ORSAY
N^o D'ORDRE : 7130

UNIVERSITÉ DE PARIS-SUD
U.F.R. SCIENTIFIQUE D'ORSAY

THÈSE

présentée
pour obtenir

Le GRADE de DOCTEUR en SCIENCES
DE L'UNIVERSITÉ PARIS XI ORSAY

Spécialité : INFORMATIQUE

par

José Ignacio Alvarez-Hamelin

Routage dans Internet : trafic autosimilaire, multicast et modèles de topologie

Soutenue le 20/12/2002 devant la Commission d'examen

M. Alberto E. Dams	Examinateur
M. Pierre Fraigniaud	Directeur
M. Michel Habib	Examinateur
M. Philippe Jacquet	Rapporteur
M. Jean-Jacques Pansiot	Rapporteur
M. Olivier Temam	Président

A todos los que investigan y enseñan
A los sabios antepasados ...
A las lenguas, que expresan ...

Remerciements

Je tiens tout d'abord à remercier au Programme Alfa-Cordial et les bourses Peruilh de la Facultad de Ingeniería de la Universidad de Buenos Aires qui ont financé cette thèse.
Je tiens à remercier au membres du jury :
- Philippe Jacquet et Jean-Jacques Pansiot pour accepter d'être rapporteurs de ma thèse dans si court délai.
- Pierre Fraigniaud qui a dirigé ma thèse. Je suis très reconnaissance à sa dévouement et travail au niveau scientifique. Je le remercie avoir supporté mon english-français du début de la thèse, et tout son appui dans la langue française. Je suis très reconnaissance à son aide dans la rédaction du manuscrit. J'apprécie aussi son amitié.
- Alberto Dams qui a été mon codirecteur en Argentine. Je suis très reconnaissance à son appui pour faire le doctorat, à ses conseils, et à tout son aide pour obtenir le financement. J'apprécie aussi son amitié.
- Je remercie aussi à Olivier Temam, le Président, et aux examinateurs Michel Habib et Alberto Dams.
Je tiens vivement à remercier Delia Kesner, la responsable du programme Alfa-Cordial pour l'université Paris-Sud, pour son aide et son appui dans la thèse.
Je suis très reconnaissance envers Sara Tressens pour les discutions scientifiques et pour la estimation des paramètres du trafic autosimilaire.
Claire Kenyon et Nicolas Schabanel pour les discutions scientifiques du modèle d'`Internet`.
Stéphane Boucheron pour les discutions scientifiques sur la variance infinie.
Je tiens à remercier Sophie Laplante pour sa grande aide dans l'écriture en français, les innombrables corrections et sa bonne humeur.
Moaiz Ben-Dhaou pour la dernière correction du franąis du manuscrit. Charles Delorme et Odile Favaron pour les corrections du français.
Jean-Paul Allouche pour son aide avec le français et pour certaines discutions scientifiques.
Mario Valencia pour sa grande amitié et son aide à mon arrivé en France.
Tous mes copains de bureau, Wadie Benajam, Taoufik Faik, Alexis Troubnikoff et Moaiz Ben-Dhaou.
C'est avec grand plaisir que j'associe à ces remerciements l'équipe GraphComm qui m'ont accueilli depuis le début. C'est impossible d'oublier les goûtés de vendredi organisés par Maryvonne et Jean-François.
Je tiens à remercier aussi à l'équipe Staff pour son support.
Sebastian Guarino, Juan Pablo Hurcade et Gaston Pazo pour ses scripts.
Mes remerciements à toute la "Bande du Petit Pavillon", Eduardo, Jacinta, Elina, Gustavo, Tatiana, Gabriel, Guille, Guillina, Pablo et autant d'autres qui m'ont accompagné depuis le début.
Je tiens à exprimer ma plus vive gratitude à toute ma famille : mes parents José Enrique y Amalia, mes sœurs María Fernanda y María José, mes belles-tantes Ana y Milagros, à mon beau-frère Roberto, et bien évidemment aux mes nièces et mes neveux Iliana, Nadia, Ian et Nicolás.
Je remercie très spécialement à mon amour Ana qui m'accompagne dans le plus grand défi qui est la vie.
Finalement, je tiens à remercier à tout le Coro de la Facultad de Ingeniería, spécialement à Claudia y Ricardo; et aussi à Norma Basso, à Ariel Alonso, aux Petits Chanteurs de St Louis, et à tout "Mélanges".

Table des matières

Liste des tableaux

Table des figures

.

Chapitre 1

Introduction

Le développement d'`Internet` permet aujourd'hui l'échange d'information à un rythme jamais atteint jusqu'alors, et sur une échelle quasiment mondiale. Ce grand réseau public est administré de manière décentralisée. Ses principaux éléments constitutifs sont d'une part le matériel, et d'autre part les protocoles de transmission et de routage. La gamme de matériel utilisé dans `Internet` est très vaste. Cela va des media de transmission (câbles ou fibre optique) jusqu'aux ordinateurs et serveurs, en passant par les équipements de communications (commutateurs et routeurs). Les protocoles de transmission et de routage s'occupent d'assurer le transfert de l'information sans erreur des sources aux destinataires. Cette thèse étudie certains aspects d'`Internet` en relation avec les protocoles de routage.

Le premier objectif d'un protocole de routage est de rendre possibles les communications. Les premiers protocoles de routage pour `Internet` n'ont été dédiés qu'aux communications point-à-point. L'évolution d'`Internet` engendre cependant de nouveaux besoins. Ainsi, l'apparition d'applications temps réel (la transmission de la voix par exemple) a rendu nécessaire l'amélioration des protocoles, c'est-à-dire principalement permettre d'assurer une qualité de service suffisante. Or, offrir une bonne qualité de service devient de plus en plus difficile à assurer du fait de la croissance permanente de la taille d'`Internet`. D'autre part, l'apparition d'applications nécessitant un support pour les communications de groupe a nécessité de revoir le concept même de communication point-à-point, pourtant central dans `Internet`. Assurer une bonne qualité de service (QoS) pour une application multipoint est un des défis d'`Internet`, qu'il convient de résoudre à relativement court terme.

Il est maintenant couramment admis que la réalisation de protocoles de routage plus efficaces nécessite une bonne connaissance de la topologie du réseau dans lequel ils opèrent. Les propriétés de la topologie sous-jacente peuvent en effet être capitales pour la mise en œuvre d'un protocole de routage. Par exemple les protocoles qui se basent sur l'inondation du réseau pour leur mise à jour ne peuvent que difficilement être très performants dans un réseau ayant des cycles, car certains paquets peuvent être plusieurs fois dupliqués. La nécessité d'appréhender la topologie du réseau afin de concevoir de meilleurs protocoles nous a poussé à étudier les modèles de topologie pour `Internet`. Il existe de nombreux modèles d'`Internet` proposés dans la littérature, comme par exemple le modèle de Waxman [Wax88], un peu simpliste, mais prenant toutefois en compte la notion physique de distance géographique.

D'autres modèles plus sophistiqués cherchent à coller aux principales propriétés observées dans `Internet`, dont en particulier les fameuses lois de puissance mises en évidence par M. Faloutsos, P. Faloutsos et C. Faloutsos [FFF99]. Une des premières contributions de cette thèse consiste en l'étude des modèles pour `Internet`, et en la proposition d'un nouveau modèle.

Outre la topologie, il est également important de prendre en compte la relation entre le trafic et les applications. L'objectif consiste à trouver des protocoles de routage qui profitent des capacités du réseau en utilisant non seulement le volume mais aussi la nature du trafic. De nombreux travaux ont ainsi montré le caractère autosimilaire du trafic sur `Internet`, que celui-ci soit engendré par les applications (par exemple le web) ou par les systèmes de contrôle dans le réseau (par exemple TCP). Les propriétés d'autosimilarité indiquent une corrélation à long terme du trafic. Il devrait être possible de profiter de cette caractéristique pour améliorer les protocoles de routage. Plus spécifiquement, un nombre élevé de paquets perdus réduit la performance des communications. Or `Internet` utilise la pile de protocoles TCP/IP dont le niveau IP n'assure pas que tous les paquets envoyés arrivent à leur destination (la méthode de contrôle de congestion pour les routeurs consiste à détruire un paquet quand il ne peut être stocké dans un tampon). Dans cette thèse nous proposons un nouveau paramètre de qualité de service, basé sur le caractère autosimilaire du trafic, pouvant servir à limiter les pertes de paquets.

Les applications multipoint ont particulièrement attiré notre attention car c'est un exemple de nouvelles applications où les ressources du réseau subissent le plus de contraintes. Les applications multipoint ont normalement besoin d'une qualité de service minimale raisonnable pour s'exécuter. En principe on peut distinguer deux approches pour assurer cette QoS. La qualité de service peut être fournie grâce à l'accroissement des ressources du réseau. De façon complémentaire, la qualité de service peut être fournie grâce à une meilleure utilisation du réseau. C'est dans ce second cadre que nous nous plaçons. Cette thèse propose deux contributions à l'étude des applications multipoints : d'une part un protocole d'ordonnancement de l'envoi des message dans un arbre multicast (pour améliorer la latence des protocoles existants), et d'autre part un nouveau protocole de construction d'arbre multicast, permettant de trouver les routes optimisant une certaine QoS.

Organisation de la thèse

Le chapitre 2 est consacré à l'étude de la topologie d'`Internet`. Il commence par une description de l'organisation logique d'`Internet`, et se poursuit par un bref état de l'art des différents modèles d'`Internet` proposés dans la littérature. Dans la troisième partie du chapitre, nous présentons un nouveau modèle de la topologie d'`Internet`. Ce chapitre a pour but de mettre en évidence les aspects les plus importants de la topologie d'`Internet`. Notre nouveau modèle nous servira d'outil pour vérifier des comportements de protocoles de routage unicast et multicast discutés plus loin dans le document. Une des qualités principales de notre modèle est d'engendrer des topologies très semblables à celles d'`Internet`, et ce même pour des réseaux de petite taille.

Le chapitre 3 est consacré à l'étude des protocoles multipoint. Nous nous intéressons dans

un premier temps aux techniques de construction d'arbres multipoint, au travers d'un bref état de l'art. Nous proposons ensuite deux protocoles. Le premier a pour objet l'optimisation de l'utilisation d'un arbre de multicast donné. Son utilisation ne dépend pas de la méthode de construction de l'arbre. Ce protocole est appelé MSDA (pour "Multicast Scheduling Distributed Algorithm"). MSDA s'occupe de l'ordonnancement des envois des messages dans un arbre. Étant donné un arbre de multicast quelconque, nous montrons que MSDA ordonnance les messages optimalement, c'est-à-dire permettant un délai de transmission source-destinations minimum. Nous avons réalisé des expérimentations sur des topologies obtenues avec notre générateur de topologie du chapitre 2 afin de mesurer le rapport entre le délai moyen (celui que subira en moyenne un paquet diffusé dans l'arbre) et le délai minimum (celui que subira le même paquet lorsque les envois des messages sont ordonnés selon MSDA). Le délai moyen est calculé comme la moyenne de tous les délais obtenus suivant tous les ordonnancements possibles. Ces expérimentations nous ont permis d'identifier des cadres d'applications possibles pour le protocole MSDA. Toujours dans le chapitre 3, nous proposons un nouveau protocole de construction d'arbre de multicast. Ce protocole s'appelle MCT (pour "Minimum Congestion Tree"). Il permet de construire des arbres optimaux selon tout paramètre de qualité de service donné. Pour toute paire source-destination (s, u), MCT explore tous les chemins de longueur au plus $\rho \cdot d(s, u) + r$ entre s et u où $d(s, u)$ dénote la distance (en nombre de sauts) entre s et u. Nous avons effectué des simulations sur la topologie du réseau ARPANET (1995) en comparant MCT à d'autres protocoles multipoint classiques. Ces expériences montrent que des améliorations significatives sont obtenues grâce à MCT, même pour des petites valeurs de ρ et r, c'est-à-dire au prix d'un faible accroissement du trafic de contrôle.

Le chapitre 4 est consacré à la présentation d'un modèle autosimilaire du trafic d'`Internet`, et d'un nouveau paramètre de qualité de service : la *congestion probabiliste*. La congestion probabiliste permet en particulier d'approximer efficacement le taux de pertes. Elle est adaptée à une description à long terme du trafic, et peut être utilisée dans des protocoles de routage point-à-point ou multipoint. La valeur de la congestion probabiliste au temps t est une estimation du trafic pour le t_u temps suivant, permettant une meilleure gestion du trafic. Nous avons montré comment mettre en œuvre cette gestion du trafic. Nous avons effectué des simulations pour valider notre approche. Ces simulations ont été menées sur la topologie du réseau UUNET et sur une topologie obtenue à partir de notre modèle d'`Internet`. Ces simulations démontrent l'utilité de la congestion probabiliste pour les protocoles multipoint. Spécifiquement, nous avons fait une campagne de simulations en utilisant le protocole multipoint QoSMIC développé par Faloutsos, Banerjea et Pankaj [FBP98]. L'utilisation de la congestion probabiliste permet d'améliorer significativement QoSMIC.

Enfin, le chapitre 5 conclut le document, et présente quelques directions de recherche à court et moyen termes.

Chapitre 2

La topologie d'Internet

Comme cela a été signalé maintes fois (cf [BE90, ZGLA91, WE94]), la construction de protocoles de routage efficaces doit reposer sur la prise en compte du "champs d'opération" de ces protocoles, c'est-à-dire du réseau, et en particulier de la topologie du réseau. Ce dernier point est d'ailleurs valable aussi bien pour les protocoles de routage point-à-point (unicast) que multipoint (multicast). Il est possible aujourd'hui d'étudier la topologie du réseau Internet à différentes échelles, et chaque échelle a son importance selon le protocole de routage considéré. Dans ce chapitre nous allons discuter de ces différentes échelles, et des modèles correspondants. Ce chapitre est entièrement consacré à la topologie d'Internet, et à sa modélisation. Dans un premier temps, nous présenterons une description des principales caractéristiques d'Internet tel qu'il est aujourd'hui. Dans la section 2.1.2, nous détaillerons les méthodes permettant l'obtention de données réelles relatives à Internet. Nous discuterons ensuite certaines propriétés du graphe d'Internet, notamment les lois de puissance, et nous effectuerons un rapide état de l'art des générateurs de topologies pour Internet. Finalement nous décrirons la contribution principale de ce chapitre : un nouveau modèle pour Internet.

2.1 La topologie physique d'Internet

Cette section décrit le développement d'Internet et son évolution. Nous rappelons tout d'abord les concepts de base relatifs à ce réseau, et nous mettrons en évidence quelques propriétés élémentaires satisfaites par sa topologie. Ceci nous permettra de mieux comprendre les modèles proposés dans le chapitre suivant, et de juger de la pertinence de notre nouveau modèle proposé au chapitre 2.3.

2.1.1 Quelques rappels

Internet est né dans les années 70, à la suite d'un projet de la DARPA (l'agence de défense des États Unis). C'est désormais un réseau public dont l'administration est distribuée. C'est

principalement le développement du protocole TCP/IP qui a permis l'existence d'`Internet`. Ce protocole a pour charge, au niveau IP, de connecter des réseaux LANs ("Local Area Networks"). Il offre au niveau transport deux types de communications : avec connexion (TCP) ou sans connexion (UDP).

On peut voir `Internet` comme ayant deux niveaux : le niveau LAN d'une part, et le niveau WAN ("Wide Area Networks") qui relie tous les LANs d'autre part. Le protocole IP prévoit un adressage pour chaque dispositif connecté à `Internet`, composé de deux parties : une dédiée au numéro du réseau LAN, et l'autre au numéro de l'hôte dans le réseau LAN. De cette manière, il est possible de faire facilement la distinction entre les messages qui sont pour le réseau LAN local et ceux qui devront emprunter le réseau WAN pour atteindre un autre LAN.

Dès son début, la taille d'`Internet` a crû de façon approximativement exponentielle. L'impact de cette croissance sur sa gestion est considérable. Ainsi, les premières tables de routage ont été faites manuellement, mais la croissance d'`Internet` a rapidement imposé un système automatique. RIP [Hed88] ("Routing Information Protocol") est un protocole dont le rôle est de maintenir les tables de routage en utilisant l'algorithme de Bellman-Ford [FF62] jusqu'à une distance de 15 sauts (le graphe qui représente le réseau a donc un diamètre maximum de 15). Ce nombre de sauts est motivé par le temps de convergence de l'algorithme, et par quelques problèmes d'instabilité dûs à certains changements pendant la phase de construction des tables [Hed88]. OSPF [Moy97b] ("Open Shortest Path First") est un autre type de protocole de routage. Il utilise l'information de toute la topologie du réseau pour construire les tables, en se basant sur l'algorithme de Dijkstra. Pour obtenir l'information de la topologie, le protocole OSPF procède par inondation.

La limitation du nombre de sauts défini dans RIP implique une limitation dans la taille du réseau pour lequel RIP peut être mis en œuvre. OSPF est également limité par sa méthode d'inondation lors de la construction des tables de routage. Ces limitations ont impliqué la mise en œuvre indispensable d'un autre niveau hiérarchique dans le routage : les *systèmes autonomes*. Chaque système autonome est un domaine propre, possédant une administration centralisée, en charge et responsable de son fonctionnement. On peut toutefois faire une distinction entre différents types de systèmes autonomes qui se distinguent par leurs fonctions dans `Internet`(voir figure 2.1). Il existe ainsi des systèmes autonomes qui englobent une entité, par exemple un campus universitaire, dont chaque membre peut souhaiter accéder au réseau global. Il existe également des systèmes autonomes qui servent au réseau en interconnectant plusieurs sous-systèmes autonomes, par exemple un fournisseur d'`Internet` pour les grands clients. Un système autonome du premier type est appelé "Stub-Domain" parce qu'il forme une feuille dans le graphe qui représente les connexions entre systèmes autonomes. Un système autonome du deuxième type est appelé système autonome d'interconnexion ou "Transit-Domain" à cause de sa fonction de relais. Bien sûr, comme toute classification, elle peut être mise en défaut. Il y a ainsi des systèmes autonomes du premier type qui peuvent aussi fournir une fonction d'interconnexion, et qui sont donc également appelés d'interconnexion. Il y a également des systèmes autonomes du premier type qui se connectent à plusieurs systèmes autonomes d'interconnexion sans router les messages des autres systèmes autonomes. Ils s'appellent systèmes autonomes multi-connectés ou "Multihomed-Domains".

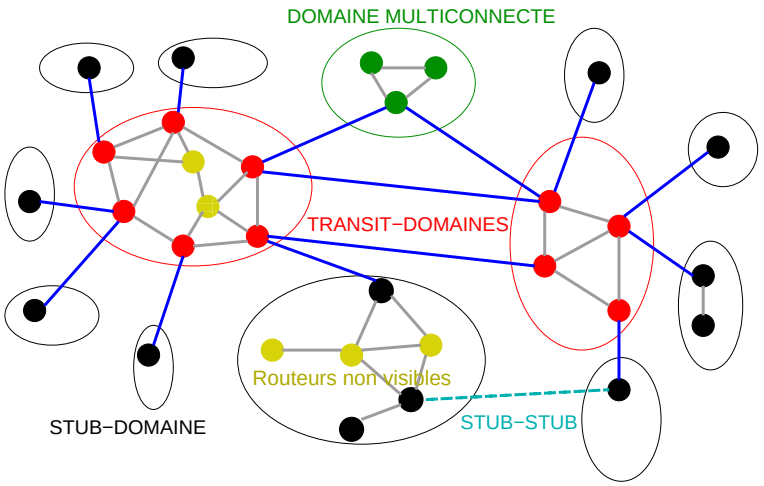

FIG. 2.1: *Systèmes autonomes dans* `Internet`

Un système autonome est composé d'un réseau WAN, et éventuellement de plusieurs LANs. Il est possible qu'il y ait des routeurs et des adresses dans le système autonome que ne soient pas visibles de l'extérieur de ce système autonome. Il est même possible qu'il y ait des adresses qui ne soient visibles que par certaines autres adresses. L'unique manière de connaître la topologie entière d'un système autonome est donc d'en faire partie. L'ensemble des systèmes autonomes forme `Internet`. Le protocole de routage entre les systèmes autonomes est BGP ("Border Gateway Protocol") [Rek95]. Ce protocole permet de faire le routage en respectant des contrats entre systèmes autonomes, relatifs par exemple au trafic que tel ou tel système autonome acceptera de faire transiter, ou aux routes que devront emprunter les paquets. En effet chaque routeur habilité d'un système autonome reçoit les tables de routage des autres systèmes autonomes, d'une manière telle qu'il pourra calculer ses tables de routage en cherchant le plus court chemin vers chaque système autonome (en utilisant une version de l'algorithme de Bellman-Ford [FF62]). Comme les routeurs ne gardent qu'un plus court chemin vers chaque destination, si une exploration de la topologie se base uniquement sur le routage selon les tables BGP, elle manquera peut être des liens, et parfois même des routeurs.

2.1.2 Comment obtenir la topologie d'`Internet` ?

L'exploration de la topologie d'`Internet` est un problème capital parce que seule une connaissance suffisamment précise de cette topologie permet de juger de la pertinence des modèles proposés. Nous décrivons rapidement dans cette section quelques méthodes qu'il est possible d'utiliser pour explorer la topologie d'`Internet`. Remarquons tout d'abord qu'il y a principalement deux graphes associés à `Internet` : le graphe des systèmes autonomes, et le graphe des routeurs. Ce second graphe est très difficile à obtenir à cause de sa taille, tandis que le graphe des systèmes autonomes est plus facile à obtenir, par l'intermédiaire des tables de routage BGP.

On peut trouver dans l'article de M. Faloutsos, P. Faloutsos et C. Faloutsos [FFF99] un

exemple du graphe des systèmes autonomes. La critique que l'on peut faire à l'étude de ce graphe est qu'il ne représente qu'un sous-graphe du réseau physique, et qu'il ne montre que les contrats entre les différents systèmes autonomes (du fait de la méthode utilisée pour obtenir l'information de la topologie à partir des tables de routage de BGP). Cette dernière critique est justifiée par le fait qu'il y a des liens qui existent mais qui ne sont pas visibles dans le graphe à cause des contrats et de BGP. De plus, il y a parfois des liens qui ne sont utilisés que pour connecter un petit groupe des systèmes autonomes, et ils ne sont pas utilisés si le trafic vient d'un système autonome qui n'appartient pas à ce groupe. Il est difficile de prendre connaissance de ces liens de l'extérieur.

Dans le cas du graphe des routeurs, la méthode la plus utilisée est fournie par le logiciel `traceroute` de Van Jacobson sous UNIX. Ce logiciel envoie des paquets IP vers une destination spécifiée. Tous les paquets IP ont un champ TTL ("Time To Live") qui sert à éliminer les paquets qui n'ont pas trouvé de route après un certain temps. Ce champ est décrémenté chaque fois que le paquet passe par un routeur, et le paquet est détruit quand il est à zéro. La destruction s'accompagne toutefois d'une notification à la source du paquet. Cette notification est envoyée par le protocole ICMP ("Internet Control Message Protocol") d'`echo-request` qui fonctionne sur la couche IP et qui fournit l'adresse du nœud où TTL$= 0$. Si une source envoie des paquets IP avec des champs TTL fixés à $1, 2, \ldots,$ jusqu'à n vers une destination v, on peut connaître les n nœuds qui appartiennent au chemin de la source à v. Notons que `traceroute` envoie trois paquets IP pour chaque valeur du champ TTL, jusqu'à la destination. Il y a donc trois possibilités de recevoir une réponse pour chaque nœud qui appartient à la route. Pour bien profiter de l'information recueillie par `traceroute` il faut connaître des adresses IP valides, c'est-à-dire des adresses actives dans le réseau. D'autre part, `traceroute` ne permet de suivre que les chemins codés dans les tables de routage, ce qui peut empêcher de suivre certains chemins qui ne sont pas les plus courts. Le principal avantage de cette méthode est toutefois sa simplicité et sa robustesse. (Elle est robuste parce qu'elle se sert du protocole ICMP qui est un standard pour `Internet`.)

Pansiot et Grad [PG98] ont étudié le graphe de routeurs, et ont construit un graphe obtenu en effectuant une exploration à partir de 11 sources dispersées dans `Internet`. La méthode d'exploration est principalement celle de `traceroute`, avec certaines optimisations, comme par exemple initialiser TTL à une distance telle qu'en deça le graphe est déjà exploré. Une autre optimisation utilisée consiste à diminuer le nombre de paquets envoyés à un nœud : `traceroute` envoie trois paquets tandis que la version proposée par les auteurs n'envoie un paquet que s'il n'y a pas de réponse. Il y a deux avantages de cette modification. Le premièr est qu'elle ne charge pas les routeurs (elle envoie *a priori* un seul paquet). Le deuxième est qu'elle assure de toujours recevoir une réponse des routeurs qui sont dans un chemin où le nombre de paquets perdus est élevé. Par contre, Pansiot et Grad n'ont considéré que les adresses IP ayant leur nom résolu par le DNS ("Domain Name Server"). Le DNS s'occupe de faire la liaison entre noms et numéros IP. Donc, avec cette technique, la résolution d'alias est basée sur l'information dont le DNS a connaissance. Or l'information que possède le DNS peut être incomplète et ne pas permettre de résoudre tous les alias. La taille du graphe obtenu est de 3.888 nœuds. Pansiot et Grad ont utilisé cette exploration pour étudier les caractéristiques des arbres obtenus dans les applications multipoint. Le graphe obtenu

dans [PG98] a été utilisé par Faloutsos *et al.* [FFF99] pour vérifier les lois de puissance que ces derniers ont proposé.

Burch et Cheswick [BC99] ont mené des explorations avec `traceroute` à partir d'une seule machine source. Pour sélectionner les destinations, ils se sont servis du DNS et des tables de routage BGP. Ils ne résolvent pas les problèmes d'alias. Or un même routeur peut posséder plusieurs numéros IP. Ainsi, le graphe obtenu peut avoir plus de nœuds qu'il n'y en a dans le réseau original. Burch et Cheswick ont obtenu un graphe de 88.000 nœuds dans un délai de 13 jours.

Govindan et Tangmunarunkit [GT00] ont examiné la possibilité de récupérer le graphe d'`Internet` à partir d'une seule machine connectée à n'importe quelle partie d'`Internet`, et sans utiliser aucune base de données préexistante (comme DNS ou les tables BGP). Ils profitent du fait qu'il existe presque 8% de routeurs qui sont munis de la capacité de définir une partie du chemin de routage à la source ("source routing"). Cette capacité permet d'envoyer un paquet IP d'un source à une destination en passant par des routeurs spécifiés dans l'en-tête du paquet IP. Elle permet ainsi de suivre différents chemins en plus de ceux qui sont les plus courts. En profitant de cette capacité, il est possible de découvrir plusieurs liens qui peuvent rester cachés si on suit seulement les plus courts chemins. De plus, l'utilisation de la spécification d'un partie du chemin permet de connaître tous les ports (ou interfaces) des routeurs, et ainsi de résoudre les problèmes d'alias[1]. Govindan et Tangmunarunkit ont montré qu'avec 5% de routeurs ayant la capacité "source routing" dans un graphe de 1.000 nœuds de degré maximum 5, il est possible de découvrir 90% du graphe. Ce résultat est toutefois basé sur le modèle de Waxman [Wax88] (voir section 2.2.2). Une autre technique utilisée pour découvrir des chemins alternatifs consiste à envoyer des paquets d'essai (tels que ceux de `traceroute`) à des temps différents. Cette technique permet de découvrir des routes alternatives et des liens de sécurité parfois peu utilisés. Govindan et Tangmunarunkit ont développé le logiciel *Mercator* qui se sert d'une version modifiée de `traceroute` (couplée à d'autres techniques). Ce logiciel prend approximativement trois semaines pour relever une topologie d'`Internet`. Il n'obtient donc pas une *photo* de la topologie, mais une version de la topologie d'`Internet` moyennée pendent la période de trois semaines. Le graphe obtenu possède 200.000 liens.

La CAIDA ("Cooperative Association for Internet Data Analysis") [wwwb] a mené un projet pour obtenir un plan d'`Internet`. Broido et Claffy [BkC01] ont donné les résultats de l'analyse des données de la topologie d'`Internet` obtenues par CAIDA. L'obtention des données de la topologie s'est faite avec une vingtaine de serveurs répartis dans `Internet`, et qui envoient des paquets IP utilisant la technique de `traceroute`. Il en résulte un graphe de 655.000 nœuds. Broido et Claffy ont représenté des graphes orientés parce que la méthode de `traceroute` ne donne que les adresses dans un sens. À partir de graphes orientés, Broido et Claffy ont fait des analyses de connectivité, c'est-à-dire cherché l'ensemble des nœuds dans une composante fortement connexe. Une des caractéristiques de ce travail est la mise en évidence d'une composante connexe géante, qui est normalement appelée le *noyau* de réseau. La principale critique que l'on peut faire à l'approche de Broido et Claffy est qu'ils partent

1. Les numéros IP sont assignés selon les sous-réseaux qu'un routeur interconnecte. Il n'y a donc *a priori* aucune relation entre les numéros IP d'un même routeur, ce qui pose des problèmes pour résoudre les alias.

d'un nombre de sources trop petit pour espérer obtenir le graphe entier. Ils ne se basent en effet que sur les chemins suivis par les routeurs pour relever la topologie. Les résultats peuvent donc être biaisés. C'est cependant la "carte" d'`Internet` la plus grande obtenue jusqu'à présent.

Remarque Drineas, Frieze, Kannan, Vempala et Vinay [DFK+99] ont cherché une "échelle" appropriée pour l'étude des réseaux. Les auteurs proposent ainsi une carte moins complète que le graphe physique des routeurs, mais plus complète que le graphe des systèmes autonomes. La construction du graphe des routeurs est obtenue en regroupant les adresses qui appartiennent au même sous-réseau. Cela permet d'enlever des parties de réseau qui appartiennent à un même réseau physique (normalement un même lieu physique) mais qui n'apportent aucune information sur la topologie d'un point de vue global. Les réseaux numérotés avec un même préfixe IP[2] appartiennent en effet à la même entité administrative. On peut donc les regrouper dans un seule nœud.

Cette technique de réduction a pour principal objet de réduire le graphe obtenu avec des techniques basées sur `traceroute`. Le problème qu'elle présente est qu'il n'est pas possible de connaître *a priori* la division en sous-réseaux du fait de l'utilisation de *masques* variables pour distinguer la partie de l'adresse IP correspondant au réseau de celle correspondant à l'hôte. Comme il a été vu dans la section 2.1, le *masque* permet de séparer la partie réseau de la partie hôte. Au début d'`Internet` le masque n'a été défini que pour la classe d'adresse IP. Cette classe est identifiée à partir des premiers bits de l'adresse. Elle a été utilisé par les routeurs pour construire les tables de routage. Ces tables contenaient le préfixe IP (c'est-à-dire le nom du réseau) et la sortie correspondante. Le nombre de sous-réseaux d'une classe peut cependant être très grand ou très petit. L'espace d'adressage d'IP n'était donc pas bien utilisé. L'incorporation des masques variables (voir [FLJY93]) a permis une meilleure distribution des adresses IP.

2.1.3 Les lois de puissance

Un premier travail qui mentionne la loi de puissance dans le cadre d'`Internet` peut se trouver dans l'article de Chuang et Sirbu [CS98, CS01]. Ils ont montré expérimentalement que le rapport entre le nombre de liens $L(m)$ utilisés par un arbre multicast d'un groupe de m membres (cf. chapitre 3) et la distance moyenne $\overline{d}$ entre toutes les paires de nœuds du réseau, suit la loi suivante :

$$\frac{L(m)}{\overline{d}} = m^{\beta}, \text{ pour } m \ll n,$$

où n est le nombre de sommets dans le réseau, et $\beta = 0.8$. Les expériences ont été faites sur des topologies réelles (ARPANET : cf. section 3.3.3, MBone : réseau conçu pour le test de protocoles multicast), et générées aléatoirement (telles que décrites dans la section 2.2). Cette loi révèle une relation entre les arbres multicast et la topologie sous-jacente.

2. Le préfixe est la partie de l'adresse IP correspondant au réseau.

Une première tentative pour expliquer ce comportement a été faite par Phillips, Shenker et Tangmunarunkit [PST99]. Afin de circonscrire l'étude, ils ont considéré les arbres v-aires complets. Dans ce cadre ils ont trouvé une fonction asymptotique qui n'est pas une loi de puissance, ce qui démontre l'impact de la topologie du réseau. Par contre, Adjih, Georgiadis, Jacquet et Szpankowski [AGJS01, AGJS02] ont montré que, dans un cadre similaire, la loi de puissance a une explication analytique. Ils ont considéré les arbres v-aires "autosimilaires" suivants : à partir d'un arbre v-aire complet, on remplace les liens entre les nœuds à profondeur $k-1$ et les nœuds à profondeur k par une chaîne de longueur aléatoire, tel que la longueur moyenne soit

$$\ell_k = v^{(D-k)\Theta}$$

où D est la profondeur de l'arbre et Θ est un facteur d'autosimilarité. Les auteurs ont montré que, dans ce cadre, les arbres multicast suivent la loi suivante

$$\frac{L(m)}{\bar{d}} \simeq m^{1-\Theta}, \text{ pour } m \ll n.$$

Ils ont également mené des expérimentations sur des topologies générées aléatoirement, sur ARPANET, et sur des topologies obtenues par Burch et Cheswick [BC99] (cf. section 2.1.2). Ces expérimentations montrent que les arbres multicast générés vérifient $1-\Theta = 0.88$ dans ces topologies. Le modèle de Adjih, Georgiadis, Jacquet et Szpankowski semble donc être un bon modèle de la structure d'un arbre multicast dans `Internet`.

Faloutsos, Faloutsos et Faloutsos [FFF99] ont évalué l'évolution du graphe d'`Internet` pendant une année, avec trois mesures faites en novembre 1997, avril 1998 et décembre 1998. Ils ont mis en évidence de nombreuses lois de puissance dans la topologie des systèmes autonomes (obtenus à partir des tables du protocole BGP) : les *lois de puissance*. Une première loi lie le degré sortant d_v d'un sommet v à son ordre r_v lorsque les sommets sont rangés par ordre décroissant selon leur degré :

$$d_v \simeq r_v^{\alpha}. \tag{2.1}$$

La constante α de l'équation 2.1 est mesurée à -0.74, obtenue avec une corrélation de 97% (décembre 1998).

Une autre loi caractérise le nombre f_d de nœuds ayant un degré sortant d :

$$f_d \simeq d^{\beta}. \tag{2.2}$$

La valeur de β a été mesurée à -2.20, avec une corrélation de 97% (décembre 1998).

Une troisième loi caractérise les valeurs propres de la matrice d'adjacence du graphe rangées par ordre décroissant :

$$\lambda_i \simeq i^{\gamma}. \tag{2.3}$$

La valeur de γ a été mesurée à -0.487, avec une corrélation de 99% (décembre 1998).

Finalement, une dernière loi caractérise le nombre x_d de paires de nœuds à distance d :

$$x_d \simeq d^{\delta} \tag{2.4}$$

pour d significativement plus petit que le diamètre du graphe. La valeur de δ a été mesurée à 4.86 avec une corrélation de 98% (décembre 1998).

Ces lois illustrent des particularités essentielles du graphe d'`Internet`. En particulier, il est possible d'obtenir certaines informations utiles. Citons par exemple, la définition suivante extraite de Faloutsos *et al.* [FFF99] : Soit un graphe de n nœuds, e arêtes et vérifiant l'égalité de l'équation 2.4. Ce graphe est dit de diamètre *effectif*

$$\mathcal{D}_{eff} = \left(\frac{n^2}{n + 2\,e} \right)^{1/\delta}.$$

L'utilité de cette définition est d'estimer avec grande probabilité le nombre de sauts qu'il y a entre deux nœuds quelconques du graphe. C'est intéressant car une application d'`Internet` peut avoir par exemple besoin de connaître la distance maximale pour faire une diffusion d'un message spécifique avec un grande probabilité.

Ces lois ont naturellement marqué la tendance des propositions de modèles d'`Internet`. Nous en verrons des illustrations plus loin. D'autre part, Broder, Kumar, Maghoul, Raghavan, Rajagopalan, Stata, Tomkins et Wiener [BKM+00] ont également trouvé des ressemblances entre le graphe d'`Internet` et celui de la toile WWW. La toile WWW est un graphe orienté, très dynamique, de plus de 250 millions de nœuds (pages). S'il est très difficile d'obtenir une représentation de ce graphe à cause de sa taille et de sa dynamicité, il est cependant possible d'en obtenir quelques caractéristiques. Broder *et al.* ont ainsi trouvé que le graphe de la toile suit des lois de puissance, avec un exposant approximativement égal à -2.7 pour la loi de puissance (2.2).

Remarque 1. Les lois de puissance exprimées dans les équations (2.1) et (2.2) sont très similaires, et on peut en fait vérifier leur équivalence. A cette fin, commençons par calculer d_v à partir de l'équation 2.2. En effet, d_v est mis en relation avec le rang et le rang est mis en relation avec le nombre de nœuds de degré $\geq d_v$.

Soit $c_d > 0$ tel que $f_d = c_d \cdot d^{\beta}$. Soit alors d_0 tel que

$$\sum_{d=d_0}^{d_{max}} f_d \leq r_v < \sum_{d=d_0-1}^{d_{max}} f_d$$

Il y a au moins r_v nœuds de degré $\geq d_v$. Donc $d_v = d_0$. Appelons $F(x)$ le nombre de sommets de degré $\geq x$. On a

$$F(x) = \sum_{d=x}^{d_{max}} f_d = \sum_{d=x}^{d_{max}} c_d \cdot d^{\beta} \simeq \int_x^{d_{max}} c_d \cdot d^{\beta}\, \partial d = \frac{c_d}{\beta+1} \left(d_{max}^{\beta+1} - x^{\beta+1} \right). \tag{2.5}$$

	Valeur de α	Valeur de β	Valeur calculée de α	Valeur calculée de β
`Internet` 11-97	**-0.81**	**-2.15**	-0.87	-2.23
`Internet` 04-98	**-0.82**	**-2.16**	-0.86	-2.22
`Internet` 12-98	**-0.74**	**-2.20**	-0.83	-2.35
Routeurs 95	**-0.48**	**-2.48**	-0.68	-3.08

TAB. 2.1: *Les valeurs calculées*

D'après les valeurs expérimentales de β observées dans l'article de Faloutsos *et al.* [FFF99], on peut affirmer que $\beta < -1$.

Donc, si $x \ll d_{max}$, alors $d_{max}^{\beta+1} \ll x^{\beta+1}$, et donc

$$F(x) \simeq -\frac{c_d}{\beta+1} \cdot x^{\beta+1} \tag{2.6}$$

Finalement, on obtient

$$\begin{aligned} -\frac{c_d}{\beta+1} \cdot d_0^{\beta+1} \leq & \quad r_v \quad < -\frac{c_d}{\beta+1} \cdot (d_0-1)^{\beta+1} \\ d_0^{\beta+1} \leq & \quad \frac{\beta+1}{-c_d} \cdot r_v \quad < (d_0-1)^{\beta+1} \\ d_0 \leq & \left(\frac{\beta+1}{-c_d}\right)^{\frac{1}{\beta+1}} \cdot r_v^{\frac{1}{\beta+1}} < d_0 - 1 \end{aligned} \tag{2.7}$$

Donc $d_v = d_0 \simeq \left(\frac{\beta+1}{-c_d}\right)^{\frac{1}{\beta+1}} \cdot r_v^{\frac{1}{\beta+1}}$, c'est-à-dire $d_v = c_r\, r_v^{\alpha}$ avec $\alpha = \frac{1}{\beta+1}$ et $c_r = \left(\frac{-\beta-1}{c_d}\right)^{\frac{1}{\beta+1}}$.

Réciproquement, supposons que $d_v = c_r\, r_v^{\alpha}$. Soit v un sommet de degré $d-1$, et v' un sommet de degré $d+1$. On a

$$r_{v'} < F(d) < r_v$$

donc

$$\left(\frac{1}{c_r}\right)^{1/\alpha} d_{v'}^{1/\alpha} < F(d) < \left(\frac{1}{c_r}\right)^{1/\alpha} d_v^{1/\alpha}$$

donc $F(d) \simeq \left(\frac{1}{c_r}\right)^{1/\alpha} d_v^{1/\alpha}$.

Or $f_d \simeq F'(d)$, et donc $f_d \simeq \frac{1}{\alpha}\left(\frac{1}{c_r}\right)^{1/\alpha} \cdot d^{1/\alpha-1}$.

Donc f_d suit une loi de puissance $c_d\, d^{\beta}$ avec $\beta = \frac{1}{\alpha} - 1$ et $c_d = \frac{1}{\alpha}\left(\frac{1}{c_r}\right)^{1/\alpha}$.

Remarque 2. On peut faire une deuxième remarque en relation avec l'estimation des exposants des lois de puissance. Si on prend la relation ci-dessus obtenue pour α et β (c'est-à-dire $\beta = \frac{1}{\alpha} - 1$) et les valeurs obtenues par Faloutsos *et al.* [FFF99], on s'aperçoit qu'il y a des

différences notables (voir table 2.1). Bien que dans notre calcul il y ait des approximations, les différences trouvées entre les α estimés et $\frac{1}{\beta+1}$ ne se justifient pas seulement par nos approximations. Une explication possible de ces différences est que les calculs pour obtenir les exposants sont faits dans l'espace log-log. Le coefficient de corrélation absolu utilisé par Faloutsos *et al.* donne donc une erreur considérablement plus grande quand on le regarde dans l'espace linéaire. De plus, il est possible de vérifier que l'erreur $err_{lineaire}$ dans l'espace linéaire dépend fortement des asymptotes de l'approximation linéaire. Par exemple, pour la deuxième loi de puissance (équation (2.2)), on pourrait ajouter ces asymptotes et écrire une équation plus générale de la manière suivante :

$$f_d - f_0 = c_d \cdot (d - d_0)^{\beta}. \tag{2.8}$$

On peut alors tenir compte du fait que l'asymptote sur les abscisses doit être $d_0 \leq d_{min}$, où d_{min} est le degré minimum du graphe. L'asymptote sur les ordonnées doit être $f_0 \leq n$, où n est le nombre des nœuds de degré maximum. Ces asymptotes peuvent être calculées en minimisant l'erreur $err_{lineaire}$ dans l'espace linéaire, et obtenir une valeur de l'exposant plus exacte.

2.2 État de l'art des modèles proposés pour `Internet`

Dans ce chapitre, nous établissons un état de l'art des principales méthodes proposées pour modéliser Internet. Notre étude n'est pas nécessairement exhaustive mais résume toutefois les principales propositions connues. Un réseau est modélisé par un graphe $G = (V, E)$, où V est l'ensemble des nœuds et E l'ensemble des arêtes. Selon l'échelle, les nœuds peuvent représenter des routeurs, des systèmes autonomes, où tout autre objet ayant un sens fonctionnel dans `Internet`. Les arêtes représentent toujours des liens de communication entre les nœuds. Nous listerons tout d'abord les modèles non hiérarchiques, introduits principalement pour le graphe des systèmes autonomes. Nous listerons ensuite des modèles hiérarchiques plus réalistes. Pour finir, nous discuterons des différents générateurs de graphes qui tiennent compte des lois de puissance présentées dans la section 2.1.3.

2.2.1 Le modèle $\mathcal{G}_{n,p}$

Le modèle $\mathcal{G}_{n,p}$ [ER59] est le premier modèle auquel on pense pour Internet. Il a pour attrait la simplicité de sa définition. Il génère un graphe de n nœuds et avec une probabilité p d'existence d'un lien entre deux nœuds quelconques du graphe. Les principaux problèmes de ce modèle sont malheureusement les suivants :

1. Les graphes générés sont pour la plupart non connexes dès que p est petit (ce qui est nécessaire pour obtenir des graphes de faible degré) ;
2. Bien qu'il soit possible d'obtenir un degré moyen donné, le modèle ne respecte pas la bonne distribution de degrés des sommets (voir section 2.1.3) ;

3. Il n'y a pas de relation entre les sommets et leur position "géographique".

Le premier point indique qu'il faudra générer beaucoup de graphes pour espérer en obtenir un qui soit connexe. Le deuxième point est rédhibitoire parce que les lois de puissance sont largement acceptées suite aux résultats de Faloutsos *et al.* [FFF99]. Le dernier point est également très important car les réseaux physiques sont principalement construits à partir de la distribution de la population. Ces défauts font que ce modèle n'est qu'une très vague approximation de la topologie d'`Internet`.

2.2.2 Le modèle de Waxman

Le modèle proposé par Waxman [Wax88] a été intensivement utilisé car c'est une assez bonne représentation des réseaux réels, au moins au niveau géographique. Ce modèle propose la génération d'un graphe aléatoire de n sommets distribués au hasard (loi uniforme) dans un plan. La probabilité p d'avoir une arête entre les sommets u et v est donnée par

$$p = \alpha\, e^{-d/\beta L} \tag{2.9}$$

où d est la distance Euclidienne entre u et v, L est la distance Euclidienne maximale entre deux nœuds quelconques, et α et β sont deux paramètres du modèle. Lorsque α augmente, le nombre d'arêtes dans le graphe augmente. Lorsque β augmente, le rapport entre le nombre d'arêtes longues et le nombre d'arêtes courtes augmente.

