur les degrés est un problème NP-difficile [Vos90]. La NP-difficulté de ce problème ne vient pas uniquement de la NP-difficulté du problème de Steiner. L'ajout de la contrainte sur les degrés complexifie le problème. Pour s'en convaincre, on peut remarquer que le problème de trouver un arbre couvrant de poids minimum contraint sur les degrés est un problème NP-difficile [Joh85, Dou92], alors qu'il est polynomial sans la contrainte sur les degrés [Pri57, Kru56].

2.3 Heuristiques pour le routage *multicast*

Les heuristiques au problème de Steiner contraint sur les degrés sont souvent adaptées d'heuristiques au problème de Steiner sans contrainte. Nous étudions trois schémas d'adaptation : par post-traitement, par changement de topologie et par extension. Chaque description de schéma est illustrée d'heuristiques que nous comparerons en fin de paragraphe. Cette comparaison est une version étendue de notre synthèse effectuée dans [GMM03].

2.3.1 Schéma d'adaptation par post-traitement

Soit $\mathcal{H}$ une heuristique permettant de construire un arbre couvrant un groupe sans tenir compte de la contrainte de duplication (*cf* la contrainte 3). L'arbre $T_{\mathcal{H}}$ construit par $\mathcal{H}$ n'est pas réalisable au niveau optique si des routeurs de branchement de $T_{\mathcal{H}}$ ne sont pas en mesure de dupliquer les signaux. Le schéma d'adaptation par post-traitement consiste à modifier l'arbre $T_{\mathcal{H}}$ *a posteriori* pour le rendre conforme à la contrainte de duplication.

Les heuristiques RRTS et RRTA

Les heuristiques RRTS (pour *Re-route-to-Source*) et RRTA (pour *Re-route-to-Any*) ont été proposées par Zhang, Wei et Qiao dans [ZWQ00]. Dans ces deux heuristiques, pour obtenir à partir d'un arbre $T_{\mathcal{H}}$ une forêt optique satisfaisant la contrainte de duplication, les fils des nœuds de branchement de $T_{\mathcal{H}}$ qui sont incapables de dupliquer sont rattachés à des nœuds pouvant dupliquer. Dans l'heuristique RRTS, un fils m déconnecté est rattaché à son premier ancêtre pouvant dupliquer sur l'arbre. La forêt optique F obtenue par l'heuristique RRTS préserve $T_{\mathcal{H}}$ (*cf* la propriété 2, ci-dessous). Dans l'heuristique RRTA, un fils m déconnecté est rattaché au nœud le plus proche de la forêt en construction n'ayant pas encore épuisé sa capacité de duplication. Alors que l'heuristique RRTS peut considérer les nœuds de branchement dans un ordre quelconque, l'heuristique RRTA doit les considérer selon un parcours en largeur. L'heuristique RRTA ne préserve pas $T_{\mathcal{H}}$.

Propriété 2. *Soit $T_{\mathcal{H}}$ un arbre obtenu par une heuristique $\mathcal{H}$ et F la forêt optique obtenue par une heuristique de post-traitement préservant $T_{\mathcal{H}}$. Alors, on a :*

$$\bigcup_{T_i \in F} T_i = T_{\mathcal{H}}.$$

Dans cette propriété, nous assimilons un arbre à l'ensemble des arêtes qui le constitue. Si l'arbre T_1 est constitué de l'ensemble d'arêtes $\{e_1, e_2\}$ et si l'arbre T_2 est constitué de l'ensemble d'arêtes $\{e_2, e_3\}$, $T_1 \cup T_2$ représente l'ensemble d'arêtes $\{e_1, e_2, e_3\}$.

La figure 2.13 présente sur un même exemple les résultats de plusieurs heuristiques de post-traitement. L'arbre $T_{\mathcal{H}}$ obtenu par une heuristique $\mathcal{H}$ est représenté sur la figure 2.13(a). Il s'agit ici d'un arbre des plus courts chemins. Cet arbre n'est pas réalisable à cause des nœuds r_2 et r_7. La forêt optique obtenue par l'heuristique RRTS est présentée sur la figure 2.13(b). Le nœud r_2 est un nœud de branchement de $T_{\mathcal{H}}$ incapable de dupliquer. L'un de ses fils, r_4 dans l'exemple, est ajouté à l'arbre en construction. L'autre fils, r_3, est rattaché à son premier ancêtre pouvant dupliquer, le nœud r_1. Le routeur r_7 est un autre noeud de branchement de $T_{\mathcal{H}}$ incapable de dupliquer. L'un de ses fils, r_6 dans l'exemple, est ajouté à

l'arbre en construction. Les fils r_{10} et r_8 sont rattachés à la forêt optique par l'intermédiaire du nœud r_4. La forêt optique F obtenue est telle que $NB.canaux(F) = 11$, $NB.couleurs(F) = 3$ et $NB.transmetteurs(F) = 5$. Nous avons $NB.oeo(F) = 2$ car r_4 est racine de deux arbres alors qu'il ne peut pas convertir.

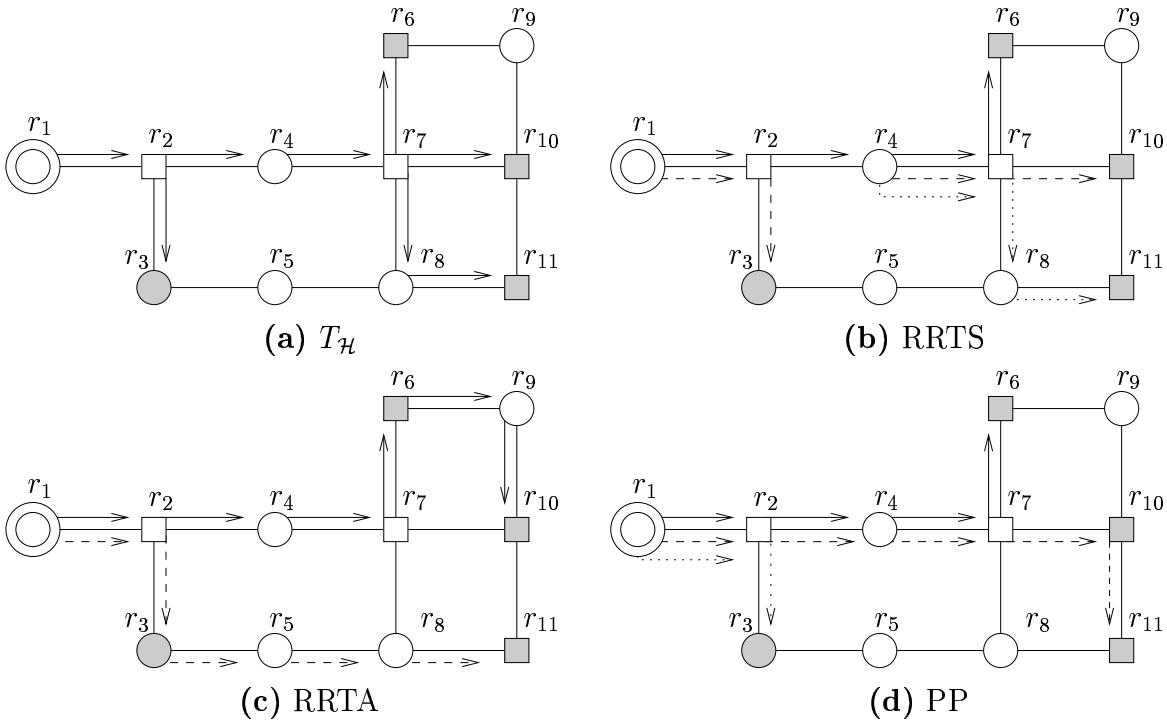


Figure 2.13 – *Les heuristiques RRTS, RRTA et PP sur un exemple.*

La forêt optique obtenue par l'heuristique RRTA est présentée sur la figure 2.13(c). Le nœud r_2 est un nœud de branchement de $T_{\mathcal{H}}$ incapable de dupliquer. L'un de ses fils, r_4 dans l'exemple, est ajouté à l'arbre en construction. L'autre fils, r_3 dans l'exemple, est rattaché au nœud le plus proche satisfaisant la contrainte, r_1 dans notre exemple. Le nœud r_3 n'a pas été rattaché au nœud r_8 car r_8 n'a pas encore été considéré par le parcours en largeur. Le routeur r_7 est un autre nœud de branchement de $T_{\mathcal{H}}$ incapable de dupliquer. L'un de ses fils, r_6 dans l'exemple, est ajouté à l'arbre en cours. Le fils r_{10} de r_7 est rattaché à la forêt optique par l'intermédiaire de r_6. Le fils r_8 est rattaché au nœud le plus proche, r_3. La forêt optique F obtenue est telle que $NB.canaux(F) = 11$, $NB.couleurs(F) = 2$ et $NB.transmetteurs(F) = 2$. Nous avons $NB.oeo(F) = 0$ dans ce cas.

La complexité dans le pire des cas de l'heuristique RRTS est $\mathcal{O}(|V|^2)$ (sans compter la complexité dans le pire des cas de l'heuristique $\mathcal{H}$), où $|V|$ dénote le nombre de nœuds du graphe.

La complexité dans le pire des cas de l'heuristique RRTA est $\mathcal{O}(|V| \cdot |E| \cdot \log |V|)$ (sans compter la complexité dans le pire des cas de l'heuristique $\mathcal{H}$), où $|V|$ dénote le nombre de nœuds du graphe et $|E|$ est le nombre d'arêtes de graphe.

L'heuristique PP

L'originalité de l'heuristique PP (pour *Post-Processing heuristic*) est qu'en plus de créer une forêt réalisable, cette forêt est améliorée par rapport à celles construites par les heuristiques RRTS et RRTA. Nous avons proposé l'heuristique PP dans [ZGM03a].

L'heuristique PP se compose de deux phases. Pendant la première phase, une forêt optique réalisable F est construite à partir de l'arbre $T_{\mathcal{H}}$ au moyen d'un mécanisme similaire à celui de RRTS. Toutefois, le rattachement des nœuds se fait systématiquement à la source (pour éviter d'utiliser des conversions O/E/O car les nœuds pouvant dupliquer ne peuvent pas forcément convertir la longueur d'onde). Cette première phase préserve $T_{\mathcal{H}}$. Pendant la seconde phase, nous procédons à l'amélioration de la forêt en réduisant autant que possible le nombre de canaux de longueurs d'onde et le nombre de longueurs d'onde. Pour cela, l'heuristique PP considère les arbres deux à deux. Deux arbres T_x et T_y sont fusionnés si et seulement si il existe x feuille de T_1 et y feuille de T_2 vérifiant $d(x,y) \leq \min\{d(x,s), d(y,s)\}$, où s est la source du groupe. S'il y a plusieurs couples (x,y) satisfaisant cette relation, l'un des couples est choisi arbitrairement. La fusion des arbres se fait en ajoutant le plus court chemin $p(x,y)$ de x à y. Comme $T_1 \cup T_2$ ne contient pas de boucle (selon la propriété 2), $T_1 \cup T_2 \cup p(x,y)$ contient une seule boucle qu'il faut retirer. L'heuristique PP retire le plus long chemin sur la boucle ayant les propriétés suivantes :

- ses extrémités sont soit des membres, soit des nœuds de degré supérieur ou égal à 3,
- ses nœuds intermédiaires sont des nœuds non membres de degré 2.

L'arbre T_3 résultant de la fusion de T_1 et de T_2 ne peut plus être reconsidéré pour une fusion ultérieure car il ne permet pas de préserver un sous-arbre de $T_{\mathcal{H}}$. Cette opération de fusion d'arbres suivie de la suppression d'arêtes pour en déduire une forêt est détaillée dans [MM00]. Quand $k = 1$, nous sommes dans le cas le plus simple où il n'y a qu'une seule boucle dans le graphe fusionné.

Cette fusion vise à réduire à la fois le nombre de canaux de longueurs d'onde utilisés par la forêt (ayant choisi deux nœuds x et y proches, le chemin supprimé a de grandes chances d'être plus long que le chemin ajouté), le nombre de longueurs d'onde nécessaires (puisqu'il n'y a plus qu'un seul arbre T_3 au lieu de deux arbres T_1 et T_2 partiellement superposés) et le nombre de transmetteurs (par rapport aux heuristiques RRTS et RRTA qui ne refusionnent pas les arbres).

Une forêt optique obtenue par l'heuristique PP est présentée sur la figure 2.13(d). Pendant la première phase de cette heuristique, l'arbre $T_{\mathcal{H}}$ est transformé en une forêt $F = \{T_{r_3}, T_{r_6}, T_{r_{10}}, T_{r_{11}}\}$, où chaque arbre T_{r_i} est sur l'exemple un chemin de r_1 à r_i. Pendant la seconde phase, les arbres de F sont considérés deux à deux. Les feuilles $x = r_{10}$ et $y = r_{11}$ des arbres $T_{r_{10}}$ et $T_{r_{11}}$ sont telles que $d(x,y) = 1 \leq \min\{d(x,r_1), d(y,r_1)\} = 4$. Nous procédons alors à la fusion de $T_{r_{10}}$ et de $T_{r_{11}}$. La boucle $(r_{10}, r_{11}, r_8, r_7, r_{10})$ apparaît dans l'arbre fusionné $T_{r_{10}r_{11}}$. Pour la retirer, il suffit de supprimer les arêtes (r_{11}, r_8) et (r_8, r_7). La nouvelle forêt obtenue est $F = \{T_{r_3}, T_{r_{10}r_{11}}, T_{r_6}\}$. Cette forêt ne peut plus être améliorée selon l'heuristique PP. Elle est telle que $NB.canaux(F) = 11$, $NB.couleurs(F) = 3$, $NB.transmetteurs(F) = 3$ et $NB.oeo(F) = 0$.

La complexité dans le pire des cas de l'heuristique PP est $\mathcal{O}(|M|^2 \cdot |E| \cdot \log|V|)$, où $|M|$ dénote le nombre de membres du groupe, $|E|$ le nombre d'arêtes du graphe et $|V|$ le nombre

de nœuds du graphe.

2.3.2 Schéma d'adaptation par changement de topologie

Le schéma d'adaptation par changement de topologie consiste à appliquer une heuristique non adaptée $\mathcal{H}$ sur une topologie modifiée afin d'intégrer les contraintes de duplication indirectement à $\mathcal{H}$. La complexité de l'adaptation réside dans la manière dont la topologie est changée. Nous avons proposé dans [GM03] deux changements de topologie, l'un direct et l'autre itératif.

L'heuristique directe

Pour prendre en compte l'incapacité des nœuds à dupliquer, l'heuristique DH (pour *Direct Heuristic*) augmente le poids des arêtes autour de ces nœuds. Plus précisément, à chaque fois qu'un chemin p est ajouté à l'arbre courant par l'heuristique $\mathcal{H}$, le poids des arêtes adjacentes aux nœuds incapables de dupliquer de p est augmenté. L'augmentation du poids d'une arête e dépend d'une part d'un paramètre α qui permet de quantifier le coût de création d'un nouvel arbre, et d'autre part de la distance du nœud incapable de dupliquer à la source.

Une fois que tous les membres ont été connectés, il faut retransformer l'unique arbre obtenu en une forêt satisfaisant la contrainte de duplication. Cela est fait grâce à un mécanisme similaire à celui de RRTS, exception faite du rattachement qui a toujours lieu à la source. Contrairement à RRTS, l'heuristique directe ne suppose pas que tous les nœuds pouvant dupliquer peuvent aussi convertir. Ainsi, les forêts construites par l'heuristique directe ne nécessitent pas de conversions O/E/O.

L'algorithme 1 présente une version plus formelle de cette description. Dans cet algorithme, $d_{T,G}(s,n)$ correspond à la distance de s à n dans l'arbre T avec le poids des arêtes du graphe G.

La suppression de p dans $\bar{G}$ sert à éviter que les arêtes de p aient leur poids augmenté deux fois. Toutefois, cela ne poserait pas de problème pour le déroulement de l'algorithme, puisque les arêtes de p ne peuvent plus être sélectionnées à nouveau au cours des itérations ultérieures.

La complexité dans le pire des cas de l'heuristique directe est $\mathcal{O}(|M| \cdot |E| \cdot |V|)$, où $|M|$ dénote le nombre de membres du groupe, $|E|$ le nombre d'arêtes du réseau et $|V|$ le nombre de nœuds du réseau.

Utilisation de sources virtuelles. Le comportement de l'heuristique directe peut être amélioré si nous considérons les sources virtuelles, c'est-à-dire les nœuds capables à la fois de dupliquer et de convertir la longueur d'onde. Dans la version initiale de l'heuristique directe, chaque arbre est reconstruit à partir de la source s. En détectant la présence d'une source virtuelle sv dans l'entourage d'un nœud n, l'heuristique directe peut obtenir de meilleures performances. La figure 2.14 présente la construction de la forêt avec un rattachement en s ou en sv. La forêt de la figure 2.14(a) a un coût relatif de $d_T(s,n)$ et la forêt de la figure 2.14(b) un coût relatif de $\alpha + 3d(sv,n)$. Cette différence de coût sert de critère pour déterminer s'il est plus avantageux de se rattacher à s ou à sv.

Pré-requis : G un réseau, s la source du groupe, M l'ensemble des membres du groupe et α le paramètre de l'heuristique
Retourne : F une forêt optique
$\bar{G} \leftarrow G$
transformer dans $\bar{G}$ les nœuds incapables de dupliquer en nœuds pouvant dupliquer
$T \leftarrow$un arbre vide enraciné en s
tantque $M \neq \emptyset$ **faire**
$(p, m) \leftarrow$un chemin dans $\bar{G}$ de T à un routeur m de M (selon une heuristique $\mathcal{H}$)
$T \leftarrow T \cup p$
$\bar{G} \leftarrow \bar{G} \backslash p$
pourtout nœud n appartenant à p, sauf m **faire**
si n ne peut pas dupliquer dans G **alors**
augmenter dans $\bar{G}$ le poids de toutes les arêtes adjacentes à n de $\alpha + d_{T,G}(s, n)$
fin si
fin pour
$M \leftarrow M \backslash \{m\}$
fin tantque
$F \leftarrow$ reconstruire une forêt optique à partir de T et G

Algorithme 1: L'heuristique directe.

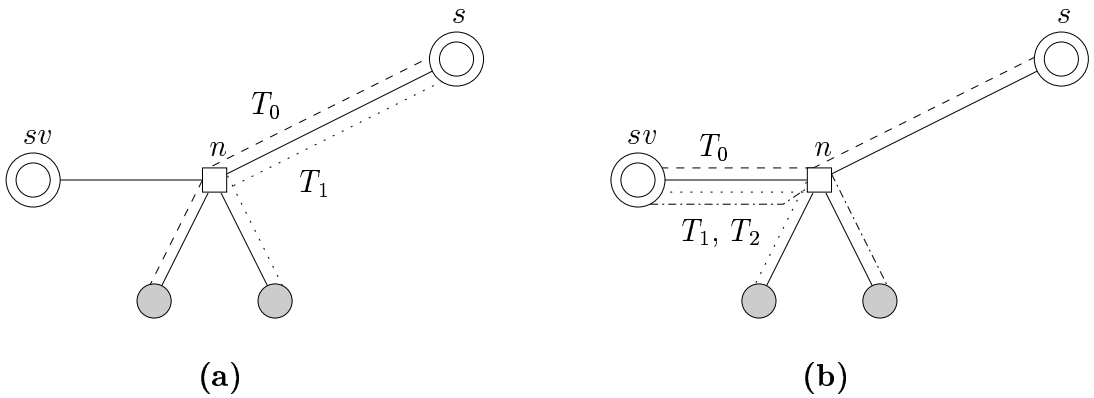


Figure 2.14 – *L'heuristique directe en présence d'une source virtuelle sv proche de n.*

L'heuristique itérative

Plutôt que de modifier l'heuristique $\mathcal{H}$ pour y inclure l'augmentation du coût des arêtes, l'heuristique IH (pour *Iterative Heuristic*) applique l'heuristique $\mathcal{H}$ inchangée sur des topologies modifiées successivement. Les changements de topologie servent à influencer les choix de $\mathcal{H}$ pour lui faire prendre en compte de manière indirecte les nœuds ne pouvant pas dupliquer. À l'itération i de l'heuristique itérative, un arbre couvrant T_i est construit sur un graphe G_i par l'heuristique $\mathcal{H}$. Si l'arbre T_i ne peut pas être réalisé à cause des contraintes de duplication, un nouveau graphe G_{i+1} est construit et le procédé recommence.

Le passage du graphe G_i au graphe G_{i+1} est déterminant pour l'heuristique itérative. Chaque nœud de branchement n de l'arbre T_i qui se révèle être incapable de dupliquer est considéré tour à tour. Notons n' le père de n dans T_i et notons $\{n_j\}$ l'ensemble des fils de n dans T_i. Le graphe G_{i+1} est obtenu en appliquant les modifications suivantes sur G_i :

- dans G_{i+1}, n est un nœud membre,
- les arêtes du type (n, p) avec $p \notin T_i$ n'apparaissent pas dans G_{i+1},
- l'arête (n, n') est conservée dans G_{i+1} et son coût est inchangé,
- chaque arête (n, n_j) de G_i, avec n_j fils de n dans T_i, est remplacée par une arête (n', n_j) dans G_{i+1}, dont le poids est égal au poids de l'arête (n, n_j) dans G_i augmenté de la valeur $d_{T,G_i}(s, n)$.

La figure 2.15 présente le passage d'un graphe G_i à un graphe G_{i+1}. Nous pouvons noter que le coût des arêtes (n', n_1) et (n', n_2) a été augmenté.

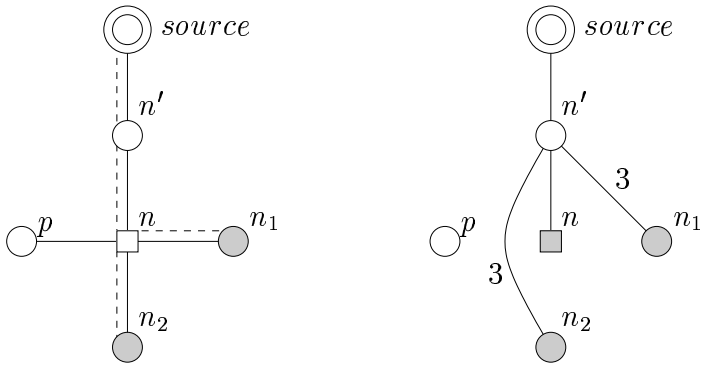


Figure 2.15 – *Passage d'un graphe G_i à un graphe G_{i+1} dans l'heuristique itérative.*

La complexité dans le pire des cas de l'heuristique itérative est $\mathcal{O}(|V| \cdot |E| \cdot \log |V|)$, où $|V|$ dénote le nombre de nœuds du graphe et $|E|$ le nombre d'arêtes du graphe.

2.3.3 Schéma d'adaptation par extension

Le schéma d'adaptation par extension consiste à intégrer la contrainte de duplication au sein d'une heuristique $\mathcal{H}$ conçue initialement pour les réseaux sans contrainte. Cette adaptation n'est pas simple en général et elle est même particulièrement difficile, par exemple dans le cas de l'heuristique KMB (décrite dans le chapitre 1) utilisant des fermetures métriques. Néanmoins, les heuristiques ainsi adaptées sont souvent très efficaces.

L'heuristique MO

L'heuristique MO (pour *Member-Only*) est une heuristique très importante dans le domaine des réseaux tout optique. Elle a été proposée par ZHANG, WEI et QIAO dans [ZWQ00]. Des versions préalables ont été présentées par MALLI, ZHANG et QIAO dans [MZQ98] et par BAUER et VARMA dans [BV95] sous le nom SPH (pour *Shortest Path Heuristic*). L'heuristique MO est une adaptation intuitive d'une heuristique très utilisée au problème de Steiner.

L'heuristique MO est basée sur le principe de l'heuristique TM (décrite dans le chapitre 1). Initialement, un arbre vide est enraciné à la source du groupe. Puis, à chaque itération, un nœud membre du groupe non encore connecté est ajouté à cet arbre par un plus court chemin particulier. Ce plus court chemin doit partir d'un nœud de l'arbre dont la capacité de duplication n'est pas encore épuisée et ne peut passer par aucun nœud de l'arbre. Si aucun nœud membre du groupe ne peut être ajouté à l'arbre par un tel chemin, un nouvel arbre est réinitialisé à la source afin de couvrir les membres restants. L'algorithme 2 formalise cette description.

Pré-requis : G un réseau, s la source du groupe et M l'ensemble des membres du groupe
Retourne : F est une forêt optique
  $F \leftarrow \emptyset$
  tantque $M \neq \emptyset$ **faire**
    $T \leftarrow$ un arbre vide enraciné en s
    faire
      $(p, m) \leftarrow$ le plus court chemin p dans G de l'arbre T à un routeur m de M, tel que p parte d'un nœud de T dont la capacité de duplication n'est pas épuisée et tel que p n'emprunte aucun autre nœud de T
      si un tel chemin p existe **alors**
        $T \leftarrow T \cup \{p\}$
        $M \leftarrow M \backslash \{m\}$
      fin si
    jusqu'à ce que il n'existe plus de tel chemin p
    $F \leftarrow F \cup T$
  fin tantque

Algorithme 2: L'heuristique MO (pour *Member-Only*).

La figure 2.16 présente la forêt obtenue par l'heuristique MO sur un exemple. Initialement, un arbre T_1 est initialisé à la racine r_1. Le membre r_3 puis le membre r_4 sont ajoutés à T_1. Comme r_7 et r_8 ne peuvent pas dupliquer, l'arbre T_1 ne peut plus être étendu. Un nouvel arbre T_2 est initialisé à la racine r_1 et connecte r_6 puis r_9. La forêt obtenue F est telle que $NB.canaux(F) = 11$, $NB.couleurs(F) = 2$ et $NB.transmetteurs(F) = 2$. Les deux arbres étant enracinés en la source, nous avons $NB.oeo(F) = 0$. Notons que l'heuristique MO a tendance à construire des forêts ayant peu d'arbres.

Considérons à présent le réseau représenté sur la figure 2.17. L'heuristique MO trouve la

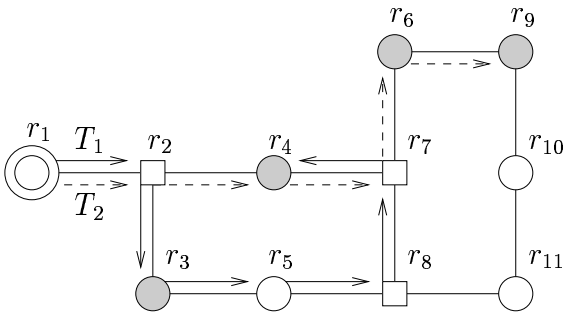


Figure 2.16 – *L'heuristique MO sur un exemple.*

forêt $F_1 = \{T_1^1\}$ représentée sur la figure 2.17(a). Cette forêt est telle que $NB.canaux(F_1) = 5$, $NB.couleurs(F_1) = 1$ et $NB.transmetteurs(F_1) = 1$. L'unique arbre de la forêt F_1 est de coût minimum en nombre de canaux : aucun arbre couvrant le groupe et respectant les contraintes n'est de coût inférieur à 5. Cependant, la forêt F_1 n'est pas de coût minimum : la forêt $F_2 = \{T_1^2, T_2^2\}$ représentée sur la figure 2.17(b) est de coût inférieur puisque $NB.canaux(F_2) = 4$. En revanche, $NB.couleurs(F_2) = 2$ et $NB.transmetteurs(F_2) = 2$.

(a) **(b)**

Figure 2.17 – *Le coût d'un arbre de coût minimum est supérieur au coût d'une forêt de coût minimum.*

La complexité dans le pire des cas de l'heuristique MO est $\mathcal{O}(|M| \cdot |E| \cdot \log |V|)$, où $|M|$ dénote le nombre de membres du groupe, $|E|$ le nombre d'arêtes du graphe et $|V|$ le nombre de nœuds du graphe.

L'heuristique MKH

De la même manière que pour l'algorithme de Prim, l'algorithme de Kruskal pour les arbres couvrants de poids minimum [Kru56] a été adapté pour les arbres couvrants partiels de poids minimum dans [Kru56]. C'est à partir de cette dernière heuristique que nous avons développé l'heuristique MKH (pour *Modified Kruskal Heuristic*). L'heuristique MKH est détaillée dans [ZGM03b].

Au début de l'heuristique, un arbre est initialisé pour la source et pour chaque membre du groupe. À chaque itération, les deux arbres les plus proches T_1 et T_2 sont connectés par un plus court chemin particulier. Ce chemin doit avoir une extrémité en une feuille ou en un nœud pouvant dupliquer de T_1, l'autre extrémité en une feuille ou en un nœud pouvant dupliquer de T_2, et aucun autre nœud sur un des arbres de la forêt. Lorsqu'il n'y a plus

d'arbres à connecter de cette manière, chaque arbre est rattaché à la source par un plus court chemin à un nœud pouvant dupliquer. S'il n'existe aucun chemin de la source à un nœud pouvant dupliquer, le rattachement a lieu en un nœud ne pouvant pas dupliquer. L'arbre est alors scindé en deux arbres autour de ce nœud.

La figure 2.18 présente les itérations de l'heuristique MKH. Initialement, les arbres de T_0 à T_6 sont construits. À la première itération, T_5 et T_6 sont fusionnés par un chemin de longueur 1 en un arbre T_7. À la cinquième itération, T_8 et T_{10} sont fusionnés par un chemin de longueur 3 en un arbre T_{11}. Les deux arbres restant, T_9 et T_{11}, ne peuvent plus être fusionnés. T_{11} est alors rattaché à la source r_1 par un plus court chemin. Comme aucun nœud de rattachement ne peut dupliquer, l'arbre T_{11} est scindé en deux arbres autour du plus proche nœud de rattachement. La forêt résultante F est telle que $NB.canaux(F) = 13$, $NB.couleurs(F) = 3$ et $NB.transmetteurs(F) = 3$. Les arbres étant enracinés à la source, nous avons toujours $NB.oeo(F) = 0$.

La complexité dans le pire des cas de l'heuristique MKH est $\mathcal{O}(|M| \cdot |E| \cdot \log |V|)$, où $|M|$ dénote le nombre de membres du groupe, $|E|$ le nombre d'arêtes du graphe et $|V|$ le nombre de nœuds du graphe.

L'heuristique CTH

L'heuristique CTH (pour *Connection Tree Heuristic*) est basée sur la famille des heuristiques des k-distances, une famille d'heuristiques au problème de Steiner. Après avoir décrit cette famille d'heuristiques, nous décrivons l'heuristique CTH.

La famille d'heuristiques des k-distances. La famille d'heuristiques des k-distances a été proposée dans [MM99] et dans [Mol99]. Alors que plusieurs heuristiques au problème de Steiner (dont l'heuristique TM et l'heuristique de Kruskal) utilisent des plus courts chemins pour les connexions, la famille d'heuristiques des k-distances utilise des arbres de Steiner ayant au plus k nœuds de branchement pour les connexions.

Plus précisément, la connexion d'un nœud n à un arbre T se fait par un arbre de connexion C satisfaisant les contraintes suivantes :

- C possède au plus k nœuds de branchement qui peuvent tous dupliquer,
- n est une feuille de C et les autres feuilles de C sont des nœuds de T,
- C n'utilise aucune arête de T,
- C est de coût minimal parmi tous les arbres de connexion respectant les propriétés précédentes.

Lorsque $k = 0$, la connexion C est un plus court chemin. Dans les autres cas, l'ajout de C à T introduit des boucles qu'il faut supprimer pour conserver une structure d'arbre. Nous associons à l'arbre C un ensemble d'arêtes E_C à supprimer. L'ensemble E_C permet donc de réviser l'arbre en construction. Cette révision permet d'améliorer les décisions prises aux itérations précédentes de l'algorithme.

La connexion optimale est celle qui minimise $w(C) - w(E_C)$, où la fonction w représente le coût d'un ensemble d'arêtes donné. Déterminer la connexion optimale est un problème NP-difficile. En effet, trouver un arbre de Steiner C à k nœuds de branchement et trouver un ensemble optimal E_C d'arêtes à supprimer sont tous deux des problèmes NP-difficiles (voir

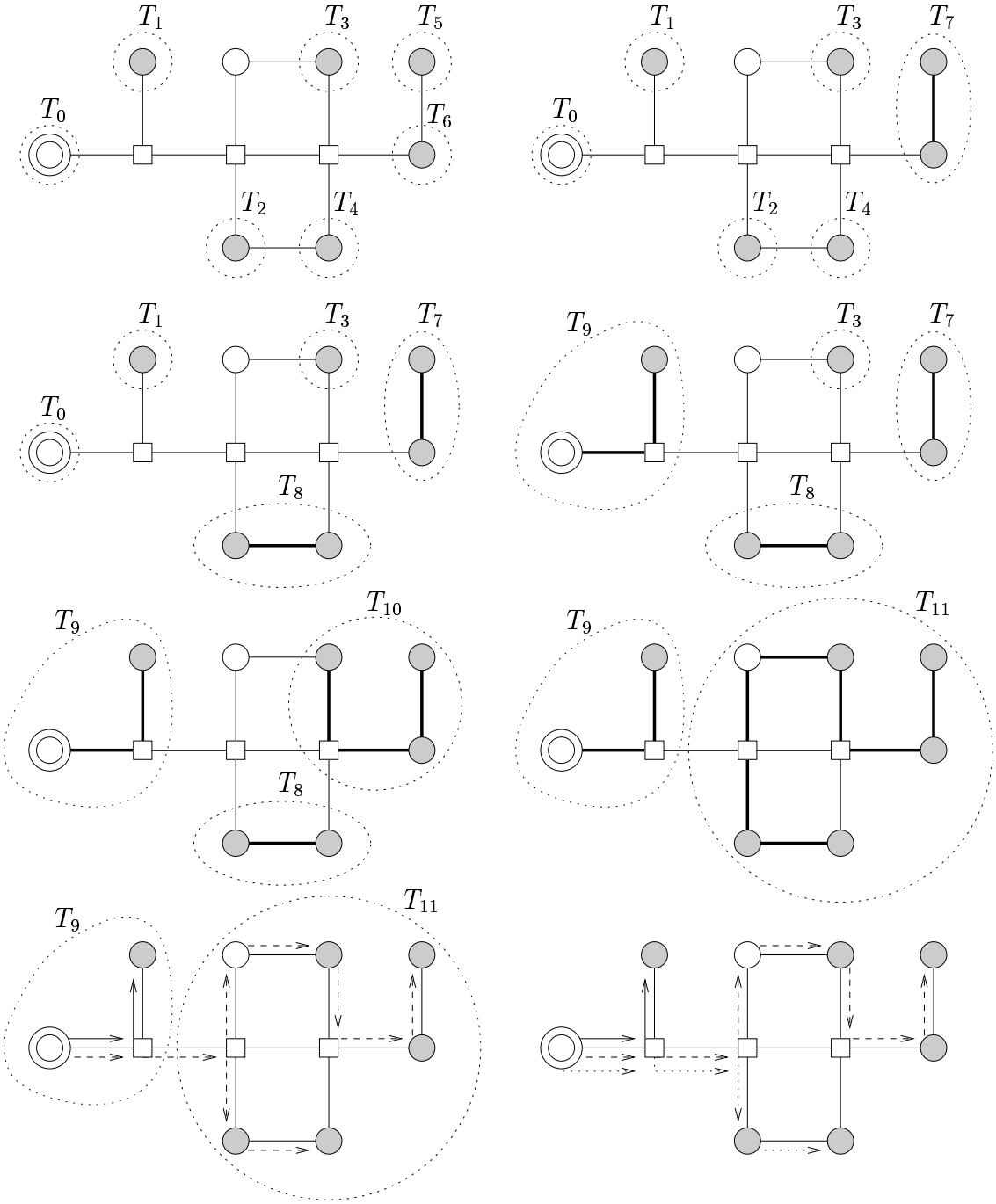


Figure 2.18 – *L'heuristique MKH sur un exemple.*

respectivement [Hak71] et [MM00]). En pratique, nous nous limitons à de petites valeurs de k pour limiter la complexité.

La figure 2.19 présente la connexion d'un nœud n à un arbre T par une heuristique de la famille des heuristiques des k-distances, avec $k = 1$. L'arbre C, représenté en pointillés, satisfait bien les contraintes suivantes :

- C possède un seul nœud de branchement b, pouvant dupliquer,
- n est une feuille de C et les deux autres feuilles de C, x_1 et x_2, sont des nœuds de T,
- C n'utilise aucune arête de T,
- C est de coût minimal parmi les arbres respectant les contraintes mentionnées.

Ajouter C à T crée un cycle (dans notre exemple, il s'agit de $(b, x_1, s, r_1, r_2, x_2, b)$). Le plus grand ensemble d'arêtes E_C que nous pouvons supprimer de T sans déconnecter un membre du groupe est $\{(x_2, r_2), (r_2, r_1)\}$. Dans le cas général, le papier [MM00] prouve que la recherche de l'ensemble maximal E_C est un problème NP-difficile. Le coût de la connexion de n à T est $w(C) - w(E_C) = 3 - 2 = 1$. Dans ce cas, la connexion par l'arbre C est moins coûteuse que par un plus court chemin.

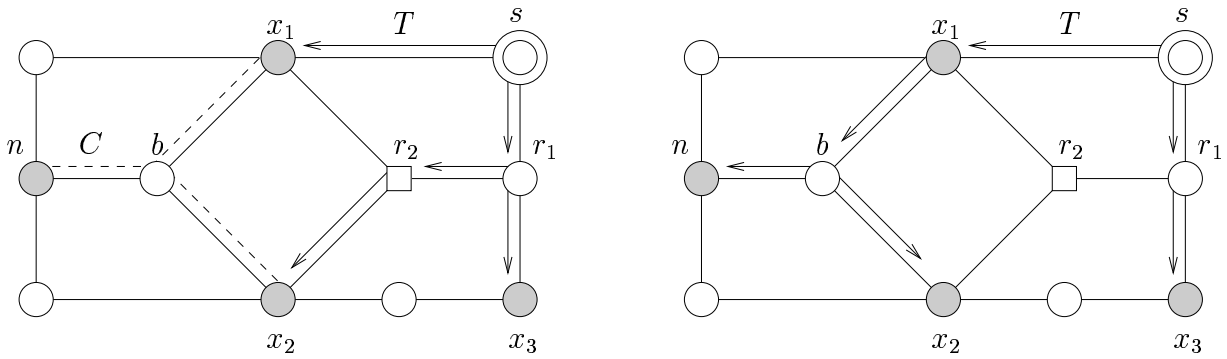


Figure 2.19 – *Connexion de n à T par une heuristique de la famille des heuristiques des k-distances.*

Description de CTH. Nous avons proposé l'heuristique CTH (pour *Connection Tree Heuristic*) dans [ZGM03b]. L'heuristique CTH est basée sur l'heuristique MO, à l'exception que les connexions se font par des 1-distances (étoiles ou chemins) plutôt que par des plus courts chemins uniquement. Quand aucun nœud du réseau ne peut dupliquer, l'heuristique CTH est incapable de réaliser des connexions par des étoiles car aucun nœud n'est susceptible d'être nœud de branchement. Toutefois, le comportement de l'heuristique CTH reste différent de celui de l'heuristique MO, même dans ce cas. En effet, l'heuristique CTH peut réviser l'arbre T en construction, même si aucun nœud de branchement n'est trouvé. C'est le cas sur l'exemple de la figure 2.20. L'arbre C, composé du chemin de x_1 à n et du chemin de n à x_2, peut être ajouté à T car la suppression de $E_C = \{(x_1, x_2)\}$ libère les nœuds x_1 et x_2 dont la capacité de duplication était épuisée. Sur cet exemple, l'heuristique MO est incapable d'étendre l'arbre T.

De plus, lorsque le réseau est très contraint, l'heuristique CTH peut être amenée à connecter plusieurs membres du groupe par itération. C'est le cas sur l'exemple de la figure 2.21. Ce

MO ne peut pas connecter n CTH connecte n

Figure 2.20 – *Dans des réseaux très contraints, CTH et MO se comportent différemment.*

n'est jamais le cas pour l'heuristique MO ou pour l'heuristique MKH. Cette caractéristique accélère l'algorithme dans ces cas, puisqu'elle réduit le nombre d'itérations de l'algorithme.

MO ne peut ni connecter n_1, ni n_2 CTH connecte n_1 et n_2 simultanément

Figure 2.21 – *Dans des réseaux très contraints, CTH peut connecter plusieurs membres à la fois.*

La complexité dans le pire des cas de l'heuristique CTH est $\mathcal{O}(|M| \cdot |V|^3)$, où $|M|$ dénote le nombre de membres du groupe et $|V|$ le nombre de nœuds du graphe.

2.3.4 Comparaison des schémas d'adaptation et des heuristiques

Dans ce paragraphe, nous comparons les heuristiques que nous venons de décrire en fonction des métriques introduites en début de chapitre. Nous ferons apparaître l'heuristique MO sur chaque figure en tant que témoin.

Chaque scénario de simulation présente les résultats des heuristiques pour 1000 groupes *multicast* concurrents. La taille de ces groupes est choisie uniformément entre 2 et le nombre de nœuds du graphe, ce qui correspond à de grands groupes. Les membres sont choisis uniformément parmi les nœuds du graphe et la source de chaque groupe est choisie uniformément parmi les membres du groupe. Nous avons fait varier le pourcentage de nœuds pouvant dupliquer dans le réseau de 0 à 100 par pas de 10. Les simulations sont faites sur les réseaux Abilène et Renater, présentés dans l'annexe B.

Les points sur les figures correspondent à la moyenne de 250 scénarii de simulation. L'heuristique $\mathcal{H}$, quand elle est utilisée (c'est-à-dire pour les heuristiques RRTS, RRTA, PP, DH et IH) construit un arbre des plus courts chemins.

Les scripts de ces simulations se trouvent dans l'annexe C.

Choix du paramètre α de l'heuristique directe

Le choix du paramètre α dans l'heuristique DH est important puisqu'il influence le comportement de l'heuristique. Lorsque la valeur du paramètre est fixée à 0, l'heuristique trouve une forêt dont le nombre d'arbres peut être très important. *A contrario*, lorsque la valeur du paramètre α est grande, l'heuristique a tendance à étendre le même arbre autant que possible, et donc à construire peu d'arbres. Ce comportement est similaire à celui de l'heuristique MO.

Notons F_α la forêt optique construite par l'heuristique directe de paramètre α. Nous pouvons définir la fonction de coût $K(\alpha) = NB.canaux(F_\alpha) + \beta \cdot NB.transmetteurs(F_\alpha)$, où β est le coût réel (que nous supposerons constant) de la création d'un nouvel arbre. La fonction $NB.transmetteurs(F_\alpha)$ est une fonction décroissante de α, alors que la fonction $NB.canaux(F_\alpha)$ est une fonction croissante de α. Ceci nous conduit à retenir la valeur α^*_β qui minimise la fonction $K(\alpha)$.

Pour évaluer l'importance du paramètre α, nous avons simulé l'heuristique directe en faisant varier α. Nous avons choisi des valeurs entières pour ce paramètre. Seules les valeurs significatives de α sont représentées sur les courbes. Pour Abilène, les valeurs de α comprises entre 2 et 11 (valeur maximale correspondant au nombre de nœuds d'Abilène) offrent des résultats similaires. C'est pourquoi n'apparaissent sur les figures que les valeurs 0, 1 et 11. Pour Renater, ce sont les valeurs de α comprises entre 3 et 25 (nombre de nœuds de Renater) qui offrent des résultats similaires. C'est pourquoi n'apparaissent sur les figures que les valeurs 0, 1, 2 et 25.

La figure 2.22 présente le nombre de canaux utilisés par les forêts construites par l'heuristique directe, en fonction de α. Nous remarquons que le nombre de canaux varie peu avec α, surtout quand le nombre de nœuds pouvant dupliquer est grand. Toutefois, nous remarquons que les petites valeurs de α réduisent le nombre de canaux. Ce comportement est attendu : en prenant peu en compte le coût α de reconstruction d'un nouvel arbre, nous réalisons les connexions nécessitant le moins de canaux. Une valeur de α plus grande aurait privilégié l'extension d'arbres existants, même si un arbre plus court aurait pu être créé. Ainsi, plus de canaux auraient été nécessités.

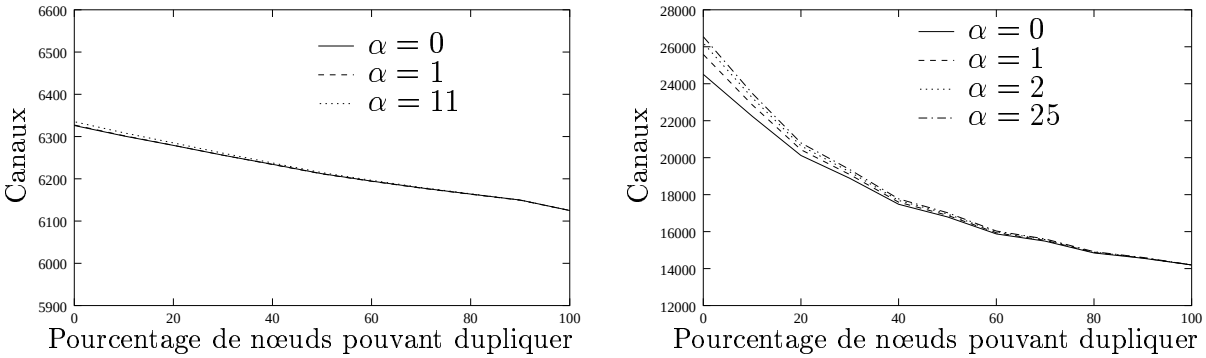


Figure 2.22 – *Nombre de canaux de longueurs d'onde utilisés, sur les réseaux Abilène (à gauche) et Renater (à droite).*

La figure 2.23 présente le nombre de transmetteurs utilisés par les forêts construites par

l'heuristique directe, en fonction de α. Nous remarquons que le nombre de transmetteurs décroît quand α augmente. En effet, quand α augmente, le nombre d'arbres créés décroît, et donc le nombre de transmetteurs utilisés.

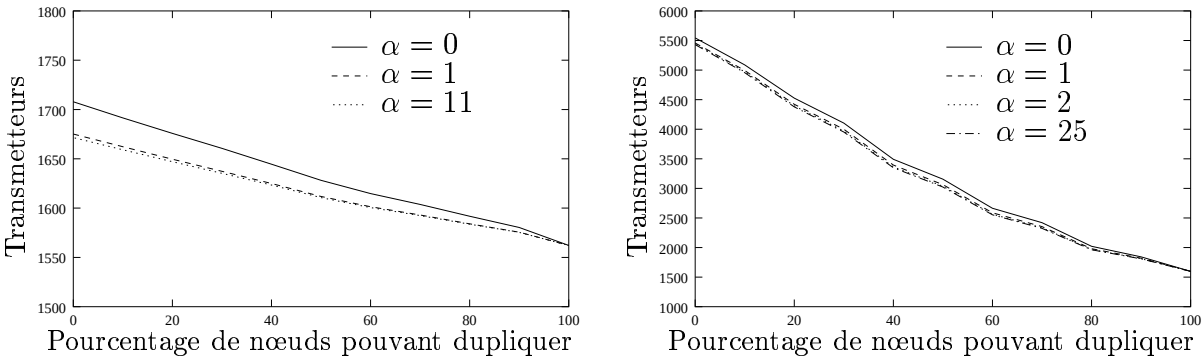


Figure 2.23 – *Nombre de transmetteurs utilisés, sur les réseaux Abilène (à gauche) et Renater (à droite).*

La figure 2.24 présente le nombre de longueurs d'onde utilisées par les forêts construites par l'heuristique directe, en fonction de α. Sur le réseau Renater, lorsque le pourcentage de nœuds pouvant dupliquer est faible (inférieur à 20 %), les petites valeurs de α construisent des forêts nécessitant très peu de longueurs d'onde. *A contrario*, lorsque le pourcentage de nœuds ne pouvant pas dupliquer est importante (supérieur à 40 %), les grandes valeurs de α sont plus performantes. Sur un petit réseau comme Abilène, toutes les valeurs de α offrent les mêmes performances.

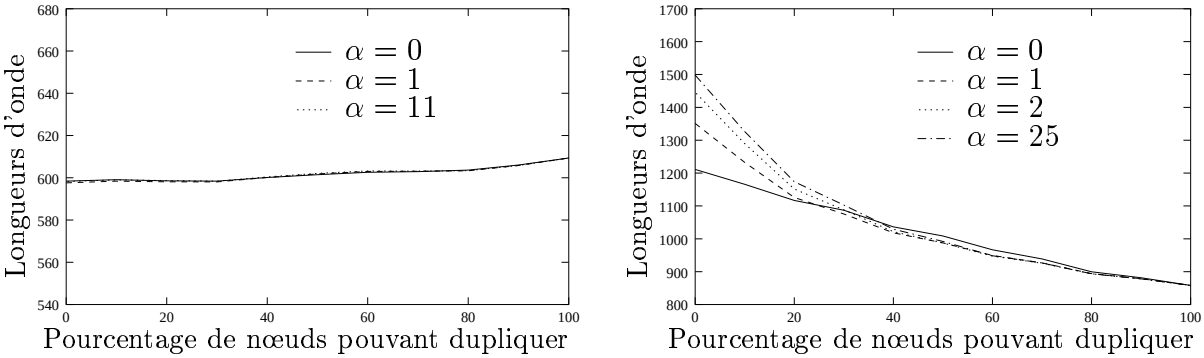


Figure 2.24 – *Nombre de longueurs d'onde utilisées, sur les réseaux Abilène (à gauche) et Renater (à droite).*

Bien que cela dépende de la valeur β que l'administrateur a choisi, il semble que la valeur $\alpha = 1$ offre de bonnes performances sur Abilène et sur Renater. C'est cette valeur que nous avons choisi par la suite pour la comparaison avec les autres heuristiques.

Complexité structurelle des schémas

La complexité structurelle d'un schéma fait référence à la complexité ajoutée par ce schéma au niveau de la conception.

Il est très facile d'adapter une heuristique $\mathcal{H}$ selon un schéma de post-traitement. Dans la plupart des cas, l'adaptation par post-traitement est même indépendante de l'heuristique $\mathcal{H}$. La complexité des adaptations selon un schéma de post-traitement est en général faible.

Le schéma par changement de topologie permet d'adapter une heuristique $\mathcal{H}$ avec un minimum de modifications. Cependant, la complexité de l'adaptation réside dans la façon dont la topologie est changée. Lorsque la topologie est changée plusieurs fois (c'est le cas avec l'heuristique IH), la complexité de l'adaptation en souffre.

Le schéma par extension nécessite la modification complète de l'heuristique $\mathcal{H}$. D'une part, la modification n'est pas toujours aisée, certaines heuristiques s'adaptant mal aux contraintes des réseaux tout optique. D'autre part, certaines heuristiques (comme CTH) présentent des comportements particuliers en présence de ces contraintes, ce qui rend leur développement et leur analyse plus complexes.

Nombre de canaux de longueurs d'onde

Le nombre de canaux de longueurs d'onde utilisés par les forêts construites par toutes les heuristiques diminue quand la proportion de nœuds pouvant dupliquer augmente. En effet, quand beaucoup de nœuds sont incapables de dupliquer, il est nécessaire d'utiliser un grand nombre de canaux pour les contourner.

Le nombre de canaux de longueurs d'onde utilisés par les heuristiques de post-traitement (RRTS, RRTA et PP) est présenté sur la figure 2.25. L'heuristique RRTA nécessite toujours moins de canaux que l'heuristique RRTS puisque le rattachement des nœuds se fait toujours à un nœud plus proche. Quand tous les nœuds peuvent dupliquer, les heuristiques RRTS et RRTA construisent une forêt d'arbres disjoints (dont la racine est de degré 1 afin de n'utiliser qu'un seul transmetteur) dont l'union est un arbre des plus courts chemins (cela est dû à notre choix d'heuristique $\mathcal{H}$), ce qui explique que les forêts construites nécessitent le même nombre de canaux. Quand tous les nœuds peuvent dupliquer, l'heuristique MO construit une forêt d'arbres disjoints dont l'union est un arbre approché de Steiner, ce qui explique qu'elle utilise moins de canaux. L'heuristique PP construit initialement une forêt d'arbres disjoints dont l'union est un arbre des plus courts chemins, qu'elle cherche ensuite à améliorer. Cette amélioration réduit significativement le nombre de canaux de longueurs d'onde dans Abilène et le laisse pratiquement inchangé dans Renater. On note que parfois, l'heuristique PP est parfois amenée à construire plus de canaux de longueurs d'onde que l'heuristique RRTS : cela est dû au fait que le critère d'amélioration est multiple. Il vise à la fois la réduction du nombre de canaux, de transmetteurs et de longueurs d'onde.

Le nombre de canaux de longueurs d'onde utilisés par les forêts construites par les heuristiques de changement de topologie (DH et IH) est présenté sur la figure 2.26. Rappelons que la borne α de l'heuristique DH est fixée à 1. Dans Abilène, l'heuristique DH se comporte presque comme l'heuristique MO. Dans Renater, cette borne α permet à l'heuristique DH d'utiliser moins de canaux de longueur d'onde que l'heuristique MO dès lors que plus de 20 % des nœuds peuvent dupliquer. Par rapport à l'heuristique DH, l'heuristique IH requiert plus de canaux de longueurs d'onde dans Abilène et autant dans Renater.

Le nombre de canaux de longueurs d'onde utilisés par les forêts construites par les heuristiques d'extension (MO, MKH et CTH) est présenté sur la figure 2.27. On remarque tout

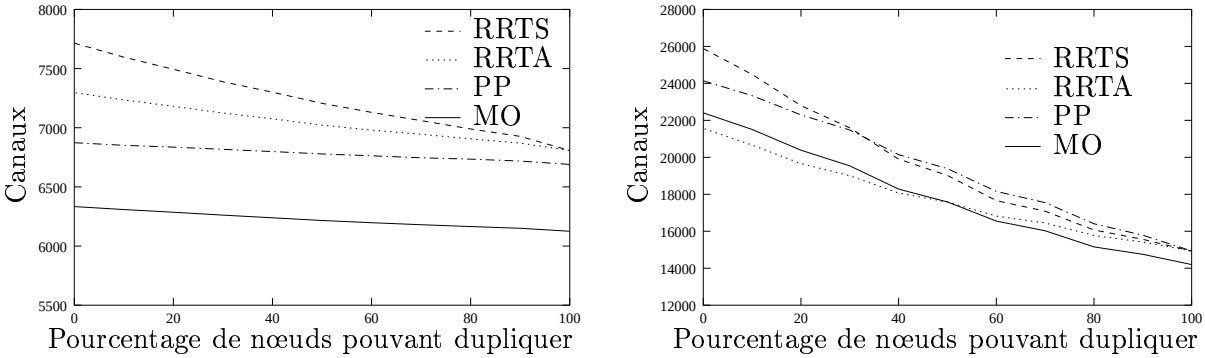


Figure 2.25 – *Nombre de canaux de longueurs d'onde utilisés, sur les réseaux Abilène (à gauche) et Renater (à droite).*

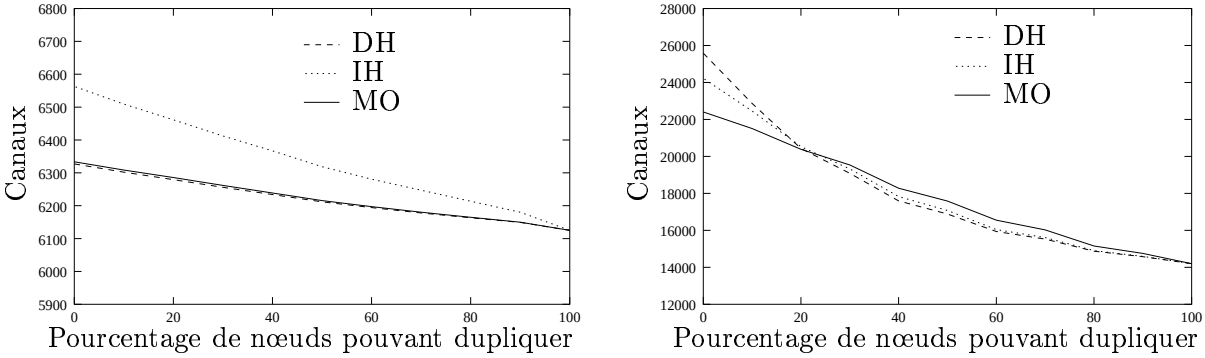


Figure 2.26 – *Nombre de canaux de longueurs d'onde utilisés, sur les réseaux Abilène (à gauche) et Renater (à droite).*

d'abord que l'heuristique CTH nécessite beaucoup moins de canaux que les heuristiques MKH et MO. En effet, l'heuristique CTH basée sur des 1-distances est plus efficace, sur les simulations que nous avons effectuées, que les heuristiques basées sur des 0-distances. Nous pensons que les heuristiques de la famille des k-distances fournissent des rapports d'approximation très faibles pour le problème de Steiner. Dans un petit réseau comme Abilène, les forêts de l'heuristique MKH nécessitent plus de canaux de longueurs d'onde que celles de l'heuristique MO à cause principalement du rattachement des arbres à la source (et du scindage qui est souvent associé). Dans un grand réseau comme Renater, les forêts de l'heuristique MKH nécessitent moins de canaux de longueurs d'onde que celles de l'heuristique MO car l'heuristique MKH compare plus de chemins que l'heuristique MO pendant la phase de construction. Par conséquent, l'heuristique MKH sélectionne des chemins plus courts.