Ce modèle génère souvent des graphes non connexes, et il n'est pas possible de contrôler la distribution des degrés des sommets. En particulier, il ne vérifie pas nécessairement les lois de puissance [MMB00]. Il n'est de fait même pas possible de contrôler le nombre d'arêtes, et donc de contrôler le degré moyen, paramètre pourtant caractéristique d'une topologie. Zegura, Kenneth, Calvert, Michael et Donahoo [ZCD97] ont obtenu les courbes de niveau du nombre d'arêtes en fonction des paramètres α et β (les courbes de niveau α en fonction de β pour le même nombre d'arêtes sont du type hyperbolique).

Malgré ses défauts, le modèle de Waxman a été largement utilisé, certainement pour la simplicité de la représentation de la géographie, et la prise en compte des relations entre distance et probabilité de connexions.

2.2.3 Le modèle de localité

Le modèle de localité a été proposé par Zegura, Kenneth, Calvert, Michael et Donahoo [ZCD97]. Il génère un graphe aléatoire de n sommets placés au hasard (loi uniforme) dans le plan. La probabilité p d'avoir une arête entre les sommets u et v est donnée par

$$p = \begin{cases} \alpha & si\ d < r \\ \beta & si\ d \geq r \end{cases}$$

où d est la distance Euclidienne sur le plan entre u et v, et r est un paramètre du modèle qui définit le rayon de localité. Ce modèle tente de proposer une solution alternative à celui de Waxman pour les relations entre distance physique d'une part, et probabilité de connexions d'autre part. Cependant, comme le modèle est basé sur le modèle aléatoire avec distribution uniforme, il a les mêmes défauts que celui de Waxman, dont en particulier le fait de générer des graphes non connexes et une distribution des degrés non conforme au lois de puissance (voir section 2.1.3). Les auteurs de ce modèle ont cependant réussi à bien illustrer la localité, c'est-à-dire la préférence des nœuds à se connecter aux nœuds locaux plutôt qu'aux nœuds éloignés. Le principal intérêt de ce modèle est qu'il permet la réutilisation de certains résultats qui existent déjà pour les graphes aléatoires $\mathcal{G}_{n,p}$.

2.2.4 Les modèles de graphes réguliers

L'utilisation d'un modèle basé sur des structures régulières (anneaux, cliques, grilles) est nécessairement restreinte à des sous-parties de la topologie. Ces modèles s'utilisent en fait principalement pour modéliser certaines technologies de réseaux LAN. Par exemple, les réseaux FDDI sont des anneaux. Les réseaux Gigabit-Ethernet sont virtuellement des cliques. Selon le niveau logique considéré, les réseaux FDDI peuvent d'ailleurs également être modélisés par des cliques. En effet, si les réseaux connectés physiquement par des anneaux sont usuellement modélisés comme tels, il existe cependant des cas de réseaux pour lesquels le rapport entre la vitesse de leurs liens et la vitesse des liens d'autres réseaux justifie l'utilisation d'autres topologies. La technologie FDDI spécifie une vitesse de 100 mega-bits par seconde pour ses liens. Si un tel réseau est relié à `Internet` par un lien de 1 mega-octet, il est alors raisonnable de modéliser FDDI comme une clique dans laquelle tous les nœuds sont interconnectés entre eux.

Plusieurs technologies peuvent être modélisées par des anneaux, en particulier SONET, SDH, FDDI, Token-Ring, et parfois ATM.

Les technologies LAN avec médium partagé (comme Ethernet et les versions de Fast-Ethernet) sont généralement modélisées par des cliques. La question qui se pose alors est relative à la possibilité d'établir réellement des connexions entre tous les membres en même temps. La technologie Ethernet ne permet en fait de connecter que deux stations à la fois. Cette limitation se produit parce que le médium est partagé entre toutes les stations. Il existe cependant des commutateurs pour les réseaux LAN qui donnent le service LAN en pouvant supporter plusieurs communications à la fois. On peut alors raisonnablement modéliser le réseau LAN correspondant comme une clique.

Pour finir, notons le cas spécifique de la grille. Ce n'est pas une topologie qui se rencontre en pratique en télécommunication, mais elle peut toutefois modéliser un noyau très dense d'un réseau, ou un réseau métropolitain (cf. [CCF01]).

2.2.5 Le modèle hiérarchique M-level

Ce modèle proposé par Zegura, Kenneth, Calvert, Michael et Donahoo [ZCD97] prend en compte le caractère hiérarchique d'`Internet`. Le modèle propose une construction récursive en commençant par le niveau le plus élevé. A chaque niveau, le modèle génère un graphe aléatoire connexe en utilisant une méthode non hiérarchique, telle que celles présentées dans les sections 2.2.1, 2.2.2 et 2.2.3. Le niveau suivant de la construction raffine la construction en "éclatant" les nœuds du niveau courant, chaque nœud étant remplacé par un sous-réseau. Plus précisément la construction s'effectue de la façon suivante :

1. Considérer un plan de taille $S \times S$, donc il y a S^2 carrés.
2. Générer un graphe aléatoire connexe tel que la position de chaque nœud soit une valeur entière comprise entre 1 et S.
3. Répéter $M - 1$ fois : dans chaque surface carrée où il y a un nœud, faire :
   (a) Diviser la surface en S^2 carrés, de manière à obtenir une grille $S \times S$.
   (b) Générer un graphe aléatoire connexe dans la grille, tel que la position de chaque nœud soit une valeur entière comprise entre 1 et S.
   (c) Re-connecter les liens[3] du nœud remplacé aux nœuds du graphe généré dans l'étape précédente.
4. Retourner à l'étape 3.

La figure 2.2 décrit l'étape de remplacement des nœuds de la première étape par des sous-réseaux. Le modèle se sert de modèles non hiérarchiques pour générer chaque étape de la hiérarchie.

La principale critique que l'on peut faire à ce modèle est qu'`Internet` reste un réseau avec principalement deux niveaux de hiérarchie : le niveau physique (LAN ou WAN), et le niveau des systèmes autonomes. On pourrait certes parler d'un troisième niveau hiérarchique en considérant une certaine organisation des systèmes autonomes, mais ce niveau n'est représenté par aucun protocole.

Une autre critique que l'on peut faire à ce modèle est qu'il est grosso modo le modèle aléatoire avec loi uniforme plus une hiérarchie greffée dessus. Il ne contient en fait pas de relation géographique très développée.

2.2.6 Le modèle hiérarchique "Transit-Stub"

Zegura, Kenneth, Calvert, Michael et Donahoo [ZCD97] ont proposé un modèle hiérarchique plus adapté au réseau `Internet` que le modèle de la section 2.2.5. Ce modèle est fondé sur

3. Dans l'article de Zegura *et al.* [ZCD97] les connexions ne sont pas spécifiées. Il est par exemple possible d'utiliser le nœud le plus proche.

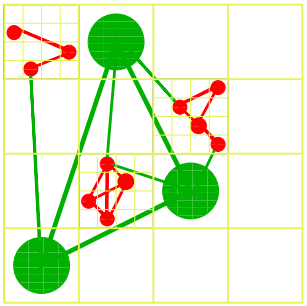

FIG. 2.2: *Exécution du modèle hiérarchique M-level*

la structure d'`Internet`, qui, comme nous l'avons vu dans la section 2.1, est composé de systèmes autonomes de différentes appellations selon leur fonction.

La première étape de la construction d'un graphe selon le modèle de Zegura *et al.* consiste en la génération d'un graphe $G = (V, E)$ de n sommets avec une méthode non hiérarchique quelconque. Cette topologie correspond à celle du *noyau* du réseau. Chaque nœud v représente un "Transit-Domain". On le remplace alors par un graphe non hiérarchique ayant n' sommets en moyenne, de manière à obtenir un autre graphe $G' = (V', E')$, tel que

$$V' = \bigcup_{v \in V} \{\text{sommet de } G_v\},$$

où G_v est le graphe qui a remplacé le nœud v. Puis, pour chaque sommet v' dans le graphe G', générer k graphes non hiérarchiques ayant n'' sommets en moyenne. Ces sommets représentent les "stub-domains". Les stub-domains sont connectés au nœud v' correspondant par un unique lien. Donc on a obtenu un graphe $G'' = (V'', E'')$, où

$$V'' = \bigcup_{v \in V'} \{\text{sommet de } G'_v\}$$

Le modèle ajoute ensuite quelques arêtes pour augmenter la connectivité entre les sommets des "stub-domains" et des "transit-domains". Les auteurs ne spécifient cependant pas le nombre d'arêtes à rajouter.

Ce modèle permet d'obtenir une superposition de certains domaines au niveau géographique. En plus il permet d'obtenir de très grands réseaux à partir de paramètres simples et réalistes. Par exemple le paramètre n est le nombre de "transit-domains" dans l'`Internet`, n' est le nombre moyen des routeurs dans un système autonome, k est le nombre en moyenne de "stub-domains" pour chaque routeur d'un "transit-domain", et n'' est le nombre en moyen de nœuds dans un "stub-domain". Finalement les liens supplémentaires permettent de modéliser les "multihomed-domains". Bien que le modèle soit basé sur une génération de graphes aléatoires, chacun de ces graphes est composé d'un petit nombre des nœuds. Ils ont donc une probabilité raisonnable d'être connectés.

Le problème que présente cependant ce modèle est lié à la distribution des degrés des nœuds qui ne suit pas une loi de puissance [MMB00].

2.2.7 Le générateur "Power law random graph"

Aiello, Chung et Lu [ACL00] ont proposé un générateur de graphes aléatoires qui vérifient les lois de puissance (cf. section 2.1.3) pour modéliser principalement le graphe du Web (Broder, Kumar, Maghoul, Raghavan, Rajagopalan, Stata, Tomkins, et Wiener [BKM+00] on en effet montré que le graphe du Web suit aussi des lois de puissance). Le générateur prend en entrée le nombre de nœuds n, et la constante α de l'équation 2.1. Il assigne aléatoirement des degrés aux n nœuds selon une distribution de loi de puissance de constante α.

En fonction du paramètre α, [ACL00] obtient différents résultats liés à la connexité du graphe.

Pour $\alpha < 1$ le graphe est presque sûrement connexe. Pour $1 < \alpha < 2$ il existe une composante connexe géante, c'est-à-dire de taille $O(n)$. Pour $2 < \alpha < \alpha_0 = 3.4785$, il existe une composante connexe géante telle que la taille de la deuxième plus grande composante est $O(\log n)$. Finalement, pour $\alpha > \alpha_0$ il n'existe pas de composante connexe géante. Évidemment, le modèle proposé dans cet article produit des graphes aléatoires qui peuvent servir comme point de départ pour générer des topologies semblables à celle d'`Internet`. Par contre ce modèle ne tient pas compte de la situation géographique.

2.2.8 Le générateur BRITE

Medina, Matta et Byers [MMB00] ont proposé le générateur BRITE (pour "Boston University Representative Internal Topology Generator"). Ce générateur est basé sur une croissance *incrémentale* du réseau, qui consiste à placer les nœuds un par un. Il est également basé sur la connectivité *préférentielle* qui spécifie la manière de choisir les nœuds auxquels un nouveau nœud se connecte. Plus précisément, la probabilité qu'un nouveau nœud v se connecte au nœud i est :

$$\frac{d_i}{\sum_{j \in \mathcal{C}} d_j} \tag{2.10}$$

où d_i est le degré du nœud i et $\mathcal{C}$ est l'ensemble des nœuds candidats, c'est-à-dire les nœuds qui sont déjà dans le graphe. Le processus est répété jusqu'à ce que le nouveau nœud v soit connecté à exactement m nœuds. Le modèle ajoute un certain nombre de connexions locales. La probabilité qu'un nouveau nœud v se connecte au nœud i est :

$$\frac{w_{v,i} \cdot d_i}{\sum_{j \in \mathcal{C}} w_{v,j} \cdot d_j} \tag{2.11}$$

où $w_{v,i} = \alpha e^{-d(v,i)/\beta L}$ est la probabilité de Waxman entre les nœud v et i, à distance $d(v,i)$ dans un plan de diamètre L (cf. équation (2.9)).

Les auteurs de cette méthode ont montré que la croissance incrémentale et la connectivité préférentielle génèrent des graphes qui vérifient les lois de puissance énoncées par Faloutsos *et al.* [FFF99]. Ce générateur fait donc preuve d'un bon comportement. Cependant, les valeurs

des constantes des lois de puissance sont un peu différentes de celles qui ont été trouvées par Faloutsos *et al.* dans [FFF99]. Magoni et Pansiot [MP02a] on montré que, pour le cas $m = 1$, les graphes obtenus suivent la loi de puissance 2 (cf. l'équation 2.2). Par contre, pour les valeurs de $m > 1$, le coefficient de corrélation est très éloigné de 0.95. Le modèle ne vérifie donc pas la loi de puissance 2. De plus, pour $m = 1$, le graphe obtenu est un arbre, tandis que la topologie d'`Internet` a plusieurs boucles.

2.2.9 Le générateur INET

C. Jin, Q. Chen et S. Jamin [JCJ00] ont proposé le générateur de topologie INET (pour "Internet Topology Generator"). Ce générateur construit un graphe vérifiant la deuxième loi de puissance de Faloutsos *et al.* [FFF99]. Le générateur a deux paramètres : le nombre de nœuds n, et le pourcentage de nœuds de degré 1. La première étape de la génération a pour but de construire un réseau connexe. INET assigne d'abord à chaque nœud son degré en suivant les lois de puissance (cf. l'équation 2.1.3). Le générateur construit ensuite un arbre avec les nœuds qui ont un degré assigné plus grand que 1. Dans un deuxième temps il ajoute les nœuds de degré 1 selon le modèle préférentiel linéaire. La préférence linéaire est la probabilité qu'un nouveau nœud i se connecte au nœud j :

$$\frac{d_j}{\sum_{k \in V} d_k}$$

où d_j est le degré du nœud j, et V est l'ensemble des nœuds. Finalement le générateur assortit les nœuds de degré plus grand que 2 en les connectant entre eux (toujours selon la préférence linéaire).

Il y a plusieurs versions d'INET dans lesquelles les auteurs améliorent successivement leurs résultats. La dernière version est INET-3.0 [WJ02], où la principale innovation porte sur la manière de connecter les nœuds : la préférence linéaire est remplacée par la probabilité $P(i, j)$ d'ajouter un lien entre les nœuds i et j définie par

$$P(i, j) = \frac{w_i^j}{\sum_{k \in V} w_i^k} \tag{2.12}$$

où V est l'ensemble des sommets du graphe des nœuds déjà connectés. Les variables w_i^k sont définies par :

$$w_i^j = d_j \cdot \max \left\{ 1, \sqrt{\left(\log \frac{d_i}{d_j} \right)^2 + \left(\log \frac{f(d_i)}{f(d_j)} \right)^2} \right\}. \tag{2.13}$$

où d_i est le degré du sommet i et $f(d_i)$ est le nombre des nœuds de degré d_i. Les expressions (2.12) et (2.13) permettent de coller aux observations expérimentales du degré des voisins, principalement pour les voisins de degré de 1 à 4. Des observations sur la topologie des systèmes autonomes d'`Internet` montrent qu'il y a un grand nombre de voisins de degré

élevé. Un nombre de voisins élevé permet de joindre un nœud quelconque en très peu de sauts. Cela donne également une grande tolérance aux pannes de liens.

Ce générateur a été conçu pour la génération du graphe correspondant aux systèmes autonomes.

2.2.10 Le modèle continu

Albert et Barabasi [AB00] ont proposé l'utilisation de la théorie du continu pour prédire les propriétés de graphes représentatifs de la topologie d'`Internet`. Dans leur article ils ont utilisé les paramètres suivants : m_0 nœuds, et les valeurs de probabilité a et b. La probabilité a est liée à l'agrégation de nouveaux liens. La probabilité b est liée à la re-connexion des liens. Finalement, la probabilité $1 - a - b$ est liée à l'agrégation des nouveaux nœuds. La description du modèle est la suivante :

- Le générateur commence avec m_0 nœuds isolés. Avec probabilité a, $m < m_0$ liens sont ajoutés de la manière suivante : un nœud k est choisi au hasard selon une loi uniforme, et l'autre extrémité i du lien l_{ij} est choisie avec la probabilité

$$p(i) = \frac{d_i + 1}{\sum_j (d_j + 1)} \tag{2.14}$$

  où d_i est le degré du nœud i. Ce processus est répété m fois.

- On effectue ensuite une étape de re-connexion d'au plus m liens. Un nœud i et son lien l_{ij} sont choisis au hasard (loi uniforme) avec probabilité b . Le lien l_{ij} est déconnecté du nœud j et connecté à j' choisi selon la probabilité $p(j')$ de l'équation 2.14. On obtient un nouveau lien $l_{ij'}$. Ce processus est répété m fois.

- Avec probabilité $1 - a - b$ un nouveau nœud i est ajouté. Ce nœud aura m liens l_{ij}, où j est un nœud dans le graphe choisi avec la probabilité donnée par l'équation 2.14.

L'innovation de ce modèle, par rapport aux modèles décrits précédemment, est l'opération de re-connexion. Chen, Chang, Govindan, Jamin, Shenker et Willinger [CCG+02] ont montré que le processus de re-connexion a un impact lorsqu'il implique la mort et la naissance de nœuds. Ces auteurs de [CCG+02] montrent qu'il est possible de voir une corrélation bien marquée entre les courbes de naissance et de mort des systèmes autonomes et de liens en fonction du temps (cf. ses figures 3.a et 3.b dans [CCG+02]).

2.2.11 Le modèle GPL

Bu et Towsley [BT02] ont analysé et comparé les générateurs de topologies incorporant les lois de puissance. C'est le cas du générateur d'Aiello *et al.* [ACL00], les générateurs Brite [MMB00] et INET [JCJ00], et du générateur d'Albert et Barabasi [AB00]. Bu et

Towsley ont proposé le nouveau générateur GPL (pour "Generalized Linear Preference") dont la principale contribution est une nouvelle probabilité pour choisir un nouveau lien. Cette modification est motivée par le fait que la préférence réelle de connexion envers les nœuds de degré élevé est plus grande que celle produite par la préférence linéaire. Bu et Towsley ont ainsi proposé l'utilisation de la probabilité :

$$\frac{d_i - \beta}{\sum_j (d_j - \beta)} \tag{2.15}$$

où $\beta \in]-\infty, 1]$ est un paramètre qui permet de contrôler la préférence pour les nœuds de degré élevé. Cette préférence diminue avec β. Bu et Towsley ont démontré que cette probabilité génère des graphes vérifiant les lois de puissance. Le reste du modèle est identique à celui d'Albert et Barabasi [AB00], sans l'opération de re-connexion.

2.2.12 Le modèle nem

Magoni et Pansiot [MP02b] ont proposé un modèle de topologie qui extrait des graphes à partir de topologies obtenues par exploration au niveau routeur d'`Internet`. Plus précisément, ils ont proposé un algorithme qui retourne un sous-graphe d'`Internet`. Les topologies de départ utilisées sont celles de :

- Pansiot et Grad [PG98] obtenue en 1995;
- Burch et Cheswick [BC99] obtenue en 1999;
- Govindan et Tangmunarunkit [GT00] obtenue en 1999.

Les graphes générés sont d'une taille plus petite que l'original (la carte d'`Internet`), mais ils conservent certaines propriétés de l'original.

L'algorithme d'extraction du sous-graphe prend en entrée le nombre de nœuds n et le degré moyen d_m souhaités. L'algorithme est composé de trois étapes. La première étape s'occupe de sélectionner un premier nœud (et d'initialiser toutes les structures de données dont a besoin l'algorithme). La deuxième étape construit un arbre de n nœuds dans la carte d'`Internet` enraciné dans le nœud choisi lors de la première étape. Durant cette étape, le modèle de topologie ne sélectionne pas les nœuds de degré élevé, car ils décroissent certaines propriétés typiques liées à la distance, comme la distance moyenne entre tous les nœuds, ou l'excentricité du graphe. La dernière étape ajoute des liens de manière à arriver au degré moyen d_m désiré. Cette étape utilise des informations recueillies lors de l'étape de construction de l'arbre. Les auteurs montrent que les graphes obtenus conservent des propriétés fondamentales de la carte originale, comme la longueur moyenne des plus courts chemins, l'excentricité moyenne, les lois de puissance de degré et de rang (définies par Faloutsos *et al.* [FFF99]), et les lois de puissance de taille et rang des arbres (définies par Magoni et Pansiot [MP01]). Avec ce modèle de topologie, il est de plus possible d'obtenir des topologies de taille suffisamment petite pour être supportées par les simulateurs de réseaux actuels.

L'idée d'obtenir des topologies à partir des cartes d'`Internet` est intéressante. Il reste toutefois à déterminer les relations entre les paramètres de la carte d'`Internet` et ceux des graphes générés. Normalement, la tendance naturelle est que ces paramètres augmentent avec la taille du graphe, mais la question est de déterminer avec quelle loi. Par exemple, on peut vérifier que la distance moyenne dans le graphe augmente avec la taille du graphe, mais la loi d'augmentation dépend du type de graphe.

2.2.13 Le modèle "Heuristically Optimized Trade-offs"

Le modèle proposé par Fabrikant *et al.* [FKP02] est un modèle incrémental, où chaque nouveau nœud i doit choisir à quel nœud j il se connecte. La construction est menée dans une aire géographique quelconque, par exemple un carré. Un nouveau nœud i choisit son voisin j en minimisant une fonction de coût qui tient compte d'une part de la distance euclidienne $d(i,j)$, et d'autre part d'une autre distance $h(j)$ qui mesure certaines caractéristiques topologiques. Dans [FKP02], $h(j)$ est défini comme la distance en nombre de sauts (ou *hops*) entre j et le premier nœud arrivé dans le réseau. La motivation de ce coût est la comparaison entre le coût de l'installation d'un lien de grande longueur et l'avantage d'être connecté au "centre" du réseau. Plus précisément, la fonction de coût est

$$W(i,j) = d(i,j) + \xi \cdot h(j) \tag{2.16}$$

où ξ est un paramètre du modèle. Le nœud i choisira le nœud j qui minimise $W(i,j)$.

Dans [FKP02], il est démontré que si le nombre de nœuds est supérieur à 64, alors, pour $\frac{1}{\sqrt{N}} < \xi \leq \frac{1}{4}$, la distribution des degrés des nœuds suit une loi de puissance. La valeur de l'exposant est plus petit que $\frac{-1}{6}$ pour $\xi \gg 1/\sqrt[3]{N}$. On peut cependant faire de nombreuses critiques sur ce modèle. La première critique est que le modèle produit un arbre, alors que les réseaux de type `Internet` possèdent bien sûr de nombreux cycles. La deuxième critique est que le modèle ne tient pas compte des modifications qui peuvent être faites dans le réseau (par exemple de nouveaux liens ajoutés pour améliorer le réseau) car il est purement incrémental. Enfin, la troisième critique est liée à la fonction $h(j)$ qui mesure le nombre de sauts vers le premier nœud de l'arbre, c'est-à-dire la racine. Dans `Internet`, il n'existe pas de centre, donc si on veut faire un modèle qui se rapproche de la réalité on doit chercher une autre fonction.

2.3 Un nouveau modèle pour `Internet`

Dans les sections précédentes, nous avons vu différents modèles d'`Internet` proposés dans la littérature. Les plus adéquats vérifient les lois de puissance [FFF99] et développent une construction par agrégation. Ces modèles ne tiennent cependant pas compte de la situation géographique des machines. Dans cette section nous proposons un nouveau modèle pour `Internet`[4]. Ce modèle est une extension du modèle de Fabrikant, Koutsoupias et Papadi-

4. Le modèle a été conçu en collaboration avec C. Kenyon et N. Schabanel

mitriou [FKP02]. Il est basé sur plusieurs paramètres afin de produire des topologies qui vérifient d'une part les lois puissance, et d'autre part les observations réalisées sur la dynamique des réseaux. De plus, notre modèle tient compte de la géographie. Pour la validation du modèle, nous allons vérifier les lois puissance, le diagramme "hop-plot" (le nombre de paires des nœuds en fonction de leur distance en sauts ou "hops"), l'excentricité des nœuds, le diamètre, le coefficient de "clustering", et la longueur moyenne des chemins. L'analyse de ces paramètres montre que les graphes générés par notre modèle vérifient de nombreuses propriétés de la topologie d'`Internet`.

2.3.1 Le modèle

Vu certains défauts du modèle de Fabrikant *et al.* [FKP02], nous y apportons les modifications suivantes :

1. Chaque nouveau nœud se connecte à m nœuds suivant la loi de l'équation 2.16 ($m = 1$ dans [FKP02]).

2. Pour chaque nœud ajouté, notre modèle ajoute k liens de la manière suivante. On cherche le lien (i, j) qui minimise la fonction suivante :

$$d(i,j) + \frac{\gamma}{n} \cdot \sum_{i=1}^{n} \big(h'(i) - h(i)\big) \tag{2.17}$$

   où

   - n est le nombre courant de sommets dans le réseau,
   - $d(i, j)$ est la distance euclidienne entre i et j,
   - $h'(i)$ est la distance entre les sommets i et la racine actuelle r dans le graphe incorporant le nouveau lien (i, j),
   - $h(i)$ est la distance *sans* incorporer le nouveau lien (i, j),
   - et γ est un paramètre de notre modèle.

   Cette opération est répétée pour les k liens à ajouter.

3. Notre modèle change dynamiquement la fonction $h(j)$ car elle mesure la distance en sauts entre le nœud j et un nœud r, appelé "racine", potentiellement différent du premier nœud. Après l'ajout d'un nœud, la racine est tirée au hasard selon le degré des nœuds. La probabilité qu'un nœud i a de devenir la racine est :

$$p_i = \frac{d_i}{\sum_{j \in V} d_j} \tag{2.18}$$

   où d_i est le degré du nœud i, et V est l'ensemble des nœuds dans le graphe courant.

Le point (1) donne plus de flexibilité en permettant la présence de liens de sécurité. Le point (2) est motivé par des mesures statistiques sur les systèmes autonomes. Chen *et al.* [CCG+02] montrent que pour chaque nouveau système autonome qui se connecte, il y a en moyenne trois liens qui sont ajoutés dans le réseau. Enfin, le point (3) propose une fonction pour mesurer la distance au "centre" du réseau. La manière la plus exhaustive de mesurer cette distance est de calculer les distances entre tous les nœuds. La complexité de ce calcul est $O(n^3)$, où n est le nombre de nœuds déjà connectés dans le réseau. La manière proposée permet un temps de calcul $O(n)$. L'approximation utilisée est que les nœuds de degré élevé sont ceux qui sont les plus connectés, donc *a priori* ils sont plus "au centre" que les autres.

2.3.2 Validation du modèle

Dans la suite, nous décrivons les résultats obtenus lors de notre campagne de simulation.

Nous avons généré plusieurs graphes selon notre modèle et nous avons calculé les paramètres suivants.

- Le degré moyen $\overline{deg}$ est calculé comme la moyenne des degrés de tous les nœuds.
- Le coefficient de voisinage ("clustering") d'un nœud v est calculé comme le rapport entre le nombre d'arêtes entre les d_v voisins de v, et le nombre d'arêtes d'une clique (un graphe complet) de taille d_v. Le coefficient de voisinage moyen $\overline{Clust}$ est la moyenne des coefficients de voisinage de tous les nœuds du graphe.
- L'excentricité d'un nœud est la distance maximale entre ce nœud et tous les autres nœuds du graphe. L'excentricité minimum ex_m est la plus petite excentricité de tous les nœuds du graphe. L'excentricité moyenne $\overline{ex}$ est la moyenne des excentricités des nœuds.
- Le diamètre D d'un graphe est la distance maximum entre deux nœuds du graphe.
- $\overline{d_h(\cdot,\cdot)}$ est la moyenne de tous les distances entre les nœuds du graphe.
- L'exposant β de la loi de puissance du degré (cf. équation (2.2)).
- Le coefficient de corrélation ACC calculé comme :

$$ACC = \frac{\sum_d (f(d) - \overline{f(d)}) \cdot (y(d) - \overline{y(d)})}{\sqrt{\sum_d (f(d) - \overline{f(d)})^2} \cdot \sqrt{\sum_d (y(d) - \overline{y(d)})^2}} \tag{2.19}$$

  où $f(d)$ est le nombre de nœuds de degré d, $\overline{f(d)}$ est la moyenne de $f(d)$, $y(d) = \alpha \cdot d^{\beta}$ est la loi puissance, et $\overline{y(d)}$ est sa moyenne. Ce coefficient donne un résultat $\in [-1, 1]$. Si $|ACC| = 1$ les deux fonctions sont totalement corrélées, tandis que pour $ACC = 0$ elles sont dé-corrélées.

Les valeurs des paramètres m et k adoptées pour faire nos simulations sont $m = 1$ et $k = 1$.

Nous avons généré plusieurs graphes avec différents paramètres ξ de [FKP02] et γ. Nous avons trouvé que pour $\gamma \leq \xi$ les graphes obtenus par notre modèle sont très semblables entre eux. Ils commencent à se différencier pour $\gamma > \xi$. Si on compare deux graphes avec le même ξ mais avec $\gamma_1 \leq \xi < \gamma_2$, on voit que les exposants correspondants à la loi de puissance vérifient $\beta_2 < \beta_1$, où β_1 es plausible pour $\xi \simeq 0.03$, mais β_2 s'éloigne des valeurs observées pour `Internet`($\beta \simeq 2.2$). Nous avons donc choisi $\gamma \leq \xi$ pour générer les graphes.

Nous avons également remarqué que pour augmenter la valeur de l'exposant β on doit diminuer ξ. Pour illustrer deux cas, nous avons choisi les deux valeurs suivantes de $\xi = \gamma = 0.03$, et $\xi = \gamma = 0.001$.

Dans la table 2.2 nous montrons les résultats obtenus en fonction de différentes tailles de graphe. Les deux premières colonnes représentent les paramètres du modèle, c'est-à-dire le nombre total N de nœuds, ξ et γ. Les colonnes suivantes montrent les paramètres calculés. Les six dernières colonnes montrent l'exposant β de la loi de puissance relative au degré, l'ACC, et l'ACC_ℓ calculé dans l'espace linéaire. Puis le même exposant β, noté ici β', calculé sans considérer les nœuds de degré 1 et les deux ACC correspondants. Le calcul de β a été fait de manière standard, c'est-à-dire en approximant une droite dans l'espace log-log. Les ACC sont calculés soit dans l'espace log-log, soit dans l'espace linéaire.

N	$\xi = \gamma$	$\overline{deg}$	$\overline{Clust}$	ex_m	$\overline{ex}$	D	$\overline{d_h(\cdot,\cdot)}$	β	ACC	ACC_ℓ	β'	ACC'	ACC'_ℓ
20	0.001	3.50	0.2619	3	3.950	5	2.345	-0.42	0.43	0.21	-1.00	0.70	0.58
20	0.030	3.50	0.2444	3	3.850	5	2.235	-0.72	0.75	0.55	-1.16	0.86	0.90
50	0.001	3.80	0.2519	4	5.560	7	3.186	-0.98	0.80	0.56	-1.47	0.89	0.91
50	0.030	3.80	0.2305	4	5.400	7	3.064	-0.88	0.76	0.54	-1.29	0.85	0.87
100	0.001	3.90	0.2268	5	6.850	9	3.862	-1.27	0.89	0.68	-1.65	0.93	0.89
100	0.030	3.90	0.2217	5	6.660	9	3.663	-1.27	0.87	0.66	-1.56	0.89	0.91
200	0.001	3.95	0.2378	5	7.565	10	4.499	-1.54	0.89	0.60	-1.97	0.93	0.92
200	0.030	3.95	0.1992	5	6.830	9	4.129	-1.45	0.95	0.74	-1.73	0.97	0.93
500	0.001	3.98	0.2038	6	9.070	12	5.495	-1.85	0.92	0.61	-2.26	0.95	0.89
500	0.030	3.98	0.1910	5	7.480	9	4.864	-1.77	0.94	0.72	-2.05	0.96	0.97
1000	0.001	3.99	0.2186	7	10.436	14	6.240	-1.94	0.93	0.56	-2.37	0.97	0.88
1000	0.030	3.99	0.1887	6	8.510	11	5.377	-2.02	0.94	0.73	-2.31	0.96	0.93
2000	0.001	3.99	0.2186	7	10.436	14	6.240	-2.14	0.93	0.49	-2.61	0.97	0.89
2000	0.030	3.99	0.1798	6	8.773	11	5.821	-2.12	0.96	0.77	-2.38	0.98	0.95

TAB. 2.2: *Paramètres de graphes de différentes tailles*

On observe tout d'abord que notre modèle permet d'obtenir des topologies de petite taille. Bien que l'ACC soit petit dans les cas où N est petit, ACC montre une tendance à respecter la loi de puissance. Le nombre de nœuds de degré 1 reste toutefois petit, ce qui réduit la valeur du coefficient ACC. C'est un point à améliorer dans le modèle. On observe aussi que, même pour des valeurs $ACC \simeq 0.95$, les ACC linéaires sont très bas, environ 0.73. La raison peut en être la remarque 2 faite à la section 2.1.3.

Comme on a généré des topologies avec $m = 1$ et $k = 1$, on n'ajoute que 2 liens en moyenne pour chaque nœud. Donc $\overline{deg}$ tend vers 4. Pour la topologie d'`Internet` au niveau routeur,

$\overline{deg} \simeq 3.1$ (voir [MP02a]). Une amélioration qu'il est possible d'apporter au modèle est de remplacer le paramètre k par une fonction de probabilité exponentielle qui relie sa moyenne avec le degré moyen désiré. Ce changement aura des répercutions sur le nombre de nœuds de degré 1. Nous comptons faire cette étude prochainement.

Une autre particularité de notre modèle est la stabilité de certains paramètres comme le diamètre, l'excentricité et la distance moyenne $\overline{d_h(\cdot,\cdot)}$. Ils augmentent avec la taille du graphe généré mais le font lentement. En particulier la distance moyenne se rapproche de celle du modèle de topologie "nem" (cf. section 2.2.12).

Nous présentons les diagrammes "Hop-plot" dans la figure 2.3, pour deux graphes de $N = 1000$ et $N = 2000$ nœuds. L'axe des ordonnées est le nombre de paires de nœuds à distance d, et l'axe des abscisses est la distance $d+1$. L'exposant δ de l'équation (2.4) est 3.330 pour le graphe de $N = 1000$, et 3.526 pour le graphe de $N = 2000$, avec $ACC = 0.99$ pour les deux graphes. Pour faire ce calcul, on n'a utilisé que les valeurs plus petites que l'asymptote horizontale $y = N^2$. Ces approximations ont un $ACC = 0.98$ dans l'espace linéaire.

Finalement nous montrons dans la figure 2.4 un graphe de 500 nœuds généré avec $\xi = \gamma = 0.001$, et un graphe de 500 nœuds avec $\xi = 0.03 = \gamma = 0.03$ est présenté dans la figure 2.5. Les deux graphes sont semblables pourtant il y a quelques différences importantes. Dans le cas $\xi = \gamma = 0.001$ le nombre de nœuds de degré 1 est plus élevé. Il a donc plus de sous-graphes en arbre que pour $\xi = \gamma = 0.03$. En plus, on voit dans la table 2.2 que les propriétés liées à la distance sont plus élevées pour $\xi = \gamma = 0.001$ que pour $\xi = 0.03 = \gamma = 0.03$.

Nous montrons à titre de comparaison un plan du noyau du réseau UUNET à la figure 2.6, obtenu avec *Mapnet*[5]. Le noyau d'UUNET a 129 nœuds. Cette représentation montre les liens multiples avec de multiples lignes, donc le dessin reste un peu confus. Nous montrons le plan mondial, un zoom sur les États-Unis, et un autre sur la côte Est. Comme il s'agit du noyau, il n'y a pas beaucoup des nœuds de degré 1. On s'aperçoit que la distribution de population est très importante pour le placement des nœuds. Nous prévoyons donc de l'inclure dans la prochaine version de notre générateur de topologies.

Si on compare les topologies générées avec le noyau d'UUNET on voit qu'il y a une certaine ressemblance. On trouve en particulier dans tous les cas des nœuds de degré élevé qui sont interconnectés entre eux. On trouve aussi un nombre de triangles élevé dans le noyau d'UUNET comme dans les modèles générés.

2.3.3 Conclusion

Nous avons présenté un nouveau modèle d'`Internet` qui permet de générer des topologies de taille arbitraire, tout en vérifiant la deuxième loi de puissance.

Le modèle est basé sur le modèle de Fabrikant, Koutsoupias et Papadimitriou [FKP02]. Les avantages de ce modèle sont l'inclusion de la géographie au travers de la prise en compte de la position des nœuds, et du développement incrémental du graphe. Nous avons ajouté

5. Mapnet est un outil pour dessiner les noyaux des réseaux de recherche et commerciaux. Voir `http://www.caida.org/tools/visualization/mapnet/Backbones/`.

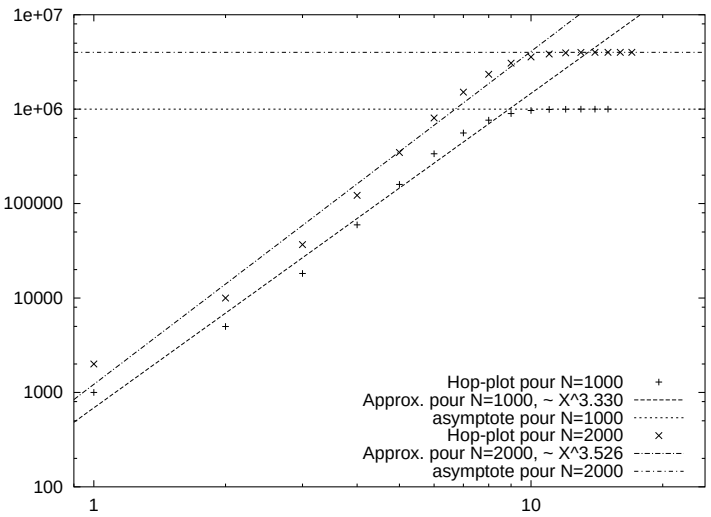

FIG. 2.3: *Diagramme "Hop-plot" des topologies générées*

l'agrégation de liens pour créer des graphes qui ne soient pas uniquement des arbres. Nous permettons le changement de la mesure au "centre du graphe". Les topologies engendrées sont prometteuses puisqu'elles vérifient certaines lois de puissance, et donnent des paramètres semblables à ceux d'`Internet`.

Cependant, il faut continuer à travailler pour améliorer le nombre de nœuds de degré 1 et pour vérifier toutes les lois de puissance, comme celles qui ont une relation avec la taille des arbres et le rang proposées par Magoni et Pansiot [MP01]. Il faudra aussi inclure une information relative à la population pour placer les nœuds, ce qui nous donnera une topologie plus réaliste au niveau géographique. En effet notre modèle connecte les nouveaux nœuds en optimisant une fonction qui prend déjà en compte les distances géographiques et les distances dans le réseau.

Un autre point ouvert est d'établir comment varient les paramètres avec la taille du réseau. Une approche sur les lois de puissance est donnée par Winick et Jamin [WJ02].

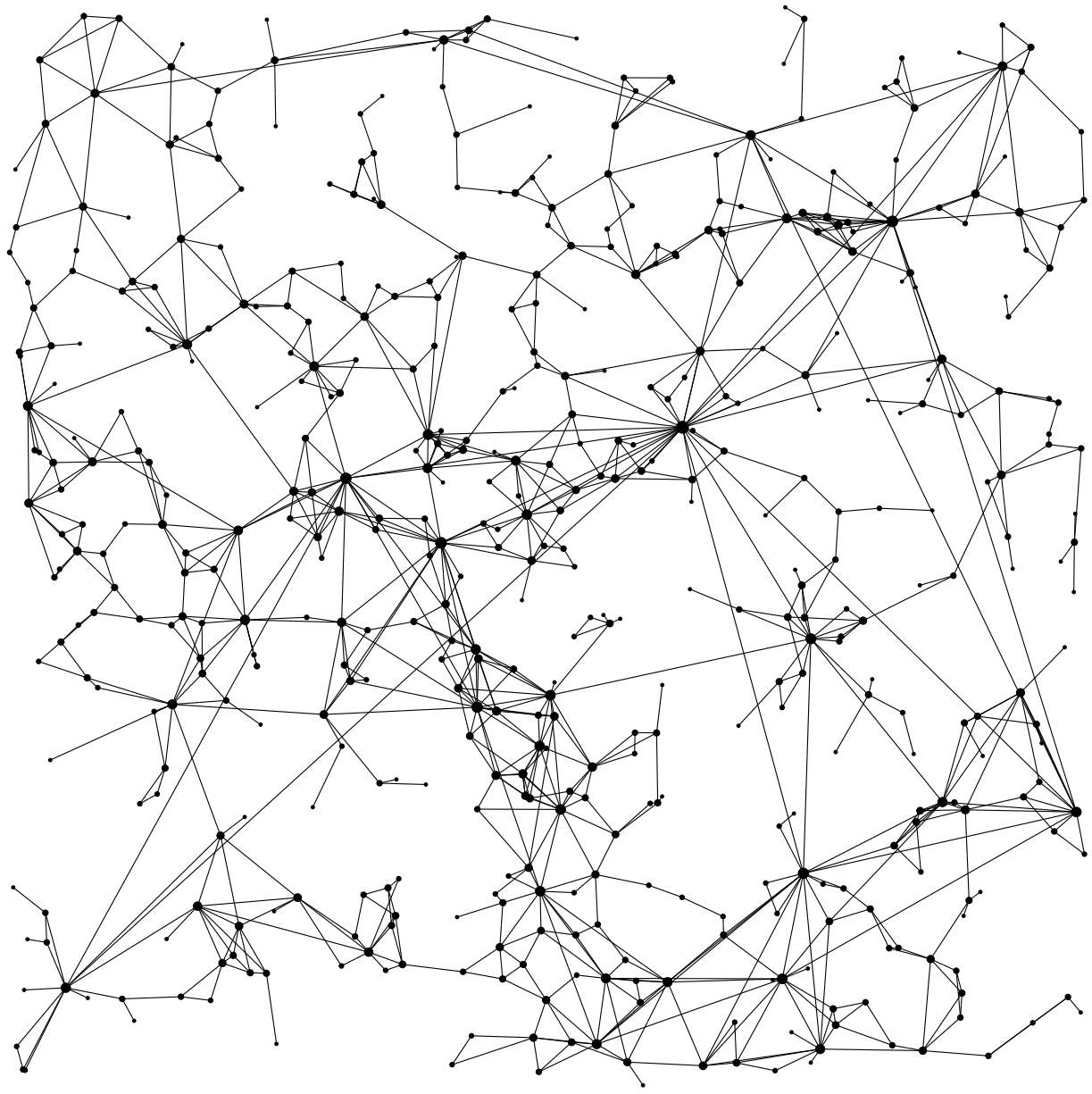

FIG. 2.4: *Modèle d'*`Internet` *avec* $\xi = \gamma = 0.001$

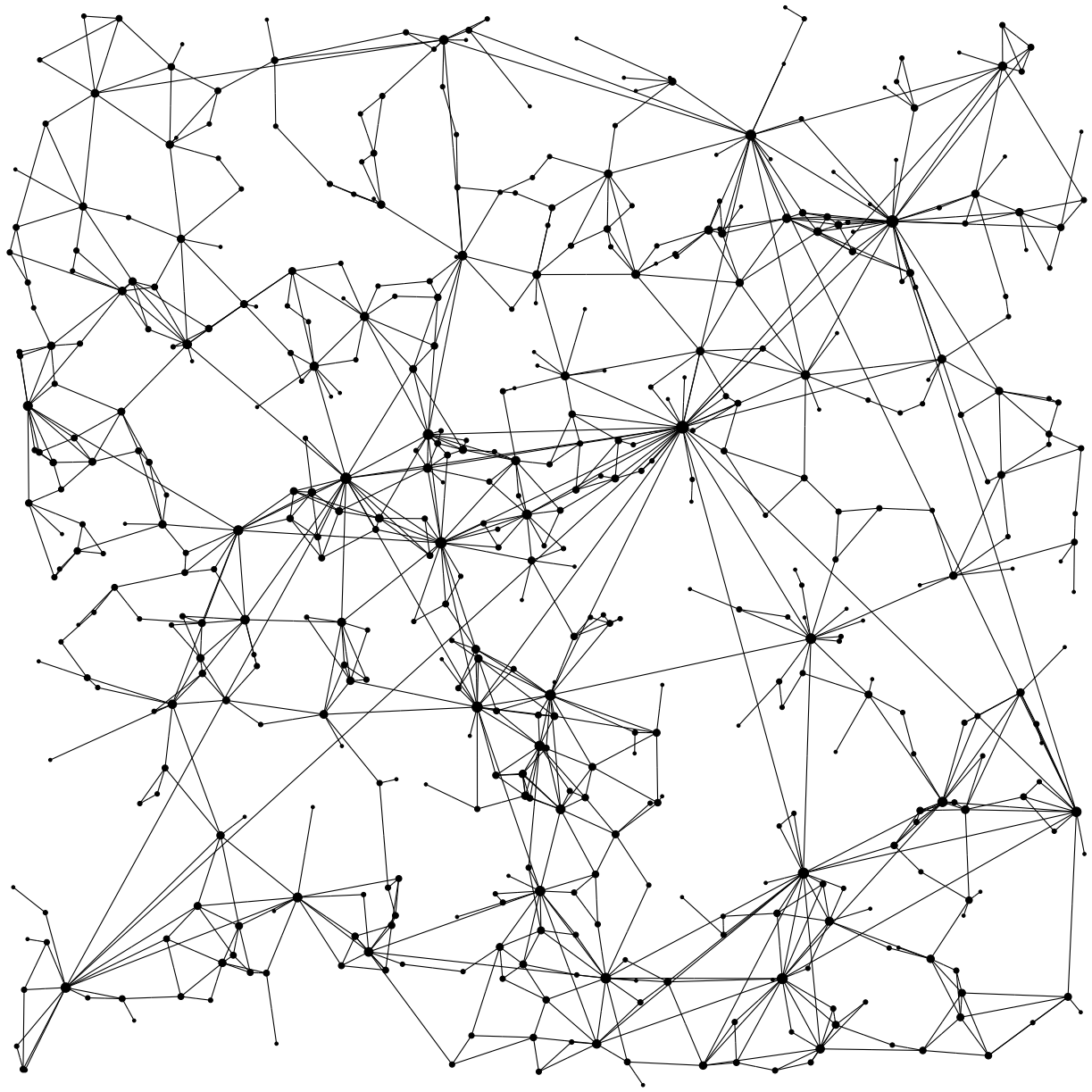

FIG. 2.5: *Modèle d'*`Internet` *avec* $\xi = \gamma = 0.03$

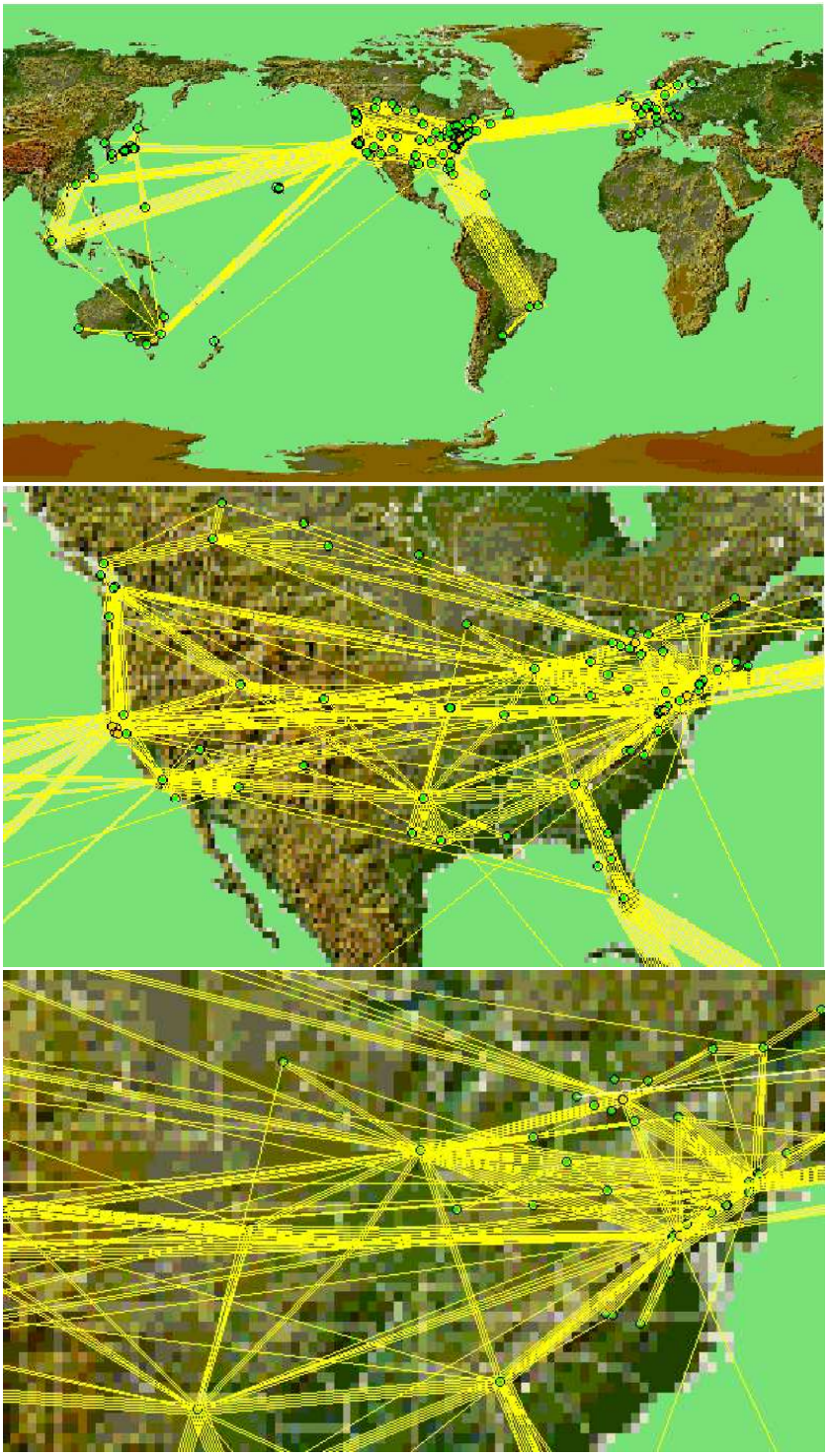

FIG. 2.6: *Réseau* UUNET

Chapitre 3

Communications multipoint

`Internet` est né dans les années 70 pour être un réseau d'interconnexion entre ordinateurs. Il fonctionnait selon le paradigme "client-serveur". Dans ce cadre, un ordinateur devient serveur d'un autre qui est client. Toutes les applications ont donc été développées dans le mode point-à-point ("unicast"). La pile de protocoles TCP/IP (TCP est pour "Transport Control Protocol" et IP est pour "Internet Protocol") a été mise en œuvre pour rendre possibles et efficaces les communications point-à-point. Or, `Internet` s'est ensuite considérablement agrandi, et de nouveaux besoins sont apparus. Parmi ces derniers, les applications multipoint ("multicast") correspondent à un besoin croissant, par exemple, la diffusion d'une radio sur `Internet`, ou le partage d'un tableau blanc virtuel. Ces nouvelles applications ont donné lieu au développement de nouvelles techniques de routage multipoint.

Sur `Internet`, on cherche toujours à concevoir des protocoles les plus simples possibles afin de permettre des implémentations rapides. Il y a concurrence entre la vitesse des liens et les capacités de calcul du matériel. Le paradigme qui s'applique généralement est le suivant : si l'algorithme de routage est simple, alors on peut trouver des moyens pour le faire tourner sur n'importe quelle technologie de communication. Il convient cependant de modérer ce point de vue. En effet, on peut profiter des progrès effectués sur l'architecture des processeurs pour proposer maintenant des algorithmes certes plus complexes, mais plus performants aussi. Par exemple une solution triviale pour les communications multipoint est l'utilisation de l'inondation dans tout le réseau. Cette solution surcharge tout le réseau avec des paquets qui ne concernent pas la plupart des utilisateurs, et est donc uniquement applicable dans le cas où le réseau est petit. Les protocoles multipoint sont un peu plus complexes que les protocoles point-à-point, mais le gain en performance peut être significatif.

Dans ce chapitre nous présenterons tout d'abord un état de l'art des protocoles multipoint actuels. Cette étude nous permettra de mettre en évidence certains problèmes inhérents aux communications multipoint.

Nous présentons ensuite deux protocoles dédiés aux communications multipoint. Le protocole MSDA retourne l'ordre optimal d'envoi de messages à l'intérieur d'un arbre multipoint. Nous avons étudié l'amélioration potentielle pour un réseau réel où les temps de transmission t_{trans} des liens et de commutation t_{comm} des nœuds peuvent varier significativement. Nous allons

montrer que si $t_{trans} < c \cdot t_{comm}$, l'amélioration apportée par MSDA est d'au moins 10% dès que $c \leq 6$.

Enfin, nous proposons le protocole MCT, permettant de construire un arbre multipoint minimisant un paramètre de QoS spécifié. MCT explore plusieurs routes entre la source et les destinations. Cela est rendu possible en se basant sur un autre protocole : le protocole MPDR. Ce dernier protocole maintient des chemins multiples dans les tables de routage. Pour chaque destination, MPDR maintient en chaque nœud v une route qui passe par chaque voisin de v. Utilisant MPDR, MCT est capable d'explorer, pour chaque membre v du groupe multipoint, tous les chemins de longueur au plus $\rho \cdot d + r$, où d est la distance entre la racine (ou "cœur") de l'arbre et v, et ρ et r sont deux paramètres du protocole. Le nombre de messages générés par MCT pour construire son arbre dépend donc de ρ et de r. Nous montrons que les arbres obtenus sont déjà très proches de l'arbre optimal pour des petites valeurs de ρ et de r, typiquement $\rho = 1$ et $r = 2$.

3.1 Quelques algorithmes multipoint connus

Cette section est dédiée à un bref état de l'art des protocoles multipoint proposés jusqu'à ce jour. Nous respecterons l'ordre arbitraire chronologique.

Les réseaux LAN ("Local Area Network") permettent la diffusion de façon inhérente, et la communication multipoint peut donc être faite en profitant de cette propriété. Par contre il n'existe pas de mécanisme de diffusion naturel dans les réseaux WAN ("Wide Area Network"). Le protocole IGMP ("Internet Group Management Protocol") prend en charge l'interconnexion entre les protocoles multicast des réseaux LAN et WAN, et même s'il n'y a que des LANs. Ce protocole s'occupe de maintenir sur chaque routeur de sortie d'un LAN une table de routage. Cette table contient la liste de tous les nœuds qui sont membres de tel ou tel groupe multicast. Le routeur de sortie du LAN est en charge de diffuser les messages multicast dans le LAN s'il existe au moins un membre de ce groupe dans le LAN. Dans la suite nous ne descendrons pas au niveau LAN (dont le routeur est la sortie que les messages multipoint du LAN utilisent pour gagner le reste du réseau WAN).

Remarque. `Internet` est en fait une union de systèmes autonomes. Un système autonome est un réseau WAN, et ses réseaux LANs associés ont une administration commune. On parlera donc très souvent d'intra-domaine pour caractériser l'intérieur du domaine, et d'inter-domaine pour faire référence à l'interaction entre les domaines.

3.1.1 DVMRP

Le protocole DVMRP (pour "Distance Vector Multicast Routing Protocol") a été le premier protocole proposé pour les communications multipoint. Son appellation vient de la méthode utilisée pour la construction des arbres. La méthode de la "distance vectorielle" avait en effet déjà été mise en œuvre pour le premier algorithme de routage distribué point-à-point :

le protocole RIP (pour "Routing Information Protocol"). Ce dernier protocole se sert de l'algorithme de Ford-Fulkerson pour calculer les tables de routage. Ces tables contiennent, pour toute destination u, la distance mesurée en nombres de sauts entre le nœud courant et u. Pour construire ces tables, les routeurs échangent leurs vecteurs de distance (contenant toutes les paires $\{destination, distance\}$). Après un nombre fini d'échanges, tous les routeurs ont leur table à jour.

Intra-domaine, DVMRP construit ses arbres multipoint en utilisant l'inondation et l'élagage de la manière suivante. La source s inonde le réseau avec un message M contenant le numéro m du groupe multipoint. Un nœud v recevant le message pour la première fois marque alors l'interface physique sur laquelle il a reçu le message comme interface d'entrée pour le groupe m. Le nœud voisin de v à travers cette interface est appelé le père de v. Pour éviter que tout le réseau soit inondé par les messages de tous les groupes multipoint, une procédure permet d'élaguer les chemins qui ne sont pas concernés par un groupe. Cette procédure se sert de l'information du protocole IGMP pour savoir s'il y a dans un réseau LAN des membres d'un groupe multipoint déterminé. S'il n'y a pas de membre, le LAN envoie un message d'élagage. Chaque routeur qui reçoit ce message le renvoie à son père si et seulement si il n'y pas de membres de ce groupe atteignables au travers de ses autres interfaces de sortie. De cette manière, DVMRP assure l'élimination de tous les chemins non nécessaires. Pour assurer la connectivité, DVMRP élimine périodiquement ses filtres d'élagage. Il élimine également ses filtres dès qu'un nouveau client arrive dans le groupe. Chaque routeur garde finalement dans une table l'interface d'entrée i associée au numéro m du groupe, et les interfaces qui sont élaguées. À partir de sa table, un routeur pourra renvoyer tous les messages multicast qui arrivent par l'interface i et qui appartiennent au groupe m à toutes les interfaces qui ne sont pas élaguées.

Entre domaines DVMRP procède à partir de la construction de tunnels, c'est-à-dire d'interconnexion entre routeurs spécifiés qui appartiennent aux deux domaines. Un tunnel consiste en un canal virtuel entre deux routeurs. Ce canal ressemble à une connexion physique du point de vue du protocole. La mise en place de ces tunnels se fait "à la main", suite à un accord entre les administrateurs des deux domaines.

Remarque. DVMRP construit un arbre par source et par groupe multipoint, ce qui peut être coûteux en ressources. Ainsi, bien que le protocole soit simple, le nombre d'états maintenus par les routeurs peut être très élevé.

Actuellement, DVMRP est un protocole standardisé sur `Internet`. Il est utilisé de manière expérimentale dans le réseau Abilene [wwwc].

3.1.2 MOSPF

MOSPF [Moy97a] (pour "Multicast Open Shortest Path First") est un protocole dérivé d'OSPF. OSPF est un protocole pour les communications point-à-point. Dans OSPF, chaque routeur connaît le graphe de tout le réseau. Cette information est maintenue par inondation de messages. Les routeurs se servent de la connaissance de tout le graphe pour construire

un arbre en utilisant l'algorithme de Dijkstra. Évidemment, cette inondation peut introduire beaucoup de trafic dans le réseau si sa taille est grande. Ainsi OSPF est conçu pour opérer intra-domaine, où il y a une administration centralisée. OSPF a la possibilité d'utiliser plusieurs métriques, par exemple le délai moyen ou le nombre de sauts. Notons qu'il est possible de fragmenter un domaine pour éviter des inondations dans tout le réseau. Il y a cependant des contraintes topologiques pour que cette fragmentation fonctionne. En particulier, tous les sous-domaines doivent se connecter au sous-domaine central, formant une étoile au niveau des connexions entre sous-domaines.

MOSPF profite de la connaissance de tout le graphe du réseau pour construire un sous-arbre de l'arbre obtenu pour les communications point-à-point, et cela pour chaque source et chaque groupe multipoint (la construction de l'arbre devient un peu plus complexe dans le cas de fragmentation du domaine). Les arbres multipoint obtenus sont donc unidirectionnels.

Bien que MOSPF soit une extension logique du protocole OSPF, il a les mêmes problèmes que le protocole DVMRP. En particulier, le nombre d'états dans chaque routeur reste important à cause de la construction d'un arbre pour chaque source. De plus, la construction de chaque arbre demande des temps de calcul important sur chaque routeur qui utilise l'algorithme de Dijkstra (cet algorithme a une complexité $O(n \cdot \log n)$). Enfin, pour que MOSPF puisse opérer inter-domaine, il faut la mise en place de tunnels. Ce mécanisme n'est toutefois pas encore implémenté dans la version actuelle de MOSPF. L'interaction LAN-WAN est toujours laissée au protocole IGMP.

3.1.3 CBT

CBT [BFC93, Bal97] (pour "Core-Based Tree") a été le premier protocole utilisant un arbre partagé. CBT construit en effet un unique arbre pour chaque groupe multicast. Cet arbre est partagé entre toutes les sources. De cette manière, CBT limite le nombre d'états dans chaque routeur. CBT se base sur un arbre bidirectionnel, de sorte que tous les routeurs puissent envoyer des paquets d'information en faisant une diffusion dans l'arbre multipoint. La distance entre deux membres d'un groupe multipoint utilisant CBT est au plus le double de celle obtenue en utilisant RSP [Wax88]. Étant donné que les arbres partagés réduisent le nombre d'états dans les routeurs, le bilan reste donc positif.

La construction de l'arbre commence à un nœud spécifique : la racine (appelé "cœur" dans CBT). A partir de la racine, tous les nouveaux clients se connectent en utilisant le chemin le plus court obtenu à partir du protocole point-à-point sous-jacent.

L'utilisation d'une racine pose quelques problèmes. Ainsi, comment choisir la meilleure racine ? Comment la modifier lorsque le groupe évolue (par exemple quand plusieurs membres changent pendant le déroulement de l'application multipoint) ? Ces problèmes doivent toutefois être modérés par la constatation que l'unique fonction différentielle de la racine apparaît lors de l'étape de connexion des nouveaux membres. Une fois que cette étape est passée, la racine se comporte comme un nœud quelconque dans l'arbre : il renvoie les messages. L'arbre peut toutefois être déséquilibré si la racine n'est pas au "centre" du groupe.

Remarque. Une caractéristique de CBT est de pouvoir s'utiliser intra-domaine et inter-domaine.

Le défaut le plus important de CBT apparaît lorsque de nombreuses sources engendrent beaucoup de trafic. Dans ce cas, les liens de l'arbre unique sont très chargés, à l'inverse du cas où les sources peuvent construire chacune leur arbre, ce qui permet de mieux distribuer le trafic dans le réseau.

Actuellement CBT est un protocole standardisé dans `Internet`. Il s'utilise de manière expérimentale dans le réseau Abilene [wwwc].

3.1.4 PIM-SM

PIM-SM [DEF+94, EFH+98] (pour "Protocol Independent Multicast - Sparse Mode") a été défini presque en même temps que CBT. Il construit un arbre partagé comme CBT, mais permet cependant également de construire des arbres dédiés aux sources générant un trafic important.

La construction de l'arbre partagé est faite à partir d'une racine (point de *rendez-vous* dans la terminologie PIM-SM). Les clients s'agrègent en suivant le plus court chemin généré par le protocole de communication point-à-point sous-jacent. L'arbre est unidirectionnel. Les sources envoient leurs paquets à la racine, et c'est cette racine qui distribue les paquets. S'il y a une source qui envoie beaucoup de trafic, elle peut se construire un arbre propre, avec les plus courts chemins entre la source et toutes les destinations. Cette procédure opère de façon à passer d'un arbre partagé à un arbre enraciné en une source de fort débit. Si la source diminue son trafic, elle ne revient cependant pas à l'arbre partagé.

Remarque. Pour garder leur connexion, les membres d'un groupe doivent envoyer périodiquement des messages de connexion. Ceci engendre un trafic supplémentaire pour maintenir la connectivité.

PIM-SM utilise une fonction de hachage pour sélectionner le nœud de *rendez-vous* parmi certains routeurs qui participent à l'arbre multipoint (sélectionnés selon le mécanisme décrit dans [DEF+94, EFH+98]). Cette méthode assure la réélection du nœud de *rendez-vous* si l'actuel a disparu.

Ce protocole est prévu pour fonctionner intra et inter-domaine. Toutefois, s'il y a plusieurs groupes multicast avec quelques sources de trafic élevé, le problème du nombre d'états dans les routeurs se produit de nouveau.

Actuellement, PIM-SM est standardisé pour `Internet`. Il s'utilise de manière expérimentale dans le réseau Abilene [wwwc].

3.1.5 YAM

YAM [CC97] (pour "Yet Another Multicast protocol") est le premier protocole proposé capable de prendre en compte la qualité de service ("Quality of Sevice" ou "QoS"). De plus, YAM utilise une autre méthode pour la construction des arbres multipoint, la construction gloutonne. Un nouveau membre se connecte en cherchant le point de l'arbre multipoint le plus proche de lui. Plus précisément, un nouveau membre commence à chercher à distance 1 (ses voisins), puis 2, puis 3, etc. S'il ne trouve pas l'arbre à distance d, il cherche à distance $d+1$. Ce processus est répété jusqu'à ce qu'il trouve l'arbre. Cette méthode d'agrégation à l'avantage d'être simple. Les arbres obtenus sont toutefois différents de ceux produits par la technique des plus courts chemins au nœud de contact (racine, cœur, ou point de rendez-vous). Nous étudierons ces différences à la section 4.2.3.

Pour inclure la QoS il faut que le nouveau membre cherche plusieurs chemins, et qu'il compare la QoS de chaque chemin pour choisir celle qui est la meilleure. Les paramètres de QoS considérés sont statiques. Par exemple, la bande passante, ou le délai des liens.

YAM est proposé tant intra-domaine comme inter-domaine. Les arbres inter-domaine construit par YAM sont partagés, et la racine est le routeur de sortie du système autonome de la source. Notons que YAM propose aussi un mécanisme d'élection de la racine en cas de défaut de la racine courante.

3.1.6 QoSMIC

QoSMIC [FBP98] (pour "Quality of Service sensitive Multicast Internet protoCol") partage avec YAM le désir de prendre en compte la QoS. QoSMIC construit des arbres partagés ou des arbres dédiés à une source. A différence de YAM, QoSMIC utilise des paramètres dynamiques, comme la bande passante disponible ou le délai produit par l'attente dans le tampon d'un lien.

L'agrégation de nouveaux clients se fait de la même manière que YAM, c'est-à-dire de façon gloutonne. Cela permet à QoSMIC d'opérer aussi bien intra qu'inter-domaine. De plus QoSMIC permet une deuxième méthode d'agrégation de nouveaux clients : la recherche à distance. Cette méthode est effectuée par un routeur spécifique (le manager) qui cherche les meilleurs candidats parmi les routeurs qui appartiennent à l'arbre. Ce sont les routeurs candidats qui envoient des messages vers le nouveau membre, collectant les paramètres de QoS correspondants au chemin connectant chaque candidat au nouveau membre. Le nouveau membre choisit alors le candidat auquel il se connecte. Cette méthode permet d'obtenir de très bonnes performances en inter-domaine car on évite la recherche locale (distance $1, 2, 3, \ldots$).

Remarque. QoSMIC utilise des arbres partagés, ce qui induit les problèmes classiques liés à un cœur (dont la sélection n'est pas le moindre).

3.1.7 MIP

MIP [PGLA97] (pour "Multicast Internet Protocol") a la propriété de créer des arbres multicast à partir des sources, et à partir des membres. Il supporte des arbres partagés et des arbres enracinés à la source. De plus, MIP permet la transformation des arbres, de partagés à non-partagés et vice-versa, de manière dynamique, selon la quantité de trafic des sources. Les arbres sont construits à partir d'une racine qui peut être soit une source, soit un membre du groupe. MIP maintient de plus un anneau entre la racine est ses voisins afin de remplacer la racine le cas échéant.

Remarque. Il a été démontré formellement que MIP ne génère pas de boucles, à aucun instant de l'exécution, et ceci même si les tables de routage point-à-point ont des boucles pendant un court instant. Malheureusement, la procédure de vérification peut générer une charge de trafic importante pendant la construction de l'arbre.

La construction des arbres initiés par les membres se fait en utilisant la technique gloutonne. Les arbres construits par les sources le sont selon les plus courts chemins. Dans ce cas on parle de "Shortest Path" par opposition à "Reverse Shortest Path" (RSP) parce que les chemins sont dans le sens direct, de la source vers les membres.

3.1.8 BGMP

BGMP [KTA+98] (pour "Border Gateway Multicast Protocol") est un protocole multicast basé sur le protocole point-à-point BGP (pour "Border Gateway Protocol"). Comme BGP, BGMP est un protocole inter-domaine. Il s'occupe de recueillir l'information des systèmes autonomes, et de respecter leurs contraintes. BGMP peut également faire l'attribution des adresses multipoint, c'est-à-dire la classe "**D**" d'adressage IP.

La construction des arbres BGMP est faite à partir de l'information des tableaux BGP, et elle utilise la technique RSP. BGMP construit un arbre partagé pour chaque groupe. Cet arbre est bidirectionnel. Le cœur est tout un système autonome complet, et donc tous les routeurs supportant BGP peuvent devenir potentiellement cœur. Cela simplifie l'élection du cœur et permet d'avoir le cœur dans un domaine où il y a des sources et des membres. Les tables BGMP contiennent les *contrats* entre les différents systèmes autonomes (les poids assignés aux liens sont généralement choisis par les administrateurs).

Remarque. La sélection des adresses multipoint est un problème complexe dans un réseau d'administration distribué. Le nombre de ces adresses est limité. BGMP utilise donc le protocole MASC (pour "Multicast Address-Set Claim") pour obtenir une adresse. MASC s'occupe de maintenir de façon hiérarchique et dynamique l'attribution d'adresses dans tous les domaines. La structure hiérarchique lui permet d'être capable de supporter les changements d'échelle (`Internet` est en constante croissance). La procédure pour attribuer une adresse est du type "écoutez et réclamez" avec détection des adresses déjà utilisées.

3.1.9 SM

SM [PBC+99] (pour "Simple Multicast") est un protocole conçu pour "être simple". Son objectif est de profiter des avantages de tous les autres protocoles pour servir intra et inter-domaine. L'architecture de ce protocole ressemble à celle de CBT. Il construit des arbres partagés avec agrégation des clients en utilisant les plus courts chemins (RSP). Les arbres sont bi-directionnels. Un problème soulevé par SM lors de son utilisation inter-domaine est l'attribution des adresses. SM propose d'identifier le groupe avec la paire $\{address\_of\_core, multicast\_address\}$. Cette identification permet d'une part la minimisation du temps pour obtenir une adresse multipoint, et offre d'autre part un nombre suffisant d'adresses multipoint grâce à l'agrandissement de l'espace d'adressage. La construction d'un arbre multipoint est simplifiée en évitant la demande d'une adresse unique de la classe "**D**" d'IP. Le nœud qui sert de cœur est connu de tous les membres (par exemple, ils peuvent l'obtenir d'une page Web).

Les auteurs de SM ont conçu une procédure d'installation compatible avec les autres protocoles de multicast. Cela est important pour pouvoir faire éventuellement une migration d'un protocole de multicast vers le protocole SM. Une autre caractéristique importante de SM est qu'il détecte l'apparition de boucles.

3.1.10 Comparaison des protocoles

Ce chapitre effectue une comparaison rapide des protocoles multipoint présentés dans les sections précédentes. La comparaison est focalisée principalement sur les méthodes de sélection des chemins pour construire l'arbre multipoint. La qualité des communications dépend en effet fortement des chemins que les paquets suivent. Nous avons retenu certains critères qui permettent d'étudier la construction des arbres multipoint. Ces critères sont les suivants.

Type de connexion. C'est-à-dire les différentes manières pour relier les sources et les membres d'un groupe. Il y en a principalement deux : les arbres partagés entre toutes les sources, et les arbres dédiés à chaque source.

Construction. C'est-à-dire les techniques de construction des arbres multipoint. Celles-ci sont principalement l'utilisation du protocole point-à-point sous-jacent ("unicast"), la recherche de chemins multiples, ou la recherche en largeur. Ces techniques opèrent de manières différentes, mais les résultats de l'agrégation des membres peuvent être assez semblables.

Agrégation des membres. C'est-à-dire la méthode pour ajouter un nouveau client qui arrive. Les méthodes typiques sont RSP (pour "Reverse Shortest Path") ou gloutonne. La technique RSP construit un plus court chemin du membre à la source. On notera SP (pour "Shortest Path") le cas d'un plus court chemin de la source vers un membre du groupe multipoint.

Qualité de service. C'est-à-dire la capacité des protocoles multipoint de pouvoir construire des arbres avec une qualité de service ("QoS") donnée.

	Connexion	Construction	Agrégation
DVMRP	Source-Dédié	Diffusion/Élagage	RSP
MOSPF	Source-Dédié	OSPF + Groupe	SP
CBT	Partagé	P.-à-P.	RSP
PIM-SM	Shrd & S-D	P.-à-P./Élagage	RSP
YAM	Partagé	Multiples chemins	Gloutonne
QoSMIC	Part. & S-D	Multiples chemins	Gloutonne
MIP	Part. & S-D	Recherche en largeur	SP & Gloutonne
BGMP	Part. & S-D	BGP	RSP
SM	Partagé	P.-à-P.	RSP

	QoS	Mise à jour	Boucles	Domaine
DVMRP	Non	Périodique	-	Intra
MOSPF	Non	Selon point-à-point	-	Intra
CBT	Non	No	-	Intra, Inter
PIM-SM	Non	Partagé → Source-Dédié	-	Intra, Inter
YAM	Oui	No	-	Intra, Inter
QoSMIC	Oui	Partagé → Source-Dédié	Non	Intra, Inter
MIP	Non	Selon point-à-point	Non	Intra, Inter
BGMP	Non	Selon point-à-point	-	Inter
SM	Non	Selon point-à-point	Non	Intra, Inter

TAB. 3.1: *Propriétés des protocoles multipoint*

Mise à jour. C'est-à-dire le contrôle pour maintenir les données utilisées par les communications multipoint. Le nombre de messages de contrôle peut être important et ils chargent le réseau. Nous ferons la distinction entre les messages qui s'occupent d'arranger l'arbre, et ceux qui s'occupent de transmettre les changements des conditions du réseau. Les premiers messages sont générés suite à un changement important du groupe. Les deuxièmes sont générés en relation avec la situation générale du réseau.

Boucles. C'est-à-dire l'aptitude à la prévention des boucles dans les protocoles de routage. Dans une situation stable, il n'y a pas de boucles dans aucun protocole. Cependant, il peut y en avoir pendant la construction des chemins. Il existe toutefois certains protocoles qui n'ont pas de boucles, à aucun moment de leur exécution

Domaine. C'est-à-dire le domaine d'opération des protocoles : intra-domaine ou inter-domaine.

Le bilan de la comparaison entre les protocoles multipoint est donné dans le tableau 3.1.

La plupart des protocoles ont été conçus pour permettre une connexion entre les membres, mais certains ne se préoccupent guère de savoir si les routes sélectionnées ont une bonne qualité. En effet, cette qualité dépend en fait souvent de l'application. Pour des applications temps réel, le but est de faire arriver les paquets dans un temps borné. Donc un taux élevé

de perte de paquets peut empêcher l'accomplissement de ce but. Pour des applications de transmission de données, le but est de les transférer le plus vite possible. Une perte de paquets importante implique des retransmissions, et par conséquent une diminution du débit. De plus, si les demandes de retransmission des paquets sont faites par les applications, cela augmente le temps de réception (mécanismes complexes à couche application). Il existe des protocoles qui offrent une couche transport assurant l'arrivée des messages à tous les membres d'un groupe multipoint (voir par exemple le article de Rhee, Balaguru et Roukuskas [RBR98]). Cela demande un minimum de qualité de service pour les applications multipoint. Dans le cas où le protocole multipoint utilisé est sans QoS, l'unique manière de fournir cette qualité minimum est d'augmenter les ressources, comme par exemple la capacité des liens.

Tous les protocoles qui n'ont pas de QoS se basent sur le protocole de routage point-à-point sous-jacent, sauf MIP. Les deux protocoles qui supportent la QoS sont YAM et QoSMIC. Ils utilisent la méthode gloutonne. Ils cherchent des routes pour connecter le nouveau nœud en comparant différents chemins qui le relient à l'arbre courant. Ces chemins sont trouvés en suivant les tables de routage point-à-point. Il y a donc une limitation dans le choix de la route. C'est un point que nous chercherons à améliorer dans cette thèse.

Un autre aspect important des protocoles est le type de connexion possible (arbres partagés ou dédiés à chaque source). Les arbres partagés permettent de minimiser le nombre d'états dans les routeurs pour le cas des groupes de grande taille ayant un grand nombre de sources. Mais à l'inverse, si une source envoie beaucoup de trafic, alors les arbres dédiés sont une meilleure solution. Les protocoles qui supportent les deux types d'arbres sont PIM-SM, QoSMIC et MIP.

La mise à jour de l'arbre engendre du trafic mais permet d'actualiser l'arbre en cas de modifications importantes. Les situations nécessitant une mise à jour de l'arbre sont principalement une modification des membres du groupe, l'apparition d'une source avec un trafic très élevé, et une modification des paramètres de QoS. Le plus important est que le trafic engendré dans l'arbre soit petit. Si l'apparition soudaine d'une source qui envoie beaucoup de trafic n'est pas très fréquente alors le trafic engendré par le trafic de contrôle reste faible.

Le domaine d'opérations des protocoles est un critère de comparaison important. Opérer au niveau des systèmes autonomes, c'est-à-dire de façon intra-domaine, limite le trafic engendré pour mettre en œuvre le protocole. C'est la raison pour laquelle DVMRP et MOSPF sont intra-domaine. Il n'existe qu'un seul protocole pour le routage point-à-point inter-domaine, le protocole BGP. Cela limite les choix pour les routes. Et bien sûr le protocole MBGP est naturellement le plus adapté à opérer à ce niveau.

En conclusion, le protocole multipoint idéal devrait supporter un minimum de QoS, offrir la capacité de chercher des chemins différents de ceux proposés par le protocole unicast sous-jacent, supporter à la fois les arbres partagés et dédiés, et opérer aux deux niveaux intra et inter-domaine, le tout en engendrant le moins de trafic de contrôle possible.

3.2 Minimisation du délai dans un arbre de multicast

Nous avons vu dans la section 3.1 que tous les protocoles de communication multipoint proposés jusqu'à ce jour sont basés sur un (ou plusieurs) arbre(s) couvrant des nœuds du groupe. L'arbre étant construit, les protocoles procèdent par diffusion de messages dans cet arbre. Le délai total d'une communication multipoint (c'est-à-dire le temps entre l'envoi d'un message par la racine de l'arbre et la réception de ce message par tous les membres du groupe) est un paramètre qu'il convient de minimiser. Dans cette section nous proposons un algorithme distribué pour l'ordonnancement de l'envoi des messages dans un arbre. Cet algorithme permet de minimiser le délai total d'un multicast dans l'arbre. Nous montrons que l'efficacité de cet algorithme dépend fortement du rapport entre le temps de commutation des nœuds et le temps de transmission des liens. Nous montrons qu'il existe des technologies de réseaux pour lesquelles cet algorithme permet d'améliorer significativement le délai d'une communication multipoint. Notre algorithme distribué est basé sur un algorithme centralisé proposé par Slater, Cockayne et Hedetniemi [SCH81]. Ce dernier algorithme est défini pour le modèle de communication "temps-constant 1-port" : chaque paquet est supposé prendre un temps unitaire pour transiter d'un nœud à un voisin, et un nœud ne peut envoyer un message qu'à au plus un voisin à chaque unité de temps. Notre algorithme généralise ces hypothèses pour pouvoir être appliqué dans un cadre plus réaliste. Il pourrait être mis en œuvre de façon aisée sous forme d'un protocole complémentaire des protocoles de multicast classiques.

3.2.1 Algorithme distribué d'ordonnancement des messages

Nous supposons donné un arbre de multipoint construit par un des protocoles décrits dans l'état de l'art de la section précédente. Nous nous concentrons sur l'envoi d'un message de la racine de l'arbre à tous les membres du groupe. Notons que si tous les membres du groupe appartiennent à l'arbre, il peut bien sûr y avoir d'autre nœuds dans l'arbre pour assurer sa connexion. Le message arrivera à tous les nœuds qui appartiennent à l'arbre. Nous utiliserons le modèle simplifié suivant. Un routeur est modélisé comme une file d'attente à une entrée et un processeur central qui utilise l'adresse de destination de chaque paquet pour choisir la file d'attente de sortie adéquate.

Nous utilisons le modèle suivant :

- Un lien e a un temps de *transmission* $t_{trans}(e)$. C'est-à-dire qu'un paquet partant de u au temps t, et transmis sur le lien $e = (u, v)$, arrivera en v au temps $t + t_{trans}(e)$. Le paramètre $t_{trans}(e)$ est la somme des temps d'attente dans la file d'attente du lien e et du temps de traversé du lien e.

- Un nœud v a un temps de *commutation* $t_{comm}(v)$. Plus précisément $t_{comm}(v)$ est le temps d'attente dans la file d'entrée plus le temps nécessaire au nœud v pour distribuer un paquet dans les files d'attente de sortie. Ainsi,un paquet arrivant en v au temps t quittera v sur le premier lien au temps $t + t_{comm}(v)$, sur le second lien au temps

$t + 2t_{comm}(v)$, etc. En général le paquet arrivera à l'autre extrémité du k-ème lien e incident à v au temps $t + k \cdot t_{comm}(v) + t_{trans}(e)$, et le nœud v est libre dès le temps $t + k \cdot t_{comm}(v)$.

Par exemple, le modèle "temps-constant 1-port" correspond à $t_{trans}(e) = 0$ pour tout lien e, et $t_{comm}(v) = 1$ pour tout nœud v. Le modèle "temps-constant all-port" correspond à $t_{trans}(e) = 1$ pour tout lien e, et $t_{comm}(v) = 0$ pour tout nœud v. Le rapport t_{trans}/t_{comm} sera le critère dominant de l'efficacité du protocole présenté dans cette section. Si t_{trans}/t_{comm} est grand alors notre protocole n'aura guère d'intérêt pratique. Par contre, si t_{trans}/t_{comm} reste raisonnablement faible, alors notre protocole offrira un avantage. Nous serons plus explicite sur les termes "raisonnablement faible" et "avantage" dans la suite du chapitre.

Notre protocole est nommé MSDA, à partir de la terminologie anglophone "Multicast Scheduling Distributed Algorithm". MDSA a deux composantes : la première consiste en l'algorithme d'ordonnancement des appels SA (pour "Scheduling Algorithm") ; la deuxième consiste en l'algorithme d'envoi des messages SMA (pour "Scheduled Multicast Algorithm"). SA est la version distribuée de l'algorithme de [SCH81].

La description de l'algorithme SA est donné dans la table 3.2.

Considérons un arbre $T = (V, E)$ où V est l'ensemble des nœuds (ou routeurs), et E est l'ensemble des liens de communication. Supposons que tous les nœuds de T soient capables d'exécuter un protocole multipoint. L'algorithme d'ordonnancement SA s'exécute en parallèle dans tous les nœuds de T. Chaque nœud $v \in V$ maintient une variable de temps t_v et une permutation σ_v qui décrit l'ordre dans lequel les communications aux fils doivent se faire. Un nœud déclenche l'exécution de l'algorithme d'ordonnancement à la réception d'un message *change* de son père ou d'un de ses fils. Un nœud quittant l'arbre envoie un message *change* à son père avant de quitter l'arbre. L'arrivée d'un nouveau nœud non déjà dans l'arbre (soit un nouveau membre, soit un nouveau nœud nécessaire à la connexion d'un nouveau membre) déclenche également l'envoi d'un message *change*. Un nœud quittant l'arbre est supposé feuille. En effet, nous ne traitons pas ici du crash d'un nœud, mais du retrait du nœud d'un groupe multipoint. Si un nœud interne se retire du groupe, il ne participe plus à l'application (par exemple réception d'une vidéo) mais reste présent dans l'arbre pour le relais des messages. Un nœud ne quitte l'arbre que s'il ne participe plus à l'application et si sa présence n'est plus nécessaire pour le relais des messages dans l'arbre. Un nœud ne peut donc quitter l'arbre que s'il est une feuille.

La fonction $Calc()$ s'occupe d'obtenir le temps $t(v)$ correspondant au nœud local v selon les temps de ses fils u_i pour $i \in [1, k]$. Plus précisément, soit t_i le temps d'une diffusion optimale à partir de u_i dans le sous-arbre T_i enraciné en u_i. La fonction $Calc()$ reçoit comme argument l'ensemble $\mathcal{T}$ des temps t_i, c'est-à-dire $\mathcal{T} = \{t_1, \ldots, t_k\}$. L'objectif de $Calc(\mathcal{T})$ est alors de calculer le temps optimal d'une diffusion à partir de v dans le sous-arbre enraciné en v. Soit $e_i = \{v, u_i\}$ le lien entre v et u_i, $i = 1, \ldots, k$. Soit σ une permutation de k éléments telle que

$$t_{trans}(e_{\sigma(i)} + t_{\sigma(i)}) \geq t_{trans}(e_{\sigma(i+1)} + t_{\sigma(i+1)}) \tag{3.1}$$

pour tout $1 \leq i < k$. La fonction calcule $Calc(\mathcal{T})$ comme le temps $t(v)$ tel que

Algorithme 3.1

1 *Initialisation :*
2 $t(v) \leftarrow 0$ et $\sigma_v \leftarrow Identite$;
3 Envoyer un message $(change, t(v))$ au père w;
4 *Attente des messages :*
5 **Message du fils u_i :** $(change, t(u_i))$
6 Faire $Calc(\mathcal{T})$ à l'instruction 17;
7 Si $t(v)$ a changé, alors envoyer un message $(change, t(v))$ au père w;
8 Retourner à l'instruction 4;
9 **Message du père w :** $(recalc)$
10 Envoyer un message $(recalc)$ à tout fils u_i;
11 Après réception de tous les messages $(change, t(u_i))$ des fils u_i faire $Calc(\mathcal{T})$ à l'instruction 17;
12 Envoyer un message $(change, t(v))$ au père w;
13 Retourner à l'instruction 4;
14 **Message de l'application en v :** $(change, -1)$
15 Si v n'a plus des fils u_i, envoyer un message $(change, -1)$ au père w et `stop`. Sinon, retourner à l'instruction 4;
16
17 $Calc(\mathcal{T})$: `/* calcule la permutation` σ `et le temps` t_v `*/`
18 Soit $\mathcal{T} = \{t(u_1), t(u_2), ..., t(u_k)\}$ l'ensemble des temps des k fils;
19 Soit σ une permutation sur $\mathcal{T}$ telle que

$$t(u_{\sigma(i)}) + t_{trans}(\{v, u_{\sigma(i)}\}) \geq t(u_{\sigma(i+1)}) + t_{trans}(\{v, u_{\sigma(i+1)}\});$$

20 $t(v) \leftarrow \max\limits_{1 \leq i \leq k} \big(i \cdot t_{comm}(v) + t_{trans}(\{v, u_{\sigma(i)}\}) + t(u_{\sigma(i)})\big)$.
21 Retourner $t(v)$ et la permutation σ ;

TAB. 3.2: *Algorithme d'ordonnancement des appels SA sur le nœud v*

$$t(v) = \max_{1 \leq i \leq k} (i \cdot t_{comm}(v) + t_{trans}(e_{\sigma(i)}) + t_{\sigma(i)}). \tag{3.2}$$

Le temps de diffusion $t(v)$ est obtenu en informant les fils de v dans l'ordre stipulé par σ, c'est-à-dire

$$v \text{ envoie le message à } u_i \text{ à la phase } \sigma^{-1}(i). \tag{3.3}$$

En d'autres termes, v appelle $u_{\sigma(1)}$ en premier, puis $u_{\sigma(2)}$, puis $u_{\sigma(3)}$, etc. Les appels se font dans l'ordre décroissant des sous-arbres qui nécessitent le plus d'étapes de diffusion en interne. Nous allons voir dans la suite que cette stratégie d'ordonnancement permet effectivement d'obtenir un protocole de diffusion optimal à partir de v dans son sous-arbre. Notre protocole construit le protocole de diffusion à partir de la racine dans tout l'arbre en commençant aux feuilles, et en remontant vers la racine. Aux feuilles, le temps de diffusion dans le sous-arbre est nul puisque le sous-arbre est vide. Toutes les feuilles sont donc initialisées à 0. On remonte

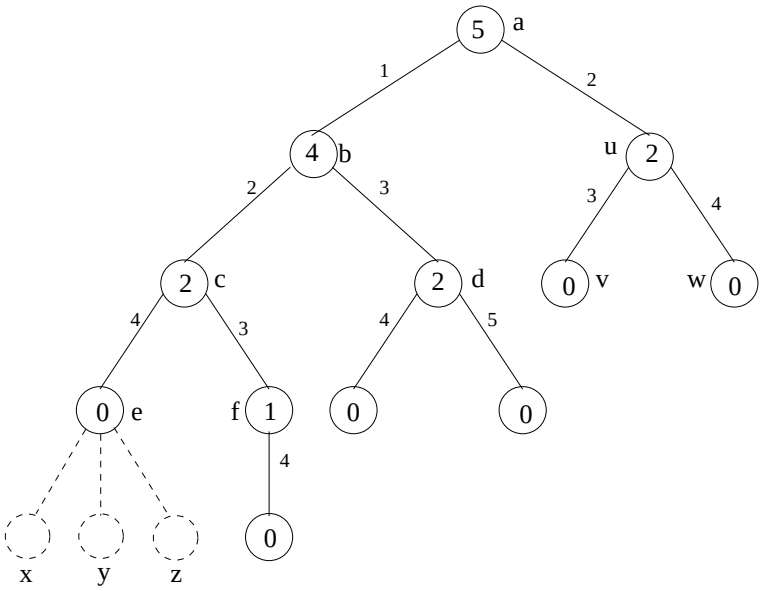

FIG. 3.1: *Un exemple d'exécution de SA*

dans l'arbre en appliquant le calcul de l'équation 3.2 à partir de la permutation σ vérifiant l'équation 3.1.