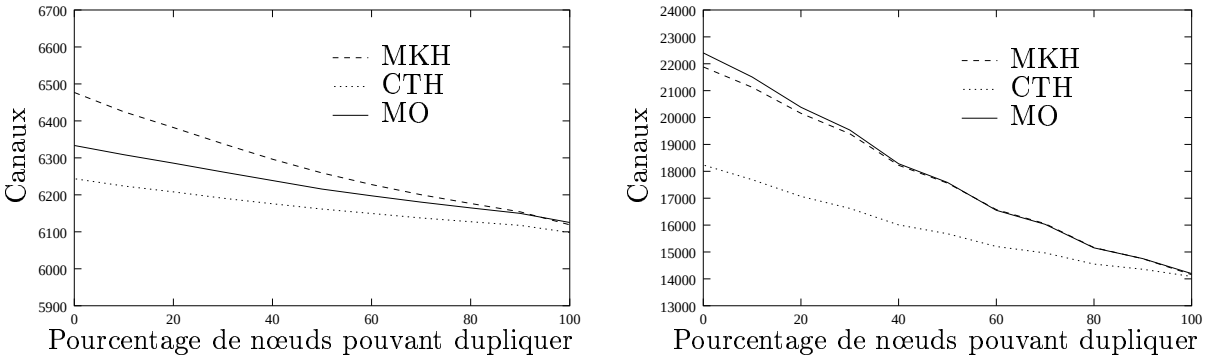


Figure 2.27 – *Nombre de canaux de longueurs d'onde utilisés, sur les réseaux Abilène (à gauche) et Renater (à droite).*

Nombre de transmetteurs

Le nombre de transmetteurs utilisés par les forêts construites par les heuristiques de post-traitement (RRTS, RRTA et PP) est présenté sur la figure 2.28. Les heuristiques RRTS et RRTA construisent des forêts qui nécessitent autant de transmetteurs. En effet, la seule différence entre ces heuristiques est dans le nœud de rattachement, c'est-à-dire dans le nœud où se situent les transmetteurs. En réduisant le nombre d'arbres, l'heuristique PP construit des forêts nécessitant beaucoup moins de transmetteurs que l'heuristique RRTS. Finalement, l'heuristique MO qui tend à construire très peu d'arbres (en étendant autant que possible l'arbre courant) nécessite elle aussi très peu de transmetteurs.

Le nombre de transmetteurs utilisés par les forêts construites par les heuristiques de changement de topologie (DH et IH) est présenté sur la figure 2.29. Comme les heuristiques DH et MO se comportent presque de la même façon sur Abilène pour la valeur de α sélectionnée, ces deux heuristiques nécessitent autant de transmetteurs. L'heuristique IH construit des arbres nécessitant plus de transmetteurs, même dans le réseau Renater où elle construisait des arbres nécessitant moins de canaux de longueurs d'onde. Sur Renater, l'heuristique DH a des performances intermédiaires entre l'heuristique MO et l'heuristique IH.

Le nombre de transmetteurs utilisés par les forêts construites par les heuristiques d'extension (MKH, CTH et MO) est présenté sur la figure 2.30. Les heuristiques MKH et MO uti-

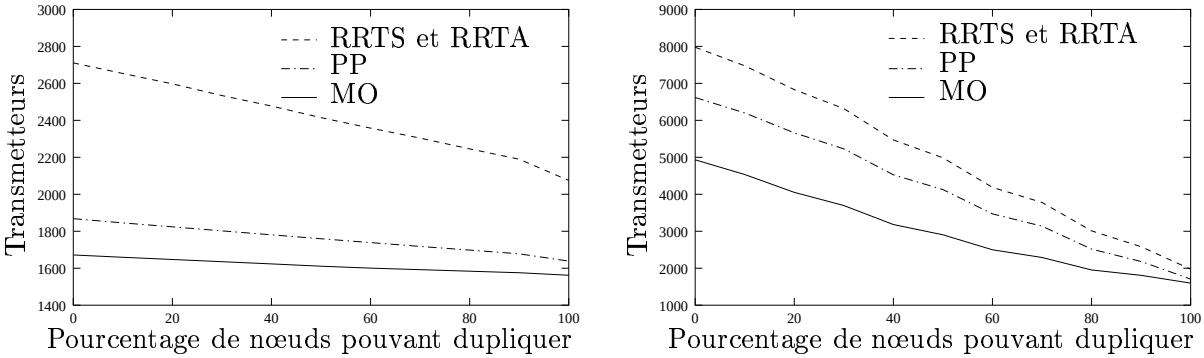


Figure 2.28 – *Nombre de transmetteurs utilisés, sur les réseaux Abilène (à gauche) et Renater (à droite).*

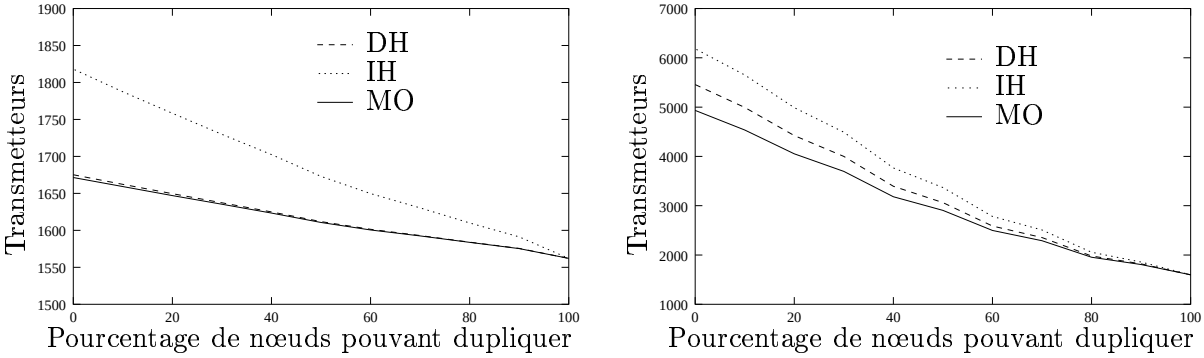


Figure 2.29 – *Nombre de transmetteurs utilisés, sur les réseaux Abilène (à gauche) et Renater (à droite).*

lisent un nombre voisin de transmetteurs puisqu'elles construisent un nombre voisin d'arbres. L'écart (important surtout pour Abilène) est dû au rattachement des arbres dans MKH. L'heuristique CTH construit beaucoup moins d'arbres que les deux heuristiques précédentes car elle est capable d'étendre l'arbre courant dans des cas où les heuristiques MO et MKH en sont incapables. Le très faible nombre de transmetteurs nécessités par l'heuristique CTH dans Renater montre que cette caractéristique de l'heuristique CTH est très importante.

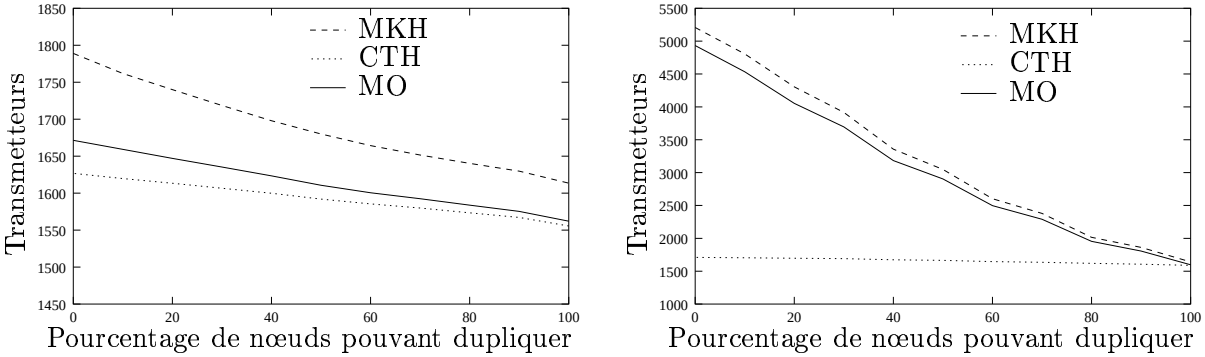


Figure 2.30 – *Nombre de transmetteurs utilisés, sur les réseaux Abilène (à gauche) et Renater (à droite).*

Nombre de longueurs d'onde

Le nombre de longueurs d'onde est une métrique difficile à analyser. En effet, elle dépend de la façon dont les arbres sont créés et n'est significative que lorsque beaucoup de groupes sont présents. En effet, pour avoir une analyse complète, il faudrait être en mesure de déduire l'arête la plus chargée en fonction de la topologie, des groupes et de l'algorithme de construction.

Le nombre de longueurs d'onde utilisées pour les forêts construites par les heuristiques de post-traitement (RRTS, RRTA et PP) est présenté sur la figure 2.31. Quand tous les nœuds peuvent dupliquer, les heuristiques RRTS et RRTA construisent exactement les mêmes forêts et nécessitent donc autant de longueurs d'onde. Dans les autres cas, les forêts de l'heuristique RRTS nécessitent plus de longueurs d'onde que celles de l'heuristique RRTA puisque le rattachement des nœuds à la source emprunte plusieurs fois les mêmes arêtes. Pour cette métrique, les performances de l'heuristique PP sont meilleures que celles de l'heuristique RRTS. En effet, l'heuristique PP est basée sur l'heuristique RRTS, optimisée *a posteriori*. Le nombre de longueurs d'onde que nécessitent les forêts de l'heuristique MO est très faible car cette dernière construit des forêts souvent formées de peu d'arbres.

Le nombre de longueurs d'onde utilisées pour les forêts construites par les heuristiques de changement de topologie (DH et IH) est présenté sur la figure 2.32. Sur Abilène, les heuristiques DH et MO se comportent de manière similaire. Sur Renater, les forêts construites par l'heuristique DH utilisent un peu plus de longueurs d'onde que celles construites par l'heuristique MO. Toutefois, la différence est relativement faible. On peut en conclure que la modification de topologie de l'heuristique DH est performante. Sur Abilène, l'heuristique IH construit des forêts nécessitant un peu plus de longueurs d'onde que l'heuristique MO. Par

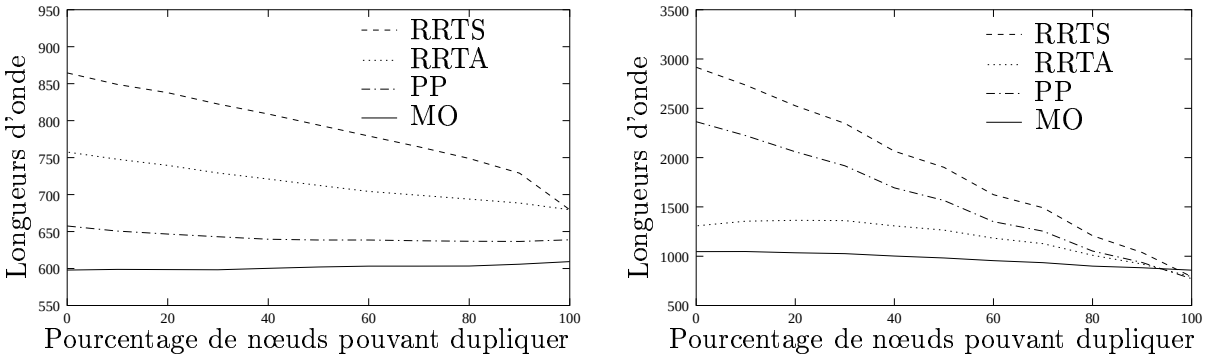


Figure 2.31 – *Nombre de longueurs d'onde utilisées, sur les réseaux Abilène (à gauche) et Renater (à droite).*

contre, l'augmentation du nombre de longueurs d'onde devient significative sur Renater. La modification de la topologie de l'heuristique IH n'est pas adaptée aux grands réseaux.

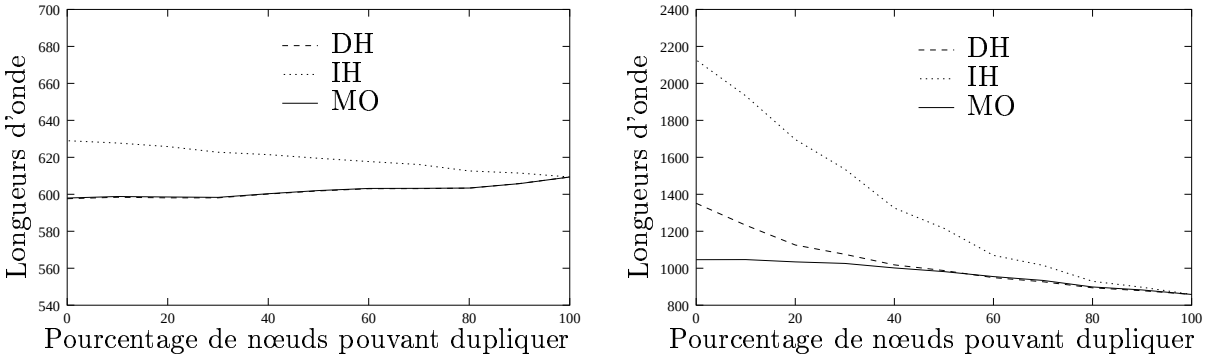


Figure 2.32 – *Nombre de longueurs d'onde utilisées, sur les réseaux Abilène (à gauche) et Renater (à droite).*

Le nombre de longueurs d'onde utilisées pour les heuristiques d'extension (MKH, CTH et MO) est présenté sur la figure 2.33. Les forêts construites par l'heuristique CTH nécessitent moins de longueurs d'onde que l'heuristique MO car elles sont de plus faible coût et sont composées de moins d'arbres. L'heuristique MKH est plus performante que l'heuristique CTH sur le réseau Abilène mais moins performante que l'heuristique MO sur le réseau Renater. Cette différence vient du fait que les heuristiques MO et CTH sont toutes deux basées sur l'algorithme de Prim alors que l'heuristique MKH est basée sur l'algorithme de Kruskal. Sur de petit réseaux comme Abilène, les heuristiques MO et CTH vont construire des forêts utilisant plusieurs fois les mêmes arêtes, alors que l'heuristique MKH va mieux répartir leur utilisation.

Nombre de conversions O/E/O

Le nombre de conversions O/E/O nécessaires pour réaliser les forêts construites les heuristiques de post-traitement (RRTS, RRTA et PP) est présenté sur la figure 2.34. Les forêts construites par les heuristiques MO et PP ne nécessitent aucune conversion O/E/O, puisque

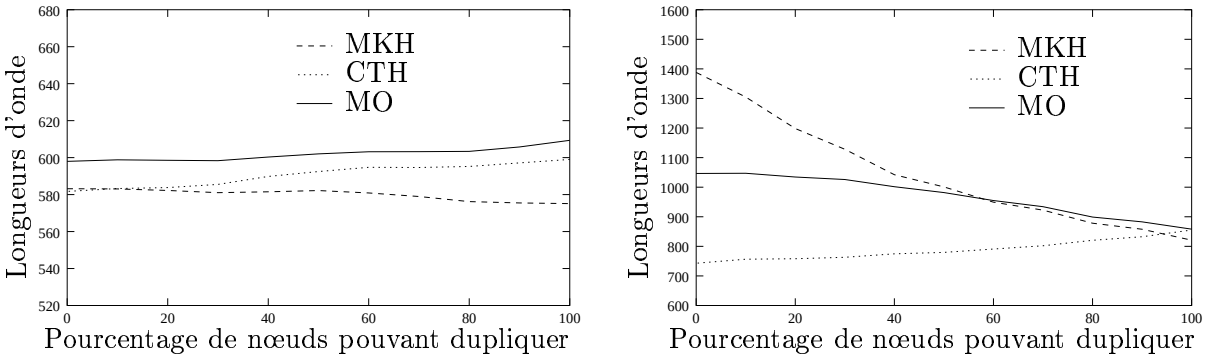


Figure 2.33 – *Nombre de longueurs d'onde utilisées, sur les réseaux Abilène (à gauche) et Renater (à droite).*

le rattachement des arbres a toujours lieu à la source du groupe. Dans RRTS, le nombre de conversions O/E/O est assez important. Pour expliquer la forme de la courbe, il faut prendre en compte deux aspects : (i) plus le pourcentage de routeurs pouvant dupliquer est important, plus les arbres de RRTS vont être rattachés à un nœud différent de la source (ce qui va augmenter le nombre de conversions O/E/O) et (ii) plus le pourcentage de routeurs pouvant dupliquer est faible, plus d'arbres seront construits donc plus de nœuds devront être rattachés (ce qui augmente le nombre de conversions potentielles). Pour RRTA, le rattachement des nœuds est très rarement fait à la source. Contrairement à RRTS, le nombre de conversions O/E/O nécessité n'est donc influencé que par le nombre d'arbres (c'est-à-dire le deuxième point).

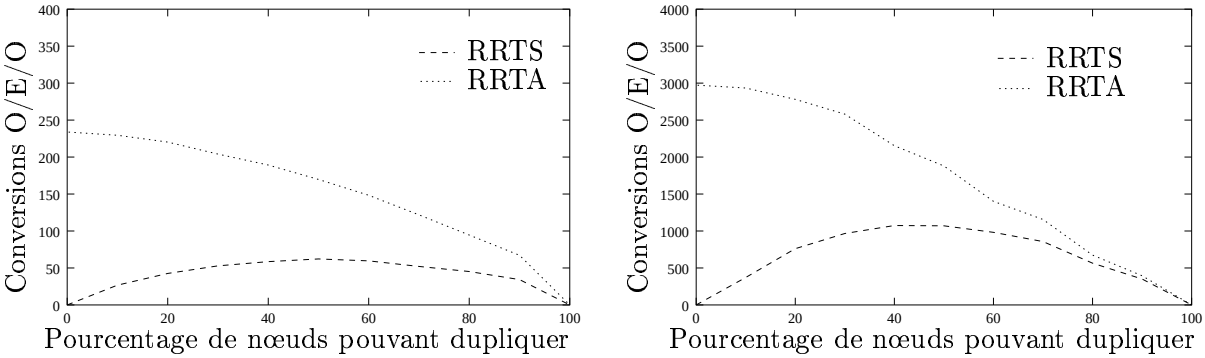


Figure 2.34 – *Nombre de conversions O/E/O utilisées, sur les réseaux Abilène (à gauche) et Renater (à droite).*

Les heuristiques par changement de topologie et par extension ne nécessitent pas de conversions O/E/O puisqu'elles rattachent toujours les arbres à la source du groupe.

Résumé de la comparaison

Le tableau 2.3 récapitule les éléments de la comparaison.

Heuristique	Complexité	Canaux	Transmetteurs	Longueurs d'onde	Conversions O/E/O
RRTS	très faible	beaucoup	beaucoup	beaucoup	peu
RRTA	très faible	beaucoup	beaucoup	peu	beaucoup
PP	faible	beaucoup	moyen	moyen	aucune
DH	faible	beaucoup	peu	peu	aucune
IH	importante	peu	beaucoup	beaucoup	aucune
MO	faible	peu	peu	peu	aucune
MKH	faible	peu	peu	peu	aucune
CTH	importante	très peu	très peu	peu	aucune

TAB. 2.3 – *Récapitulatif des performances des heuristiques.*

2.4 Structures optiques optimales

Comme nous venons de le voir, la plupart des heuristiques au problème de routage *multicast* dans les réseaux tout optique considèrent la contrainte de duplication des nœuds (*cf* la contrainte 3) comme une contrainte sur le degré des nœuds des arbres de la forêt. Cependant, ALI et DEOGUN ont montré dans [AD00] que la contrainte de degrés est trop forte et ont proposé une contrainte plus faible, la contrainte de piste optique.

Dans le paragraphe suivant, nous détaillons la contrainte de piste optique. Puis, nous étudions le problème consistant à trouver des structures optiques de coût minimum et de longueur minimum utilisant des pistes optiques [GL05].

2.4.1 La piste optique

La contrainte 3, présentée plus tôt dans ce chapitre, stipule que dans un arbre optique T, le degré sortant $d_T^+(n)$ d'un nœud n incapable de dupliquer est inférieur ou égal à 2. ALI et DEOGUN ont proposé dans [AD00] la contrainte 4, plus explicite que la précédente.

Contrainte 4 (Contrainte de piste optique). *Si un routeur n n'est pas capable de dupliquer, le nombre de signaux de sortie de ce routeur dans un arbre optique T est inférieur ou égal au nombre de signaux d'entrée du routeur : $d_T^+(n) \leq d_T^-(n)$.*

La figure 2.35(a) présente un arbre de coût minimum sans piste optique (c'est-à-dire avec la contrainte sur les degrés) et la figure 2.35(b) un arbre de coût minimum utilisant les pistes optiques (c'est-à-dire avec la contrainte de piste optique). La contrainte de degré est respectée sur l'arbre de gauche puisqu'aucun nœud n'est de degré strictement supérieur à 2. La piste optique est valide pour le routeur r_3 sur l'arbre de la figure 2.35(b) comme le montre la figure 2.36. Le nombre de canaux de longueurs d'onde utilisés par l'arbre de gauche est de 8 tandis qu'il est de 6 pour l'arbre de droite. La contrainte de piste optique permet de construire des arbres utilisant moins de canaux de longueurs d'onde.

À notre connaissance, il n'existe qu'un seul algorithme de routage qui utilise la notion de piste optique, l'heuristique FT (décrite plus loin dans ce chapitre). Une raison historique explique cette faible utilisation : dans les réseaux traditionnels, il n'y a aucun avantage à repasser plusieurs fois par les mêmes routeurs ; la spécificité des réseaux tout optique rendant

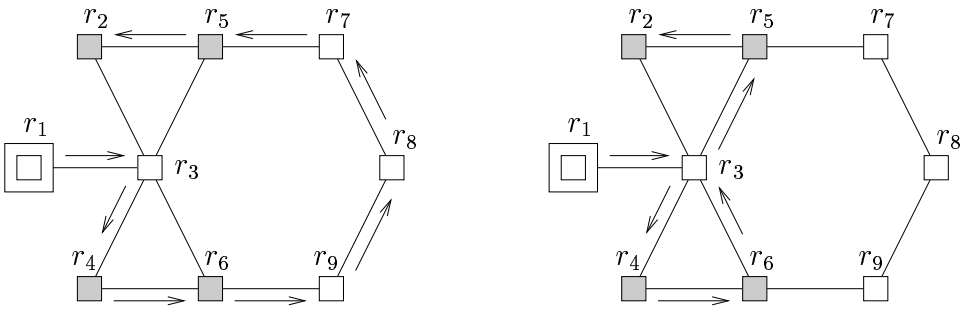


Figure 2.35 – *Un arbre optique sans piste optique est de coût supérieur à un arbre optique utilisant les pistes optiques.*

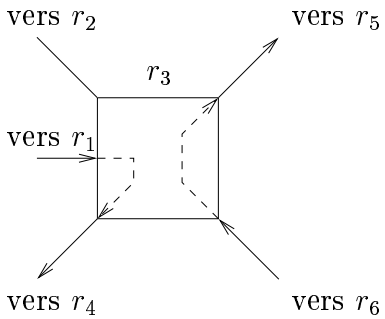


Figure 2.36 – *Le routeur r_3 de la figure 2.35(b) utilise bien une piste optique.*

cette possibilité intéressante n'a pas été remarquée de prime abord. La piste optique n'est intéressante que si le temps de copie d'un paquet est beaucoup plus grand que les temps de traitement et de propagation. Comme c'est le cas dans les réseaux tout optique à capacité de duplication limitée, il est toujours possible d'utiliser des pistes optiques dans les routeurs tout optique. Outre le fait qu'un arbre optique utilisant des pistes optiques soit moins coûteux qu'un arbre optique sans piste optique, une communication *multicast* peut être réalisée avec un unique arbre optique, plutôt qu'avec une forêt optique.

À présent, nous allons donner une définition formelle d'une instance de communication *multicast* et d'un arbre optique utilisant les pistes optiques. Ces définitions vont nous servir à étudier de manière théorique les problèmes liés à cette nouvelle contrainte.

Définition d'une instance de communication

Une instance $\mathcal{I} = (G, M, s)$ de communication *multicast* est composée des éléments suivants :

- Un graphe $G = (V, E)$, orienté, symétrique et fortement connexe. G représente le réseau, V l'ensemble des routeurs optiques et E l'ensemble des fibres optiques.
- $V = V_S \cup V_N$, où V_S est l'ensemble des routeurs pouvant dupliquer et V_N l'ensemble des routeurs ne pouvant pas dupliquer (V_S et V_N sont deux ensembles disjoints).
- $M \subset V$ est l'ensemble des membres du groupe *multicast* et $s \in V - M$ est la source du groupe.

Définition d'une structure optique utilisant les pistes optiques

Soit $\mathcal{I} = (G, M, s)$ une instance de communication *multicast*. Une solution $\mathcal{S} = (T, f)$ pour $\mathcal{I}$, c'est-à-dire un arbre optique utilisant les pistes optiques et réalisant la communication *multicast*, est composée des éléments suivants :

- Un sous-graphe $T = (V', E')$, appelé la structure de la solution $\mathcal{S}$ qui couvre $\{s\} \cup M$.
- La fonction f de E' dans $\mathcal{P}(E')$, appelée la fonction d'acheminement de la solution $\mathcal{S}$, vérifie les contraintes suivantes :
  - Acheminement local : $\forall e = (u, v) \in E', \Big(f(e) \subset \{(x, y) \in E' | x = v\}\Big)$.
  - Affectation de sorties distinctes : $\forall v \in V, \Big(\forall e_1 \in \{(x, y) \in E' | y = v\}, \Big(\forall e_2 \in \{(x, y) \in E' | y = v\}, \Big(e_1 \neq e_2 \Rightarrow (f(e_1) \cap f(e_2) = \emptyset)\Big)\Big)\Big)$. En d'autres termes, deux signaux entrant en un nœud v sur deux liens différents e_1 et e_2 doivent être acheminés sur des sorties différentes (pour respecter la contrainte 2).
  - Contrainte de piste optique : $\forall v \in V, \Big(\forall e = (u, v) \in E', \Big(|f(e)| \leq s(v)\Big)\Big)$.
- Pour tout membre $m \in M$, il existe une suite $P(m) = (e_1, \ldots, e_{l_m})$ de l_m arcs de E' qui satisfont : $e_1 = (s, x)$, $e_{l_m} = (y, m)$, pour tout $i = 1, \ldots, l_m - 1$, $e_{i+1} \in f(e_i)$ et la source s n'apparaît qu'au début de $P(m)$. Pour un même membre m, il peut y avoir plusieurs suites $P(m)$ puisque le nœud m peut être visité plusieurs fois par l'arbre optique. Nous

appelons « suite d'information » de m et nous notons $P^*(m)$ la plus courte de ces suites.

Sur l'exemple à droite de la figure 2.35, l'instance est $\mathcal{I} = (G, \{r_2, r_4, r_5, r_6\}, s = r_1)$. L'acheminement par le routeur r_3 est donné par $f((r_1, r_3)) = (r_3, r_4)$ et $f((r_6, r_3)) = (r_3, r_5)$. La suite d'information pour le membre r_4 est $P^*(r_4) = ((r_1, r_3), (r_3, r_4))$ et pour le membre r_5 est $P^*(r_5) = ((r_1, r_3), (r_3, r_4), (r_4, r_6), (r_6, r_3), (r_3, r_5))$.

Dans la suite, nous nous intéressons à des solutions $\mathcal{S}$ optimales selon certains critères.

2.4.2 Minimisation du coût

Dans ce paragraphe, nous nous intéressons à la construction d'une structure *multicast* optique de coût minimal utilisant les pistes optiques. Le coût d'une structure fera référence au nombre de canaux de longueurs d'onde qu'elle utilise.

Définition 2 (Coût d'une structure optique). *Soit $\mathcal{I} = (G = (V, E), M, s)$ une instance de communication et $\mathcal{S}_\mathcal{I} = (T = (V', E'), f)$ une solution pour $\mathcal{I}$. Le coût $\mathcal{C}(\mathcal{S}_\mathcal{I})$ de la structure optique $\mathcal{S}_\mathcal{I}$ est :*

$$\mathcal{C}(\mathcal{S}_\mathcal{I}) = |E'|.$$

Dans [AD00], Ali et Deogun ont prouvé le théorème 1 ci-dessous par réduction au problème du chemin hamiltonien. Rappelons que le problème du chemin hamiltonien revient à trouver un chemin dans un graphe passant par tous les nœuds une et une seule fois. Il s'agit d'un problème NP-complet. De plus, Ali et Deogun ont proposé l'heuristique MDT (pour *Multiple Destination Trail*) qui a un rapport d'approximation de 4. La complexité de l'heuristique MDT est $\mathcal{O}(|M|^2 \cdot |V|)$, où $|M|$ dénote le nombre de membres du groupe et $|V|$ le nombre de nœuds du graphe.

L'heuristique MDT consiste à construire un arbre de faible coût et à effectuer un parcours Eulérien de cet arbre. Toutes ses arêtes sont parcourues deux fois, une fois dans chaque sens. Les solutions trouvées par l'heuristique MDT sont donc de longs chemins.

Théorème 1. *Soit $\mathcal{I} = (G, M, s)$ une instance de communication. Trouver une solution $\mathcal{S}^*_\mathcal{I}$ de coût minimal pour $\mathcal{I}$ est un problème NP-difficile lorsqu'aucun nœud ne peut dupliquer.*

On notera que dans le cas où tous les nœuds peuvent dupliquer, le problème est identique au problème de Steiner.

On peut montrer simplement que l'heuristique MDT possède un rapport d'approximation de 2ρ, où ρ représente le facteur d'approximation d'une heuristique au problème de Steiner. Cette bonne approximabilité montre que le problème de trouver une structure optique de coût minimum et le problème de Steiner sont fondamentalement proches.

Afin d'étudier l'impact des pistes optiques, nous avons effectué des simulations de l'heuristique MO en autorisant ou en interdisant le retour à un nœud déjà utilisé. Nous avons appelé SOCM (pour Structure Optique de Coût Minimum) l'heuristique autorisant ces pistes optiques. Cette heuristique est décrite par l'algorithme 3. On notera sa ressemblance avec l'heuristique MO décrite par l'algorithme 2. La complexité de l'heuristique SOCM est la même que celle de l'heuristique MO, à savoir $\mathcal{O}(|M| \cdot |E| \cdot \log |V|)$, où $|M|$ dénote le nombre de membres du groupe, $|E|$ le nombre d'arêtes du graphe et $|V|$ le nombre de nœuds du graphe. Pour nos simulations, nous reprenons les mêmes paramètres de simulation que dans

la partie 2.3.4, sur le réseau Renater. Une comparaison entre les heuristiques SOCM et MO sur d'autres paramètres de simulation se trouve dans [GZ04].

Rappelons que la taille des groupes est choisie uniformément entre 2 et 25 (le nombre de nœuds de Renater). En moyenne, les groupes sont donc relativement denses. Ce choix permet de mieux dicerner les différences entre SOCM et MO. En effet, c'est lorsque MO est incapable d'étendre les arbres que les différences sont notables avec SOCM. Ce comportement n'est obtenu que très rarement lorsque les groupes sont très petits, l'extension de l'unique arbre existant étant souvent possible (au prix d'une augmentation d'une légère augmentation du nombre de canaux de longueurs d'onde).

Pré-requis : G un réseau, s la source du groupe et M l'ensemble des membres du groupe
Retourne : F est une forêt optique

$F \leftarrow \emptyset$
tantque $M \neq \emptyset$ **faire**
  $T \leftarrow$ un arbre vide enraciné en s
  faire
    $(p, m) \leftarrow$ le plus court chemin dans G de l'arbre T à un routeur m de M, tel que p parte d'un nœud de T dont la capacité de duplication n'est pas épuisée et tel que p n'emprunte aucune arête de T
    si un tel chemin p existe **alors**
      $T \leftarrow T \cup \{p\}$
      $M \leftarrow M \backslash \{m\}$
    fin si
  jusqu'à ce que il n'existe plus de tel chemin p
  $F \leftarrow F \cup T$
fin tantque

Algorithme 3: L'heuristique SOCM.

Dans la plupart des cas, l'heuristique SOCM construit des arbres ayant moins de canaux que l'heuristique MO. La figure 2.37 présente l'évolution du nombre de canaux de longueurs d'onde utilisés par les heuristiques SOCM et MO, en fonction du pourcentage de nœuds pouvant dupliquer. Quand tous les nœuds peuvent dupliquer, le comportement des deux heuristiques est le même puisque ni la contrainte de duplication, ni les pistes optiques ne s'appliquent. Sur Renater, quand moins de 20 % des nœuds peuvent dupliquer, l'heuristique SOCM construit des forêts utilisant plus de canaux que celles construites par l'heuristique MO. Ce comportement s'inverse dès lors que plus de 20 % des nœuds peuvent dupliquer. Dans les cas très contraints (moins de 20 % des nœuds peuvent dupliquer), SOCM construit peu d'arbres, mais ils très longs. MO, incapable d'étendre les arbres autant que SOCM, construit beaucoup d'arbres. Il s'avère que ces arbres sont plus courts que ceux de SOCM qui étend à tout prix.

La figure 2.38 présente l'évolution du nombre de transmetteurs optiques utilisés par les forêts construites par les heuristiques SOCM et MO en fonction du pourcentage de nœuds

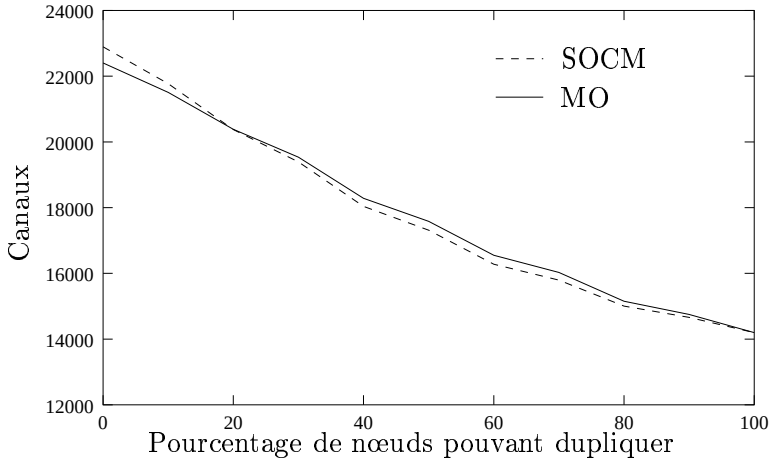


Figure 2.37 – *Nombre de canaux de longueurs d'onde utilisés sur le réseau Renater.*

ne pouvant pas dupliquer. Sur Renater, l'utilisation de l'heuristique SOCM réduit d'environ 20 % le besoin en transmetteurs, ce qui est un gain très important. SOCM construit bien moins d'arbres que MO.

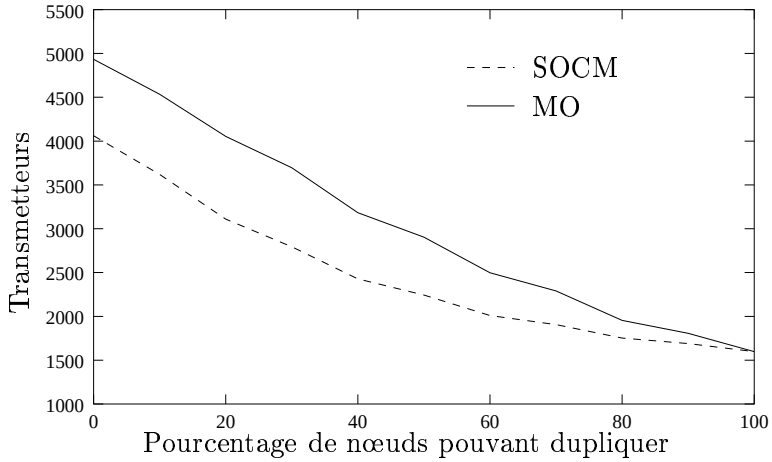


Figure 2.38 – *Nombre de transmetteurs utilisés sur le réseau Renater.*

La figure 2.39 présente l'évolution du nombre de longueurs d'onde utilisées par les forêts construites par les heuristiques SOCM et MO en fonction du pourcentage de nœuds ne pouvant pas dupliquer. Sur Renater, les forêts construites par l'heuristique SOCM utilisent moins de longueurs d'onde que celles construites par l'heuristique MO, principalement car SOCM construit peu d'arbres de faible coût.

2.4.3 Minimisation de la longueur maximale

Dans ce paragraphe, nous nous intéressons à la construction d'une structure optique dont chaque chemin utilise peu d'amplificateurs. En supposant que le nombre d'amplificateurs $NB.amplificateurs(p)$ sur un chemin p est proportionnel à la longueur de ce chemin, nous

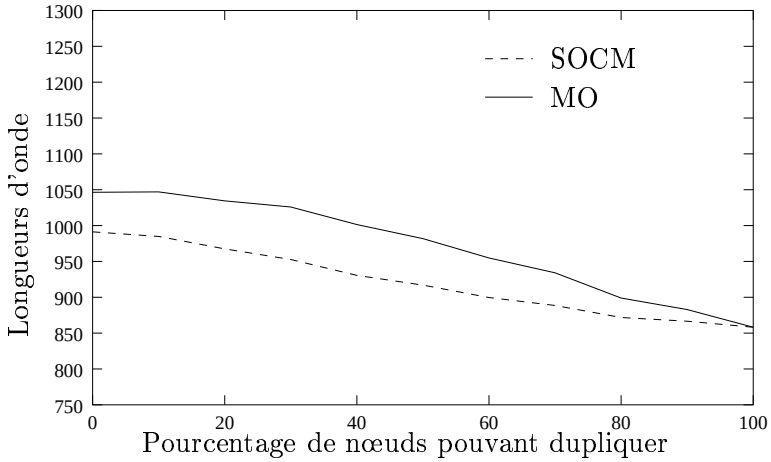


Figure 2.39 – *Nombre de longueurs d'onde utilisées sur le réseau Renater.*

nous proposons d'étudier les structures optiques dont la longueur maximale est minimale. Bien sûr, nous autorisons les pistes optiques.

Définition 3 (Longueur d'une structure optique). *Soit $\mathcal{I} = (G = (V, E), M, s)$ une instance de communication et $\mathcal{S}_\mathcal{I} = (T, f)$ une solution pour $\mathcal{I}$. La longueur $\mathcal{L}(\mathcal{S}_\mathcal{I})$ de la structure optique $\mathcal{S}_\mathcal{I}$ est :*

$$\mathcal{L}(\mathcal{S}_\mathcal{I}) = \max_{x \in M} |P^*(x)|,$$

où $P^*(x)$ est la suite d'information de x.

Nous montrons d'abord en quoi les structures optiques de longueur minimale sont intéressantes d'un point de vue pratique. Puis, nous décrivons et discutons du problème de trouver une structure optique de longueur minimale.

Intérêt des structures optiques de longueur minimale

Réduire la longueur d'une structure optique permet d'une part de réduire le nombre de régénérateurs optiques et d'autre part de limiter la dispersion. Comme le nombre d'amplificateurs traversés est proportionnel à la distance parcourue par les signaux, nous assimilons la longueur de la structure au nombre maximum d'amplificateurs traversés.

Réduction du nombre de régénérateurs optiques. En parcourant les fibres optiques, les signaux sont atténués et se dispersent. Pour pouvoir atteindre leur destination dans de bonnes conditions, il est souvent nécessaire de les régénérer régulièrement. Avec la technologie actuelle, les signaux sont typiquement 1R-régénérés tous les 100 km (voir [IKT+04] ou [RFB+04]), 2R-régénérés et 3R-régénérés tous les 500 km (voir [SM00] ou [MES+03]).

Il est important de noter que seuls les liens supportant des arbres optiques dont la longueur est supérieure à 100 km (respectivement à 500 km) doivent être équipés d'amplificateurs (respectivement de répéteurs). Comme ces appareils optiques amplifient à la fois le signal et le bruit, il est important de construire des arbres optiques dont la longueur est faible.

Pour réaliser des arbres optiques valides, les algorithmes de routage doivent connaître la position des régénérateurs optiques. Plusieurs travaux de recherches concernent le placement des régénérateurs optiques dans des réseaux tout optique [LTGC94, RIM98]. Notre problème est différent : nous cherchons ici à construire des arbres optiques dont les chemins nécessitent un nombre minimum de régénérations[3]. En d'autres termes, nous cherchons à construire des structures optiques de longueur minimale. Notre étude peut faciliter celles visant à placer les régénérateurs.

Réduction de la dispersion. La dispersion stimulée de Brillouin provient de faibles variations de l'indice du milieu de propagation. Cette dispersion limite la puissance de transmission des émetteurs optiques. Si la puissance utilisée est supérieure à un seuil SBS, une grande partie de la lumière émise est renvoyée à l'émetteur. Cela a pour conséquence de saturer le récepteur associé au transmetteur et de bruiter le signal. Le seuil SBS est inversement proportionnel au nombre d'amplificateurs traversés. Ainsi, les chemins utilisant beaucoup d'amplificateurs (y compris les amplificateurs à fibres dopées en erbium, qui sont les amplificateurs les plus fréquents) possèdent un seuil SBS très faible, ce qui a pour conséquence de réduire les performances du système [YZF99]. Pour limiter ce phénomène, les signaux doivent traverser peu d'amplificateurs.

La dispersion stimulée de Raman est due à l'interaction de la lumière avec des particules de la fibre qui absorbent puis ré-émettent les photons avec une fréquence qui peut varier. À cause de cette dispersion, une partie de l'énergie d'une longueur d'onde courte est transférée aux longueurs d'ondes plus longues[4]. Cette dispersion impose aussi un seuil SRS sur la puissance d'émission. Bien que ce seuil soit de l'ordre de mille fois plus grand que le seuil SBS, ce phénomène de dispersion apparaît toutefois dès lors qu'un chemin possède trois amplificateurs (voir [Gof02], pages 66–67 et 186–187). Le seuil SRS est lui aussi inversement proportionnel au nombre d'amplificateurs traversés.

Construire des structures optiques de longueur faible diminue le nombre d'amplificateurs requis et permet d'utiliser de plus grandes puissances d'émission des signaux.

Approximabilité de la longueur minimale

Problème 2 (SOLM). *Étant donnés un graphe G et une source s d'un groupe dont les membres forment l'ensemble M, le problème de la structure optique dont la longueur maximale est minimale (en abrégé SOLM) consiste à trouver une solution $\mathcal{S}_{\mathcal{I}}^*$ à l'instance $\mathcal{I} = (G, M, s)$ telle que, pour toute solution $\mathcal{S}_{\mathcal{I}}$ de $\mathcal{I}$, on ait :*

$$\mathcal{L}(\mathcal{S}_{\mathcal{I}}^*) \leq \mathcal{L}(\mathcal{S}_{\mathcal{I}}).$$

Dans ce paragraphe, nous nous intéressons à l'approximabilité de SOLM. Pour démontrer que le problème SOLM est NP-difficile, nous le réduisons au problème de couverture d'ensemble X3C. Rappelons ce problème de couverture d'ensemble.

Problème 3 (X3C). *Soit X et Y deux ensembles tels que chaque élément de Y est un*

3. Notons que la construction d'un arbre nécessitant un nombre minimum de régénérateurs est un problème de coût minimum, comme détaillé dans la partie précédente.

4. Ce phénomène est très important puisqu'il est à la base des amplificateurs à fibres dopées en erbium.

ensemble de 3 éléments de X. Le problème de couverture exacte X3C consiste à trouver un sous-ensemble Y' de Y tel que chaque élément de X est présent dans exactement un élément de Y'.

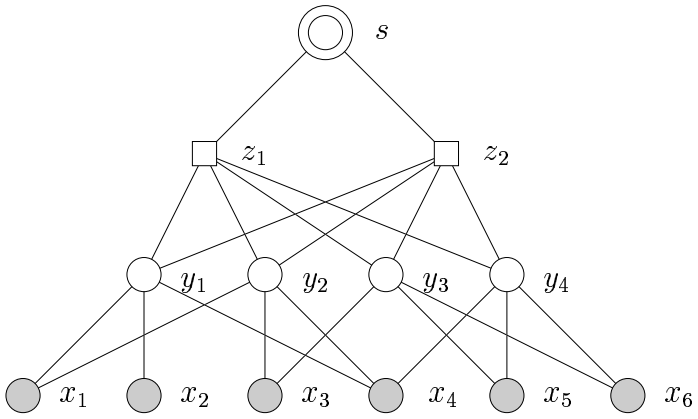
Théorème 2. *Le problème X3C est NP-difficile.*

Démonstration. La preuve de la NP-difficulté de X3C se trouve dans [GJ79]. □

Pour faire la réduction du problème SOLM au problème X3C, nous allons travailler sur des instances particulières de graphes, que nous appellerons « graphes X3C ». Un graphe X3C est un graphe orienté, symétrique et fortement connexe $G = (V, E)$ satisfaisant les propriétés suivantes :

- V est composé d'un nœud s et de trois ensembles de nœuds non vides et disjoints X, Y et Z,
- Z comporte trois fois moins d'éléments que X, $|Z| = |X|/3$,
- seuls les nœuds de Z sont incapables de dupliquer, $V_N = Z$,
- le nœud s est connecté à tous les nœuds de Z,
- chaque nœud de Z est connecté à tous les nœuds de Y,
- chaque nœud de Y est connecté à exactement trois nœuds de X,
- chaque nœud de X est connecté à au moins un nœud de Y.

La figure 2.40 montre un graphe X3C correspondant aux ensembles $X = \{x_i\}_{i=1..6}$ et $Y = \{\{x_1, x_2, x_4\}, \{x_1, x_3, x_4\}, \{x_3, x_5, x_6\}, \{x_4, x_5, x_6\}\}$. On peut noter que le sous-ensemble $Y' = \{\{x_1, x_2, x_4\}, \{x_3, x_5, x_6\}\}$ de Y couvre exactement X.



$X = \{x_1, x_2, x_3, x_4, x_5, x_6\}$
$Y = \{\{x_1, x_2, x_4\}, \{x_1, x_3, x_4\}, \{x_3, x_5, x_6\}, \{x_4, x_5, x_6\}\}$

Figure 2.40 – *Un graphe X3C et l'instance de X3C correspondante.*

Théorème 3. *Le problème SOLM est NP-difficile, même si $\mathcal{O}(n)$ nœuds peuvent dupliquer.*

Démonstration. Pour prouver que le problème SOLM est NP-difficile, nous allons montrer que le problème de décision associé est NP-complet sur les graphes X3C. Ce problème de décision peut être énoncé comme suit : étant donné un graphe X3C $G = (\{s\} \cup X \cup Y \cup Z, E)$, est-ce qu'il existe une solution $\mathcal{S}_\mathcal{I}$ de longueur $\mathcal{L}(\mathcal{S}_\mathcal{I}) = 3$ pour l'instance $\mathcal{I} = (G, X, s)$?

Notons tout d'abord que ce problème de décision est dans NP. Ensuite, observons qu'il est facile de passer d'un couple (X, Y) du problème X3C à un graphe X3C et réciproquement. Ces deux transformations peuvent être faites en temps polynomial.

Supposons que la réponse au problème de décision de X3C soit positive. Cela signifie qu'il existe un ensemble $Y' \subset Y$ qui couvre exactement X. Construisons une solution $\mathcal{S}_\mathcal{I} = (T, f)$ au problème SOLM. Dans T, s est connecté à tous les nœuds de Z car $|Y'| = |X|/3 = |Z|$. Le i-ème élément de Y' peut être connecté dans T au i-ème élément de Z car chaque nœud de Z est connecté à tous les nœuds de Y. Chaque élément de Y' est connecté dans T aux trois éléments de X correspondant. La figure 2.41 montre un exemple de construction de $\mathcal{S}_\mathcal{I}$ pour $Y' = \{y_1, y_3\}$. La longueur de $\mathcal{S}_\mathcal{I}$ est $\mathcal{L}(\mathcal{S}_\mathcal{I}) = 3$. La réponse au problème de décision de SOLM est positive.

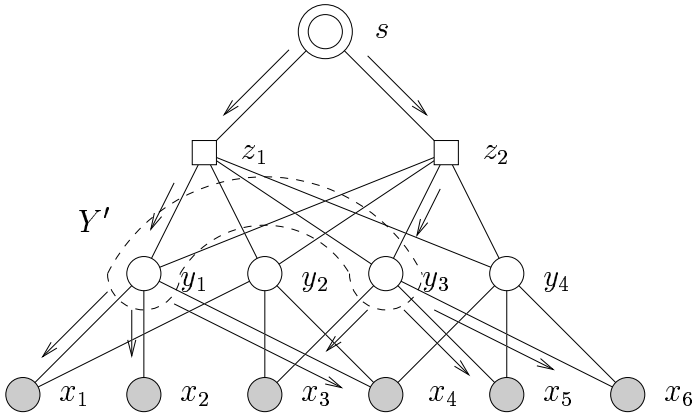


Figure 2.41 – *Construction d'une solution $\mathcal{S}_\mathcal{I}$ telle que $\mathcal{L}(\mathcal{S}_\mathcal{I}) = 3$.*

À présent, supposons que la réponse au problème de décision de SOLM soit positive. Il existe une solution $\mathcal{S}_\mathcal{I} = (T, f)$ de longueur $\mathcal{L}(\mathcal{S}_\mathcal{I}) = 3$. Dans T, la longueur de s à chaque membre x de X est inférieure ou égale à 3. Comme le plus court chemin de s à X est exactement 3, T est un arbre des plus courts chemins de s à X. Soit A l'ensemble des nœuds de Y qui sont connectés à un nœud de Z dans $\mathcal{S}_\mathcal{I}$. A est une couverture de X donc $|A| \geq |X|/3$. Puisque chaque nœud de A est connecté à un unique nœud de Z (aucun nœud de Z ne peut dupliquer), nous avons $|A| \leq |Z|$, d'où $|A| \leq |X|/3$. De ces deux inégalités, nous obtenons $|A| = |X|/3$. L'ensemble $A \subset Y$ est de taille $|X|/3$ et couvre X, c'est donc une couverture exacte de X. La réponse au problème de décision de X3C est positive.

Ainsi, le problème de décision associé à SOLM sur les graphes X3C peut être réduit en temps polynomial au problème de décision de X3C. Comme ce dernier est NP-complet, on en déduit que le problème de décision associé à SOLM est NP-complet et le théorème 3 en découle. □

À présent, nous allons montrer que le problème SOLM n'est pas $(5/3 - \varepsilon)$-approximable pour tout $\varepsilon > 0$. Pour cela, nous allons utiliser les trois lemmes 1, 2 et 3.

Lemme 1. *Soit $\mathcal{I}$ une instance du problème SOLM telle que l'instance X3C correspondante admette une solution. La longueur de la solution optimale $\mathcal{S}_\mathcal{I}^*$ pour $\mathcal{I}$ est $\mathcal{L}(\mathcal{S}_\mathcal{I}^*) = 3$.*

Démonstration. Soit $\mathcal{I} = (G = (\{s\} \cup X \cup Y \cup Z, E), M, s)$ une instance du problème SOLM telle que l'instance X3C correspondante (X, Y) admette une solution. Soit $A \subset Y$ une couverture exacte de X. On peut alors construire une solution $\mathcal{S}_\mathcal{I}^*$ de la manière suivante :

- $\mathcal{S}_\mathcal{I}^*$ contient les $|X|/3$ arcs de s aux nœuds de Z,
- chaque nœud $z \in Z$ est connecté dans $\mathcal{S}_\mathcal{I}^*$ à un nœud distinct de A, ce qui est possible car $|Z| = |X|/3 = |A|$,
- chaque nœud $a \in A$ est connecté aux trois nœuds correspondants de X.

$\mathcal{S}_\mathcal{I}^*$ est bien une solution pour $\mathcal{I}$. Par construction, $\mathcal{L}(\mathcal{S}_\mathcal{I}^*) = 3$. Cette longueur est minimale car tous les nœuds de X sont à distance 3 de s dans G. Ainsi, $\mathcal{S}_\mathcal{I}^*$ est une solution optimale. □

Lemme 2. *Soit $\mathcal{I}$ une instance du problème SOLM. Soit $\mathcal{S}_\mathcal{I}$ une solution pour $\mathcal{I}$. Soit $\mathcal{L}(\mathcal{S}_\mathcal{I}) = 3$, soit $\mathcal{L}(\mathcal{S}_\mathcal{I}) \geq 5$.*

Démonstration. Supposons que $\mathcal{L}(\mathcal{S}_\mathcal{I}) < 5$. Nous devons montrer que $\mathcal{L}(\mathcal{S}_\mathcal{I}) = 3$. Le plus court chemin de s à un nœud de X étant 3, nous avons $\mathcal{L}(\mathcal{S}_\mathcal{I}) \geq 3$. Or, $\mathcal{L}(\mathcal{S}_\mathcal{I}) \neq 4$ puisqu'il n'y a pas de chemin de longueur 4 de s à un nœud de X. □

Lemme 3. *Soit $\mathcal{I}$ une instance du problème SOLM. Soit $\mathcal{S}_\mathcal{I}$ une solution pour $\mathcal{I}$ telle que $\mathcal{L}(\mathcal{S}_\mathcal{I}) = 3$. Alors, une solution A pour l'instance X3C correspondant à $\mathcal{I}$ peut être trouvée en temps polynomial.*

Démonstration. Soit $\mathcal{S}_\mathcal{I}$ une solution pour $\mathcal{I}$ telle que $\mathcal{L}(\mathcal{S}_\mathcal{I}) = 3$. $\mathcal{S}_\mathcal{I}$ couvre tous les nœuds de X et tout chemin de longueur inférieure ou égale à 3 de s à un nœud de X dans $\mathcal{S}_\mathcal{I}$ ne passe qu'une seule fois par un nœud de Y. Soit A l'ensemble des nœuds de Y utilisés par au moins un chemin de s aux nœuds de X dans $\mathcal{S}_\mathcal{I}$. Puisqu'il y a $|X|/3$ nœuds de Z, il y a au plus $|X|/3$ nœuds dans A. Puisque A couvre X, A contient au moins $|X|/3$ nœuds. Nous avons donc $|A| = |X|/3$. A est une couverture exacte de X obtenue en temps polynomial à partir de $\mathcal{S}_\mathcal{I}$. □

Théorème 4. *Le problème SOLM n'est pas $(5/3 - \varepsilon)$-approximable, pour tout $\varepsilon > 0$.*

Démonstration. Si le problème SOLM était $(5/3 - \varepsilon)$-approximable pour tout $\varepsilon > 0$, on aurait la propriété suivante : $\exists \mathcal{A}, \left(\forall \mathcal{I}, \left(\mathcal{L}(\mathcal{S}_\mathcal{I}^\mathcal{A}) \leq (5/3 - \varepsilon)\mathcal{L}(S_\mathcal{I}^*)\right)\right)$, avec $\mathcal{A}$ un algorithme polynomial, $S_\mathcal{I}^\mathcal{A}$ la solution obtenue par l'algorithme $\mathcal{A}$ pour l'instance $\mathcal{I}$ et $\mathcal{S}_\mathcal{I}^*$ la solution optimale pour l'instance $\mathcal{I}$. Nous allons montrer que c'est impossible en raisonnant par contradiction.

Supposons qu'il existe un algorithme $\mathcal{A}$ polynomial tel que pour toute instance $\mathcal{I}$, on a $\mathcal{L}(\mathcal{S}_\mathcal{I}^\mathcal{A}) \leq (5/3 - \varepsilon)\mathcal{L}(\mathcal{S}_\mathcal{I}^*)$. L'algorithme $\mathcal{A}$ serait une k-approximation, avec $k < 5/3$. Nous allons montrer qu'un tel algorithme ne peut pas exister en restreignant l'ensemble des instances $\mathcal{I}$ possibles. Rappelons que l'algorithme doit fournir une solution k-approchée pour toutes les instances.

Soit $\mathcal{I} = (G = (\{s\} \cup X \cup Y \cup Z, E), M, s)$ une instance du problème SOLM. Si l'instance X3C correspondante (X, Y) n'admet pas de solution, $\mathcal{A}$ peut trouver en temps polynomial

une solution de longueur optimale (la longueur est 5 dans ce cas). Si l'instance X3C correspondante (X, Y) admet une solution, il existe alors une solution optimale $\mathcal{S}_{\mathcal{I}}^*$ pour $\mathcal{I}$ telle que $\mathcal{L}(\mathcal{S}_{\mathcal{I}}^*) = 3$, selon le lemme 1. L'algorithme $\mathcal{A}$ fournit une solution $\mathcal{S}_{\mathcal{I}}^{\mathcal{A}}$ telle que $\mathcal{L}(\mathcal{S}_{\mathcal{I}}^{\mathcal{A}}) \leq k\mathcal{L}(\mathcal{S}_{\mathcal{I}}^*) = 3k < 5$. Selon le lemme 2, $\mathcal{L}(\mathcal{S}_{\mathcal{I}}^{\mathcal{A}}) = 3$. Selon le lemme 3, nous pouvons extraire de $\mathcal{S}_{\mathcal{I}}^{\mathcal{A}}$ une couverture exacte A de X en temps polynomial. L'algorithme $\mathcal{A}$ a été utilisé pour résoudre X3C en temps polynomial. En effet, quand $\mathcal{L}(\mathcal{S}_{\mathcal{I}}^{\mathcal{A}}) = 3$, on peut utiliser l'algorithme $\mathcal{A}$ pour fournir une solution à l'instance X3C et quand $\mathcal{L}(\mathcal{S}_{\mathcal{I}}^*) = 5$, l'instance X3C n'a pas de solution. Il n'existe pas de tel algorithme $\mathcal{A}$, à moins que P=NP. □

Ce théorème indique qu'il n'est pas possible de concevoir un algorithme polynomial $\mathcal{A}$ dont le rapport d'approximation au problème SOLM est strictement inférieur à 5/3. En effet, sur certains graphes (comme les graphes X3C), résoudre en temps polynomial SOLM revient à résoudre en temps polynomial X3C, ce qui est impossible à moins que P=NP.

À présent, nous montrons que le problème SOLM n'est pas $(2-\varepsilon)$-approximable, pour tout $\varepsilon > 0$. Pour cela, nous étudions une nouvelle classe de graphe, que nous appelons « graphes X3C pondérés ». Pour ces graphes X3C pondérés, le problème de trouver une structure optique de longueur minimale n'est pas $(2-\varepsilon)$-approximable. Un graphe X3C pondéré est un graphe X3C valué dans lequel le coût de tous les arcs est 1, à l'exception des arcs entre s et les nœuds de Z dont le coût est a, avec $0 < a \leq 1$. Le lemme 4 sert pour la preuve.

Lemme 4. *Soit $\mathcal{I}$ une instance du problème SOLM pondéré. Soit $\mathcal{S}_{\mathcal{I}}$ une solution pour $\mathcal{I}$. Soit $\mathcal{L}(\mathcal{S}_{\mathcal{I}}) = 2 + a$, soit $\mathcal{L}(\mathcal{S}_{\mathcal{I}}) \geq 4 + a$.*

Démonstration. Ce lemme est équivalent au lemme 2 pour les graphes X3C pondérés.

Supposons que $\mathcal{L}(\mathcal{S}_{\mathcal{I}}) < 4 + a$. Nous devons montrer que $\mathcal{L}(\mathcal{S}_{\mathcal{I}}) = 2 + a$. Soit p le plus long chemin dans $\mathcal{S}_{\mathcal{I}}$, c'est-à-dire la plus longue séquence d'information de $\mathcal{S}_{\mathcal{I}}$.

- Est-ce que p peut avoir deux arcs sortants de s? Non. Par définition d'une solution $\mathcal{S}_{\mathcal{I}}$, le nœud s ne doit apparaître qu'au début de toute séquence d'information de $\mathcal{S}_{\mathcal{I}}$.
- Est-ce que p peut avoir deux arcs sortants de Z? Non. Si c'était le cas, la longueur du chemin p serait au moins $4 + a$, car il faudrait deux arcs entrants en Z. On aurait alors $\mathcal{L}(\mathcal{S}_{\mathcal{I}}) \geq 4 + a$, ce qui est contradictoire avec notre hypothèse.
- Est-ce que p peut avoir deux arcs sortants de Y? Non. Si c'était le cas, la longueur du chemin p serait au moins $4 + a$, car il faudrait deux arcs entrants en Y. On aurait alors $\mathcal{L}(\mathcal{S}_{\mathcal{I}}) \geq 4 + a$, ce qui est contradictoire avec notre hypothèse.

On aboutit à la conclusion que p ne contient qu'un seul arc sortant de s, un seul arc sortant de Z et un seul arc sortant de Y. Ainsi, la longueur de p est $2 + a$ et $\mathcal{L}(\mathcal{S}_{\mathcal{I}}) = 2 + a$. □

Théorème 5. *Le problème SOLM pondéré n'est pas $(2-\varepsilon)$-approximable, pour tout $\varepsilon > 0$.*

Démonstration. La preuve de ce théorème est très proche de celle du théorème 4. Cela consiste à appliquer les modifications suivantes:

- Dans le lemme 1, le lemme 3 et la preuve du théorème 4, les longueurs de 3 sont changées en $2 + a$ et les longueurs de 5 sont changées en $4 + a$.
- Dans la preuve du théorème 4, le lemme 2 est remplacé par le lemme 4.

– Dans la preuve du théorème 4, au lieu de $k < 5/3$, nous avons maintenant $k < (4 + a)/(2 + a)$, pour tout $0 < a \leq 1$. En faisant tendre a vers 0, nous obtenons $k < 2$.

□

Le problème SOLM pondéré n'est pas $(2 - \varepsilon)$-approximable, même si un seul nœud est incapable de dupliquer. Pour s'en convaincre, il suffit de considérer un multigraphe X3C pondéré dans lequel tous les nœuds de Z sont fusionnés en un seul nœud $\bar{z}$. Dans ce multigraphe, il y a bien $|X|/3$ arcs de s à $\bar{z}$ et $|X|/3$ arcs entre $\bar{z}$ et chaque nœud y de Y. La preuve du théorème 5 reste la même.

Une $\mathcal{O}(n/\log n)$-approximation lorsque tous les nœuds du graphe appartiennent au groupe

Considérons un réseau $G = (V, E)$ pour lequel tous les nœuds appartiennent au groupe $(V = \{s\} \cup M)$. Posons $n = |V|$ et $k = |V_S|$ le nombre de nœuds pouvant dupliquer. Posons Δ le degré maximal des nœuds pouvant dupliquer.

On peut construire un arbre T couvrant G. T aura n nœuds et $n - 1$ arêtes. La solution $\mathcal{S}^T$ associée à T en utilisant l'heuristique MDT [AD00] est telle que $\mathcal{L}(\mathcal{S}^T) < 2(n - 1)$, soit encore $\mathcal{L}(\mathcal{S}^T) \in \mathcal{O}(n)$. En effet, les arêtes de T sont utilisées au plus deux fois, une fois dans chaque sens.

La longueur de la solution optimale $\mathcal{S}^*$ est bornée par un arbre équilibré possédant k nœuds de branchement de degré Δ. En effet, G ne contient que k nœuds de branchement. La hauteur d'un tel arbre est $\log_\Delta(k(\Delta - 1) + 1) - 1 = o(\log k)$. Il faut aussi prendre en compte les $n - k$ nœuds qui ne sont pas des nœuds de branchement. La solution la moins coûteuse est de répartir ces $n - k$ nœuds de branchement de manière équitable sur chacun des k chemins de la source aux feuilles de l'arbre. Cela augmente la hauteur de l'arbre de $(n - k)/k$. Nous avons $\mathcal{L}(\mathcal{S}^*) \in o(\log k + n/k)$.

On en déduit que $\mathcal{L}(\mathcal{S}^T)/\mathcal{L}(\mathcal{S}^*) \in O(n/(\log k + n/k))$. Comme le minimum de $\log k + n/k$ est atteint en $k = n$, nous avons une $\mathcal{O}(n/\log n)$-approximation. En reprenant l'expression de $\mathcal{L}(\mathcal{S}^*)$ faisant apparaître k, nous nous rendons compte que :

– Quand k est petit devant n, $\log k$ est négligeable devant n/k, $\mathcal{L}(\mathcal{S}^*) \in o(n)$ et nous avons une $\mathcal{O}(1)$-approximation de $\mathcal{S}^*$.

– Quand k est grand, $\mathcal{L}(\mathcal{S}^*) \in o(\log n)$ mais il semble que l'on puisse trouver une solution $\mathcal{S}$ de coût logarithmique en fonction de n puisque beaucoup de nœuds peuvent dupliquer. Si c'est le cas, nous obtiendrions une $\mathcal{O}(1)$-approximation de $\mathcal{S}^*$.

L'approximabilité de la structure optique de longueur minimale est donc un problème ouvert.

Discussion

Dans le cas où tous les nœuds peuvent dupliquer, trouver une structure optique de longueur minimum revient à trouver un arbre des plus courts chemins. Cependant, dès lors qu'un seul nœud ne peut pas dupliquer, le problème devient NP-difficile, et même non approximable avec un facteur strictement inférieur à 2. L'ajout des pistes optiques change donc fondamentalement le problème de structure de longueur minimum. Il est étonnant que cette contrainte ne complique pas beaucoup le problème de trouver des structures optiques de coût

minimum (dans le pire des cas, l'approximabilité est seulement complexifiée par un facteur 2).

2.4.4 Minimisation de la longueur maximale d'une structure de coût minimal

Étant donnés un graphe G et une source s d'un groupe dont les membres constituent un ensemble M, nous avons cherché dans le paragraphe 2.4.2 à construire une structure dont le coût était minimal. Il existe plusieurs structures dont le coût est minimal. On peut chercher à réduire le nombre d'amplificateurs utilisés sur le plus long des chemins de telles structures.

Problème 4. *Étant donnée une instance $\mathcal{I}$, notons $\mathcal{EC}_\mathcal{I}$ l'ensemble des solutions de coût minimal pour $\mathcal{I}$. Nous cherchons une solution $\mathcal{S}^*_\mathcal{I}$ de $\mathcal{EC}_\mathcal{I}$ telle que, pour toute solution $\mathcal{S}_\mathcal{I}$ de $\mathcal{EC}_\mathcal{I}$, on ait :*

$$\mathcal{L}(\mathcal{S}^*_\mathcal{I}) \leq \mathcal{L}(\mathcal{S}_\mathcal{I}).$$

2.4.5 Minimisation du coût d'une structure dont la longueur maximale est minimale

Étant donnés un graphe G et une source s d'un groupe dont les membres constituent un ensemble M, nous avons cherché dans le paragraphe 2.4.3 à construire une structure dont la longueur maximale était minimale. Il existe plusieurs structures dont la longueur maximale est minimale. On peut chercher à réduire le nombre d'amplificateurs total utilisé par une telle structure.

Problème 5. *Étant donnée une instance $\mathcal{I}$, notons $\mathcal{EL}_\mathcal{I}$ l'ensemble des solutions de longueur minimale pour $\mathcal{I}$. Nous cherchons une solution $\mathcal{S}^*_\mathcal{I}$ de $\mathcal{EL}_\mathcal{I}$ telle que, pour toute solution $\mathcal{S}_\mathcal{I}$ de $\mathcal{EL}_\mathcal{I}$, on ait :*

$$\mathcal{C}(\mathcal{S}^*_\mathcal{I}) \leq \mathcal{C}(\mathcal{S}_\mathcal{I}).$$

2.4.6 Récapitulatif

Le tableau 2.4 récapitule les résultats obtenus dans ce paragraphe.

2.5 Conclusion

Le développement d'algorithmes prenant en compte les contraintes physiques des routeurs est nécessaire pour les protocoles de routage. L'incapacité de certains routeurs à dupliquer est un problème très important pour la conception de protocoles *multicast*, puisqu'il remet en cause jusqu'à l'existence (sous certaines conditions) d'un unique arbre pouvant couvrir un groupe et complexifie les problèmes théoriques sous-jacents.

Nous avons étudié le problème de non duplication de certains routeurs d'un point de vue pratique et théorique afin de mesurer l'étendue des implications. Même si l'étude est

Problème	Sans utiliser de piste optique	En utilisant des pistes optiques
Structure dont le coût est minimum	Steiner, NP-difficile, 1.55-approximable	NP-difficile, 3.10-approximable
Structure dont la longueur maximale est minimale	arbre des plus courts chemins, polynomial	NP-difficile, non $(2-\varepsilon)$-approximable
Structure de coût minimum dont la longueur maximale est minimale	inconnu (probablement NP-difficile)	inconnu (probablement NP-difficile)
Structure de longueur maximale minimale dont le coût est minimum	inconnu	inconnu (probablement NP-difficile)

TAB. 2.4 – *Récapitulatif de la complexité des problèmes décrits.*

focalisée sur les réseaux tout optique, le problème des routeurs non-*multicast* est très fréquent, notamment dans les réseaux MPLS ou lors du déploiement incrémental de protocoles *multicast*. Nous reprendrons d'ailleurs ce dernier point plus loin dans la thèse.

Dans ce chapitre, nos contributions sont les suivantes :

- La proposition de l'heuristique CTH, très performante selon toutes nos métriques. Plus particulièrement, l'heuristique CTH offre de meilleures performances que l'heuristique MO, mais au détriment de la complexité temporelle. Toutefois, il faut mentionner que l'augmentation de la complexité temporelle n'est pas très importante dans les réseaux contraints, ce qui semble être le cas de la plupart des réseaux tout optique. En effet, la différence entre l'heuristique CTH et l'heuristique MO réside principalement dans la recherche des nœuds de branchement, devant obligatoirement être des routeurs duplicateurs. L'heuristique CTH est une bonne candidate pour remplacer l'heuristique MO quand la performance prime.
- La démonstration de la non approximabilité par une constante strictement inférieure à 2 du problème des arbres des plus courts chemins avec pistes optiques. Ce problème pratique est très difficile à résoudre d'un point de vue théorique. À ce sujet, plusieurs problèmes restent ouverts : l'approximabilité du problème par une constante, ou encore la recherche du plus court des arbres, parmi l'ensemble des arbres de coût minimum.

Références

[AD00] M. ALI et J. DEOGUN. « Cost-Effective Implementation of Multicasting in Wavelength-Routed Networks ». *IEEE/OSA Journal of Lightwave Technology, Special Issue on Optical Networks*, 18(12):1628–1638, décembre 2000.

[BFC+01] S. BIGO, Y. FRIGNAC, G. CHARLET, W. IDLER, S. BORNE, H. GROSS, R. DISCHLER, W. POEHLMANN, P. TRAN, C. SIMONNEAU, D. BAYART, G. VEITH, A. JOURDAN, et J.-P. HAMAIDE. « 10.2 Tbit/s (256 × 42.7 Gbit/s PDM/WDM) transmission over 100 km TeraLight fiber with 1.28 bit/s/Hz spec-

tral efficiency ». Dans *Optical Fiber Communications Conference (OFC) Technical Digest*, papier post-deadline 25, 2001.

[BV95] F. Bauer et A. Varma. « Degree-constrained multicasting in point-to-point networks ». Dans *IEEE INFOCOM*, volume 1, pages 369–376, avril 1995.

[Dee91] S. E. Deering. « *Multicast Routing in a Datagram Internetwork* ». Thèse de doctorat, Stanford University, décembre 1991.

[Dix03] S. Dixit. *IP over WDM: Building the Next Generation Optical Internet*. John Wiley & Sons - Interscience, Hardcover, avril 2003.

[Dou92] R. Douglas. « NP-completeness and degree restricted spanning trees ». *Discrete Mathematics*, 105:41–47, 1992.

[FKM+01] K. Fukuchi, T. Kasamatsu, M. Morie, R. Ohhira, T. Ito, K. Sekiya, D. Ogasahara, et T. Ono. « 10.92 Tb/s (273 × 40 Gb/s) Triple-Band/Ultra-Dense WDM Optical-Repeatered Transmission Experiment ». Dans *Optical Fiber Communications Conference (OFC) Technical Digest*, papier post-deadline 24, 2001.

[GJ79] M. R. Garey et D. S. Johnson. *Computers and Intractability - A guide to the theory of NP-completeness*. Freeman, W. H., 1979.

[GL05] A. Guitton et C. Laforest. « Structures optiques de longueur minimum ». Travail en cours, 2005.

[GM03] A. Guitton et R. Marie. « Nouvelles heuristiques de routage *multicast* dans les réseaux optiques basés sur WDM ». Dans *Sciences Électroniques, Techologies de l'Information et des Télécommunications (SETIT)*, mars 2003.

[GMM03] A. Guitton, R. Marie, et M. Molnár. « Adaptation d'algorithmes *multicast* aux réseaux optiques ». Dans *Rencontres Francophones sur les Aspects Algorithmiques des Télécommunications (Algotel)*, mai 2003.

[GZ04] A. Guitton et A. Zsigri. « A New Way of Doing Multicast on Wavelength-Routed Optical Networks ». Dans *IFIP Optical Network Design and Modelling (ONDM)*, février 2004.

[Gof02] D. R. Goff. *Fiber Optic Video Transmission: The Complete Guide*. Focal Press, décembre 2002.

[Hak71] S. L. Hakimi. « Steiner's problem in graphs and its implications ». *Networks*, 1:113–133, 1971.

[IKT+04] T. Inoue, Y. Kurosawa, N. Takeda, E. Shibano, H. Taga, et K. Goto. « First Transpacific Distance Transmission Experiment using Raman Assisted EDF Amplifier with 100 km Repeater Span ». Dans *European Conference on Optical Communication (ECOC)*, septembre 2004.

[Joh85] D. S. Johnson. « The NP-Completeness Column: An Ongoing Guide ». *Journal of Algorithms*, 6(3):434–451, septembre 1985.

[Kru56] J. Kruskal. « On the Shortest Spanning Tree of a Graph and the Traveling Salesman Problem ». *American Mathematical Society*, 7(1):48–50, février 1956.

[LTGC94] C.-S. Li, F.-K. Tong, C. J. Georgiou, et M. Chen. « Gain Equalization in Metropolitan and Wide Area Optical Networks using Optical Amplifiers ». Dans *IEEE Infocom*, pages 130–137, juin 1994.

[Liu02] K. H. Liu. *IP Over WDM*. John Wiley & Sons, Hardcover, novembre 2002.

[MES+03] F. Matera, V. Eramo, A. Schiffini, M. Guglielmucci, et M. Settembre. « Numerical Investigation on Design of Wide Geographical Optical-Transport Networks Based on $n\times$40-Gb/s Transmission ». *IEEE/OSA Journal of Lightwave Technology*, 21(2):456–465, février 2003.

[MM00] M. Molnár et R. Marie. « Forêts déductibles maximales ». Rapport de recherche 3974, Inria, juillet 2000.

[MM99] R. Marie et M. Molnár. « Arbre couvrant partiel pseudo-optimal pour diffusion multipoint ». Rapport de recherche 3636, Inria, mars 1999.

[MZQ98] R. Malli, X. Zhang, et C. Qiao. « Benefits of Multicasting in All-Optical Networks ». Dans *SPIE All-optical networking*, volume 3531, pages 209–220, novembre 1998.

[Mol99] M. Molnár. « Construction of More Advantageous Multicast Diffusion Trees ». Dans *International Conference on Dynamic Control and Systems (DYCONS)*, août 1999.

[Pri57] R. C. Prim. « Shortest connection networks and some generalizations ». *The Bell Systems Technical Journal*, 36(11):1389–1401, novembre 1957.

[RFB+04] C. Rasmussen, T. Fjelde, J. Bennike, F. Liu, S. Dey, B. Mikkelsen, P. Mamyshev, P. Serbe, P. van der Wagt, Y. Akasaka, D. Harris, D. Gapontsev, V. Ivshin, et P. Reeves-Hall. « DWDM 40G Transmission Over Trans-Pacific Distance (10 000 km) Using CSRZ-DPSK, Enhanced FEC, and All-Raman-Amplified 100 km UltraWave Fiber Spans ». *IEEE/OSA Journal of Lightwave Technology*, 22(1):203–207, janvier 2004.

[RFC 3471] L. Berger. « Generalized Multi-Protocol Label Switching (GMPLS) Signaling Functional Description ». RFC 3471, IETF, janvier 2003.

[RIM98] B. Ramamurthy, J. Iness, et B. Mukherjee. « Optimizing Amplifier Placements in a Multiwavelength Optical LAN/MAN: The Equally Powered-Wavelengths Case ». *IEEE/OSA Journal of Lightwave Technology*, 16(9):1560–1569, septembre 1998.

[SM00] F. Schiroli et F. Matera. « The role of the 3R optical regeneration in G.652 links ». Rapport technique D211, IST, 2000.

[SM99] L. H. Sahasrabuddhe et B. Mukherjee. « Light-Trees: Optical Multicasting for Improved Performance in Wavelength-Routed Networks ». *IEEE Communications Magazine*, 37(2):67–73, février 1999.

[SRMG01] C. Siva Ram Murthy et M. Gurusamy. *WDM Optical Networks: Concepts, Design, and Algorithms*. Prentice Hall PTR, première édition, novembre 2001.

[VPM01] G. Vareille, F. Pitel, et J. F. Marcerou. « 3 Tb/s (300 $\times$ 11.6 Gb/s) Transmission over 7 380 km using 28 nm C+L-Band with 25 GHz Channel Spacing and NRZ Format ». Dans *Optical Fiber Communications Conference (OFC) Technical Digest*, papier post-deadline 22, 2001.

[Vos90] S. Voss. « Steiner-Probleme in Graphen ». *Frankfurt/Main: Hain*, pages 179–184, 1990.

[Vos92] S. Voss. « Problems with Generalized Steiner Problems ». *Algorithmica*, 7(2,3):333–335, 1992.

[YZF99] Q. Yu, L. Zhou, et C. Fan. « Stimulated Brillouin scattering of the compensating signal in all-optical link-controlled amplifier systems ». Dans *Optical Fiber Communications Conference and International Conference on Integrated Optics and Optical Fiber Communication (OFC/IOOC)*, volume 3, pages 304–306, février 1999.

[ZGM03a] A. Zsigri, A. Guitton, et M. Molnár. « Construction of Light-trees for WDM Multicasting under Splitting Capability Constraints ». Dans *International Conference on Telecommunications (ICT)*, pages 171–175, février 2003.

[ZGM03b] A. Zsigri, A. Guitton, et M. Molnár. « Two Multicast Algorithms for Sparse Splitting Capable Networks ». Dans *IFIP Optical Network Design and Modelling (ONDM)*, volume 1, pages 3–21, février 2003.

[ZLG+02] B. Zhu, L. Leng, A. H. Gnauck, M. O. Pedersen, D. Peckham, L. E. Nelson, S. Stulz, S. Kado, L. Grüner-Nielsen, R. L. Lingle, S. Knudsen, J. Leuthold, C. Doerr, S. Chandrasekhar, G. Baynham, P. Gaarde, Y. Emori, et S. Namiki. « Transmission of 3.2 Tb/s (80 × 42.7 Gb/s) over 5200 km of UltraWave fiber with 100 km dispersion managed spans using RZ-DPSK format ». Dans *European Conference on Optical Communication (ECOC)*, papier post-deadline 4.2, septembre 2002.

[ZNS+03] B. Zhu, L. E. Nelson, S. Stulz, A. H. Gnauck, C. Doerr, J. Leuthold, L. Grüner-Nielsen, M. O. Pederson, J. Kim, R. Lingle, Y. Emori, Y. Ohki, N. Tsukiji, A. Oguri, et S. Namiki. « 6.4-Tb/s (160 × 42.7 Gb/s) transmission with 0.8 bit/s/Hz spectral efficiency over 32 × 100 km of fiber using CSRZ-DPSK format ». Dans *Optical Fiber Communications Conference (OFC)*, papier post-deadline 19, mars 2003.

[ZWQ00] X. Zhang, J. Wei, et C. Qiao. « Constrained Multicast Routing in WDM Networks with Sparse Light Splitting ». *Journal of Lightwave Technology*, 18(12):1917–1927, décembre 2000.

Chapitre 3

Routage *multicast* explicite

Le modèle de *multicast* utilisé sur Internet a été proposé par Deering, comme indiqué dans le chapitre 1. Les protocoles de routage basés sur ce modèle, appelés protocoles traditionnels, associent à chaque groupe *multicast* une adresse unique, construisent un arbre de routage et configurent les routeurs de cet arbre. Ainsi, chaque routeur contient dans sa table d'acheminement une entrée pour chacun des arbres auquel il appartient. Lorsqu'un paquet ayant pour destination un groupe *multicast* atteint un routeur, le routeur examine sa table d'acheminement, extrait la liste des interfaces sortantes pour ce paquet et envoie une copie du paquet sur chacune de ces interfaces. De plus, rappelons que chaque arbre est maintenu grâce à des messages de contrôle.

D'ici quelques années, le nombre de groupes *multicast* est susceptible de devenir très important. Les routeurs de cœur de réseau, par leur position centrale, risquent d'appartenir à un grand nombre d'arbres *multicast*. Comme chaque routeur sur un arbre doit mémoriser une entrée pour cet arbre, le nombre d'entrées dans les tables d'acheminement des routeurs de cœur de réseau risque d'être très élevé. D'une part, ce phénomène peut conduire à la saturation de la mémoire des routeurs et, d'autre part, il peut ralentir le routage, c'est-à-dire diminuer les performances globales du réseau. Cela peut être amplifié par l'augmentation du nombre de messages de contrôle nécessaires à la maintenance des arbres. Ainsi, le modèle de Deering ne passe pas à l'échelle vis-à-vis du nombre de groupes.

Dans ce chapitre, nous étudions une proposition qui tend à résoudre ce problème : le routage *multicast* explicite. Le routage *multicast* explicite est un modèle assez éloigné du modèle de Deering, puisqu'il adopte une autre vision de la gestion des groupes *multicast*, mais il lui est néanmoins complémentaire. Le routage *multicast* explicite est encore appelé modèle du *multicast* pour les petits groupes ou SGM (pour *Small Group Multicast*).

Dans le modèle SGM, aucun état de routage n'est stocké dans les routeurs intermédiaires. Cette définition que nous proposons éloigne plusieurs protocoles souvent associés au modèle

SGM, tels que Reunite [SENZ00], HBH[1] [CFD01] ou SEM [BC03]. En effet, ces protocoles stockent des états de routage dans les nœuds de branchement de l'arbre de routage. Une autre particularité des protocoles du modèle SGM est qu'ils ne se basent pas sur une adresse *multicast*. Afin d'être en mesure d'acheminer correctement les paquets aux membres du groupe, chaque paquet contient explicitement les informations concernant son routage. C'est de cette particularité que le routage *multicast* explicite tire son nom. Il est évident que ce procédé n'est pas adapté aux grands groupes, composés de beaucoup de membres, puisqu'il faudrait stocker un grand nombre d'informations de routage dans chaque paquet.

Dans ce chapitre, nous décrivons deux types de routage *multicast* explicite : le plat et l'arborescent. Dans le routage *multicast* explicite plat, l'en-tête des paquets *multicast* contient l'adresses de tous les membres. Dans le routage *multicast* explicite arborescent, l'en-tête des paquets *multicast* contient l'arbre total de routage. Le tableau 3.1 récapitule les principales différences entre le modèle de Deering et le routage *multicast* explicite.

Modèle		Adresse des membres	Arbre de routage
Modèle de Deering		implicite	implicite
Routage *multicast* explicite	plat	explicite	implicite
	arborescent	explicite	explicite

TAB. 3.1 – *Principales différences entre le modèle de Deering et le routage multicast explicite.*

3.1 Le routage *multicast* explicite plat

Le routage *multicast* explicite plat consiste à stocker la liste de tous les destinataires d'un paquet dans l'en-tête de ce paquet. Historiquement, l'approche plate est la première proposée pour le SGM.

Le protocole Xcast est une synthèse de toutes les propositions précédentes concernant le routage explicite plat. Ainsi, le protocole Xcast est devenu le standard du routage *multicast* explicite plat. À l'heure actuelle, il existe plusieurs implémentations de ce protocole (notamment pour Linux et Free BSD) et une épine dorsale Xcast existe (à laquelle l'Irisa est connectée). Dans ce paragraphe, nous introduisons le protocole Xcast, son extension le protocole Xcast+ et nous proposons le protocole GXcast.

3.1.1 Le protocole Xcast

Le protocole Xcast (pour *Explicit Multicast*) est présenté brièvement dans [BFM00] et est spécifié dans [BFI+05]. Le protocole Xcast6 pour IPv6, très similaire au protocole Xcast[2], est décrit dans [IKSK03]. Des détails techniques concernant le protocole Xcast se trouvent dans [HMB+05b] et un retour d'expérience des implémentations se trouve dans [HMB+05a].

1. Voir l'entrée marquée (1) dans le glossaire.

2. Alors que le protocole Xcast introduit un nouvel en-tête Xcast entre l'en-tête IP et l'en-tête UDP des paquets, le protocole Xcast6 utilise l'option HBH (pour *Hop-By-Hop*, voir l'entrée marquée (2) dans le glossaire) d'IPv6 pour stocker la liste des destinataires. De ce fait, le protocole Xcast6 est spécifique à IPv6.

Description de Xcast

Lors de la réception d'un paquet Xcast par un routeur r, r extrait du paquet la liste l des membres. Pour chaque membre d_i de cette liste, r recherche le routeur suivant pour d_i selon sa table de routage *unicast*. Les membres de l sont alors classifiés selon leur routeur suivant. Pour chaque routeur suivant r_j, un paquet Xcast contenant la liste des membres associés à r_j est envoyé à r_j.

Lorsque la liste des membres associés à un routeur r_j ne contient qu'un seul membre d_i, l'optimisation X2U (pour *Xcast to unicast*) est appliquée. Cette optimisation consiste à envoyer un paquet *unicast* pour d_i plutôt qu'un paquet Xcast.

L'en-tête Xcast est situé entre l'en-tête IP et l'en-tête UDP. Il est composé de deux parties : une partie de taille fixe et une partie de taille variable. La partie de taille fixe occupe 12 octets. Elle contient principalement le numéro de version du protocole Xcast, le nombre de destinataires (sur 7 bits), une somme de contrôle et des drapeaux. La partie de taille variable contient la liste des adresses des destinataires.

Xcast sur un exemple

Considérons le réseau représenté sur la figure 3.1 et le groupe g formé d'une source s et des six membres d_1, d_2, d_3, d_4, d_5 et d_6. Notons que le groupe g n'est pas identifié par une adresse de groupe *multicast*. Pour émettre des données au groupe g, la source s génère la liste des membres de g, soit $\{d_1, d_2, d_3, d_4, d_5, d_6\}$. Pour chacun des membres de la liste, s recherche le routeur suivant selon sa table *unicast*. s détecte que r_1 est le routeur suivant pour d_1, d_2, d_3, d_4, d_5 et d_6. s envoie donc un unique paquet Xcast contenant dans son en-tête la liste $\{d_1, d_2, d_3, d_4, d_5, d_6\}$ à r_1.

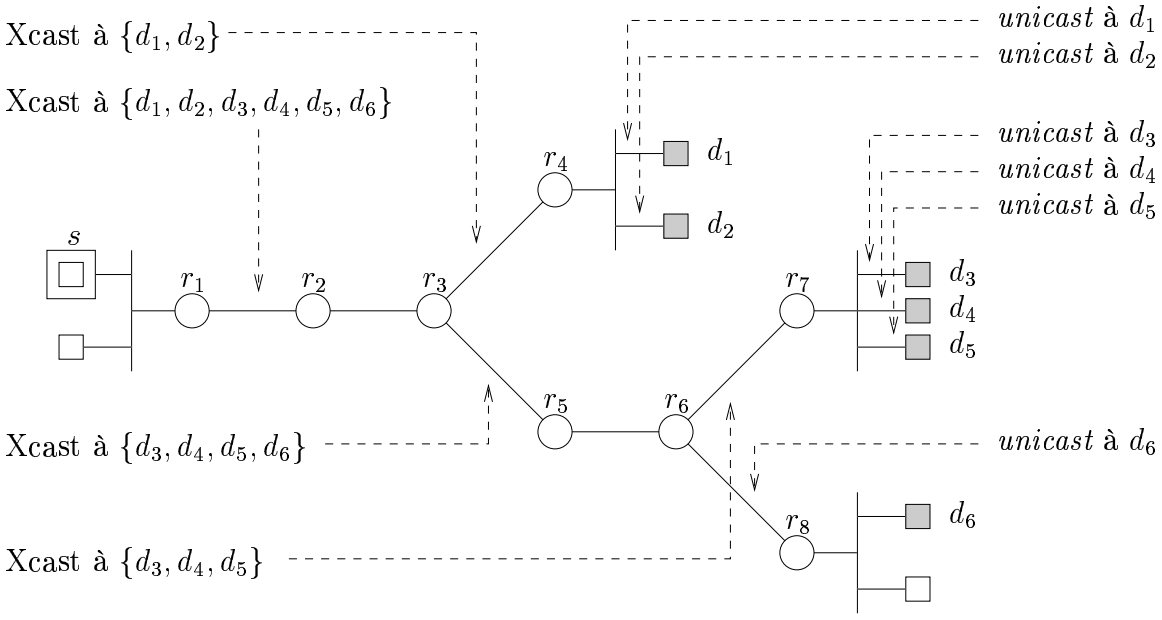


Figure 3.1 – *L'acheminement des paquets dans le protocole Xcast.*

Quand r_1 reçoit le paquet Xcast, il en extrait la liste de membres $\{d_1, d_2, d_3, d_4, d_5, d_6\}$.

Pour chaque membre de la liste, r_1 recherche le routeur suivant selon sa table *unicast* et détecte que r_2 est le routeur suivant pour tous ces membres. r_1 génère donc un unique paquet Xcast contenant dans son en-tête la liste $\{d_1, d_2, d_3, d_4, d_5, d_6\}$ et envoie ce paquet au routeur r_2. r_2 procède de la même manière et envoie un unique paquet au routeur r_3.

Quand r_3 reçoit le paquet Xcast, il en extrait la liste de membres $\{d_1, d_2, d_3, d_4, d_5, d_6\}$. Pour chaque membre de la liste, il recherche le routeur suivant selon sa table *unicast* et détecte que r_4 est le routeur suivant pour d_1 et d_2 et que r_5 est le routeur suivant pour d_3, d_4, d_5 et d_6. r_3 génère alors deux paquets Xcast. Le premier paquet Xcast contient dans son en-tête la liste $\{d_1, d_2\}$ et est envoyé au routeur r_4. Le second paquet Xcast contient dans son en-tête la liste $\{d_3, d_4, d_5, d_6\}$ et est envoyé au routeur r_5.

Quand r_4 reçoit le paquet Xcast, il en extrait la liste de membres $\{d_1, d_2\}$. Pour chaque membre de la liste, il recherche le routeur suivant selon sa table *unicast* et détecte que d_1 et d_2 sont directement connectés à lui. r_4 génère alors un paquet *unicast* au moyen de l'algorithme X2U à destination de d_1 et un paquet *unicast* à destination de d_2.

Quand d_1 (ou tout autre membre) reçoit ce paquet *unicast*, il peut en extraire les données. Le procédé est similaire pour les autres routeurs et membres.

La gestion de la dynamicité des groupes dans Xcast

Dans Xcast, la gestion des groupes est centralisée à la source. Il est alors possible pour elle de faire du contrôle d'accès. C'est un des avantages du routage *multicast* explicite par rapport au routage *multicast* traditionnel. Du point de vue de l'hôte, pour joindre ou quitter le groupe, il suffit de prévenir la source en lui envoyant un message d'adhésion ou de départ. Du point de vue de la source, les messages d'adhésion sont traités en ajoutant l'adresse de l'hôte à la liste des membres. Les messages de départ sont traités en retirant l'adresse de l'hôte de la liste des membres.

Dans [HA03], He et Ammar proposent un routage mixte combinant Xcast et PIM-SSM. Un petit groupe *multicast* est géré au moyen du protocole Xcast. Quand la taille du groupe dépasse un certain seuil t_{x2p}, le routage bascule de Xcast à PIM-SSM. Quand la taille du groupe descend sous un seuil t_{p2x}, le routage bascule de PIM-SSM à Xcast. En choisissant bien les seuils t_{x2p} et t_{p2x}, il est possible de bénéficier des avantages des deux protocoles de manière quasiment transparente. Cela confirme la complémentarité du routage *multicast* explicite et du routage *multicast* traditionnel.

La sécurité dans Xcast

Quelques éléments de sécurité du protocole Xcast sont traités dans [POS02]. Dans ce paragraphe, nous ne reprenons que les aspects de leur étude qui sont propres au routage explicite. Notre but n'est pas de détailler la sécurité de Xcast en détail, mais plutôt d'insister sur les éléments de sécurité qui sont changés par rapport au *multicast* traditionnel.

Dans le protocole Xcast, le contrôle de l'appartenance des membres au groupe est sous la seule responsabilité de la source. Elle est en mesure de faire du contrôle d'accès facilement puisqu'elle connaît tous les membres explicitement. Dans le *multicast* traditionnel, cette tâche n'est pas facile à réaliser pour les raisons suivantes :

– l'utilisation du protocole IGMP [RFC 3376] cache les membres réels, dans une certaine

mesure,

- si les adhésions sont initiées par le récepteur (ce qui est souvent le cas), le routage est mis à jour sans que la source soit prévenue,
- si la construction de l'arbre *multicast* est basée sur un routeur cœur, c'est lui qui est responsable du contrôle d'accès ; il peut moins bien gérer les destinataires que la source puisqu'il n'est pas participant du groupe.

L'anonymat est difficile à réaliser avec le protocole Xcast puisque l'ensemble des membres est inclus dans chaque paquet. Un routeur donné connaît au moins l'ensemble des membres de son sous-arbre. Ce problème n'est pas aussi marqué dans le routage *multicast* traditionnel. Dans la spécification actuelle du protocole, il existe un bit `A` de l'en-tête Xcast qui impose aux routeurs d'effacer de la liste des destinataires ceux qui ne sont pas dans le sous-arbre courant. Cela implique de recalculer la somme de contrôle Xcast des paquets en chaque nœud de branchement.

Enfin, un attaquant peut générer des faux paquets Xcast contenant un grand nombre de membres pour surcharger le réseau. Ce problème existe aussi en *multicast* traditionnel et est résolu (en partie) en testant l'interface d'entrée des paquets pour un groupe (s, g). Si l'interface d'entrée n'est pas celle qui conduit à la source s, le paquet est détruit silencieusement.

Le tableau 3.2 récapitule les différences entre le protocole Xcast et le *multicast* traditionnel au niveau de la sécurité.

contrôle d'accès	confidentialité	attaques de déni de service
très bon	très mauvaise	grandement facilitées

TAB. 3.2 – *La sécurité dans Xcast par rapport à la sécurité en multicast traditionnel.*

3.1.2 Le protocole Xcast+

Le protocole Xcast+ est décrit dans [SKPK01]. Il utilise les informations fournies par le protocole IGMP pour diminuer le nombre de destinataires par paquet.

Description de Xcast+

Le protocole IGMP place chaque réseau local sous la responsabilité d'un routeur désigné. Ce routeur mémorise un état de routage pour tous les groupes *multicast* dont la source ou au moins un membre est sous sa responsabilité. Les routeurs désignés sont les seuls routeurs possédant des états de routage *multicast* dans le protocole Xcast+. Cependant, cela ne remet pas en cause le fait que le protocole Xcast+ soit un protocole de routage *multicast* explicite : les routeurs intermédiaires (en dehors des routeurs désignés) n'ont pas d'états de routage.

Lorsque la source s émet un paquet *multicast* pour le groupe g (au moyen d'une adresse *multicast*, ce qui n'est pas le cas avec le protocole Xcast), le routeur désigné $rd(s)$ de la source intercepte ce paquet et examine une table interne associant au couple (s, g) la liste des routeurs désignés des membres. Cet algorithme est appelé M2X (pour *multicast to Xcast*). Tous les routeurs suivants traitent les paquets Xcast+ de la même manière que dans le protocole Xcast, sauf que le protocole Xcast+ n'utilise pas l'optimisation X2U.

Lorsqu'un routeur désigné $rd(d)$ d'un membre d reçoit un paquet Xcast+, il en extrait l'adresse de la source s et l'adresse du groupe g puis consulte une table interne associant à (s, g) l'ensemble de ses réseaux locaux contenant des membres pour (s, g). Il envoie alors un paquet *multicast* sur ces réseaux locaux pour le groupe (s, g). Les membres reçoivent les données de manière transparente, comme si le *multicast* traditionnel était utilisé de bout en bout. Cet algorithme est appelé X2M (pour *Xcast to multicast*).

En résumé, le protocole Xcast+ est un protocole de routage *multicast* explicite plat entre les routeurs désignés des membres.

Xcast+ sur un exemple

Considérons le réseau représenté sur la figure 3.2 et le groupe g formé d'une source s et des six membres d_1, d_2, d_3, d_4, d_5 et d_6. Pour émettre des données au groupe g, la source s crée un paquet *multicast* contenant les données et dont l'adresse de destination est l'adresse du groupe g. s envoie ce paquet sur son réseau local.

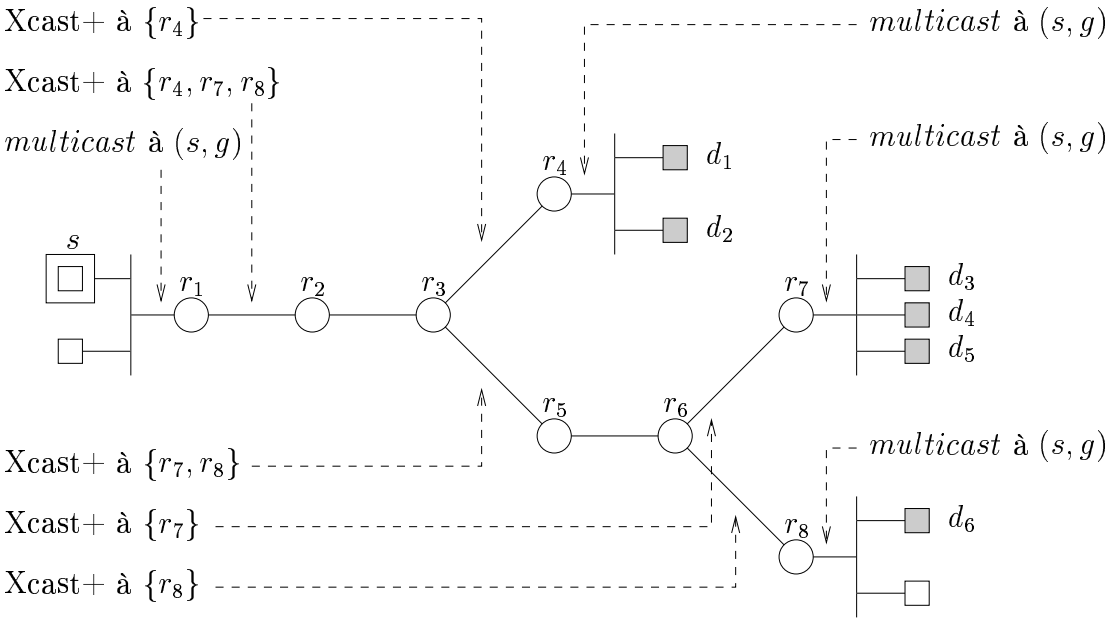


Figure 3.2 – *L'acheminement des paquets dans le protocole Xcast+.*

Quand le routeur désigné r_1 de la source s reçoit ce paquet *multicast*, il utilise une table de correspondance interne qui associe à (s, g) la liste $\{r_4, r_7, r_8\}$. Pour chaque routeur désigné de la liste, r_1 recherche le routeur suivant selon sa table *unicast* et détecte que r_2 est le routeur suivant pour tous ces membres. r_1 génère donc un unique paquet Xcast+ au moyen de l'algorithme M2X contenant dans son en-tête la liste $\{r_4, r_7, r_8\}$ et envoie ce paquet vers r_2. Quand r_2 reçoit le paquet Xcast+, il en extrait la liste de membres $\{r_4, r_7, r_8\}$. Pour chaque membre de la liste, il recherche le routeur suivant selon sa table *unicast* et détecte que r_3 est le routeur suivant pour tous ces membres. r_2 génère donc un unique paquet Xcast+ contenant dans son en-tête la liste $\{r_4, r_7, r_8\}$ et envoie ce paquet vers le routeur r_3.

Puis, r_3 reçoit le paquet Xcast+ et en transmet un à $\{r_4\}$ et un à $\{r_7, r_8\}$. Nous notons que, contrairement au protocole Xcast, le protocole Xcast+ génère des paquets ayant des listes d'un seul élément. Quand r_4 reçoit le paquet Xcast+, il en extrait la liste $\{r_4\}$. Étant dans cette liste, il transforme le paquet Xcast+ en un paquet *multicast*, au moyen de l'algorithme X2M. Au moyen d'une table de correspondance interne, (s, g) est associé aux réseaux locaux de r_4 contenant des membres. r_4 envoie un unique paquet *multicast* sur le réseau local contenant d_1 et d_2. Quand d_1 (ou tout autre membre) reçoit ce paquet *multicast* à destination du groupe (s, g), il peut en extraire les données.

La gestion de la dynamicité des groupes dans Xcast+

Dans Xcast+, la gestion des groupes est centralisée dans le routeur désigné de la source.

Du point de vue de l'hôte, le protocole IGMP est utilisé. Le routeur désigné de l'hôte envoie un message d'adhésion ou de départ en fonction du nombre de membres dont il est responsable.

Du point de vue du routeur désigné de la source, les messages d'adhésion et de départ mettent à jour la liste des routeurs désignés des membres. Le contrôle d'accès n'est plus possible : dès qu'un routeur désigné membre est dans la liste, tous les hôtes qui lui sont attachés peuvent s'abonner ou se désabonner au groupe de façon transparente pour le routeur désigné de la source. Ainsi, le contrôle d'accès est plus difficile avec le protocole Xcast+ qu'avec le protocole Xcast. Ce désavantage par rapport aux autres protocoles de routage *multicast* explicite vient de l'ajout du protocole traditionnel IGMP.

La sécurité dans Xcast+

La sécurité dans le protocole Xcast+ ne peut pas être traitée de la même manière que dans le protocole Xcast puisque le protocole IGMP est utilisée.

Le contrôle de l'appartenance des membres au groupe est plus difficile qu'avec le protocole Xcast, puisqu'il est sous la seule responsabilité du routeur désigné de la source, au lieu de la source elle-même. Par rapport au *multicast* traditionnel, cette tâche est cependant légèrement facilitée car le routeur désigné de la source connaît explicitement les routeurs désignés des membres.

Dans le protocole Xcast+, les membres n'apparaissent plus dans l'en-tête des paquets, mais on y trouve leurs routeurs désignés. De plus, les hôtes ne connaissent plus la liste des membres puisqu'ils reçoivent des paquets *multicast*.

Enfin, il est plus difficile à un attaquant de générer de faux paquets Xcast+ que de faux paquets Xcast, puisque seuls les routeurs désignés sont capables d'émettre des paquets Xcast+. L'attaquant aurait à compromettre un routeur désigné.

Le tableau 3.3 récapitule les différences entre Xcast+ et le *multicast* traditionnel au niveau de la sécurité.

contrôle d'accès	confidentialité	attaques de déni de service
assez bon	assez mauvaise	légèrement facilitées

TAB. 3.3 – *La sécurité dans Xcast+ par rapport à la sécurité en multicast traditionnel.*

3.1.3 Le protocole GXcast

Comme nous allons le voir dans ce paragraphe, les protocoles Xcast et Xcast+ sont conçus pour les petits groupes et ne supportent pas la fragmentation IP. Pour résoudre ce problème, nous avons proposé dans [BGC03, BGC04a] le protocole GXcast.

Le protocole GXcast ajoute au protocole Xcast une segmentation à la source. Une segmentation à la source peut être adaptée au protocole Xcast+ de la même manière.

Le problème de la fragmentation

Pour découper les messages trop longs, le protocole IP met en place un mécanisme appelé « la fragmentation ». Dans ce paragraphe, nous montrons pourquoi la fragmentation IP doit être interdite pour le protocole Xcast.

La fragmentation IP. Pour des raisons physiques et techniques, un lien ne peut transférer qu'un volume limité d'informations par trame. Le protocole IP dispose d'un mécanisme de fragmentation permettant de rendre cette limite transparente aux protocoles de niveaux supérieurs des stations d'extrémités de la communication. Le mécanisme de fragmentation du protocole IP découpe un paquet trop grand en plusieurs fragments. Chaque fragment est un paquet IP autonome dont la destination est celle du paquet IP initial. Chaque fragment est suffisamment petit pour être transféré sur le lien ayant requis la fragmentation. En d'autres termes, la taille de chaque fragment est inférieure ou égale à la capacité maximale du lien, nommée MTU (pour *Maximum Transmission Unit*). Le réassemblage des fragments n'a lieu qu'à la destination du paquet. Toutefois, la source d'un paquet peut marquer les paquets comme non-fragmentables. Dans ce cas, un routeur qui ne peut acheminer les paquets sans les fragmenter les détruit. L'algorithme décrivant la fragmentation des paquets IPv4 est présenté dans [RFC 791]. L'algorithme décrivant la fragmentation des paquets IPv6 est présenté dans [RFC 2460], mais il faut noter qu'IPv6 ne réalise une fragmentation qu'à la source, en tirant profit du PMTU (pour *Path MTU*). Le PMTU est le minimum du MTU des liens d'un chemin. Le PMTU est souvent bien supérieur au MTU minimum garanti par IP. L'obtention du PMTU est faite selon [RFC 1191] pour IPv4 et selon [RFC 1981] pour IPv6. Nous verrons comment nous pouvons tirer profit de la connaissance du PMTU. Notons que les informations de fragmentation sont présentes dans un en-tête optionnel pour IPv6 (voir les parties 4, 4.5 et 5 de [RFC 2460]).

La segmentation des paquets Xcast. Considérons le cas où un paquet Xcast doit être fragmenté dans un routeur IP. L'en-tête d'un paquet Xcast pouvant être grand, deux cas doivent être étudiés : soit l'en-tête complet du paquet Xcast tient dans le premier fragment, soit l'en-tête du paquet Xcast doit être réparti entre les fragments. Le cas où l'en-tête complet du paquet Xcast tient dans le premier fragment est présenté sur la figure 3.3. Supposons que le paquet initial soit découpé en trois fragments. Seul le premier fragment contient un en-tête Xcast. Il va donc être le seul fragment atteignant les membres. Les deux autres fragments n'ont pas d'en-tête Xcast. Ils seront détruits par les routeurs car le champ `protocol` de leur en-tête IP contient la valeur `Xcast` alors qu'ils n'ont pas d'en-tête Xcast. De plus, seul le début des données émises par la source va atteindre les membres.

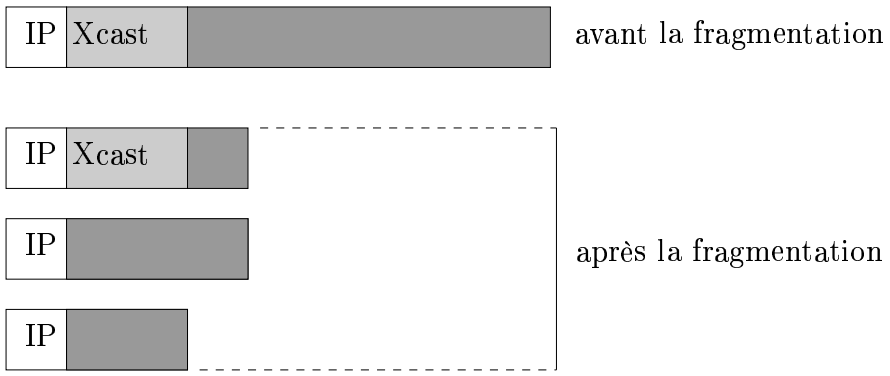


Figure 3.3 – *Premier cas : l'en-tête Xcast tient dans le premier fragment.*

Le cas où l'en-tête du paquet Xcast doit être réparti entre les fragments est présenté sur la figure 3.4. Supposons que le paquet initial soit découpé en trois fragments. Le premier fragment contient un en-tête Xcast tronqué. Il y a de grandes chances que ce fragment soit considéré comme un paquet Xcast valide, bien que cela dépende de l'implémentation des routeurs. Ainsi, le paquet va être acheminé à un sous-ensemble des membres. Toutefois, ce premier fragment ne contenant aucune donnée, il aura été acheminé inutilement. Le second fragment contient la fin de l'en-tête Xcast. Cet en-tête ne contient que des membres et n'est donc pas un en-tête Xcast valide. N'étant pas des paquets Xcast corrects, le deuxième et le troisième fragment vont être détruits par les routeurs.

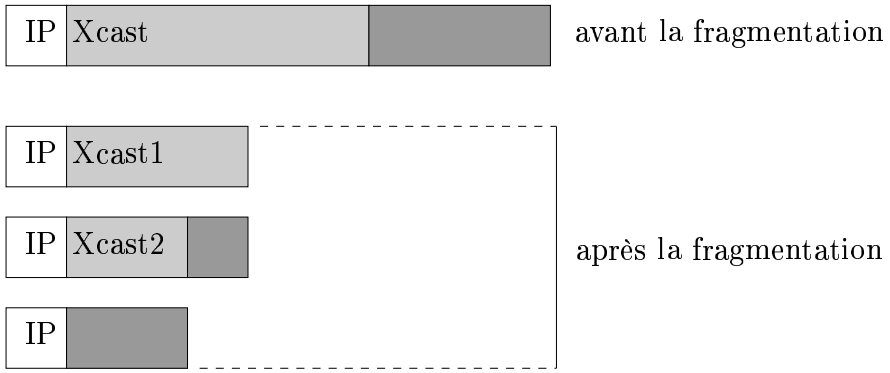


Figure 3.4 – *Second cas : l'en-tête Xcast ne tient pas dans le premier fragment.*

Nous venons de voir qu'en présence d'une fragmentation IP des paquets Xcast, les membres ne reçoivent pas l'ensemble des données envoyées par la source. Par conséquent, la fragmentation IP de paquets Xcast doit être évitée. Dans IPv4, il est possible d'interdire la fragmentation IP en utilisant le drapeau DF (pour `Don't Fragment`) de l'en-tête IP. Pour garantir que les paquets pourront être transmis jusqu'aux membres sans être fragmentatés, la source doit limiter la taille des paquets au MTU minimum garanti par IP. Pour IPv4, le MTU minimum pour n'importe quel lien est de 576 octets [RFC 791]. Pour IPv6, le MTU minimum pour n'importe quel lien est 1280 octets [RFC 2460]. Cette limitation dans la taille des paquets Xcast limite la taille de la liste de membres, et par conséquent le nombre de membres.

Comme nous allons le voir plus tard, le nombre de membres est limité pour IPv4 à 135 (ou plus précisément à 127 puisque la taille de la liste est codée sur 7 bits dans l'en-tête Xcast) et à 76 pour IPv6. Nous pensons que cette limite dans un protocole gérant des groupes dynamiques est trop restrictive. En effet, un groupe *multicast* qui est accepté pourrait être refusé *a posteriori* si la dynamique du groupe fait que le nombre de membres dépasse cette limite. Cela pose de nombreux problèmes que l'application devrait gérer, le refus systématique de l'adhésion au delà du seuil n'étant pas un comportement acceptable. Le protocole GXcast supprime cette limite.

Description du protocole GXcast

Le protocole GXcast n'est soumis à aucune limite sur le nombre de membres. Il est basé sur le protocole Xcast mais inclus un mécanisme de segmentation à la source. C'est ce mécanisme qui permet de dépasser la limite dans le nombre de membres du protocole Xcast. Bien entendu, en tant que protocole *multicast* explicite, le protocole GXcast n'est pas conçu pour des groupes de très grande taille. Toutefois, il apporte deux avantages majeurs :

- l'application n'a plus à contrôler le nombre de membres,
- le protocole GXcast permet de définir un cadre d'analyse des performances du routage explicite plat.

Le protocole GXcast est similaire au protocole Xcast, sauf qu'il fragmente les paquets à la source. Pour cela, il a besoin de connaître le MTU de l'arbre (implicite) de routage. Il peut tirer profit de l'utilisation d'une valeur de MTU dynamique obtenue au moyen des PMTU des chemins de la source aux membres (le MTU d'un arbre est égal au minimum des PMTU des chemins de cet arbre). Dans IPv4, la manière de calculer le PMTU est décrite dans [RFC 1191]. Dans IPv6, la manière de calculer le PMTU est décrite dans [RFC 1981].

Connaissant la valeur MTU du plus petit MTU des liens de l'arbre de routage (ou une valeur inférieure), la taille E d'un en-tête IP et d'un en-tête GXcast (sans la liste de membres) et la taille IP d'une adresse IP, le nombre maximum n_{max} de membres dans un paquet GXcast peut être obtenu par la formule suivante :

$$n_{max} = \lfloor \frac{MTU - E}{IP} \rfloor .$$

En effet, un paquet contenant $n_{max} + 1$ membres aurait une taille de $E + (n_{max} + 1) \cdot IP$, soit au moins $MTU + IP$ octets. Ce paquet serait de taille supérieure à MTU et pourrait être fragmenté, ce que GXcast s'interdit. Notons que cette formulation considère qu'aucune donnée n'est envoyée.