Exemple. La figure 3.1 montre un exemple d'exécution de l'algorithme SA pour un arbre de 12 nœuds où $t_{comm} = 1$ et $t_{trans} = 0$. Supposons d'abord que l'arbre est statique et concentrons nous sur la partie $Calc(\mathcal{T})$ à l'instruction 17 de SA. Toutes les feuilles v sont marquées $t_v = 0$. Le nœud f est l'unique nœud qui a un seul fils, donc $t_f = 1$ parce qu'envoyer un message à son sous-arbre prend une unité de temps. Le père c de f a deux fils. Par conséquent, il devra utiliser la formule de l'équation 3.2 de la table 3.2. Comme $t_f > t_e$, on a $t_1 = t_f = 1$ et $t_2 = t_e = 0$. Ainsi $t_c = \max\{t_f + 1, t_e + 2\} = 2$. Le temps total pour faire la diffusion dans le sous-arbre enraciné au nœud c est bien 2. La permutation $\sigma_c = \{f, e\}$ spécifie que le premier nœud à être informé est f, et le deuxième est e. On continue avec le nœud d qui a deux feuilles. Comme toutes les deux ont le même temps $t = 0$, on peut choisir l'ordre d'envoi arbitrairement et $t_d = 2$. Au nœud b, on a $t_b = \max\{t_c + 1, t_d + 2\} = 4$ (ici on peut encore choisir arbitrairement l'ordre car $t_c = t_d$). Finalement, en appliquant le calcul à tout l'arbre on obtient $t_a = 5$ qui est le temps minimal pour faire une diffusion dans l'arbre de la figure 3.1. Dans la figure 3.1, le temps auquel un nœud est informé est indiqué sur les arêtes entrantes.

Supposons maintenant que trois nœuds se connectent à l'arbre, dans l'ordre x, y et z. L'arrivée de x fait changer $t_e = 1$ et $t_c = 3$. Par contre, t_b ne change pas. Donc le temps de diffusion de l'arbre ne change pas. Si on ajoute y, $t_e = 2$ et $t_c = 3$, et t_b ne change toujours pas. Néanmoins la permutation $\sigma_c = \{e, f\}$ change afin de maintenir le temps de diffusion global $t_a = 5$. Finalement, lorsqu'on ajoute z, le temps de diffusion global augmente parce que $t_e = 3$, $t_c = 4$, $t_b = 5$, et $t_a = 6$. Remarquons qu'il est possible d'ajouter le nœud z dans

une autre partie de l'arbre sans que le temps de diffusion global n'augmente. Cependant SA s'occupe seulement de trouver le temps optimal de diffusion, et ne s'occupe pas d'optimiser l'agrégation des membres du groupe, ce qui relève du protocole de construction de l'arbre sous-jacent.

L'algorithme SMA qui s'occupe d'envoyer les messages est décrit dans le tableau 3.3.

Algorithme 3.2	
1	Pour chaque message multipoint reçu du père, faire :
2	Envoyer le message aux fils $u_1, \ldots, u_k$ selon l'ordre de la permutation σ_v obtenu par l'algorithme 3.2. C'est-à-dire envoyer à u_i à la phase $\sigma^{-1}(i)$

TAB. 3.3: *Algorithme SMA pour le nœud v*

Propriété 3.1 *L'algorithme MSDA satisfait les propriétés suivantes :*

1. *Le temps de multicast dans l'arbre est minimal.*
2. *La complexité local du calcul de t_v et de σ_v est $O(k \log k)$ pour un nœud v ayant k fils.*
3. *La complexité de l'espace de stockage local est $O(k)$.*

Preuve. Soit v un nœud d'un arbre T enraciné en r, et soit $u_1, \ldots, u_k$ ses k fils. Soit T_i le sous-arbre de T enraciné en u_i, et soit $e_i = \{v, u_i\}$. Soit $t_i = t_{trans}(e_i) + b(u_i, T_i)$. En d'autres termes, t_i est le temps $t_{trans}(e_i)$ de traverser le lien e_i, plus le temps $b(u_i, T_i)$ de diffusion optimale dans T_i à partir de u_i. Pour prouver la propriété 1, soit σ la permutation au nœud v, telle que pour tout $i < j$, $t_{\sigma(i)} \geq t_{\sigma(j)}$. Dans notre protocole d'ordonnancement des messages, u_i est informé par v à la phase $\sigma^{-1}(i)$, et reçoit le message au temps $\sigma^{-1}(i) \cdot t_{comm}(v) + t_{trans}(e_i)$. Le temps global de notre protocole d'ordonnancement est $\max_{1 \leq i \leq k}(i \cdot t_{comm}(v) + t_{\sigma(i)})$.

Supposons que l'ordonnancement optimal ne respecte pas l'ordre σ, mais une permutation $\beta \neq \sigma$, c'est-à-dire v informe $u_{\beta(i)}$ à la phase i.

Si β vérifie que, pour tout $i < j$, $t_{\beta(i)} \geq t_{\beta(j)}$, alors les temps de diffusion induits par les permutations σ et β sont identiques.

S'il existe une paire (i, j), $i < j$ et $t_{\beta(i)} < t_{\beta(j)}$, alors inversons les appels i et j de l'ordonnancement optimal, c'est-à-dire considérons les appels suivant la permutation β' où $\beta'(l) = \beta(l)$ pour $l \notin \{i, j\}$, $\beta'(i) = \beta(j)$, et $\beta'(j) = \beta(i)$. Soit $\tau = \max_{l \notin \{i,j\}}(l \cdot t_{comm}(v) + t_{\beta(l)})$. Le temps t de la diffusion suivant β est de

$$t = \max\{\tau,\ i \cdot t_{comm}(v) + t_{\beta(i)},\ j \cdot t_{comm}(v) + t_{\beta(j)}.\}$$

Le temps t' induit par la permutation β' est de

$$t' = \max\{\tau,\ i \cdot t_{comm}(v) + t_{\beta(j)},\ j \cdot t_{comm}(v) + t_{\beta(i)}.\}$$

Or $i \cdot t_{comm}(v) + t_{\beta(j)} < j \cdot t_{comm}(v) + t_{\beta(j)}$ puisque $i < j$. De même, $j \cdot t_{comm}(v) + t_{\beta(i)} < j \cdot t_{comm}(v) + t_{\beta(j)}$ puisque $t_{\beta(i)} < t_{\beta(j)}$. On a donc $t' \leq t$.

On peut donc construire comme ci-dessus une suite de permutations $\beta^{(0)}, \beta^{(1)}, \beta^{(2)}, \ldots$, avec $\beta^{(0)} = \beta$, et telle que le nombre de paires d'indices ne satisfaisant pas la propriété caractéristique de la permutation σ diminue strictement entre $\beta^{(i)}$ et $\beta^{(i+1)}$. Cette séquence vérifie donc $\beta^{(i_0)} = \sigma$ pour un $i_0 \geq 1$. Le temps $t^{(i)}$ induit par la permutation $\beta^{(i)}$ satisfait $t^{(i)} \leq t$ pour tout i. Le temps de la permutation $\sigma = \beta^{(i_0)}$ est donc au plus celui de la permutation optimale, et est donc optimal.

La propriété 2 est une conséquence du procédé d'ordonnancement de la liste des temps de chaque fils qui implique un tri.

La propriété 3 est dûe au stockage de la permutation σ_v qui a k éléments. ■

3.2.2 Performances du protocole MSDA

Le temps minimum pour diffuser un message de la racine r vers tous les nœuds d'un arbre T enraciné en r est noté $b(T, r)$. La propriété 3.1 stipule que MSDA s'exécute en temps $b(T, r)$. Cette section a pour objet de comparer l'ordonnancement optimal retourné par MSDA à un ordonnancement arbitraire (tel que celui effectué par les protocoles actuels). La mesure de performance est

$$\gamma = \frac{\overline{b}(T, r) - b(T, r)}{\overline{b}(T, r)} \tag{3.4}$$

où $\overline{b}(T, r)$ est la moyenne des temps de diffusion pour la racine r et l'arbre T pour tous les ordonnancements possibles. C'est-à-dire $\overline{b}(T, r) = \frac{1}{|S|} \sum_S b_s(T, r)$, où S est l'ensemble de tous les ordonnancements. Il faut noter qu'on peut aussi utiliser $b_{max}(T, r) = \max_S b_S(T, r)$, même si le nombre d'occurrences de $b_{max}(T, r)$ n'est pas nécessairement représentatif du cas typique.

$$\gamma_{max} = \frac{b_{max}(T, r) - b(T, r)}{b_{max}(T, r)} \tag{3.5}$$

La propriété suivante compare les paramètres $b(T, r)$, $\overline{b}(T, r)$ et $b_{max}(T, r)$.

Propriété 3.2 *Supposons $t_{trans} = 0$ et $t_{comm} = 1$. Pour tout arbre T enraciné en r :*

$$b(T, r) \leq \overline{b}(T, r) \leq b_{max}(T, r) \leq (b(T, r))^2.$$

De plus, il existe des arbres tels que $b(T, r) = b_{max}(T, r)$, et il existe des arbres tels que $\overline{b}(T, r) = \Theta((b(T, r))^2)$.

Preuve. Les inégalités $b(T, r) \leq \overline{b}(T, r) \leq b_{max}(T, r)$ sont triviales. Soit Δ le degré maximum de T enraciné en r, et D la profondeur de T. D'une part $b_{max}(T, r) \leq \Delta + (D-1) \cdot (\Delta - 1) \leq \Delta \cdot D$. D'autre $b(T, r) \geq \Delta$ et $b(T, r) \geq D$. Donc $b_{max}(T, r) \leq (b(T, r))^2$. Le chemin P_n de

n nœuds, avec r à une extrémité, satisfait $b(P_n, r) = b_{max}(P_n, r) = n - 1$, parce qu'il y a un seul ordonnancement pour faire la diffusion. Notons que, pour l'étoile S_n de n nœuds avec r au centre, l'équation $b(S_n, r) = b_{max}(S_n, r) = n - 1$ est aussi satisfaite, même s'il y a $(n-1)!$ manières différentes de faire la diffusion. Considérons finalement les arbres binomiaux d-dimensionnels définis par Vuillemin [Vui78]. B_0 est un seul nœud r. B_d est obtenu à partir de deux copies de B_{d-1} en ajoutant une arête entre les deux racines et en sélectionnant une des deux racines comme la nouvelle racine. Il est très simple de vérifier que le nombre de nœuds de B_d est $n = 2^d$, et que $b(B_d, r) = \log_2 n = d$. Soit un chemin $x_0, x_1, \ldots, x_d$ de longueur d à partir de la racine $x_0 = r$ jusqu'au nœud $s = x_d$. Le degré de x_i est $d-i+1$ pour $i \geq 1$. Le temps pour qu'un message arrive de x_i à x_{i+1} est $\frac{d-i}{2}$ en moyenne. Par conséquent on obtient $\overline{b}(B_d, r) = \sum_i \frac{d-i}{2} = \Theta(d^2) = \Theta((b(T, r))^2)$. ■

Le fait qu'il existe des arbres pour lesquels le temps de diffusion moyen est le carré du temps optimal montre à quel point il est nécessaire d'optimiser l'ordonnancement des messages de façon optimale.

Revenons dans le cas général. Notre modèle permet d'obtenir des résultats intermédiaires entre le modèle "1-port" (envoi séquentiel à chaque nœud) et le modèle "all-ports" (où les envois peuvent se faire en parallèle sur tous les ports). Comme nous l'avons déjà dit plus haut, lorsque $t_{trans} \ll t_{comm}$, on est dans le cas du modèle "1-port". Quand $t_{trans} \gg t_{comm}$ c'est le cas du modèle "all-ports" pour lequel l'algorithme d'ordonnancement n'a bien sûr pas d'objet. Nous étudions la performance totale de MSDA en fonction du rapport $\alpha = t_{trans}/t_{comm}$.

Nous avons fait des expériences dans des arbres multipoint obtenus par la technique RSP dans deux réseaux : UUNET[1] et une topologie générée par le modèle d'`Internet` que nous avons présenté dans la section 2.3. Dans les deux cas, les membres ont été tirés au hasard parmi tous les nœuds du graphe. Nous avons également mené des expériences dans l'arbre $A1$ de 10 nœuds représenté dans la figure 3.2. Cet arbre pourrait représenter l'interconnexion des membres d'un groupe multipoint dans un réseau LAN. Les trois arbres sont dessinés sur la figure 3.2. Dans nos expériences nous avons calculé les valeurs de $\overline{b}(T, r)$, $b(T, r)$, et $b_{max}(T, r)$ en fonction du rapport t_{trans}/t_{comm}.

Nous donnons deux représentations de nos résultats de simulation.

Dans un premier temps, nous présentons le paramètre γ de l'équation (3.4) en fonction de t_{trans}/t_{comm}, pour les trois arbres (cf. figure 3.3). Nous présentons également le paramètre γ_{max} de l'équation (3.5).

Tout d'abord on s'aperçoit qu'il existe une grande gamme de valeurs t_{trans}/t_{comm} pour lesquelles l'application du MSDA est utile. On constate ainsi sur la figure 3.3 que pour $t_{trans}/t_{comm} \simeq 4$ on obtient environ 10% d'amélioration du temps de multicast par rapport au cas moyen dans UUNET. $t_{trans}/t_{comm} \simeq 3$ donne approximativement la même amélioration dans notre modèle d'`Internet`. Bien sûr la comparaison avec le pire cas (cf. les courbes de γ_{max}) montre que MSDA peut être très performante. La probabilité d'ordonnancer les messages pour obtenir un temps de multicast b_{max} est cependant très faible.

Dans un deuxième temps, nous avons généré, pour chacun des réseaux UUNET et du modèle,

1. UUNET est le réseau d'un fournisseur d'accès à `Internet` (voir section 4.2.3).

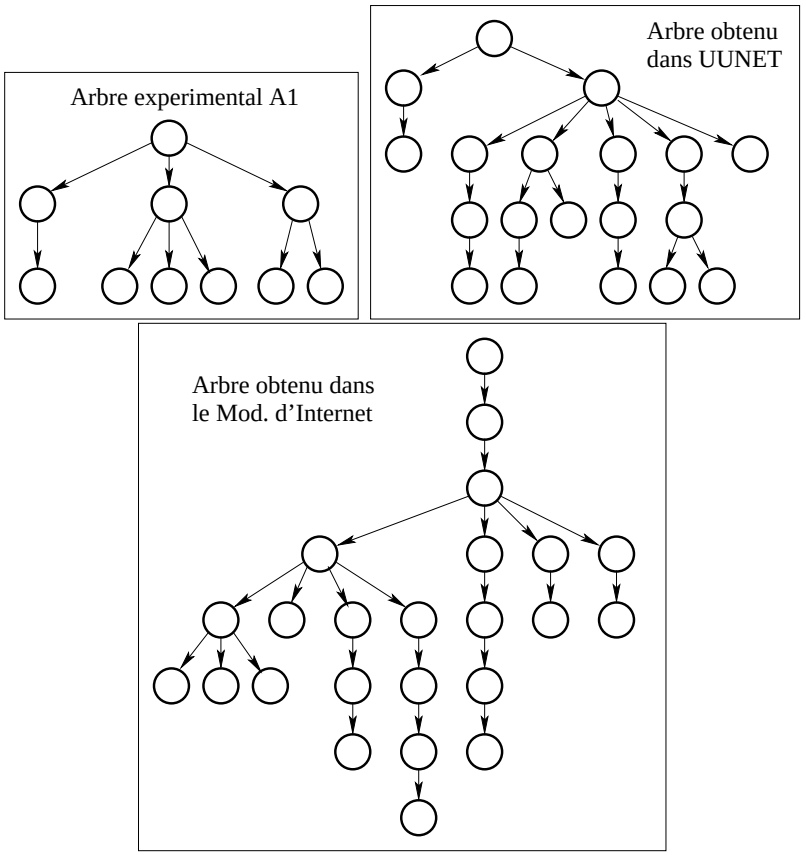

FIG. 3.2: *Arbres expérimentaux*

100 groupes multipoints tirés au hasard. Ces groupes ont 5, 10 ou 15 membres. Nous avons calculé la moyenne et la déviation standard de γ en fonction de t_{trans}/t_{comm}. Ces résultats sont montrés sur la figure 3.4.

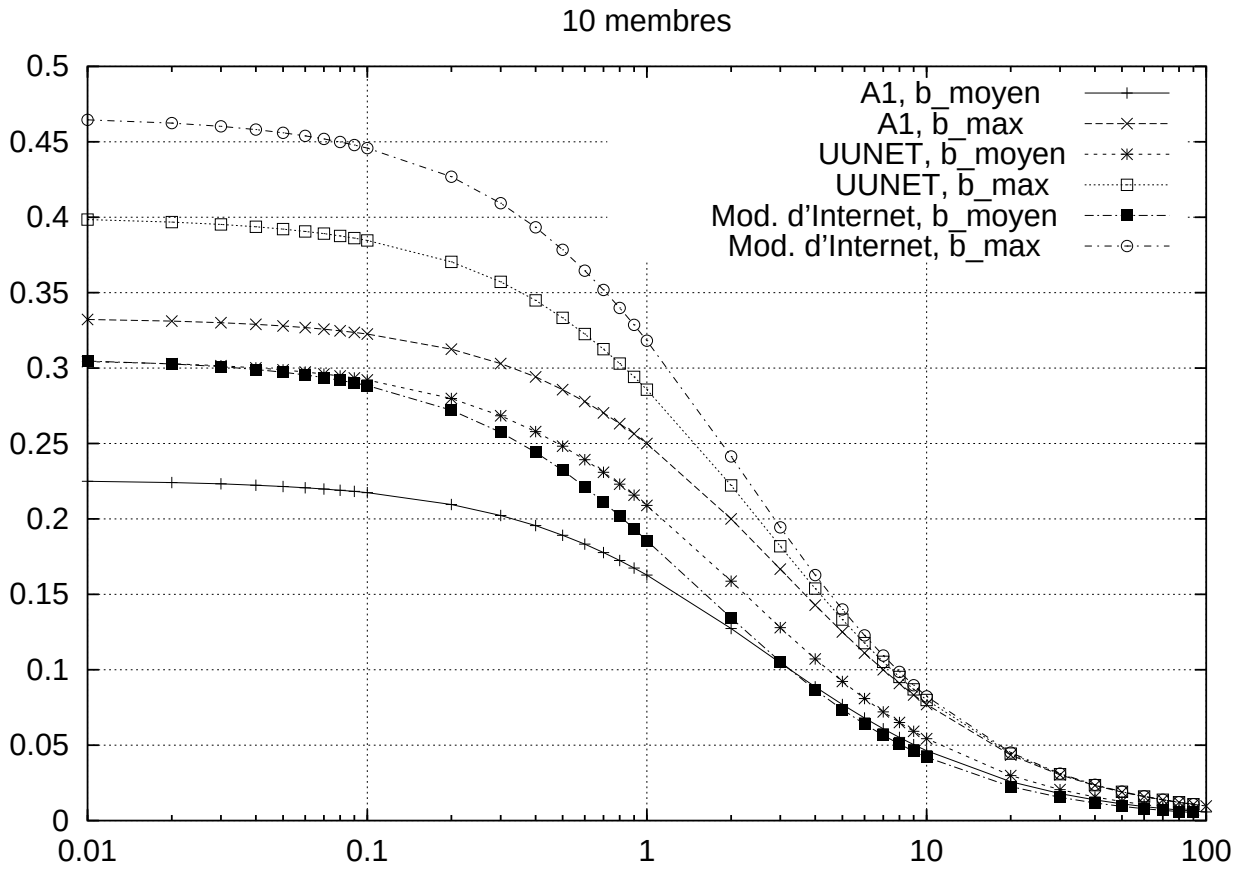

FIG. 3.3: γ *et* γ_{max} *en fonction de* t_{trans}/t_{comm}

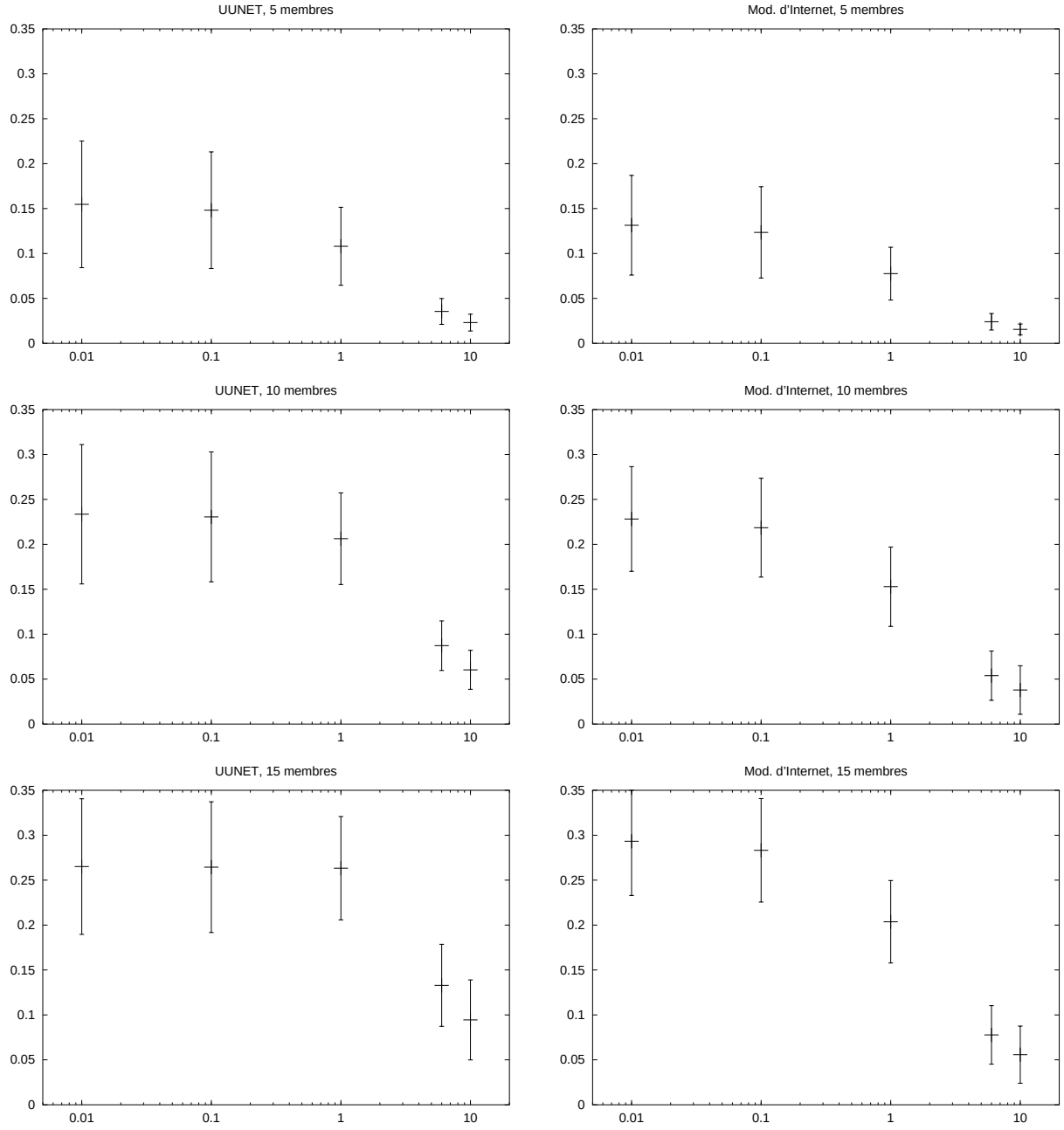

FIG. 3.4: γ *pour différentes tailles de groupes*

Dans la figure 3.4 on constate que lorsque le groupe est petit (5 membres) on obtient déjà de l'ordre de 10% d'amélioration pour des valeurs de t_{trans}/t_{comm} proches de 1. Pour des groupes aussi petit, l'avantage que MSDA procure s'atténue cependant très vite dès que t_{trans}/t_{comm} augmente. Par contre, pour les groupes ayant plus de membres, on constate une amélioration d'au moins 10% à partir de $t_{trans}/t_{comm} \simeq 6$.

Nous avons simulé des groupes d'au plus 15 membres à cause de la taille en mémoire nécessaire pour générer tous les cas possibles de diffusion. En effet, pour un sommet v de n fils, le nombre de cas est $N_{diff}(v) = n! \cdot \sum_{i=1}^{n} N_{diff}(i)$, où $N_{diff}(i)$ est le nombre de cas pour le fils i. Bien que nous n'ayons pas pu étudier des groupes plus grands, on s'aperçoit que l'amélioration augmente avec la taille du groupe. Donc il est attendu de meilleures performances pour des groupes ayant un grand nombre de participants.

Dans la section suivante, nous étudions les cadres possibles d'application de MSDA au vu des résultats de simulation précédents.

3.2.3 Cadres d'application du protocole MSDA

Nous avons montré dans la section précédente que, dans les réseaux où le rapport t_{trans}/t_{comm} est petit, l'amélioration produite par MSDA est significative (au moins 10%). Pour que l'application de MSDA soit rentable, il faut, d'après nos simulations, que le réseau vérifie $t_{trans} \leq 6 \cdot t_{comm}$. Or, il existe des exemples de sous-réseaux formant `Internet` où le temps de transmission est faible, grâce à des temps de propagation faibles et des capacités de liens élevées. C'est typiquement le cas des réseaux LANs et des réseaux de campus. MSDA peut trouver également une application aussi bien au niveau de la couche réseau (par exemple IP, ATM) qu'au niveau de la couche liaison (par exemple les LANs). Dans la suite, nous détaillons ces cadres d'application.

Les LANs

On trouve dans la couche liaison les protocoles pour LAN : Ethernet, Fast-Ethernet, Gigabit-Ethernet, FDDI, etc. Actuellement les composants les plus utilisés pour construire ces réseaux sont des commutateurs ("LAN switches"). Ces commutateurs permettent non seulement d'avoir plusieurs paires d'hôtes en communication simultanée, mais supportent également les communications multipoint. C'est un cadre possible d'application de MSDA. En effet, dans la technologie Ethernet par exemple, le délai maximum entre deux stations est de $t_{eth-\max} = 51.2\mu s$. Ce temps est induit par la méthode d'accès au niveau physique de transmission : CSMA/CD ("Carrier Sense Multiple Access/Collision Detect"). Cette méthode nécessite, pour qu'une station a puisse transmettre, qu'il n'y ait pas d'autres stations en transmission car le support de la transmission est partagé. La station a commence donc à transmettre en écoutant si une autre station veut transmettre. Dès que le temps $t_{eth-\max}$ est passé, il ne peut pas y avoir de station en transmission en dehors de a.

Initialement, les composants utilisés pour construire un LAN étaient les hubs, c'est-à-dire des répéteurs. Pour un support physique UTP ("Unshielded Twisted Pair"), la norme permet

jusqu'à 5 segments de $100m$ connectés avec quatre répéteurs entre ces segments du LAN. Si on utilise des commutateurs entre les segments du LAN, ces derniers peuvent soit stocker la trame d'information et recommencer le processus de transmission avec CSMA/CD dans le segment suivant, soit renvoyer la trame d'information tout de suite comme un répéteur. Dans le second cas, cet envoi est néanmoins dirigé uniquement vers la sortie correspondant à la destination. La première technique est appelée en anglais "store and forward"; la deuxième "cut through".

Dans le premier cas (store and forward) il n'y a pas de contrainte sur le nombre de commutateurs car, à chaque étape, le processus de demande d'accès au canal de transmission est recommencé sans tenir compte du délai total de la transmission. Dans le second cas (cut through), le système se comporte comme un réseau avec répéteurs entre les deux stations concernées, et donc il doit respecter le délai $t_{eth-\max}$. Nous allons nous intéresser à ce second cas. Le temps pour traverser la distance maximum d'un segment est de $t_{trans} = 100/C' \simeq 0.5\mu s$, où C' est la vitesse de propagation dans le cuivre[2]. Or le temps maximum de transit par tous les segments de câble est de $t_{trans-500m} = 500m/C' \simeq 2.5\mu s$. Si on compare ce temps avec le temps maximum $t_{eth-\max}$ d'Ethernet, on obtient $t^{max}_{comm} = 51.2\mu s - 2.5\mu s$ pour quatre commutateurs, soit un maximum de $12.175\mu s$ par commutateur pour la commutation. Dans ce cas, on a donc $t^{max}_{comm} > t_{trans}$. Pour Fast-Ethernet, le temps maximum est de $t_{eth-\max} = 5.12\mu s$ et il y a des technologies qui permettent 3 ou 4 commutateurs. On trouve donc un temps $t^{max}_{comm} = 0.655\mu s$ pour quatre commutateurs, et $t^{max}_{comm} = 1.04\mu s$ pour trois commutateurs avec 4 segments, tandis que le temps de transmission est toujours $t_{trans} \simeq 0.5\mu s$. Dans ce cas, on a donc $t_{trans} \simeq \frac{1}{2} t^{max}_{comm}$. Dans tous ces cas, MSDA peut être utilisé pour la phase d'initiation de la transmission de la trame (phase de détection de collisions), car on peut arriver à tous les utilisateurs en avance s'il n'y a pas de collisions (sinon il faudra définir la politique à suivre).

Les réseaux ATM

La technologie de commutation de paquets ATM opère au niveau réseau. Normalement ATM est utilisé comme support pour d'autres technologies, comme IP ou même pour des réseaux téléphoniques. ATM permet d'envoyer des paquets sur des canaux virtuels créés pour les applications, c'est-à-dire les couches qui opèrent au dessus d'ATM. Sous ce protocole, il est possible de trouver $t_{trans} < 6 \cdot t_{comm}$. En effet, l'unité de transmission est une cellule de 53 octets. Par conséquent on aura un scénario d'application pour le MSDA dès que les liens sont suffisamment courts, induisant un temps de transmission petit. Ce sera typiquement le cas à l'intérieur d'un même bâtiment.

Les réseaux IP

Dans les réseaux IP, le matériel utilisé consiste en un assemblage de routeurs. Un routeur est un dispositif capable d'envoyer des paquets de données vers leurs destinations. La destination

2. Cette vitesse est approximativement 2/3 de la vitesse de la lumière C dans le vide.

d'un paquet est stipulée dans l'entête. Un routeur décide sur quelle sortie renvoyer le paquet selon sa table de routage. Il place le paquet dans la file d'attente correspondant à cette sortie, et peut s'occuper d'autres paquets. Il n'y a alors aucune contrainte sur le temps maximum de commutation (même s'il est désirable que cette opération soit rapide). Un routeur peut ainsi prendre un temps de commutation t_{comm} élevé en comparaison de celui d'Ethernet. En effet le temps de recherche dans les tables de routage est plus élevé que pour les commutateurs des LANs. Comme t_{trans} peut rester petit, par exemple si les liens ne sont pas très longs et si leur capacité est élevée, on obtient potentiellement des cas où l'inégalité $t_{trans} < 6 \cdot t_{comm}$ est vérifiée.

3.3 Un nouveau protocole de multicast

L'objectif de cette section est de montrer comment construire un arbre multipoint minimisant un paramètre donné dans un réseau de paquets. Ce paramètre peut représenter différentes caractéristiques selon le contexte. Ce peut être par exemple : la bande passante, le délai, la congestion, le taux de pertes, etc. Dans le cas du routage point-à-point, la méthode usuelle de construction des tables de routage utilise des variantes des algorithmes de Dijkstra et de Ford-Fulkerson. Le poids des liens est fixé à la valeur d'un paramètre à minimiser. Cette résolution fonctionne parfaitement dans le cas point-à-point car le problème de routage se réduit, pour chaque nœud, à obtenir un arbre couvrant du graphe enraciné en ce nœud. Les communications point-à-point de même source n'interagissent que faiblement. Simplement on assure qu'un paquet va suivre un chemin qui minimise le paramètre donné pour arriver à sa destination. A l'inverse, dans le cadre de communications multipoint, il y a toujours une relation qui lie les trafics d'une même source puisqu'on envoie le même paquet à tous les membres d'un groupe donné. Il y a donc une corrélation directe entre les trafics vers des destinations distinctes. La solution de l'algorithme de Dijkstra n'est efficace que dans le cas où les trafics qui sortent de la source vers toutes les destinations n'ont pas de corrélation mutuelle directe. La minimisation d'un paramètre donné dans le cas où le trafic est corrélé nous a donc poussé à adopter un autre point de vue.

Nous avons séparé le problème en deux parties. Nous décrivons d'abord un nouveau protocole qui permet la construction de tables point-à-point proposant plusieurs alternatives pour le choix de la route. Ces tables prennent en compte seulement la topologie du réseau (distance en nombre de sauts). Nous décrivons ensuite un nouveau protocole de multicast tirant partie des différentes alternatives proposées pour le routage point-à-point afin de construire un arbre de multicast optimisé en fonction d'un paramètre de QoS donné.

Afin d'estimer les performances de ce nouveau protocole de multicast, nous avons effectué des simulations sur deux types de topologies : des topologies correspondant à des réseaux réels, et des topologies générées selon le modèle de la section 2.3. Dans ces simulations nous comparons notre protocole multipoint avec certains protocoles existants. L'enseignement principal retiré de ces simulations est que la construction d'un arbre de multicast peut être significativement améliorée au prix d'un accroissement relativement faible du trafic de contrôle. Nous verrons en effet que nos protocoles sont paramètrés et que les simulations montrent qu'il est possible

de choisir les paramètres afin d'obtenir un compromis raisonnable entre performances de l'arbre d'une part, et trafic de contrôle d'autre part.

3.3.1 Un protocole de routage à chemins multiples

Cette section décrit un nouveau protocole distribué pour maintenir des tables de routage "multiples" dans tous les nœuds du réseau. Ce protocole est appelé MPDR à partir de la terminologie anglaise *Multi-Paths Distance Routing protocol*. Pour chaque destination, MPDR maintient en tout nœud une route passant par chaque voisin du nœud. Les chemins multiples dans les tables de routage permettent de considérer des chemins qui ne sont pas nécessairement les plus courts. Le fait que les protocoles de routage se limitent à chercher un plus court chemin limite la liberté de construction de l'arbre de multicast optimisé. En effet, si la valeur des poids sur lequel l'algorithme de Dijkstra se base pour la construction des tables (par exemple, la congestion des liens) change dynamiquement, toutes les tables de routage doivent changer. Dans notre proposition, les tables ne gardent que les données topologiques, en permettant qu'un autre protocole s'occupe de trouver les routes les plus appropriées.

Spécification : MPDR construit pour chaque nœud v du graphe une liste $\mathcal{R}$ dont les éléments sont des triplets $\{x, d, i\}$, où x est la destination, d est la longueur d'un plus court chemin entre v et x passant par le voisin u_i de v, et i est le numéro du lien qui relie v à u_i.

La description formelle de l'algorithme suivi par MPDR est donnée dans la table 3.4, où le nœud courant est noté v, et ses voisins u_i.

Notation. On définit la fonction $\mathcal{D}[i, x](\mathcal{R})$ qui donne la distance d correspondant à la paire (x, i) dans la liste $\mathcal{R}$. MPDR utilise également en chaque sommet v une liste $\mathcal{R}'$ dont les éléments sont les paires $\{x, d\}$, où x est la destination, et d est la longueur d'un plus court chemin entre v et x. On définit la fonction correspondante $\mathcal{D}'[x](\mathcal{R}')$ qui donne la distance $d = d(v, x)$ correspondant à x dans $\mathcal{R}'$. On utilise la notation suivante pour les opérations sur les listes. $\mathcal{R} \leftarrow \{\mathcal{R}, l\}$ est l'agrégation de l'élément $l = \{x, d, i\}$ à la liste $\mathcal{R}$. $\mathcal{R}[x]$ fait référence à l'élément de $\mathcal{R}$ qui a le nœud x comme destination. Finalement $\mathcal{R}[x] \leftarrow \{x, d, i\}$ modifie les valeurs courantes de distance pour la destination x en passant par le voisin u_i stockées dans la liste $\mathcal{R}$. La notation est la même pour la liste $\mathcal{R}'$.

MPDR est composé de deux parties. La première est une étape d'initialisation qui s'occupe de créer les listes $\mathcal{R}$ et $\mathcal{R}'$, et de prévenir les voisins par un message de mise à jour (instructions 1 à 5). La deuxième partie est une boucle infinie qui reçoit et traite les messages (instructions 6 à 23).

Il y a deux types de messages :

1. **Nouveau nœud.** Il annonce qu'un nouveau nœud v arrive dans le réseau. Il est supposé envoyé par un nœud v à tous ses voisins u_i. Ce message a la forme $(\mathsf{N}, \mathcal{R}'_v, v)$, où N est

l'indicateur de message ("N" pour "New"), v est le nœud émetteur, et $\mathcal{R}'_v$ est la liste $\mathcal{R}'$ de v. Avec ce message, le nœud v peut construire ses tables de routage puisque ses voisins lui enverront leur liste $\mathcal{R}'_{u_i}$.

2. **Mise à jour.** Ce message a pour rôle d'annoncer les changements de distances minimums, soit parce qu'elles ont diminué, soit parce qu'elles ont augmenté. Ce message a la forme $(\mathsf{U}, \mathcal{R}'_{u_i}, u_i)$, où U est l'indicateur de message ("U" pour "update"), $\mathcal{R}'_{u_i}$ est la liste $\mathcal{R}'$ de l'émetteur, et u_i est le nœud émetteur, voisin du récepteur.

Lorsqu'un nœud v se réveille (instruction 1), il crée ses listes $\mathcal{R}$ et $\mathcal{R}'$ vides, ajoute les voisins u_i dont il a connaissance[3] à la liste $\mathcal{R}$, et fait le calcul de $\mathcal{R}' \leftarrow F(\mathcal{R})$. La fonction $F(\cdot)$ extrait la longueur du plus court chemin de v vers toutes les destinations x. Finalement le nœud v envoie un message $(\mathsf{N}, \mathcal{R}'_v, v)$ à tous ses voisins et commence l'attente de messages (instruction 6). (Les voisins de v qui reçoivent le message $(\mathsf{N}, \mathcal{R}', v)$, lui répondent avec un message de mise à jour $(\mathsf{U}, \mathcal{R}'_{u_i}, u_i)$).

Dans les instructions $7-23$, l'algorithme s'occupe de traiter l'information qui arrive pour modifier ses données internes et pour signaler éventuellement des modifications à prendre en compte par les autres sommets. Cette étape se sert de la fonction $Calc(\mathcal{R}, \mathcal{R}'_{u_i}, u_i)$ qui ajoute des modifications à la liste $\mathcal{R}$ quand cela est nécessaire. Cette fonction utilise également la fonction $F(\mathcal{R})$ pour calculer la nouvelle liste $\mathcal{R}_{actuel}$. Si la liste $\mathcal{R}'_{actuel}$ diffère de $\mathcal{R}'$ il faudra envoyer une liste $\mathcal{R}'_\Delta$, de même format que $\mathcal{R}'$, à tous les autres voisins u_j, $j \neq i$ (cf. table 3.4).

Nous concluons la description du protocole MPDR par quelques remarques.

– MPDR n'annonce que les changements dans la liste $\mathcal{R}'$ (celle des plus courts chemins). Cette connaissance suffit à la construction des listes $\mathcal{R}$ (celles de plus courts chemins en passant par chaque voisin). Cette méthode nous permet de réduire le trafic généré par les messages de contrôle.

– L'algorithme que nous avons décrit n'est conçu que pour les nœuds qui se réveillent. En pratique il leur faudra également surveiller leurs voisins pour vérifier qu'ils sont en fonctionnement. Il y a plusieurs manières de surveiller ses voisins. Par exemple en envoyant régulièrement un "ping". S'il n'y a pas de réponse le nœud est considéré comme disparu. Nous ne traitons pas ici des détails (certes importants) de cette partie de l'implémentation, principalement parce qu'ils dépendent fortement des technologies utilisés.

– Dans le cas où un nœud z disparaît, tous ses voisins peuvent résoudre le problème en émulant la réception d'un message $(\mathsf{U}, \mathcal{R}'_z, z)$ avec $\mathcal{R}'_z[x] \leftarrow \{x, \infty\}$ pour tous les nœuds x.

3.3.2 Le protocole de multicast MCT

Dans cette section, nous introduisons un nouveau protocole distribué pour la construction d'un arbre dédié aux communications multipoint. Cet arbre a principalement pour but de mi-

3. Cette connaissance dépend du type de lien. Pour les liens point-à-point les adresses des voisins sont usuellement connues. Par contre, dans le cas où les voisins sont connectés par un réseau LAN, il faut mettre en œuvre un système de recherche des voisins.

Algorithme 3.3

1 *Initialisation :* /* le nœud local est appelé v */
2 $\mathcal{R}' \leftarrow \emptyset$; $\mathcal{R} \leftarrow \emptyset$;
3 Pour chaque voisin u_i faire $\mathcal{R} \leftarrow \{\mathcal{R}, \{u_i, 1, i\}\}$;
4 $\mathcal{R}' \leftarrow F(\mathcal{R})$; /* cf. instruction 33 */
5 Envoyer le message $(\mathsf{N}, \mathcal{R}', v)$ à tous les voisins;
6 *Attente de messages :* /* boucle infinie, attente des messages des voisins */
7 **Nouveau nœud :** $(\mathsf{N}, \mathcal{R}'_{u_i}, u_i)$ est reçu du voisin u_i;
8 Envoyer à u_i le message : $(\mathsf{U}, \mathcal{R}', v)$;
9 $\mathcal{R} \leftarrow Calc(\mathcal{R}, \mathcal{R}'_{u_i}, u_i)$; /* cf. instruction 25 */
10 $\mathcal{R}'_{actuel} \leftarrow F(\mathcal{R})$; /* cf. instruction 33 */
11 Si $\mathcal{R}'_{actuel} \neq \mathcal{R}'$;
12 $\mathcal{R}'_{\Delta} \leftarrow \{\mathcal{R}'_{actuel}[y] \ / \ \forall y \ : \ \mathcal{D}'[y](\mathcal{R}') \neq \mathcal{D}'[y](\mathcal{R}'_{actuel})\}$; /* cf. instruction 39 */
13 $\mathcal{R}' \leftarrow \mathcal{R}'_{actuel}$;
14 Envoyer à tout $u_j \neq u_i$ le message : $(\mathsf{U}, \mathcal{R}'_{\Delta}, v)$
15 Retourner à *Attente de messages.* /* cf. instruction 6 */
16 **Mise à jour :** $(\mathsf{U}, \mathcal{R}'_{u_i}, u_i)$ est reçu du voisin u_i;
17 $\mathcal{R} \leftarrow Calc(\mathcal{R}, \mathcal{R}'_{u_i}, u_i)$; /* cf. instruction 25 */
18 $\mathcal{R}'_{actuel} \leftarrow F(\mathcal{R})$; /* cf. instruction 33 */
19 Si $\mathcal{R}'_{actuel} \neq \mathcal{R}'$;
20 $\mathcal{R}'_{\Delta} \leftarrow \{\mathcal{R}'_{actuel}[y] \ / \ \forall y \ : \ \mathcal{D}'[y](\mathcal{R}') \neq \mathcal{D}'[y](\mathcal{R}'_{actuel})\}$; /* cf. instruction 39 */
21 $\mathcal{R}' \leftarrow \mathcal{R}'_{actuel}$;
22 Envoyer à tout $u_j \neq u_i$ le message : $(\mathsf{U}, \mathcal{R}'_{\Delta}, v)$
23 Retourner à *Attente de messages.* /* cf. instruction 6 */.
24
25 $Calc(\mathcal{R}, \mathcal{R}'_{u_i}, u_i)$: /* calcule la table $\mathcal{R}$ en fonction de l'ancienne $\mathcal{R}$, la table $\mathcal{R}'_{u_i}$ reçue du voisin u_i */
26 $\mathcal{R}_{actuel} \leftarrow \mathcal{R}$
27 Pour tout $x \in \mathcal{R}'_{u_i}$, faire
28 $\mathcal{R}_{actuel}[i, x] \leftarrow \{x, \mathcal{D}'[x](\mathcal{R}'_{u_i}), i\}$; /* cf. instruction 39 */
29 Pour tout $x \in \mathcal{R}'_{u_i}$, si $\mathcal{D}'[x](\mathcal{R}'_{u_i}) + 1 < \mathcal{D}[j, x](\mathcal{R})$ pour tout j, faire
30 $\mathcal{R}_{actuel}[i, x] \leftarrow \{x, \mathcal{D}'[x](\mathcal{R}'_{w_i}) + 1, i\}$; /* cf. instruction 39 */
31 Retourner $\mathcal{R}_{actuel}$.
32
33 $F(\mathcal{R})$: /* calcule la table $\mathcal{R}'$ en fonction de la table $\mathcal{R}$ */
34 $\mathcal{R}'_{actuel} \leftarrow \emptyset$;
35 Pour tout $x \in \mathcal{R}$ faire
36 $\mathcal{R}'_{actuel}[x] \leftarrow \{x, \min_i(\mathcal{D}[i, x](\mathcal{R}))\}$; /* cf. instruction 42 */
37 Retourner $\mathcal{R}'_{actuel}$.
38
39 $\mathcal{D}'[x](\mathcal{R}')$: /* calcule la distance $d(v, x)$ */
40 Retourner d, où d est tel que $\mathcal{R}'[x] = \{x, d\}$.
41
42 $\mathcal{D}[i, x](\mathcal{R})$: /* calcule la distance $d(v, x)$ en passant pour le voisin i */
43 Retourner d, où d est tel que $\mathcal{R}[x] = \{x, d, i\}$.