IPv4 garantissant que tout lien a un MTU supérieur ou égal à 576 octets, nous pouvons calculer la valeur minimale de n_{max}. Ayant $MTU = 576$, $E = 20 + 16$ (sans l'en-tête UDP) et $IP = 4$ pour IPv4, nous avons $n_{max} = 135$. IPv6 garantissant que tout lien a un MTU supérieur ou égal à 1280 octets, nous avons $MTU = 1280$, $E = 40 + 16$ (sans l'en-tête UDP) et $IP = 16$ pour IPv6, et donc $n_{max} = 76$.

Afin de segmenter les paquets à la source, plusieurs politiques peuvent être implémentées. Dans un premier temps, considérons une segmentation stricte. Une telle segmentation suit

le schéma suivant :

1. Étant donné n le nombre de membres actuel, d la quantité de données à envoyer et n_{max} le nombre maximum de membres par paquet pour éviter la fragmentation IP, calculer $n_p \in [1; n_{max}]$, le nombre maximum de membres par paquet GXcast.
2. Découper la liste des n membres en sous-listes contenant toutes n_p membres, sauf potentiellement la dernière.
3. Pour chacune de ces sous-listes, créer autant de paquets GXcast que nécessaires pour envoyer les d octets de données. Rappelons que la quantité de données disponibles par paquet est $MTU - E - n_p \cdot IP$.

La méthode décrite est en fait une segmentation de la liste des membres et des données : plusieurs paquets GXcast peuvent être générés afin d'atteindre tous les destinataires et de leur envoyer les d octets de données voulus.

Notons que cette segmentation stricte est propre à chaque source (et plus précisément à chaque paquet), les sources n'ont pas à se mettre d'accord avec les membres ou entre elles sur une valeur particulière de n_p. n_p est appelée le paramètre du protocole GXcast. Il influe sur les performances du protocole.

Dans [HA03], une segmentation à la source similaire à celle du protocole GXcast est décrite sous le nom Xcast à division de groupe (en anglais : *split-group Xcast*). Les auteurs n'étudient pas la manière dont les groupes sont divisés, mais plutôt l'impact des transitions de Xcast (ou GXcast) à PIM-SSM. De plus, ils n'étudient pas l'impact de la division de groupe sur les performances du protocole.

GXcast sur un exemple

Considérons le réseau représenté sur la figure 3.5 et le groupe g formé d'une source s et de six membres d_1, d_2, d_3, d_4, d_5 et d_6. Notons que le groupe g n'est pas identifié par une adresse de groupe *multicast*, mais plutôt par l'ensemble des membres qui le composent. Pour cet exemple, nous supposerons que le paramètre de GXcast, c'est-à-dire le nombre maximal de membres autorisés par paquet, vaut $n_p = 3$.

Pour émettre des données au groupe g, la source génère la liste de membres du groupe, soit $\{d_1, d_2, d_3, d_4, d_5, d_6\}$. Détectant que cette liste est plus longue que n_p, s la découpe en deux sous-listes de taille au plus n_p : $\{d_1, d_2, d_3\}$ et $\{d_4, d_5, d_6\}$. En examinant la première liste, s détecte que r_1 est le routeur suivant pour d_1, d_2 et d_3. s génère donc un paquet Xcast contenant dans son en-tête la liste $\{d_1, d_2, d_3\}$ et envoie ce paquet vers r_1. En examinant la seconde liste, s détecte que r_1 est le routeur suivant pour d_4, d_5 et d_6. s génère donc un second paquet Xcast contenant dans son en-tête la liste $\{d_4, d_5, d_6\}$ et envoie ce paquet vers r_1. L'exemple montre aussi que le protocole GXcast envoie le même contenu plusieurs fois sur certains liens.

Par la suite, chacun de ces deux paquets Xcast est traité comme un paquet Xcast quelconque. Notons que les routeurs intermédiaires ou les membres n'ont pas les moyens de détecter que le paquet original a été fragmenté (au sens GXcast et non pas IP) à la source. Pour eux, cette segmentation est transparente.

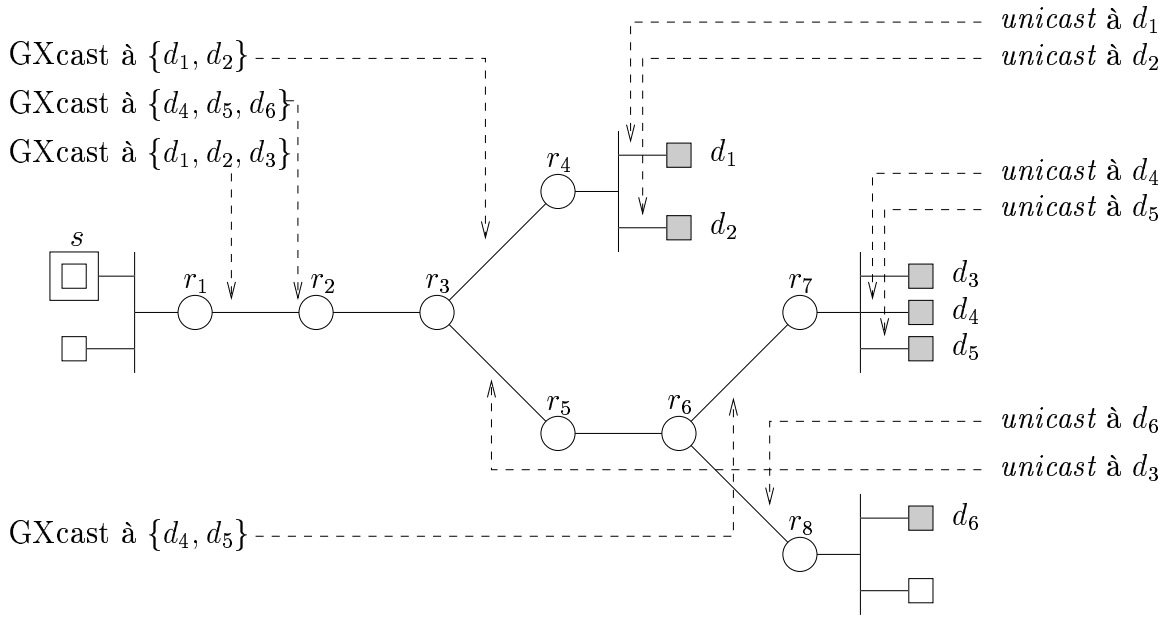


Figure 3.5 – *L'acheminement des données dans le protocole GXcast.*

La sécurité dans GXcast

La sécurité du protocole GXcast est semblable à celle du protocole Xcast. Néanmoins, l'utilisation de plusieurs sous-listes peut servir à masquer quelques destinataires puisqu'un destinataire ne connaît que les membres contenus dans sa sous-liste. Nous reviendrons sur cet aspect plus loin.

Le tableau 3.4 récapitule les différences entre GXcast et le *multicast* traditionnel au niveau de la sécurité.

contrôle d'accès	confidentialité des membres	attaques
très bon	très mauvaise	grandement facilitées

TAB. 3.4 – *La sécurité dans GXcast par rapport à la sécurité en multicast traditionnel.*

3.1.4 L'analyse du protocole GXcast

Les performances du protocole GXcast dépendent de la topologie, de la position des membres et du paramètre n_p de GXcast. Comme la topologie et la position des membres n'est pas connue du protocole GXcast (c'est une des hypothèses du protocole Xcast), nous nous proposons d'étudier uniquement l'influence du paramètre n_p sur ses performances. Nous présentons ensuite une optimisation du protocole GXcast indépendante du paramètre n_p.

Cette analyse est une version combinée et étendue de celles que nous avons proposées dans [BGC03, BGC04a] et dans [BGC04b].

Étude du paramètre de GXcast

Rappelons que MTU désigne la valeur minimale garantie du MTU sur l'arbre implicite qu'utilise le protocole GXcast, que E désigne la taille de l'en-tête IP et de l'en-tête GXcast sans la liste des membres, et que IP est la taille d'une adresse IP.

Choix simple du paramètre. Le choix le plus simple pour la valeur du paramètre de GXcast est $n_p = n_{max}$ (ou $n_p = n_{max} - 1$ lorsque la valeur $n_p = n_{max}$ ne laisse pas de place pour les données). Cependant, ce choix n'est pas efficace pour les groupes ayant beaucoup de membres (typiquement de l'ordre de n_{max}). Plaçons nous dans le cadre d'un réseau IPv6 et supposons que le groupe contienne $n = 70$ membres. Supposons que $n_p = n_{max} = 76$. Chaque paquet peut contenir $1280 - (40 + 16) - 70 \cdot 16 = 104$ octets de données. Pour envoyer 10000 octets de données, $\lceil 10000/104 \rceil = 97$ paquets doivent être envoyés par la source. Moins de paquets auraient été nécessaires si n_p avait été mieux choisi. Supposons à présent que $n_p = 38$. Chaque paquet peut contenir $1280 - (40 + 16) - 38 \cdot 16 = 616$. En prenant en compte le fait que deux sous-listes de taille au plus $n_p = 38$ sont nécessaires pour couvrir $n = 70$ membres, le nombre de paquets que doit envoyer la source est à présent de $2 \cdot \lceil 10000/616 \rceil = 34$. Ce choix judicieux de n_p a pu diviser par presque trois le nombre de paquets que doit envoyer la source.

Nous pouvons noter que le choix $n_p = n_{max}$ a été fait dans l'implémentation actuelle de Xcast[3] sous Linux et Free BSD.

Le nombre de paquets générés. Nous avons vu qu'un choix judicieux de n_p permet de réduire grandement le nombre de paquets générés par la source. Dans ce paragraphe, nous cherchons la valeur de n_p qui rend le nombre de paquets générés minimal. Le nombre de paquets générés par la source pour envoyer d octets de données à n membres est :

$$p(n, d, n_p) = \lceil \frac{n}{n_p} \rceil \lceil \frac{d}{MTU - E - \min(n, n_p) \cdot IP} \rceil.$$

Le premier facteur de cette formule donne le nombre de sous-listes de taille au plus n_p qui sont nécessaires pour couvrir les n membres. Le second facteur donne le nombre de paquets nécessaires pour envoyer d octets de données quand la charge utile est limitée. En posant $\min(n, n_p) = n_p$, nous faisons l'approximation suivante :

$$\bar{p}(n, d, n_p) = \frac{n}{n_p} \frac{d}{MTU - E - n_p \cdot IP} \approx p(n, d, n_p).$$

$\bar{p}$ est une borne inférieure de p. Nous pouvons aussi obtenir une borne supérieure de p facilement. La figure 3.6 montre sur un exemple que les valeurs de la fonction $p(n, d, n_p)$ sont très proches de celles de la fonction $\bar{p}(n, d, n_p)$ (et de celles de la borne supérieure). Plus d (et n) augmentent, plus la courbe se lisse. La petite valeur choisie pour d montre que ce lissage est très rapide. Nous supposerons que le minimum de la fonction $p(n, d, n_p)$ est proche du

3. Bien que la spécification du protocole Xcast n'inclue pas la segmentation à la source, l'implémentation la réalise. En d'autres termes, l'implémentation correspond au protocole GXcast. C'est pourquoi nous pouvons parler du choix de n_p pour l'implémentation de Xcast.

minimum de $\bar{p}(n, d, n_p)$. Cela est d'autant plus vrai que n et d sont grands. Quand d tend vers l'infini, le rapport $\bar{p}(n, d, n_p)/p(n, d, n_p)$ tend vers n_p/n.

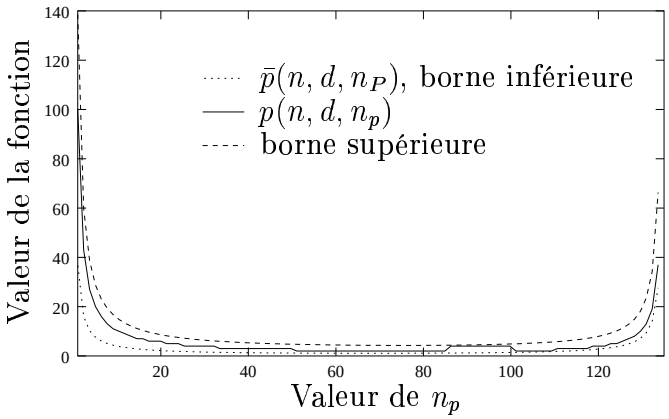


Figure 3.6 – *Encadrement de $p(n, d, n_p)$, quand $n = 100$ et $d = 200$, pour IPv4.*

Propriété 3. *La fonction $\bar{p}(n, d, n_p)$ admet un minimum pour :*

$$n_p = \frac{n_{max}}{2}.$$

Démonstration. La dérivée de $\bar{p}(n, d, n_p)$ par rapport à n_p s'écrit :

$$\frac{\partial \bar{p}(n, d, n_p)}{\partial n_p} = \frac{n.d.(E + 2n_p \cdot IP - MTU)}{n_p^2.(MTU - E - n_p \cdot IP)^2}.$$

Cette dérivée s'annule pour :

$$n_p = \frac{MTU - E}{2 \cdot IP} \approx \frac{n_{max}}{2}.$$

À partir du signe de la dérivée, nous déduisons que cette valeur de n_p réalise un minimum de la fonction $\bar{p}(n, d, n_p)$. □

Comme le montre la propriété 3, la valeur $n_{max}/2$ minimise la valeur de $\bar{p}(n, d, n_p)$. Comme $p(n, d, n_p)$ et $\bar{p}(n, d, n_p)$ sont proches, nous considérons que le nombre de paquets générés pour $p(n, d, n_p)$ est quasiment minimal autour de $n_{max}/2$. C'est une valeur approchée, indépendante du nombre de membres n et du nombre d'octets à envoyer d. Si n et d sont connus, la valeur n_p optimale peut être calculée par la formule suivante :

$$n_p = \arg \min_{1 \leq n_p \leq n_{max}} p(n, d, n_p).$$

Ce calcul peut être fait en $\mathcal{O}(n_{max})$. Comme n_{max} est limité à de petites valeurs (dans tous les cas par 127 puisque la taille de la liste est codée sur 7 bits), le calcul peut être fait dynamiquement de manière rapide.

Le nombre de membres influencés par la perte d'un paquet. S'il y a au plus n_p membres par paquet, la perte d'un paquet influence au plus n_p membres. Si l'on note p la probabilité qu'un paquet soit perdu, l'espérance du nombre de membres influencé par la panne est $p \cdot n_p \cdot p(n, d, n_p)$. L'étude de cette fonction montre que le minimum est atteint pour les très petites valeurs de n_p (si l'on ignore les parties entières supérieures, le minimum est atteint pour $n_p = 1$).

La pénalité de délai ajoutée par le protocole GXcast. Certains routeurs GXcast sont amenés à générer plusieurs paquets GXcast pour des interfaces de sortie différentes. Contrairement aux paquets *multicast* traditionnels, l'émission (en fait, le placement dans les files d'attente de sortie) des différents paquets n'est pas simultanée. À titre de comparaison, la figure 3.7(a) représente l'écoulement du temps pour un routeur dupliquant un même paquet depuis une interface d'entrée vers deux interfaces de sortie. Le cas de GXcast est présenté sur la figure 3.7(b). Dans ce cas, les paquets pour les deux interfaces de sorties sont différents, puisqu'ils ne contiennent pas la même liste de membres. Il y a un délai supplémentaire entre l'émission (ou plutôt la mise en file d'attente) du premier paquet sur la première interface de sortie et l'émission du second paquet sur la seconde interface de sortie. Ce délai est composé du temps nécessaire à créer la sous-liste des adresses du second paquet, à calculer la somme de contrôle pour ce paquet et à l'émettre (ou à mettre en file d'attente). Il est représenté en gris foncé sur la figure. Nous appelons ce délai la pénalité de délai et nous cherchons à réduire cette pénalité[4] induite par le protocole GXcast en modifiant le paramètre n_p.

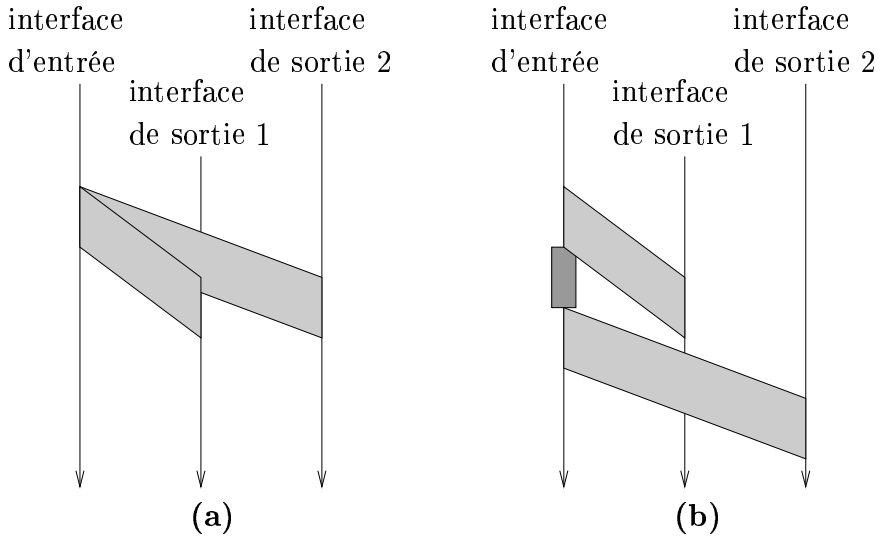


Figure 3.7 – *Émission de paquets dans un routeur (a) parallèlement pour des paquets identiques ou (b) séquentiellement pour des paquets différents.*

Considérons le réseau représenté sur la figure 3.8, et le groupe g formé d'une source s et des quatre membres d_1, d_2, d_3 et d_4. Nous noterons $\delta(d_i, n_p)$ la pénalité de délai ajoutée par

4. Celle-ci est souvent très courte dans la plupart des réseaux. Le gain obtenu ne sera généralement pas considérable.

le protocole GXcast pour le membre d_i et le paramètre n_p. Comme il est difficile d'estimer le délai t qui sépare l'émission (ou la mise en file d'attente) de deux paquets différents, la mesure de la pénalité sera donnée sans unité, c'est-à-dire en nombre de retards. Nous faisons ici l'hypothèse que cette pénalité est indépendante du routeur ou du paquet.

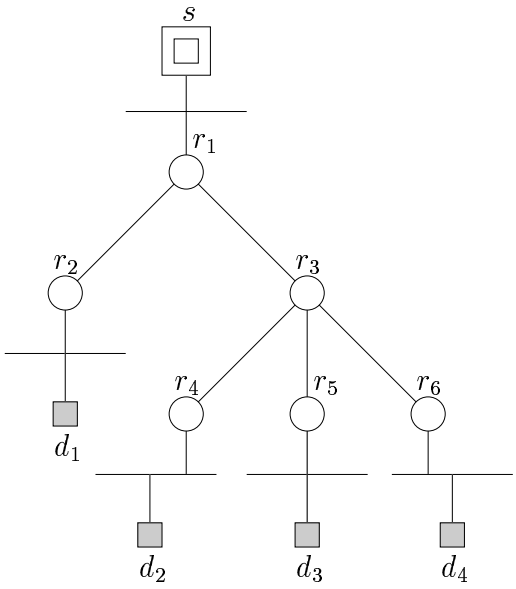


Figure 3.8 – *Le réseau utilisé pour les figures 3.9, 3.10 et 3.11.*

Considérons le cas où $n_p = 4$, représenté sur la figure 3.9. En suivant le chemin du paquet de s à d_1, nous observons que ce paquet n'a subi aucune pénalité de délai. Nous observons que le paquet de s à d_2 a subi une pénalité de délai en r_1 : r_1 a d'abord envoyé le paquet vers d_1 avant de l'envoyer vers r_3. Finalement, le paquet de s à d_3 a subi deux pénalités de délai (une en r_1 et une en r_3) et le paquet de s à d_4 a subi trois pénalités de délai (une en r_1 et deux en r_3).

Considérons le cas où $n_p = 2$, représenté sur la figure 3.10. En suivant le chemin de s à d_1, nous observons que ce paquet n'a subi aucune pénalité de délai. Quand le second paquet de s arrive en r_1, r_1 ne peut pas le traiter instantanément car il est en train de traiter le premier paquet de s. Le second paquet est retardé d'un temps que nous noterons ε, qui correspond au temps de mise en file d'attente d'un paquet. Nous supposerons que ce temps d'attente est très petit (de plusieurs ordres de grandeur) devant le temps de copie, et donc négligeable. Le paquet de s à d_2 a subi une pénalité de délai ainsi que le paquet de s à d_3 (en négligeant le temps de mise en file d'attente). Finalement, le paquet de s à d_4 subi deux pénalités de délai (une en s et une en r_3).

Considérons le cas où $n_p = 1$, représenté sur la figure 3.11. Le paquet de s à d_1 n'est jamais retardé, le paquet de s à d_2 a subi une pénalité de délai (en s), le paquet de s à d_3 a subi deux pénalités de délai (en s) et le paquet de s à d_4 a subi trois pénalités de délai (en s).

Le tableau 3.5 récapitule la pénalité totale de délai ajoutée par le protocole GXcast, c'est-à-dire la somme des pénalités de délai pour chacun des destinataires.

Nous avons vu que la pénalité de délai totale ajoutée par le protocole GXcast dépend de

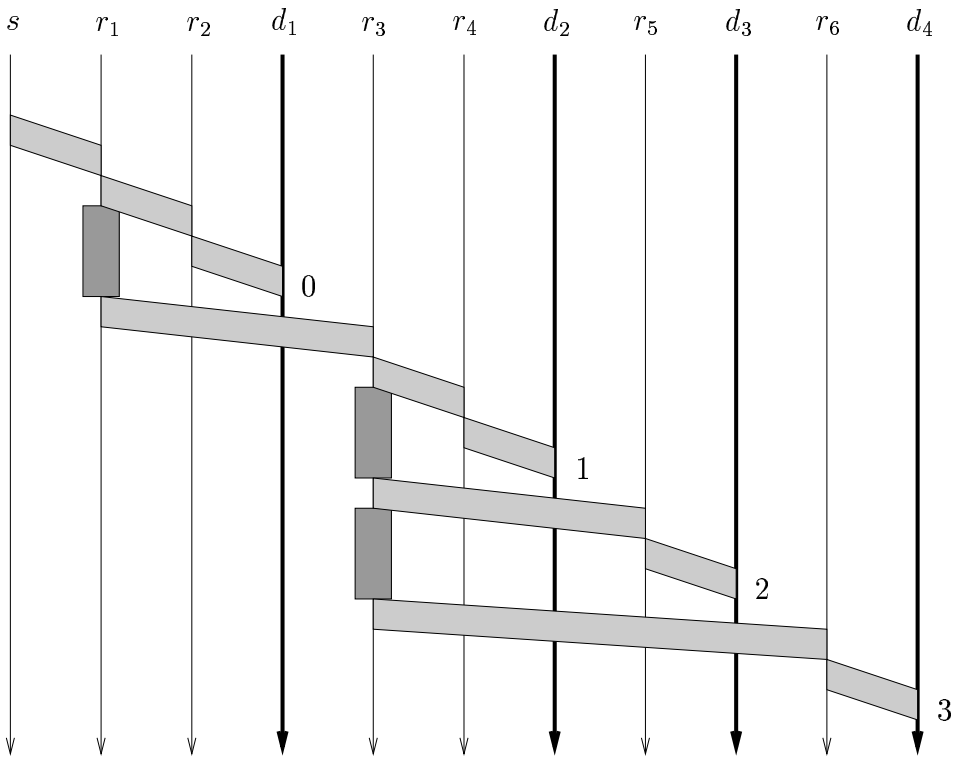


Figure 3.9 – *La pénalité de délai ajoutée par le protocole GXcast pour* $n_p = 4$.

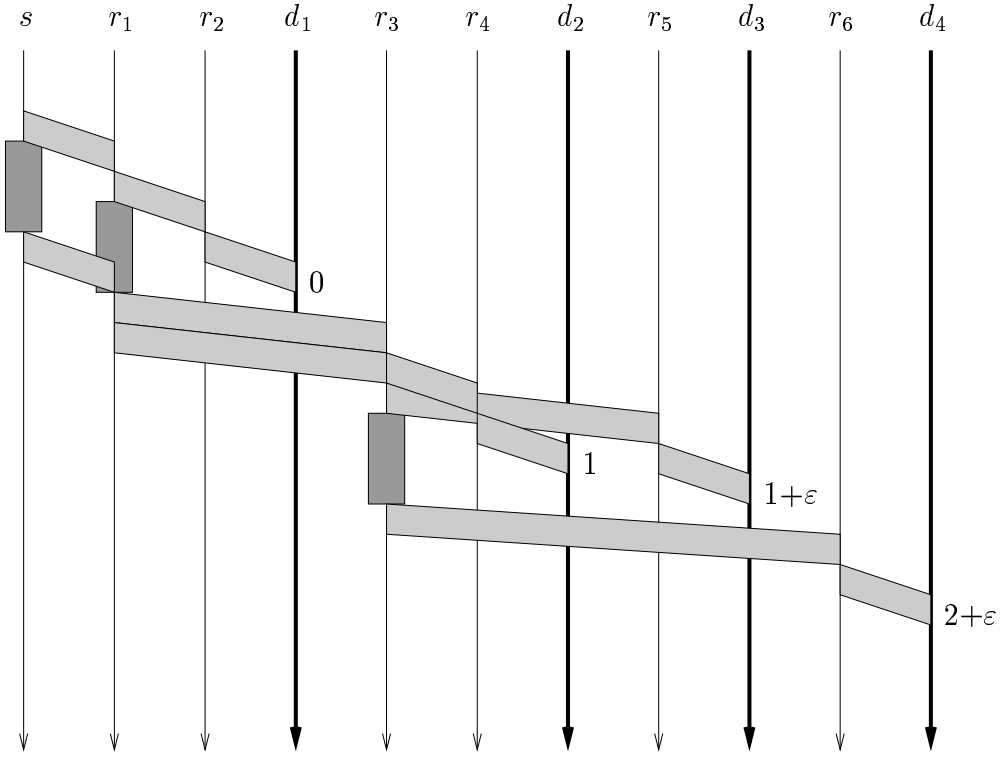


Figure 3.10 – *La pénalité de délai ajoutée par le protocole GXcast pour* $n_p = 2$.

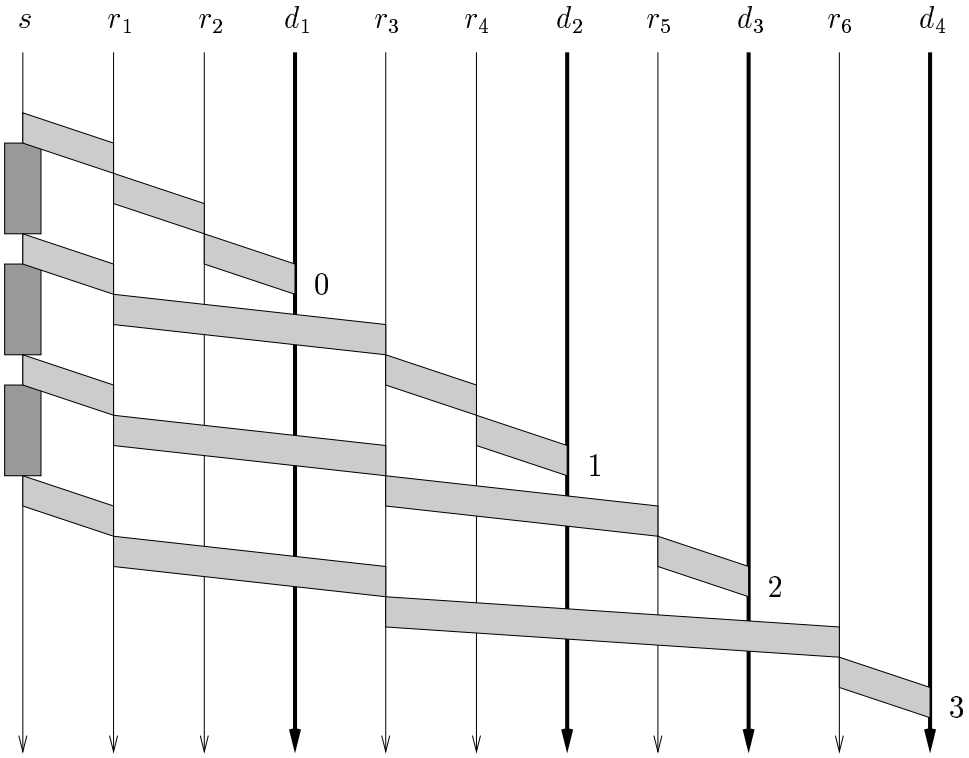


Figure 3.11 – *La pénalité de délai ajoutée par le protocole GXcast pour* $n_p = 1$.

	$\delta(d_1, n_p)$	$\delta(d_2, n_p)$	$\delta(d_3, n_p)$	$\delta(d_4, n_p)$	Pénalités totales de délai ajoutées
$n_p = 4$	0	1	2	3	6
$n_p = 2$	0	1	$1+\varepsilon$	$2+\varepsilon$	$4+2\varepsilon \approx 4$
$n_p = 1$	0	1	2	3	6

TAB. 3.5 – *Pénalité totale de délai ajoutée par le protocole GXcast en fonction de* n_p.

n_p. De plus, il apparaît sur l'exemple précédent que le minimum n'est atteint ni en $n_p = 1$ ni en $n_p = n_{max}$. La valeur de n_p minimisant[5] le total (c'est-à-dire la somme) des pénalités de délai est obtenue par les propriétés 4 et 5 suivantes.

Propriété 4. *Pour le membre d_i, on a :*

$$\delta(d_i, n_p) \leq \lfloor \frac{i-1}{n_p} \rfloor + (i \ mod \ n_p) - 1.$$

Démonstration. Nous avons vu que la pénalité de délai $\delta(d_i, n_p)$ dépend d'une part du nombre de paquets que la source envoie avant celui contenant d_i et d'autre part du nombre de paquets envoyés par les routeurs de branchement avant celui contenant d_i. Si les membres sont traités dans l'ordre par la source, le membre d_i est dans le $(\lfloor (i-1)/n_p \rfloor + 1)$-ème paquet émis par la source. Ainsi, d_i subit une pénalité de délai d'au moins $\lfloor (i-1)/n_p \rfloor$. L'autre partie de $\delta(d_i, n_p)$ est plus difficile à exprimer puisqu'elle dépend de la topologie. Comme nous avons fait l'hypothèse que l'arbre de routage nous était inconnu, nous nous plaçons dans le pire des cas. Le membre d_i est retardé du nombre de membres le précédant dans le paquet[6]. d_i étant le (i mod n_p)-ème membre de sa sous-liste, le retard subit est au maximum de (i mod n_p)-1. Le résultat suit. □

Propriété 5. *La borne maximale du délai de copie total est de taille minimale pour :*

$$n_p = \sqrt{n}.$$

Démonstration. Soit n_p la valeur du paramètre du protocole GXcast. Posons $k = n/n_p$, $k \in I\!R$. Pour tout membre d_i, nous avons $\lfloor (i-1)/n_p \rfloor \leq k$ et (i mod n_p)$-1 \leq (n_p - 1) - 1 = n/k - 2$. Ainsi, selon la propriété 4, nous pouvons affirmer que $\delta(d_i, n_p) \leq k + n/k - 2$. Cette fonction admet un minimum en $k^* = \sqrt{n}$. Or, $k^* = n/n_p^*$, d'où $n_p^* = n/k^*$. La valeur optimale n_p^* correspondant à $k^* = \sqrt{n}$ est donc $n_p^* = n/\sqrt{n} = \sqrt{n}$. □

Ainsi, nous proposons d'utiliser la valeur $n_p = \min\{\sqrt{n}, n_{max}\}$ pour les applications sensibles au délai[7].

Tri de la liste des membres

La liste des membres que mémorise une source est *a priori* quelconque. L'ordre des membres dans cette liste peut être quelconque. Toutefois, nous pouvons améliorer les performances du protocole GXcast (ou même du protocole Xcast) en triant cette liste selon certains critères. Nous proposons ici un tri topologique. Plus loin, nous envisagerons d'autres tris réalisables.

5. Plus précisément, la propriété 5 donne la valeur de n_p qui minimise la taille de l'intervalle dans lequel se trouve la pénalité de délai totale. Bien que le délai total exact ne soit pas connu à cause de l'inégalité donnée par la propriété 4, la valeur $n_p = \sqrt{n}$ offre de grandes chances de correspondre à une faible pénalité totale ajoutée.

6. C'est le cas si l'arbre (implicite) suivi par le paquet forme un peigne droit.

7. Nous avons fait l'hypothèse dans les preuves que $\sqrt{n} \leq n_{max}$. Ce n'est pas forcément le cas mais le protocole GXcast n'est pas adapté pour les groupes dont le nombre de membres dépasse n_{max}^2. En effet, pour IPv4, cela correspondrait à des groupes de 18225 membres et pour IPv6, à des groupes de 5776 membres.

Deux membres proches au sens topologique ont peu de chances d'être proches dans une liste de membres non triée, surtout s'ils ont joint le groupe à des instants différents. La figure 3.12 présente un mauvais regroupement des membres dans la liste de s avec les paramètres suivants : $n = 6$, $n_p = 3$ et $d = 1$. En effet, six liens entre routeurs ont un stress élevé puisqu'ils supportent deux paquets. Par exemple, le lien (r_3, r_4) supporte un paquet *unicast* à d_1 et un à d_6, et le lien (r_3, r_5) supporte un paquet GXcast à $\{d_2, d_3\}$ et un à $\{d_4, d_5\}$. Les performances du protocole pourraient être améliorées par un meilleur regroupement. Le regroupement présenté sur la figure 3.5 conduit à une meilleure utilisation des liens : le lien (r_3, r_4) n'est utilisé que par un seul paquet GXcast à $\{d_1, d_2\}$. Ainsi, seuls cinq liens supportent deux paquets.

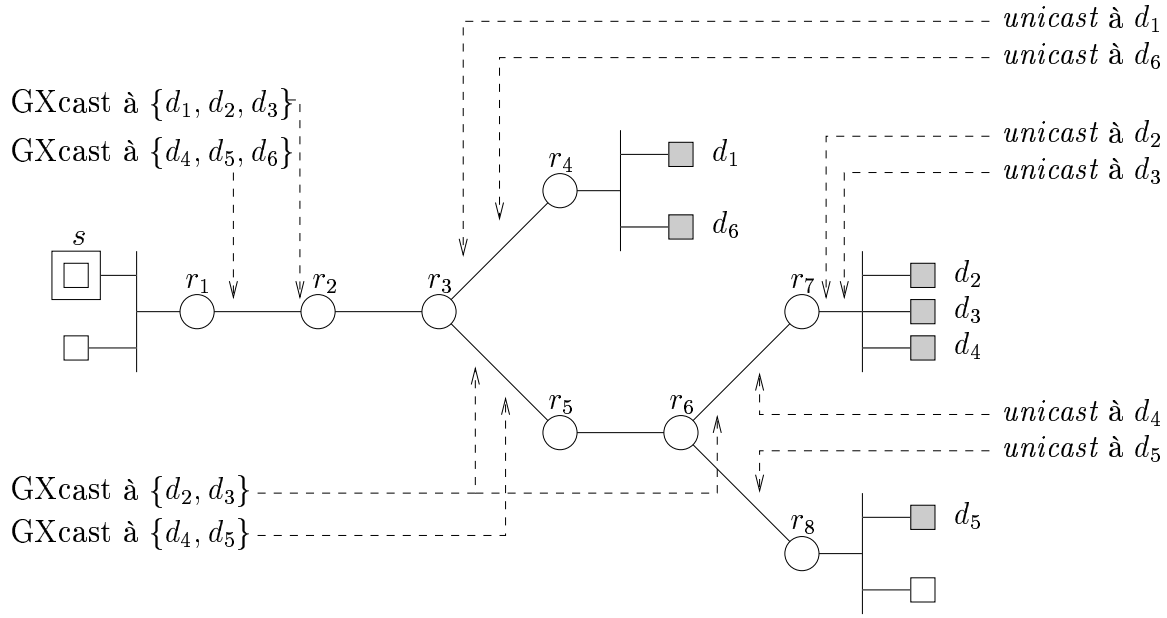


Figure 3.12 – *Un mauvais regroupement des membres détériore les performances de GXcast.*

La position topologique d'un membre ne peut pas être connue avec précision sur Internet. Cependant, nous proposons d'utiliser la proximité des adresses IP comme indicateur de proximité de deux membres. Cet indicateur est bien sûr très imprécis, mais il permet, comme nous le verrons, d'augmenter significativement les performances, même quand l'imprécision est grande.

Nous proposons de trier la liste des membres à la source pour réduire les chances d'avoir un mauvais regroupement. Les membres sont triés par ordre lexicographique de leurs adresses IP. Ainsi, les membres qui se trouvent dans un même réseau local auront de grandes chances d'être dans la même sous-liste.

Étude des performances du tri. Pour évaluer l'impact du tri de la liste des membres, nous avons effectué des simulations du protocole GXcast sur la topologie d'Abilène. Abilène est l'épine dorsale américaine du réseau Internet2. Ce réseau est constitué de 11 nœuds et de 14 arêtes.

Pour mesurer les performances du tri sur GXcast, nous avons attaché à chaque routeur entre 1 et 20 réseaux locaux. Ici, nous considérons qu'un réseau local correspond à une plage d'adresses de classe C, c'est-à-dire 256 adresses consécutives. Il faut noter que les adresses des réseaux locaux attachés à un même routeur ne sont pas consécutives en général. Nous modélisons par cette indépendance l'incertitude sur la localisation topologique des membres. Nous avons généré $n = 210$ membres parmi toutes les adresses potentielles[8]. Puis, nous avons mesuré le nombre de paquets générés par le protocole GXcast avec et sans tri de la liste des membres, pour $n_p = 70$ (valeur proche de $n_{max}/2$ pour IPv4) et $d = 1$. Ces paramètres permettent de g'enérer trois paquets par groupe. Le nombre moyen de paquets générés par groupe, sur un ensemble de 1000 groupes, est représentée sur la figure 3.13 en fonction du nombre de réseaux locaux par routeur. Sans tri, chaque groupe est acheminé par des arbres de routage (ou plutôt des forêts) utilisant en moyenne 30 liens. Sans tri, le comportement de GXcast est indépendant du nombre de réseaux locaux attachés à un routeur. En effet, le choix de 210 membres parmi le faible nombre de routeurs d'Abilène (seulement 11) assure avec une grande probabilité que tous les routeurs seront représentés. Le nombre de paquets générés, 30, correspond à trois arbres de taille 10, c'est-à-dire que chacun des trois arbres couvre bien les 11 nœuds. En triant la liste des membres, nous remarquons que le nombre de paquets générés par GXcast (qui est une mesure de la qualité du regroupement) est réduit. Moins il y a de réseaux locaux attachés à un routeur, plus le nombre de paquets générés est faible. Cela s'explique par le fait que la localité déduite des adresses IP augmente quand le nombre de réseaux locaux par routeur diminue. Cependant, même lorsque le nombre de réseaux locaux attachés à un routeur est très grand, le nombre de paquets générés par GXcast est plus faible en triant la liste des membres que sans la trier.

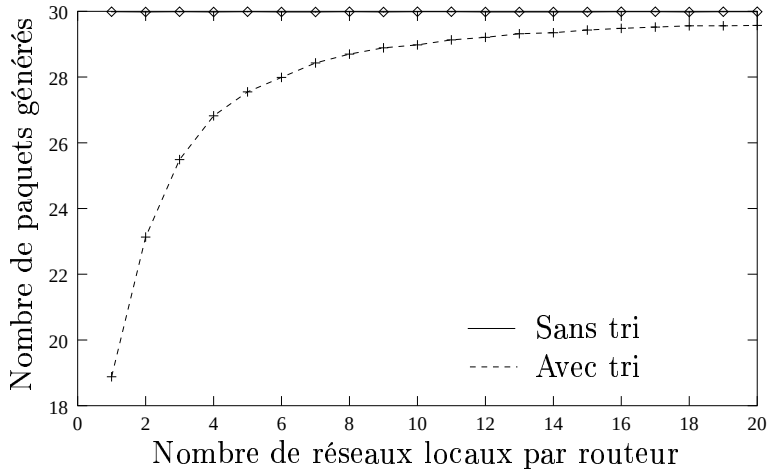


Figure 3.13 – *Le tri de la liste des membres améliore sensiblement le regroupement.*

À présent, nous mesurons l'influence du nombre de membres sur le regroupement. Nous attachons à chaque routeur 5 réseaux locaux. Nous conservons $n_p = 70$ et nous générons de $n = 210$ à $n = 910$ membres par pas de 70 parmi toutes les adresses potentielles. Cela nous

8. Puisque chacun des 11 nœuds a entre 1 et 20 plages de 256 adresses, le nombre d'adresses potentielles est compris entre $1 \cdot 11 \cdot 256 = 2816$ et $20 \cdot 11 \cdot 256 = 56320$ adresses. Les 210 adresses générées correspondent donc à entre 0.37 % et 7.46 % des adresses potentielles.

permet d'avoir un nombre de membres par paquet exact. Puis, nous mesurons le nombre de paquets générés par GXcast avec et sans tri de la liste des membres, pour $d = 1$. La moyenne sur 1000 groupes est représentée sur la figure 3.14.

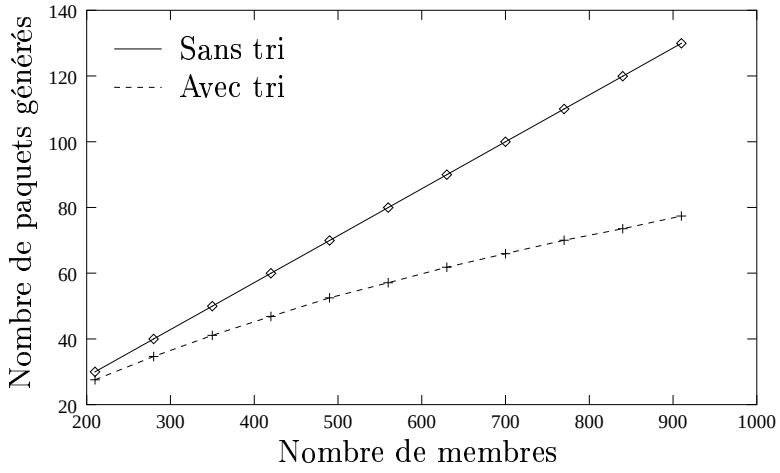


Figure 3.14 – *Un grand nombre de membres améliore le regroupement.*

Sans tri, GXcast génère dans ces simulations un nombre de paquets qui varie linéairement par rapport au nombre de membres, comme dans le résultat de simulation précédent. Nous remarquons que la qualité du regroupement augmente lorsque le nombre de membres augmente. En effet, un grand nombre de membres augmente la probabilité que deux membres soient dans un même sous-réseau. Pour $n = 350$ membres (5 sous-listes sont nécessaires pour couvrir ces membres car $\lceil n/n_p \rceil = 5$), le regroupement réduit d'environ 20 % le nombre de paquets générés par GXcast.

Étude de la complexité du tri à la source. Il est important d'étudier le coût du tri de la liste des membres à la source. Les trois opérations à considérer sont : l'ajout d'un membre à la liste, le retrait d'un membre de la liste et l'opération de segmentation.

Nous proposons de mémoriser la liste de membres comme un arbre rouge-noir (décrits au chapitre 14 de [CLRS90]). L'intérêt principal de cette structure dans notre cas est que l'insertion, la suppression et la recherche[9] d'un élément dans un tel arbre peuvent être effectués en temps logarithmique en fonction du nombre d'éléments. L'opération de segmentation de la liste, qui consiste en fait à découper la liste des n membres en sous-listes d'au plus n_p membres, peut être faite en temps linéaire en fonction de n. Pour cela, on procède en deux étapes : (1) une liste des membres triés par ordre lexicographique est obtenue par un parcours infixe de l'arbre de recherche puis (2) la liste est parcourue puis découpée en sous-listes d'au plus n_p éléments. Ainsi, la complexité de notre tri à la source est faible : logarithmique en n pour l'ajout d'un membre et le retrait d'un membre, linéaire en n pour l'opération de segmentation.

9. La suppression d'un élément nécessite que cet élément soit préalablement trouvé, ce qui justifie le besoin d'une recherche rapide dans la structure.

En conclusion, l'amélioration qui consiste à trier la liste des membres introduit une faible complexité (du même ordre que sans tri) et améliore sensiblement les performances de GXcast.

3.1.5 La gestion de la dynamicité des groupes dans GXcast

Appelons « segmentation stricte » la segmentation que nous avons proposée, qui consiste à avoir un seuil n_p sur le nombre de membres par paquet.

Définition 4 (Segmentation stricte). *Une segmentation stricte est définie par une valeur n_p. Elle réalise un découpage d'une liste de n membres en sous-listes contenant chacune exactement n_p membres, sauf éventuellement la dernière.*

La segmentation stricte que nous avons proposée n'est pas adaptée aux groupes dynamiques. Pour s'en convaincre, supposons que $n = n_{max}/2 + 1$. Choisir $n_p = n_{max}/2$ conduit à créer deux paquets alors que vraisemblablement, il aurait été plus intéressant de n'en faire qu'un seul si la quantité de données à transmettre d est petite. Même si le découpage dans GXcast était équilibré (c'est-à-dire qu'autant de membres sont dans chaque sous-liste, ce qui ne correspond pas à notre définition de segmentation stricte), deux paquets auraient été générés au lieu d'un seul.

Ainsi, nous proposons une méthode originale de segmentation à la source : la segmentation souple. Plutôt que de définir un seul seuil n_p, nous définissons un intervalle $[n_p^1; n_p^2]$. Si n est compris entre n_p^1 et n_p^2, un seul paquet est généré. Si n est plus grand que n_p^2, plusieurs paquets seront générés. La sous-liste de chacun des paquets sera de taille comprise entre n_p^1 et n_p^2, à l'exception éventuellement de la dernière sous-liste.

Définition 5 (Segmentation souple). *Une segmentation souple est définie par un intervalle $[n_p^1; n_p^2]$. Elle réalise un découpage d'une liste de n membres en sous-listes contenant chacune entre n_p^1 et n_p^2 membres, sauf la dernière.*

L'exception faite pour la dernière sous-liste vient du fait qu'il n'est pas toujours possible de découper une liste de n membres en sous-listes de taille comprises entre n_p^1 et n_p^2. Nous introduisons à présent la notion d'homogénéité d'une segmentation souple.

Définition 6 (Segmentation souple homogène). *Une segmentation souple sur $[n_p^1; n_p^2]$ est homogène si pour tout $n \geq n_p^1$, elle réalise un découpage d'une liste de n membres en sous-listes contenant chacune entre n_p^1 et n_p^2 membres, y compris la dernière.*

La propriété 6 donne une condition nécessaire et suffisante sur l'homogénéité d'une segmentation souple.

Propriété 6. *Une segmentation souple sur $[n_p^1; n_p^2]$ soit homogène si et seulement si pour tout $n \geq n_p^1$, il existe un entier k non nul tel que $k.n_p^1 \leq n \leq k.n_p^2$.*

Démonstration. Supposons que la segmentation souple sur $[n_p^1; n_p^2]$ est homogène. Alors, il existe un découpage d'une liste de n membres en k sous-listes, k non nul. Notons n_i le nombre d'éléments de la i-ème sous-liste. Pour tout i, on a $n_p^1 \leq n_i \leq n_p^2$. En sommant pour tous les i, on obtient $k.n_p^1 \leq \sum_{i=1}^{k} n_i \leq k.n_p^2$ puis $k.n_p^1 \leq n \leq k.n_p^2$.

Supposons à présent qu'il existe un entier k non nul tel que $k.n_p^1 \leq n \leq k.n_p^2$. En divisant par k, on obtient $n_p^1 \leq n/k \leq n_p^2$. Nous pouvons donc découper la liste de n membres

en k sous-listes de n/k (à l'arrondi près) membres, n/k étant compris entre n_p^1 et n_p^2. La segmentation est souple. □

La propriété 6 indique que l'on peut découper la liste en sous-listes de même taille (à l'arrondi près) sans perte de généralité.

Intuitivement, pour qu'une segmentation souple soit homogène, il faut que l'intervalle $[n_p^1; n_p^2]$ soit suffisamment large. Cependant, choisir un intervalle trop large éloigne le découpage de la borne $n_{max}/2$. En s'appuyant sur l'intervalle de taille minimale pour que la segmentation soit homogène, le théorème 6 donne le rapport d'approximation minimal pour qu'une segmentation souple soit homogène. Ce théorème va s'appuyer sur le lemme suivant.

Lemme 5. *Soit $\alpha \geq 0$, $\beta = \sqrt{\alpha/(\alpha+1)}$, $n_p^1 = n_{max}(1-\beta)/2$ et $n_p^2 = n_{max}(1+\beta)/2$. Le rapport d'approximation (par rapport à $n_{max}/2$) du nombre de paquets pour la segmentation souple sur $[n_p^1; n_p^2]$ est $(1+\alpha)$.*

Démonstration. Soit $p(n, d, n_p)$ le nombre de paquets générés par une segmentation stricte de n_p pour envoyer d octets à n membres. Nous cherchons à montrer que le rapport d'approximation de la segmentation souple sur $[n_p^1; n_p^2]$ est de $(1+\alpha)$ par rapport à $n_{max}/2$, c'est-à-dire que pour tout $n_p \in [n_p^1; n_p^2]$, on a $p(n, d, n_p) \leq (1+\alpha)p(n, d, n_{max}/2)$. Nous approximons $p(n, d, n_p)$ par $\bar{p}(n, d, n_p)$.

$$
\begin{aligned}
& p(n,d,n_p) \leq (1+\alpha)p(n,d,n_{max}/2) \\
\Leftrightarrow\ & \frac{n}{n_p} \cdot \frac{d}{MTU - E - n_p \cdot IP} \leq (1+\alpha)\frac{n}{n_{max}/2} \cdot \frac{d}{MTU - E - n_{max} \cdot IP/2} \\
\Leftrightarrow\ & n_p \cdot (MTU - E - n_p) \cdot IP \geq \frac{n_{max}}{2 \cdot (1+\alpha)}(MTU - E - n_{max} \cdot IP/2) \\
\Leftrightarrow\ & n_p \cdot MTU - n_p \cdot E - n_p^2 \cdot IP - \frac{(MTU-E)(MTU-E)}{4 \cdot IP \cdot (1+\alpha)} \geq 0 \\
\Leftrightarrow\ & IP \cdot (n_p)^2 - (MTU - E) \cdot n_p + \frac{(MTU-E)^2}{4 \cdot IP \cdot (1+\alpha)} \leq 0 \\
\Leftrightarrow\ & \frac{(MTU-E)(1-\beta)}{2 \cdot IP} \leq n_p \leq \frac{(MTU-E)(1+\beta)}{2 \cdot IP},
\end{aligned}
$$

avec $\beta = \sqrt{\alpha/(\alpha+1)}$. Nous avons bien $n_p \in [n_p^1; n_p^2]$. L'intervalle $[n_p^1; n_p^2]$ est de taille minimale : une segmentation souple de rapport d'approximation $(1+\alpha)$ couvre un intervalle de taille supérieure ou égale à $n_p^2 - n_p^1 + 1$. □

Fort de ce lemme, nous pouvons maintenant prouver le théorème 6.

Théorème 6. *Le rapport d'approximation du nombre de paquets pour toute segmentation souple homogène est supérieur ou égal à $9/8$.*

Démonstration du théorème 6. Le lemme 5 nous donne une segmentation souple (non nécessairement homogène) de rapport d'approximation $(1+\alpha)$ et dont l'intervalle est de taille minimale. Nous allons donner une condition nécessaire et suffisante sur α (et donc sur la taille de l'intervalle) pour que cette segmentation souple soit homogène.

La segmentation souple sur $[n_p^1; n_p^2]$ donnée par le lemme 5 est homogène si et seulement si pour tout $n \geq n_p^1$, il existe un entier non nul k tel que $k.n_p^1 \leq n \leq k.n_p^2$ (d'après la propriété 6). Cette propriété est équivalente à $k.n_p^2 \geq (k+1).n_p^1$ et $k.n_p^1 \leq (k-1)n.p^2$. En effet, si n est supérieur à la borne supérieure $k.n_p^2$, n doit être aussi supérieur à la borne inférieure suivante $(k+1).n_p^1$. De la même manière, si n est inférieur à la borne inférieure $k.n_p^1$, n doit être aussi inférieur à la borne supérieure précédente $(k-1).n_p^2$. Ces deux conditions sont équivalentes. Pour que la segmentation souple soit homogène, il faut et il suffit donc de montrer que $(k+1).n_p^1 \leq k.n_p^2$.

Posons $\beta = \sqrt{\alpha/(\alpha+1)}$. On a :

$$\begin{aligned}
(k+1).n_p^1 \leq k.n_p^2 &\Leftrightarrow (k+1).n_{max}(1-\beta)/2 \leq k.n_{max}(1+\beta)/2 \\
&\Leftrightarrow (k+1)(1-\beta) - k(1+\beta) \leq 0 \\
&\Leftrightarrow \beta \geq \frac{1}{2k+1} \\
&\Leftrightarrow \frac{1}{1+1/\alpha} \geq \frac{1}{(2k+1)^2} \\
&\Leftrightarrow \alpha \geq \frac{1}{4k(k+1)}.
\end{aligned}$$

Comme k est un entier non nul et que α doit être supérieur à $1/(4k.(k+1))$ pour tout k, la segmentation est homogène si et seulement si $\alpha \geq 1/8$. Pour ce choix de α, on a $\beta = 1/3$, $n_p^1 = n_{max}/3$ et $n_p^2 = 2.n_{max}/3$. □

On peut remarquer que tout intervalle de la forme $[i+1; 2i+1]$, avec $i > 0$ conduit à une segmentation souple homogène.

À présent, nous disposons d'une segmentation souple homogène. Comme nous l'avons dit précédemment, le ratio d'approximation d'une telle segmentation est borné. Un autre avantage d'une telle segmentation est qu'elle facilite le regroupement des membres d'un groupe.

Définition 7 (Regroupement). *Un regroupement des membres est une relation d'équivalence $\mathcal{R}$.*

Définition 8 (Communauté). *Les communautés sont les classes d'équivalence d'un regroupement.*

Il est important que les membres d'une même communauté soient dans les mêmes paquets. De plus, il faut éviter, dans la mesure du possible, qu'un paquet contienne les membres de plusieurs communautés distinctes.

Plusieurs regroupements peuvent être envisagés. Parmi ceux-ci, il y a :

- les regroupements topologiques,
- les regroupements sémantiques,
- les regroupements historiques.

Les regroupements topologiques regroupent les membres qui sont proches les uns des autres (au sens topologique mais pas nécessairement géographique) afin de tirer profit de la

localité. Le tri de la liste des destinations présenté plus tôt était un regroupement topologique. Il peut être défini plus formellement par la relation $\mathcal{R}$ définie par

$$m_1 \mathcal{R} m_2 \Leftrightarrow m_1 \text{ et } m_2 \text{ sont dans le même sous-réseau,}$$

m_1 et m_2 étant deux membres. Les regroupements topologiques sont ceux qui offrent les meilleures performances.

Les regroupements sémantiques traduisent une proximité sémantique. Cette proximité sémantique permet souvent une plus grande équité entre les membres: les membres d'une même communauté sont influencés de la même manière en cas de perte ou de congestion.

Les regroupements historiques tendent à conserver les sous-listes stables en dépit des changements de groupe. Sans regroupement historique, un membre peut être transféré d'une sous-liste à une autre en fonction des changements du groupe. Les inconvénients de ne pas faire de regroupement historique sont les suivants:

- le membre peut apprendre au fur et à mesure tous les autres membres du groupe,
- selon la position du membre dans les sous-listes successives, la gigue du délai des paquets pour ce membre peut être importante.

Il est donc important d'inclure la gestion des communautés dans l'algorithme de segmentation de la liste afin de tirer profit du regroupement.

Nous proposons donc un algorithme réalisant une segmentation souple homogène en présence d'un critère de regroupement quelconque. Ce critère définit des communautés. L'algorithme de regroupement est composé de deux phases. La première phase est illustrée sur la figure 3.15. Chaque communauté c_i de taille inférieure à n_p^2 est affectée à un unique paquet (c'est le cas pour les communautés c_2 et c_4 de la figure), alors que chaque communauté c_i de taille strictement supérieure à n_p^2 est affectée à plusieurs paquets, tous de taille exactement n_p^2 sauf le dernier (c'est le cas pour les communautés c_1 et c_3). Pendant la seconde phase de l'algorithme, tous les paquets dont la taille est inférieure à n_p^1 sont considérés (sur la figure, c'est le cas des paquets p_3 et p_5). Ces paquets sont fusionnés deux à deux, jusqu'à ce qu'ils soient tous de taille supérieure à n_p^1 ou jusqu'à ce qu'il n'en reste qu'un seul. On peut noter que nous faisons ici l'hypothèse que $2n_p^1 \leq n_p^2$, ce qui est le cas pour la segmentation souple homogène optimale ($n_p^1 = n_{max}/3$ et $n_p^2 = 2n_{max}/3$).

3.1.6 Le déploiement incrémental d'un protocole *multicast* explicite plat

Dans ce paragraphe, nous étudions le déploiement incrémental du protocole Xcast. Cette analyse est tirée de [GB05]. Nous décrivons tout d'abord trois mécanismes permettant un déploiement incrémental du protocole Xcast. Ensuite, nous analysons les performances de Xcast selon ces mécanismes et nous mettons en avant le compromis entre le volume d'information transmis sur le réseau, le temps de traitement des paquets et la longueur du chemin pour atteindre les destinataires. Puis, nous proposons quatre stratégies heuristiques de sélection des routeurs d'un réseau sur lesquels il convient d'implémenter Xcast pour améliorer ses performances. Nous comparons ces stratégies heuristiques à une stratégie optimale, non réalisable en pratique car elle suppose que les groupes sont connus *a priori*.

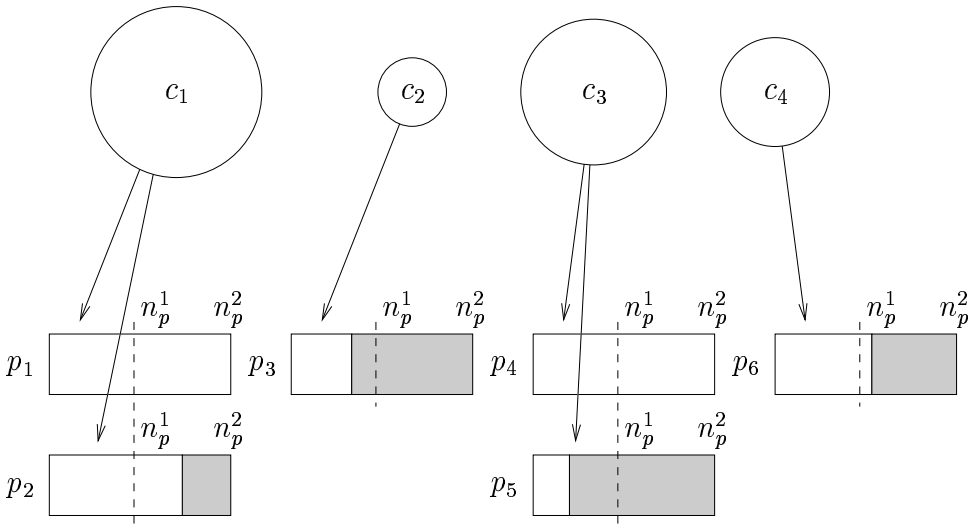


Figure 3.15 – *Regroupement de communautés en paquets.*

Mécanismes de déploiement incrémental de Xcast

Le déploiement du protocole Xcast est limité par la position intermédiaire de l'en-tête Xcast entre l'en-tête IP et l'en-tête UDP. En effet, pour être utilisable sur Internet, Xcast devrait être implémenté dans tous les routeurs, ce qui n'est pas envisageable. Il est donc important d'étudier le déploiement incrémental de Xcast.

Dans les spécifications du protocole Xcast [BFI+05], trois mécanismes de déploiement incrémental sont décrits. La partie 11.1 de cette spécification décrit le mécanisme de tunnels, la partie 11.2 décrit le mécanisme X2U prématuré et les parties 11.3 et 11.4 décrivent les tunnels semi-perméables. Le premier mécanisme, les tunnels, utilise des tunnels configurés manuellement entre les routeurs Xcast. Le deuxième mécanisme, X2U prématuré, utilise la capacité de duplication d'un routeur Xcast dans le cas où le routeur suivant n'est pas Xcast. Le troisième mécanisme, les tunnels semi-perméables, réalise un réseau virtuel (hybride) entre les destinataires.

Les tunnels. Le mécanisme de tunnels nécessite que chaque routeur Xcast connaisse ses plus proches pairs Xcast, c'est-à-dire ses voisins au sens de la connectivité Xcast. Ainsi, un réseau virtuel est construit au dessus de la topologie réelle du réseau.

Cette proposition est difficile à déployer puisqu'elle exige une configuration manuelle des tunnels. En effet :

- les routeurs Xcast sont incapables de détecter efficacement leurs voisins Xcast (au sens de la connectivité Xcast),
- lorsqu'un nouveau routeur Xcast est déployé dans le réseau, non seulement le nouveau routeur mais aussi ses routeurs voisins (au sens de la connectivité Xcast) doivent être (re)configurés.

Dans ce paragraphe, nous n'étudions pas ce mécanisme qui n'est pas automatique, et donc pas envisageable pour un déploiement à grande échelle.

Le X2U prématuré. Le mécanisme X2U prématuré nécessite qu'un routeur Xcast puisse savoir si ses routeurs voisins (directs) sont des routeurs Xcast ou non. Cela peut être fait efficacement par une légère modification des informations échangées entre routeurs voisins pour le routage IP[10]. Notons que sans cette modification, le mécanisme X2U prématuré n'est pas réalisable. Lorsqu'un routeur Xcast désire envoyer un paquet Xcast à un routeur voisin, il examine une table interne lui indiquant si ce routeur est Xcast. Si le routeur voisin est Xcast, le paquet Xcast est envoyé au routeur. Si le routeur voisin n'est pas Xcast, la liste des destinataires du paquet Xcast est extraite et un paquet *unicast* contenant les données est envoyé à chacun des destinataires. En d'autres termes, l'algorithme X2U décrit précédemment est utilisé de manière prématurée. De cette manière, un routeur non Xcast ne voit jamais passer de paquets Xcast.

Ce mécanisme peut être mis en relation avec le mécanisme multi-*unicast*. Dans le multi-*unicast*, la source émet un paquet *unicast* par destinataire. Le comportement du multi-*unicast* et de X2U prématuré sont proches si le réseau comporte peu de routeurs Xcast. En revanche, lorsque beaucoup de routeurs du réseau sont Xcast, X2U prématuré adopte un comportement se rapprochant de celui du protocole Xcast totalement déployé.

Il convient de disposer d'un mécanisme plus efficace lorsque peu de routeurs Xcast sont déployés dans le réseau. Le mécanisme de tunnels semi-perméables va dans ce sens.

Les tunnels semi-perméables. Le mécanisme de tunnels semi-perméables utilise l'option de routage HBH (pour *Hop-By-Hop*) d'IPv6 [RFC 2460]. Cette option permet à un routeur d'ignorer un en-tête inconnu sans détruire le paquet. Lorsqu'un routeur Xcast désire envoyer un paquet Xcast à un routeur voisin, il place dans l'adresse de destination IP du paquet Xcast l'adresse du premier destinataire et envoie ce paquet au routeur voisin. Si le routeur voisin est un routeur Xcast, il détecte l'option Xcast et achemine le paquet selon le protocole Xcast. Si le routeur voisin n'est pas un routeur Xcast, il ignore l'en-tête de routage Xcast et va acheminer le paquet à l'adresse IP du paquet, c'est-à-dire au premier destinataire de la liste. Pour bien fonctionner, ce mécanisme exige des destinataires qu'ils soient Xcast, puisqu'un destinataire recevant un paquet Xcast peut être amené à acheminer ce paquet aux destinataires suivant. Cette hypothèse est valide puisque les destinataires sont souvent des routeurs désignés et non des hôtes [SKPK01].

Le surcoût de la option HBH en termes de nombre d'octets est négligeable. En revanche, utiliser cette option présente deux inconvénients :

- tous les routeurs doivent être en mesure de reconnaître et d'analyser correctement cette option.
- en cas de congestion, les paquets possédant des options, donc tous les paquets Xcast, sont rejetés en priorité.

Lorsque peu de routeurs Xcast sont déployés, la longueur des chemins de la source aux destinataires peut être arbitrairement grande.

10. Plus précisément, il s'agit de l'ajout d'un champ dans les LSA (pour *Link State Advertisement*) du protocole OSPF (pour Open Shortest Path First [RFC 2328]) ou dans les messages de mise à jour du protocole RIP (pour Routing Information Protocol [RFC 2453]).

Performances des mécanismes

Notre analyse des performances porte sur les deux mécanismes de déploiement automatiques[11] proposés dans [BFI+05] : le mécanisme X2U prématuré et le mécanisme de tunnels semi-perméables. Ces deux mécanismes sont comparés au multi-*unicast*.

Les paramètres de simulation. Nous étudions les performances du protocole Xcast dans le réseau Abilène et dans le réseau Eurorings. Abilène est constitué de 11 routeurs et de 14 arêtes. Eurorings est constitué de 43 routeurs et de 55 arêtes[12].

Nous faisons varier le nombre de routeurs Xcast déployés dans le réseau entre 0 et 11 (c'est-à-dire de 0 % à 100 %) pour Abilène et entre 0 et 43, par pas de 5, pour Eurorings. À chaque fois, nous générons aléatoirement 10 positions des routeurs Xcast[13]. Pour chaque position de routeurs Xcast, nous faisons la moyenne des performances pour 1000 groupes *multicast*. Chaque groupe *multicast* a une source aléatoire (considérée comme étant Xcast pour ce groupe) et est de taille aléatoire. De la même manière, chaque destinataire du groupe est considéré comme étant Xcast pour ce groupe. Chaque point des courbes est la moyenne de 10 jeux de simulations.

Pour avoir une analyse complète des mécanismes étudiés, nous considérons les trois critères suivants : le nombre de paquets total, le nombre d'examens de la table *unicast* dans un routeur et la longueur du chemin pour atteindre les destinataires.

Nombre de paquets total. Le nombre de paquets total est le nombre de paquets Xcast et *unicast* circulant sur le réseau. La figure 3.16 montre le nombre de paquets total induits par les trois mécanismes en fonction du nombre de routeurs Xcast, sur Abilène et Eurorings.

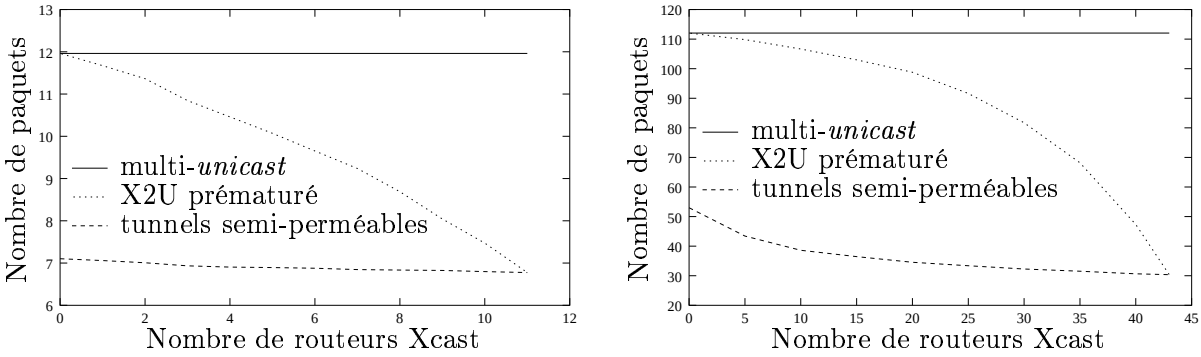


Figure 3.16 – *Évolution du nombre de paquets total en fonction du nombre de routeurs Xcast déployés sur Abilène (à gauche) et Eurorings (à droite).*

Comme attendu, le nombre de paquets total induits par le multi-*unicast* est indépendant du nombre de routeurs Xcast déployés, puisque le multi-*unicast* n'utilise pas le protocole

11. Comme indiqué précédemment, le mécanisme utilisant les tunnels n'est pas automatique.

12. Sur des réseaux de taille importante, nous n'avons pas pu mettre en œuvre la stratégie optimale, c'est pourquoi les résultats de cette stratégie n'apparaissent pas sur les courbes d'Eurorings.

13. Dans le paragraphe 3.1.6, cette stratégie de placement des routeurs Xcast sera appelée la stratégie aléatoire.

Xcast.

Lorsque peu de routeurs sont Xcast, le mécanisme X2U prématuré a des performances voisines de celles du multi-*unicast*. L'intuition que X2U prématuré se comporte bien dans le cas des réseaux avec un fort déploiement de Xcast est vérifiée : dès lors que 5 des 11 routeurs d'Abilène sont Xcast, X2U prématuré génère 17 % de paquets de moins que le *multi-unicast*. Pour Eurorings, il faut déployer 25 routeurs pour que le X2U prématuré génère 17 % de paquets de moins que le multi-*unicast*.

Les tunnels semi-perméables génèrent très peu de paquets : 41 % de moins sur Abilène que le multi-*unicast* quand aucun routeur Xcast n'est déployé, et 52 % de moins sur Eurorings. Il apparaît que ce mécanisme est assez peu dépendant de la proportion de routeurs Xcast.

Le nombre d'examens de tables *unicast*. Le nombre d'examens de tables *unicast* est une mesure du temps de traitement global des paquets. L'examen de la liste des destinataires l d'un paquet Xcast par un routeur Xcast nécessite $|l|$ examens de la table *unicast*. L'acheminement d'un paquet *unicast* quant-à lui ne nécessite qu'un examen de la table *unicast* par routeur traversé. La figure 3.17 montre le nombre d'examens de la table *unicast* en fonction du nombre de routeurs Xcast.

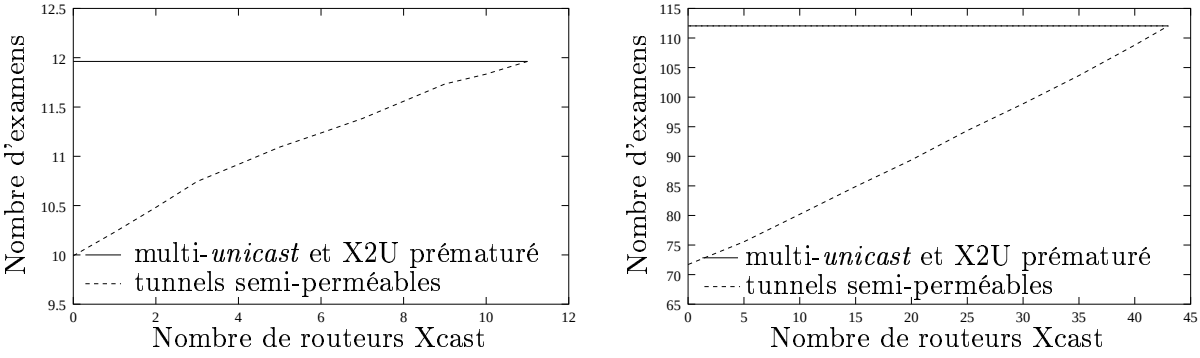


Figure 3.17 – *Évolution du nombre d'examens de la table unicast en fonction du nombre de routeurs Xcast déployés sur Abilène (à gauche) et Eurorings (à droite).*

Les mécanismes multi-*unicast* et X2U prématuré examinent autant de fois la table *unicast*. En effet, examiner un en-tête *unicast* par paquet *unicast* généré par le mécanisme multi-*unicast* ou examiner un en-tête *unicast* par destinataire contenu dans l'en-tête Xcast des paquets induits par le mécanisme X2U prématuré revient au même.

En revanche, l'utilisation de tunnels semi-perméables réduit considérablement le nombre d'en-têtes *unicast* examinés, surtout dans le cas où peu de routeurs Xcast sont déployés. Lorsque peu de routeurs Xcast sont présents, l'acheminement d'un paquet Xcast ne nécessite dans les routeurs non Xcast qu'un seul examen de la table *unicast* par paquet (correspondant à la recherche du routeur suivant pour le premier destinataire de la liste). Le peu de paquets générés par le mécanisme de tunnels semi-perméables accentue ce phénomène. Lorsque aucun routeur Xcast n'est déployé dans le réseau, le mécanisme de tunnels semi-perméables fait 17 % d'examens de la table *unicast* de moins que le mécanisme multi-*unicast* sur Abilène et 36 % de moins sur Eurorings.

La longueur du chemin pour atteindre les destinataires. La longueur maximale du chemin emprunté par les données pendant leur acheminement est une approximation du délai de réception pour les destinataires. Cette approximation ne prend pas en compte la latence introduite par les files d'attentes des routeurs ou par le temps de traitement d'un paquet. La figure 3.18 montre la longueur maximale du chemin[14] pour atteindre les destinataires par les trois mécanismes, en fonction du nombre de routeurs Xcast.

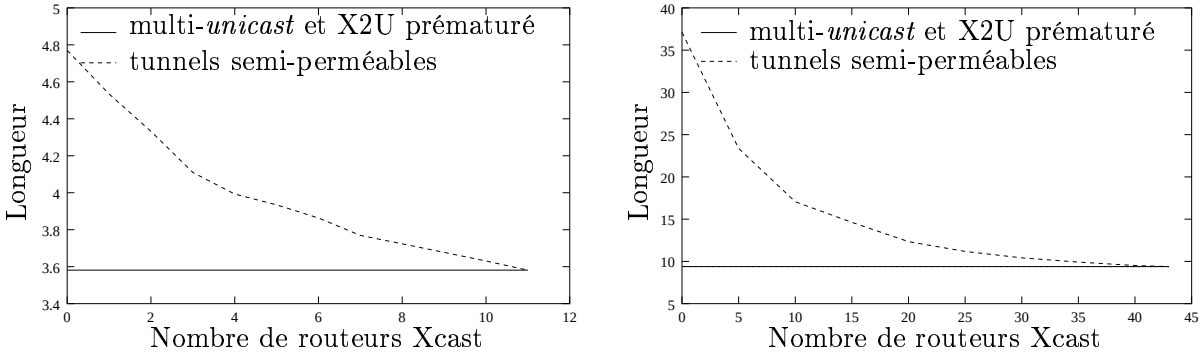


Figure 3.18 – *Évolution de la longueur maximale du chemin pour atteindre les destinataires en fonction du nombre de routeurs Xcast déployés sur Abilène (à gauche) et Eurorings (à droite).*

Utilisant tous les deux le routage *unicast*, donc les plus courts chemins de la source aux destinataires, les mécanismes X2U prématuré et multi-*unicast* offrent les mêmes performances.

L'utilisation de tunnels semi-perméables augmente la longueur maximale de réception des données par construction. Dans le pire des cas, lorsque aucun routeur Xcast n'est déployé, la longueur maximale pour atteindre un récepteur avec les tunnels semi-perméables est 33 % plus grande qu'avec le multi-*unicast* sur Abilène, et plus de quatre fois plus grande sur Eurorings. Lorsque seulement 4 des 11 routeurs d'Abilène sont Xcast, la longueur maximale avec les tunnels semi-perméables a un surcoût de 11 % par rapport au multi-*unicast*. On voit que le déploiement de peu de routeurs réduit beaucoup ce surcoût.

Un récapitulatif. La première conclusion de notre analyse est que le mécanisme X2U prématuré se comporte mieux que le mécanisme multi-*unicast*. Cette conclusion justifie *a posteriori* l'utilisation du protocole Xcast par rapport au multi-*unicast*.

Lorsque plus de la moitié des routeurs sont Xcast, le mécanisme X2U prématuré et le mécanisme de tunnels semi-perméables ont des performances voisines. En revanche, lorsque peu de routeurs Xcast sont déployés, le mécanisme X2U prématuré garantit un chemin court pour atteindre les destinataires, au détriment du nombre de paquets total. *A contrario*, le mécanisme de tunnels semi-perméables garantit un nombre de paquets très faible, mais augmente sensiblement la longueur du chemin pour atteindre les destinataires.

Le mécanisme de tunnels semi-perméables semble le plus approprié aux réseaux contenant peu de routeurs Xcast, en raison du faible nombre de paquets générés, de sa stabilité vis-à-vis

14. En fait, il s'agit de la moyenne des 10 longueurs maximales sur l'ensemble des simulations.

du nombre de paquets générés et du faible nombre d'examens de la table *unicast*.

Stratégies de déploiement des routeurs Xcast

Dans ce paragraphe, nous nous intéressons à déployer un minimum de routeurs Xcast sur des routeurs stratégiques du réseau afin de maximiser le gain de performance. Nous proposons quatre stratégies heuristiques de déploiement des routeurs Xcast que nous comparons à une stratégie de placement optimale. Puis, nous analysons l'impact de ces stratégies sur les performances des trois mécanismes suivants : multi-*unicast*, X2U prématuré et tunnels semi-perméables.

Les quatre stratégies heuristiques de déploiement des routeurs Xcast que nous proposons sont : la stratégie par degrés, la stratégie interne, la stratégie externe et la stratégie aléatoire. Nous comparons ces stratégies heuristiques à la stratégie optimale. Nous décrivons ces cinq stratégies dans ce paragraphe.

La stratégie par degré. Elle consiste à déployer prioritairement Xcast sur les routeurs du réseau ayant un fort degré. Cette stratégie exploite le fait qu'un paquet Xcast arrivant en un routeur Xcast de fort degré a de grandes chances d'être découpé en un grand nombre de paquets. La liste de priorité de déploiement est triée par ordre décroissant de degré des routeurs.

La stratégie interne et la stratégie externe. La stratégie interne et la stratégie externe utilisent la notion de centralité d'un routeur. Plus un routeur est central, plus il est proche de tous les autres. Plus formellement, étant donné R l'ensemble des routeurs du réseau, nous définissons la centralité $c(r)$ d'un routeur $r \in R$ comme :

$$c(r) = \frac{1}{|R|} \sum_{n \in R} d(r, n),$$

où $d(r, n)$ représente la longueur du plus court chemin de r à n. Plus $c(r)$ est petit, plus nous considérons que le routeur r est central.

La stratégie de déploiement interne consiste à déployer prioritairement Xcast sur les routeurs internes du réseau. Un routeur interne a une position centrale : de nombreux plus courts chemins vont passer par lui. Comme le protocole Xcast achemine les paquets selon un plus court chemin, beaucoup de paquets vont passer par ces routeurs internes. La liste de priorité de déploiement est triée par ordre croissant de centralité.

La stratégie de déploiement externe est diamétralement opposée à la stratégie de déploiement interne. Elle consiste à déployer prioritairement Xcast sur les routeurs situés près des frontières du réseau. L'idée est de déployer le protocole Xcast sur les routeurs du premier *mile*. La liste de priorité de déploiement est triée par ordre décroissant de centralité.

La figure 3.19 présente la centralité des routeurs du réseau Abilène. La figure 3.20 présente la centralité des routeurs du réseau Eurorings. Sur ces deux figures, plus le nœud représentant un routeur est clair, plus le nœud est central.

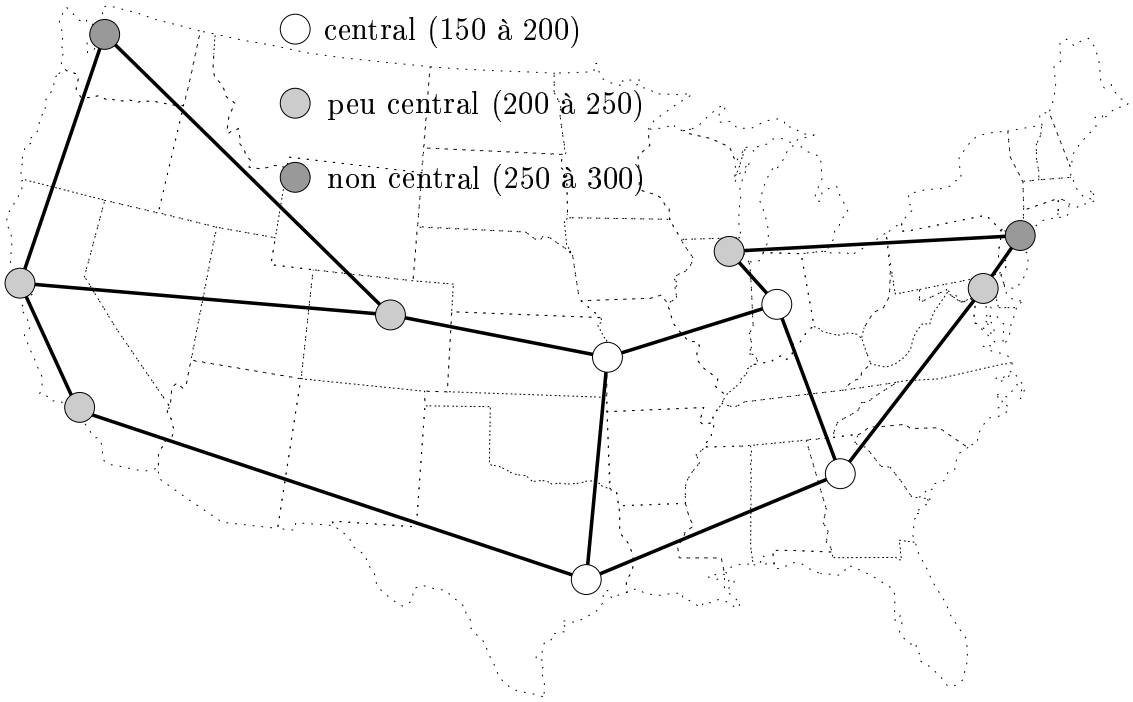


Figure 3.19 – *La centralité des routeurs d'Abilène.*

La stratégie aléatoire. Elle consiste à choisir aléatoirement les routeurs sur lesquels déployer le protocole Xcast. Cette stratégie sert en quelque sorte de témoin dans nos simulations.

La stratégie optimale. Les quatre stratégies heuristiques que nous venons de décrire permettent de placer les routeurs Xcast *a priori*, c'est-à-dire sans connaissance particulière sur les groupes *multicast* qui vont apparaître. Pour évaluer la performance de ces heuristiques, nous les comparons à une stratégie de placement optimale.

La stratégie optimale connaît les 1000 groupes *multicast* générés avant de décider du placement des routeurs Xcast. Les routeurs Xcast à déployer sont déterminés en fonction de ces 1000 groupes de manière à minimiser tour à tour le nombre de paquets total, le nombre d'examens de la table *unicast* et la longueur du plus long chemin. Pour trouver le placement optimal, cette stratégie génère exhaustivement toutes les positions possibles des routeurs Xcast. Il est important de noter que cette stratégie optimale n'est pas réalisable en pratique. De plus, le placement correspondant à une stratégie optimale dépend du mécanisme (X2U prématuré ou tunnels semi-perméables) et de la métrique considérée (nombre de paquets total, nombre d'examens de la table *unicast* ou longueur du plus long chemin).

Performance des stratégies

Dans ce paragraphe, nous évaluons la performance des stratégies sur les métriques introduites précédemment. Les paramètres de simulation sont les même que dans le paragraphe précédent.

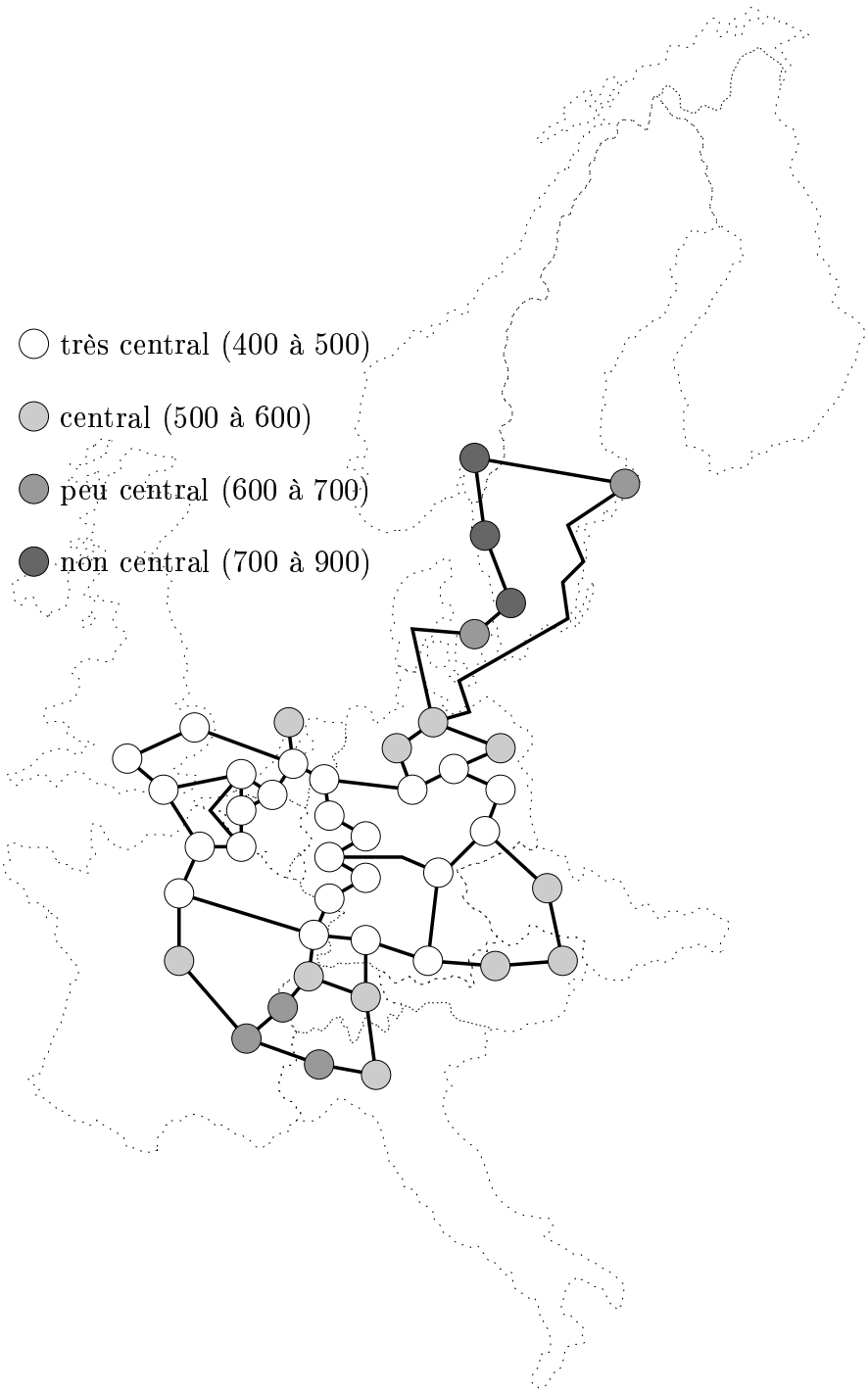


Figure 3.20 – *La centralité des routeurs d'Eurorings.*

Le nombre de paquets total. Les figures 3.21 et 3.22 présentent l'évolution du nombre de paquets total induits par le mécanisme X2U prématuré en fonction du nombre de routeurs sur lesquels Xcast est déployé, selon les cinq stratégies.

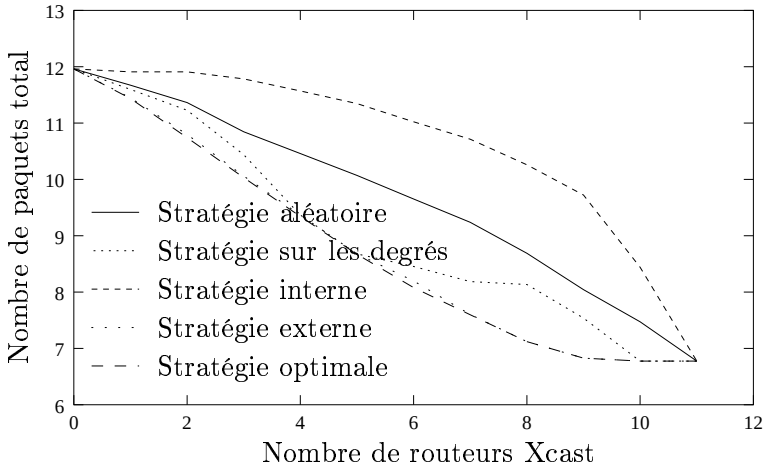


Figure 3.21 – *Nombre de paquets total induits par X2U prématuré, sur Abilène.*

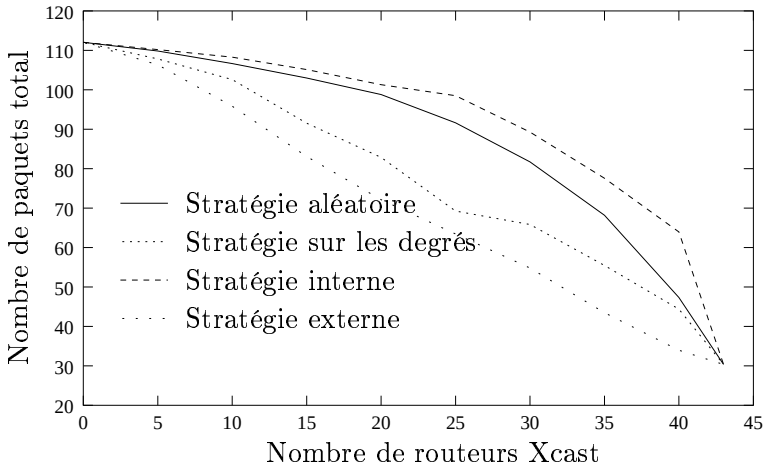


Figure 3.22 – *Nombre de paquets total induits par X2U prématuré, sur Eurorings.*

Comme attendu, nous remarquons que le nombre de paquets total induits par X2U prématuré décroît avec le nombre de routeurs Xcast, quelque soit la stratégie employée. La stratégie interne est plus mauvaise que la stratégie aléatoire ; en effet, X2U prématuré duplique les paquets dès le premier routeur non Xcast rencontré. Comme les premiers routeurs rencontrés ont tendance à ne pas être des routeurs centraux, et donc à ne pas être des routeurs Xcast, X2U prématuré se comporte très mal. *A contrario*, X2U prématuré se comporte très bien pour la stratégie externe ; nous pouvons nous en convaincre par le même raisonnement. La stratégie sur les degrés, quant-à elle, a un comportement voisin de la stratégie externe pour X2U prématuré. Ce résultat vient du fait qu'un nœud r ayant un fort degré a de grandes chances d'être central ; puisque r est de fort degré, il a beaucoup de voisins

directs. Pour chaque voisin direct r_i de r, $d(r_i, r) = 1$. Cette faible distance à un grand nombre de voisins directs diminue d'autant la valeur de $c(r)$. De même, il y a de grandes chances que le routeur r ait beaucoup de voisins à distance 2 et ainsi de suite. Ainsi, $c(r)$ a de grandes chances d'être faible pour les nœuds de fort degré. Dans la suite de l'analyse, nous pouvons remarquer que la stratégie sur les degrés est proche (mais légèrement moins performante) que la stratégie externe. Le comportement du mécanisme X2U prématuré pour les deux stratégies internes et externes est présenté sur la figure 3.23.

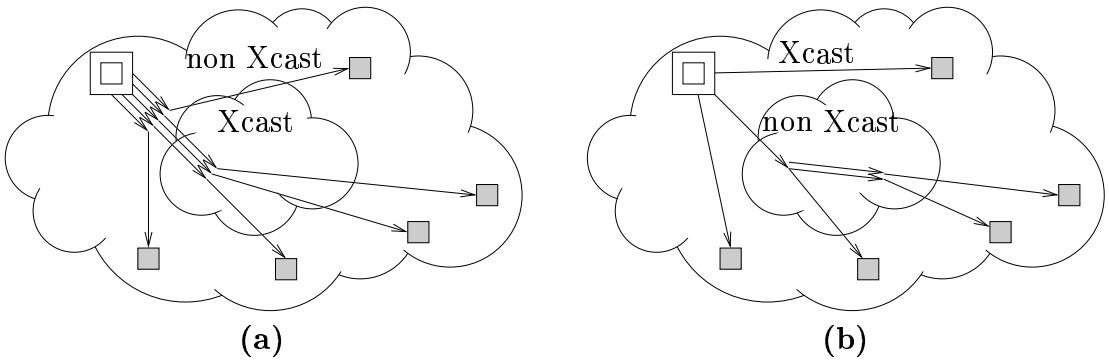


Figure 3.23 – *Comportement du mécanisme X2U prématuré lors d'un déploiement (a) interne et (b) externe.*

La stratégie externe est très proche de la stratégie optimale, à tel point qu'il est difficile de les différencier sur la figure. La stratégie externe est donc une très bonne stratégie. Pour que la stratégie aléatoire atteigne les mêmes performances en nombre de paquets total que la stratégie externe, pour X2U prématuré, il faut dans la plupart des cas déployer entre 2 et 3 routeurs Xcast supplémentaires.

Les figures 3.24 et 3.25 présentent l'évolution du nombre de paquets total induits par le mécanisme de tunnels semi-perméables en fonction du nombre de routeurs sur lesquels Xcast est déployé, selon les cinq stratégies.

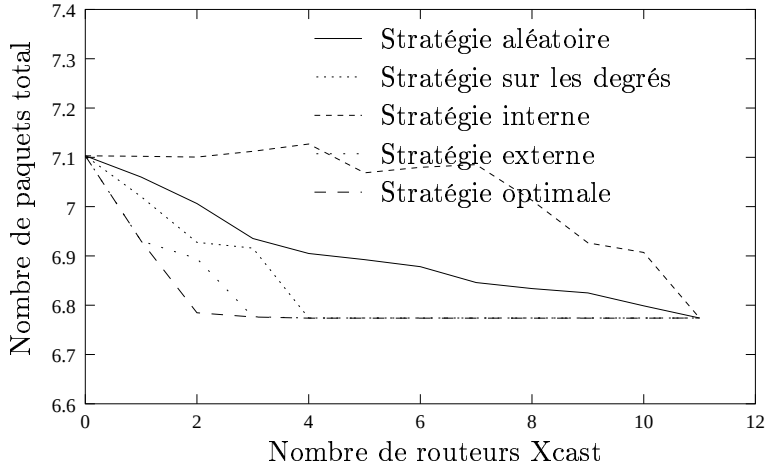


Figure 3.24 – *Nombre de paquets total induits par les tunnels semi-perméables, sur Abilène.*

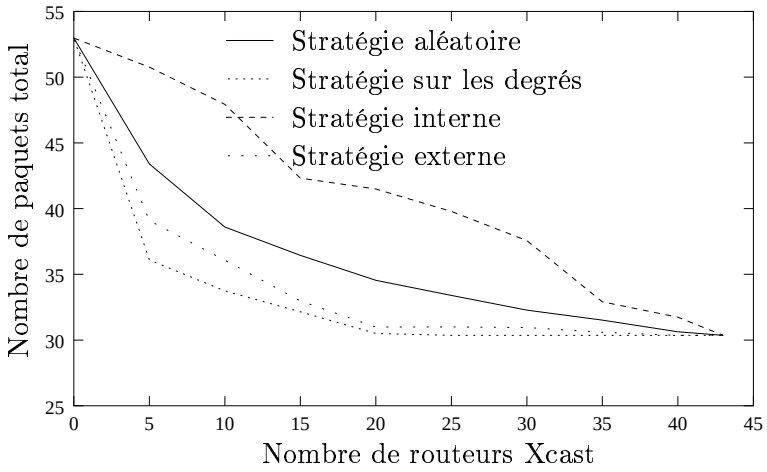


Figure 3.25 – *Nombre de paquets total induits par les tunnels semi-perméables, sur Eurorings.*

Comme nous l'avons indiqué précédemment, le nombre de paquets total induits par le mécanisme de tunnels semi-perméables est assez peu sensible au nombre de routeurs Xcast déployés. Les différences entre les deux stratégies sont peu importantes. On peut toutefois expliquer le bon comportement de la stratégie externe grâce à la figure 3.26. La stratégie externe a tendance à découper rapidement les paquets Xcast à destination de beaucoup de destinataires en sous-paquets (cela est fait dans la partie déployée). Ces sous-paquets vont être rapidement acheminés à leur destinataires. Pour la stratégie interne, les paquets vont tout d'abord être acheminés au premier destinataire, puis au deuxième et ainsi de suite, tant qu'ils ne passent pas par un routeur central Xcast. Ainsi, la stratégie interne va générer davantage de paquets.

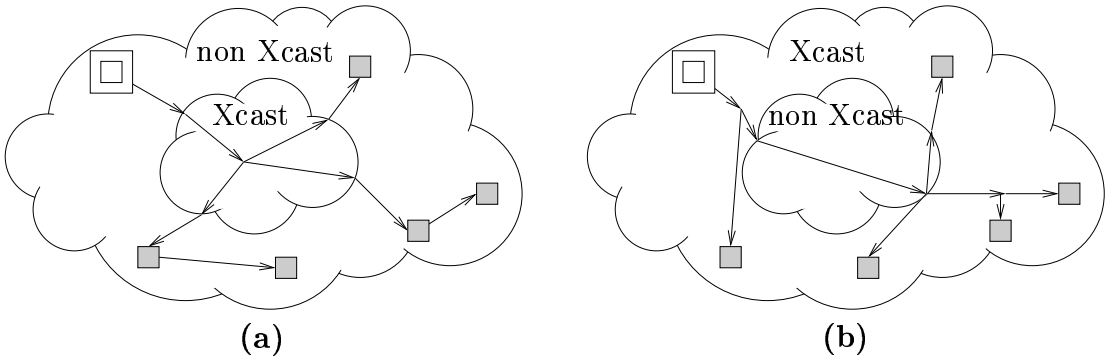


Figure 3.26 – *Comportement du mécanisme de tunnels semi-perméables lors d'un déploiement (a) interne et (b) externe.*

Encore une fois, la stratégie optimale et la stratégie externe sont très proches. Pour que la stratégie aléatoire atteigne les mêmes performances en nombre de paquets total que la stratégie externe, pour les tunnels semi-perméables, il faut déployer entre 2 et 7 routeurs Xcast supplémentaires.

La longueur du chemin pour atteindre les destinataires. La longueur du chemin induit par les mécanismes X2U prématuré et multi-*unicast* est indépendant du nombre et de la localisation des routeurs Xcast dans le réseau.

Les figures 3.27 et 3.28 présentent l'évolution de la longueur maximale du chemin pour atteindre les destinataires induit par le mécanisme de tunnels semi-perméables en fonction du nombre de routeurs Xcast, selon les cinq stratégies.

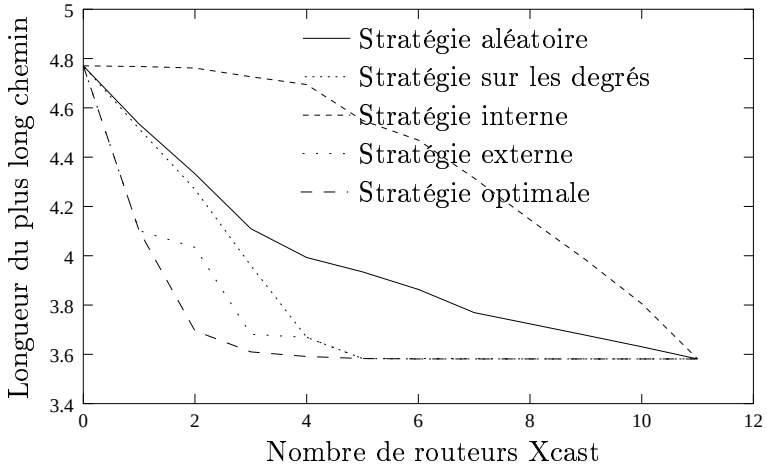


Figure 3.27 – *Longueur maximale du chemin pour atteindre les destinataires induit par les tunnels semi-perméables, sur Abilène.*

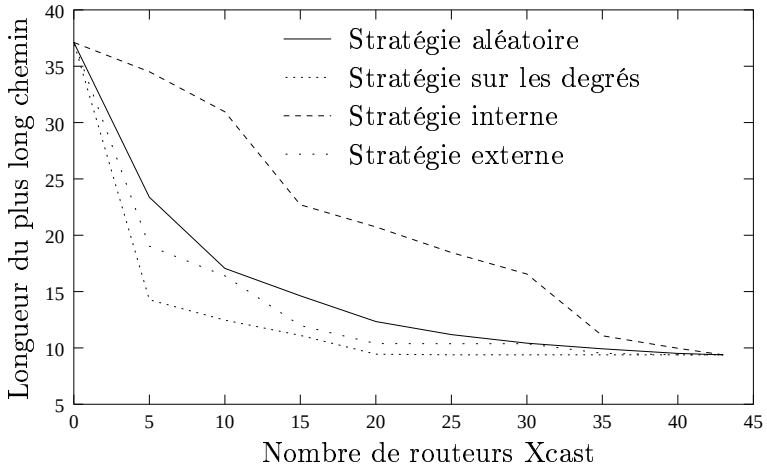


Figure 3.28 – *Longueur maximale du chemin pour atteindre les destinataires induit par les tunnels semi-perméables, sur Eurorings.*

La stratégie interne a un mauvais comportement. En fait, dans un réseau où peu de routeurs sont Xcast, l'acheminement à la première destination a de grandes chances d'utiliser des plus courts chemins plutôt que des arbres. Quelques plus courts chemins entre destinations vont avoir tendance à éviter les routeurs centraux et ainsi les routeurs Xcast déployés ne vont pas voir certains paquets Xcast circuler. Le bon comportement de la stratégie externe

confirme cette remarque, les plus courts chemins exploitant les routeurs non centraux. La stratégie sur les degrés reste encore une fois proche de la stratégie externe.

La stratégie optimale et la stratégie externe sont très proches. Il convient d'insister sur le fait que la stratégie externe trouve un placement des routeurs commun à toutes les métriques alors que la stratégie optimale choisit un placement potentiellement différent pour chaque métrique. Pour que la stratégie aléatoire atteigne les mêmes performances en longueur maximale du chemin pour atteindre les destinataires que la stratégie externe, pour les tunnels semi-perméables, il faut déployer entre 1 et 5 routeurs Xcast supplémentaires.

Le nombre d'examens de la table *unicast*. Le nombre d'examens de la table *unicast* induits par les mécanismes X2U prématuré et multi-*unicast* est indépendant du nombre et de la localisation des routeurs Xcast dans le réseau.

Les figures 3.29 et 3.30 présentent l'évolution du nombre d'examens de la table *unicast* induits par les tunnels semi-perméables.

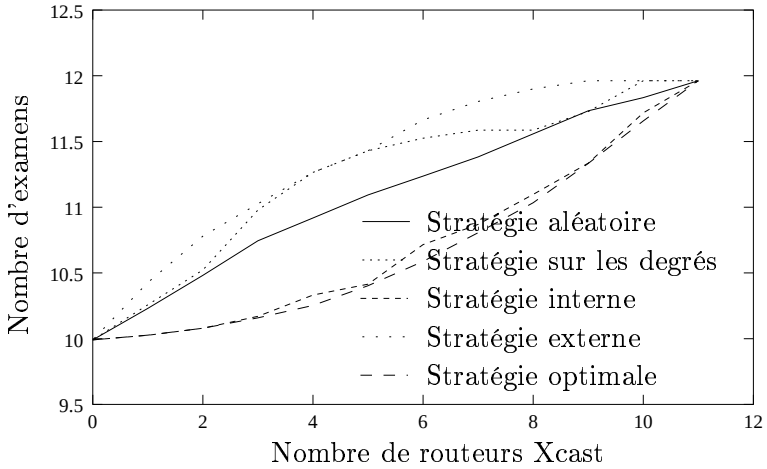


Figure 3.29 – *Nombre d'examens de la table unicast induits par les tunnels semi-perméables, sur Abilène.*

La meilleure stratégie est cette fois la stratégie interne. Dès qu'un paquet Xcast passe par un routeur central, il est découpé en sous-paquets et chaque sous-paquet est envoyé dans la bonne direction. Ainsi, chaque sous-paquet va être confiné à une sous-partie du réseau et ne va pas circuler un grand nombre de fois d'un bout à l'autre du réseau. La stratégie externe est très mauvaise. En effet, le découpage d'un paquet Xcast en sous-paquets est très limité puisque les routeurs Xcast ne sont pas les routeurs centraux. Peu de sous-paquets vont être générés. Ces sous-paquets vont avoir tendance à faire plusieurs aller-retours entre des parties éloignées du réseau, ce qui augmente le nombre d'examens de la table *unicast*. L'écart du nombre d'examens de la table *unicast* entre les différentes stratégies reste cependant limité puisque la différence porte au plus sur 2 examens, soit environ 17 %.

Le nombre d'examens de la table *unicast* est très proche pour la stratégie externe et pour la stratégie optimale. La stratégie externe est donc efficace. Pour que la stratégie aléatoire atteigne les mêmes performances en nombre d'examens de la table *unicast* que la straté-

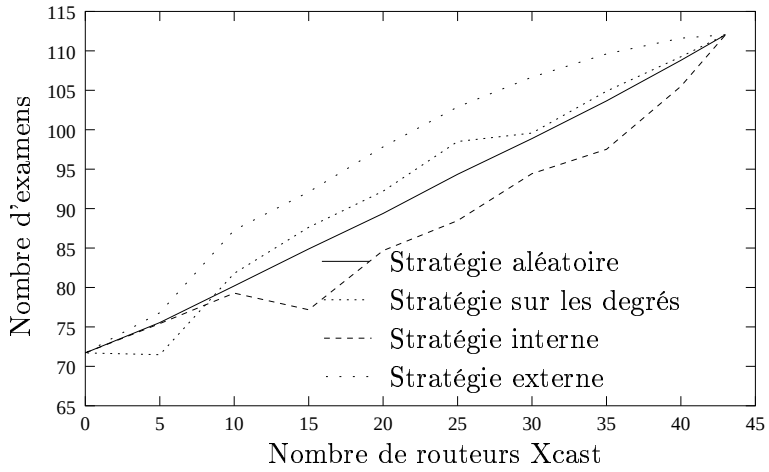


Figure 3.30 – *Nombre d'examens de la table unicast induits par les tunnels semi-perméables, sur Eurorings.*

gie interne, pour les tunnels semi-perméables, il faut déployer entre 2 et 3 routeurs Xcast supplémentaires.

Un récapitulatif. La conclusion principale de notre analyse est que la stratégie externe, qui consiste à déployer les routeurs Xcast en priorité sur les routeurs éloignés du centre du réseau, offre des performances très proches de la stratégie optimale. Pour que la stratégie aléatoire atteigne les mêmes performances que la stratégie externe, il faut souvent déployer plus de 2 routeurs Xcast supplémentaires, c'est-à-dire augmenter le déploiement de Xcast (et donc le coût de ce déploiement) sur l'ensemble du réseau de 18 %.

La stratégie interne qui consiste à déployer les routeurs Xcast en priorité sur les routeurs centraux est à éviter, puisqu'elle est significativement moins efficace que la stratégie aléatoire pour la plupart des métriques. En pratique, cette stratégie est difficile à implémenter, puisqu'une grande partie du routage réalisé par les routeurs internes se fait au niveau matériel.

3.2 Le routage *multicast* explicite arborescent

La principale limite du routage *multicast* explicite plat est le temps de traitement des paquets au sein des routeurs. En effet, chaque routeur traversé doit examiner toute la liste des destinataires. Ce temps de traitement est systématique, même pour les routeurs n'étant pas routeurs de branchement de l'arbre. Le routage explicite arborescent propose une solution à ce problème.

Le routage *multicast* explicite arborescent consiste à mémoriser explicitement l'arbre dans l'en-tête des paquets plutôt que la liste des membres. Deux désavantages par rapport au routage *multicast* explicite plat apparaissent : d'une part l'en-tête des paquets est plus important et, d'autre part, mémoriser explicitement l'arbre oblige à le calculer, donc nécessite la connaissance de la topologie. En général, une connaissance partielle de la topologie permet de

déterminer les nœuds significatifs de l'arbre. En revanche, chaque routeur n'a plus qu'à analyser une petite partie de l'en-tête pour décider du routage, ce qui réduit considérablement le temps de traitement des paquets.

Le protocole ERM, le protocole Linkcast et le protocole TBXcast sont des protocoles de routage *multicast* explicite arborescent. Nous allons à présent les décrire.

3.2.1 Le protocole ERM

Le protocole ERM (pour *Explicit Route Multicast*) est décrit dans [BFST00].

Description du protocole ERM

Dans un domaine ERM, on distingue les routeurs *ingress* et les routeurs *egress*. Les routeurs *ingress* sont les routeurs par lesquels le trafic entre dans un domaine, tandis que les routeurs *egress* sont les routeurs par lesquels il sort. Les routeurs *ingress* seront appelés routeurs sources tandis que les routeurs *egress* seront appelés routeurs destinataires.

Pour joindre un groupe, un routeur destinataire envoie un paquet `trace` vers le routeur source. Chaque routeur ERM sur le passage de ce paquet y inclus son adresse IP[15]. En combinant les informations contenues dans les paquets de trace de tous les routeurs destinataires, le routeur source est capable de construire l'arbre des plus courts chemins inverses. L'arbre est un arbre réduit (voir la définition 9). Les arbres réduits sont utilisés dans plusieurs propositions de *multicast* traditionnel comme Reunite [SENZ00], HBH [CFD01] ou SEM [BC03].

Définition 9 (Arbre réduit). *Un arbre réduit est un arbre dans lequel chaque nœud représente un routeur significatif, c'est-à-dire un routeur de branchement ou un routeur destinataire.*

Lorsqu'un routeur r reçoit un paquet ERM, il l'achemine vers la racine de l'arbre encodé. Si r est la racine de l'arbre, il parcourt l'en-tête pour trouver les sous-arbres de la racine. Pour chaque sous-arbre, r envoie un paquet ERM vers la racine de ce sous-arbre, en limitant chaque en-tête au sous-arbre en question. De plus, r extrait du paquet initial le couple (s, g) et recherche dans une table interne s'il est routeur destinataire pour ce groupe ou pas. Si c'est le cas, il achemine les données en dehors du domaine.

Le protocole ERM sur un exemple

Dans la suite de ce chapitre, nous représenterons un arbre réduit encodé au moyen des règles suivantes :

- Si n est un routeur, $n : ()$ représente un arbre encodé.
- Si $a_1, \ldots, a_p$ sont des arbres encodés et n un routeur, $n : (a_1, \ldots, a_p)$ représente un arbre encodé.

Pour ne pas alourdir la notation, $n : ()$ sera remplacé par n, sauf quand il s'agira de l'arbre complet. Par exemple, l'arbre $n_1 : (n_2 : (), n_3 : ())$ sera représenté $n_1 : (n_2, n_3)$.

15. En fait, c'est la plus petite adresse IP des interfaces du routeur qui est incluse. Cette technique permet d'identifier les routeurs.

Considérons le réseau représenté sur la figure 3.31 et le groupe g formé d'une source s et des sept destinataires d_1, d_2, d_3, d_4, d_5, d_6 et d_7. Le groupe g est identifié par une adresse *multicast*. Le domaine ERM que nous considérons est constitué des routeurs r_i.

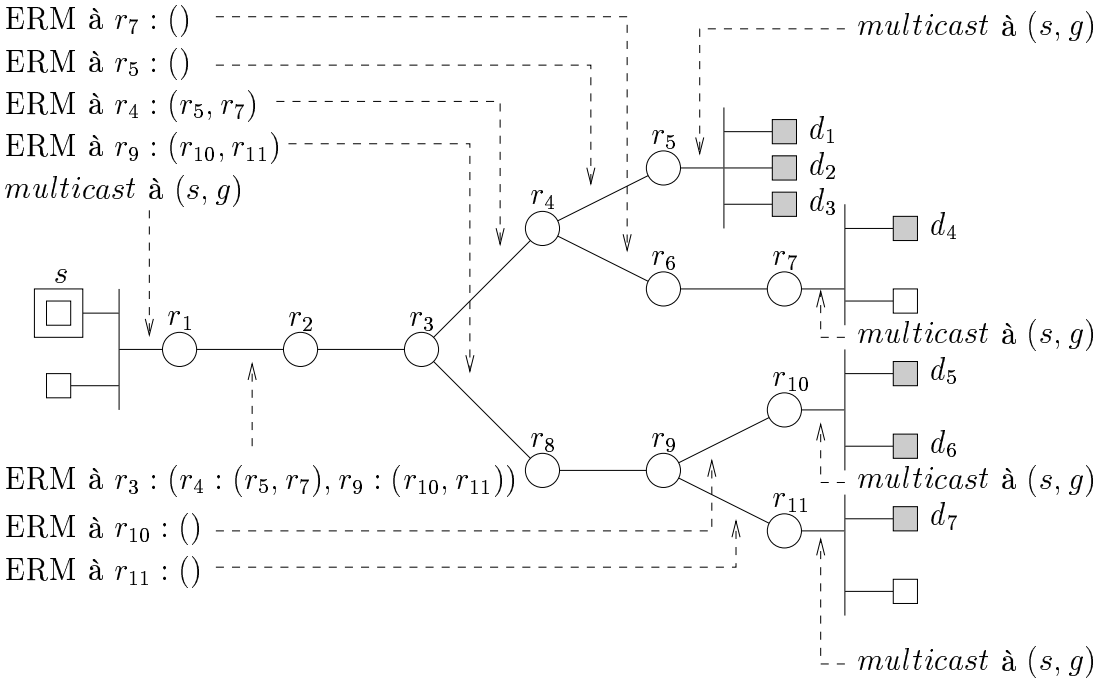


Figure 3.31 – *L'acheminement des paquets dans ERM.*

Pour émettre des données au groupe g, la source s crée un paquet *multicast* contenant les données et dont l'adresse de destination est l'adresse du groupe g. s envoie ce paquet sur son réseau local.