TAB. 3.4: *Algorithme du protocole MPDR*

nimiser un paramètre λ quelconque (par exemple une forme de "congestion"). Ce paramètre sera simplement supposé satisfaire certaines relations spécifiques (cf. les équations 3.6 et 3.7 ci-après). Ce nouveau protocole est appelé MCT, à partir de la terminologie anglaise "Minimum Congestion Tree". MCT utilise deux paramètres ϱ et r pour l'exploration du réseau. Ces paramètres, fixés *a priori*, contrôlent l'ensemble des arbres que MCT considère, et le protocole trouve l'arbre optimal dans cet ensemble. Bien sûr, MCT doit être mis en œuvre dans tous les nœuds du réseau afin de trouver l'arbre optimal, ou sinon il trouvera seulement l'arbre optimal pour le sous-ensemble de nœuds supportant MCT. Ce dernier cas peut en particulier se produire lors d'une phase d'installation de MCT dans le réseau.

Spécification : MCT construit un arbre de multicast enraciné en s tel que, pour tout membre x du graphe, le chemin de s à x dans l'arbre a un λ minimum parmi tous les chemins de s à x de longueur au plus $\varrho \cdot d(s,x) + r$.

MCT utilise le protocole sous-jacent MPDR défini dans la section 3.3.1, et se base sur les informations contenues dans les listes $\mathcal{R}$ et $\mathcal{R}'$. Pour simplicité, dans cette section on utilisera la notation $\mathcal{D}[i,x]$ et $\mathcal{D}'[x]$ respectivement, en omettant les listes $\mathcal{R}$ et $\mathcal{R}'$.

Le protocole MCT est divisé en trois phases :

- La *phase d'exploration* commence à la racine s de l'arbre. Cette racine opère comme le nœud *cœur* (ou *rendez-vous*) des autres protocoles multipoint. Elle reçoit la liste des membres du groupe multipoint m. Elle initialise alors une exploration qui a pour objet de visiter, pour chaque membre $x \in m$, tous les chemins de longueur au plus $\varrho \cdot d(s,x) + r$ entre s et x, où $d(.,.)$ est la distance (mesurée en nombre de sauts) entre deux nœuds, et ϱ et r sont les deux paramètres mentionnés précédemment.

- La *phase de sélection* consiste à choisir, parmi tous les chemins obtenus lors de la première phase, celui qui minimise le paramètre donné λ entre s et chaque membre du groupe m.

- Finalement, la *phase de construction* a pour objet de mettre à jour les tables de routage pour le groupe multipoint m, de manière à ce que tous les paquets relatifs à m puissent suivre les chemins minimaux (au sens du paramètre λ).

La description formelle de l'algorithme suivi par MCT est donnée dans la table 3.5 où le nœud courant est noté v, et ses voisins u_i. Le sommet w est le père de v. Le sommet s est la racine de l'arbre. L est une liste de k éléments, et $L[i]$ est le i-ème élément de la liste L, $i \in \{1,\ldots,k\}$. $L \leftarrow \{L, y\}$ indique l'ajout d'un nouvel élément y dans la liste L. De même, $\{L', y\} \leftarrow L$ indique le retrait du dernier élément y de la liste L, et le stockage de $L - y$ dans la liste L'.

Le paramètre à minimiser correspondant au lien e est noté $\lambda(e)$.

Pour un chemin $P = (e_1, e_2, \ldots, e_k)$, le paramètre correspondant est défini suivant deux versions possibles :

$$\lambda(P) = \begin{cases} \sum_{i=1}^{k} \lambda(e_i) & \text{version (a)} \\ \max_i \{\lambda(e_i)\} & \text{version (b)} \end{cases} \tag{3.6}$$

Par exemple, dans le cas $\lambda = \{\texttt{délai}\}$, la version (a) est la plus appropriée. Pour $\lambda = 1/\{\texttt{bande passante libre}\}$, la version (b) est la plus appropriée.

Lorsque deux chemins P_1 et P_2 sont considérés pour aller de x à y, MCT sélectionne celui dont le paramètre λ est le plus petit, et retournera $P \in \{P_1, P_2\}$ tel que :

$$\lambda(P) = \min\{\lambda(P_1), \lambda(P_2)\}. \tag{3.7}$$

Finalement, si on a un arbre T, la valeur du paramètre à minimiser dans cet arbre est noté $\lambda(T)$. Sa définition dépend de la version considérée :

– Dans la version (a), le paramètre $\lambda(T)$ est calculé à la racine r de l'arbre T comme

$$\max_i \lambda(T_i) + \lambda(e_i)$$

où T_i est le sous-arbre de T enraciné en le i-ème fils u_i de r, et $e_i = (r, u_i)$. On vérifie que la valeur pour un chemin simple est bien la somme des $\lambda(e_i)$ correspondants aux liens e_i du chemin. Cette variante correspond par exemple à un calcul de délais.

– Dans la version (b), le paramètre λ_T est calculé à la racine de l'arbre comme

$$\max_i \max\{\lambda(T_i) + \lambda(e_i)\}$$

On vérifie que la valeur d'un chemin est le maximum des $\lambda(e_i)$ correspondant aux liens e_i du chemin. Cette variante correspond par exemple à un calcul de bande passante libre.

Les équations (3.6) et (3.7) prennent en compte tous les cas standards pour lesquels il est nécessaire de minimiser un paramètre. Si le problème se modélise en la maximisation d'un paramètre Λ, il est alors possible d'utiliser $\lambda = 1/\Lambda$. La table 3.5 montre la version (a), pour la version (b) il suffit de changer les lignes 25 et 27 de la manière suivante :

ligne 25 : $\lambda_m \leftarrow \min_i(\max(\lambda(v, u_i), S^i_\lambda[j]))$ pour toute paire (i, j) telle que $S^i[j] = x$;
ligne 27 : $I \leftarrow \{I, u_k\}$, où k et j vérifient $\lambda_m = \max\{\lambda(v, u_k), S^k_\lambda[j]\}$, et $S^k[j] = x$;

Le protocole MCT manipule trois types de messages (cf. table 3.5).

1. Les messages M_{expl}. Ces messages sont utilisés lors de la phase d'exploration. Un message M_{expl} est un quadruplet (T, T_D, L, m), où T est la liste des nœuds x qui appartiennent à un groupe multipoint m, T_D est la liste des distances $d(s, x) = \mathcal{D}'[x]$ de la racine s vers la destination $x \in M$ (dans le même ordre que la liste T, c'est-à-dire $T_D[k]$ correspond à $T[k]$), L est la liste des nœuds déjà visités, et m est le numéro du groupe multipoint m.

Algorithme 3.4

Phase d'exploration : Activée lorsque v reçoit un message d'exploration $M_{expl} = (T, T_D, L, m)$ de w, ou lorsque v est la racine s de l'arbre.
  $L' \leftarrow \{L, v\}$;
  Pour chaque u_i voisin de v tel que $u_i \notin L$ faire
    $T^i \leftarrow \emptyset$; $T^i_D \leftarrow \emptyset$;
    Pour chaque client x du groupe m tel que $x = T[j]$ faire
      Si $|L| + \mathcal{D}_v[i, x] \leq \varrho \cdot T_D[j] + r$, faire
        $T^i \leftarrow \{T^i, T[j]\}$;
        $T^i_D \leftarrow \{T^i_D, T_D[j]\}$;
    Si $T^i \neq \emptyset$ envoyer (T^i, T^i_D, L', m) à u_i;
  Si $T^i = \emptyset$ pour tout i, faire
    $S \leftarrow \emptyset$; $S_\lambda \leftarrow \emptyset$;
    Si v est un membre du groupe m alors
      $S \leftarrow \{S, v\}$;
      $S_\lambda \leftarrow \{S_\lambda, 0\}$;
    Envoyer $M_{sel} = (S, S_\lambda, L, m)$ à son père w;
Fin de la phase d'exploration.

Phase de sélection : Activée lorsque v a reçu un message $M^i_{sel} = (S^i, S^i_\lambda, L, m)$ de toutes les interfaces par lesquelles il a envoyé un message d'exploration (T^i, T^i_D, L, m).
  $S \leftarrow \emptyset$; $S_\lambda \leftarrow \emptyset$;
  Si v est un membre du groupe m, alors
    $S \leftarrow \{S, v\}$; $S_\lambda \leftarrow \{S_\lambda, 0\}$;
  Pour tout membre x du groupe m, faire
    Si $S^i \neq \emptyset$, faire
      $S \leftarrow \{S, x\}$;
      $\lambda_m \leftarrow \min_i(\lambda(v, u_i) + S^i_\lambda[j])$, pour toute paire (i, j) telle que $S^i[j] = x$;
      $S_\lambda \leftarrow \{S_\lambda, \lambda_m\}$;
      $I \leftarrow \{I, u_k\}$, où k et j vérifient $\lambda_m = \lambda(v, u_k) + S^k_\lambda[j]$, et $S^k[j] = x$;
  Si $v \neq s$ faire
    $\{L', w\} \leftarrow L$;
    Envoyer $M_{sel} = (S, S_\lambda, L', m)$ à w;
  Sinon
    Pour chaque voisin u_i de s faire
      $C \leftarrow \emptyset$;
      S'il existe j tel que $u_i = I[j]$ faire
        $C \leftarrow \{C, S[j]\}$;
      Si $C \neq \emptyset$ envoyer $M_{cons} = (C, \{s\}, m)$ à u_i;
Fin de la phase de sélection.

Phase de construction : Activée lorsque $v \neq s$ reçoit un message $M_{cons} = (C, L, m)$.
  $\{L', w\} \leftarrow L$;
  Pour tous les voisins $u_i \neq w$ de v faire
    $C' \leftarrow \emptyset$;
    Pour chaque nœud $x \neq v$ dans C faire
      S'il existe j tel que $x = S[j]$ et $u_i = I[j]$, faire
        $C' \leftarrow \{C', S[j]\}$;
    Si $C' \neq \emptyset$ envoyer $M_{cons} = (C', \{L, v\}, m)$ à u_i;
Fin de la phase de construction.

TAB. 3.5: *Algorithme du protocole MCT*

2. Les messages M_{sel}. Ces messages sont utilisés lors de la phase de sélection. Un message M_{sel} est un quadruplet (S, S_λ, L, m), où S est la liste des nœuds x qui appartiennent au groupe multipoint m de numéro m, S_λ est la liste des valeurs de λ correspondant aux nœuds de S (ordonnés dans le même ordre que la liste S), L est la liste des nœuds dans le chemin de v vers la racine s, et m est le numéro du groupe multipoint m. Notons que, lors de la phase de sélection, la liste I reste dans le nœud v. Cette liste est classée dans le même ordre que la liste S et elle stocke, pour chaque destination x, le prochain nœud dans le chemin à suivre pour se rendre en x.

3. Les messages M_{cons}. Ces messages sont utilisés lors de la phase de construction. Un message M_{cons} est un triplet (C, L, m), où C est la liste des membres que le nœud v doit traiter en vue d'une connexion à l'arbre multipoint, L est la liste des nœuds déjà visités, et m est le numéro du groupe multipoint m.

L'algorithme distribué MCT commence à la racine s (donc $v = s$ dans l'algorithme de la table 3.5) avec un message d'exploration $M_{expl} = (T, T_D, L, m)$, où T contient tous les membres du groupe multipoint, T_D contient les distances de la racine s vers tous les membres x du groupe (si $x_k = T[k]$ alors $T_D[k] = d(s, x)$), et $L = \emptyset$. Le nœud courant envoie sur le lien i, qui joint v au voisin u_i, un message d'exploration $M_{expl} = (T^i, T_D^i, \{s\}, m)$. Les listes T^i et T_D^i sont construites selon la condition suivante. Le membre x_k de la liste T est ajouté si $|L| + \mathcal{D}_v[i, x_k] \leq \varrho \cdot d(s, x_k) + r$, où $|L|$ est le nombre d'éléments de la liste L. Initialement on a $L = \{s\}$ car s est le premier nœud visité. Le nœud qui recevra le message d'exploration M_{expl} cherchera tous les chemins de longueur au plus $\varrho \cdot d(s, x) + r$, et la phase d'exploration se termine en un nœud z qui reçoit la liste $T = \{z\}$, c'est-à-dire qu'il est le dernier élément de la liste T.

Lorsque la phase d'exploration est terminée au nœud z, la phase de sélection commence en ce nœud. Le nœud z envoie à son prédécesseur w le message $M_{sel} = (S, S_\lambda, L, m)$, où $S = T = \{v\}$, $S_\lambda = \{0\}$, et L est la liste reçue dans le message d'exploration. Le nœud w reçoit tous les messages $M_{sel} = (S^i, S_\lambda^i, L, m)$ de ses descendants et commence à construire les paramètres du message $M_{sel} = (S, S_\lambda, L, m)$ pour envoyer à son père. La liste S_λ est construite à partir des équations (3.6) et (3.7), c'est-à-dire en ajoutant la valeur $\lambda(v, u_i)$ à la liste S_λ^i, puis en choisissant le chemin qui minimise le paramètre λ à partir des valeurs $S_\lambda^i[j] = \lambda_{\min}(v, x_j)$ reçues pour tous les x_j de T. Ce calcul effectué, il envoie à son père le message $M_{sel} = (S, S_\lambda, L, m)$.

Finalement tous les messages M_{sel} convergent vers la racine s qui effectue la sélection. La phase de construction commence alors. La racine envoie les messages $M_{cons} = (C^i, \{s\}, m)$ sur l'interface i, où C^i est l'ensemble de nœuds qui appartiennent au groupe multipoint m avec λ minimum. Ces messages sont diffusés dans le réseau.

Une fois que la phase de construction est finie, l'arbre est prêt à commencer à envoyer les paquets de la communication multipoint.

Remarque. En utilisation normale, il faut exécuter MCT régulièrement afin de tenir compte des modifications potentielles du groupe, et surtout pour capturer la caractéristique

dynamique du trafic. On peut par exemple envisager d'exécuter MCT toutes les τ secondes. Ce dernier point sera discuté en détail à la section 4.2 dans les cas où le paramètre λ est la *congestion probabiliste* définie dans la section 4.1.3.

Nous montrons ci-dessous que le protocole MCT n'a pas de boucle, et qu'il termine en temps fini.

Propriété 3.3 *L'algorithme MCT n'a pas de boucles.*

Preuve. Une boucle implique qu'un message M_{expl}, M_{sel} ou M_{cons} passe deux fois par le même nœud. Dans le cas d'un message $M_{expl} = (T, T_D, L, m)$, le nœud courant v est ajouté dans la liste des nœuds visités L, et v vérifie de ne pas envoyer un message M_{expl} aux membres de cette liste (voir les instructions 2 et 3 dans la table 3.5). Il ne peut donc y avoir de boucles liées au messages d'exploration. Un message de sélection $M_{sel} = (S, S_\lambda, L, m)$ est envoyé au père w du nœud v, l'unicité de ce père découlant de la construction de la liste L. Il ne peut donc pas non plus y avoir de boucles liées au messages de sélection. Enfin, un message de construction $M_{cons} = (C, L, m)$ est envoyé à un sous-ensemble de la liste T obtenue dans le message d'exploration M_{expl}. Comme ces derniers messages n'induisent pas de boucle, les messages de construction n'induisent pas non plus de boucles. ■

Propriété 3.4 *L'algorithme MCT termine en temps fini.*

Preuve. La racine s possède la liste T de tous les membres du groupe multipoint m. Un nœud v quelconque recevant un message d'exploration ajoute un membre x du groupe multipoint à la liste T du message M_{expl} qui sera envoyé à son voisin u_i seulement si la distance $d_{u_i}(v, x)$ en passant par le voisin u_i est au plus $|L| + \varrho \cdot d(s, x) + r$. À chaque étape on effectue le même calcul en utilisant la distance originale à partir de la racine plus le nombre de nœuds déjà traversés depuis la racine (c'est-à-dire le nombre d'éléments dans la liste L). Donc le nombre de nœuds traversés pour arriver à chaque destination ne peut pas dépasser un nombre fini, et donc la phase d'exploration termine. (On peut également remarquer que la liste T envoyée dans les messages d'exploration est modifiée chaque fois que le nœud courant v appartient à T. En ce cas, v est enlevé de la liste T. Ainsi, la liste T devient de plus en plus petite jusqu'au dernier nœud qui appartient au groupe multipoint m.) La phase de sélection termine quand les messages arrivent à la racine, et la phase de construction termine parce que la liste C est construite à partir des éléments x de la liste T vérifiant $v \neq T[i]$ (voir l'instruction 43 dans la table 3.5). ■

3.3.3 Simulations sur le réseau ARPANET (1995)

Le réseau ARPANET a formé la partie centrale d'`Internet` pendant ses premières années. ARPANET est le réseau du ministère de la défense des États Unis. Il a motivé la création de la pile de protocoles TCP/IP (voir la section 2.1). Nous avons utilisé la topologie de ce réseau

en 1995 pour simuler et comparer des protocoles multipoint. ARPANET avait 47 nœuds en 1995, son diamètre était 8, et il avait 90 liens.

Pour évaluer le comportement des protocoles, nous avons créé différents scénarios. L'ensemble des valeurs du paramètre λ assignées aux liens définit un scénario. On a généré 10.000 scénarios différents, c'est-à-dire 10.000 versions d'ARPANET, pour 10.000 ensembles différents de valeurs λ assignées aux lien. Les valeurs de λ ont été choisi entre 0 et 100 selon deux lois statistiques : la loi uniforme et la loi exponentielle. Nous avons étudié le comportement de MCT dans ses deux versions `(a)` et `(b)`. Pour évaluer les résultats, nous avons mis en œuvre le protocole de simulation suivant. Nous choisissons un groupe multipoint de n sommets tirés au hasard (loi uniforme) entre les nœuds du graphe d'ARPANET. Nous calculons l'arbre pour les 10.000 scénarios générés, selon les deux lois. Pour chaque arbre T obtenu nous calculons le paramètre $\lambda(T)$ correspondant.

Nous avons représenté les résultats obtenus dans nos simulations par une fonction de répartition F correspondant à la probabilité d'avoir un arbre dont la valeur $\lambda(T)$ soit $\leq \lambda_0$. Les abscisses sont donc les valeurs possibles de $\lambda(T)$. L'axe des ordonnées correspondant au pourcentage d'expériences pour lesquelles $\lambda(T) \leq \lambda_0$. Ainsi,

$$F(\lambda_0) = P(\lambda(T) \leq \lambda_0). \tag{3.8}$$

Un protocole de construction d'arbre est donc d'autant meilleur que sa fonction F croît rapidement.

Intéressons nous d'abord au choix des paramètre ϱ et r de MCT. La figure 3.5 présente les simulations obtenues en faisant varier ϱ et r pour un groupe de 5 membres (version `(a)`). Dans la table 3.6, on indique également le nombre de messages générés en moyenne par MCT, pour des valeurs différentes de ϱ et r, et pour des tailles différentes de groupes. On peut également lire le nombre moyen de liens utilisés. Par l'arbre construit on observe que le nombre de messages croît avec r et ϱ, mais surtout avec ϱ. Pour les cas où les valeurs λ des liens ont été générées avec la loi uniforme, tous les cas ont un comportement semblable (cf. figure 3.5). Dans le cas où les λ ont été générés avec la loi exponentielle on distingue nettement la courbe correspondant a $(\varrho, r) = (1, 2)$, mais les autres courbes sont très proches. Choisir $(\varrho, r) = (1, 3)$ semble un bon compromis, car le nombre de messages pour construire l'arbre reste petit, et les résultats sont significativement meilleures que pour $(\varrho, r) = (1, 2)$.

Pour la version `(b)` dans les mêmes conditions, les simulations sont montrées sur la figure 3.6. De nouveau celles qui correspondent à le loi uniforme donnent des courbes très semblables. Elles sont presque pareils sauf pour les valeurs de $60 < \lambda(T) < 90$. Pour ces valeurs, la courbe $(\varrho, r) = (1, 2)$ est un peut écartée et elle montre un comportement moins bon que les autres. Pour le cas de la loi exponentielle, on observe la courbe $(\varrho, r) = (1, 2)$ avec un comportement presque linéaire. De bas en haut, on trouve ensuite la courbe $(\varrho, r) = (1, 3)$, puis $(\varrho, r) = (1, 4)$, et finalement toutes les autres avec le meilleur comportement. De nouveau, un choix équilibrant le nombre de paquets générés et trouvant un arbre $\lambda(T)$ proche à l'optimal semble être $(\varrho, r) = (1, 3)$.

Dans la suite nous allons donc utiliser $\varrho = 1$ et $r = 3$.

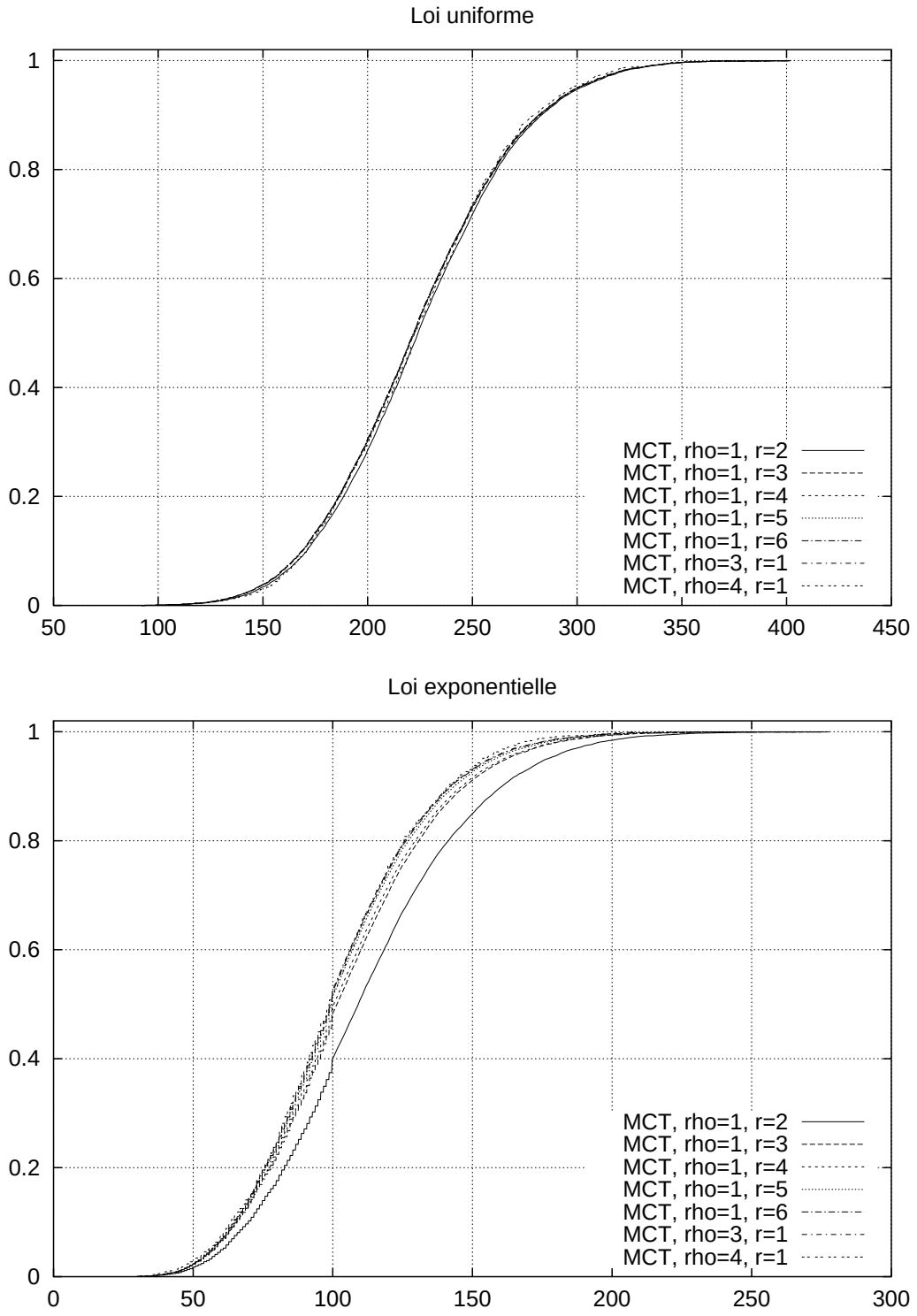

FIG. 3.5: *ARPANET, groupe de* 5 *membres, influence de* ϱ *et* r *(version* `(a)`*).*

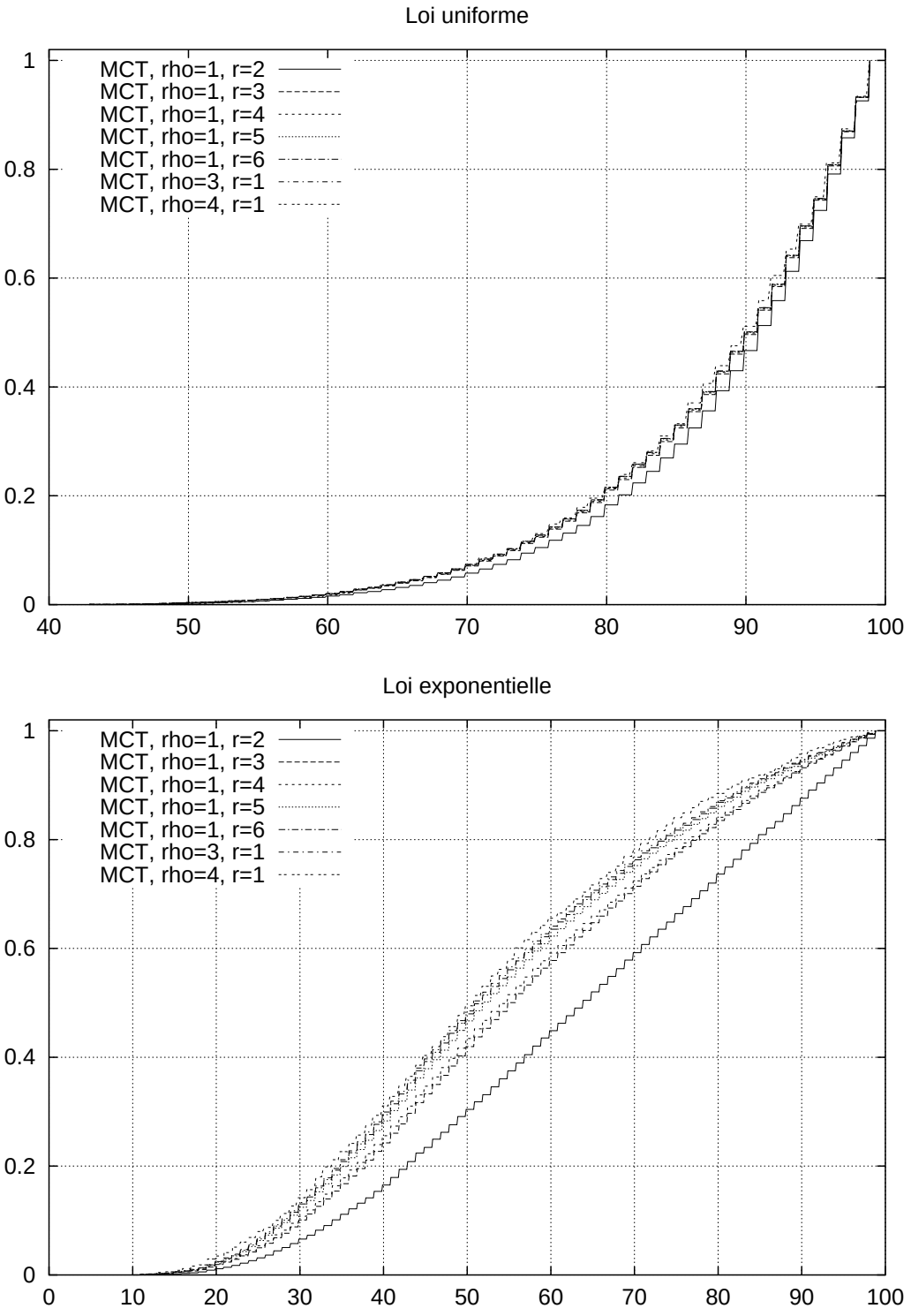

FIG. 3.6: *ARPANET, groupe de* 5 *membres, influence de* ϱ *et de* r *(version* `(b)`*).*

Nb. de membres	ϱ	r	Nb. de messages	Nb. de liens
5	1	2	99	14.1
5	1	3	171	14.5
5	1	4	318	14.6
5	1	5	615	14.8
5	1	6	1067	14.9
5	2	1	901	14.8
5	3	1	12568	15.0
5	4	1	114174	15.0
5	1	3	171	14.5
10	1	3	247	22.5
20	1	3	425	32.1

TAB. 3.6: *Nombre moyen de messages envoyés et nombre moyen de liens utilisés par MCT.*

Remarque. Il faut noter que ces résultats ne sont valides que pour la topologie du réseau ARPANET (de 1995). Pour chaque topologie il faudra vérifier quelles valeurs de ϱ et de r conviennent, afin d'obtenir un bon rapport entre qualité de l'arbre et nombre de messages envoyés.

Nous avons ensuite mené des simulations ayant pour objectif la comparaison entre différentes techniques de construction d'arbres multipoint. Nous avons retenus les méthodes suivantes :

- Plus court chemin (noté RSP, pour *Reverse Shortest Paths*), c'est-à-dire que le chemin de la racine à un nœud x de l'arbre est un plus court chemin de x à r dans le réseau.
- Glouton (noté Greedy), c'est-à-dire la méthode consistant à connecter nouveau un nœud au plus proche nœud parmi les nœuds de l'arbre courant.
- QoSMIC [FBP98], c'est-à-dire glouton avec prise en compte de la qualité de service dans le choix du chemin connectant un nouveau membre à l'arbre courant.
- MCT avec $\varrho = 1$ et r=3, dans les deux versions `(a)` et `(b)`.

La table 3.7 montre le nombre moyen de messages envoyés par chaque protocole. La même table donne également la moyenne du nombre de liens des arbres construits. Les valeurs des deux tableaux 3.6 et 3.7 ont été obtenues de la manière suivante. Pour RSP on a compté un message pour chaque lien ajouté. Pour la méthode gloutonne on a compté un message pour chaque voisin à distance $1, .., k$, où k est la distance à l'arbre. Pour QoSMIC on a compté de la même manière que glouton sauf que k est la distance au nœud le plus éloigné de l'arbre courant. On observe que le trafic de contrôle de MCT est assez important comparé aux autres protocoles. La proportion reste toutefois indépendante de la taille du groupe. De tous les protocoles, RSP est le plus économe en trafic de contrôle pour la construction de l'arbre multipoint.

Les résultats de la version `(a)` de MCT sont montrés dans la figure 3.7 pour un groupe de 5 membres, dans la figure 3.8 pour un groupe de 10 membres, et dans la figure 3.9 pour un groupe de 20 membres. Les résultats pour la version `(b)` sont montrés dans la figure 3.10

pour un groupe de 5 membres, dans la figure 3.11 pour un groupe de 10 membres, et dans la figure 3.12 pour un groupe de 20 membres.

Pour la version `(a)`, on constate que MCT a un meilleur comportement que tous les autres protocoles. En effet, la plupart des arbres obtenus par MCT ont un λ plus petit que celui des autres protocoles. Par exemple, la figure 3.7 montre que, pour le cas de distributions exponentielles, 80% des arbres obtenus par MCT ont un $\lambda = 130$, tandis que, pour la même valeur borne 130, on n'obtient que 40% des arbres pour QoSMIC, 32% pour RSP, et 23% pour Greedy. Les autres protocoles se comportent de manières différentes selon la taille du groupe, et la statistique avec laquelle les poids des arêtes ont été fixés. Par exemple, QoSMIC minimise la qualité de service pour chaque membre qui se connecte à l'arbre déjà existant. Une application de la version `(a)` pour $\lambda =$`{délai}` n'assure pas la minimisation globale du délai par QoSMIC. On peut noter clairement le problème dans le cas de la loi uniforme où le comportement de RSP est meilleur que celui de QoSMIC ! Ceci s'explique par le fait que RSP construit des arbres en minimisant la distance de chaque membre à la racine.

Pour la version `(b)`, on constate que le comportement de MCT est bon bien que, dans certains cas, il reste comparable à celui de QoSMIC. Par exemple, dans le cas de la loi uniforme pour un groupe de 20 membres, QoSMIC est légèrement meilleur que MCT avec $\varrho = 1$ et $r = 3$. Néanmoins, si MCT était utilisé avec des paramètres qui permettraient d'explorer le réseau plus largement, alors il serait meilleur que QoSMIC, mais au prix d'échanges importants de messages pour la construction de l'arbre. Pour la loi exponentielle, le comportement de MCT est très bon. On peut toutefois constater que QoSMIC se rapproche de MCT quand la taille du groupe multipoint croît. Le comportement de RSP et de Greedy se ressemblent. Comme on a vu, une application possible de la version `(b)` est $\lambda = 1/$`{bande passante libre}`. Dans ce cas QoSMIC a une très bonne performance comparé aux autres protocoles.

Membres	Protocole	Messages	Liens
5	RSP	13	13.0
5	Greedy	63	12.0
5	QoSMIC	68	12.7
5	MCT, $\varrho = 1, r = 3$	171	14.5
10	RSP	17	17.0
10	Greedy	89	18.0
10	QoSMIC	105	19.3
10	MCT, $\varrho = 1, r = 3$	247	22.5
20	RSP	30	30.0
20	Greedy	122	28.0
20	QoSMIC	150	29.7
20	MCT, $\varrho = 1, r = 3$	425	32.1

TAB. 3.7: *Nombre moyen de messages envoyés et nombre moyen de liens utilisés.*

3.3.4 Conclusion

La conclusion générale que nous tirons de cette section et que MCT peut être utilisé avec des valeurs petites de ϱ et r (pour les deux versions proposées) en obtenant toutefois des

performances souvent bien meilleures que celles de protocoles classiques. Dans le chapitre suivant, nous allons considérer une méthode plus économe pour générer de bons arbres de multicast en prenant en compte la nature du trafic.

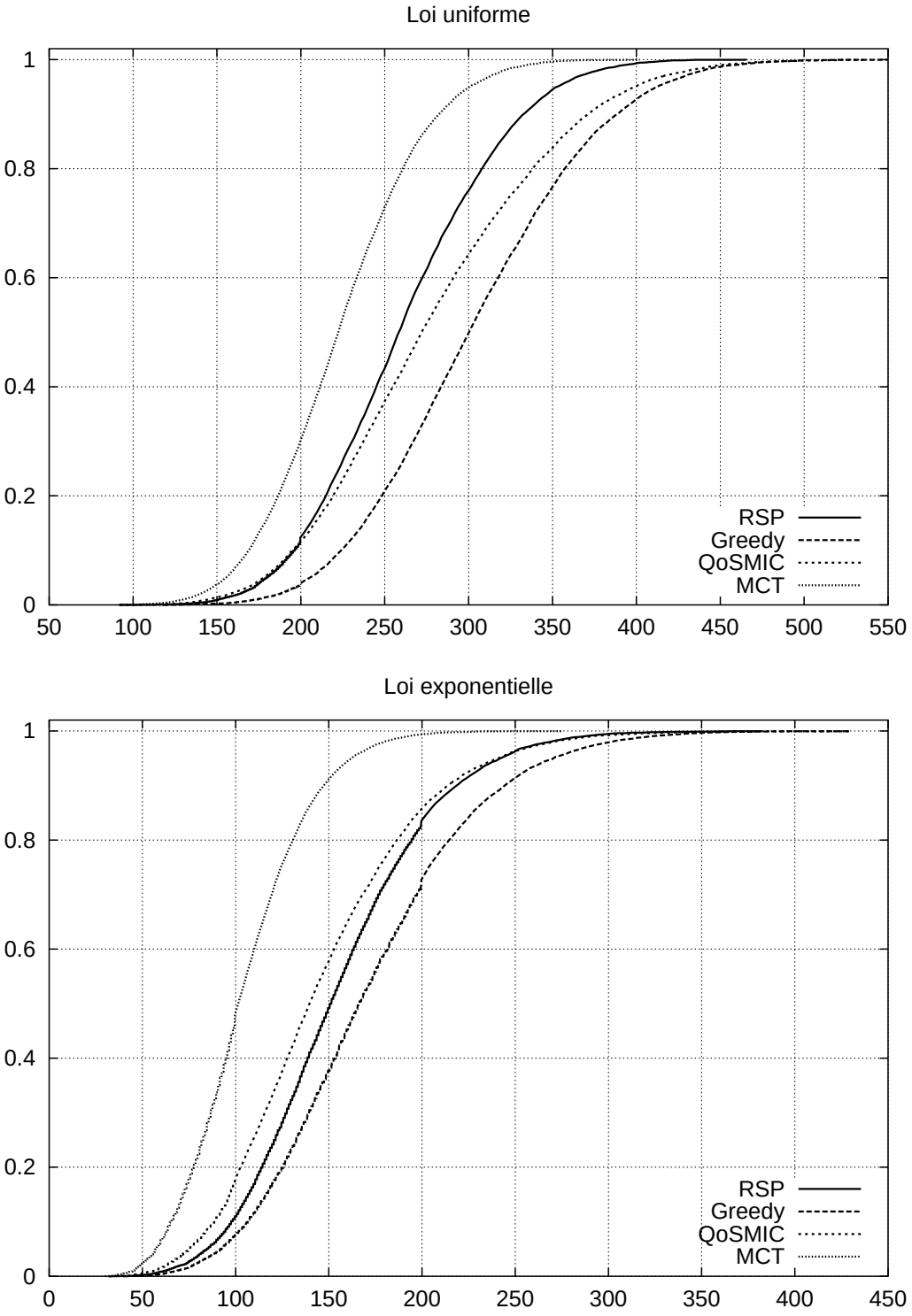

FIG. 3.7: *ARPANET, groupe de* 5 *membres, version* (a).

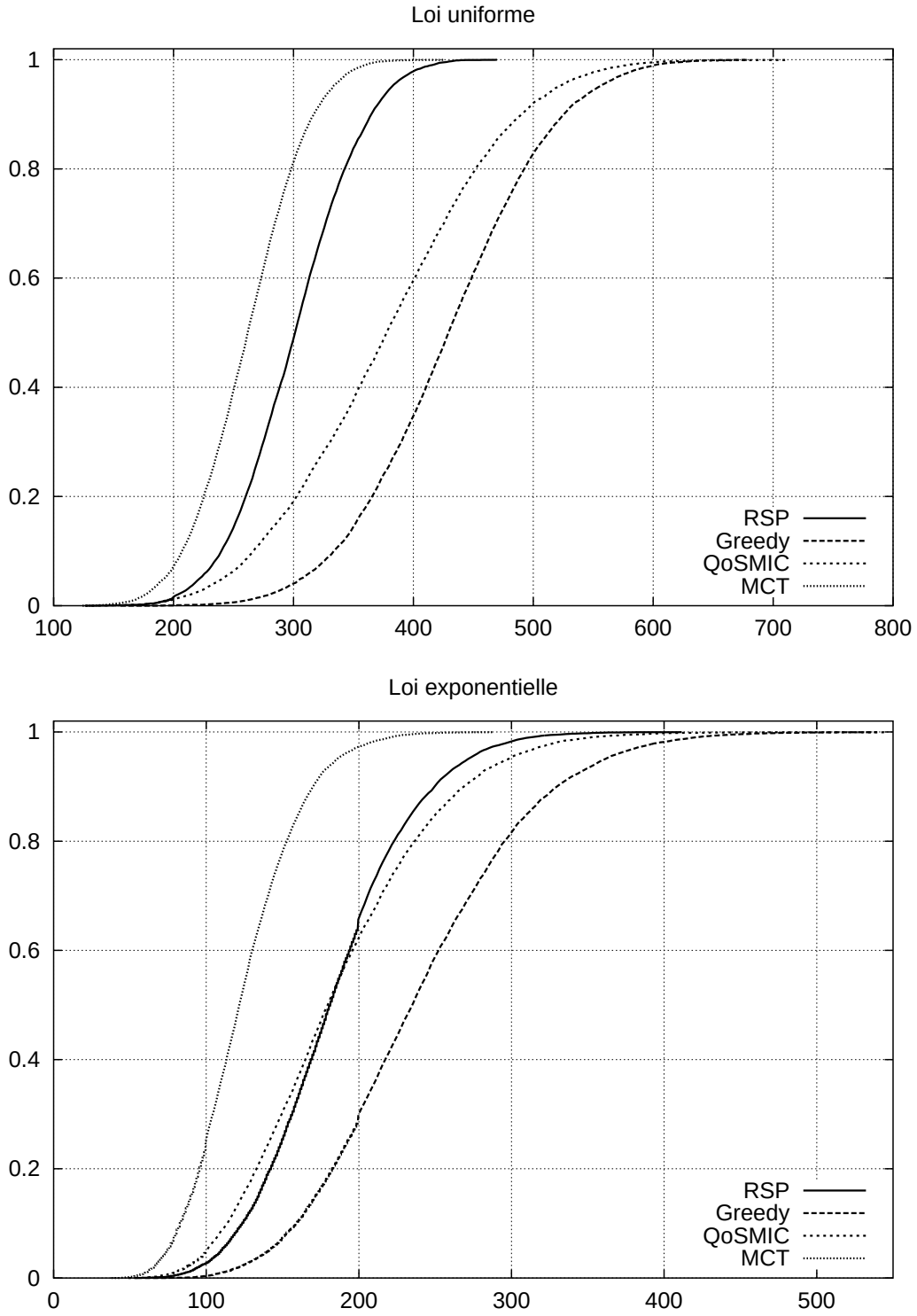

FIG. 3.8: *ARPANET, groupe de* 10 *membres, version* (a).

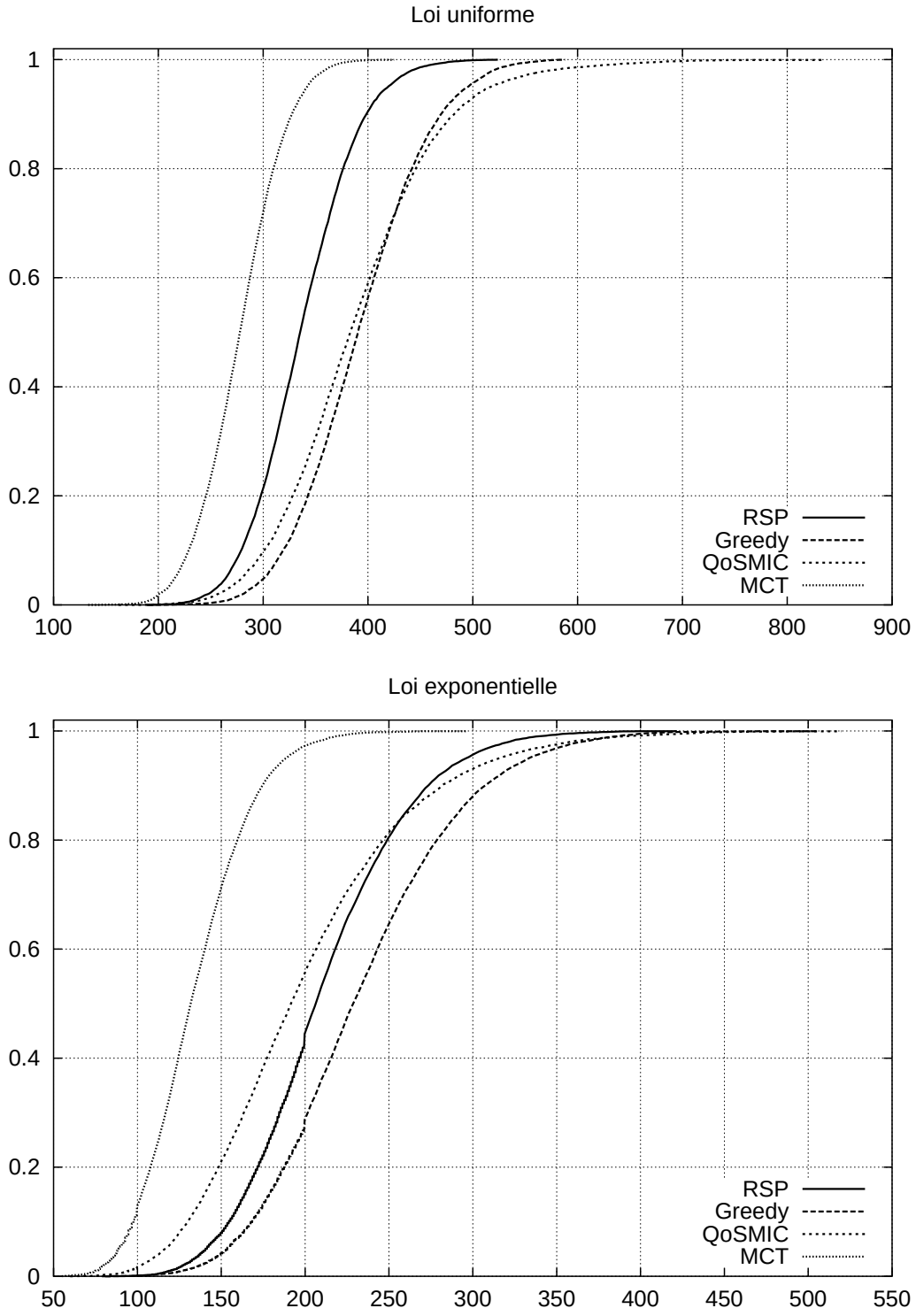

FIG. 3.9: *ARPANET, groupe de* 20 *membres, version* (a).