Quand r_1 reçoit ce paquet *multicast*, il utilise une table de correspondance interne qui associe à (s, g) la liste des routeurs destinataires $(r_5, r_7, r_{10}, r_{11})$. À partir de sa connaissance de la topologie du réseau, r_1 construit un arbre réduit couvrant ces destinataires dont la racine est r_1. L'arbre réduit obtenu est $r_3 : (r_4 : (r_5, r_7), r_9 : (r_{10}, r_{11}))$. Cet arbre est encodé dans un paquet ERM contenant une copie des données initiales et ce paquet est envoyé en *unicast* vers la racine du sous-arbre, r_3.

Quand r_2 reçoit le paquet ERM, il détecte qu'il n'est pas la racine de l'arbre encodé et achemine le paquet vers la racine de l'arbre, r_3.

Quand r_3 reçoit le paquet ERM, il détecte qu'il est la racine de l'arbre encodé. r_3 recherche dans l'en-tête la racine de chaque sous-arbre en parcourant l'en-tête. Le premier sous-arbre $r_4 : (r_5, r_7)$ est encodé dans un paquet ERM contenant une copie des données initiales et est envoyé vers r_4. Le second sous-arbre $r_9 : (r_{10}, r_{11})$ est lui aussi encodé dans un paquet ERM contenant une copie des données initiales et est envoyé vers r_9.

Quand r_5 reçoit le paquet ERM, il détecte qu'il est la racine de l'arbre encodé. En extrayant (s, g) du paquet original encapsulé, il détecte qu'il est routeur destinataire pour ce groupe. r_5 envoie un paquet *multicast* contenant les données sur son réseau local.

Quand le membre d_1 reçoit le paquet *multicast* à destination du groupe (s, g), il peut en extraire les données.

L'encodage de l'arbre

Dans ERM, seuls les nœuds de branchement et les nœuds destinataires de l'arbre sont mémorisés. À chaque nœud est associé un pointeur sur son père. La figure 3.32 présente l'encodage ERM de l'arbre de la figure 3.31.

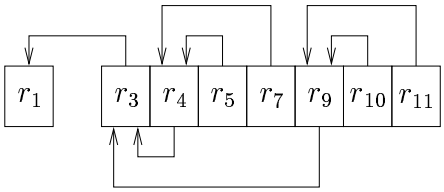


Figure 3.32 – *L'encodage de l'arbre de la figure 3.31 selon ERM.*

Les adresses IPv4 étant mémorisées sur 4 octets et le pointeur sur un nœud père étant mémorisé sur 1 octet, une entrée occupe 5 octets. L'encodage de l'arbre de la figure 3.31 nécessite 7 entrées de 5 octets, soit 35 octets. Nous pouvons noter que l'entrée correspondant à la racine r_1 n'est pas comptabilisée puisqu'elle est enregistrée dans l'en-tête IP du paquet et non pas dans l'en-tête ERM du paquet. Cet encodage présente cependant un désavantage important : pour trouver la racine des sous-arbres d'un nœud, il est nécessaire de parcourir tout l'en-tête.

La gestion de la dynamicité des groupes

À chaque fois qu'un routeur destinataire désire adhérer au groupe, il doit envoyer un paquet de trace. En recevant ce paquet, la source combine les informations des paquets `trace` avec celles reçues des autres routeurs destinataires. Seules les adresses des routeurs ERM apparaissent dans les paquets de `trace`. La vision partielle de la topologie du protocole ERM ne contient donc aucun routeur non ERM.

De même, lorsqu'un routeur destinataire désire quitter le groupe, il prévient la source qui combine les informations de trace pour calculer un nouvel arbre. Cette procédure est nécessaire pour supprimer de l'encodage les routeurs qui ne sont plus significatifs.

La sécurité dans ERM

Les protocoles *multicast* explicites arborescents posent des problèmes de sécurité supplémentaires par rapport aux protocoles *multicast* explicites plats. Si nous considérons le réseau de la figure 3.31, un routeur source compromis peut encoder l'arbre $r_5 : (r_1 : (r_5 : (r_1 : (\ldots))))$ afin de congestionner les liens entre r_1 et r_5.

3.2.2 Le protocole Linkcast

Le protocole Linkcast est décrit dans [BMSBY04]. Ce protocole est dérivé du protocole ERM. Sa différence principale avec ERM est dans l'encodage de l'arbre dans les paquets. Dans la suite, nous ne décrivons que cette particularité.

Description du protocole Linkcast

Le protocole Linkcast vise à réduire la charge due à l'encodage de l'arbre dans l'en-tête des paquets. Plutôt que de stocker dans l'en-tête des adresses IP, Linkcast mémorise des identifiants d'interfaces. Ces identifiants sont stockés sur un octet, un routeur ayant rarement plus de 256 interfaces de sortie. Deux encodages sont proposés : l'encodage SBM (pour *Sparse Branching Mode*) et l'encodage DBM (pour *Dense Branching Mode*). Ces deux encodages sont expliqués au travers de l'exemple représenté sur la figure 3.33, où s est la source du groupe g constitué des membres d_1, d_2, d_3 et d_4.

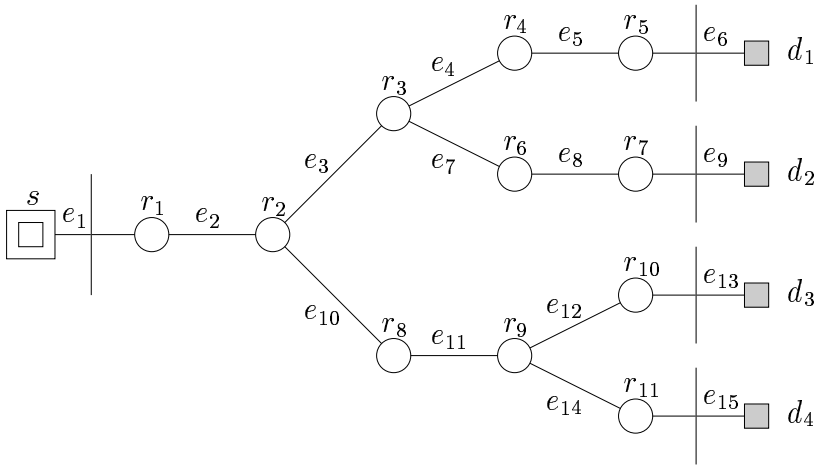


Figure 3.33 – *Un arbre Linkcast à encoder.*

L'encodage SBM est adapté pour des arbres ayant beaucoup de routeurs de relais (c'est-à-dire des routeurs qui ne sont pas de branchement). Dans l'encodage SBM, seules les interfaces correspondant à l'arbre sont encodées. Plusieurs pointeurs sont associés à chaque interface. Afin de réduire la taille de l'encodage au minimum, les interfaces sont organisées de manière à éliminer le besoin de pointeurs autour des routeurs de relais. La figure 3.34 montre l'encodage de l'arbre de la figure 3.33, avec la représentation en octets. Notons T un arbre Linkcast, $aretes(T)$ le nombre d'arêtes de T dans le domaine, $br(T)$ le nombre de nœuds de branchement de T et $bf(T)$ la somme des degrés sortant des nœuds de branchement. La taille en octets de l'encodage SBM de T peut s'écrire $aretes(T) + br(T) + bf(T)$. Sur l'exemple de la figure 3.34, la taille de l'encodage est de $10 + 3 + 6 = 19$ octets.

L'encodage DBM est adapté pour des arbres ayant beaucoup de routeurs de branchement. Dans l'encodage DBM, chaque interface est associée au routeur à son extrémité, à l'exception des routeurs destinataires qui sont omis (sur notre exemple : r_5, r_7, r_{10} et r_{11}). Les routeurs sont marqués afin d'être distingués des interfaces. Les interfaces sortantes d'un routeur sont mémorisées entre ce routeur et le routeur suivant. La figure 3.34 montre l'encodage de l'arbre de la figure 3.33, avec la représentation en octets. Nous notons que les entrées marquées (représentant les routeurs) stockent un identifiant unique pour le routeur sur 1 octet et non pas son adresse IP. Notons T un arbre Linkcast, $aretes(T)$ le nombre d'arêtes de T, $re(T)$ le nombre de nœuds de relais de T et $bf(T)$ la somme des degrés sortant des nœuds de branchement. La taille en octets de l'encodage DBM de T peut s'écrire $aretes(T) - 1 +$

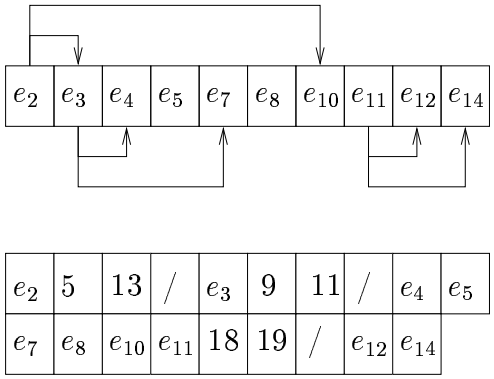


Figure 3.34 – *L'encodage (logique et physique) de l'arbre de la figure 3.33 selon SBM.*

$re(T)+bf(T)$. Sur l'exemple de la figure 3.35, la taille de l'encodage est de $10-1+3+6=18$ octets, les quatre routeurs de relais étant r_4, r_6 et r_8. Dans ce cas, l'encodage DBM est plus petit que l'encodage SBM.

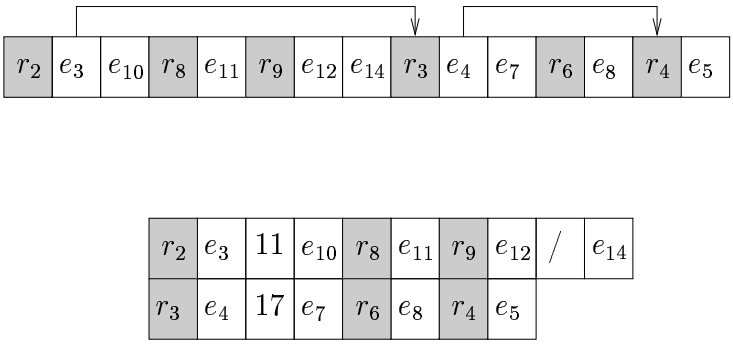


Figure 3.35 – *L'encodage (logique et physique) de l'arbre de la figure 3.33 selon DBM.*

Avantages

L'avantage du protocole Linkcast sur le protocole ERM est que les deux encodages proposés sont souvent très compacts. Ces deux encodages sont interchangeables dynamiquement en fonction de l'évolution de l'arbre de diffusion. À titre comparatif, l'encodage de l'arbre de la figure 3.33 selon ERM nécessite 35 octets et seulement 19 octets selon SBM ou 18 octets selon DBM.

Inconvénients

Le protocole Linkcast souffre de plusieurs désavantages importants par rapport au protocole ERM. Tout d'abord, le routage Linkcast est moins résistant aux pannes que le routage ERM : la panne d'une interface utilisée dans Linkcast déconnecte tous les membres du sous-arbre correspondant, tandis que pour ERM, aucun membre n'est déconnecté tant que la connectivité *unicast* est maintenue.

Mémoriser les interfaces plutôt que les adresses IP des routeurs empêche l'élimination des routeurs de relais. Pour des groupes épars (dans des domaines ayant beaucoup de routeurs), l'encodage SBM et l'encodage DBM peuvent se révéler significativement moins efficaces que l'encodage d'ERM. Pour s'en convaincre, considérons le cas où un seul routeur du domaine est *egress* pour le groupe. Avec ERM, un paquet ERM comportant un arbre vide serait envoyé. Avec SBM ou DBM, l'en-tête Linkcast du paquet serait composé de toutes les interfaces sur le chemin du routeur *ingress* au routeur *egress*. Pour peu que le diamètre du réseau soit suffisamment grand, la taille de l'en-tête peut devenir arbitrairement grande.

Trouver un identifiant unique pour un routeur dans l'encodage DBM pose des problèmes. Entre autres, le nombre maximal de routeurs dans le domaine que Linkcast peut gérer est 128, puisque l'identifiant est sur 7 bits.

Finalement, la source doit connaître l'identification des interfaces des routeurs. Cette information n'est pas usuelle dans les protocoles de routage *unicast* et sa mémorisation induit une surcharge qui incombe au protocole Linkcast.

Les encodages Link+, Link* et Link**

Récemment, plusieurs nouveaux encodages ont été proposés dans [ATK05] : Link+, Link* et Link**. Ces encodages sont plus compacts que les encodages SBM et DBM du protocole Linkcast.

L'encodage Link+ ne stocke pas l'identificateur des nœuds de relais. Un avantage de l'encodage Link+ est que l'opération d'acheminement est peu coûteuse.

L'encodage Link* utilise un encodage basé sur une représentation de l'arbre correctement parenthésée et sur une liste des indices de lien en préordre. Dans l'encodage Link*, les identifiants de liens peuvent être remplacés par des identifiants de nœud (en général si les nœuds sont éloignés). L'inconvénient de l'encodage Link* est que l'opération d'acheminement est relativement coûteuse.

L'encodage Link** réduit l'encodage de Link* en stockant efficacement les liens conduisant à des routeurs de relais. Comme les routeurs de relais sont majoritaires sur les arbres *multicast* d'Internet, l'encodage Link** est plus performant que l'encodage Link*. Dans l'encodage Link**, les identifiants de liens peuvent être remplacés par des identifiants de nœuds. Comme pour l'encodage Link*, l'opération d'acheminement pour l'encodage Link** est relativement coûteuse.

3.2.3 Le protocole TBXcast

Le protocole TBXcast que nous proposons dans ce paragraphe permet une analyse plus approfondie du routage *multicast* explicite arborescent que les protocoles ERM et Linkcast. En effet, dans le protocole ERM, le mécanisme de traces utilisé pour construire l'arbre explicite n'est pas souple. Dans le protocole Linkcast, l'encodage est compact mais il est assez difficile à réaliser et peut échouer quand le diamètre du réseau est grand, comme nous allons le voir.

Nous présentons d'abord les motivations du protocole TBXcast. Puis, nous décrivons les avantages de la segmentation d'un arbre. Enfin, nous détaillons notre protocole. La description initiale du protocole TBXcast se trouve dans [MGC05].

Motivations

L'encodage utilisé par le protocole Linkcast est souvent plus compact que celui du protocole ERM. En revanche, dans les réseaux de grand diamètre, l'encodage du protocole Linkcast échoue quand l'encodage explicite de toutes les arêtes d'un chemin est supérieur au MTU. De plus, les encodages de Linkcast font de nombreuses hypothèses (moins de 128 nœuds dans le réseau, étiquetage des interfaces de sortie des routeurs, etc.) et sont peu robustes aux pannes. Nous pouvons aussi noter que l'encodage d'un arbre en utilisant le protocole ERM peut être très grand et dépasser le MTU.

La taille des paquets peut être limitée à une valeur bien inférieure au MTU pour des raisons de performance. C'est le cas dans certains réseaux sans fils. En effet, des paquets de grande taille mettent beaucoup de temps à être transmis, ce qui provoque potentiellement beaucoup de collisions et donc un grand retard dans les transmissions. Une autre conséquence est que le débit total est réduit. C'est pour cela que les applications peuvent limiter la taille des paquets à une petite valeur.

Finalement, comme l'étude du protocole GXcast a montré qu'il est intéressant de prendre en compte la segmentation pour les protocoles explicites plats, nous voulons voir ce qu'il en est pour les protocoles explicites arborescents.

Segmentation d'un arbre

Nous appelons segmentation l'opération qui consiste à découper un arbre en une forêt couvrant les mêmes membres. Chacun des arbres de cette forêt doit avoir un encodage inférieur à un seuil donné. La segmentation d'un arbre est complexe. Elle dépend de l'encodage, de l'arbre et de la manière dont les membres sont répartis. Le but principal de la segmentation d'un arbre en une forêt est de s'assurer que l'encodage de chaque arbre résultant est limité. Il convient aussi de réduire le surcoût, c'est-à-dire de minimiser la somme des encodages des arbres de la forêt. Cependant, la somme des encodages des arbres d'une forêt est supérieur ou égal à l'encodage de l'arbre initial, comme le montre la figure 3.36. L'arbre initial, représenté sur la figure 3.36(a), a pour encodage $b : (c : (e, f), g : (h, i, j))$. Le coût de l'encodage selon ERM est de 40 octets. Après la segmentation, nous obtenons par exemple les deux arbres de la figure 3.36(b), ayant pour encodages $b : (c : (e, f), g)$ et $g : (h, i, j)$. Le coût de ces encodages est respectivement de 25 et de 20 octets. La somme des coûts des encodages est supérieure au coût de l'encodage de l'arbre initial.

Un autre critère important à minimiser et le volume d'informations transitant dans le réseau. Sur l'arbre initial représenté sur la figure 3.36(a), 9 arêtes sont utilisées par les paquets. Sur la forêt représentée sur la figure 3.36(b), 12 arêtes sont utilisées.

Avant de se préoccuper de la segmentation, nous pouvons chercher à construire un arbre de faible coût et dont l'encodage est faible. D'une part, trouver un arbre de coût minimum est un problème NP-difficile. Il existe des approximations mais l'arbre optimal et les arbres approchées peuvent avoir jusqu'à $n-2$ nœuds de branchement s'il y a n membres. L'encodage de ces arbres serait très coûteux et la segmentation qu'il faudrait réaliser ensuite serait compliquée. D'autre part, trouver un arbre dont l'encodage est minimal revient à trouver un arbre n'ayant pas de nœuds de branchements non membres du groupe. Cependant, le coût d'un tel arbre peut être grand et le délai ressenti par les destinataires très important.

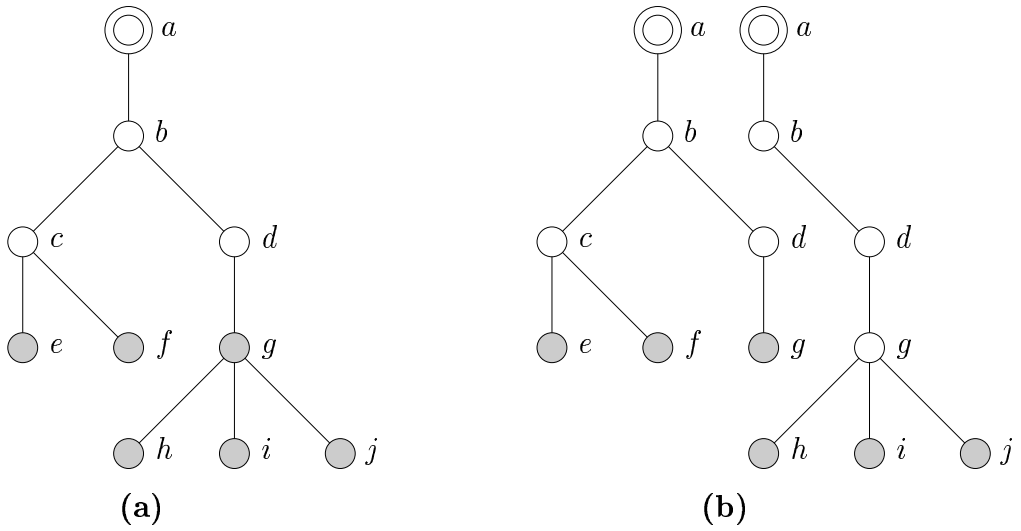


Figure 3.36 – *La somme des encodages des arbres d'une forêt est supérieur ou égal à l'encodage de l'arbre initial.*

3.2.4 L'analyse du protocole TBXcast

Nous venons de voir que la recherche d'une forêt de faible coût telle que l'encodage de chaque arbre soit limité est un problème difficile. C'est pourquoi nous proposons de le résoudre en deux temps : d'abord, nous cherchons à construire un arbre de faible coût propice à la segmentation (c'est-à-dire ayant peu de nœuds de branchement) et, ensuite, nous cherchons à segmenter au mieux cet arbre. Pour chacune de ces parties, nous décrivons des heuristiques.

Construction d'un arbre propice à la segmentation

Habituellement, les arbres construits par les protocoles sont des arbres des plus courts chemins (inverses ou non) ou des arbres approchés de Steiner. Cependant, le nombre de nœuds significatifs dans un arbre des plus courts chemins ou dans un arbre approché de Steiner peut être très grand, voire du même ordre que le nombre de feuilles de cet arbre. C'est pourquoi nous avons développé l'heuristique TMBP qui construit des arbres ayant peu de nœuds de branchement.

Heuristique TMBP. L'heuristique TMBP (pour *TM with Branching Penalty*) est une adaptation de l'heuristique TM visant à construire des arbres ayant peu de nœuds de branchement. Un nœud de relais de l'arbre qui devient un nœud de branchement est pénalisé. Un nœud de branchement de l'arbre dont le degré dans l'arbre augmente n'est pas pénalisé (il est déjà compté dans dans l'en-tête).

En d'autres termes, l'heuristique TMBP suit le même procédé que l'heuristique TM, mais en utilisant une nouvelle distance $\bar{d}$. La distance $\bar{d}(n,t)$ d'un nœud n à un arbre t (en cours

de construction) est définie par :

$$\bar{d}(n,t) = d(n,t) + \begin{cases} 0 & \text{si } n' \text{ est déjà un nœud significatif de } t \\ p & \text{sinon} \end{cases},$$

où $d(n,t)$ est la plus courte distance entre le nœud n et l'arbre t, n' est le nœud de rattachement de n à t par un plus court chemin, et p est une pénalité paramétrable.

Simulations de la phase de construction de l'arbre. Nous avons simulé l'heuristique TMBP sur le réseau Eurorings (comportant 43 nœuds et 55 arêtes). Nous avons comparé les arbres construits par l'heuristique TMBP avec ceux construits par l'heuristique TM sans modification, ainsi qu'à des arbres des plus courts chemins (dénotés sur les figures par SPT, pour *Shortest Path Tree*). Le nombre de nœuds significatifs autorisés est de 20. Nous avons fait varier la taille des groupes de 10 à 35. Chaque point des courbes correspond à la moyenne de 100 simulations. Dans un premier temps, nous avons choisi de fixer le paramètre de TMBP à la valeur 5.

La figure 3.37 présente le coût des arbres construits en fonction de la taille des groupes. Comme attendu, les arbres des plus courts chemins sont plus coûteux que les arbres de faible coût. L'heuristique TMBP trouve des solutions dégradées par rapport à l'heuristique TM puisqu'elle pénalise certains nœuds.

La figure 3.38 présente le nombre de nœuds significatifs des arbres construits. Comme attendu, l'heuristique TMBP, qui pénalise les nœuds de branchement non membres, construit des arbres ayant moins de nœuds significatifs. On remarque que pour cette métrique, les arbres construits par l'heuristique TM sont meilleurs que les arbres des plus courts chemins.

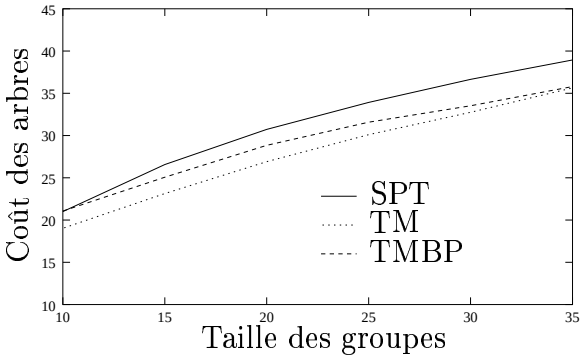


Figure 3.37 – *Coût des arbres construits.*

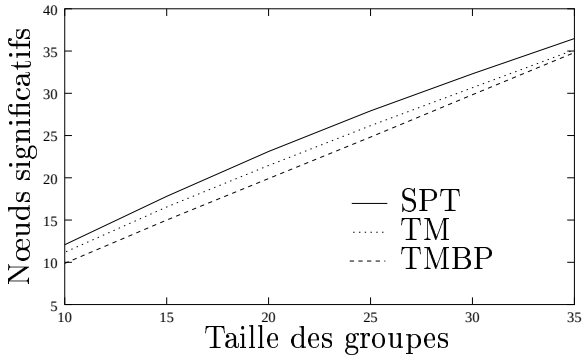


Figure 3.38 – *Nombre de nœuds significatifs des arbres construits.*

Ensuite, nous avons cherché à évaluer l'impact du paramètre p de l'heuristique TMBP. Pour cela, nous avons choisi trois valeurs du paramètre : 1, 3 et 5. Ces valeurs, assez petites, reflètent le choix dans les simulations d'un réseau ayant relativement peu d'arêtes.

Plus le paramètre est élevé, plus les arbres construits sont coûteux (voir la figure 3.39). Simultanément, les grandes valeurs du paramètre de l'heuristique TMBP réduisent le nombre de nœuds significatifs des arbres (voir la figure 3.40).

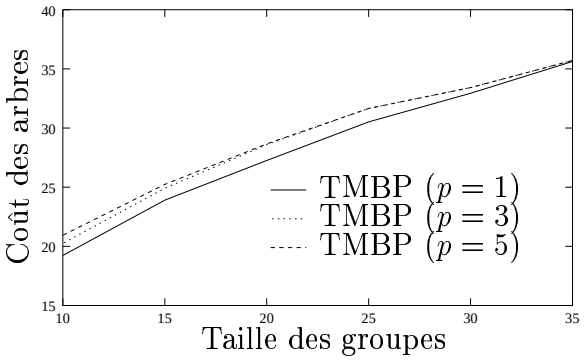


Figure 3.39 – *Coût des arbres construits.*

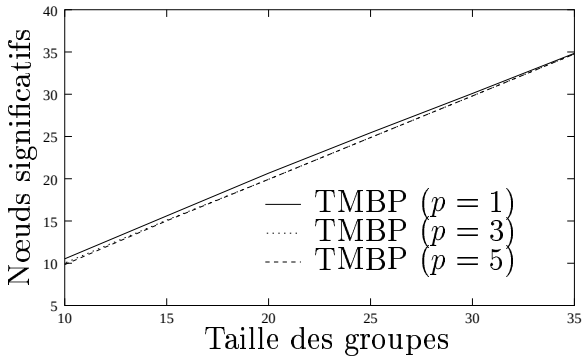


Figure 3.40 – *Nombre de nœuds significatifs des arbres construits.*

Segmentation d'un arbre propice

À présent, nous considérons la segmentation d'un arbre T (obtenu d'une manière quelconque) en une forêt F. Chaque arbre $t \in F$ doit contenir moins de n_p nœuds significatifs. Rappelons qu'un nœud significatif est un nœud de branchement ou un membre du groupe. Nous avons proposé les heuristiques TS, MCPFM et MCPFL.

Segmentation initiale. Les heuristiques de segmentation ne considèrent pas directement l'arbre T mais une forêt. En effet, une première segmentation, dite « segmentation initiale », survient à la source : les paquets à destination des fils de la source empruntent des liens différents. Cette segmentation initiale est indépendante du nombre de nœuds significatifs autorisés par arbre.

La forêt initiale F_T, obtenue par segmentation initiale de l'arbre T, contient autant d'arbres que la racine de T a de fils. Pour chaque fils f de la racine r, la forêt F_T contient le sous-arbre de T enraciné en f auquel est ajouté l'arête (r, f). La figure 3.41 présente cette segmentation initiale.

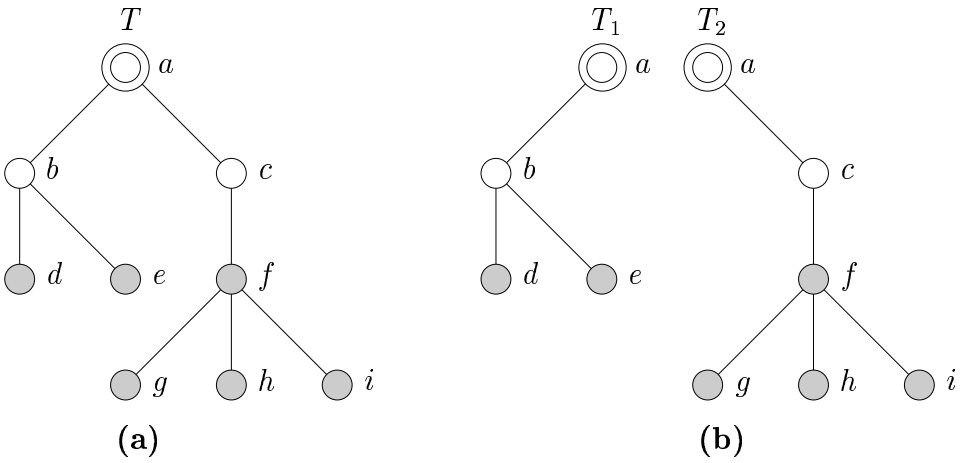


Figure 3.41 – *Un arbre T et la forêt $F_T = \{T_1, T_2\}$ à l'issue de la segmentation initiale.*

Les heuristiques suivantes ne considèrent donc pas l'arbre T, mais sont appliquées sur

chaque arbre t de la forêt initiale F_T.

Heuristique TS. L'heuristique TS (pour *Topological Segmentation*) est une heuristique gloutonne qui réalise un compromis entre le coût de la forêt segmentée et le nombre d'arbres de cette forêt.

Soit T un arbre *multicast* enraciné en s et couvrant un groupe dont M est l'ensemble des membres. Le but de l'heuristique est de fragmenter T en une forêt F telle que chaque arbre $t \in F$ n'a pas plus de n_p nœuds significatifs. Pour cela, elle associe à chaque nœud n de T une valeur v_n. La valeur v_n est égale au nombre de nœuds significatifs du sous-arbre de T enraciné en n. Plus précisément, on peut calculer v_n récursivement de la manière suivante :

$$v_n = 1 + \sum_{c \text{ fils de } n \text{ dans } T} v_c.$$

À chaque itération de l'heuristique TS, le nœud n^* ayant la plus grande valeur v_{n^*} inférieure ou égale à n_p est choisi. Le sous-arbre t_{n^*} de racine n^* est ajouté à la forêt F et tous les membres couverts par t_{n^*} sont retirés de T. Les valeurs des nœuds de T sont alors recalculées. Ce procédé est réitéré jusqu'à ce que T soit vide. Cette heuristique est décrite par l'algorithme 4.

Pré-requis : (s, M) un groupe, n_p le nombre maximum de nœuds significatifs d'un arbre
Retourne : $F = \{t_i\}$ une forêt telle que chaque sous-arbre t_i a au plus n_p nœuds significatifs

```
T ←un arbre couvrant {s} ∪ M
F ← ∅
tantque T n'est pas vide faire
  pour chaque nœud n de T faire
    calculer la valeur v_n associée à n
  fin pour
  n* ←le nœud de T ayant la plus grande valeur inférieure ou égale à n_p
  t_{n*} ← le sous-arbre de T enraciné en n*
  ajouter à t_{n*} un plus court chemin de s à n*
  F ← F ∪ {t_{n*}}
  enlever de T les membres du sous-arbre t_{n*}
fin tantque
```

Algorithme 4: L'heuristique TS.

Heuristiques MCPF. Les heuristiques MCPF (pour *Maximal Common Path First*) sont au nombre de deux : MCPFM (pour *MCPF on Members*) et MCPFL (pour *MCPF on Leaves*). Elles utilisent une nouvelle distance κ définie entre deux destinataires d_i et d_j. $\kappa(d_i, d_j)$ donne le nombre d'arêtes communes entre le chemin de la source s à d_i dans l'arbre

T et le chemin de s à d_j dans l'arbre T. En d'autres termes, nous avons :

$$\kappa(d_i, d_j) = |p(s, d_i) \cap p(s, d_j)|,$$

où $p(s, d)$ est un plus court chemin de s à d.

L'heuristique MCPFM construit un graphe complet sur l'ensemble des destinataires de T. À la première itération, l'heuristique MCPFM choisit les deux destinataires n et n' tels que $\kappa(n, n')$ est maximal. Un arbre t est initialisé avec le chemin de s à n et le chemin de s à n'. Puis, à chaque itération, l'heuristique MCPFM choisit les deux destinataires n et n' tels que $\kappa(n, n')$ est maximal avec $n \in t$ et $n' \notin t$. L'arbre t est étendu avec le chemin de s à n' si et seulement si l'arbre résultant ne possède pas trop de nœuds significatifs. Lorsque c'est le cas, un nouvel arbre est réinitialisé.

L'heuristique MCPFL construit un graphe complet sur l'ensemble des feuilles de T (c'est-à-dire un sous-ensemble des destinataires). À la première itération, l'heuristique MCPFL choisit les deux feuilles n et n' telles que $\kappa(n, n')$ est maximal. Un arbre t est initialisé avec le chemin de s à n et le chemin de s à n'. Les nœuds destinataires se trouvant sur ces chemins sont ajoutés à l'arbre en construction t, du nœud feuille jusqu'à la racine s de l'arbre, tant qu'il n'y a pas trop de nœuds significatifs dans l'arbre. Puis, à chaque itération, l'heuristique MCPFL choisit les deux feuilles n et n' telles que $\kappa(n, n')$ est maximal avec $n \in t$ et $n' \notin t$. L'arbre t est étendu avec le chemin de s à n' si et seulement si l'arbre résultant ne possède pas trop de nœuds significatifs. Les nœuds destinataires se trouvant sur ce chemin sont ajoutés à t tant qu'il n'y a pas trop de nœuds significatifs. Lorsque l'arbre en construction ne peut plus être étendu (par un chemin ou par ajout de nœuds destinataires en remontant le chemin), un nouvel arbre est réinitialisé.

Théorème 7. *Le rapport d'approximation de l'heuristique MCPFM pour le nombre d'arbres est de* 2.

Démonstration. Un nœud n' est ajouté à l'arbre t si et seulement si $t \cup p(s, n')$ contient moins de n_p nœuds significatifs. Le pire des cas pour l'heuristique MCPFM est quand tous les arbres ont $n_p - 1$ nœuds significatifs. En effet, l'ajout d'un nœud n' à t n'augmente que d'au plus 2 le nombre de nœuds significatifs. L'en-tête de chacun des arbres de la solution F trouvée par l'heuristique MCPFM contenant au moins $(n_p - 1)/2 + 1$ membres, nous avons :

$$\begin{aligned} NB.arbres(F) &\leq \frac{n}{(n_p - 1)/2 + 1} \\ &\leq \frac{2 \cdot n}{n_p + 1} \\ &\leq 2\frac{n}{n_p} \end{aligned}$$

Or, le nombre d'arbres de la solution optimale est supérieur ou égal à n/n_p. Ainsi, le rapport d'approximation de l'heuristique MCPFM est 2 pour le nombre d'arbres. □

Simulations. Nous avons implémenté les heuristiques TS, MCPFM et MCPFL afin de comparer leurs performances. Pour les simulations, toutes les heuristiques partent d'un arbre

construit par l'heuristique TM. Le réseau choisi est toujours Eurorings. Rappelons que le seuil de segmentation est fixé à 20 nœuds significatifs et que les groupes ont des tailles de 10 à 35 par pas de 5. Chaque point des courbes est la moyenne de 100 simulations.

La figure 3.42 montre le nombre d'arbres dans les forêts en fonction du nombre de membres par groupe. Lorsque les groupes ont une taille qui avoisine ou qui dépasse 20, le nombre d'arbres construit par les heuristiques augmente significativement. L'heuristique construisant le plus d'arbres est l'heuristique TS.

La figure 3.43 montre l'évolution du nombre de fois où l'algorithme de segmentation est appelé, en fonction du nombre de membres par groupe. Comme pour le nombre d'arbres, l'heuristique TS est celle qui segmente le plus les arbres de la forêt. Il en résulte notamment que le stress des liens est plus important avec l'heuristique TS.

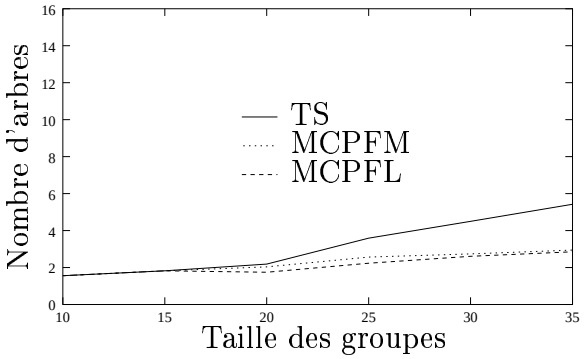


Figure 3.42 – *Nombre d'arbres en fonction de la taille des groupes.*

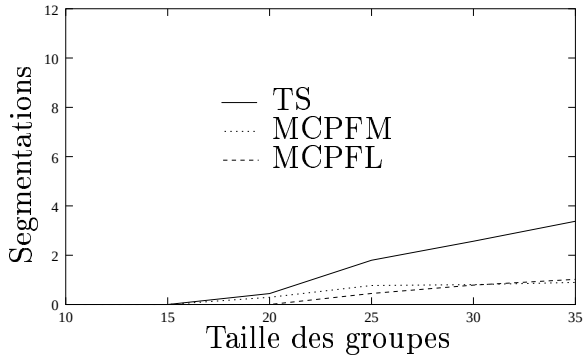


Figure 3.43 – *Nombre de segmentations en fonction de la taille des groupes.*

La figure 3.44 montre l'évolution du coût des forêts en fonction du nombre de membres par groupe. La différence de coût entre les différentes forêts n'est pas significative.

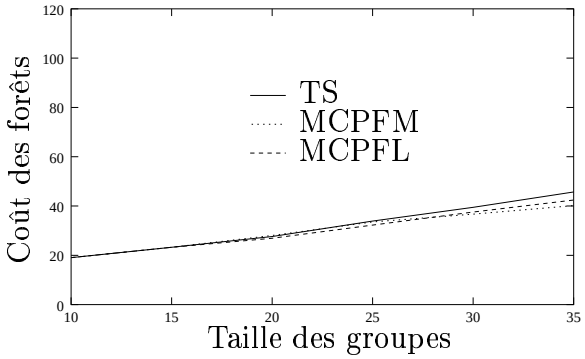


Figure 3.44 – *Coût des forêts en fonction de la taille des groupes.*

3.2.5 La sûreté de fonctionnement dans un protocole explicite arborescent

Un protocole explicite arborescent se base sur un protocole *unicast*. Cependant, même si ce protocole *unicast* sous-jacent permet une récupération très rapide des pannes, le protocole

explicite arborescent ne les supporte pas toutes. Pour s'en convaincre, considérons l'exemple de la figure 3.31. Le routeur r_2 envoie au routeur r_3 un paquet dont l'en-tête est $r_3 : (r_4 : (r_5, r_7), r_9 : (r_{10}, r_{11}))$. Même si un protocole *unicast* assure que r_3 recevra le paquet si le lien (r_2, r_3) tombe en panne, il ne peut pas gérer la panne du routeur r_3 lui-même.

Dans ce paragraphe, nous nous proposons d'améliorer la sûreté de fonctionnement d'un protocole explicite arborescent. Nous présentons d'abord les hypothèses que nous faisons sur le routage *unicast* et nos objectifs. Puis, nous décrivons deux méthodes de protection : la réorganisation de l'arbre ou le contournement des pannes. Enfin, nous récapitulons brièvement notre étude.

Hypothèses et objectifs

Notre proposition visant à améliorer la sûreté de fonctionnement d'un protocole explicite arborescent dépend du niveau de protection du protocole *unicast* sous-jacent. Nous avons identifié trois niveaux de protection, correspondant aux hypothèses 1, 2 et 3.

Hypothèse 1. *Le protocole unicast sous-jacent dispose d'un mécanisme de protection dont la convergence est instantanée.*

Hypothèse 2. *Le protocole unicast sous-jacent dispose d'un mécanisme de protection dont la convergence est lente.*

Hypothèse 3. *Le protocole unicast sous-jacent ne dispose pas de mécanisme de protection.*

Notre objectif est de munir le protocole explicite arborescent d'une protection explicite. Afin de ne pas surcharger l'en-tête des paquets, cette protection doit être aussi compacte que possible. Si le protocole *unicast* sous-jacent dispose d'une protection dont la convergence est instantanée (hypothèse 1), une simple réorganisation de l'arbre assure la protection. Autrement (hypothèses 2 et 3), il faut un mécanisme de contournement des pannes.

Réorganisation de l'arbre

Dans ce paragraphe, nous considérons l'hypothèse 1. Sous cette hypothèse, si un routeur de branchement de l'arbre tombe en panne, tous ses descendants perdent le message. La réorganisation de l'arbre consiste à faire en sorte que le routeur qui tombe en panne ne soit plus un routeur de branchement de l'arbre, mais une feuille par exemple.

Un arbre quelconque est représenté sur la figure 3.45(a) et sur la figure 3.45(b), la situation après la panne du routeur r_2. La représentation de l'arbre après la panne peut se déduire facilement de la représentation de l'arbre avant la panne :

- l'adresse du routeur en panne r_p (r_2 sur l'exemple) est retirée,
- le premier fils r_f de r_p dans l'arbre devient le prochain nœud (r_3 sur l'exemple),
- tous les fils du routeur en panne r_p (r_2 sur l'exemple) deviennent fils de r_f.

Le routeur en charge de réorganiser l'arbre est celui qui détecte que son successeur est tombé en panne. Sur la figure 3.45, il s'agit du routeur r_1. Dans un environnement hétérogène où tous les routeurs ne sont pas forcément TBXcast, c'est le premier routeur TBXcast sur le chemin du prédécesseur de r_p à la source qui s'en charge. Pour que cette technique puisse être implémentée, nous considérons qu'au lieu de rejeter les paquets si le prochain routeur est tombé en panne, un routeur non TBXcast les retourne vers la source.

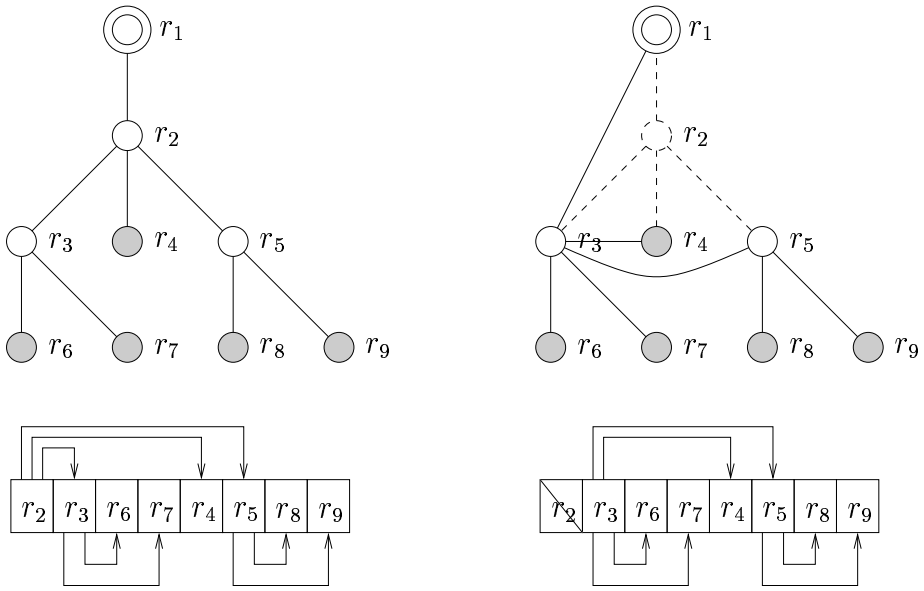


Figure 3.45 – *Réorganisation de l'arbre après la panne du routeur r_3.*

Contournement des pannes

Dans ce paragraphe, nous considérons les hypothèses 2 et 3. La difficulté du contournement des pannes consiste à utiliser le routage *unicast* alors qu'il n'a pas encore convergé.

Considérons l'exemple de la figure 3.46. Le routeur r_1 envoie des paquets au routeur r_3. Si le routeur r_2 tombe en panne, tous les paquets à destination de r_3 seront perdus jusqu'à ce que le protocole *unicast* ait convergé. Cependant, cette panne peut être contournée en passant par r_6, comme le montre la figure 3.46. En effet, le chemin de r_1 à r_6 et le chemin de r_6 à r_3 ne passent pas par r_2.

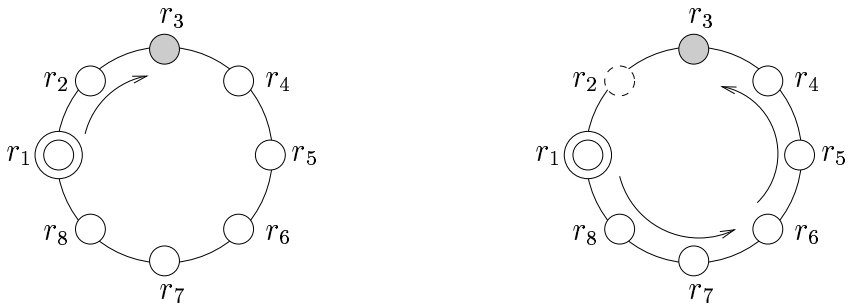


Figure 3.46 – *Contournement de la panne du routeur r_2.*

Certaines pannes ne peuvent pas être contournées car la topologie n'est pas assez redondante. De plus, pour contourner certaines pannes, il est nécessaire d'utiliser plusieurs nœuds intermédiaires. Le stockage de ces nœuds intermédiaires dans l'en-tête de chaque paquet générant une surcharge importante, nous proposons de ne contourner les pannes qu'avec un seul nœud intermédiaire.

À un nœud de branchement n d'un arbre T, de prédécesseur p, on peut associer l'ensemble

$C(n)$ des nœuds pouvant servir à contourner n. Un nœud n' est dans l'ensemble $C(n)$ si :

- le plus court chemin de p à n' ne passe pas par n,
- aucun chemin de n' à un fils significatif f_i de n dans T ne passe pas par n.

Nous pouvons vérifier sur la figure 3.46 que ces conditions sont satisfaites pour $n = r_2$, $p = r_1$, $f = r_3$ et $n' = r_6$.

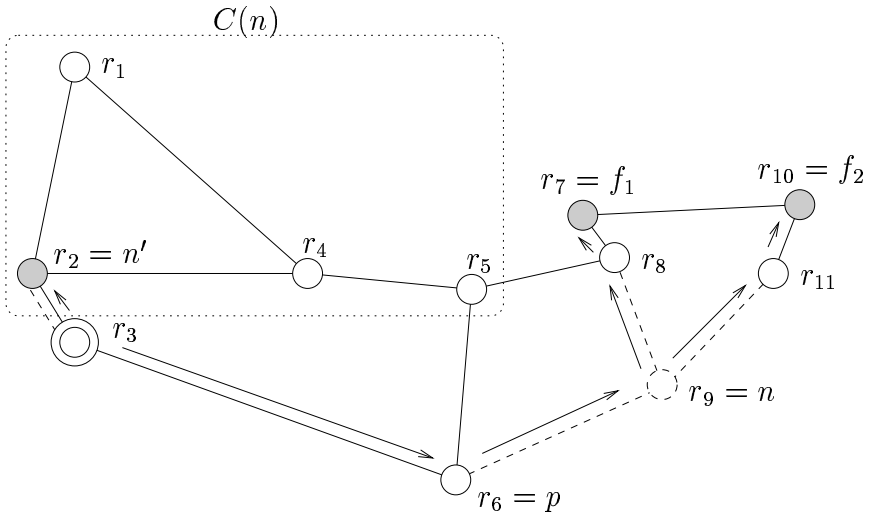


Figure 3.47 – *Ensemble des nœuds de contournement d'un nœud n.*

Sur la figure 3.47, l'ensemble $C(r_9) = \{r_1, r_2, r_4, r_5\}$ est représenté. Vérifions que le routeur r_2 appartient bien à $C(r_9)$:

- Le plus court chemin de $p = r_6$ à $n' = r_2$ est $((r_6, r_3), (r_3, r_2))$. Ce chemin ne passe pas par $n = r_9$.
- Le chemin de $n' = r_2$ à $f_1 = r_7$ est $((r_2, r_4), (r_4, r_5), (r_5, r_8), (r_8, r_7))$. Ce chemin ne passe pas par $n = r_9$.
- Le chemin de $n' = r_2$ à $f_2 = r_{10}$ est $((r_2, r_4), (r_4, r_5), (r_5, r_8), (r_8, r_7), (r_7, r_{10}))$. Ce chemin ne passe pas par $n = r_9$.

Nous pouvons remarquer que les nœuds de $C(n)$ sont en général éloignés de n. Si l'ensemble $C(n)$ est vide, le nœud n n'est pas protégeable.

Nous avons lancé 100000 simulations sur les réseaux Abilène et Eurorings pour connaître le nombre de nœuds de branchement des arbres *multicast* et le nombre de nœuds de branchement pouvant être protégés (en dehors de la source). Sur le réseau Abilène, nous avons obtenu une moyenne de 0.634 nœuds de branchement par arbre. Parmi ces 0.634 nœuds de branchement, en moyenne 0.292 sont protégeables (c'est-à-dire que l'ensemble $C(n)$ qui leur est associé n'est pas vide). En moyenne, 46 % des nœuds d'Abilène sont protégeables. Sur le réseau Eurorings, nous avons obtenu une moyenne de 5.788 nœuds de branchement par arbre. Parmi ces 5.788 nœuds de branchement, en moyenne 2.973 nœuds sont protégeables. En moyenne, 51 % des nœuds d'Eurorings sont protégeables. En résumé, environ la moitié des nœuds de branchement peuvent être protégés par un mécanisme de contournement à un nœud intermédiaire. Avec deux nœuds intermédiaires, 89 % des nœuds d'Abilène et 68 % des nœuds d'Eurorings sont protégeables.

Récapitulatif

Il est possible de protéger un protocole explicite arborescent en réorganisant l'arbre ou en encodant dans l'en-tête de chaque paquet des nœuds de contournement. L'efficacité de la protection dépend en partie du type de protection *unicast* sous-jacent.

3.2.6 Le déploiement incrémental d'un protocole explicite arborescent

Dans un protocole explicite arborescent, la topologie est connue (partiellement dans le cas du protocole ERM). Le routeur source peut notamment savoir quels sont les routeurs déployés et agir en conséquence (ce sont d'ailleurs les seuls qui apparaissent dans les traces du protocole ERM). Il est important de disposer d'un algorithme de construction d'arbres prenant en compte ce déploiement partiel et d'avoir bien choisi les routeurs déployés. Dans ce paragraphe, nous considérons que le protocole explicite arborescent est TBXcast même si les remarques s'appliquent aux autres protocoles explicites arborescents.

Algorithme de construction pour un réseau hétérogène

Un réseau hétérogène est constitué de routeurs TBXcast et de routeurs non TBXcast. Seuls les routeurs TBXcast peuvent être routeurs de branchement de l'arbre explicite. Ainsi, les routeurs non TBXcast sont obligatoirement des routeurs de relais de l'arbre explicite. Un algorithme de construction d'arbre valides doit prendre en compte cette contrainte.

La formulation de la contrainte d'hétérogénéité est très similaire à la contrainte de duplication des réseaux tout optique : les routeurs TBXcast peuvent être assimilés à des routeurs optiques pouvant dupliquer, tandis que les routeurs non TBXcast peuvent être assimilés à des routeurs optiques ne pouvant pas dupliquer. Les heuristiques décrites dans le chapitre 2 peuvent ainsi être utilisées dans le contexte d'un réseau hétérogène TBXcast.

Stratégies de déploiement des routeurs

Le problème du déploiement incrémental d'un protocole explicite arborescent est similaire au problème du placement de routeurs capables de dupliquer dans les réseaux tout optique. Ce problème a été étudié dans la littérature et l'heuristique MADD (pour *Multicast-ADD*) a été proposée dans [Ali02].

L'heuristique MADD (pour Multicast-ADD) est une heuristique itérative cherchant à minimiser la probabilité de blocage, c'est-à-dire à maximiser le nombre de groupes à placer. Dans le contexte du déploiement incrémental d'un protocole explicite arborescent, la fonction d'objectif doit être modifiée afin de minimiser le coût des arbres. L'heuristique MADD peut être facilement utilisée pour le déploiement incrémental de TBXcast.

3.3 Conclusion

Dans ce chapitre, nous avons traité du routage *multicast* explicite. L'avantage principal de ce routage est que les routeurs n'ont pas besoin de mémoriser des informations propres aux

groupes *multicast*, les informations nécessaires étant contenues dans chaque paquet. Ainsi, beaucoup de petits groupes *multicast* concurrents peuvent être gérés efficacement.

Tout d'abord, nous avons présenté et analysé le routage explicite plat. Cependant, ce routage requiert un grand nombre d'examens d'en-tête *unicast*. Le routage *multicast* explicite arborescent, présenté ensuite, pallie à ce problème mais diminue la charge utile de tous les paquets.

Dans ce chapitre, nos contributions sont les suivantes :

- La preuve que la fragmentation IP doit être évitée dans les protocoles de routage *multicast* explicites.
- L'optimisation du protocole Xcast en découpant la liste en sous-listes d'une certaine taille, selon que l'on veuille minimiser le volume d'informations envoyées ou le délai pour que les destinataires recoivent les données. Nous avons aussi montré que le tri de la liste des membres par ordre lexicographique des adresses IP offrait de bonnes performances, même lorsque peu de localité existait pour ces adresses.
- Le protocole Xcast doit être déployé en bordure des réseaux. La question de ce déploiement est actuellement soulevée au Japon. Pour les opérateurs, déployer le protocole en bordure est bien moins coûteux puisqu'il n'est pas nécessaire de changer les routeurs de cœur du réseau (ou du moins de leur ajouter un nouveau protocole de routage).
- La proposition d'heuristiques permettant de construire des arbres ayant peu de nœuds significatifs, dans le cadre du routage explicite arborescent. En fait, ces heuristiques dépassent ce cadre : plusieurs protocoles pourraient profiter d'un faible nombre de routeurs de branchement (comme HBH, Reunite ou SEM).
- La proposition d'une protection propre au routage explicite arborescent. Cette protection peut être dynamiquement ajustable et peut être utilisée alors que le protocole *unicast* sous-jacent n'a pas encore convergé.

Références

[ATK05] V. Arya, T. Turletti, et S. Kalyanaraman. « Encodings of Multicast Trees ». Dans *IFIP Networking*, numéro 3462 dans Springer, LNCS, pages 992–1004, mai 2005.

[Ali02] M. Ali. « Optimization of Splitting Node Placement in Wavelength-Routed Optical Networks ». *IEEE Journal on Selected Areas in Communications*, 20(8):1571–1579, octobre 2002.

[BC03] A. Boudani et B. Cousin. « SEM: A new small group multicast routing protocol ». Dans *IEEE International Conference on Telecommunications (ICT)*, février 2003.

[BFI+05] R. Boivie, N. Feldman, Y. Imai, W. Livens, D. Ooms, et O. Paridaens. « Explicit multicast (Xcast) basic specification ». Draft, IETF, janvier 2005. draft-ooms-xcast-basic-spec-07.txt.

[BFM00] R. Boivie, N. Feldman, et C. Metz. « Small Group Multicast: A New Solution for Multicasting on the Internet ». *IEEE Internet Computing*, 4(3):75–79, mai 2000.

[BFST00] J. Bion, D. Farinacci, M. Shand, et A. Tweedly. « Explicit Route Multicast (ERM) ». Draft, IETF, juin 2000. draft-shand-erm-00.txt.

[BGC03] A. Boudani, A. Guitton, et B. Cousin. « GXcast : une généralisation du protocole Xcast ». Dans *Manifestation des jeunes chercheurs STIC (Majecstic)*, octobre 2003.

[BGC04a] A. Boudani, A. Guitton, et B. Cousin. « GXcast : une généralisation du protocole Xcast ». *Numéro spécial de la revue Informations, Savoirs, Décisions et Médiations sur Majecstic (ISDM'Majecstic)*, 13, février 2004.

[BGC04b] A. Boudani, A. Guitton, et B. Cousin. « GXcast: Generalized Explicit Multicast Routing Protocol ». Dans *IEEE Symposium on Computers and Communications (ISCC)*, juin 2004.

[BMSBY04] M. Bag-Mohammadi, S. Samadian-Barzoki, et N. Yazdani. « Linkcast: Fast and Scalable Multicast Routing Protocol ». Dans *IFIP Networking*, pages 1282–1287, mai 2004.

[CFD01] L. H. M. K. Costa, S. Fdida, et O. C. M. B. Duarte. « Hop-by-Hop Multicast Routing Protocol ». Dans *ACM SIGCOMM*, pages 249–259, août 2001.

[CLRS90] T. Cormen, C. Leiserson, R. Rivest, et C. Stein. *Introduction to Algorithms*. McGraw-Hill, 1990.

[GB05] A. Guitton et A. Boudani. « Analyse du déploiement incrémental du protocole Xcast ». Dans *Colloque Francophone sur l'Ingénierie des Protocoles (CFIP)*, pages 169–184, mars 2005.

[HA03] Q. He et M. Ammar. « Dynamic Host-Group/Multi-Destination Routing for Multicast Sessions ». Dans *IEEE International Conference on Computer Communications and Networks (IC3N)*, pages 248–254, octobre 2003.

[HMB+05a] C. Hsu, E. Muramoto, J. Buford, Y. Imai, A. Boudani, et R. Boivie. « Best Current Practices of XCAST (Explicit Multi-Unicast) by 2004 ». Draft, IETF, avril 2005. draft-hsu-xcast-bcp-2004-00.txt.

[HMB+05b] C. Hsu, E. Muramoto, J. Buford, Y. Imai, A. Boudani, et R. Boivie. « IANA Considerations for XCAST (Explicit Multi-Unicast) ». Draft, IETF, avril 2005. draft-hsu-xcast-iana-considerations-00.txt.

[IKSK03] Y. Imai, H. Kishimoto, M.-K. Shin, et Y.-H. Kim. « XCAST6: eXplicit Multicast on IPv6 ». Dans *International Symposium on Applications and the Internet (SAINT)*, janvier 2003.

[MGC05] M. Molnár, A. Guitton, et B. Cousin. « TBXcast: Tree-Based Explicit Multicast Protocol ». Publication interne, Irisa, 2005. À paraître.

[POS02] O. Paridaens, D. Ooms, et B. Sales. « Security Framework for Explicit Multicast ». draft draft-paridaens-xcast-sec-framework-02.txt, IETF, juin 2002.

[RFC 1191] J. C. Mogul et S. E. Deering. « Path MTU discovery ». RFC 1191, IETF, novembre 1990.

[RFC 1981] J. McCann, S. E. Deering, et J. C. Mogul. « Path MTU Discovery for IP version 6 ». RFC 1981, IETF, août 1996.

[RFC 2328] J. Moy. « OSPF Version 2 ». RFC 2328, IETF, avril 1998.

[RFC 2453] G. Malkin. « RIP Version 2 ». RFC 2453, IETF, novembre 1998.

[RFC 2460] S. E. Deering et R. Hinden. « Internet Protocol, Version 6 (IPv6) Specification ». RFC 2460, IETF, décembre 1998.

[RFC 3376] B. Cain, S. E. Deering, I. Kouvelas, B. Fenner, et A. Thyagarajan. « Internet Group Management Protocol, Version 3 ». RFC 3376, IETF, octobre 2002.

[RFC 791] J. Postel. « Internet Protocol ». RFC 791, IETF, septembre 1981.

[SENZ00] I. Stoica, T. S. Eugene Ng, et H. Zhang. « REUNITE: A Recursive Unicast Approach to Multicast ». Dans *IEEE Infocom*, volume 3, pages 1644–1653, 2000.

[SKPK01] M.-K. Shin, Y.-J. Kim, K.-S. Park, et S.-H. Kim. « Explicit Multicast Extension (Xcast+) for Efficient Multicast Packet Delivery ». *Electronics and Telecommunication Research Institute (ETRI) Journal*, 23(4), décembre 2001.

Chapitre 4

Agrégation d'arbres

Parmi les raisons principales qui empêchent le déploiement du *multicast* sur Internet, nous pouvons citer le passage à l'échelle des entrées de routage *multicast* et du nombre de messages de contrôle. Alors qu'en *unicast*, l'allocation hiérarchique des adresses a permis d'agréger efficacement les entrées, une technique similaire est impossible en *multicast*. Cela vient du fait que les adresses *unicast* ont une signification topologique (et géographique), alors que les adresses *multicast* n'ont qu'une signification logique.

Dans le modèle de Deering, un routeur possède une entrée de routage par arbre qui le couvre. Ainsi, le nombre d'entrées de routage dans un routeur est égal au nombre d'arbres qui couvrent ce routeur. Si le nombre de groupes concurrents devient important, le nombre d'arbres couvrant un routeur donné augmente, ce qui entraîne l'augmentation du nombre d'entrées de routage. Un grand nombre d'entrées nécessite une grande capacité de stockage et un temps important de recherche dans cette table. Le nombre de messages de contrôle nécessaires pour maintenir ces arbres croît lui aussi avec le nombre d'arbres et utilise donc d'autant plus de bande passante qu'il y a de groupes.

Il existe plusieurs méthodes pour réduire la taille des tables *multicast* : la compression de la table, l'utilisation d'arbres réduits, l'agrégation d'entrées et l'agrégation d'arbres. Après avoir décrit ces méthodes, nous nous concentrons sur l'agrégation d'arbres, qui a l'avantage de réduire aussi le nombre de messages de contrôle nécessaires à la maintenance des arbres *multicast*. Puis, nous décrivons les protocoles existants et nous les comparons avec ceux que nous proposons.

4.1 Réduction de la taille des tables *multicast*

Dans ce paragraphe, nous décrivons trois méthodes permettant de réduire la taille des tables *multicast* : la compression de la table, l'utilisation d'arbres réduits et l'agrégation

d'entrées. L'agrégation d'arbres est décrite dans la section suivante.

4.1.1 Compression de la table

Pour réduire la mémoire utilisée par la table de routage, il est possible de compresser cette table.

Dans [DBCP97], les auteurs proposent une structure de données très compacte permettant de représenter des tables de routage occupant peu d'espaces et autorisant des recherches rapides. La structure est basée sur un arbre à trois niveaux, le temps de recherche dans chaque niveau étant limité. Les auteurs mentionnent que la table peut souvent être stockée entièrement en mémoire rapide, ce qui augmente son efficacité.

Cependant, l'utilisation d'une telle structure nécessite d'intégrer de nouveaux algorithmes dans les routeurs. De plus, le passage à IPv6 est mal supporté, l'important espace d'adressage augmentant significativement la taille de la structure et diminuant l'intérêt de son utilisation.

4.1.2 Utilisation d'arbres réduits

L'utilisation d'arbres réduits permet de supprimer certaines entrées de la table de routage. Dans le *multicast* traditionnel, un état est stocké dans chaque nœud d'un arbre. La notion d'arbre réduit consiste à ne stocker des états que dans les routeurs de branchement de l'arbre, comme indiqué sur la figure 4.1. Des chemins *unicast* sont utilisés entre les nœuds de branchement.

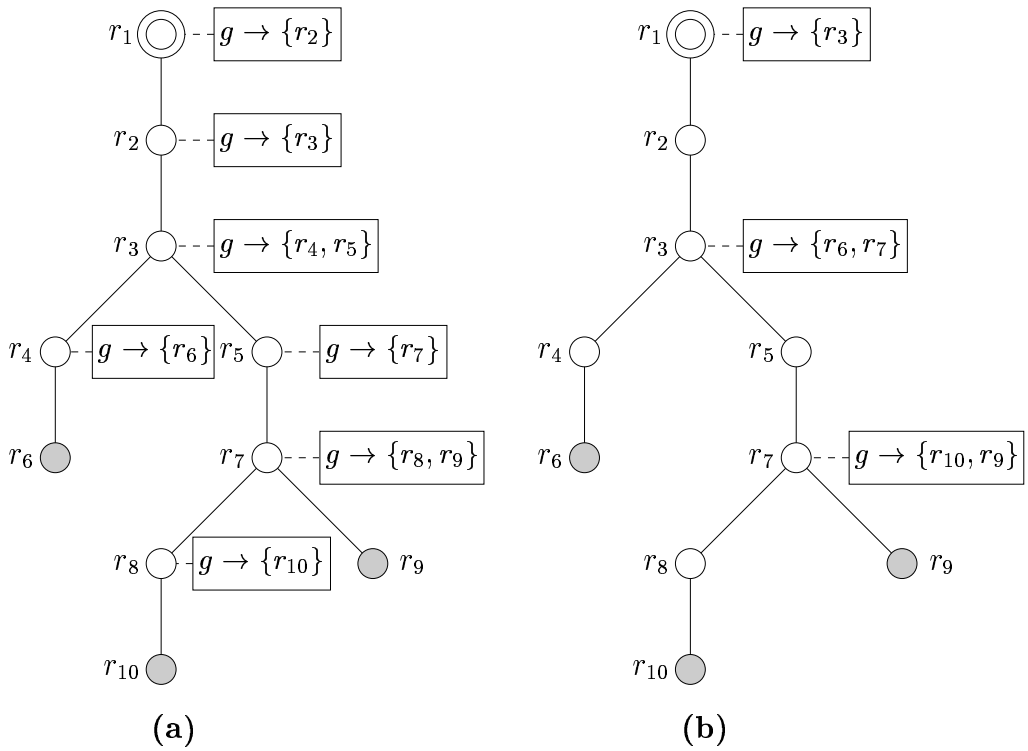


Figure 4.1 – *Entrées multicast dans (a) un arbre non réduit et dans (b) un arbre réduit.*

Il y a 7 entrées de routage *multicast* dans l'arbre non réduit présenté sur la figure 4.1(a). Sur l'arbre réduit présenté sur la figure 4.1(b), la source possède l'entrée $g \rightarrow \{r_3\}$ puisque r_3 est son prochain routeur de branchement. Le routeur de relais r_2 n'apparaît pas. Ainsi, il n'y a que 3 entrées dans cet arbre réduit. Dans des conditions réelles, le gain des arbres réduits est très important car les arbres possèdent souvent très peu de routeurs de branchement par rapport au nombre de routeurs de relais [PG98].

Les protocoles Reunite [SENZ00], HBH [CFD01] et SEM [BC05] utilisent cette notion d'arbre réduit.

4.1.3 Agrégation d'entrées

Dans [REG99], les auteurs proposent d'agréger les entrées dans les tables de routage *multicast*. Considérons un exemple pour les deux groupes `224.0.1.0` et `224.0.1.1`. Sans agrégation, la table de routage *multicast* contient les deux entrées `224.0.1.0/32` et `224.0.1.1/32`. Le préfixe `32` indique qu'un groupe g correspond à une de ces deux entrées si les 32 premiers bits de g correspondent à l'entrée. Comme une adresse IPv4 fait exactement 32 bits, la recherche est exacte. Si l'ensemble des interfaces de sortie correspondent pour les deux groupes `224.0.1.0` et `224.0.1.1`, les deux entrées `224.0.1.0/32` et `224.0.1.1/32` peuvent être agrégées en une unique entrée `224.0.1.0/31`. Nous remarquons que le préfixe de cette entrée agrégée est maintenant `31`. Ainsi, la correspondance ne se fait que sur les 31 premiers bits de l'adresse, plus sur le dernier bit.

Les auteurs ont aussi considérés la sur-agrégation des entrées. Si l'on note i_a l'ensemble des interfaces de sortie associées à `224.0.1.0/32` et i_b l'ensemble des interfaces de sortie associées à `224.0.1.1/32`, ces deux entrées peuvent être agrégées en une unique entrée `224.0.1.0/31` auquel est associé l'ensemble d'interfaces de sortie $i_a \cup i_b$, comme indiqué sur la figure 4.2. En sur-agrégant, le nombre d'entrées est réduit. Par contre, les paquets à destination du groupe `224.0.1.0` vont être acheminés inutilement sur les interfaces de sortie de $i_b \backslash i_a$ (c'est-à-dire i_3 sur l'exemple) et les paquets à destination du groupe `224.0.1.1` vont être acheminés inutilement sur les interfaces de sortie de $i_a \backslash i_b$ (c'est-à-dire i_1 sur l'exemple). Il devient important de considérer le compromis entre la taille de la table de routage *multicast* et la perte engendrée par la sur-agrégation. Notons que les paquets inutilement acheminés sont détruits automatiquement au prochain routeur.

Dans [TH00], les auteurs associent à chaque interface de sortie i un filtre F_i. F_i est un prédicat associant à un couple (s, g) un booléen. Les paquets pour (s, g) doivent être acheminés sur l'interface i si et seulement si $F_i(s, g)$ vaut `vrai`. L'agrégation d'entrées consiste à implémenter ce filtre F_i de manière compacte. Le filtre décrit dans l'article utilise l'existence d'intervalles $[(s_a, g_a); (s_b, g_b)]$ tels que tous les couples $(s_a, g_a) < (s, g) < (s_b, g_b)$ ont la même valeur $F_i(s, g)$. Lorsque les interfaces pour un groupe changent, peu d'intervalles doivent être modifiés.

La taille de l'implémentation d'un filtre F_i dépend du mécanisme d'allocation d'adresses et du positionnement des membres. Si l'on considère le pire des cas où la distribution des adresses est uniforme et où les membres sont répartis uniformément parmi les routeurs, l'implémentation d'un filtre nécessite toujours au moins 4 fois moins de place que l'implémentation d'une table, d'après les résultats des auteurs.

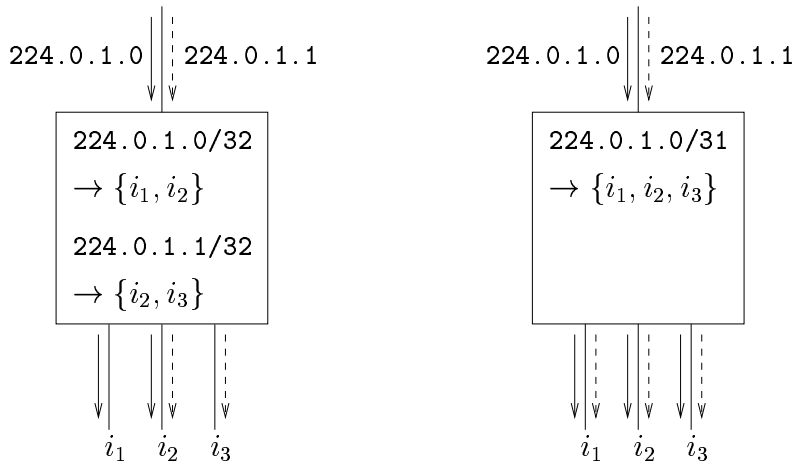


Figure 4.2 – *Sur-agrégation des entrées de routage multicast.*

4.2 Agrégation d'arbres

L'agrégation d'arbres a été introduite dans [GFCF01]. Il s'agit d'un mécanisme réduisant la taille des tables de routage à l'intérieur d'un domaine en réduisant le nombre d'entrées spécifiques aux arbres. Contrairement aux méthodes présentées dans la section précédente, l'agrégation d'arbres réduit aussi le nombre de messages de contrôle *multicast*, comme nous allons le voir.

À l'intérieur d'un domaine, plusieurs groupes constitués de membres différents peuvent correspondre aux mêmes routeurs de bordure du domaine. Ces groupes sont couverts par des arbres identiques dans le domaine. Basée sur cette remarque, l'agrégation d'arbres consiste à associer un même arbre à plusieurs groupes *multicast* au sein d'un domaine IP. Ce procédé réduit donc le nombre d'arbres et par conséquent à la fois le nombre d'entrées spécifiques à ces arbres et le nombre de messages de contrôle pour maintenir ces entrées. En revanche, des entrées spécifiques aux groupes doivent être ajoutées à l'entrée du domaine. Ainsi, tous les paquets *multicast* sont encapsulés avec une étiquette correspondant à un arbre. Dans le domaine, l'acheminement se fait selon cette étiquette. Le paquet est décapsulé à la sortie du domaine (et plus généralement dans chaque routeur de bordure) pour être acheminé selon les règles en vigueur à l'extérieur du domaine.

Un exemple sans agrégation d'arbres est présenté sur la figure 4.3(a) et avec agrégation d'arbres sur la figure 4.3(b). Les routeurs r_1, r_4, r_5 et r_6 sont des routeurs de bordure et les routeurs r_2 et r_3 sont des routeurs de cœur. À gauche, trois arbres sont créés : un pour le groupe g_1 avec des membres dans r_4, r_5 et r_6, un pour le groupe g_2 avec des membres dans r_4, r_5 et r_6 et un pour le groupe g_3 avec des membres dans r_4 et r_6. Au total, 9 entrées de routage sont nécessaires. Avec l'agrégation d'arbres, les trois groupes g_1, g_2 et g_3 peuvent être agrégés au même arbre t. Seulement 3 entrées de routage sont nécessaires. En revanche, il faut que le routeur r_1 mémorise que les groupes sont tous les trois associés au label t. Il faut donc stocker 3 entrées spécifiques aux groupes. Nous pouvons noter que de la bande passante est perdue quand les paquets pour le groupe g_3 empruntent le lien (r_3, r_5) inutilement.

Dans l'exemple précédant, l'agrégation est parfaite pour g_1 et g_2 : le même ensemble de

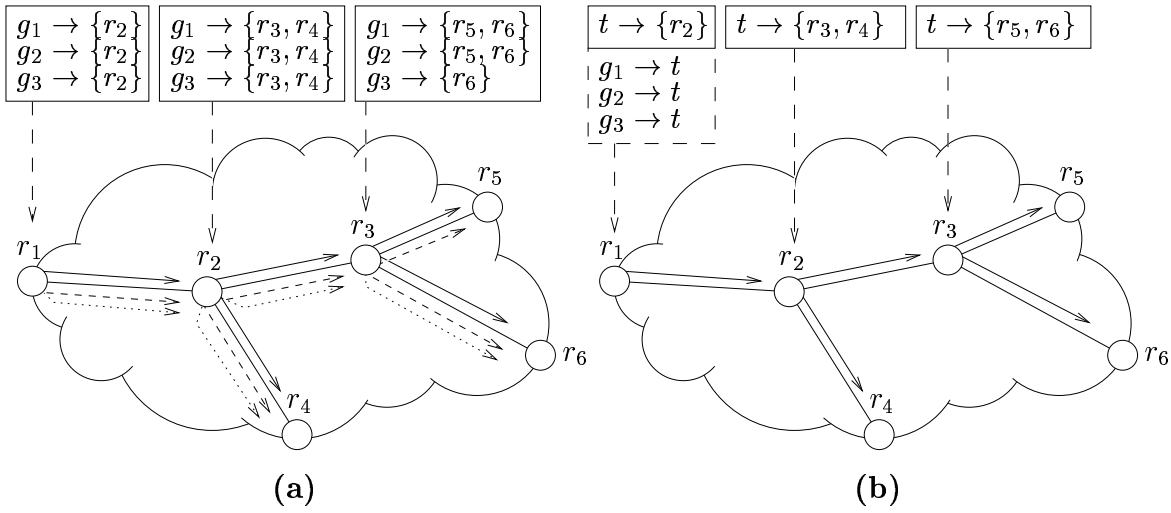


Figure 4.3 – *Acheminement sans et avec agrégation d'arbres.*

routeurs de bordure est associé à ces deux groupes. L'agrégation du groupe g_3 à l'arbre t n'est pas obligatoire, mais elle permet de réduire davantage la taille des tables de routage *multicast*, au détriment de la bande passante. L'agrégation du groupe g_3 à l'arbre t est une sur-agrégation : certaines feuilles de t ne sont pas membres de g_3. Il est possible de considérer des sous-agrégations, quand certains membres d'un groupe g ne sont pas couverts par l'arbre auquel le groupe g doit être agrégé. Dans ce cas, il faut construire des tunnels *unicast* pour atteindre les membres de g non couverts. Ces tunnels nécessitent la présence d'états de routage supplémentaires. Cependant, les tunnels augmentent beaucoup la bande passante utilisée (car ils sont point-à-point) et ils nécessitent beaucoup d'états (car ils ne peuvent pas être agrégés entre eux). Dans la suite, nous ne considérerons plus les tunnels[1]. À titre informatif, il existe un autre modèle d'agrégation d'arbres permettant l'agrégation d'un groupe à plusieurs sous-arbres. Le protocole AMBTS (pour *Aggregated Multicast Based on Tree Splitting*) [LDL04] est le seul protocole basé sur ce modèle. Nous ne le détaillerons pas dans cette thèse car l'algorithme de découpage des arbres qu'il utilise n'est pas décrit en détails.

En résumé, nous avons vu que :

- l'agrégation parfaite construit beaucoup d'arbres mais n'induit pas de perte de bande passante ;
- la sous-agrégation utilisant des tunnels réduit le nombre d'arbres mais introduit des états pour les tunnels ; de plus, les tunnels ne sont pas agrégeables entre eux et induisent une perte de bande passante ;
- la sur-agrégation réduit le nombre d'arbres mais induit une perte de bande passante.

L'agrégation d'arbres telle qu'elle a été introduite dans [GFCF01] est basée sur des réseaux IP. L'utilisation de labels paraît toutefois très proche de MPLS. On peut se demander

1. Les versions initiales des protocoles réalisant l'agrégation d'arbres utilisaient ces tunnels. Cependant, les versions les plus récentes ne mentionnent plus les tunnels ou les sous-agrégations. En concevant nos protocoles, nous sommes aussi parvenus à la conclusion qu'il était plus avantageux d'éviter les sous-agrégations utilisant des tunnels car ces tunnels ne peuvent pas être agrégés efficacement entre eux.

pourquoi l'agrégation d'arbres n'a pas été proposée dans ce cadre. L'apport principal de MPLS par rapport à IP est la possibilité de commuter le label à chaque routeur ; il serait alors envisageable de considérer qu'un arbre agrégé soit couvert par plusieurs labels. Cela résulte en une plus grande capacité d'agrégation, mais la complexité du mécanisme d'agrégation augmente considérablement. On peut voir l'agrégation d'arbres sur IP comme une vision simplificatrice de l'agrégation d'arbres sur MPLS. Dans notre étude, nous nous basons sur l'agrégation dans les réseaux IP. Le double avantage est que nous avons une base de travail simplifiée et que nous pouvons comparer nos protocoles à ceux de la littérature. L'étude de l'agrégation d'arbres dans les domaines MPLS est toutefois l'une de nos perspectives.

4.3 Les protocoles existants

Étant une proposition récente, il existe relativement peu de protocoles réalisant l'agrégation d'arbres. Nous avons regroupé ces protocoles en trois catégories. La première catégorie regroupe les protocoles mettant en avant les compromis de l'agrégation d'arbres. La deuxième catégorie concerne les protocoles gérant la capacité limitée des liens et les contraintes de bande passante des groupes. Enfin, la troisième catégorie regroupe des protocoles non centralisés.

4.3.1 Le compromis de l'agrégation

Pour réduire au maximum le nombre d'arbres maintenus dans le réseau, il est souvent nécessaire de perdre de la bande passante en réalisant des sur-agrégations. Il est important de prendre en compte le compromis entre le nombre d'arbres maintenus et la bande passante utilisée. Dans cette partie, nous décrivons deux protocoles, AM et STA, qui mettent en avant ce compromis de l'agrégation d'arbres.

Le protocole AM

Le protocole AM (pour *Aggregated Multicast*) est le premier à avoir introduit la notion d'agrégation d'arbres. Il est décrit en détail dans [CKM+03].

Le protocole AM possède un gestionnaire d'agrégation centralisé. Ce gestionnaire est en charge de réaliser l'agrégation d'arbres et de configurer les arbres dans le réseau.

La figure 4.4 décrit le fonctionnement du protocole AM. Lorsqu'un nouveau groupe g apparaît (événement 1), le gestionnaire d'agrégation est averti (message 2) et est en charge de retourner l'étiquette d'un arbre pouvant couvrir g. Le gestionnaire d'agrégation parcourt l'ensemble T des arbres actuellement configurés (algorithme 3a). Un arbre t de T est considéré comme un candidat pour l'agrégation si le coût de l'agrégation de g à t (calculé selon une fonction $f(g,t)$, décrite plus loin) est inférieur ou égal à une borne b. Parmi les candidats pour l'agrégation, g est agrégé à celui minimisant la fonction f. L'algorithme 5 récapitule ce processus (notons que la recherche du candidat minimisant la fonction f peut se faire au sein de la première itération). Si aucun candidat n'est trouvé, un arbre natif $t^{nat}(g)$ est construit pour g et est configuré dans le réseau (non représenté sur la figure). Finalement, tous les routeurs de bordure de l'arbre correspondant à g (qu'il s'agisse d'un nouvel arbre

ou non) sont configurés (message 4) pour associer à g l'étiquette de l'arbre et ainsi mettre à jour les entrées spécifiques au groupe.

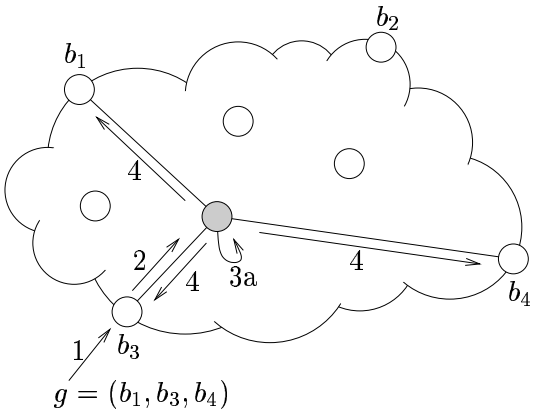


Figure 4.4 – *L'agrégation avec le protocole AM.*

Les changements des membres d'un groupe g sont gérés de manière similaire. Notons g_{old} le groupe g avant une modification et g_{new} le groupe g après la modification dans les membres. Notons $t(g)$ l'arbre auquel g est agrégé. Si l'ancien arbre $t(g_{old})$ couvre le nouveau groupe sans que le gaspillage de bande passante dépasse le seuil fixé (c'est-à-dire que $f(g_{new}, t(g_{old})) \leq b$), l'agrégation du groupe à l'arbre $t(g_{old})$ est maintenue. Sinon, le protocole AM considère que le groupe g_{old} part et qu'un nouveau groupe g_{new} arrive. Dans ce cas, il faut enlever les entrées spécifiques au groupe g_{old}. De plus, si plus aucun groupe n'est agrégé à $t(g_{old})$, il faut supprimer cet arbre est les entrées qui lui correspondent. Finalement, dans tous les cas, les entrées spécifiques au groupe g sont changées dans les routeurs de bordure.

La fonction f est définie de la manière suivante [CKM+02] :

$$f(g,t) = \frac{c(t) - c(t^{nat}(g))}{c(t^{nat}(g))},$$

où $c(t)$ est le coût de l'arbre t et $t^{nat}(g)$ l'arbre natif pour g. Cette définition de f ne prend pas en compte les sous-agrégations utilisant des tunnels qui sont bien moins efficaces que les sur-agrégations.

Problèmes du protocole AM

L'inconvénient majeur du protocole AM est que la complexité de l'algorithme d'agrégation est importante. D'une part, tous les arbres sont évalués à chaque fois qu'un nouveau groupe arrive ou qu'un groupe change et, d'autre part, la fonction de comparaison $f(g,t)$ est coûteuse. Nous avons donc cherché à accélérer les agrégations du protocole AM.

On notera aussi qu'une fonction d'agrégation plus appropriée aurait pu être :

$$f(g,t) = \frac{c(t \backslash t^{nat}(g))}{c(t^{nat}(g))},$$

où $t \backslash t^{nat}(g)$ désigne l'ensemble des arêtes de t n'étant pas sur $t^{nat}(g)$. Toutefois, cette définition de f n'est toujours pas satisfaisante : l'arbre t peut être un bon candidat pour l'agrégation en ayant toutes ses arêtes différentes de celles de t^{nat}.

Pré-requis : T l'ensemble des arbres existants, g un groupe, b une borne
Retourne : un arbre de T qui couvre g

$C \leftarrow \emptyset$
pour $t \in T$ **faire**
  si t couvre g **alors**
    si $f(g,t) \leq b$ **alors**
      $C \leftarrow C \cup \{t\}$
    fin si
  fin si
fin pour
si $C = \emptyset$ **alors**
  construire l'arbre natif $t^{nat}(g)$ pour g
  $t^{nat}(g)$ est configuré
  $T \leftarrow T \cup \{t^{nat}(g)\}$
  retourner $t^{nat}(g)$
sinon
  $t^{agg} \leftarrow$l'arbre de C tel que $f(g, t^{agg})$ est minimum
  g est agrégé à t^{agg}
  retourner t^{agg}
fin si

Algorithme 5: L'algorithme d'agrégation du protocole AM.

Le protocole STA

Le protocole STA (pour *Scalable Tree Aggregation*) est une première approche visant à améliorer le passage à l'échelle du protocole AM. Nous l'avons proposé dans [GM05]. En quelques mots, le protocole STA consiste à accélérer l'algorithme 3a de la figure 4.4.

Dans le protocole AM, l'agrégation d'un nouveau groupe g se fait en parcourant exhaustivement l'ensemble T des arbres du réseau. Lorsque le nombre de groupes est important, le nombre d'arbres l'est aussi, et chaque agrégation devient coûteuse en temps. Il en va de même lorsque les groupes sont très dynamiques. En effet, le gestionnaire d'arbres est sollicité à chaque changement de groupe pour savoir si l'agrégation du groupe à l'arbre actuel est conservée.

Dans le protocole STA, nous proposons de ne parcourir qu'un sous-ensemble de l'ensemble T de tous les arbres afin d'accélérer le processus d'agrégation pour chaque groupe. Ce sous-ensemble doit être suffisamment petit pour que l'accélération soit significative et il doit contenir tous les candidats que le protocole AM aurait sélectionné pour conserver les mêmes performances. De plus, la fonction de recherche du sous-ensemble associé à un groupe g doit être rapide. Nous avons choisi de ne considérer pour un groupe g que le sous-ensemble de T constitué des arbres dont le coût est compris entre $c(t^{nat}(g))$ et $c(t^{nat}(g))(1+b)$, où $t^{nat}(g)$ est l'arbre natif pour g, $c(t)$ le coût d'un arbre t et b la borne du protocole STA, similaire à celle du protocole AM. Dans le protocole STA, l'arbre natif $t^{nat}(g)$ est un arbre de coût minimum (ou un arbre approché). Ainsi, aucun arbre de coût inférieur à $c(t^{nat}(g))$ ne peut couvrir g, d'où le choix de cette borne inférieure. Il arrive parfois qu'un arbre de coût inférieur à $c(t^{nat}(g))$ puisse couvrir g car l'arbre $t^{nat}(g)$ est un arbre approché. Cependant, cela arrive très peu souvent en pratique, les approximations étant en général très bonnes.

L'ensemble T est partitionné en sous-ensembles, tels que chaque sous-ensemble T_i contient tous les arbres de T de coût égal à i. Cela permet de repérer rapidement les arbres à évaluer pour un groupe g. L'algorithme d'agrégation du protocole STA est décrit sur l'algorithme 6.

Pré-requis : T l'ensemble des arbres existants, g un groupe, b une borne
Retourne : un arbre de T qui couvre g

```
t^nat(g) ←l'arbre natif de g
pour i de c(t^nat(g)) à c(t^nat(g))(1 + b) faire
  pour t ∈ T_i faire
    si t couvre g alors
      retourner t
    fin si
  fin pour
fin pour
T ← T ∪ {t^nat(g)}
retourner t^nat(g)
```

Algorithme 6: L'algorithme d'agrégation du protocole STA.

Nous avons aussi modifié la fonction de sélection de l'arbre pour la rendre plus rapide. À

présent, un groupe g est représenté par un vecteur de bits $\vec{g}$. Le i-ème bit du vecteur $\vec{g}$ est à 1 si et seulement si le i-ème routeur de bordure du domaine est un participant (une source ou un membre) du groupe g. Sur l'exemple de la figure 4.4, le groupe g ayant des participants dans b_1, b_3 et b_4 est représenté par le vecteur $\vec{g} = 1011$. De la même manière, un vecteur $\vec{t}$ est associé à chaque arbre t. Le i-ème bit du vecteur $\vec{t}$ est à 1 si et seulement si l'arbre couvre le i-ème routeur de bordure du domaine. Dans l'exemple de la figure 4.4, le vecteur associé à t est $\vec{t} = 1011$. Un arbre t couvre un groupe g en agrégation ou en sur-agrégation si et seulement si $\vec{t} \vee (\neg \vec{g}) = 11 \cdots 1$. À titre d'exemple, nous pouvons voir que t est un candidat pour une sur-agrégation de g : $\vec{t} \vee (\neg \vec{g}) = 1011 \vee (\neg 1011) = 1011 \vee 0100 = 1111$. L'intérêt de cette représentation en vecteurs de bits est de limiter la surcharge mémoire requise pour réaliser l'agrégation d'arbres et d'accélérer l'algorithme.

Finalement, plutôt que d'établir une liste C d'arbres candidats et de sélectionner l'arbre de coût minimum parmi ceux-ci, nous examinons les arbres par coût croissant et nous sélectionnons le premier qui couvre le groupe g.

Comparaison des protocoles AM et STA

Pour comparer les protocoles AM et STA, nous avons fait des simulations sur le réseau Eurorings. Les groupes arrivent selon une loi de Poisson de taux $\lambda = 0.9999$ (ce qui est volontairement très élevé pour réduire le temps de simulation) et leur durée de vie suit une distribution exponentielle de moyenne $\mu^{-1} = 50000$ unités de temps. Nous avons choisi ces paramètres λ et μ de manière à obtenir 45000 groupes concurrents. Dans nos simulations, nous avons généré environ 350000 requêtes de groupes[2].

La figure 4.5 donne le nombre d'arbres construits par les protocoles AM (avec une borne de 0 et de 0.2) et STA (avec une borne de 0 et de 0.2). Nous remarquons que les protocoles AM et STA ont les mêmes performances pour cette métrique quand leurs bornes sont égales. Ainsi, le protocole STA sélectionne des bons arbres en ne considérant qu'un sous-ensemble des arbres.

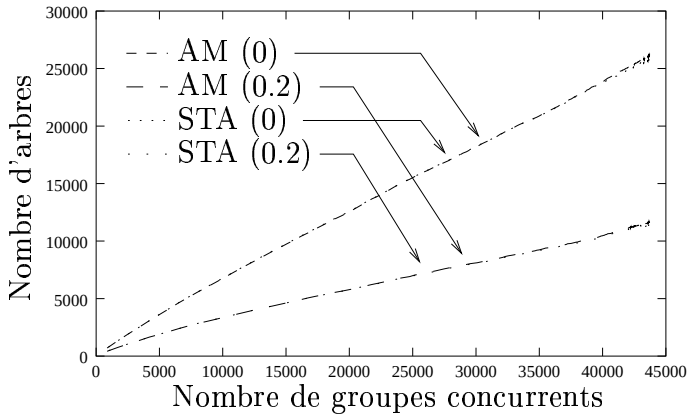


Figure 4.5 – *Nombre d'arbres construits par les protocoles AM et STA sur Eurorings.*

2. Si nous avions généré plus de requêtes de groupes, nous aurions obtenu une moyenne de $\lambda/\mu = 49995$ groupes concurrents. Cependant, le temps de simulation aurait été démesuré.

Contrairement à ce que pourrait laisser supposer la figure, le nombre d'arbres ne croît pas linéairement avec le nombre de groupes concurrents. Quand le nombre de groupes concurrents devient très important (de l'ordre de 2^B ou plus, où B est le nombre de routeurs de bordure du domaine), le nombre d'arbres stagne. En effet, tous les arbres possibles ont été crées et tous les nouveaux groupes peuvent être agrégés à un arbre existant. Sur le réseau Abilène qui comporte 11 nœuds, il est aisé d'exhiber ce comportement (voir la figure 4.6). Notons encore une fois que les protocoles AM et STA se comportent de la même manière. En revanche, sur le réseau Eurorings comportant 43 nœuds, ce comportement est peu perceptible avec nos paramètres de simulation.

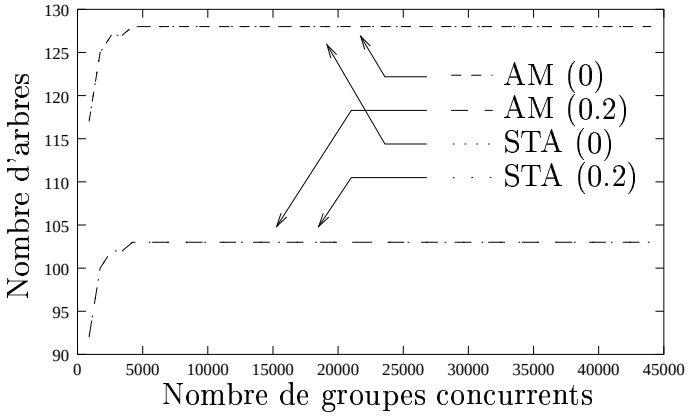


Figure 4.6 – *Nombre d'arbres construits par les protocoles AM et STA sur Abilène.*

La figure 4.7 donne le nombre total d'états de routage *multicast* (c'est-à-dire la somme du nombre d'états spécifiques aux arbres et spécifiques aux groupes) divisé par le nombre de routeurs du domaine, pour les protocoles AM (avec une borne de 0 et de 0.2) et STA (avec une borne de 0 et de 0.2). Le nombre total d'états est très proche pour les protocoles AM et STA et dépend majoritairement du nombre d'arbres construits. Cette courbe confirme que le nombre d'états de routage *multicast* croît linéairement avec le nombre d'arbres.

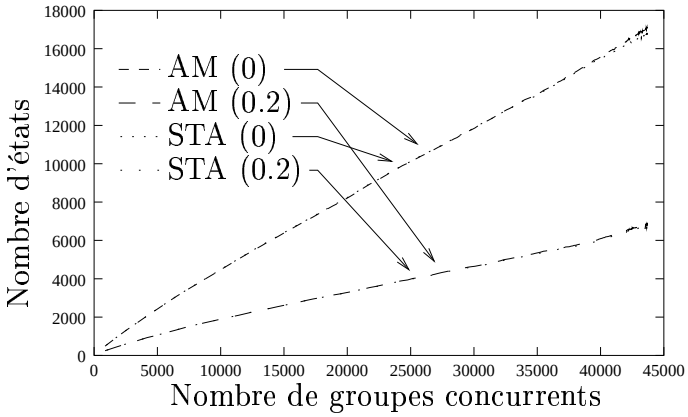


Figure 4.7 – *Nombre d'états multicast pour les protocoles AM et STA sur Eurorings.*

La bande passante moyenne utilisée par les protocoles AM et STA est présentée sur la

figure 4.8. Elle est calculée de la manière suivante :

$$BW = \frac{\sum_{g \in G} c(t(g))}{|G|},$$

où G désigne l'ensemble des groupes concurrents à un instant donné. Lorsque le seuil t_b est égal à 0, plus il y a de groupes concurrents, plus la bande passante diminue (jusqu'à stagner, ce que l'on ne voit pas sur cette figure). Cela s'explique par une meilleure utilisation des arbres lorsque beaucoup de groupes sont présents. Le protocole STA avec un seuil égal à 0 utilise légèrement plus de bande passante que le protocole AM avec un seuil égal à 0. De même, le protocole STA avec un seuil de 0.2 utilise légèrement plus de bande passante que le protocole AM avec un seuil de 0.2. Cette légère différence est due à la non optimalité des choix effectués par le protocole STA. Lorsque le seuil de ces deux protocoles est fixé à 0.2, chaque groupe peut être agrégé à un arbre dont le surcoût est de 20 %. Ainsi, le gaspillage maximum de bande passante est inférieur à 20 %. Nous pouvons voir sur la courbe que le gaspillage est limité à environ 7.8 %.

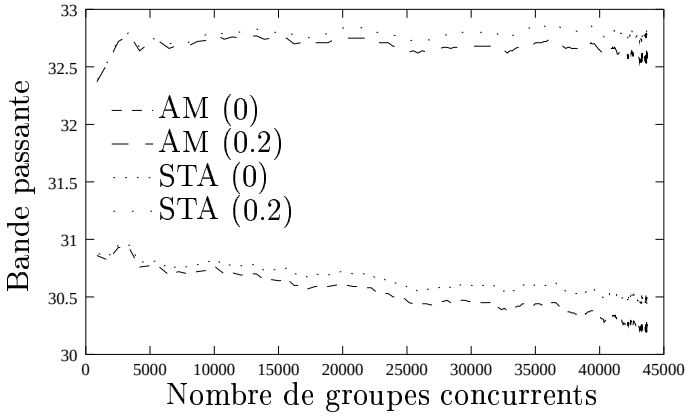


Figure 4.8 – *Bande passante utilisée par les protocoles AM et STA sur Eurorings.*

Pour le protocole AM, à chaque fois qu'un groupe arrive, tous les arbres existants sont évalués. Pour le protocole STA, seulement une fraction de l'ensemble des arbres existants est évaluée. La figure 4.9 présente le nombre d'arbres évalués par groupe. Nous remarquons que le protocole STA ne compare que très peu d'arbres par rapport au protocole AM, alors qu'il atteint les mêmes performances pour le nombre d'états concurrents et la bande passante utilisée. Nous remarquons aussi que l'utilisation de la borne de 0.2 pour le protocole AM réduit presque de moitié le nombre d'arbres à évaluer (puisqu'elle réduit presque de moitié le nombre d'arbres existants, voir sur la figure 4.5).

Les simulations que nous avons menées montrent que le protocole STA est beaucoup plus rapide que le protocole AM alors qu'il conserve les mêmes performances. Ainsi, la méthode proposée pour rechercher les arbres dans l'ensemble de tous les arbres agrégés T est une bonne approximation de la méthode recherchant la solution exhaustivement.

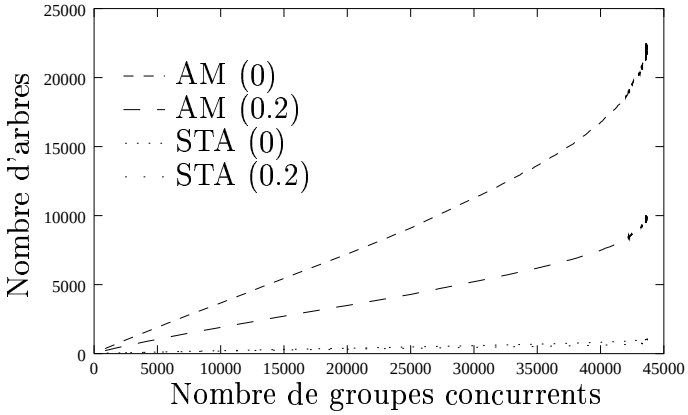


Figure 4.9 – *Nombre d'arbres évalués par les protocoles AM et STA sur Eurorings.*

Résumé

En résumé, nous venons de voir qu'il est nécessaire de perdre un peu de bande passante pour que le nombre d'arbres maintenus soit acceptable. En règle générale, une perte maximum autorisée de 20 % offre de bons résultats : peu d'arbres sont construits et peu de bande passante est utilisée. Nous avons aussi vu que le nombre d'entrées *multicast* et le nombre d'arbres maintenus dans le réseau ont le même comportement. Le nombre d'arbres maintenus est donc un bon indicateur du nombre d'entrées *multicast*.

Nous avons aussi montré grâce au protocole STA que la recherche du meilleur arbre pour l'agrégation peut se faire très rapidement, sans avoir à examiner l'ensemble des arbres existants. Le protocole STA agrège en moyenne quatre fois plus vite que le protocole AM et ses performances sont comparables.

4.3.2 L'agrégation d'arbres et la QoS

Dans la partie précédente, nous avons vu que l'agrégation d'arbres conduisait généralement à une perte de bande passante. On peut se demander ce qu'il en est lorsque les ressources en bande passante des liens sont limitées. Dans cette partie, nous décrivons les protocoles AQoSM et Q-STA qui gèrent les capacités limitées des liens et les besoins en bande passante des groupes.

Le protocole AQoSM

Le protocole AQoSM (pour *Aggregated QoS Multicast*, QoS étant le sigle pour *Quality of Service*) est décrit dans [CKF+02]. Contrairement au protocole AM et à la plupart des protocoles basés sur l'agrégation d'arbres, le protocole AQoSM ne fait pas l'hypothèse que les liens du réseau ont une capacité infinie.

Dans le protocole AQoSM, un groupe g ayant un besoin en bande passante $b(g)$ ne peut pas être agrégé à un arbre t si la bande passante disponible sur un des liens de t est inférieure à $b(g)$. Si aucun arbre satisfaisant cette contrainte ne peut être trouvé pour g, le groupe g doit être rejeté. Le but du protocole AQoSM est donc d'accepter un maximum de groupes

quitte à construire davantage d'arbres.

L'algorithme d'agrégation du protocole AQoSM est très similaire à celui du protocole AM. La différence principale est qu'un arbre t est dit candidat si le coût de l'agrégation de g à t est inférieur ou égal à la borne b et si t possède suffisamment de bande passante disponible pour g. Ainsi, même si l'arbre natif $t^{nat}(g)$ associé à g ne possède plus assez de bande passante pour g, un autre arbre peut être trouvé pour g et ce groupe g peut être accepté. Dans le protocole AQoSM, l'arbre natif $t^{nat}(g)$ construit pour g est un arbre des plus courts chemins.

Problèmes du protocole AQoSM

Le protocole AQoSM est basé sur un mécanisme d'agrégation similaire à celui du protocole AM, ce qui en fait un protocole lent. De plus, les résultats de simulation du protocole AQoSM sont assez peu convaincants : le protocole AQoSM permet d'accepter un faible nombre de groupes par rapport au protocole PIM-SM qui ne gère pas les contraintes de bande passante. Nous avons donc cherché un moyen d'accepter plus de groupes que le protocole AQoSM tout en étant plus rapide.

Le protocole Q-STA

Le protocole Q-STA (pour *QoS Scalable Tree Aggregation*) se base sur les mêmes hypothèses que le protocole AQoSM. Nous l'avons proposé dans [MG05].

Le protocole Q-STA est basé sur trois points :

- il construit des arbres natifs utilisant les liens les moins chargés du réseau,
- il utilise un mécanisme d'agrégation similaire à celui du protocole STA,
- il agrège les groupes à des arbres de coût minimum.

Le premier point est une différence importante entre le protocole AQoSM et le protocole Q-STA. Dans le protocole AQoSM, l'arbre natif construit pour un groupe g est un arbre de coût minimum dont la capacité de chaque lien est suffisante pour le groupe. Dans le protocole Q-STA, l'arbre construit est un arbre maximisant la capacité disponible sur ses liens. La figure 4.10(a) montre l'arbre choisi par le protocole AQoSM et la figure 4.10(b) montre l'arbre choisi par le protocole Q-STA pour un groupe g nécessitant 3 unités de bande passante. Notons que les valeurs sur les liens représentent la bande passante disponible sur ces liens. Cette manière de construire l'arbre natif est très efficace pour l'agrégation d'arbres puisqu'elle permet à de futurs groupes d'être agrégés à cet arbre. Elle est présentée sur l'algorithme 7. Le deuxième point permet au protocole Q-STA d'être très rapide. Plutôt que de comparer exhaustivement les arbres de T comme le fait le protocole AQoSM, le protocole Q-STA n'examine que les arbres qui sont susceptibles d'être candidats pour l'agrégation. Le troisième point permet aux arbres construits par le protocole Q-STA de n'utiliser qu'un minimum de ressources. Plutôt que de chercher à agréger à tout prix les premiers groupes en utilisant la bande passante des liens centraux, le protocole Q-STA cherche à économiser cette ressource critique dès le premier groupe, et non dès lors qu'un lien est congestionné.

L'agrégation du protocole Q-STA est décrite sur l'algorithme 8. Nous pouvons noter une différence avec le protocole STA : au lieu de parcourir les arbres dont le coût est compris entre

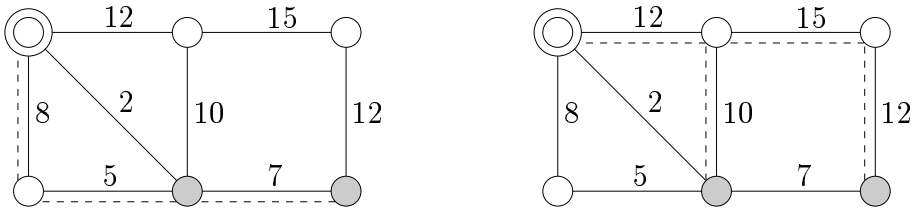


Figure 4.10 – *Construction d'arbres natifs selon (a) le protocole AQoSM et (b) le protocole Q-STA, pour un groupe g nécessitant* 3 *unités de bande passante.*

Pré-requis : g un groupe, G le réseau
Retourne : $t^*(g)$ l'arbre couvrant g et maximisant la bande passante disponible
$b \leftarrow +\infty$
faire
$b \leftarrow$ la plus grande bande passante disponible sur une arête de G qui est strictement inférieure à la valeur courante de b
$\bar{G} \leftarrow$ une copie de G dans laquelle seulement les arêtes de coût supérieur ou égal à b apparaissent
construire (lorsque c'est possible) un arbre $t^*(g)$ couvrant g sur $\bar{G}$
jusqu'à ce que l'arbre $t^*(g)$ existe
retourner $t^*(g)$

Algorithme 7: La construction des arbres natifs dans Q-STA.

$c(t^{nat}(g))$ et $c(t^{nat}(g))(1+b)$, $t^{nat}(g)$ étant l'arbre natif pour g, le protocole Q-STA examine les arbres dont le coût est compris entre $|g|-1$ et $c(t^*(g))$, $|g|$ étant le nombre de membres de g et $t^*(g)$ l'arbre maximisant la bande passante disponible. Dans le protocole Q-STA, les arbres natifs ne sont pas des arbres de coût minimum (ou proche du minimum). Ainsi, nous ne pouvons pas garantir que les arbres de coûts inférieurs à $c(t^*(g))$ ne soient pas de bons candidats à l'agrégation. Nous avons donc descendu la borne inférieure à $|g|-1$. Même si l'agrégation n'a pas forcément lieu avec l'arbre $t^*(g)$ mais à un arbre ayant moins de bande passante, le protocole Q-STA garantit que $t^*(g)$ pourra être utilisé en dernier recours. Dans nos simulations, nous remarquons que l'arbre $t^*(g)$ est souvent sélectionné dès que quelques liens du réseau sont congestionnés.

Pré-requis : T_i l'ensemble des arbres existants de coût i ($i \in [1; +\infty]$), g un groupe nécessitant $b(g)$ unités de bande passante
Retourne : g est accepté ou g est rejeté
$t^*(g)$ un arbre couvrant g maximisant la bande passante disponible
si il existe un lien de $t^*(g)$ ayant moins de $b(g)$ unités de bande passante disponible **alors**
rejeter g
sinon
pour i de $|g|-1$ à $c(t^*(g))$ **faire**
pour $t \in T_i$ **faire**
si t couvre g et tous les liens de t ont au moins $b(g)$ unités de bande passante disponible **alors**
agréger g à t
accepter g
fin si
fin pour
fin pour
agréger g à $t^*(g)$
accepter g
fin si

Algorithme 8: L'algorithme d'agrégation du protocole Q-STA.

Comparaison des protocoles AQoSM et Q-STA

Pour comparer les protocoles AQoSM et Q-STA, nous avons effectué les simulations sur le réseau Eurorings, pour 10000 groupes. Dans nos simulations, nous avons généré aléatoirement 50 % de groupes ayant un besoin de bande passante de 10 Kbps (pour Kilobits par secondes), 30 % ayant un besoin de bande passante de 100 Kbps et 20 % ayant un besoin de bande passante de 1 Mbps (pour Megabits par secondes). Nous avons supposé que la bande passante disponible sur chaque arête du réseau est 1000 Mbps. Chaque point des courbes est la moyenne de 100 simulations.

La figure 4.11 présente le nombre de groupes acceptés par les protocoles PIM-SM (qui ne s'occupe pas de la bande passante disponible), AQoSM et Q-STA. Sur les 10000 groupes, le protocole AQoSM en accepte 500 de plus que le protocole PIM-SM : il est donc possible de faire de l'agrégation d'arbres économisant la bande passante. Le protocole Q-STA accepte 1200 groupes de plus que le protocole PIM-SM, soit 700 groupes de plus que le protocole AQoSM, ce qui en fait un protocole très performant. En effet, si l'on considère le nombre de groupes au-delà du seuil imposé par le protocole PIM-SM, les performances du protocole Q-STA sont plus de deux fois meilleures que celles du protocole AQoSM.

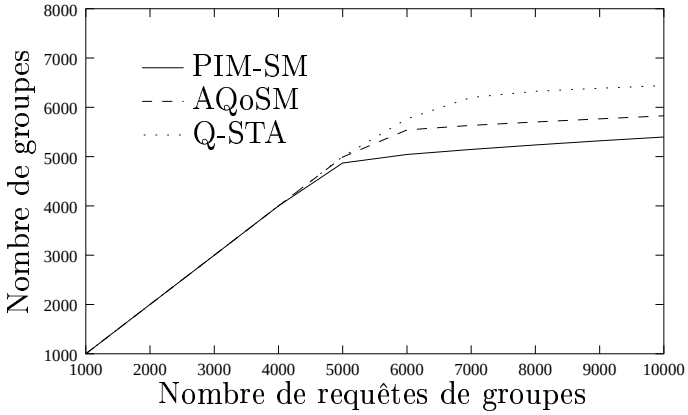


Figure 4.11 – *Nombre de groupes acceptés sur Eurorings.*

Le nombre de groupes acceptés n'est pas une mesure suffisante pour juger de la qualité de l'agrégation, car les groupes ont des besoins de bande passante différents. Il faut s'assurer que le protocole Q-STA accepte les groupes indépendamment de leur bande passante. Si cela n'était pas le cas, il serait facile d'accepter beaucoup de groupes en ne choisissant que ceux ayant un faible besoin de bande passante. La figure 4.12 montre le pourcentage de groupes acceptés en fonction de leur besoin en bande passante. Cette figure montre que le protocole Q-STA est équitable. La même conclusion peut être faite pour les protocoles AQoSM et PIM-SM.

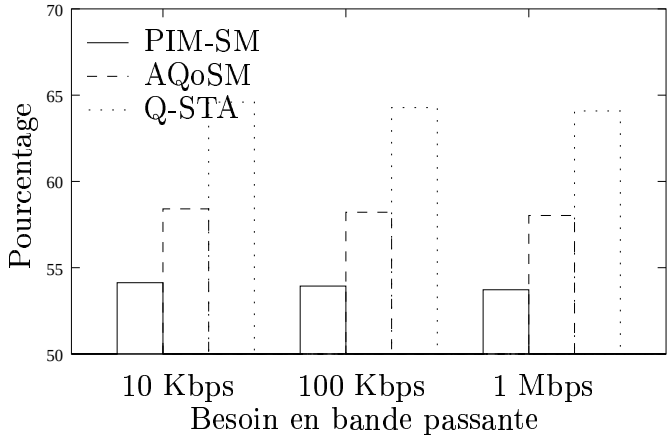


Figure 4.12 – *Pourcentage de groupes acceptés sur Eurorings.*

La figure 4.13 montre le nombre d'arbres construits par les protocoles PIM-SM, AQoSM et Q-STA. Les protocoles AQoSM et Q-STA construisent évidemment beaucoup moins d'arbres que le protocole PIM-SM. Cependant, nous pouvons noter que le protocole Q-STA construit plus d'arbres que le protocole AQoSM mais qu'il accepte plus de groupes. Si l'on calcule le rapport d'agrégation, c'est-à-dire le nombre de groupes agrégés par arbre, les deux protocoles Q-STA et AQoSM ont des performances similaires après 10000 requêtes.

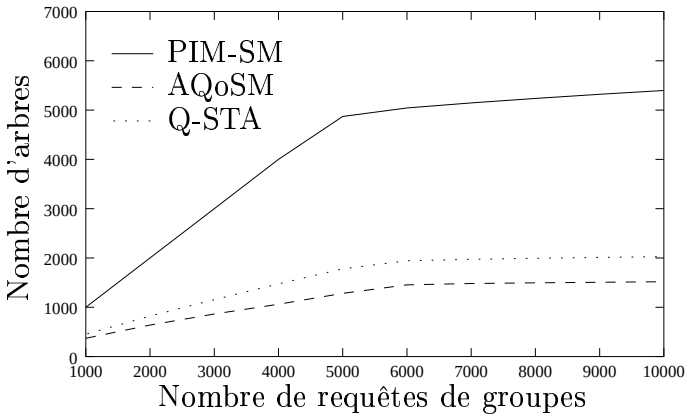


Figure 4.13 – *Nombre d'arbres.*

Résumé

Les protocoles AQoSM et Q-STA montrent qu'il est possible de combiner l'agrégation d'arbres et la gestion de la bande passante. L'agrégation d'arbres peut être équitable et peut conduire à l'acceptation d'un grand nombre de groupes.

Les résultats de simulation du protocole Q-STA montrent que la bande passante peut être exploitée très efficacement sans que le nombre d'arbres construit ne devienne trop important. Le protocole Q-STA accepte beaucoup plus de groupes que le protocole AQoSM mais, pour cela, il construit un peu plus d'arbres.