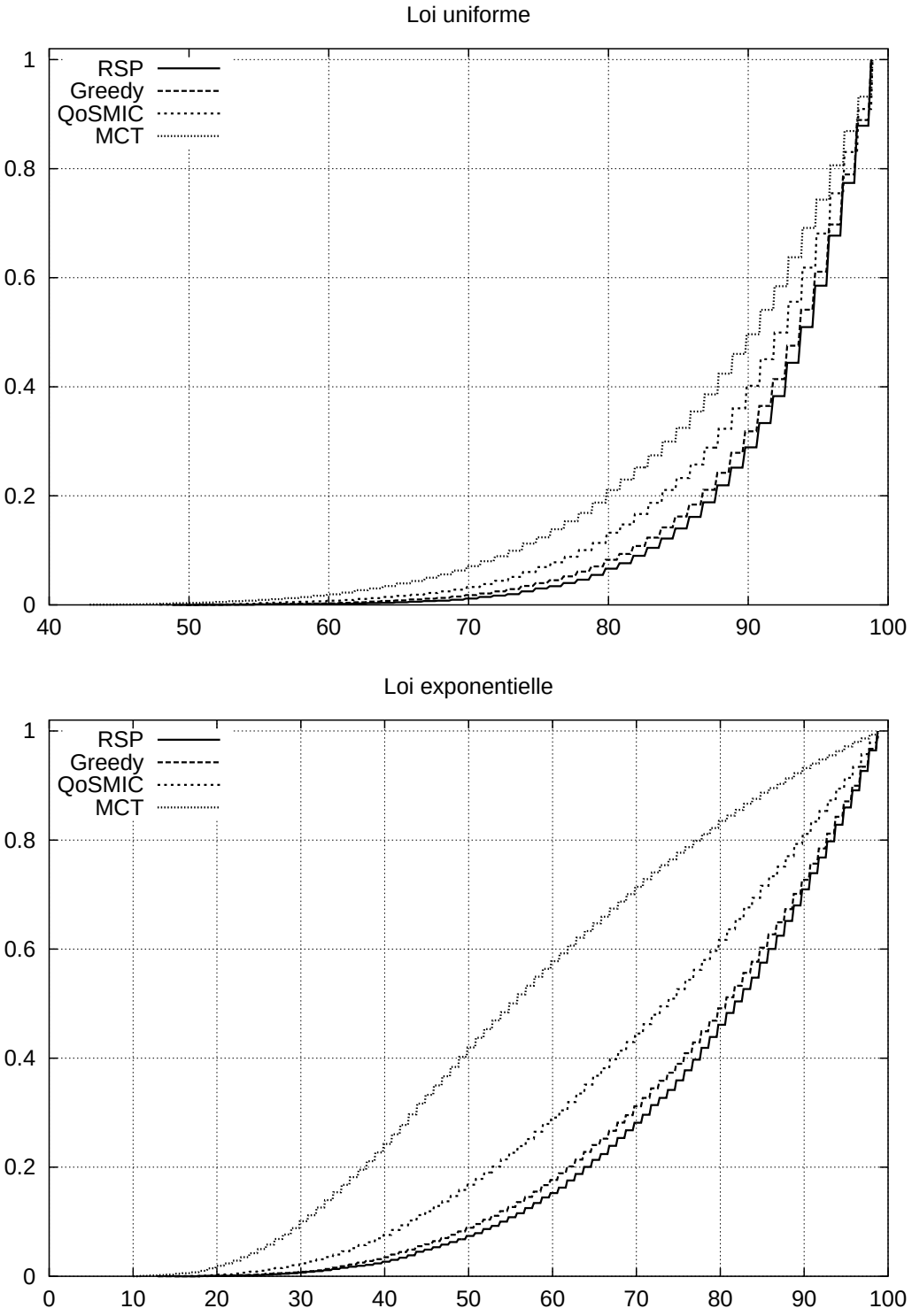

FIG. 3.10: *ARPANET, groupe de* 5 *membres, version* (b).

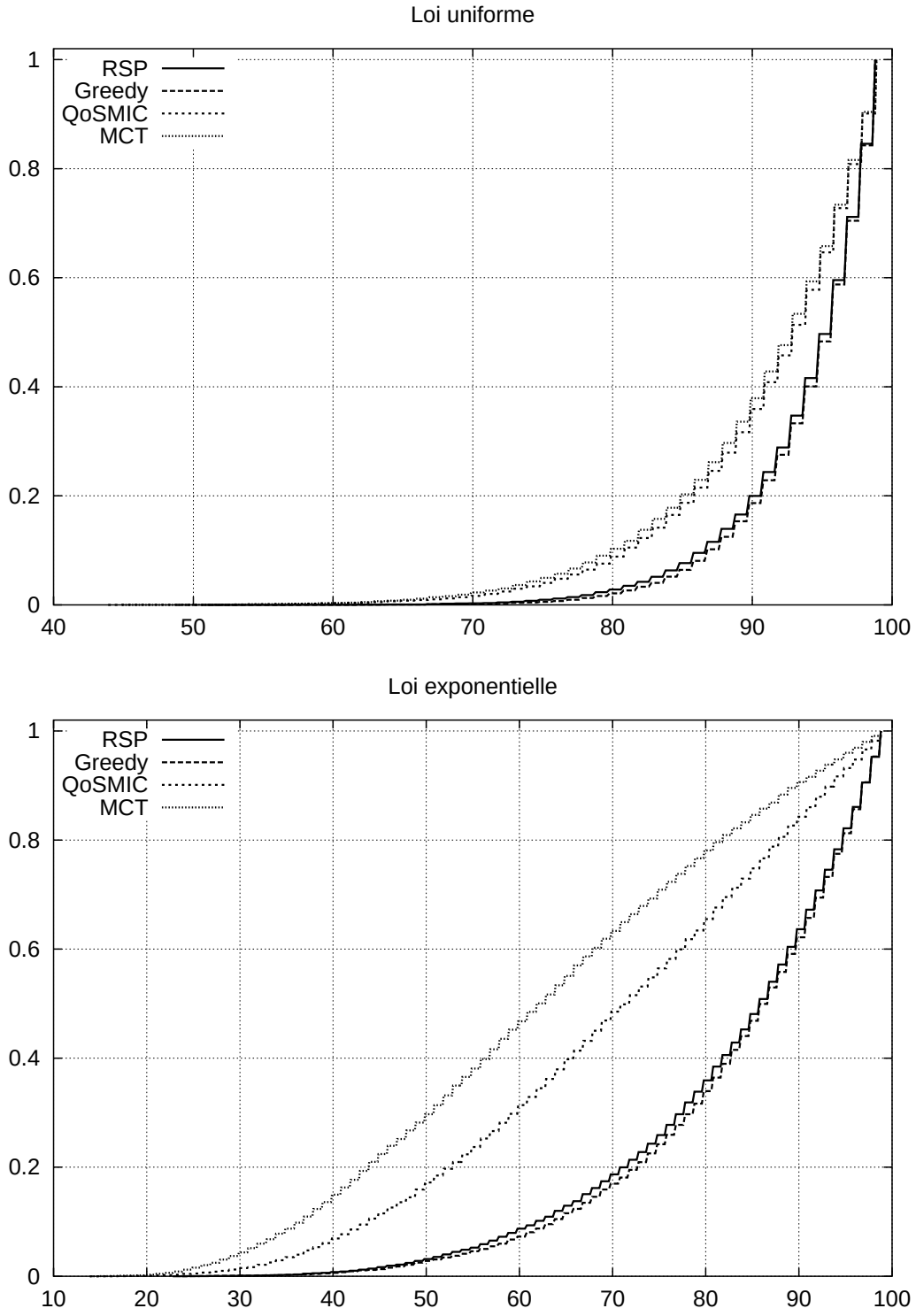

FIG. 3.11: *ARPANET, groupe de* 10 *membres, version* (b).

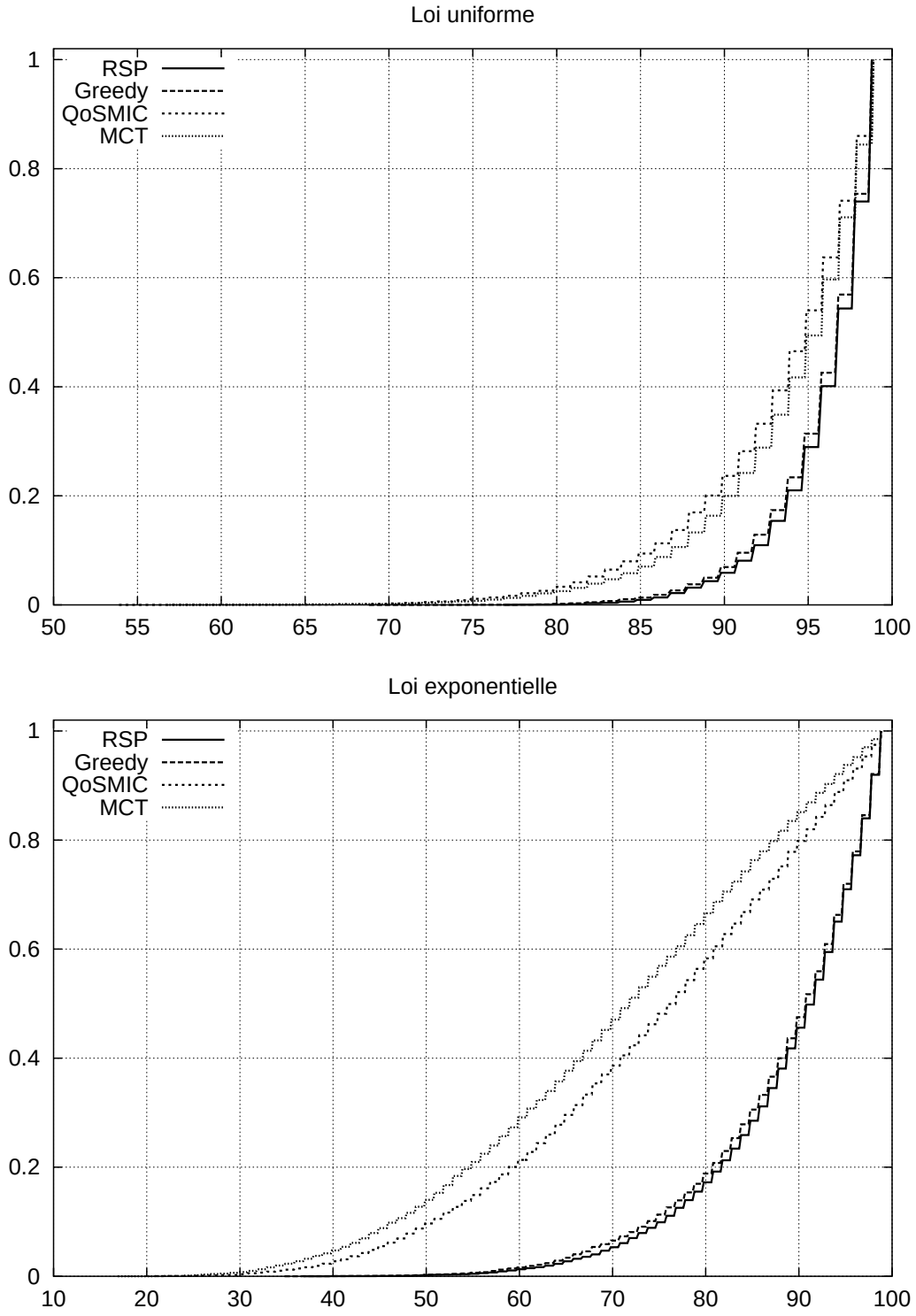

FIG. 3.12: *ARPANET, groupe de* 20 *membres, version* (b).

Chapitre 4

Un nouveau paramètre de QoS : la congestion probabiliste

Dans ce chapitre nous introduisons un nouveau paramètre de qualité de service : la *congestion probabiliste*. Ce paramètre est basé sur un modèle autosimilaire du trafic, modèle qui tend actuellement à remplacer le modèle poissonnien. L'intérêt de ce nouveau paramètre est qu'il permet, sous des hypothèses d'autosimilarité, de capturer une mesure de taux de perte de paquets. Nous nous sommes intéressés à la perte de paquets car un taux de perte important implique de grands délais de communication et induit une dégradation des performances des applications. Dans un réseau "best effort", dans lequel la livraison des paquets n'est pas assurée, il est important de maximiser la probabilité qu'un paquet soit délivré afin d'éviter qu'une application ne demande de renvoyer plusieurs fois le même paquet, ce qui induit une dégradation des performances. La congestion probabiliste nous permettra d'améliorer le fonctionnement des applications, notamment les applications multipoint. Les protocoles de routage basés sur la congestion probabiliste pourront en effet trouver des routes moins congestionnées, et donc des taux de perte faible.

Ce chapitre est divisé en deux sections. La première section est dédiée au modèle autosimilaire du trafic. Nous listons les expériences qui ont permis de mettre en évidence la nature autosimilaire du trafic dans les réseaux de paquets IP. Nous décrivons ensuite la modélisation du trafic sur un lien d'un tel réseau, nous donnons la définition de la congestion probabiliste et montrons comment effectuer son estimation. Dans la deuxième section nous donnons des résultats des simulations effectuées d'une part sur des topologies réelles, et d'autre part sur des topologies générées avec le modèle proposé dans la section 2.3. Ces simulations ont pour but de montrer que le nombre de paquets perdus dans un arbre multicast peut être diminué significativement grâce à l'utilisation de la congestion probabiliste. Enfin, nous montrons comment diminuer encore le taux de paquets perdus en combinant le protocole défini à la section 3.3 et la congestion probabiliste.

4.1 Modèle de trafic autosimilaire

Au début des années 90 plusieurs études empiriques ont montré que le trafic d'un réseau IP offrait un comportement différent de celui des réseaux téléphoniques. En particulier, il n'offrait pas un caractère poissonnien, mais plutôt *autosimilaire*. Le comportement d'un trafic autosimilaire est très différent du trafic poissonnien car il présente de grandes variations à différentes échelles de temps. L'hypothèse d'autosimilarité du trafic permet bien de retrouver la dépendance à long terme observée empiriquement. Dans le cas du modèle poissonnien la fonction d'auto-corrélation baisse de manière exponentielle, tandis que la fonction d'auto-corrélation du modèle autosimilaire baisse très lentement, ce qui semble correspondre aux observations.

La première observation de l'autosimilarité du trafic a été faite dans un réseau local LAN à Bellcore par Leland, Taqqu, Willinger et Wilson [LTWW93]. Plus tard, Paxon et Floyd [PF95] ont montré qu'il y a également de l'autosimilarité dans les réseaux WAN. A partir de ces observations, le débat a été ouvert pour trouver une explication au phénomène. Taqqu, Willinger et Sherman [TWS97] d'une part, et Willinger, Taqqu, Sherman et Wilson [WTSW97] d'autre part, ont montré que l'autosimilarité dans les LANs pourrait être provoquée par la superposition de sources de type *on-off* avec une grand variation des périodes *on* et *off*. Crovella et Bestavros [CB97], et Crovella, Taqqu et Bestavros [CTB98], ont également mis en évidence des phénomènes autosimilaires dans la toile WWW (*World Wide Web*). Richard et Cappello [RC98] ont montré le caractère autosimilaire de la charge de travail des serveurs WWW. Beran, Sherman, Taqqu et Willinger [BSTW95] l'ont montré pour le trafic vidéo de débit variable VBR (*Variable Bit Rate*). Il a été également montré par Park, Kim et Crovella [PKC97] que les mécanismes de transport peuvent engendrer de l'autosimilarité dans le trafic. En particulier, le mécanisme du contrôle d'écoulement du trafic de TCP peut maintenir la dépendance à long-terme créée par la distribution à queue épaisse de la taille des fichiers transmis [PKC96]. A cause de son absence de contrôle d'écoulement du trafic, UDP ne propage pas l'autosimilarité. A l'inverse, Veres, Kenesi, Molnár et Vattay [VKMV00] ont montré comment TCP propage l'autosimilarité. Si un flux TCP non autosimilaire rencontre une deuxième flux TCP autosimilaire, le premier flux acquiert la caractéristique autosimilaire du deuxième. Veres et Boda [VB00] ont de plus montré que les flux TCP peuvent devenir autosimilaires en transitant dans le réseau. En effet, si TCP a un contrôle de congestion déterministe, il peut néanmoins créer du chaos qui génère l'autosimilarité dans certaines conditions. Finalement, on retrouve un comportement autosimilaire dans d'autres technologies de réseaux de paquets, en particulier ATM [MV97, TG97, VMGC98].

La question naturelle qui découle de ces études est : comment prendre en compte l'autosimilarité du trafic pour augmenter les performances du réseau. Giordano, Pagano, Pannocchia et Russo [GPPR96] ont présenté un nouveau schéma d'admission d'appels pour un réseau du type ATM. Il profite de l'autosimilarité du trafic pour calculer un volume adéquat de flux de trafic afin de maximiser l'utilisation des liens. En utilisant le fait que le trafic soit autosimilaire, Östring, Sirisena et Hudson [ÖSH00] ont présenté un contrôle de congestion basé sur une prédiction pour le trafic ABR[1] d'ATM. Ils ont également présenté un contrôle

1. *Available Bit Rate* : c'est un des quatre types de trafics définis dans la technologie ATM.

de congestion pour les réseaux IP basé sur l'élimination de paquets dans la file d'attente, toujours en utilisant les résultats du modèle autosimilaire. Dans leur article, les auteurs ont mené une étude comparative avec d'autres systèmes de contrôle de congestion, comme par exemple RED [FJ93]. Ils ont montré que, pour la même taille de file d'attente, leur méthode a un pourcentage d'utilisation du lien plus élevé que dans l'application de RED. Enfin, Park et Tuan [PT00] ont décrit une application des caractéristiques autosimilaires du trafic au contrôle de congestion inhérent à TCP. Ils ont en particulier décrit une stratégie permettant l'augmentation du taux de transmission.

Il existe ainsi un grand nombre d'études qui démontrent le caractère autosimilaire du trafic dans les réseaux IP et dans les réseaux de paquets en général. Nous allons donc baser notre étude sur l'hypothèse d'autosimilarité du trafic, et nous renvoyons le lecteur aux articles cités ci-dessus pour se convaincre de la validité de cette hypothèse.

Cette section est organisée de la manière suivante. Tout d'abord nous rappelons le modèle autosimilaire de trafic proposé par Norros, ainsi que le modèle de lien avec file d'attente d'entrées autosimilaires. Nous définirons ensuite notre nouveau paramètre de qualité de service : la congestion probabiliste.

4.1.1 Modèle de trafic autosimilaire Gaussien

Norros a étudié en profondeur le modèle autosimilaire du trafic dans une série d'articles [Nor94, Nor95, Nor99]. Notre modèle se base sur cette étude, dont nous rappelons ici les grandes lignes. Considérons un élément d'un réseau de paquets sans connexion (par exemple un routeur). On note $A(t)$ la quantité de trafic que le réseau transmet à cet élément pendant l'intervalle de temps $[0, t[$, pour $t > 0$, et $]t, 0]$, pour $t < 0$. Pour $t \in]-\infty, \infty[$, le trafic offert pendant l'intervalle de temps $[s, t[$ est noté comme $A(s,t) = A(t) - A(s)$. On dit que le processus $A(t)$ a des *incréments stationnaires* si pour tout $t \in \mathbb{R}$ et toute suite $s_1 < \cdots < s_n$ finie, la distribution de

$$(A(t + s_1, t + s_2), \ldots, A(t + s_{n-1}, t + s_n)) \tag{4.1}$$

est indépendant de t. Dans la suite, nous supposerons que $A(t)$ a des incréments stationnaires. On suppose également que le carré de la fonction est intégrable, de façon à ce que

$$E(A(t)^2) < \infty. \tag{4.2}$$

De la propriété d'incréments stationnaires, on peut déduire que $E(A(t)) = m \cdot t$, où m est la moyenne de la quantité de données envoyées au routeur par le réseau. De manière analogue, la variance $Var(A(t))$ est une fonction du temps. Si on note $v(t) = Var(A(t))$, alors on a

$$Cov(A(s), A(t)) = \frac{1}{2}\left(v(t) - v(s) - v(t-s)\right) \qquad \forall\, (s,t) \in \mathbb{R}^2 \;:\; 0 < s < t \tag{4.3}$$

La structure de corrélation de $A(t)$ (propriété du deuxième ordre) est donc déterminée uniquement par la fonction $v(t)$.

Le processus $A(t)$ est dit dépendant *à court terme* si pour tout $s < t \leq u < v$ le coefficient de corrélation entre $A(\alpha s, \alpha t)$ et $A(\alpha v, \alpha u)$ converge vers zéro quand l'échelle de temps α tend vers l'infini. Sinon, le processus est dit dépendant *à long terme*.

Selon les travaux récents de Leland *et al.* [LTWW93] déjà mentionné dans l'introduction de la section 4.1, le trafic IP réel suit une fonction de variance $v(t)$ qui est proportionelle à t^ξ, où $\xi \in [1, 2[$ pour six ordres de grandeur de temps. Par conséquent, le trafic IP est bien dépendant à long terme. De ces résultats on supposera que la variance est de la forme $v(t) = t^\xi$. Donc pour un processus étudié à différentes échelles de temps, on a

$$Var(A(\alpha t)) = v(\alpha t) = (\alpha t)^\xi = \alpha^\xi \cdot t^\xi = \alpha^\xi \cdot Var(A(t)) \tag{4.4}$$

Cette dernière équation implique que $A(\alpha t)$ et $\alpha^{\xi/2} A(t)$ ont la même structure de corrélation, c'est-à-dire $A(t)$ est autosimilaire.

Les processus gaussiens se prêtent à la construction d'un modèle vérifiant les propriétés de deuxième ordre mentionnées ci-dessus. Un *mouvement brownien fractionnaire* est un processus gaussien autosimilaire. Plus précisément, un mouvement brownien fractionnaire normalisé de paramètre d'autosimilarité[2] $H \in [\frac{1}{2}, 1[$ est un processus stochastique Z_t, avec $t \in]-\infty, \infty[$, caractérisé par les propriétés suivantes :

1. Z_t a des incréments stationnaires ;
2. $Z_0 = 0$, et $E(Z_t) = 0 \qquad \forall\, t \in \mathbb{R}$;
3. $E(Z_t)^2 = Var(Z_t) = |t|^{2H} \qquad \forall\, t \in \mathbb{R}$;
4. Z_t a des trajectoires continues ;
5. Z_t est gaussien, c'est-à-dire que toutes les distributions marginales sont gaussiennes.

Le premier travail sur ce type de processus a été mené par Mandelbrot et Van Ness [MN68] pour le cas de $H = \frac{1}{2}$. L'autre cas extrême est $H = 1$, pour lequel Z_t est un processus déterministe avec trajectoires linéaires [Nor95]. La covariance des incréments d'un mouvement brownien fractionnaire sur deux intervalles non superposés est toujours positive, et son expression est:

$$Cov(Z_{t_2} - Z_{t_1}, Z_{t_4} - Z_{t_3}) = \frac{1}{2}\left((t_4 - t_1)^{2H} - (t_3 - t_1)^{2H} + (t_3 - t_2)^{2H} - (t_4 - t_2)^{2H}\right) \tag{4.5}$$

pour tout $t_1 < t_2 \leq t_3 < t_4$.

Définition 4.1 (Norros [Nor95]) *Un* trafic brownien fractionnaire *(TBF) est un processus*

$$A(t) = m\, t + \sqrt{a\, m}\, Z_t, \tag{4.6}$$

où Z_t est le mouvement brownien fractionnaire normalisé avec paramètre d'autosimilarité $H \in [\frac{1}{2}, 1[$, m est la moyenne, et a est le coefficient de variance.

2. H est aussi appelé paramètre de Hurst

Le facteur $\sqrt{a\ m}$ de l'équation (4.6) est motivé par la superposition de plusieurs trafics browniens fractionnaires indépendants $A^{(i)}(t) = m_i\ t + \sqrt{a\ m_i} Z_t$ de même paramètres H et a. En appliquant la propriété de la superposition, on a

$$\begin{aligned} A(t) &= \sum_{i=1}^{K} A^{(i)}(t) \\ E(A(t)) &= \sum_{i=1}^{K} E(A^{(i)}(t)) \\ &= \sum_{i=1}^{K} m_i = m; \qquad (4.7) \\ Var(A(t)) &= \sum_{i=1}^{k} Var(A^{(i)}(t)) \\ &= \sum_{i=1}^{k} a\ m_i\ Var(Z_t^{(i)}). \\ &= a\ m \qquad (4.8) \end{aligned}$$

En appliquant (4.7) et (4.8), on obtient

$$A(t) = m\ t + \sqrt{a\ m}\ Z_t.$$

Ces équations illustrent les fonctions des trois paramètres du modèle de trafic décrit dans la définition 4.1 : H et a caractérisent la "qualité" du trafic, alors que m est caractéristique de la "quantité" de trafic.

4.1.2 File d'attente avec entrées autosimilaires

Dans cette section nous étudions le comportement d'une file d'attente avec en entrée un trafic autosimilaire. Ceci modélise le comportement d'un lien dans un réseau. Après la définition formelle de la file, nous rappellerons les bases du problème de changement d'échelle de temps. Ceci nous servira à décrire le calcul d'une borne inférieure de la taille maximum de la file d'attente. C'est cette borne qui nous permettra de définir la congestion probabiliste dans la section 4.1.3.

Un composant du réseau, par exemple un routeur, reçoit des paquets par les interfaces d'entrée. Après avoir déterminé le lien de sortie adéquat d'un paquet, le routeur place le paquet dans la file d'attente correspondante à ce lien de sortie. Supposons dans un premier temps que l'espace de stockage soit infini. Supposons également que la sortie soit un lien de capacité C, et que le trafic qui arrive en entrée soit autosimilaire de paramètres H, a et $m < C$. Norros a défini la notion suivante :

Définition 4.2 (Norros [Nor94]) *Le* stockage fractionnel autosimilaire brownien *de paramètres m, a et H, et de capacité de sortie $C > m$ est le processus stochastique X_t défini par*

$$X_t = \sup_{s \le t} \big(A(t) - A(s) - C\,(t-s) \big), \quad t \in]-\infty, \infty[\tag{4.9}$$

où $A(t)$ est le processus fractionnel brownien de l'équation (4.6).

La stationnarité de X_t vient de la stationnarité des incréments de $A(t)$. Selon le théorème d'ergodicité de Birkhoff [Dur96], X_t est fini puisque l'ergodicité de Z_t implique que $\lim_{t\to\infty} Z_t/t = E(Z_t) = 0$. Avec de plus l'hypothèse $m < C$ on obtient

$$\lim_{s \to -\infty} (A(t) - A(s) - C(t-s)) = -\infty \tag{4.10}$$

Remarque. Le processus de stockage X_t est toujours positif bien que les processus d'arrivée puissent avoir des incréments négatifs.

Les processus autosimilaires ont la propriété importante suivante :

Théorème 4.1 (Norros [Nor94]) *Soit X_t un stockage fractionnel autosimilaire brownien de paramètres H, a, m et C, et soit $\alpha > 0$ un nombre réel. Alors, le processus $X(\alpha t)$ est distribué comme $\alpha^H X_t$ avec un taux de service de $m + \alpha^{1-H}(C-m)$.*

Ce comportement vis-à-vis du changement d'échelle est central dans la théorie des processus autosimilaires. Nous rappelons donc ci-dessous les grandes lignes de la preuve de ce comportement.

Preuve. La distribution $X(\alpha t)$ se calcule comme suit

$$\begin{aligned} X(\alpha t) &= \sup_{s \le t} \big(A(\alpha t) - A(\alpha s) - C(\alpha t - \alpha s) \big) \\ &= \sup_{s \le t} \Big(m\alpha(t-s) + \sqrt{ma}\big(Z(\alpha t) - Z(\alpha s)\big) - C\alpha(t-s) \Big) \\ &\overset{(d)}{=} \alpha^H \sup_{s \le t} \Big(m\alpha^{1-H}(t-s) + \sqrt{ma}\big(Z(t) - Z(s)\big) - C\alpha^{1-H}(t-s) \Big) \\ &= \alpha^H \sup_{s \le t} \Big(A(t) - A(s) - \big(m + \alpha^{1-H}(C-m)\big)(t-s) \Big), \end{aligned} \tag{4.11}$$

où $\overset{(d)}{=}$ signifie que les processus impliqués ont la même distribution. On trouve que le taux de service C de l'équation (4.9) devient $m + \alpha^{1-H}(C-m)$ dans l'équation (4.11) par la comparaison de ces deux équations. ■

Un critère de qualité typique des télécommunications est la probabilité que le nombre d'unités de trafic dans une file d'attente excède la taille de la mémoire tampon b de cette file. Cette probabilité est un paramètre de qualité de service ϵ simplement défini comme :

$$\epsilon = P(X > b) \tag{4.12}$$

Cette équation exprime une condition de stockage. Norros a donné une expression explicite de cette condition en fonction des paramètres m, C, a et H.

Théorème 4.2 (Norros [Nor95]) *Soit $\rho = m/C$. L'expression suivante*

$$\frac{1-\rho}{\rho^{1/2H}} \cdot C^{(H-\frac{1}{2})/H} \cdot b^{(1-H)/H} \tag{4.13}$$

est une constante qui ne dépend que de H, a et ϵ (et pas de ρ, C, m ni b).

Ici encore, nous rappelons les grandes lignes de la preuve de cette propriété remarquable.

Preuve. Comme $X(0)$ a la même distribution que la fonction d'accumulation $sup_{s\leq t}\big(A(t) - C\,t\big)$, considérons la fonction

$$q(b,\beta) = P\Big(\sup_{t\geq 0}\big(Z(t) - \beta t\big) > b\Big). \tag{4.14}$$

En appliquant la propriété d'autosimilarité de $Z(t)$, on a

$$q(\alpha b, \beta) = P\left(\sup_{t\geq 0}\left[Z\left(\frac{t}{\alpha^{1/H}}\right) - \frac{\beta}{\alpha}t\right] > b\right) = q(b, \alpha^{\frac{1-H}{H}}\beta), \tag{4.15}$$

donc,

$$q(b,\beta) = q(1, b^{\frac{1-H}{H}}\beta) = f(b^{\frac{1-H}{H}}\beta), \tag{4.16}$$

où la fonction f est définie par

$$f(y) = q(1,y) = P\left(\sup_{t\geq 0}\left[Z(t) - yt > 1\right]\right). \tag{4.17}$$

La fonction $f(y)$ est strictement décroissante pour $y \geq 0$. De plus $f(0) = 1$ et $f(\infty) = 0$. On peut donc écrire

$$\begin{aligned}
\epsilon &= P(X > b) \\
&= P\left(\sup_{t\geq 0}\left[Z(t) - \frac{C-m}{\sqrt{am}}\,t > \frac{b}{\sqrt{am}}\right]\right) \\
&= f\left(\left(\frac{b}{\sqrt{am}}\right)^{\frac{1-H}{H}} \cdot \frac{C-m}{\sqrt{am}}\right)
\end{aligned} \tag{4.18}$$

En substituant $\rho = m/C$ nous obtenons l'équation désirée

$$\frac{1-\rho}{\rho^{1/2H}} \cdot C^{(H-\frac{1}{2})/H} \cdot b^{(1-H)/H} = a^{\frac{1}{2H}\cdot f^{-1}(\epsilon)} \tag{4.19}$$

■

Pour le cas $H = \frac{1}{2}$, l'équation (4.12) se réduit au cas de l'approximation de la file d'attente $M/D/1$, où

$$\frac{1-\rho}{\rho} \cdot b \tag{4.20}$$

est constant. En revanche, pour le cas $\frac{1}{2} < H < 1$ la situation est différente. Fixons C et cherchons la taille b de la mémoire tampon en fonction de ρ

$$b = b(\rho) = \Theta\left(\rho^{\frac{1}{2(1-H)}} \cdot (1-\rho)^{-\frac{H}{1-H}}\right) \tag{4.21}$$

Pour les valeurs élevées de H (par exemple $H = 0.8$ ou 0.9) l'espace de stockage nécessaire s'accroit très rapidement. Par conséquent l'augmentation de la taille de la file d'attente ne permet pas en pratique d'augmenter le facteur d'utilisation ρ. Résolvons l'équation (4.19) pour obtenir le taux de service C en fonction de l'utilisation $m = \rho \cdot C$. On obtient

$$C = C(m) = m + f^{-1}(\epsilon) a^{\frac{1}{2H}} b^{\frac{H-1}{H}} \cdot m^{\frac{1}{2H}} \tag{4.22}$$

L'équation (4.22) montre que la capacité du lien C augmente plus lentement qu'une loi linéaire de paramètre m. On peut tirer un meilleur profit de la capacité C en multiplexant plusieurs sources et sans augmenter la taille de la mémoire tampon b.

Le théorème suivant donne une borne inférieure pour l'équation 4.12, ce qui nous donnera une forme explicite de la fonction $f^{-1}(\epsilon)$ dans l'équation (4.19).

Théorème 4.3 (Norros [Nor94]) *Soit $X(t)$ un processus de stockage fractionnel autosimilaire brownien.*

$$P(X(t) > x) \geq \overline{\Phi}\left(\frac{1}{\sqrt{am}} \cdot \left(\frac{C-m}{H}\right)^{H} \cdot \left(\frac{x}{1-H}\right)^{1-H}\right) \tag{4.23}$$

où $\overline{\Phi} = P(Z(1) > y)$ est la distribution résiduelle de la fonction gaussienne.

Voici ci-dessous les grandes lignes de la preuve.

Preuve. Comme dans la preuve du théorème 4.2, on inverse le temps. Alors,

$$P(X > b) \geq \max_{t \geq 0} P\big(A(t) > t + b\big) = \max_{t \geq 0} \overline{\Phi}\left(\frac{(1-m)\, t + b}{\sqrt{am}\, t^{H}}\right), \tag{4.24}$$

où l'égalité vient de l'autosimilarité de $Z(t)$. Le maximum de l'équation (4.24) est obtenu par différenciation, lorsque

$$t = \frac{H\, x}{(1-H)(1-m)}. \tag{4.25}$$

■

Dans la suite du document, nous utiliserons l'approximation classique suivante :

$$\overline{\Phi}(y) \approx \frac{(2\pi)^{-1/2}}{(1+y)} \exp\left(\frac{-y^2}{2}\right) \sim \exp\left(\frac{-y^2}{2}\right). \tag{4.26}$$

L'expression de la probabilité de l'équation 4.12 devient alors

$$P(X > x) \sim \exp\left(-\frac{(C-m)^{2H} \cdot x^{2-2H}}{2H^{2H} \cdot (1-H)^{2-2H} \cdot a \cdot m}\right). \tag{4.27}$$

Le comportement de la file d'attente est Weibullien, c'est-à-dire $P(X > b) \sim \exp(-\gamma x^{\beta})$, $\beta \leq 1$. La valeur du paramètre H a donc un grand impact sur le critère de stockage, et il met en évidence la différence avec la modélisation traditionnelle du trafic basée sur des statistiques poissonniennes.

4.1.3 Congestion probabiliste

Les rappels nécessaires de la théorie des processus autosimilaires ayant été faits (cf. les sections 4.1.1 et 4.1.2), nous pouvons maintenant définir un nouveau paramètre de qualité de service, relié au taux de perte de paquets dans le réseau, facteur critique dans `Internet`. Nous montrerons également comment ce paramètre peut être estimé en temps réel par chaque routeur. Sa définition découle de la probabilité de pertes :

Définition 4.3 *La* congestion probabiliste $\mathcal{CP}$ *d'un lien de capacité* C*, possédant une mémoire tampon de taille* b*, et recevant un trafic autosimilaire de paramètres* m*,* a *et* H*, est défini comme*

$$\mathcal{CP} = -\ln\left(1 - \exp\left(-\frac{(C-m)^{2H} \cdot b^{2-2H}}{2H^{2H} \cdot (1-H)^{2-2H} \cdot a \cdot m}\right)\right) \tag{4.28}$$

La congestion probabiliste vérifie donc $\mathcal{CP} = -\ln(1-P(X(t) > b))$, où le terme $1-P(X(t) > b)$ est la probabilité que la capacité de la mémoire tampon ne soit pas dépassée. Autrement dit, la congestion probabiliste est la probabilité de ne pas perdre de données. L'utilisation du logarithme est motivée par deux raisons. D'une part, cette représentation atténue les grandes variations de la probabilité. D'autre part, cette représentation simplifie les calculs. Par exemple, la congestion probabiliste d'un chemin simple $S = \{e_1, \ldots, e_k\}$ est calculée comme $\mathcal{CP}_S = \sum_{i=1}^{k} \mathcal{CP}_i$. En effet, la probabilité totale de ne pas perdre un élément d'information dans le chemin S est bien le produit des probabilités de ne pas perdre un élément d'information dans chaque lien e_i. Cette dernière affirmation est vraie sous l'hypothèse d'événements indépendants, ce qui se justifie par le fait que le réseau soit "ouvert", c'est-à-dire que les nœuds reçoivent et envoient du trafic de l'extérieur du réseau, et que le nombre de paquets perdus est petit en comparaison de la totalité du trafic.

La congestion probabiliste d'un lien peut être estimée localement par le routeur incident à ce lien. Elle repose sur une estimation des paramètres du trafic qui peut être effectuée avec les statistiques des données transmises par chaque routeur du réseau. Par exemple, Abry et Veitch [AV98] et Roughan, Veitch et Abry [RVA00] proposent un estimateur robuste, non

décentré et rapide (basé sur une batterie de filtres numériques) pour estimer le paramètre H. L'estimateur utilise la théorie des ondelettes pour faire une analyse multi-résolution de la variance des données. L'estimation de H obtenue est très bonne mais nécessite 30.000 données (voir la figure 4 dans [RVA00]). Quand le nombre des données disponibles n'est pas très important, il est préférable d'utiliser l'estimateur suivant qui utilise le fait que chaque routeur garde les statistiques de l'utilisation de chacune de ses interfaces (par exemple il compte le nombre d'octets qui sont envoyés sur chaque interface).

Estimation des paramètres du trafic Pour chaque lien, on obtient donc la mesure du nombre d'octets A_i, $1 \leq i \leq N$, envoyés pendant l'intervalle de temps $I_i =]t-it_u, t-(1-i)t_u[$, où t_u est le temps entre deux observations, et N est le nombre total d'observations. Pour faire l'estimation des paramètres du modèle autosimilaire au temps t, un routeur se sert des N observations faites durant un temps $\Delta = N \cdot t_u$.

Soit $B_i = A_i/t_u$. L'estimation de m et a est obtenue par

$$\widehat{m} = \quad \overline{B} \quad = \quad \frac{1}{N} \quad \sum_{i=1}^{N} B_i \tag{4.29}$$

$$\widehat{a} = \frac{\sigma^2(B)}{\widehat{m}} = \frac{1}{\widehat{m}(N-1)} \sum_{i=1}^{N} B_i - \overline{B} \tag{4.30}$$

Pour l'estimation de H, on utilise le fait (cf. [LTWW93] que le trafic des réseaux suit des processus autosimilaires de deuxième ordre possédant une fonction d'auto-covariance de type $\alpha \cdot x^{-\beta}$, où $H = 1-\beta/2$. On estimera donc H par régression linéaire afin de minimiser l'erreur quadratique moyenne. Plus précisément, la fonction d'auto-covariance est

$$\check{B}_k = \sum_{i=1}^{N} (B_i - \overline{B}) \cdot (B_{i-k} - \overline{B}) \tag{4.31}$$

Si $N = 2^j$, et $j \in \mathbb{N}$, il est possible d'utiliser la transformée rapide de Fourier (FFT) de la manière suivante

$$\check{B} = FFT^{-1}(FFT(B) \circ FFT(B^*)) \tag{4.32}$$

où $\check{B} = \{B_1, B_2, \ldots, B_N\}$, et le symbole "$\circ$" est la multiplication composantes à composantes[3], et B^* est le vecteur conjugué du vecteur B, toujours composantes à composantes. La régression linéaire s'obtient par

$$\check{B}_k = \alpha I^{-\beta}$$
$$\ln \check{B}_k = \ln \alpha - \beta \ln I$$

3. c'est-à-dire $D = F \circ G$ satisfait $D_i = F_i \cdot G_i$ pour tout $1 \leq i \leq N$

En écriture matricielle

$$\begin{bmatrix} \log \check{B}_1 \\ \vdots \\ \log \check{B}_N \end{bmatrix} = \begin{bmatrix} 1 & \ln I_1 \\ \vdots & \vdots \\ 1 & \ln I_N \end{bmatrix} \begin{bmatrix} \ln \alpha \\ -\beta \end{bmatrix}$$

$$\ln \check{B} = Z \begin{bmatrix} \ln \alpha \\ -\beta \end{bmatrix}$$

$$Z^T \ln \check{B} = Z^T Z \begin{bmatrix} \ln \alpha \\ -\beta \end{bmatrix}$$

$$\left[Z^T Z\right]^{-1} Z^T \ln \check{B} = \begin{bmatrix} \ln \alpha \\ -\beta \end{bmatrix} \tag{4.33}$$

On fait le calcul pour $\left[Z^T Z\right]^{-1}$:

$$Z^T Z = \begin{bmatrix} 1 & \dots & 1 \\ \ln I_1 & \dots & \ln I_N \end{bmatrix} \begin{bmatrix} 1 & \ln I_1 \\ \vdots & \vdots \\ 1 & \ln I_N \end{bmatrix}$$

$$= \begin{bmatrix} N & \sum_{i=1}^N \ln I_i \\ \sum_{i=1}^N \ln I_i & \sum_{i=1}^N (\ln I_i)^2 \end{bmatrix}$$

$$\left[Z^T Z\right]^{-1} = \frac{\begin{bmatrix} \sum_{i=1}^N (\ln I_i)^2 & -\sum_{i=1}^N \ln I_i \\ -\sum_{i=1}^N I_i & N \end{bmatrix}}{N \sum_{i=1}^N (\ln I_i)^2 - (\sum_{i=1}^N \ln I_i)^2} \tag{4.34}$$

En utilisant (4.34) en (4.33)

$$\begin{bmatrix} \ln \alpha \\ -\beta \end{bmatrix} = \frac{\begin{bmatrix} \sum_{i=1}^N (\ln I_i)^2 & -\sum_{i=1}^N \ln I_i \\ -\sum_{i=1}^N \ln I_i & N \end{bmatrix}}{N \sum_{i=1}^N (\ln I_i)^2 - (\sum_{i=1}^N \ln I_i)^2} \begin{bmatrix} 1 & \dots & 1 \\ \ln I_1 & \dots & \ln I_N \end{bmatrix} \ln \check{B}$$

Soit

$$det = N \sum_{i=1}^N (\ln I_i)^2 - \left(\sum_{i=1}^N \ln I_i \right)^2$$

Alors, finalement

$$\begin{bmatrix} \ln \alpha \\ -\beta \end{bmatrix} = \begin{bmatrix} \frac{\sum_{i=1}^N \ln I_i \sum_{i=1}^N \ln \check{B}_i - \sum_{i=1}^N \ln I_i \sum_{i=1}^N \ln I_i \ln \check{B}_i}{det} \\ \frac{-\sum_{i=1}^N \ln I_i \sum_{i=1}^N \ln \check{B}_i + N \sum_{i=1}^N \ln I_i \sum_{i=1}^N \ln I_i \ln \check{B}_i}{det} \end{bmatrix} \tag{4.35}$$

De l'équation (4.35) on obtient β, et $\widehat{H}$ comme

$$\widehat{H} = 1 - \frac{\beta}{2} \tag{4.36}$$

Pour évaluer la qualité de l'estimateur proposé ci-dessus, nous avons généré des traces autosimilaires avec le générateur proposé par Norros, Mannersalo et Wang [NMW99]. Ces auteurs ont validé leur générateur avec l'estimateur d'Abry et Veitch [AV98]. Nous avons généré 1000 traces de 2^{16} valeurs avec les paramètres du simulateur $m = 50$ et $n = 25$ pour obtenir des traces bien représentatives des valeurs de $H = \{0.7, 0.8, 0.9\}$. Pour utiliser l'estimateur proposé ci-dessus il faut tenir compte du fait que le calcul de l'auto-corrélation est une approximation d'autant moins valable que l'on s'éloigne de l'origine de l'axe des abscisses. Le paramètre N_e utilisé pour faire l'estimation est donc plus petit que le paramètre N des valeurs du trafic original. Dans nos expériences nous avons utilisé $N_e = \frac{1}{2}N$. La figure 4.1 présente des résultats pour les traces complètes (appelées "sans moyenne"), modifiées en prenant les moyennes par groupes de 2^5 valeurs (appelées "moyenne sur 32"), et modifiées en prenant les moyennes par groupes de 2^{10} valeurs (appelées "moyenne sur 1024"). Notons que pour la trace "moyenne sur 32", $N = 2048$. Pour la trace "moyenne sur 1024", $N = 64$. Ces valeurs sont motivées par l'estimation de H à partir des traces du système MRTG [wwwa]. Nous reviendrons sur ce point dans la section 4.2.2. Bien que l'estimateur ne soit pas parfait, il permet toutefois d'estimer H quand le nombre d'observations N est petit. En plus, nous verrons dans la section 4.2.4 que le paramètre de qualité de service proposé (la congestion probabiliste) est robuste face à des variations de H. Il est donc possible d'accepter une certaine imprécision dans l'estimation de H. On observe que la déviation standard augmente avec le nombre de valeurs moyennées, ce qui est raisonnable du fait de la grande diminution des valeurs de N_e pour faire le calcul. L'estimation de $H = 0.9$ reste la plus difficile à cause de l'estimation de l'auto-corrélation. La dérivation standard est cependant plus petite car l'auto-corrélation est plus lissée que pour les valeurs de H plus petites.

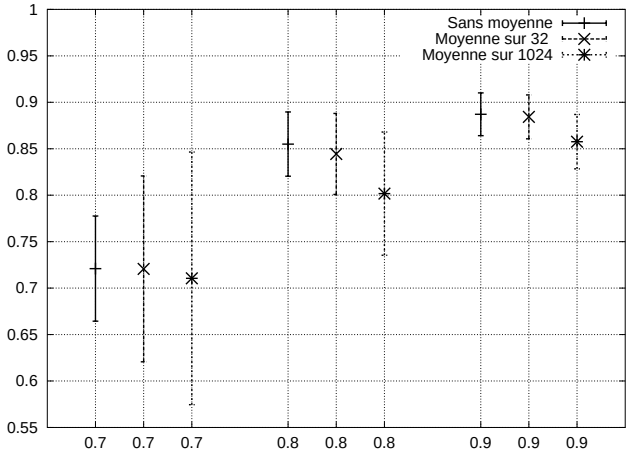

FIG. 4.1: *Estimations de H selon différentes moyennes.*

4.2 Application de la congestion probabiliste

L'objet de cette section est de démontrer l'utilité de la congestion probabiliste comme nouveau paramètre de qualité de service. Nous allons ainsi montrer comment utiliser la congestion probabiliste pour la construction optimisée d'arbre de multicast, soit en l'utilisant comme critère de sélection de chemins dans QoSMIC [FBP98], soit comme paramètre de congestion dans notre protocole MCT (voir chapitre 3.3). Le chapitre ci-après décrit en détail d'une part les raisons qui motivent l'utilisation de la congestion probabiliste dans les protocoles de multicast, et d'autre part comment ce paramètre de qualité de service peut être effectivement utilisé. La section 4.2.2 décrit ensuite le cadre de la longue campagne de simulation que nous avons menée pour valider notre utilisation de la congestion probabiliste. Ces simulations (sections 4.2.3 et 4.2.4) considèrent des simulations de topologies réelles à partir de données réelles, et aussi des modèles de topologies (telles que celles du chapitre 2.3). Ce dernier type de simulations nous a permis d'aborder des réseaux réalistes de très grande taille, et de juger ainsi du passage à l'échelle de notre approche.

4.2.1 Congestion probabiliste et applications multipoint

Les transmissions sur les réseaux de paquets IP ne sont pas fiables. En effet un paquet peut être détruit en passant par un routeur congestionné. Une autre caractéristique des réseaux IP est qu'ils sont partagés entre tous les utilisateurs. Pour offrir le service de transport, la pile de protocoles TCP/IP contient deux protocoles qui assurent différents types de services : TCP et UDP.

TCP est un protocole qui permet des communications fiables, et se comporte comme un canal virtuel entre les deux utilisateurs d'une communication point-à-point. Ce protocole assure que l'information arrive à destination dans le même ordre que celui dans lequel elle a été envoyée, et sans erreurs. Pour accomplir cela, TCP utilise un mécanisme d'acquittement. Pour chaque morceau d'information envoyé il y a un acquittement[4]. Si le nœud émetteur ne reçoit pas l'acquittement au bout d'un certain temps, il renvoie ce morceau d'information. Le temps entre l'envoi de l'information et la réception de l'acquittement est utilisé par le contrôle de congestion qui régule son débit. Ce mécanisme de contrôle de congestion aux extrémités de la communication permet de bien partager le réseau entre utilisateurs. S'il y a un nouveau utilisateur qui arrive, il pourra envoyer des donnés avec une qualité raisonnable parce que les autres diminueront leur débit automatiquement.

Toutes les applications qui ont besoin de fiabilité utilisent des communications TCP. Par exemple la transmission de fichiers avec le protocole `FTP`, ou la plupart des connexions aux serveurs Web.

UDP est un protocole sans connexion et qui n'assure pas la fiabilité. L'information est envoyée sous la forme de datagrammes, et il n'y a aucun contrôle de son arrivée à destination (ce contrôle est laissé aux applications). Comme UDP est un protocole sans connexion, il n'y a

4. Le processus est un peu plus complexe que cela parce qu'il n'est pas nécessaire d'envoyer un acquittement pour chaque paquet envoyé. Pour une explication plus détaillée, voir [Ste94].

pas de contrôle de congestion. Un exemple d'application d'UDP est le service de noms "DNS". Ce service associe les noms des stations sur `Internet` à leur numéro IP. Une demande peut être envoyée dans un seul datagramme ou paquet. Si le paquet est perdu, la source refait la demande après un temps prédéterminé. Dans ce cas c'est l'application qui résout le problème de la perte du paquet. De plus, ni l'ordre, ni les doublons ne sont pris en compte. Un autre exemple d'utilisation d'UDP est la transmission de la voix en temps réel. Les données sont envoyées à la vitesse nécessaire. Si un paquet est perdu, le temps de renvoyer le paquet est normalement plus grand que temps que le récepteur peut attendre pour bien entendre la transmission. Donc, dans cette application les paquets perdus ne sont pas renvoyés.

Évidemment la perte de paquets dégrade la qualité d'une transmission. Par exemple, une perte sous TCP provoque un ralentissement de l'envoi des données, causé par les retransmissions, et par l'action du mécanisme de contrôle de congestion qui diminue le débit. Dans UDP la perte de paquets influe directement sur la qualité de la transmission (par exemple dans la transmission de la voix). Trouver un mécanisme pour réduire les pertes est donc très important. Les pertes se produisent dans les routeurs. Un routeur reçoit un paquet, il le met dans la file d'attente du lien de sortie selon le destinataire. Si la file d'attente est pleine, le routeur résout cette situation de congestion en détruisant le paquet, et parfois en envoyant une notification à la source (protocole ICMP).

Comme il a été présenté à la section 4.1 le trafic a un comportement autosimilaire dans les réseaux IP. On peut ainsi estimer la probabilité de pertes en utilisant l'équation 4.27 déduite par Norros. On a donc montré dans la section 4.1.3 qu'il est possible d'estimer les paramètres du trafic en temps réel. Donc on a tous les éléments pour calculer la probabilité de perte, et chercher une méthode pour réduire ces pertes.

D'un point de vue général il est possible d'utiliser la congestion probabiliste pour le routage. Par exemple, on peut calculer un plus court chemin dans le réseau modélisé par un graphe où le poids des arêtes est la congestion probabiliste. Il reste à démontrer que cette technique donne un résultat stable, c'est-à-dire qu'il n'y a pas d'oscillations entre les routes dans les tables de routage. Il faut noter qu'un changement de route change le trafic, et donc la congestion probabiliste des arêtes impliquées change également. Ce sont ces changements corrélés qui peuvent générer des oscillations. Ce travail n'est pas abordé ici et dans cette thèse on considère uniquement le cas où le trafic sur lequel on applique les critères de QoS est significativement plus petit que le trafic maximal dans le réseau. Cette hypothèse permet d'assurer que les conditions de congestion des routeurs ne changeront pas à cause de l'optimisation du trafic sur lequel on applique les critères de QoS. Cette affirmation est vraie pour des trafics négligeables par rapport au trafic maximal.

Pour une application point-à-point, il est possible de chercher un chemin tel que la congestion probabiliste soit petite afin de satisfaire aux besoins de l'application. Par exemple, le protocole RSVP [ZBHJ97] (pour "Resource ReSerVation Protocol") permet de réserver un certain paramètre de QoS tout au long d'un chemin. Évidemment pour que cette réservation soit possible il faut avoir la capacité d'explorer plusieurs chemins, pas seulement le plus courts. En particulier, le protocole MPDR proposé à la section 3.3.1 est capable d'explorer de multiples chemins. Utilisé avec RSVP, il trouvera éventuellement un chemin avec la probabilité de perte désirée.

Pour une application multipoint, on a vu dans la section 3.1.10 qu'il existe des protocoles multipoint avec QoS, dont YAM et QoSMIC. Les techniques de construction des arbres sont identiques dans les deux protocoles. Le deux se servent d'une méthode gloutonne pour chercher différents chemins de connexion à l'arbre multipoint déjà existant. (QoSMIC ajoute une autre technique : il permet que les nœuds de l'arbre cherchent différents chemins vers la source). Les deux protocoles comparent les chemins et choisissent celui qui a la QoS demandée. Dans le travail que nous présentons, nous avons simulé le comportement de QoSMIC avec la congestion probabiliste comme paramètre de qualité de service. Nous avons aussi fait des simulations pour faire des comparaisons avec notre protocole multipoint MCT présenté à la section 3.3.2.

Nos simulations ont pour objectif la comparaison de protocoles multipoint avec capacité de QoS, avec des protocoles sans QoS. Les paramètres de QoS sont liés à la bande passante, le taux de perte, le délai, etc. Dans nos simulations, toutes les comparaisons on été faites en mesurant le nombre d'octets perdus dans l'arbre multipoint.

4.2.2 Impact de l'utilisation de la congestion probabiliste

Ce chapitre décrit le cadre des résultats obtenus lors de notre longue campagne de simulations menée afin de valider les idées développées dans la section précédente. L'idéal aurait été de mener des simulations à partir de trafic réel sur des topologies réelles. Malheureusement il est très difficile (voire impossible) d'obtenir tout le trafic d'un réseau. Il est cependant possible d'obtenir des traces de trafic (par exemple les valeurs moyennes du trafic pendant une période de temps) de certains réseaux réels. Nous avons mené une campagne de simulations sur la topologie du réseau UUNET et sur une topologie générée par le modèle de la section 2.3. Nous n'avons pas obtenu les traces correspondant à ce réseau. Nous avons donc extrapolé à partir de traces réelles sur le réseau Abilene aussi bien pour UUNET que pour notre modèle de topologie.

Nous commençons par décrire le calcul de la congestion probabiliste, puis la génération du trafic dans nos différentes simulations. Nous décrivons ensuite la représentation des résultats.

Calcul de la congestion probabiliste

Pour calculer la congestion probabiliste décrite dans la définition 4.3, on a besoin des paramètres du trafic, des liens et des routeurs. Considérons un lien donné d'un routeur donné. Les paramètres du trafic de ce lien sont représentés par le triplet $\{\hat{m}_t, \hat{a}_t, \hat{H}_t\}$ qui décrit le trafic au temps t. Les autres paramètres sont la capacité C du lien, et la taille b de la mémoire tampon que le routeur assigne à ce lien.

Comme il a été montré dans la section 4.1.3, les paramètres du trafic peuvent être calculés en temps réel. Les simulations faites utilisent un trafic autosimilaire généré à partir de traces réelles obtenues avec le système MRTG [wwwa]. Le format MRTG est standard pour les données de trafic et il donne le nombre total d'octets envoyés sur une interface d'un routeur, par période de temps de 5 minutes. Nous nous sommes donc heurté ici à une première

difficulté. D'une part, pour faire l'estimation des paramètres du trafic, on doit prendre un nombre de données suffisamment important. Mais d'autre part, l'intervalle de temps doit être tel que l'hypothèse d'autosimilaritée reste valable. Nous avons décidé de prendre 50 observations. Selon les notations de la section 4.1.3, on a donc :

$$t_u = 300 \text{ secondes}, \ N = 50, \text{ et } \Delta = 4\texttt{h}10.$$

Le période Δ est un peu grande car le trafic peut changer de caractéristique sur une aussi longue période, mais elle permet toutefois d'avoir un nombre suffisant d'observations du trafic pour une bonne estimation de H. A partir de l'estimation des paramètres du trafic de tous les liens selon la section 4.1.3, on obtient la congestion probabiliste de tous les liens au temps t par application de l'équation (4.28). Il faut remarquer que pour faire l'estimation de paramètres dans un routeur réel, il est préférable d'utiliser une période Δ plus petite, et un nombre N d'observations plus grand. Dans cette thèse nous ne rentrons pas dans les détails de l'estimation des paramètres.

Les autres paramètres que nous devons considérer sont la capacité du lien C et la taille b de la mémoire tampon. C est facile à obtenir. Par contre les fabricants de routeurs ne donnent pas la valeur de b, et ils utilisent souvent une politique dynamique pour la gestion de la mémoire. Toutefois, la variation de la congestion probabiliste en fonction de de b n'est pas très importante. En plus, si on fait une simulation de file d'attente en faisant varier le paramètre b, on s'aperçoit que le nombre de paquets perdus se maintient quasi constant. Cette affirmation est vraie pour b au-dessus d'une limite inférieure. Au-dessous de cette limite, le nombre de paquets perdus augmente rapidement lorsque b diminue. Dans la figure 4.2 on montre deux liens très congestionnés du réseau Abilene.

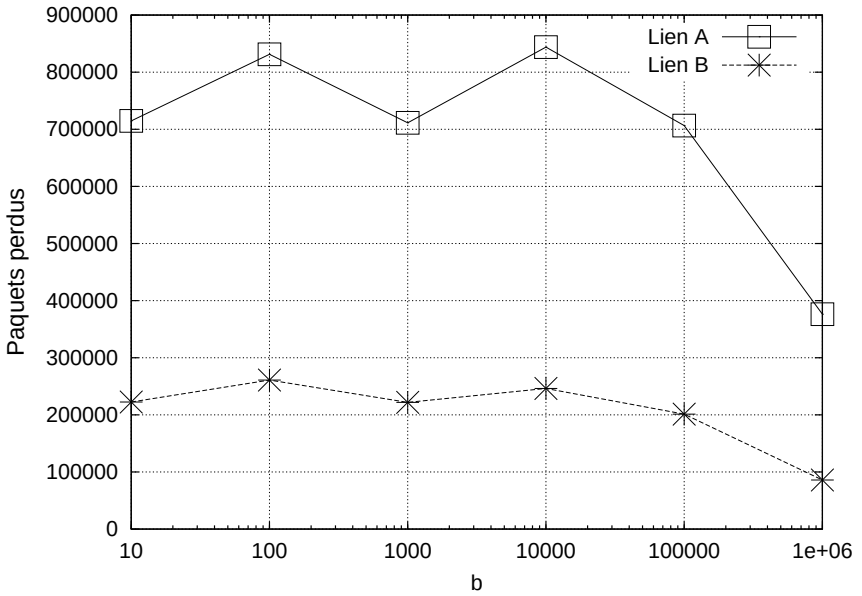

FIG. 4.2: *Paquets perdus en fonction de la taille de b sur une période de 8192 secondes.*

Il faut comprendre que la taille de la mémoire tampon b est réservée aux données qui attendent d'être envoyées. Par conséquent on a choisi une valeur de $b = 4800$ octets pour

toutes les simulations. Cette valeur représente une borne supérieure de la taille maximale d'un paquet pour les technologies de réseau *Ethernet*, *Fast-Ethernet*, et *FDDI*.

Génération d'un trafic autosimilaire

Dans nos simulations, nous avons simulé la file d'attente de chaque lien de la manière suivante. Nous avons généré le trafic A_t selon l'équation (4.6), ainsi que les triplets $\{\hat{m}_t, \hat{a}_t, \hat{H}_t\}$. Les protocoles utilisent la congestion probabiliste calculé au temps $t - t_u$, donc l'estimation des paramètres du trafic a été faite à $t - t_u$. Ensuite, pour chaque lien e, nous avons considéré le processus itératif suivant :

$$Y_{k\tau} = (m - C_e)\tau + \sqrt{am}\,(Z_{k\tau} - Z_{(k-1)\tau})$$

et

$$X_{k\tau} = \begin{cases} 0 & \text{si } X_{(k-1)\tau} + Y_{k\tau} \leq 0; \\ b & \text{si } X_{(k-1)\tau} + Y_{k\tau} \geq b; \\ X_{(k-1)\tau} + Y_{k\tau} & \text{sinon.} \end{cases}$$

où C_e est la capacité du lien e, τ est la résolution de la simulation (c'est-à-dire le tic de l'horloge), et b la taille de la mémoire tampon du lien e. Nous avons calculé ensuite le rapport entre le nombre d'octets perdus pendant tout la durée de l'expérience, et le nombre total d'octets transmis par le lien e. Ce rapport est vu comme la probabilité p_e de perdre un octet dans le lien e. On peut alors estimer la probabilité de perdre un octet pour un nouveau flux de débit très inférieur au débit maximal, comme la même probabilité p_e.

Les protocoles multipoint construisent un arbre $T = (V, E)$, où V est la suite des nœuds, et E est l'ensemble des liens. Pour chaque arbre T construit par un protocole multipoint, la probabilité de ne pas perdre d'octet est

$$P(T) = \Pi_{e \in E}(1 - p_e) \tag{4.37}$$

La validité de cette expression est basée sur la supposition que les flux qui traversent le réseau sont indépendants. La justification est donnée par le fait que le réseau est ouvert, c'est-à-dire que chaque nœud ou routeur reçoit du trafic de l'extérieur et envoie du trafic vers l'extérieur. D'autre part le nombre de flux (connexions TCP) est élevé à cause de son caractère de réseau "backbone".

Représentation des résultats

Nous avons représenté nos résultats par une fonction de distribution F correspondant à la probabilité de perte de paquets. Une abscisse est donc une probabilité de perte p, et l'ordonnée correspondante est le pourcentage d'expériences telles que la probabilité de perte soit au plus p :

$$F(p) \simeq P(q \leq p), \tag{4.38}$$

où q est la probabilité de perte. Si deux fonctions F_1 et F_2 correspondent à deux protocoles de multicast M_1 et M_2, respectivement, alors $F_1(p) \leq F_2(p)$ signifie que M_2 est "meilleur" que M_1, en p. En effet la probabilité que l'arbre T_2 retourné par M_2 ait un taux de perte au plus p est supérieur à la probabilité que l'arbre T_1 retourné par M_1 ait un taux de perte au plus p. En autres termes T_1 a plus de chance d'avoir un taux de perte élevé que T_2. Pour pouvoir comparer M_1 et M_2 plus sûrement, il faut que $F_1(p) \leq F_2(p)$ pour tout p (sauf essentiellement pour p très petit, ou pour p très grand). Nos simulations permettent généralement de discriminer les différents protocoles suivant ce critère.

Pour mesurer l'amélioration d'un protocole M par rapport à un autre protocole M', nous avons utilisé la fonction suivante :

$$R(p) = \frac{F^{-1}_{\mathcal{M}'}(p)}{F^{-1}_{\mathcal{M}}(p)} \tag{4.39}$$

où $F^{-1}_{\mathcal{M}}$ est l'inverse de la fonction F pour le protocole multipoint $\mathcal{M}$. La fonction R compare ainsi les protocoles multipoint $\mathcal{M}$ et $\mathcal{M}'$. Pour un même pourcentage p d'expériences, $R(p)$ décrit le rapport de la probabilité d'avoir un taux de perte $\leq p$ dans le protocole $\mathcal{M}'$ sur la probabilité d'avoir un taux de perte $\leq p$ du protocole $\mathcal{M}$. $R(F)$ est plus grand que 1 lorsque le protocole $\mathcal{M}$ est meilleur que le protocole $\mathcal{M}'$ en p. Pour obtenir une valeur scalaire de l'amélioration, nous avons utilisé

$$\mu(R) = \frac{1}{N_R} \sum_{k=1}^{N_R} R\left(\frac{k}{N_R}\right) \tag{4.40}$$

$$\sigma^2(R) = \frac{1}{N_R} \sum_{k=1}^{N_R} \left(R\left(\frac{k}{N_R}\right) - \mu(R)\right)^2 \tag{4.41}$$

où N_R est le nombre de points des fonctions de répartition. $\mu(R)$ est la moyenne et $\sigma^2(R)$ est la variance. On peut interpréter $\mu(R)$ comme la moyenne sur toutes les probabilités p, de l'amélioration de $\mathcal{M}$ par rapport à $\mathcal{M}'$ en p.

4.2.3 Simulations sur le réseau UUNET

Le réseau UUNET est un réseau d'un fournisseur mondial d'accès à `Internet`. L'intérêt de ce réseau pour nos simulations réside dans sa taille. Il permet la construction de grands arbres, contenant un grand nombre d'utilisateurs, ce qui nous permettra de vérifier l'utilité de la congestion probabiliste par rapport aux autres mesures de qualité de service. La taille de la topologie d'UUNET nous permettra également d'étudier les performances de notre protocole MCT, défini dans la section 3.5. Nous avons obtenu la description du cœur de ce réseau à partir d'un projet de l'association CAIDA [wwwb]. Ce cœur d'UUNET possède 129 nœuds et 321 liens, pour un diamètre de 7. Le degré moyen est $4,98$.

Nous n'avons pas pu obtenir les données du trafic réel d'UUNET. Nous avons donc généré des données de trafic en extrapolant celles du réseau Abilene. Abilene fait partie du projet `Internet` II pour développer les nouvelles technologies de communication [wwwc]. Quelques protocoles de communication multipoint, principalement MBGP et DVMRP, peuvent s'utiliser dans ce réseau. Nous avons extrait le trafic de la partie centrale du réseau ("backbone") pour faire nos simulations. Le backbone d'Abilene est un réseau de 11 nœuds et de 14 liens, pour un diamètre de 5. Nous avons obtenu des données de trafic dans le format MRTG [wwwa]. Nous avons généré 389 triplets $\{\hat{m}_t, \hat{a}_t, \hat{H}_t\}$ avec tous les intervalles de temps Δ non superposés. Comme il y a 14 liens bidirectionnels, nous avons donc obtenu plus de 10.000 triplets réalistes au total. Pour générer du trafic réaliste sur UUNET, nous normalisons chaque triplet selon la capacité C_e du lien e d'Abilene correspondant, pour obtenir des triplets de la forme $\{\hat{m}/C_e, \hat{a}/C_e, \hat{H}\}$. Nous avons créé ensuite 100 instances de trafic en choisissant au hasard des triplets $\{\hat{m}, \hat{a}, \hat{H}\}$ pour tous les liens du réseau d'UUNET. Chaque triplet a été dé-normalisé en multipliant par la capacité du lien d'UUNET correspondant. Nous avons choisi de faire l'assignation des triplets obtenus à partir de données réelles pour satisfaire à l'interdépendance des paramètres H, a et m du modèle. Il a été en effet observé que de grandes valeurs de H correspondent à de grandes valeurs de a, mais on ne connaît pas de relation explicite entre ces paramètres.

Commençons par les protocoles qui n'utilisent que des techniques RSP et gloutonne, sans QoS. La figure 4.3 montre le comportement (c'est-à-dire les fonctions de répartition) des techniques RSP et gloutonne pour deux groupes de 10 et 40 membres dans UUNET. On observe sur cette figure que les performances de ces deux techniques sont très similaires, en fait presque équivalentes. La table 4.1 donne les valeurs de la fonction R pour les deux techniques dans quatre groupes de tailles différents, et confirme cette observation, c'est-à-dire $\mu(R) \simeq 1$ pour tous les groupes. Les taux de pertes moyen donnés dans la table 4.1 permettent également de vérifier la similitude de RSP et glouton. Dans la suite, les deux techniques donnant des résultats similaires, nous n'utiliserons que RSP pour les comparaisons avec les autres protocoles. (Nous avons vérifié que la technique gloutonne donne presque toujours des résultats comparables a RSP.)

Taille du groupe		10	20	40	80
RSP		$4.35\ 10^{-7}$	$6.10\ 10^{-7}$	$9.49\ 10^{-7}$	$17.56\ 10^{-7}$
Glouton		$4.28\ 10^{-7}$	$6.26\ 10^{-7}$	$9.70\ 10^{-7}$	$18.81\ 10^{-7}$
RSP / Glouton	$\mu(R) =$	1.36	1.03	0.99	0.92
	$\sigma(R) =$	1.00	0.21	0.15	0.07

TAB. 4.1: Comparaison entre RSP et Glouton.

Nous allons maintenant évaluer le comportement de QoSMIC avec différents paramètres de QoS. La qualité de service peut être représentée par différents paramètres comme la bande passante libre, le délai, le taux de pertes, etc. La bande passante libre est un paramètre particulièrement utilisé. Il permet en effet non seulement de juger si une application peut opérer avec la portion de bande passante souhaité, mais aussi de diminuer la probabilité de pertes en choisissant le lien de plus grande bande passant libre.

Sur la figure 4.4, nous montrons des simulations de QoSMIC avec bande passante libre pour

des groupes de 10 et de 40 membres respectivement (QoSMIC avec bande passante libre est noté QoSMIC-AB pour "available bandwidth"). Nous avons considéré deux estimations de la bande passante libre, soit la moyenne sur 4h10, soit sur 5mn. A partir de la figure 4.4 et de la table 4.2, nous pouvons conclure que l'intervalle de temps le plus long semble le mieux adapté pour estimer la bande passante libre. Le paramètre $\mu(R)$ pour 5mn/4h10 est en effet plus grand que 1, et donc, selon l'équation 4.39, l'intervalle de 4h10 est préférable. Cela est confirmé par les taux de perte moyens donnés dans la table 4.2.

Taille du groupe		10	20	40	80
4h10		$3.87\ 10^{-7}$	$5.32\ 10^{-7}$	$9.33\ 10^{-7}$	$17.10\ 10^{-7}$
5mn		$4.07\ 10^{-7}$	$6.14\ 10^{-7}$	$10.30\ 10^{-7}$	$18.27\ 10^{-7}$
5mn / 4h10	$\mu(R) =$	1.03	1.10	1.13	1.11
	$\sigma(R) =$	0.18	0.19	0.09	0.13

TAB. 4.2: Moyenne sur un intervalle de temps long et sur un autre court

Les figures 4.5 et 4.6 montrent le comportement de QoSMIC avec les paramètres de qualité de service suivants : bande passante libre (QoSMIC-AB) et congestion probabiliste (QoSMIC-PC). Ces mêmes figures décrivent également les performances du protocole RSP. La table 4.3 montre la fonction R ainsi que les taux de perte moyens pour ces trois protocoles. On constate une diminution du taux de perte entre 11% et 27% pour QoSMIC-AB avec bande passante libre comparé à RSP (voir le groupe de 80, $\mu(R) = 1.11$, donc 11% d'amélioration). Mais l'amélioration pour QoSMIC-PC avec congestion probabiliste comparé à QoSMIC-AB avec bande passante libre est bien plus significative. Dans le groupe de 10 on voit dans la table 4.3 que l'écart-type est énorme. La fonction répartition $F(p)$ pour QoSMIC-PC est cependant toujours au dessus de celle de QoSMIC-AB. Dans ce cas, l'écart-type important n'est pas significatif.

Taille du groupe		10	20	40	80
QoSMIC-AB (capacité disponible)		$3.87\ 10^{-7}$	$5.32\ 10^{-7}$	$9.33\ 10^{-7}$	$17.10\ 10^{-7}$
QoSMIC-PC (congestion probabiliste)		$3.11\ 10^{-7}$	$4.03\ 10^{-7}$	$4.70\ 10^{-7}$	$8.05\ 10^{-7}$
RSP / QoSMIC-AB	$\mu(R) =$	1.27	1.27	1.19	1.11
	$\sigma(R) =$	0.32	0.22	0.28	0.14
QoSMIC-AB / QoSMIC-PC	$\mu(R) =$	6.30	2.69	3.72	2.85
	$\sigma(R) =$	18.5	2.34	3.35	1.78

TAB. 4.3: Les deux versions de QoSMIC comparées contre RSP

Nous étudions finalement le protocole de communication multipoint MCT. Nous avons indiqué sur les figures 4.7 et 4.8 une comparaison de QoSMIC-PC avec MCT et RSP, pour des groupes de 10, 20, 40 et 80 membres. La table 4.4 correspond à ces expériences. On observe une amélioration importante de MCT par rapport à QoSMIC-PC, les deux protocoles ayant la congestion probabiliste comme paramètre de QoS. (Les résultats pour les groupes de 10 et 20 membres sont extrêmement grands, car la fonction répartition de MCT monte presque verticalement au début. Ces résultats ne sont pas représentatifs, mais ils montrent que MCT a un comportement supérieur.) L'amélioration de QoSMIC-PC par rapport au RSP est aussi très importante.

Remarque. Si la taille du groupe augmente, le comportement de QoSMIC-PC se rapproche de celui de MCT. Il est vrai que le cas de 80 membres est un peu limite dans un réseau avec 129 nœuds seulement.

Taille du groupe		10	20	40	80
MCT		$0.74\ 10^{-7}$	$1.33\ 10^{-7}$	$2.75\ 10^{-7}$	$4.73\ 10^{-7}$
QoSMIC-PC / MCT	$\mu(R) =$	221	51.6	4.57	2.20
	$\sigma(R) =$	429	249	3.32	0.68
RSP / QoSMIC-PC	$\mu(R) =$	6.81	3.35	5.06	3.37
	$\sigma(R) =$	15.2	2.47	6.08	3.20

TAB. 4.4: Comparaison entre QoSMIC et MCT, et QoSMIC et RSP.

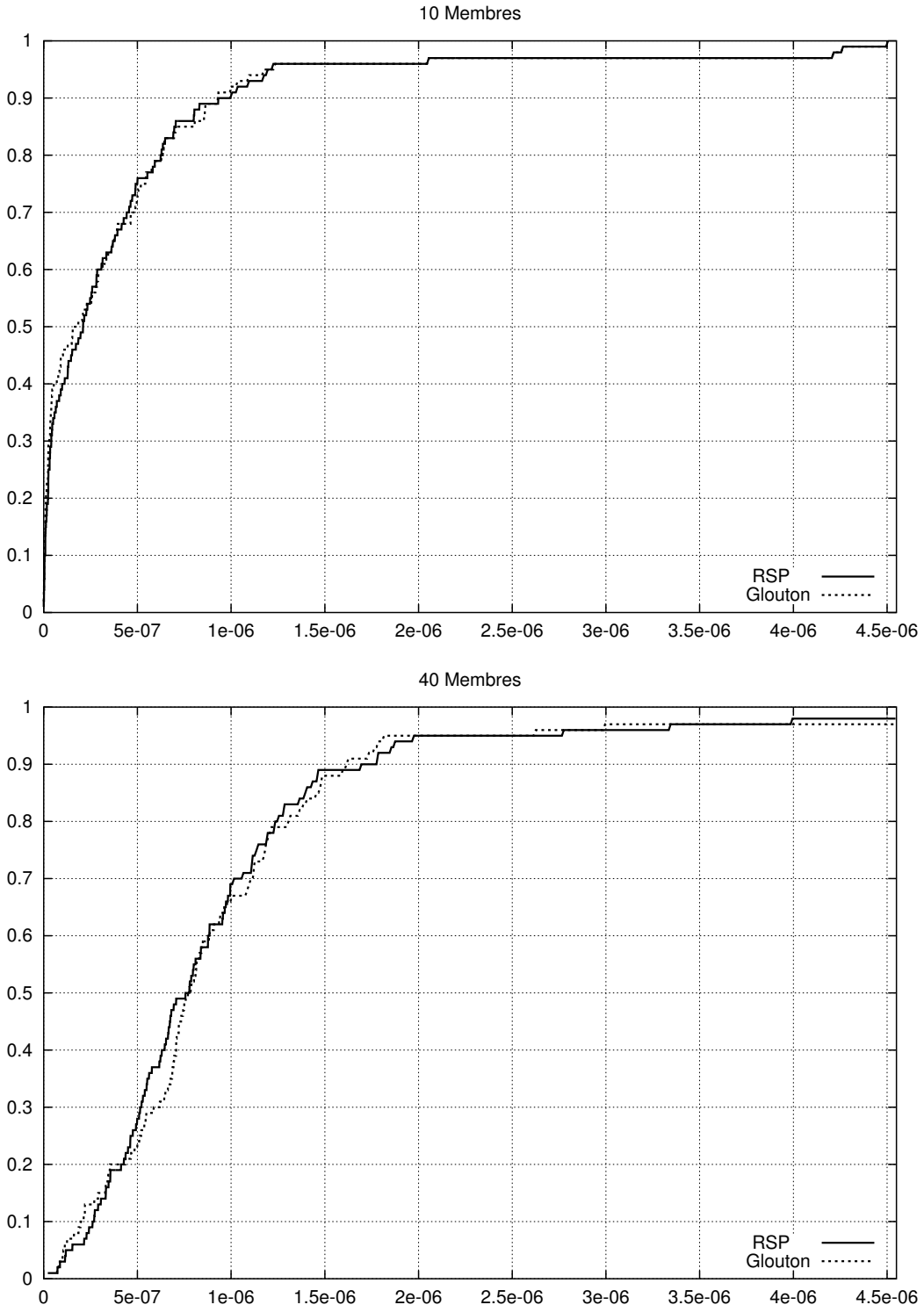

FIG. 4.3: *UUNET : comparaison entre RSP et Glouton, groupe de* 10 *et* 40 *membres.*

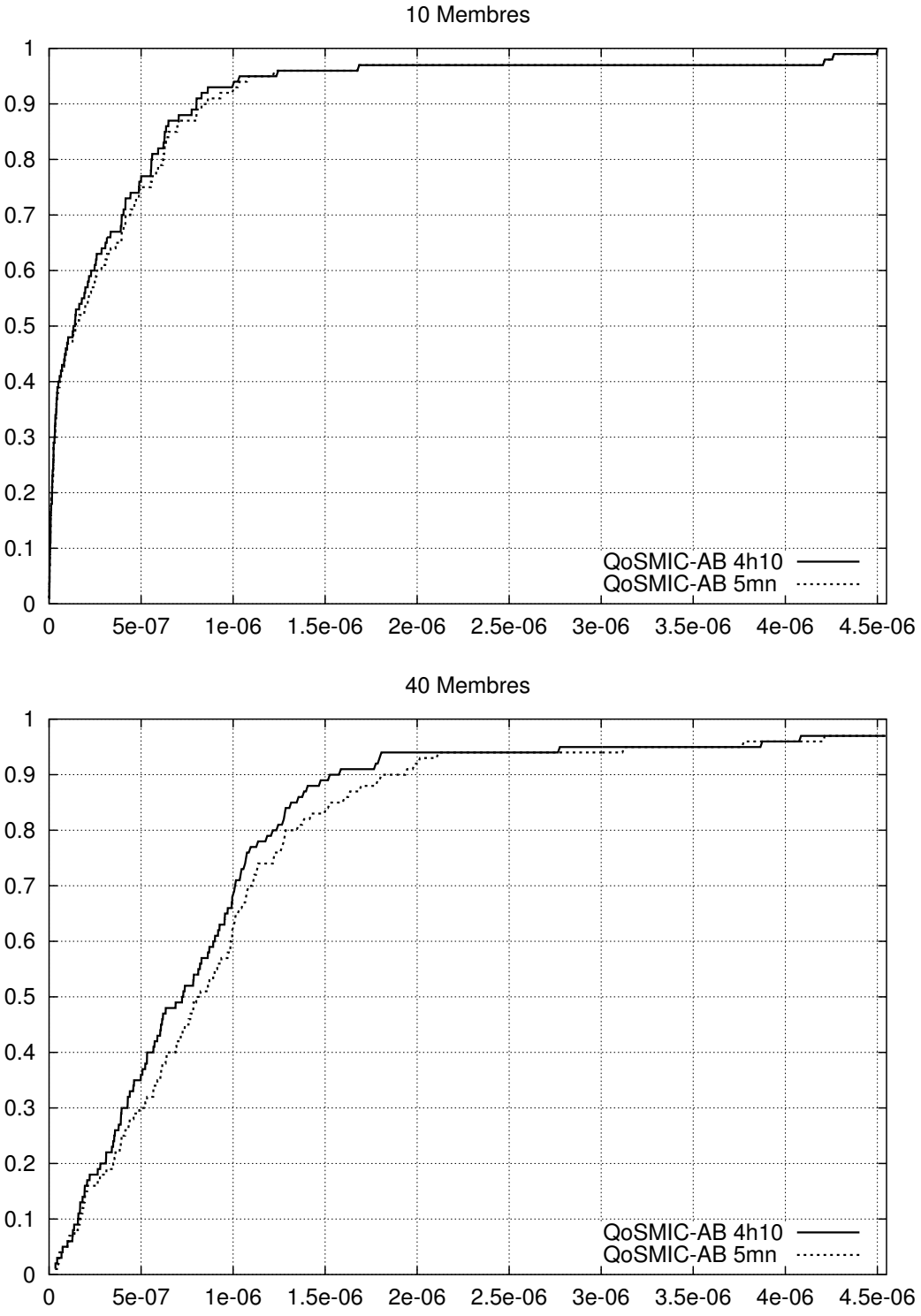

FIG. 4.4: *UUNET : comparaison entre QoSMIC-AB sur* 5*mn et* 4*h*10*, groupe de* 10 *et* 40 *membres.*

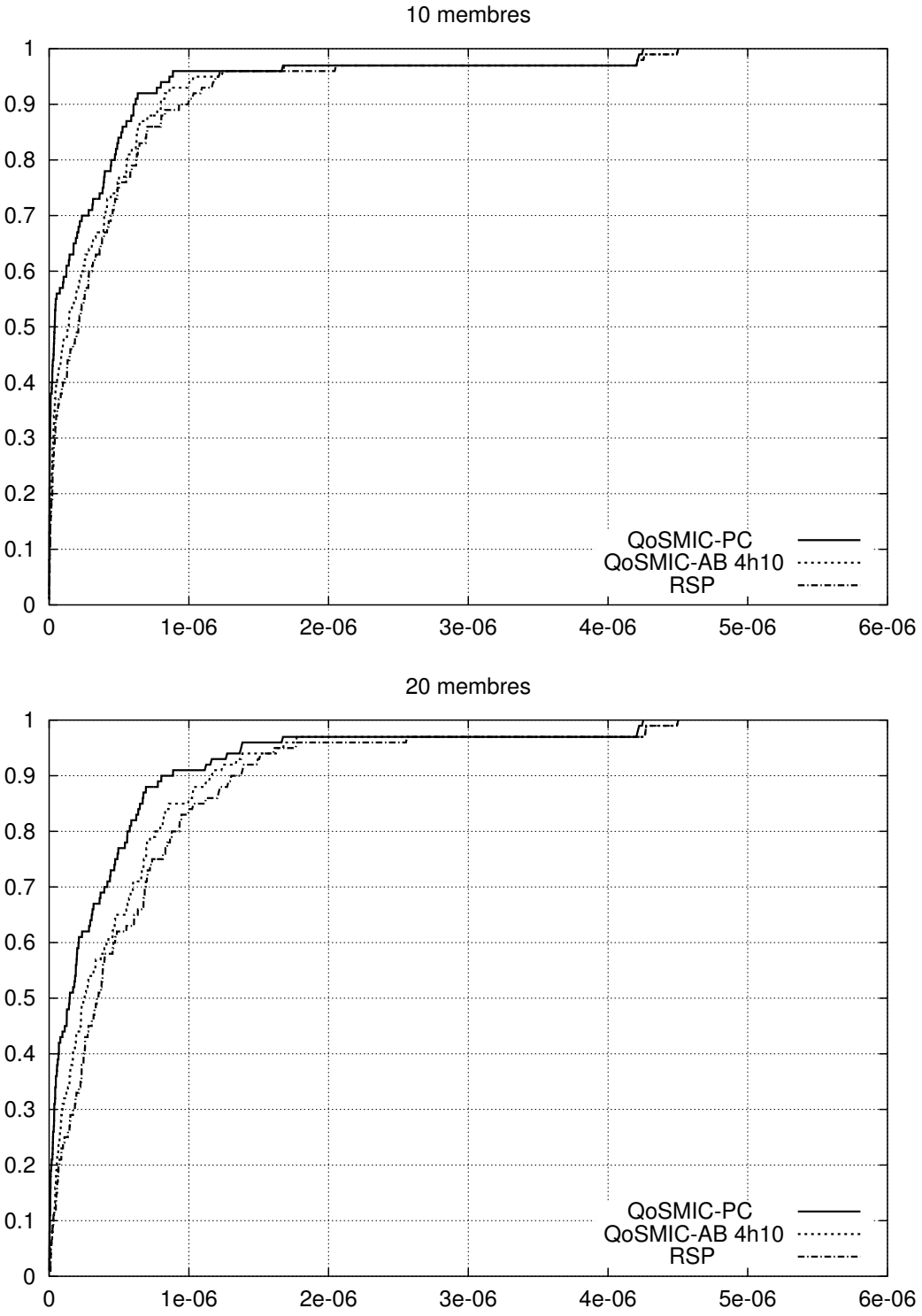

FIG. 4.5: *UUNET : comparaison entre QoSMIC et RSP, groupes de* 10 *et* 20 *membres.*

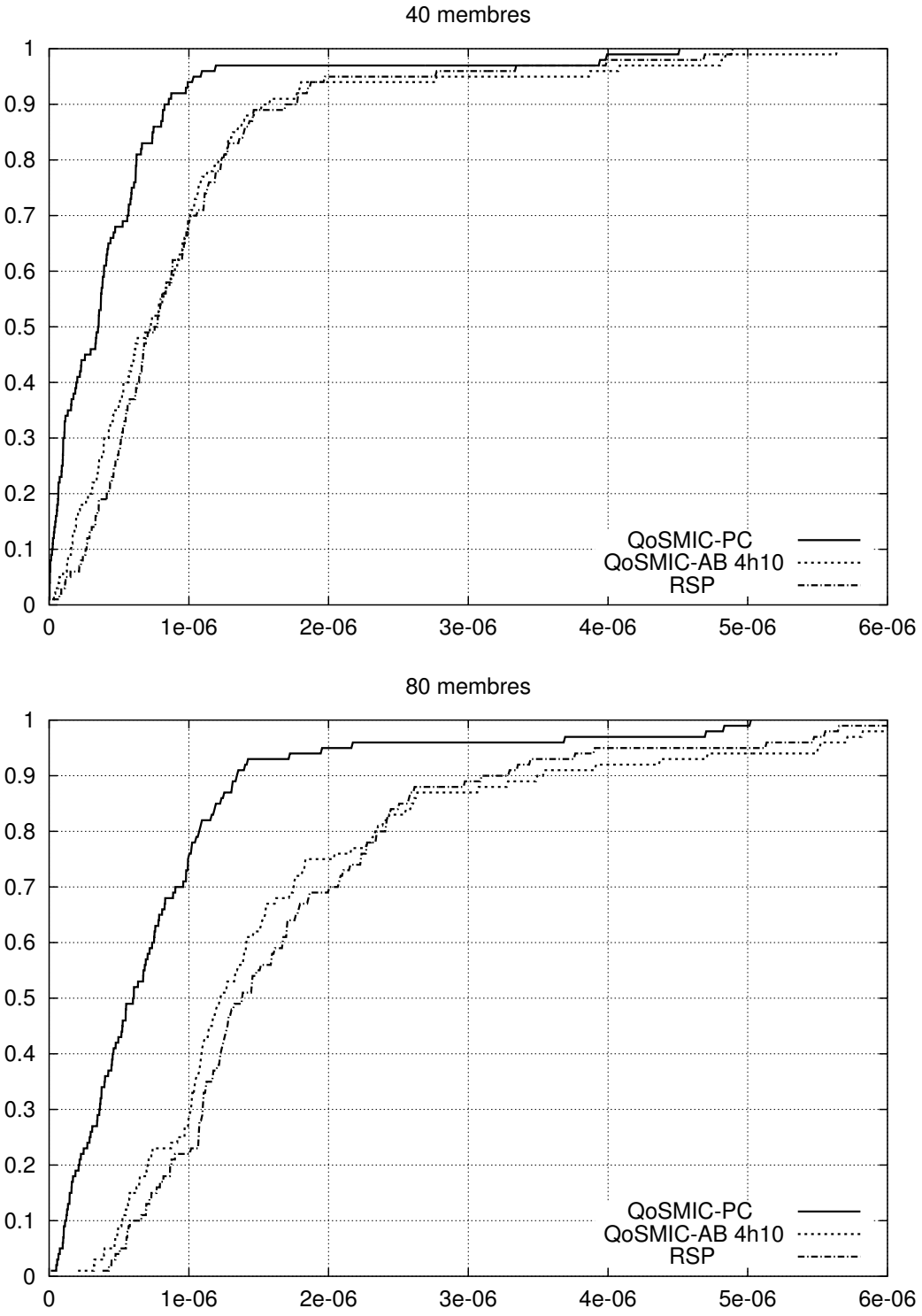

FIG. 4.6: *UUNET : comparaison entre QoSMIC et RSP, groupes de* 40 *et* 80 *membres.*