4.3.3 Les agrégations non centralisées

Les protocoles utilisant l'agrégation d'arbres sont le plus souvent basés sur le mécanisme centralisé du protocole AM. Dans cette partie, nous décrivons le protocole BEAM qui réalise une agrégation décentralisée. Puis, nous mettons en avant des problèmes importants du modèle existant, c'est-à-dire de tous les protocoles actuels réalisant l'agrégation d'arbres. Parmi ces problèmes, il y a notamment la latence d'agrégation et la robustesse. Pour résoudre ces problèmes, nous proposons le protocole DMTA qui réalise une agrégation distribuée réduisant la latence d'agrégation et augmentant la robustesse.

Le protocole BEAM

Le protocole BEAM (pour *Bidirectional Aggregated Multicast*) est décrit dans [CLMG03]. Il est très similaire au protocole AM avec un gestionnaire d'agrégation décentralisé entre

plusieurs routeurs noyaux.

Chaque routeur noyau n dispose d'un ensemble d'arbres T_n dont il est responsable. Cet ensemble d'arbre T_n dépend des agrégations qui ont eu lieu. De plus, chaque routeur noyau n est associé à un ensemble d'adresses *multicast*. La fonction de correspondance entre l'adresse du groupe et le routeur noyau est une fonction de hachage de l'adresse du groupe. Dans la spécification du protocole BEAM, il s'agit de l'adresse du groupe modulo le nombre de routeurs noyaux du domaine. Nous noterons $n(g)$ le routeur noyau associé à l'adresse g (et nous avons $n(g) = g \bmod b$, b étant le nombre de routeurs noyaux dans le domaine). Si les adresses des groupes sont bien réparties, la fonction de hachage du protocole BEAM permet de répartir la charge des routeurs noyaux.

La figure 4.14 décrit le fonctionnement du protocole BEAM. Lorsqu'un groupe g apparaît dans le réseau (événement 1), le routeur noyau $n(g)$ est averti (message 2) et tente d'agréger g avec un arbre de son ensemble d'arbres $T_{n(g)}$ (algorithme 3a). S'il ne trouve aucun candidat, $n(g)$ contacte tous les autres routeurs noyaux (message 3b) pour qu'ils tentent d'agréger g avec un arbre de leur ensemble (algorithme 3c). Finalement, les routeurs noyaux proposant une agrégation préviennent $n(g)$ (message 3d). En fonction de ces réponses, le routeur noyau $n(g)$ décide de l'arbre auquel agréger g (algorithme 3e). Finalement, tous les routeurs de bordure de l'arbre couvrant g sont avertis (message 4). Rappelons que les étapes 3b, 3c, 3d et 3e n'ont lieu que si le groupe g n'a pas pu être agrégé avec un arbre de son routeur noyau $n(g)$. C'est pourquoi elles sont indiquées en pointillés sur la figure 4.14.

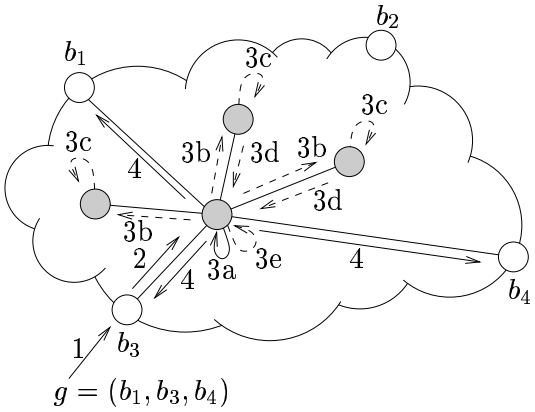


Figure 4.14 – *L'agrégation avec le protocole BEAM.*

Le modèle décentralisé du protocole BEAM souffre d'un problème de robustesse. De plus, des problèmes de latence sont aussi présents dans le modèle centralisé du protocole AM, et par conséquent dans tous les protocoles existants réalisant l'agrégation d'arbres. Après avoir présenté ces problèmes, nous décrivons notre proposition d'agrégation distribuée.

Problèmes du protocole BEAM

Il y a de nombreux problèmes dans le modèle d'agrégation d'arbres existant, et notamment dans le protocole BEAM.

Premièrement, la gestion des groupes dans les protocoles AM et BEAM induit une grande latence. Cette latence est due à l'envoi des messages 2 et 4 et à l'algorithme 3a. Dans le cas

de BEAM, si aucun arbre candidat n'est trouvé par le routeur noyau désigné d'un groupe, il faut ajouter le délai dû aux messages 3b et 3d et aux algorithmes 3c et 3e. Chaque arrivée d'un nouveau groupe, chaque départ d'un groupe et chaque changement de groupe subissent cette latence (lorsqu'un groupe change, si l'arbre agrégé de ce groupe ne change pas, il ne faut pas inclure le message 4 dans le calcul de la latence). Il convient de noter, cependant, que l'algorithme 3a pour le protocole BEAM est plus rapide que l'algorithme 3a pour le protocole AM, puisque l'espace de recherche est plus petit et que la fonction de comparaison est plus rapide.

Deuxièmement, la gestion de la dynamicité des groupes introduit beaucoup de messages de contrôle, notamment les messages 2, 3b, 3d et 4. Lorsque les groupes sont très dynamiques, ces messages utilisent une partie importante de la bande passante et peuvent surcharger le réseau.

Troisièmement, le gestionnaire d'agrégation du protocole AM (et des protocole d'agrégation centralisés) et les routeurs noyaux du protocole BEAM (et des protocoles d'agrégation décentralisés) sont des points critiques. Dans les protocoles centralisés, si le gestionnaire d'agrégation tombe en panne, plus aucun groupe ne peut être accepté et les arbres associés aux groupes ne pourront pas changer, même si les membres du groupe changent. Dans les protocoles décentralisés, si un routeur noyau tombe en panne, plus aucun des groupes associé à ce routeur noyau ne peut être accepté, BEAM ne définissant pas de routeur de remplacement. Les arbres associés aux groupes gérés par ce routeur noyau ne pourront pas changer non plus.

Quatrièmement, les protocoles actuels construisent des états inutiles dans les routeurs de bordure. En effet, ces protocoles sont basés sur la construction d'arbres partagés, utilisant moins d'états que les arbres basés à la source. Quand un groupe g est associé à l'arbre t de label l, tous les routeurs de bordure de t sont configurés pour l'association $g \rightarrow l$. Cependant, cette association n'est utile pour un routeur de bordure que si une source du groupe désire effectivement émettre des paquets pour g. Même si les groupes ont potentiellement beaucoup de sources, tous les membres ne sont pas toujours des sources.

Il est très important de résoudre ces quatre problèmes. Nous proposons un nouveau modèle, basé sur une agrégation distribuée, qui les résout. Nous proposons le protocole DMTA (pour *Distributed Multicast Tree Aggregation*) basé sur ce nouveau modèle.

L'agrégation distribuée et le protocole DMTA

L'agrégation distribuée que nous proposons est basée sur l'utilisation de labels significatifs. Soit $l(t)$ le label associé à un arbre t. Une fonction de signification s associe au label $l(t)$ l'ensemble des routeurs de bordure couverts par l'arbre t. Ainsi, un routeur de bordure quelconque connaissant tous les labels et la fonction s est en mesure de réaliser une agrégation localement, sans avoir à contacter un routeur gestionnaire. Pour que ce mécanisme soit rapide, nous devons faire en sorte que la fonction de signification s puisse être évaluée rapidement.

Pour obtenir l'ensemble des labels disponibles, nous avons besoin d'une nouvelle entité dans le domaine : le gestionnaire de maintenance. Le gestionnaire de maintenance gère uniquement des informations concernant les arbres, aucune information concernant les groupes.

Il est en charge de configurer et de maintenir les arbres et de leur attribuer un label (respectant la fonction de signification s). C'est lui qui informe les routeurs de bordure de l'ensemble des labels disponibles. Le gestionnaire de maintenance est aussi en charge de changer les arbres suite aux pannes de liens ou de routeurs.

À présent que les deux idées principales de l'agrégation distribuée sont décrites, nous pouvons présenter le protocole DMTA (pour *Distributed Multicast Tree Aggregation*) basé sur ce mécanisme.

Description du protocole DMTA. La figure 4.15 décrit le fonctionnement du protocole DMTA. Lorsqu'un groupe g apparaît (événement 1), le routeur de bordure b détectant ce changement doit trouver un arbre auquel agréger g. À partir de l'ensemble des labels disponibles et de la fonction de signification s, le routeur de bordure b est capable de trouver un arbre pouvant couvrir g (algorithme 3a). Ce routeur de bordure n'a pas à prévenir les autres routeurs de bordure de l'agrégation : si une source connectée à un autre routeur de bordure b' désire émettre pour le groupe, le routeur de bordure b' cherchera un arbre auquel agréger g. Les labels trouvés par b et b' peuvent être différents sans que cela ne pose de problèmes particuliers. Dans notre modèle, les arbres sont toujours bidirectionnels mais ils ne sont pas utilisés systématiquement dans les deux sens pour un même groupe. Cette particularité fait qu'il n'existe un état spécifique au groupe g que dans les routeurs de bordure connectés à des sources réelles des groupes.

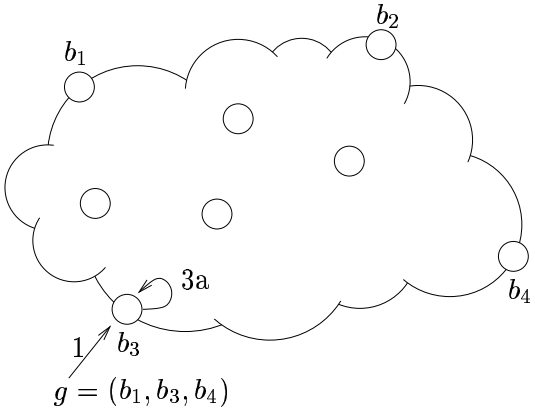


Figure 4.15 – *L'agrégation avec le protocole DMTA.*

Dans le protocole DMTA, la fonction de signification s associe à un label $l(t)$ l'ensemble des routeurs de bordure couverts par t au moyen d'un vecteur de bits : si le i-ème bit vaut 1, cela indique que le i-ème routeur de bordure est couvert par l'arbre. Ainsi, le label 1010 correspond à un arbre couvrant les routeurs de bordure b_1 et b_3.

Le gestionnaire de maintenance a deux tâches : construire les arbres initialement et les reconfigurer en cas de panne. Ainsi, dans la plupart des cas, c'est-à-dire quand il n'y a pas de pannes, il n'est pas utilisé. Notons aussi que le gestionnaire de maintenance n'intervient pas (directement) dans le processus d'agrégation.

Construction initiale des arbres. Dans le protocole DMTA, le gestionnaire de maintenance construit l'ensemble des labels disponibles initialement. Supposons qu'il y ait B routeurs de bordure. Le gestionnaire de maintenance construit les 2^B groupes *multicast* dont les membres sont des routeurs de bordure. Pour chaque groupe g, il construit un arbre natif $t(g)$. Le label $l(t(g))$, représentant tous les routeurs couverts par l'arbre, est ajouté à l'ensemble des labels disponibles s'il n'est pas déjà présent. Comme il y a beaucoup moins de labels différents que de groupes différents, il y a peu d'arbres à configurer par rapport aux 2^B. Pour s'en convaincre, considérons l'arbre représenté sur la figure 4.16. Cet arbre couvre le groupe (b_2, b_4) mais il couvre aussi les groupes (b_1, b_2, b_4), (b_2, b_3, b_4) et (b_1, b_2, b_3, b_4) sans perte de bande passante (il est d'ailleurs l'arbre natif pour ces quatre groupes). Il ne couvre pas le groupe (b_1, b_4) si l'on n'autorise pas de perte de bande passante.

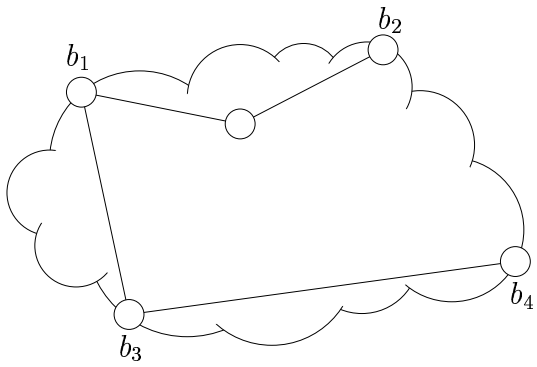


Figure 4.16 – *L'arbre natif pour (b_2, b_4) est natif pour trois autres groupes.*

Le tableau 4.1 présente le nombre d'arbres distincts que nous avons obtenu sur plusieurs réseaux[3]. Les arbres natifs construits sont des arbres des plus courts chemins, pour plusieurs raisons :

- nous pouvons calculer mathématiquement le nombre d'arbres natifs construits par des plus courts chemins (voir plus loin),
- les arbres des plus courts chemins sont souvent utilisés par les protocoles,
- la construction d'arbres des plus courts chemins offre de meilleures performances que la construction utilisant des arbres de coût faible.

Pour les grands réseaux, tels que Géant (pour *Gigabit European Academic Network*, un réseau dorsal européen), il peut paraître non envisageable de configurer initialement les 8222 arbres[4]. Dans ce cas, le gestionnaire de maintenance peut décider de ne construire qu'un sous-ensemble de ces 8222 arbres, en autorisant de la perte de bande passante : si deux arbres couvrent presque le même ensemble de nœuds, l'un des deux (celui dont le label a le moins de 1) sera supprimé. Moins d'arbres seront tolérés, plus il faudra autoriser de perte de bande passante.

3. Dans le cas d'Eurorings, nous n'avons pas pu calculer le nombre d'arbres distincts. Cependant, nous avons pu calculer le nombre d'arbres distincts enracinés en chaque source. La somme de ces nombres nous donne une borne supérieure du nombre d'arbres distincts.

4. Cela est d'autant plus vrai pour les réseaux Renater et Eurorings.

Topologie	Nombre de nœuds	Nombre de groupes différents	Nombre d'arbres distincts
Abilène	11	2048	131 (soit 6.40 % des groupes)
Nsfnet	14	16384	958 (soit 5.85 % des groupes)
Géant	18	262144	8222 (soit 3.14 % des groupes)
Renater	25	33554432	532203 (soit 1.59 % des groupes)
Eurorings	43	environ $8.7 \cdot 10^{12}$	moins de $1.2 \cdot 10^{8}$ (soit moins de 0.0014 % des groupes)

TAB. 4.1 – *Le nombre d'arbres distincts est très petit par rapport au nombre de groupes différents.*

Le nombre d'arbres distincts est bien inférieur aux 2^B groupes possibles. Pour s'en donner une intuition mathématique, notons t_n un arbre de racine n et considérons le nombre $ad(t_n)$ d'arbres distincts extraits de t_n et le nombre $gd(t_n)$ de groupes distincts extraits de t_n. Les formules 4.1 et 4.2 donnent la définition de ces deux nombres.

$$ad(t_n) = \begin{cases} 1 & \text{si } t_n \text{ ne couvre que le nœud } n \\ \prod_{c \text{ fils de } n \text{ dans } t_n} (1 + ad(t_c^n)) & \text{sinon} \end{cases}, \qquad (4.1)$$

où t_c^n désigne le sous-arbre de t_n dont la racine est c. Cette formule est valide puisque le sous-arbre d'un arbre des plus courts chemins reste un arbre des plus courts chemins. Cette formule n'est plus valide si l'on considère des arbres de coût minimum (ou approché).

$$gd(t_n) = \begin{cases} 1 & \text{si } t_n \text{ ne couvre que le nœud } n \\ \prod_{c \text{ fils de } n \text{ dans } t_n} (2.gd(t_c^n)) & \text{sinon} \end{cases}. \qquad (4.2)$$

On a bien $gd(t_n) = 2^{B-1}$ si t_n couvre B nœuds (t_n couvrant obligatoirement le nœud n).

Étant donné une racine r et un arbre des plus courts chemins t_r, le nombre d'arbres distincts $ad(t_r)$ est très petit devant le nombre de groupes distincts $gd(t_r)$. À titre d'exemple, considérons l'arbre t_r représenté sur la figure 4.17. On a $ad(t_r) = (1+4) \cdot (1+2) \cdot (1+8) = 135$ et $gd(t_r) = (2 \cdot 4) \cdot (2 \cdot 2) \cdot (2 \cdot 8) = 512$.

Le protocole DMTA résout les quatre problèmes de BEAM. Premièrement, la latence introduite par le protocole DMTA ne vient que de l'algorithme 3a. Cet algorithme est très rapide car il suit un procédé similaire à celui du protocole STA. Contrairement aux protocoles basés sur l'ancien modèle d'agrégation d'arbres, la latence du protocole DMTA n'est pas ralentie par l'envoi de messages de contrôle puisque l'agrégation est réalisée localement.

Deuxièmement, le protocole DMTA utilise très peu de messages de contrôle. Le protocole DMTA n'utilise que des messages de contrôle pour configurer et maintenir les arbres, ainsi que pour informer les routeurs de bordure des labels disponibles. Les messages de configuration et de maintenance des arbres, dont le nombre est de l'ordre du nombre d'arbres, existent aussi

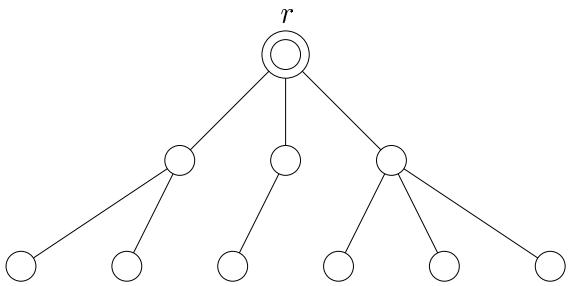


Figure 4.17 – 135 *arbres suffisent à couvrir les* 512 *groupes distincts.*

dans l'ancien modèle d'agrégation d'arbres. Les messages d'information des labels disponibles ne sont utilisés que lors du démarrage du réseau ou lorsque des changements de topologie modifient les arbres existants.

Troisièmement, le protocole DMTA est beaucoup plus robuste que les protocoles basés sur l'ancien modèle d'agrégation d'arbres. En effet, si un routeur de bordure tombe en panne (ce qui est similaire au cas d'un gestionnaire d'agrégation tombant en panne dans AM ou d'un routeur noyau tombant en panne dans BEAM), les autres routeurs de bordure ne sont pas influencés dans DMTA. De nouveaux groupes peuvent apparaître et les membres des groupes actuels peuvent changer, ce qui n'était pas le cas avec les protocoles AM et BEAM. Si le gestionnaire de maintenance du protocole DMTA tombe en panne, les labels disponibles ne peuvent plus changer. Cependant, de nouveaux groupes peuvent apparaître et les membres des groupes actuels peuvent changer. Cela ne gêne pas les mécanismes d'agrégation, tant qu'aucun changement de topologie ne détruit les arbres configurés.

Quatrièmement, le protocole DMTA ne construit des états pour un groupe g que dans les routeurs de bordure connectés aux sources réelles de g, et non pas dans tous les routeurs de bordure de l'arbre auquel g est agrégé. Ainsi, le nombre d'entrées spécifiques aux groupes est beaucoup plus petit avec le protocole DMTA qu'avec les protocoles AM et BEAM. Dans le pire des cas, quand tous les routeurs de bordure d'un groupe sont aussi des sources, les protocoles AM, BEAM et DMTA construisent autant d'entrées spécifiques aux groupes.

On peut aussi mentionner que l'ensemble des labels disponibles dans chaque routeur de bordure est très stable. En effet, même si le gestionnaire de maintenance décide de changer un arbre t en un arbre t', il se peut que le nouveau label $l(t')$ soit égal à l'ancien $l(t)$. Dans ce cas, la modification de l'arbre est transparente pour les routeurs de bordure et ils n'ont pas à en être informés, ce qui réduit davantage le nombre de messages de contrôle.

Comparaison des protocoles BEAM et DMTA

Dans ce paragraphe, nous présentons nos simulations des protocoles AM, BEAM et DMTA. Les simulations concernent le réseau Nsfnet, pour 10000 groupes. La taille des groupes est choisie aléatoirement entre 2 et le nombre de nœuds du réseau.

La figure 4.18 présente le nombre d'arbres construits par les protocoles AM, BEAM et DMTA. Notons d'abord que le nombre d'arbres construit par le protocole AM est égal au nombre d'arbres construits par le protocole BEAM. Le nombre d'arbres construits par le protocole DMTA est toujours égal à 958 (voir tableau 4.1), car ces arbres sont construits

initialement. Dès lors que 4000 groupes sont apparus, cette approche est plus efficace.

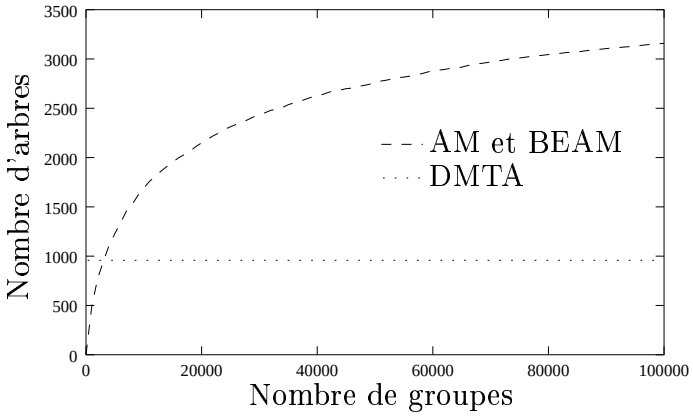


Figure 4.18 – *Nombre d'arbres.*

La figure 4.19 présente le nombre de messages de contrôle requis par les trois protocoles AM, BEAM et DMTA. Le nombre de messages de contrôle requis par le protocole DMTA est constant (et égal à 958, puisque chaque arbre est construit par un unique message). Le nombre de messages de contrôle requis par le protocole AM est bien plus important, ce qui pose des problèmes de passage à l'échelle. Le protocole BEAM utilise davantage de messages de contrôle.

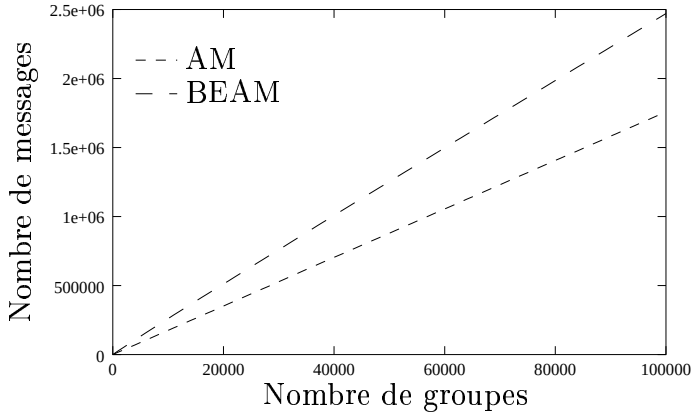


Figure 4.19 – *Nombre de messages de contrôle.*

Finalement, la figure 4.20 présente le nombre d'états spécifiques aux groupes divisé par le nombre de routeurs de bordure. Il s'agit donc de la taille moyenne des tables associant les groupes aux labels. Nous pouvons voir que le protocole DMTA est bien meilleur que les protocoles AM et BEAM, ayant tous deux les mêmes performances.

Résumé

Il est possible de réaliser une agrégation d'arbres distribuée nécessitant très peu de messages de contrôle et très robuste. Le protocole DMTA en est une illustration. En donnant

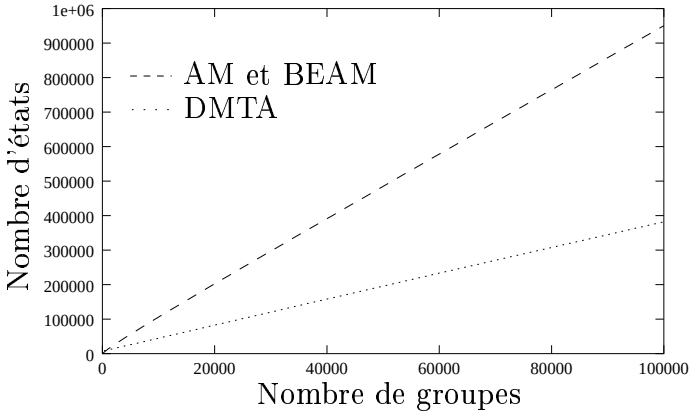


Figure 4.20 – *Nombre d'états.*

une sémantique aux labels, le protocole DMTA est capable de réaliser une agrégation localement, indépendamment des autres routeurs de bordure. Nous avons obtenu une très grande robustesse en découplant la construction des arbres du mécanisme d'agrégation : même si le gestionnaire de maintenance tombe en panne, les conséquences sont minimes.

4.4 Conclusion

L'agrégation d'arbres permet le passage à l'échelle en nombre de groupes en réduisant le nombre d'entrées *multicast* et le nombre de messages de contrôle. Nous avons étudié l'agrégation d'arbres selon plusieurs aspects : les compromis entre bande passante et temps de calcul pour l'agrégation afin de bien comprendre l'impact des différents paramètres de l'agrégation d'arbres, l'intégration de la QoS afin de relâcher une hypothèse forte, puis l'agrégation robuste et efficacement distribuée.

Nos trois propositions, les protocoles STA, Q-STA et DMTA sont tous basés sur une agrégation rapide. Nos contributions majeures sont les suivantes :

- Nous avons montré que, même dans un domaine de b routeurs de bordure, le nombre d'arbres différents était bien inférieur à 2^b. Nous avons d'ailleurs fourni un algorithme permettant de calculer le nombre d'arbres différents. Connaître cette valeur permet de savoir si l'agrégation d'arbres est applicable efficacement dans un domaine ou non. Nous avons d'ailleurs pu mettre en évidence que l'agrégation n'était pas envisageable sur des domaines de grande taille.
- Le protocole Q-STA permet d'accepter significativement plus de groupes que les protocoles existants.
- Le protocole DMTA nécessite peu de messages de contrôle. Il implique une faible latence de traitement des groupes et possède une grande robustesse.

Références

[BC05] A. Boudani et B. Cousin. « An hybrid explicit multicast/unicast recursive approach for multicast routing ». *Computer Communications*, 2005.

[CFD01] L. H. M. K. Costa, S. Fdida, et O. C. M. B. Duarte. « Hop-by-Hop Multicast Routing Protocol ». Dans *ACM SIGCOMM*, pages 249–259, août 2001.

[CKF+02] J.-H. Cui, J. Kim, A. Fei, M. Faloutsos, et M. Gerla. « Scalable QoS Multicast Provisioning in Diff-Serv-Supported MPLS Networks ». Dans *IEEE Globecom*, novembre 2002.

[CKM+02] J.-H. Cui, J. Kim, D. Maggiorini, K. Boussetta, et M. Gerla. « Aggregated Multicast – A Comparative Study ». Dans *IFIP Networking*, numéro 2345 dans Springer, LNCS, pages 1032–1044, mai 2002.

[CKM+03] J.-H. Cui, J. Kim, D. Maggiorini, K. Boussetta, et M. Gerla. « Aggregated Multicast – A Comparative Study ». *Special issue on Cluster Computing: The Journal of Networks*, 2003.

[CLMG03] J.-H. Cui, L. Lao, D. Maggiorini, et M. Gerla. « BEAM: A Distributed Aggregated Multicast Protocol Using Bi-directional Trees ». Dans *IEEE International Conference on Communications (ICC)*, mai 2003.

[DBCP97] M. Degermark, A. Brodnik, S. Carlsson, et S. Pink. « Small Forwarding Tables for Fast Routing Lookups ». Dans *ACM SIGCOMM*, pages 3–14, 1997.

[GFCF01] M. Gerla, A. Fei, J.-H. Cui, et M. Faloutsos. « Aggregated Multicast for Scalable QoS Multicast Provisioning ». Dans *Tyrrhenian International Workshop on Digital Communications (IWDC)*, septembre 2001.

[GM05] A. Guitton et J. Moulierac. « Scalable Tree Aggregation for Multicast ». Dans *International Conference on Telecommunications (ConTEL)*, juin 2005.

[LDL04] Z.-F. Liu, W.-H. Dou, et Y.-J. Liu. « AMBTS: A Scheme of Aggregated Multicast Based on Tree Splitting ». Dans *IFIP Networking*, numéro 3042 dans Springer, LNCS, pages 829–840, mai 2004.

[MG05] J. Moulierac et A. Guitton. « QoS Scalable Tree Aggregation ». Dans *IFIP Networking*, numéro 3462 dans Springer, LNCS, pages 1405–1408, mai 2005.

[PG98] J. Pansiot et D. Grad. « On routes and multicast trees in the Internet ». *ACM Computer Communication Review*, 28(1):41–50, janvier 1998.

[REG99] P. Radoslavov, D. Estrin, et R. Govindan. « Exploiting the bandwidth-memory tradeoff in multicast state aggregation ». Technical Report 99-697, Department of Computer Science, USC, février 1999.

[SENZ00] I. Stoica, T. S. Eugene Ng, et H. Zhang. « REUNITE: A Recursive Unicast Approach to Multicast ». Dans *IEEE Infocom*, numéro 3, pages 1644–1653, 2000.

[TH00] D. Thaler et M. Handley. « On the Aggregatability of Multicast Forwarding State ». Dans *IEEE INFOCOM*, numéro 3, pages 1654–1663, 2000.

Conclusion

Le routage *multicast* permet de réaliser des communications de groupe en économisant les ressources d'un réseau, mais soulève de nombreux problèmes. Le modèle de Deering a été proposé il y a quelques années pour résoudre la plupart des problèmes identifiés à l'époque. Cependant, l'émergence de réseaux de nouvelle génération rend caduque certaines hypothèses de base du modèle de Deering. Dans les réseaux tout optiques, la transmission *multicast* est basée sur la configuration d'arbres optiques. Dans ce contexte, l'incapacité de certains routeurs tout optique à dupliquer les signaux est très contraignante. Il est aussi envisagé que dans un proche avenir, le nombre de groupes *multicast* deviennent très important. Contrairement au routage *unicast*, le routage *multicast* actuel ne passe pas à l'échelle, et il sera donc très difficile de faire coexister de nombreux groupes.

Dans cette thèse, nous étudions des alternatives au modèle de Deering, dans le cadre des réseaux tout optique et de la résistance au facteur d'échelle.

Les réseaux tout optique

Les réseaux tout optique sont un exemple type de réseaux de nouvelle génération. La technologie sous-jacente, encore peu mature, impose de nombreuses contraintes sur les divers équipements. Nous nous sommes concentrés sur l'incapacité de certains routeurs optiques à dupliquer un signal d'entrée en plusieurs signaux de sortie. De tels routeurs ne peuvent pas être nœuds de branchement des arbres *multicast*. Sans cette hypothèse majeure du *multicast*, même l'existence d'un unique arbre couvrant n'est pas assurée.

Contributions

La construction d'un arbre de coût minimum dont les nœuds ont un degré limité est un problème NP-difficile, assez proche du problème de Steiner. C'est pourquoi les heuristiques

proposées dans la littérature sont souvent adaptées à partir d'heuristiques pour le problème de Steiner. Dans la première partie du chapitre 2, nous avons classé les heuristiques existantes en plusieurs catégories, selon la manière dont elles ont été adaptées. Pour chaque catégorie, nous avons proposé des heuristiques originales visant à palier les défauts des heuristiques existantes. Nous avons proposé une heuristique très performante, l'heuristique CTH, qui a un très bon comportement sur toutes les métriques que nous nous étions fixées. La complexité relativement importante de l'heuristique n'est pas un problème majeur puisque cette heuristique n'est utilisée qu'au niveau du plan de contrôle où un léger délai dû au calcul est tolérable.

La plupart des heuristiques pour les réseaux tout optique tentent de réduire le coût des arbres construits, quitte à augmenter leur longueur. Cela ne semble pas contraignant puisque le délai de propagation dans un réseau tout optique est négligeable. Toutefois nous avons montré dans la seconde partie du chapitre 2 que l'augmentation de la longueur des arbres construits pouvait être très néfaste pour les communications. Plus les chemins sont longs, plus ils utilisent d'amplificateurs et plus les signaux optiques doivent être régénérés un grand nombre de fois, ce qui implique qu'un grand nombre de régénérateurs doivent être présents dans le réseau. De plus, les chemins longs induisent une grande dispersion, ce qui dégrade les performances du réseau. Construire des arbres des plus courts chemins, toujours en considérant l'incapacité de certains routeurs à dupliquer, est un problème pratique important. Nous l'avons donc étudié d'un point de vue théorique. La première remarque que nous avons faite est que la limitation du degré des nœuds est une contrainte trop forte. Il est possible de réutiliser les nœuds non duplicateurs déjà utilisés, pourvu que le nombre de signaux d'entrée soit supérieur ou égal au nombre de signaux de sortie. Ensuite, nous avons pu montrer qu'il était NP-difficile de trouver un tel arbre de longueur minimale. Dans le cas général, il est d'ailleurs impossible de trouver des $(2-\varepsilon)$-approximations, pour $\varepsilon > 0$, même si un seul nœud est incapable de dupliquer. Ce résultat montre que le problème est très difficile d'un point de vue théorique.

Perspectives

Dans les réseaux tout optique, un autre problème *multicast* très important, mais que nous n'avons pas étudié, est le problème d'assignation de longueurs d'onde. L'assignation de longueurs d'onde consiste à associer une longueur d'onde à chaque arbre optique obtenu par le protocole de routage. Les liens entre le routage et l'assignation des longueurs d'onde sont très forts. Ce travail de thèse pourrait se continuer sur l'étude de l'impact des heuristiques de routage proposées sur l'assignation de longueurs d'onde.

Une autre piste très prometteuse touche au placement des régénérateurs optiques. Nous savons qu'il est nécessaire de placer des régénérateurs à intervalles réguliers sur les fibres optiques. Cependant, le placement des régénérateurs sur des courtes fibres optiques reste problématique. En effet, l'absence de régénérateurs sur une succession de plusieurs courtes fibres optiques est néfaste pour la communication, tandis que la pose systématique d'un régénérateur est coûteuse. Intégrer des hypothèses sur le routage aux études actuelles sur le placement des régénérateurs est d'un grand intérêt.

Dans le cadre de la construction d'un arbre réutilisant les nœuds non duplicateurs, le

résultat d'inapproximabilité par une constante strictement inférieure à 2 met en doute l'approximabilité du problème par une constante quelconque. Dans le cas général, nous avons pu obtenir une $\mathcal{O}(n/\log n)$-approximation, n étant le nombre de nœuds du graphe, mais il serait intéressant de considérer la $\mathcal{O}(\log n)$-approximabilité. Parallèlement, l'approximabilité sur des instances particulières de graphes pourrait être étudiée.

Le passage à l'échelle

Dans un avenir proche, le nombre de groupes *multicast* concurrents risque d'être très grand. Le modèle de Deering n'est pas adapté à cette croissance. En effet, chaque routeur doit stocker un état *multicast* par arbre qui le traverse. Cela se traduit par une utilisation très importante de la mémoire des routeurs et réduit la vitesse d'acheminement des paquets. De plus, chaque arbre doit être maintenu au moyen de messages de contrôle. Le grand nombre d'arbres implique une consommation importante des ressources du réseau.

Contributions

Le chapitre 3 développe l'idée d'éliminer les états de routage pour les petits groupes *multicast*. Ramenée au nombre faible de membres des petits groupes, la surcharge de contrôle est importante. Comme ils risquent d'être très nombreux à moyen terme, il apparaît primordial de se concentrer sur ces groupes. L'une des méthodes permettant de supprimer les états de routage *multicast* pour les petits groupes consiste à stocker explicitement dans chaque paquet de données les informations de routage *multicast*. C'est sur ce routage explicite que nous nous sommes focalisés. Nous avons identifié deux classes de protocoles explicites, les « plats » qui ne stockent que la quantité minimale d'information, et les « arborescents » qui stockent l'arbre de routage complet. Dans les deux cas, la quantité d'informations de contrôle présente dans chaque paquet peut être très importante et peut conduire à une fragmentation des paquets générés, ce qu'il faut éviter. Nous avons donc étudié la segmentation des paquets à la source, ce qui a pour rôle d'éviter la fragmentation, et l'impact de cette segmentation sur les performances. Dès lors que se pose la question de segmentation, se pose aussi la question du regroupement des informations segmentées. Cette piste a été détaillée pour les protocoles explicites plats. Nous avons aussi montré que pour obtenir les meilleures performances, il fallait déployer les protocoles explicites plats principalement sur les routeurs de bordure. Pour le routage explicite arborescent, nous avons proposé d'intégrer une protection explicite aux paquets, ce qui est novateur.

Le chapitre 4 se base sur l'agrégation d'arbres pour réduire le nombre d'états dans les routeurs. Le principe de l'agrégation d'arbres consiste à associer à plusieurs groupes un même arbre, au sein d'un domaine de routage. Ainsi, il y a moins d'arbres dans le réseau et donc, moins d'états de routage et de messages de contrôle. Une sur-agrégation peut aussi être envisagée, au détriment de la bande passante. Nos propositions de protocoles améliorent l'état de l'art sur trois points : l'agrégation est effectuée plus rapidement, plus de groupes sont acceptés dans le domaine et le protocole est mieux distribué. Toutefois, les performances de tous les mécanismes d'agrégation d'arbres sont limités par le nombre d'arbres différents que l'on peut construire dans un domaine. Puisqu'un même arbre peut couvrir plusieurs groupes

différents, le nombre d'arbres différents est très inférieur au nombre de groupes différents. À partir de cette remarque, nous avons étendu les mécanismes d'agrégation aux domaines de taille moyenne. Cette analyse peut être appliquée dans tous les algorithmes d'agrégation d'arbres existants pour améliorer leurs performances. D'un point de vue théorique, elle permet de calculer des bornes sur le passage à l'échelle de tels mécanismes.

Perspectives

Certains protocoles explicites plats sont en cours de standardisation et de déploiement sur Internet. Les résultats de notre analyse du déploiement incrémental nous montre qu'il est préférable de déployer ces protocoles d'abord en bordure des réseaux, ce qui pourrait orienter les choix des opérateurs. D'un autre côté, il serait très instructif de continuer l'étude du déploiement incrémental pour la situation actuelle, ce qui donnerait un cadre réel. Une autre perspective des travaux sur le routage explicite plat concerne les réseaux mobiles ou sans infrastructure fixe. Le déplacement fréquent des nœuds ou des routeurs dans ces réseaux rend coûteuse la maintenance des arbres. Comme le *multicast* explicite ne construit pas d'arbres, il semble pouvoir être appliqué à ce type de réseaux. Des approches explicites commencent à émerger dans ce cadre.

Concernant le routage explicite arborescent, nous avons remarqué que l'étude de la segmentation était très ardue. La poursuite de cette étude pourrait déboucher sur un grand nombre de résultats algorithmiques importants, comme par exemple, la construction d'arbres ayant peu de nœuds de branchement. De tels arbres peuvent être utilisés dans de nombreux de domaines. Ajoutons aussi qu'actuellement, il n'existe aucune étude sur le stockage explicite de la protection dans ces protocoles. Cette technique semble pourtant offrir beaucoup d'avantages, tels qu'une protection dynamiquement ajustable, une protection localisée ou globale, ou encore peu dépendante du temps de convergence des protocoles sous-jacents. Notons que, l'ajout de la protection aux informations de routage justifie le besoin de segmentation, puisque les informations de contrôle occupent une place d'autant plus grande.

La robustesse des protocoles d'agrégations d'arbres est peu étudiée dans la littérature. Lorsqu'un lien ou un routeur du domaine tombe en panne, il peut être nécessaire de reconstruire plusieurs arbres ou de changer les agrégations actuelles. Cela est d'autant plus vrai quand c'est le gestionnaire d'agrégations lui-même qui tombe en panne. Notre approche utilisant des labels significatifs pour les arbres peut engendrer une certaine stabilité des états de routage : même si les arbres sont reconfigurés, les gestionnaires d'agrégations n'ont pas à être informés pourvu que les labels des arbres restent inchangés. La stabilité des états réduit le nombre de messages de contrôle, puisque les changements d'états sont moins fréquents. Un autre axe de travail intéressant est l'étude de l'agrégation d'arbres dans les grands domaines. La hiérarchisation ou le découpage du domaine sont deux voies qui sont en cours d'étude au sein de l'équipe. Ces voies permettraient de réduire considérablement le nombre d'arbres dans le domaine, et pourrait rendre l'agrégation d'arbres extensible au facteur d'échelle.

Annexe A

Glossaire

AAP Address Allocation Protocol.
ADSL Asynchronous Digital Subscriber Line.
AM Aggregated Multicast.
AMBTS Aggregated Multicast Based on Tree Splitting.
AQoSM Aggregated QoS Multicast.
ASM Any-Source Multicast.
BEAM Bidirectional Aggregated Multicast.
BGP Border Gateway Protocol.
BGMP Border Gateway Multicast Protocol.
BSR Bootstrap Router.
CBT Core Based Trees Multicast.
CTH Connection Tree heuristic.
DBM Dense Branching Mode.
DF Don't Fragment.
DH Direct Heuristic.
DHCP Dynamic Host Configuration Protocol.
DMTA Distributed Multicast Tree Aggregation.
DVMRP Distance-Vector Multicast Routing Protocol.
ECMP EXPRESS Count Message Protocol.
ERM Explicit Route Multicast.
EXPRESS Explicitely Requested Single-Source multicast.
FTP File Transfer Protocol.
Gbps Gigabits par seconde.
Géant Gigabit European Academic Network.
GLOP Global Allocation Protocol.

GMPLS	Generalized MPLS.
GXcast	Generalized Xcast.
HBH (1)	Hop-By-Hop multicast routing protocol.
HBH (2)	Hop-By-Hop option.
HTTP	Hypertext Transfer Protocol.
IANA	Internet Assigned Numbers Authority.
ICANN	Internet Corporation for Assigned Names and Numbers.
IETF	Internet Engineering Task Force.
IGMP	Internet Group Management Protocol.
IH	Iterative Heuristic.
IP	Internet Protocol.
ISO	International Standards Organization.
Kbps	Kilobits par seconde.
KMB	KOU, MARKOWSKY et BERMAN.
LLC	Logical Link Control.
LSA	Link State Advertisement.
M2X	Multicast to Xcast.
MAAS	Multicast Address Allocation Server.
MAC	Media Access Control.
MADCAP	Multicast Address Dynamic Client Allocation Protocol.
MADD	Multicast Add.
MALLOC	Multicast Address Allocation Architecture.
MASC	Multicast Address-Set Claim.
MBGP	Multiprotocol extensions to BGP.
Mbps	Megabits par seconde.
MCPF	Maximal Common Path First heuristic.
MCPFL	MCPF on Leaves heuristic.
MCPFM	MCPF on Members heuristic.
MDHCP	Multicast Dynamic Host Configuration Protocol.
MDT	Multiple Destination Trail heuristic.
MKH	Modified Kruskal heuristic.
MLD	Multicast Listener Discovery Protocol.
MO	Member-Only heuristic.
MOSPF	Multicast extensions to OSPF.
MPLS	Multi-Protocol Label Switching.
MSDP	Multicast Source Discovery Protocol.
MTU	Maximum Transfer Unit.
NS	Network Simulator.
O/E/O	Optical/Electronical/Optical.
OSI	Open Systems Interconnect.
OSPF	Open Shortest Path First.
PIM-DM	Protocol Independent Multicast - Dense Mode.
PIM-SM	Protocol Independent Multicast - Sparse Mode.
PIM-SSM	Protocol Independent Multicast - Source Specific Multicast.

PMTU	Path MTU.
POP	Post Office Protocol.
PP	Post-Processing heuristic.
Q-STA	QoS STA.
QoS	Quality of Service.
RAMA	Root Addressed Multicast Architecture.
Reunite	Recursive-Unicast approach to multicast.
RIP	Routing Information Protocol.
RP	Rendez-vous Point.
RRTA	Re-Route To Any heuristic.
RRTS	Re-Route To Source heuristic.
SBM	Sparse Branching Mode.
SDH	Synchronous Digital Hierarchy.
SEM	Simple Explicit Multicast.
SGM	Small Group Multicast.
SM	Simple Multicast.
SMTP	Simple Mail Transfer Protocol.
SOCM	Structure Optique de Coût Minimal.
SOLM	Structure Optique de Longueur Minimale.
SONET	Synchronous Optical Network.
SPH	Shortest Path Heuristic.
SPT	Shortest Path Tree.
SSM	Source-Specific Multicast.
STA	Scalable Tree Aggregation.
STEA	Spanning Tree Enumeration Algorithm.
Tbps	Terabits par seconde.
TBXcast	Tree Based Xcast.
TEA	Topology Enumeration Algorithm.
TM	TAKAHASHI et MATSUYAMA.
TMBP	TM with Branching Penalty heuristic.
TS	Topological Segmentation heuristic.
UDP	User Datagram Protocol.
WADM	Wavelength Add/Drop Multiplexer.
WXC	Wavelength Selective Crossconnect.
WDM	Wavelength Dense Multiplexing.
X2M	Xcast to Multicast.
X2U	Xcast to Unicast.
X3C	Exact Cover with 3-Sets.
Xcast	Explicit Multicast.
Xcast+	Extensions for Xcast.

Annexe B

Réseaux utilisés pour les simulations

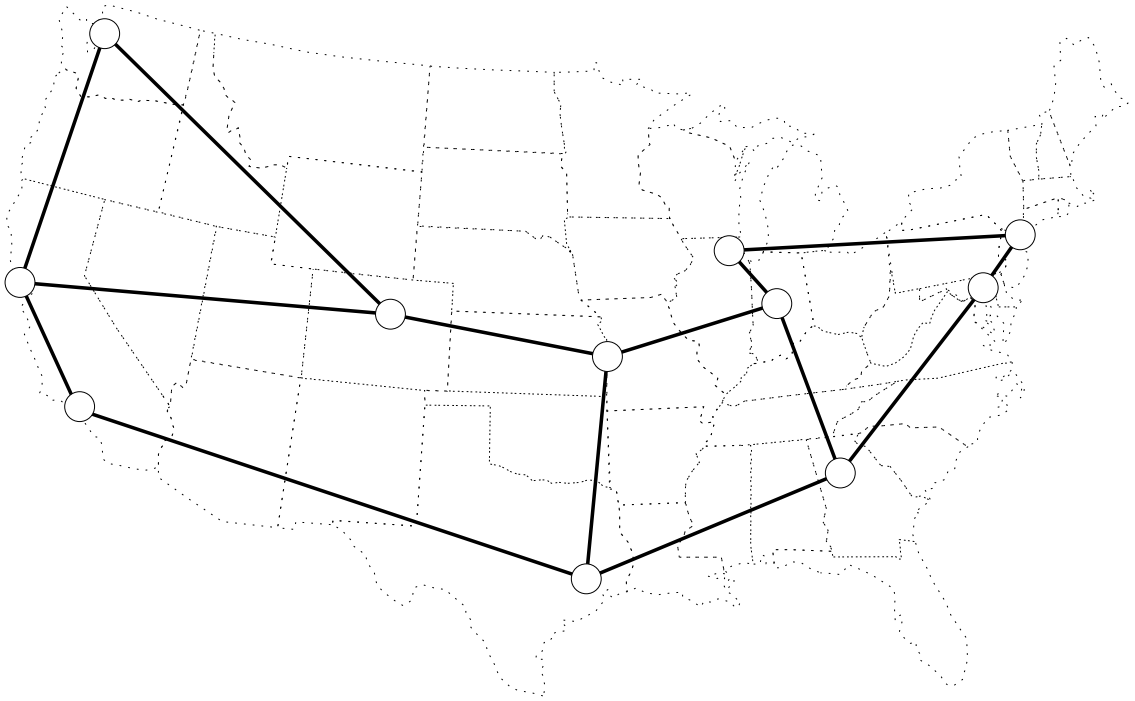


Figure B.1 – *Le réseau Abilène.*

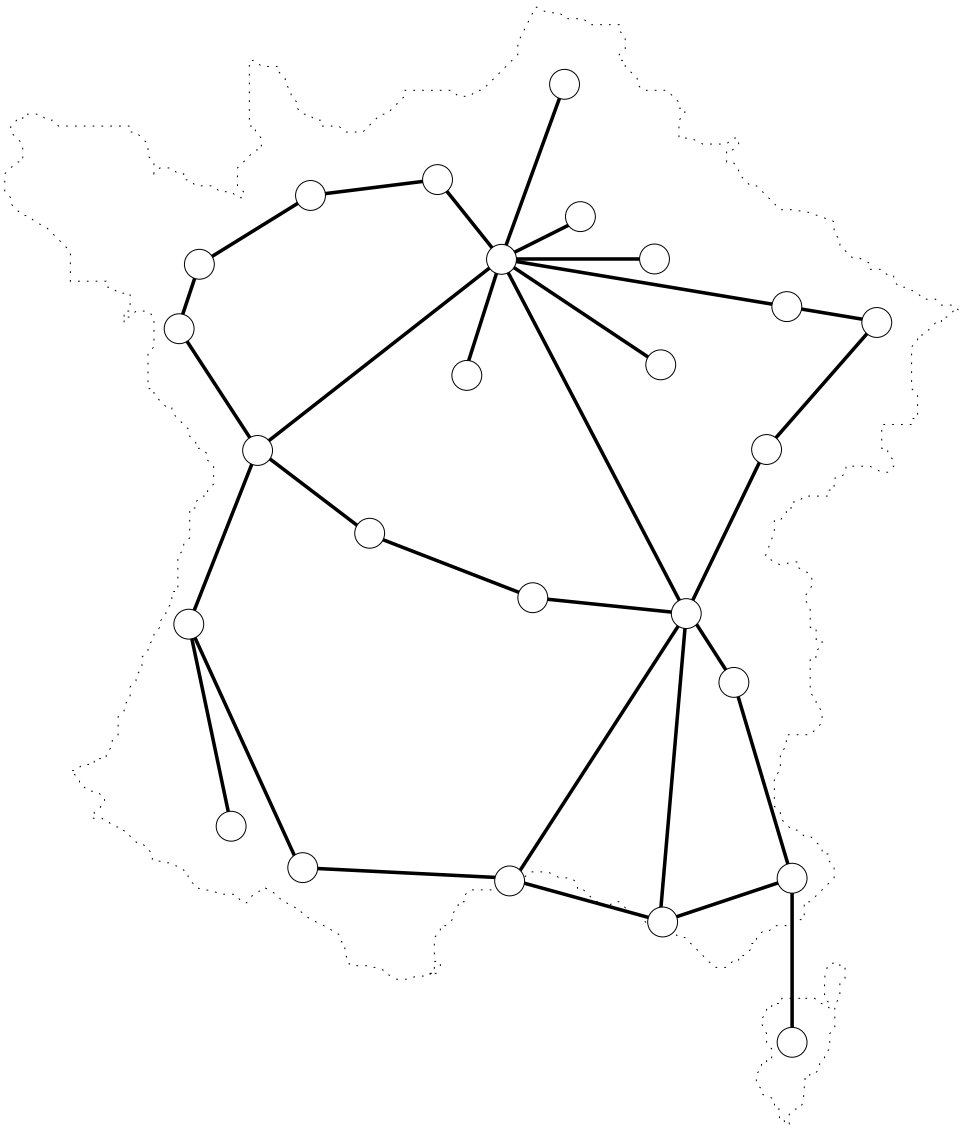


Figure B.2 – *Le réseau Renater.*

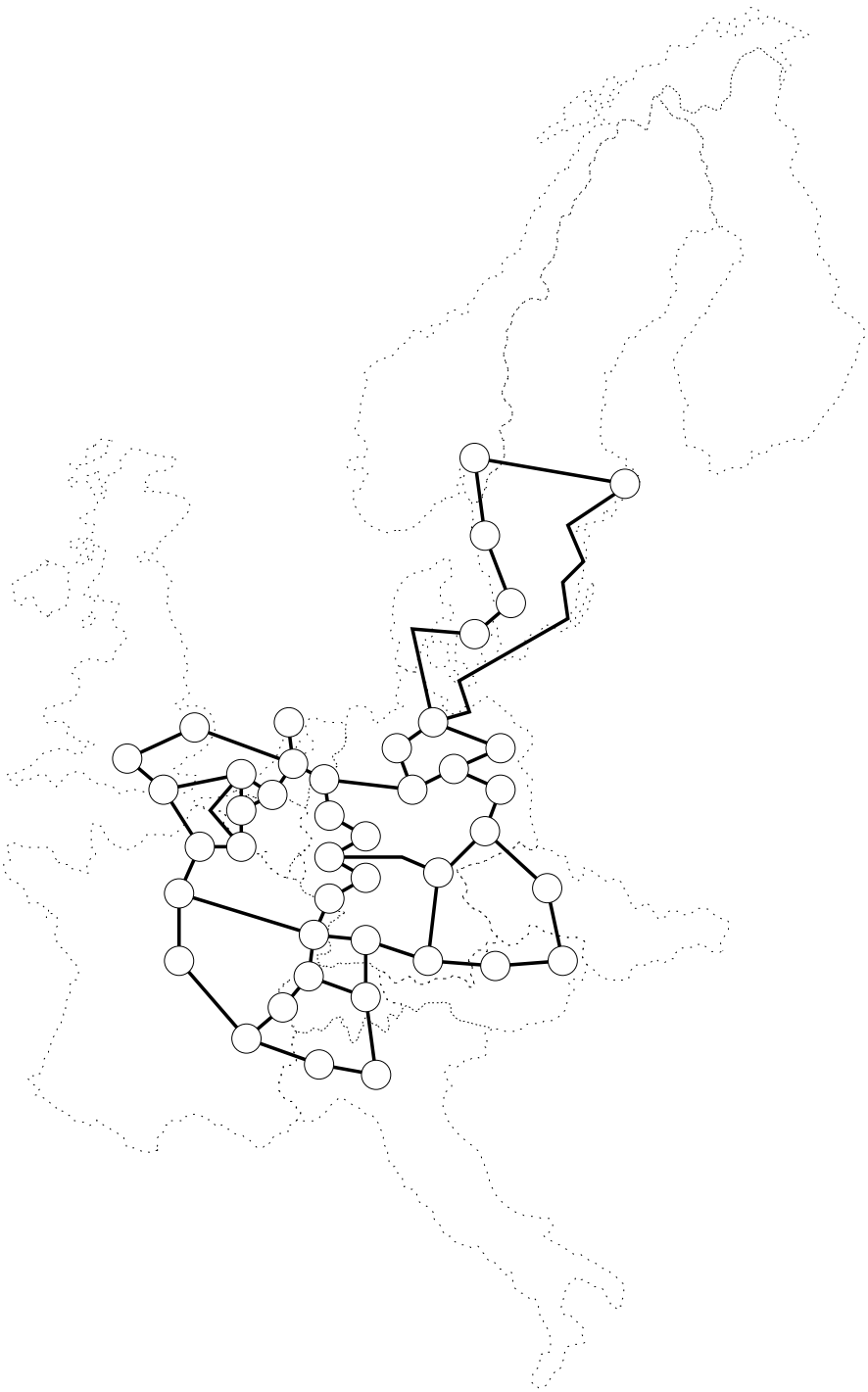


Figure B.3 – *Le réseau Eurorings.*

Annexe C

Script utilisé pour les simulations

pour *graphe* ∈ {abilene,renater} **faire**
  pour *scenario* de 1 à 250 **faire**
    pour p de 0 à 100 par pas de 10 **faire**
      lancer une simulation sur le graphe *graphe*, avec p % de nœuds pouvant dupliquer, et 1000 groupes concurrents
    fin pour
  fin pour
fin pour

Algorithme 9: Script des simulations des heuristiques du chapitre 2.

Résumé

La croissance rapide des réseaux de télécommunications nécessite une évolution importante des protocoles de routage. En effet, de nouveaux protocoles doivent être développés pour supporter les nouveaux types de réseaux et les nouveaux types de communications. Dans cette thèse, nous étudions un nouveau paradigme de communications : les communications de groupe dites « *multicast* ».

Dans un premier temps, nous considérons le cas particulier des réseaux tout optique. Ces réseaux fournissent un service de haute qualité mais possèdent de nombreuses spécificités, telles que l'incapacité de certains routeurs à dupliquer les signaux, qu'il est nécessaire de prendre en compte pour réaliser un routage efficace. Dans un second temps, nous proposons des mécanismes permettant le déploiement à grande échelle des communications *multicast*. Nous étudions les protocoles destinés exclusivement aux petits groupes ainsi que les protocoles agrégeant plusieurs communications entre elles.

Mots clés : réseau, protocoles, multicast.

Abstract

The fast growth of telecommunication networks requires an important evolution of routing protocols. Indeed, new protocols have to be developed to support new types of networks and new types of communications. In this thesis, we study a new paradigm of communications: group communications, also known as multicast communications.

Firstly, we consider the specific case of all optical networks. These networks provide high quality services, but their specificities, such as the incapacity to split signals, have to be taken into account in order to route efficiently. Secondly, we propose mechanisms that allow to deploy large scale multicast communications. We study protocols designed exclusively for small groups, and protocols that aggregate communications together.

Keywords : network, protocols, multicast.