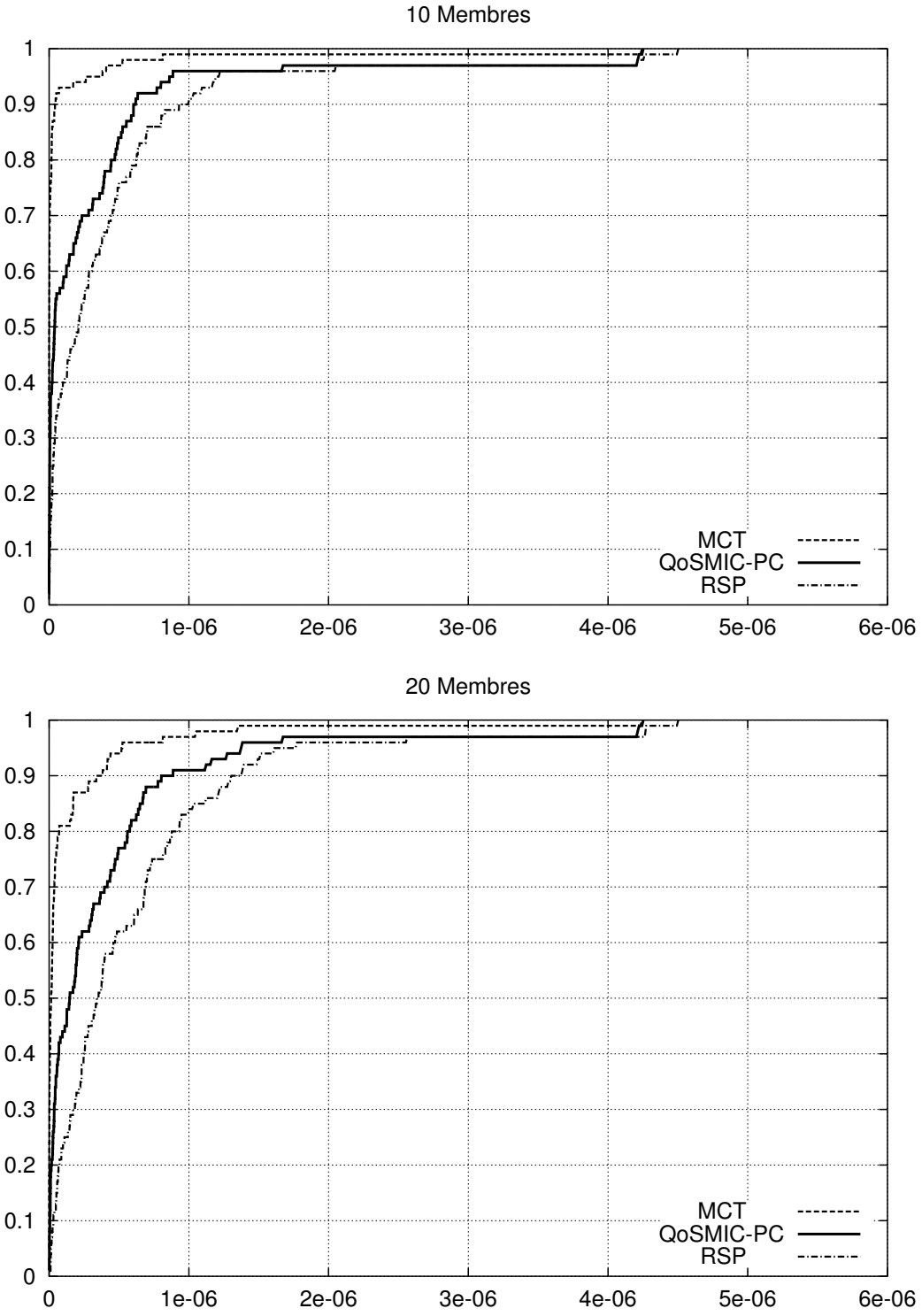

FIG. 4.7: *UUNET : comparaison entre MCT et QoSMIC, groupes de* 10 *et* 20 *membres.*

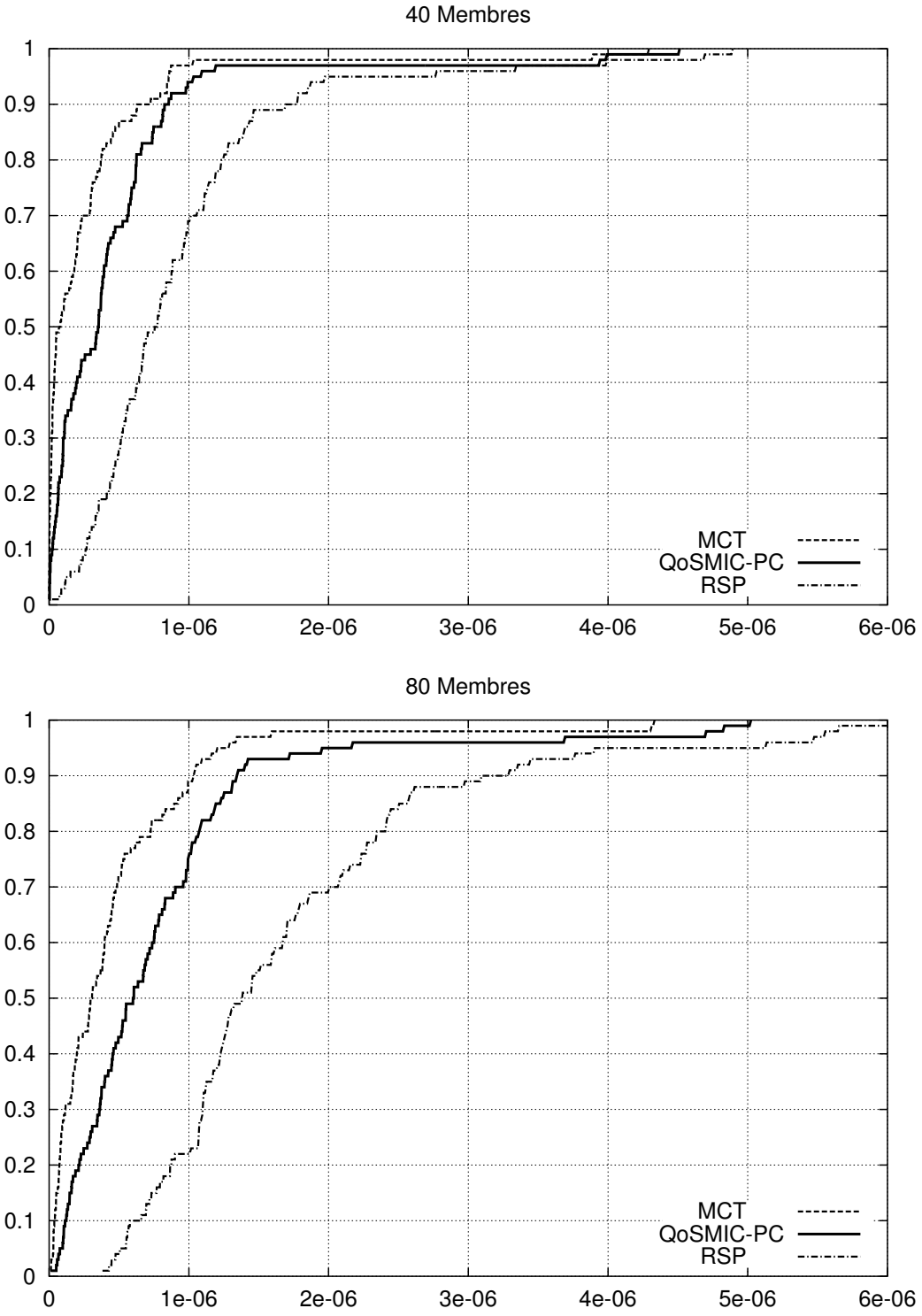

FIG. 4.8: *UUNET : comparaison entre MCT et QoSMIC, groupes de* 40 *et* 80 *membres.*

4.2.4 Simulations sur notre modèle d'`Internet` proposé

Dans cette section nous présentons des résultats de simulations des différents protocoles multipoint sur une topologie générée selon le modèle proposé dans la section 2.3. La topologie utilisée a été obtenue avec les paramètres $m = 1$, $k = 1$, $\xi = \gamma = 0.001$. Elle a 500 nœuds, 1990 liens, le degré moyen est 3.98, et le diamètre est 12. On peut voir sa représentation à la figure 2.4.

Nous avons simulé les protocoles RSP, glouton, QoSMIC et MCT. Pour les protocoles avec QoS nous avons simulé les cas avec la bande passante libre, appelé "AB", et la congestion probabiliste, appelé "PC". Les membres groupes utilisés ont été tiré au hasard avec un probabilité uniforme entre tous les nœuds du réseau.

Comme il a été dit dans la section 4.2.2, nous avons calculé la congestion probabiliste du trafic au temps t avec les paramètres estimés $\hat{m}_t, \hat{a}_t$ et $\hat{H}_t$ au temps $t - t_u$. Le trafic utilisé dans la simulation a été généré avec les paramètres du trafic estimés au temps t. Cela nous permet de simuler la situation réelle où la congestion a été estimée avant le moment où elle est utilisée. Cependant, l'estimation de H que nous avons utilisée est biaisée. Donc, nous avons fait le calcul de la congestion probabiliste avec H_{err} tel que H_{err} a $\pm 10\%$ d'erreur par rapport à H. Cette erreur a été introduite artificiellement avec une loi uniforme et elle simule l'erreur dans l'estimation de H. Même si l'estimation de H a une certaine erreur, la congestion probabiliste continue à représenter plus ou moins bien les pertes dans les routeurs. La figure 4.9 montre les performances de MCT (avec $\varrho = 1$ et $r = 1$) et de QoSMIC avec congestion probabiliste, pour un groupe de 50 membres, et pour le cas de H et de H_{err}. La table 4.5 montre les valeurs du taux de perte moyen, de $\mu(R)$ et de $\sigma(R)$ pour MCT comparé avec RSP, et pour QoSMIC comparé avec RSP aussi. Bien qu'il y ait des différences, les tendances restent inchangées.

	MCT(H)	$1.61 \cdot 10^{-6}$		QoSMIC(H)	$2.05 \cdot 10^{-6}$
RSP / MCT(H)	$\mu(R) =$	6.24	RSP / QoSMIC(H)	$\mu(R) =$	2.46
	$\sigma(R) =$	2.94		$\sigma(R) =$	0.61
	MCT(H_{err})	$1.57 \cdot 10^{-6}$		QoSMIC(H_{err})	$2.09 \cdot 10^{-6}$
RSP / MCT(H_{err})	$\mu(R) =$	4.48	RSP / MCT(H_{err})	$\mu(R) =$	2.28
	$\sigma(R) =$	1.39		$\sigma(R) =$	0.45

TAB. 4.5: *Comparaison de l'utilisation de la congestion probabiliste sans erreur H, et avec erreur H_{err}*

Ensuite, on a comparé l'exécution de MCT avec $\varrho = r = 1$, et MCT avec $\varrho = 1$ et $r = 2$ (cf. la figure 4.10). Le nombre de messages pour $\varrho = r = 1$ est de 2.363, tandis que pour $\varrho = 1$ et $r = 2$ le nombre de messages est 18.155. Pour comparer, le nombre de messages de QoSMIC pour construire l'arbre pour le même groupe multipoint est 1.806, et pour la méthode gloutonne, il est seulement de 1.237. Bien qu'il y ait une différence importante de performance entre les différentes valeurs de ϱ et r, on utilisera MCT avec $\varrho = r = 1$ parce que le nombre de messages est un ordre de magnitude plus petit et assez proche de celui de QoSMIC. Donc, pour le reste des simulations on utilisera MCT avec la congestion probabiliste estimée avec H_{err}, et avec $\varrho = 1$ et $r = 1$.

Nous avons ensuit généré différents groupes de taille 10, 30, 50 et 100 membres pour les techniques RSP et gloutonne, et les protocoles QoSMIC et MCT. Le protocole QoSMIC a été testé avec la bande passante disponible "AB" avec moyennes sur 4h10 ou 5mn, et avec la congestion probabiliste "PC". Le protocole MCT a été seulement testé avec la congestion probabiliste.

Taille du groupe		10	30	50	100
RSP / Glouton	$\mu(R) =$	1.13	1.08	0.98	1.02
	$\sigma(R) =$	0.34	0.08	0.09	0.07
RSP / QoSMIC-AB $4h10$	$\mu(R) =$	1.45	1.35	1.47	1.42
	$\sigma(R) =$	0.50	0.24	0.33	0.24
QoSMIC-AB 5mn 5mn / QoSMIC-AB 4h10	$\mu(R) =$	0.95	1.05	1.13	1.10
	$\sigma(R) =$	0.10	0.09	0.14	0.13
QoSMIC-AB $4h10$ / QoSMIC-PC	$\mu(R) =$	1.35	1.50	1.57	1.50
	$\sigma(R) =$	0.24	0.21	0.17	0.12
QoSMIC-PC / MCT	$\mu(R) =$	4.42	12.74	1.93	4.43
	$\sigma(R) =$	10.1	79.91	0.30	10

TAB. 4.6: *Comparaison entre les différents protocoles multipoint*

Les résultats sont montrés à la table 4.6, et dans les figures 4.11, 4.12 et 4.13.

Dans la figure 4.11 on montre les groupes de 10 et 50 membres. Dans le groupe de 50 membres on peut distinguer, de droite à gauche, d'abord deux protocoles avec un comportement similaire (RSP et glouton), puis QoSMIC-AB 5mn et 4h10, puis QoSMIC avec congestion probabiliste, et finalement MCT avec congestion probabiliste. Ce schéma se répète dans toutes les figures. De plus, dans la table 4.6 on peut vérifier ce comportement. Pour RSP et glouton tous les groupes sont presque équivalents ($\mu(R) \simeq 1$). Pour QoSMIC-AB 5mn et 4h10, on note une certaine amélioration pour QoSMIC-AB $4h10$. Pour QoSMIC-PC et QoSMIC-AB 5mn on constate que QoSMIC-PC a au minimum 35% d'amélioration pour tous les cas. Donc la congestion probabiliste a réussi à améliorer significativement le taux de perte. Finalement, MCT a au minimum une amélioration de 93%, avec seulement 31% plus de paquets générés pour la construction de l'arbre que QoSMIC. (L'écart-type est toujours additif.)

Le figure 4.12 montre deux cas de groupes de 30 membres. En haut un groupe choisi parmi tous les nœuds du réseau. En bas un groupe choisi parmi les nœuds de plus grand degré, c'est-à-dire ceux qui sont dans le noyau du réseau. Il est intéressant de constater que tous les protocoles avec QoS ont une meilleure performance dans le cas du groupe choisi dans le noyau du réseau. Finalement les résultats pour le groupe de 100 membres est montré à la figure 4.13. C'est un cas extrême d'utilisation mais il montre que, d'une part les protocoles avec QoS ont encore une bonne performance, et d'autre part la congestion probabiliste trouve toujours les arbres avec le moins de pertes.

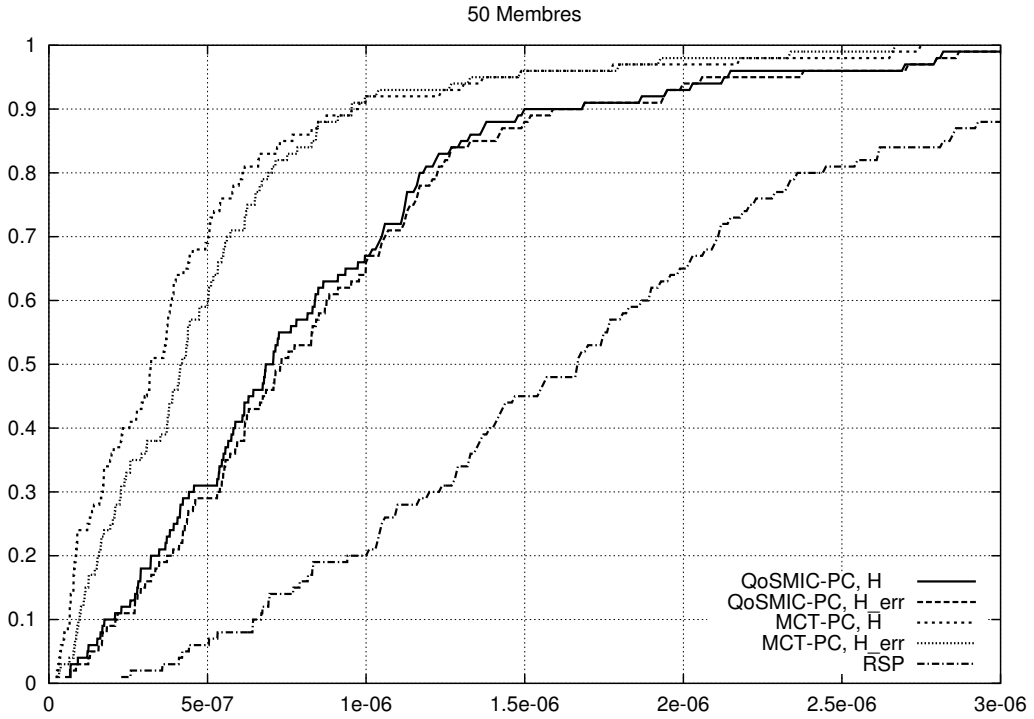

FIG. 4.9: *Comparaison de la probabilité de pertes pour différentes estimations de H*

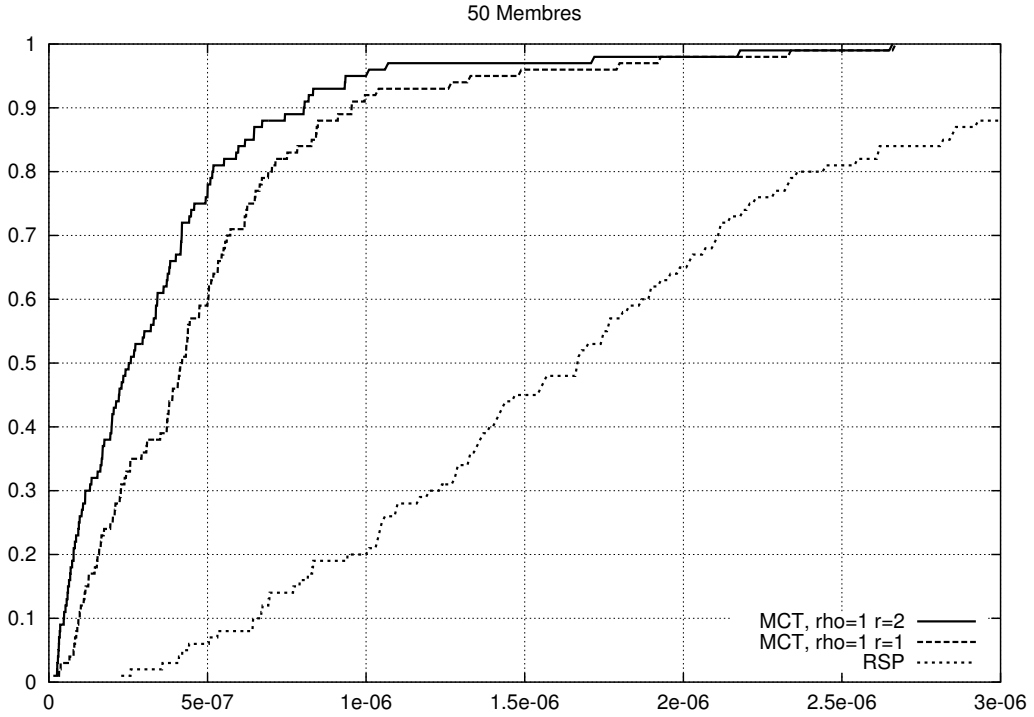

FIG. 4.10: *Comparaison de la probabilité de pertes pour différentes estimations de H*

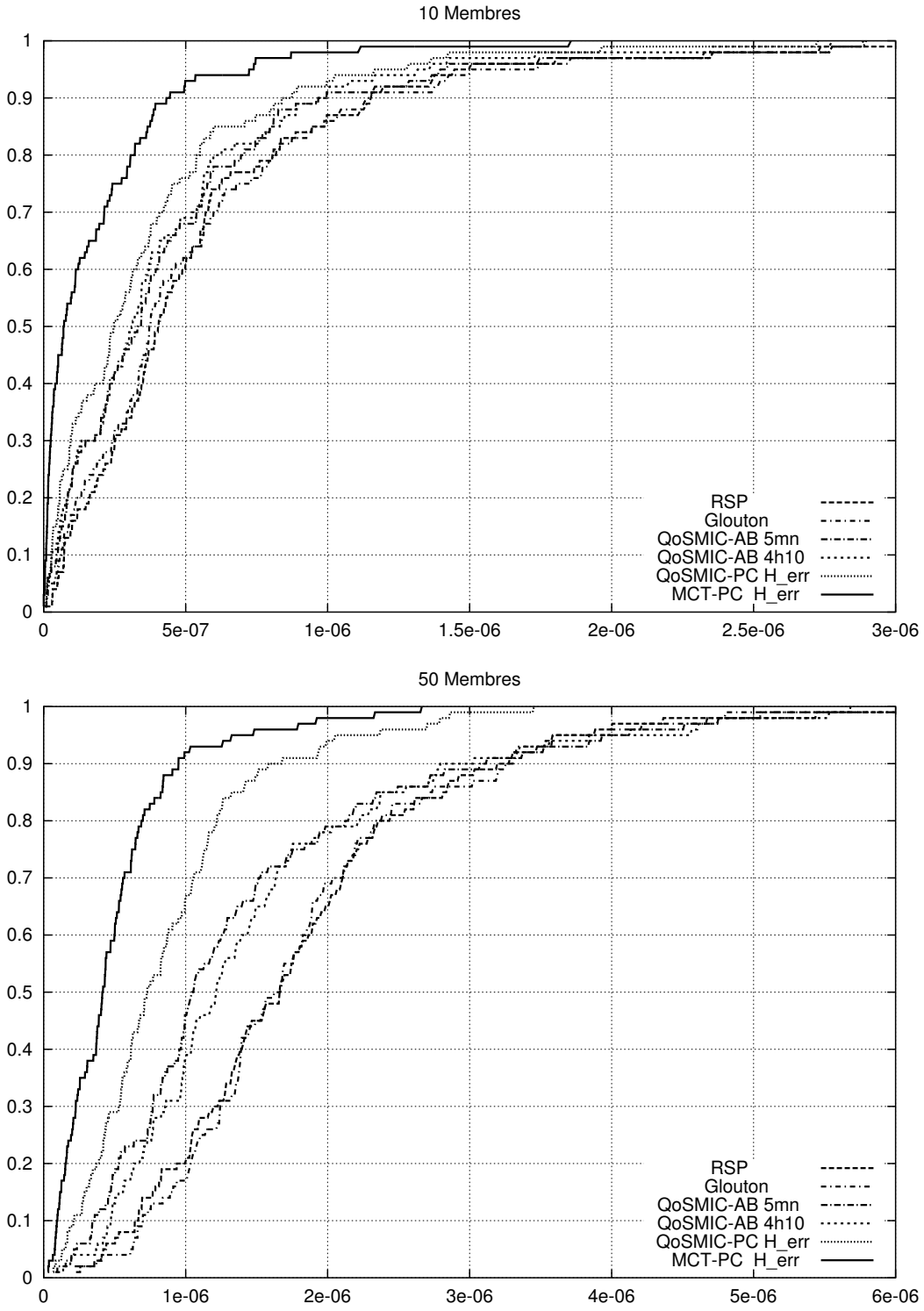

FIG. 4.11: *Comparaison des probabilités de perte pour différents protocoles*

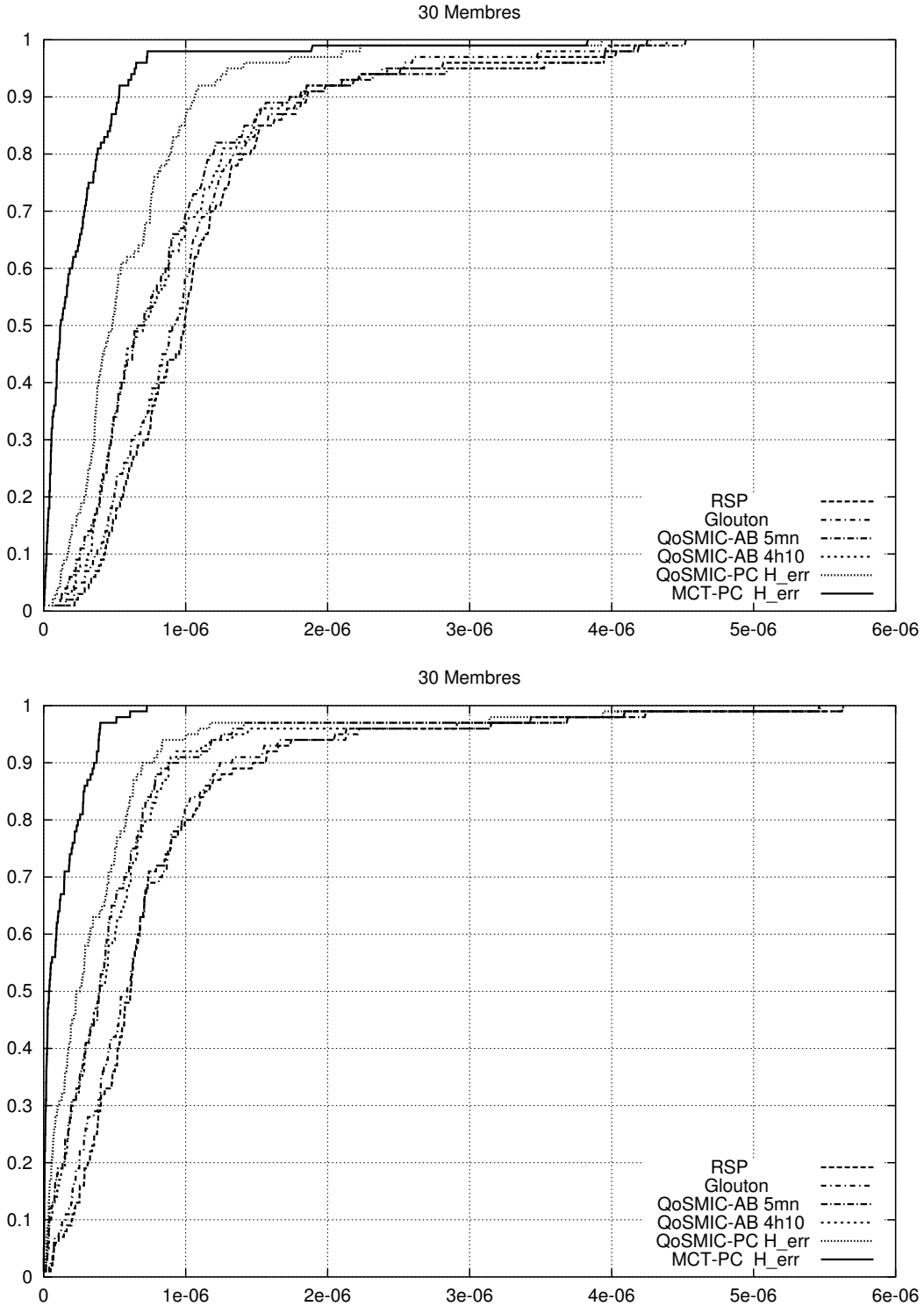

FIG. 4.12: *Comparaison entre un groupe parmi tout le réseau en haut, et un groupe dans le noyau du réseau*

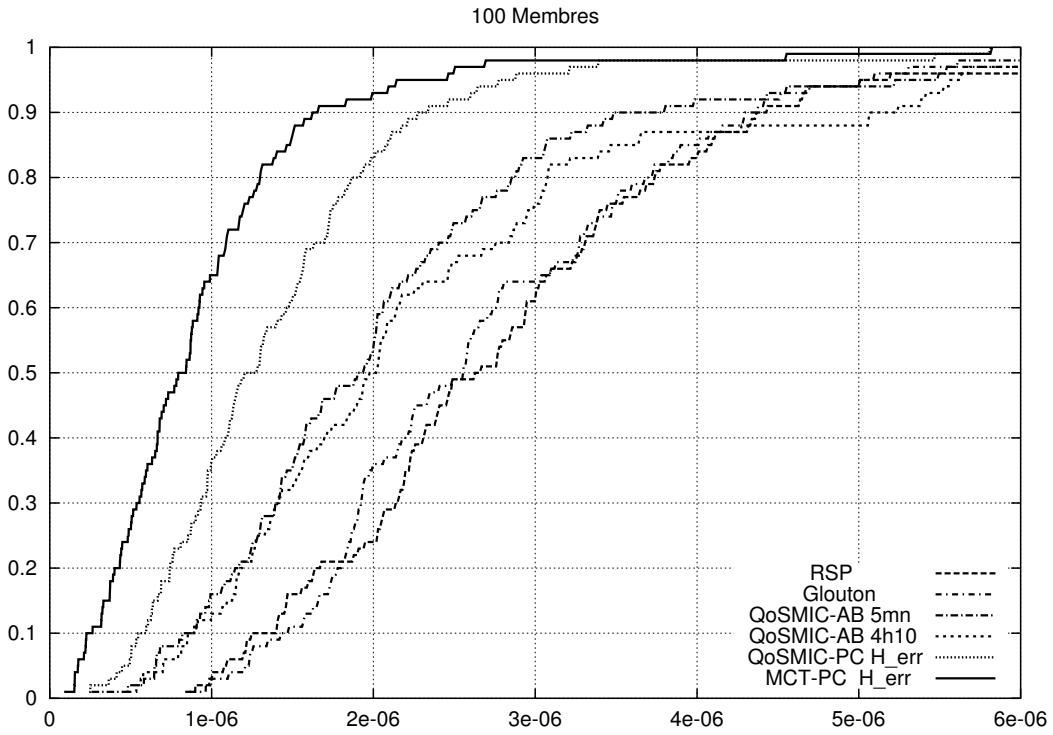

FIG. 4.13: *Comparaison des probabilités de perte pour différents protocoles*

Remarque. Nous voulons donner une idée du temps d'exécution pour une simulation. Dans le cas d'un réseau de 500 nœuds et 1990 liens, le temps d'exécution pour le temps $\tau = 8\text{mn}32$ est de 1h20 sur un processeur AMD Athlon XP à 1800 MHz. Donc, la simulation des 100 situations différentes de trafic demande un temps de 5h20 sur 25 processeurs de ce type (la plate-forme de simulation du LRI).

4.2.5 Conclusion

Dans cette section nous avons présenté des simulations des différents protocoles multipoint. Nous avons comparé les techniques de construction de plusieurs protocoles multipoint, comme RSP et glouton. Nous avons aussi comparé des protocoles intégrant la QoS, avec différents paramètres de QoS.

Nous nous sommes intéressés à mettre en évidence les pertes de paquets, car c'est un mesure pour estimer les performances d'un réseau IP. Nous avons simulé des réseaux en faisant passer un trafic autosimilaire dans tous les liens, et en calculant les pertes. Les expériences montrent que les protocoles avec QoS construisent des arbres avec moins de pertes. La congestion probabiliste a des arbres avec moins de pertes que lorsque l'on utilise les paramètres classiques de QoS. Cette amélioration reste significative dans le cas où l'estimation du paramètre du trafic H est obtenu avec $\pm 10\%$ d'erreur.

Nous avons également montré que le protocole MCT permet d'obtenir de très bonnes performances pour des valeurs de $\varrho = 1$ et $r = 1$. Le nombre de messages pour construire un arbre MCT est toutefois 30% plus élevé que dans QoSMIC.

Chapitre 5

Conclusion et Perspectives

Cette thèse a porté sur trois aspects liés au routage dans `Internet`. Tout d'abord nous avons présenté un nouveau modèle pour engendrer des topologies semblables à celle d'`Internet`. Ce modèle nous a servi d'outil pour l'étude de plusieurs protocoles sur `Internet`. En particulier, nous avons présenté deux protocoles pour améliorer les performances du routage multipoint. Le premier protocole, MSDA, retourne un ordonnancement optimal de l'envoi des messages pour minimiser le temps total d'une diffusion dans un arbre multipoint. Le second protocole, MCT, construit un arbre multicast minimisant un paramètre de qualité de service arbitraire donné. Enfin, nous avons introduit la congestion probabiliste comme nouveau paramètre de qualité de service. Ce paramètre permet de capturer la nature intrinsèquement autosimilaire du trafic sur `Internet`. Dans ce chapitre, nous présentons brièvement quelques axes de recherche à court et à moyen termes.

Modèles de topologies de réseaux

Notre modèle de topologie d'`Internet` nécessite encore quelques améliorations. En particulier, il convient d'étudier la raison pour laquelle le nombre de nœuds de degré 1 dans le modèle reste plus petit que ce qui est attendu par les lois puissance. Également, il convient d'étudier comment varient les paramètres du graphe généré en fonction de sa taille, afin de bien valider le modèle, surtout dans le cas des réseaux de petit taille. Enfin, il convient d'étudier l'influence de la distribution des nœuds d'`Internet` en fonction de la population des lieux géographiques, et de voir l'impact de l'inclusion de ce paramètre dans notre modèle.

Plus généralement, la modélisation des topologies sans fil me semble capitale. Une caractéristique de ces topologies est d'être très dynamique. De bons modèles pourraient aider au développement de protocoles de routage plus efficaces dans ces réseaux.

Protocoles pour le multicast

Évidemment il reste encore de nombreux problèmes liés à notre protocole MSDA d'ordonnancement des envois de messages dans un arbre de multicast. En particulier, nous souhaitons

en faire la mise en œuvre dans un cadre réel, par exemple sur Ethernet, TCP/IP ou ATM, afin de valider expérimentalement le bénéfice réel apporté par MSDA dans chaque cas.

En ce qui concerne la construction d'arbres de multicast, il est important de noter ici que notre protocole se décompose en un protocole qui ne s'occupe que de la topologie au travers de la proposition de plusieurs routes, et d'un autre qui construit l'arbre en choisissant parmi ces routes. Cette séparation de la topologie et du choix des routes peut aussi être utilisée par d'autres protocoles point-à-point, pour chercher un chemin avec une QoS donnée. Par exemple, le protocole RSVP [ZBHJ97] permet de réserver des ressources pour un flux donné. Nous n'avons pas avancé dans ce sens, mais cette question nous semble importante.

Notons également quelques limites de notre campagne de simulations (pour valider notre protocole MCT et l'utilisation de la congestion probabiliste dans QoSMIC), et voyons comment il serait possible d'y remédier. Pour faire nos simulations, nous avons supposé que le trafic engendré par chaque groupe multipoint est très faible par rapport au trafic sur chaque lien qu'il traverse. Cette hypothèse, couplée avec l'hypothèse que les trafics des liens ne sont pas corrélés, induit quelques restrictions sur l'impact de nos expériences. En effet dans un réseau de la taille d'`Internet`, il est raisonnable que le trafic de chaque groupe multipoint soit petit par rapport au trafic maximal. Cependant, cette hypothèse peut ne pas se vérifier systématiquement, en particulier pour le dernier lien qui connecte à l'utilisateur. La seconde hypothèse est vraie pour le noyau d'`Internet`. Néanmoins, il reste le problème de déterminer jusqu'à quel niveau elle reste valable. La condition à satisfaire pour qu'elle reste valable est que la probabilité d'envoyer un paquet à une destination soit uniforme sur l'espace de toutes les destinations.

Notons que nous travaillons également sur la mise en œuvre de MCT, sujet sur lequel j'encadre actuellement un étudiant de l'université de Buenos Aires.

Prise en compte de la nature du trafic

C'est à notre sens, le sujet sur lequel il conviendrait d'investir le plus de temps et d'énergie dans les années à venir.

Nous souhaiterions en particulier nous investir plus en profondeur sur l'estimation des paramètres du trafic. Nous avons proposé une méthode pour l'estimation en temps réel. La complexité de l'algorithme est celle de la FFT, c'est-à-dire $O(n \log n)$, et l'algorithme peut donc être utilisé avec $n = 2^{10}$ sans consommer beaucoup de temps de calcul. Pourtant notre estimation du paramètre de Hurst H est légèrement biaisée. (Malgré ce problème, la congestion probabiliste reste robuste comme il a été montré dans les expérimentations.) L'estimation en temps réel des paramètres du trafic sans consommer beaucoup de ressources dans les routeurs est donc encore à améliorer. Il existe d'autres méthodes d'estimation de qualité supérieure, mais ces méthodes nécessitent beaucoup plus de données pour effectuer les calculs. Cela augmente certes la complexité de stockage, mais la complexité de calcul peut être meilleure que notre estimation (par exemple linéaire pour le cas des ondelettes). Il peut y avoir un compromis à trouver.

Nous souhaiterions également considérer d'autres définitions de la congestion probabiliste,

basées sur d'autres modèles de trafic autosimilaire. En particulier, Jacquet [Jac98, Jac00] a étudié la taille de la file d'attente dans le modèle des sources on/off. Comme Norros dans le cadre du mouvement fractionnaire Brownien, Jacquet a donné une borne inférieure sur la taille de la file d'attente d'un lien. Cette borne fait également intervenir le paramètre de Hurst, mais également d'autres paramètres caractéristiques du modèle on/off. L'estimation en temps réel de ces derniers paramètres reste à faire. Il est cependant connu [TWS97] que quand le nombre de sources et le temps tendent vers l'infini, le modèle on/off converge vers le modèle du mouvement fractionnaire Brownien. Il est donc vraisemblable que l'utilisation de la borne de Jacquet comme base de la définition de la congestion probabiliste ne modifie pas significativement les résultats obtenus à partir de la borne de Norros. Cela reste toutefois à vérifier.

Les simulations rapportées dans la section 4.2 ont été faites pour comparer la congestion probabiliste avec d'autres paramètres de QoS. Nous ne l'avons cependant comparée qu'avec la bande passante libre. Nous comptons effectuer des comparaisons avec d'autres paramètres de QoS, par exemple le délai mesuré d'un manière plus réaliste. Ce point nous amènera à utiliser des simulateurs comme "Network Simulator-ns2" [1].

Enfin, *last but not least*, nous souhaiterions nous consacrer à l'application de la congestion probabiliste au routage point-à-point. Nous avons déjà annoncé plus haut la possibilité de l'utiliser comme un paramètre de qualité de service à demander avec le protocole RSVP [ZBHJ97]. Les protocoles point-à-point cherchent le plus court chemin en utilisant une certaine métrique, les liens ayant un poids selon cette métrique. La congestion probabiliste pourrait être utilisée comme métrique. Dans ce cas il faudra étudier comment les tables de routage et les paramètres du réseau évoluent. Les questions ouvertes sont nombreuses dans ce cadre. Par exemple, il conviendrait de déterminer si le taux de pertes du réseau est diminué, et si le trafic est bien réparti parmi tous les liens. Évidemment il faudrait mener l'étude en ajoutant un délai aux liens selon leur longueur. Il faudrait alors inclure le délai dans la métrique utilisée pour éviter que les plus courts chemins, calculés selon cette métrique, aient un délai excessivement grand.

1. Voir `http://www.isi.edu/nsnam/ns/`.

Bibliographie

[AB00] R. Albert and A.-L. Barabasi, *Topology of evolving networks: Local events and universality*, Physical Review Letters **85** (2000), 5234–5237.

[ACL00] William Aiello, Fan Chung, and Linyuan Lu, *A random graph model for massive graphs*, ACM Symposium on Theory of Computing (STOC), 2000, pp. 171–180.

[AGJS01] Cedric Adjih, Leonidas Georgiadis, Philippe Jacquet, and Wojciech Szpankowski, *Is the Internet Fractal? The Multicast Power Law Revisited*, Tech. Report 4157, INRIA, 2001.

[AGJS02] Cedric Adjih, Leonidas Georgiadis, Philippe Jacquet, and Wojciech Szpankowski, *Is the internet fractal?*, The 13th Annual ACM-SIAM Symposium On Discrete Mathematics (SODA) (New York), ACM Press, January 6–8 2002, pp. 338–345.

[AV98] P. Abry and D. Veitch, *Wavelet analysis of long-range-dependent traffic*, IEEE Transactions on Information Theory **44** (1998), no. 1, 2–15.

[Bal97] A. Ballardie, *RFC-2201: Core Based Trees (CBT) Multicast Routing Architecture*, Tech. report, IETF, September 1997.

[BC99] H. Burch and B. Cheswick, *Mapping the internet*, IEEE Computer **32** (1999), no. 4, 97–98.

[BE90] L. Breslau and D. Estrin, *Design of inter-administrative domain routing protocols*, ACM SIGCOMM, September 1990, pp. 231–241.

[BFC93] T. Ballardie, P. Francis, and J. Crowcroft, *Core based tree (CBT): an architecture for scalable inter-domain multicast routing*, proceedings of SIGCOMM, ACM press, 1993, pp. 85–95.

[BkC01] A. Broido and kc Claffy, *Internet topology: connectivity of IP graphs*, SPIE ITCom WWW Conf., 2001.

[BKM+00] A. Z. Broder, S. R. Kumar, F. Maghoul, P. Raghavan, S. Rajagopalan, R. Stata, A. Tomkins, and J. Wiener, *Graph structure in the web: experiments and models*, 9th WWW Conf., 2000, pp. 309–320.

[BSTW95] J. Beran, R. Sherman, M. Taqqu, and W. Willinger, *Long-range dependence in Variable-Bit-Rate video traffic*, IEEE Transaction on Communications **43** (1995), 1566–1579.

[BT02] T. Bu and D. Towsley, *On distinguishing between internet power law topology generators*, INFOCOM, 2002.

[CB97] Mark E. Crovella and Azer Bestavros, *Self-similarity in World Wide Web traffic: evidence and possible causes*, IEEE/ACM Transactions on Networking **5** (1997), no. 6, 835–846.

[CC97] K. Carlberg and J. Crowcroft, *Building Shared Trees using a one-to-many joining mechanism*, ACM SIGCOMM Computer Communication Review, January 1997, pp. 5–11.

[CCF01] T. Chich, J. Cohen, and P. Fraigniaud, *Unslotted deflection routing: a practical and efficient protocol for multi-hop optical networks*, IEEE/ACM Transaction on Networking **9** (2001), no. 1, 47–58.

[CCG+02] Q. Chen, H. Chang, R. Govindan, S. Jamin, S. Shenker, and W. Willinger, *The origin of power laws in internet topologies revisited*, IEEE Infocom 2002, 2002.

[CS98] John C.-I. Chuang and Marvin A. Sirbu, *Pricing multicast communication: A cost-based approach*, INET: The Internet Summit, 1998.

[CS01] John C.-I. Chuang and Marvin A. Sirbu, *Pricing multicast communication: A cost-based approach*, Telecommunication Systems **17** (2001), no. 3, 281–297.

[CTB98] M. Crovella, M. Taqqu, and A. Bestavros, *Heavy-tailed probability distributions in the World Wide Web*, chapter 1, pages 3-26, Chapman & Hall, New York, 1998.

[DEF+94] S. Deering, D. Estrin, D. Farinacci, V. Jacobson, C-G. Liu, and L. Wei, *An architecture for wide-area multicast routing*, SIGCOMM, ACM press, 1994, pp. 126–135.

[DFK+99] P. Drineas, A. Frieze, R. Kannan, S. Vempala, and V. Vinay, *Clustering in large graphs and matrices*, ACM-SIAM Symposium on Discrete Algorithms (SODA)(A Conference on Theoretical and Experimental Analysis of Discrete Algorithms), 1999.

[Dur96] Richard Durrett, *Probability: Theory and examples*, second ed., Duxbury Press, 1996.

[EFH+98] D. Estrin, D. Farinacci, A. Helmy, D. Thaler, S. Deering, M. Handley, V. Jacobson, C. Liu, and L. Wei, *RFC-2362: Protocol Independent Multicast-Sparse Mode (PIM-SM)* , Tech. report, IETF, 1998.

[ER59] P. Erdös and A. Rényi, *On random graphs I*, Publ. Math. (Debrecen) **6** (1959), 290–297.

[FBP98] M. Faloutsos, A. Banerjea, and R. Pankaj, *QoSMIC: Quality of Service sensitive Multicast Internet protoCol*, SIGCOMM, September 1998, Vancouver BC.

[FF62] L. Ford and D. Fulkerson, *Flows in Networks*, Princeton University Press, New Jork, 1962.

[FFF99] Michalis Faloutsos, Petros Faloutsos, and Christos Faloutsos, *On power-law relationships of the internet topology*, SIGCOMM, 1999, pp. 251–262.

[FJ93] S. Floyd and V. Jacobson, *Random early detection gateways for congestion avoidance*, IEEE/ACM Transactions on Networking **1** (1993), no. 4, 397–413.

[FKP02] Alex Fabrikant, Elias Koutsoupias, and Christos H. Papadimitriou, *Heuristically optimized trade-offs: A new paradigm for power laws in the internet*, LNCS (2002), no. 2380, 110–.

[FLJY93] V. Fuller, T. Li, and K. Varadhan J. Yu, *RFC-1519: Classless Inter-Domain Routing (CIDR): an Address Assignment and Aggregation Strategy*, Tech. report, IETF, September 1993.

[GPPR96] S Giordano, M. Pagano, R. Pannocchia, and F. Russo, *A new call admission control scheme based on the self similar nature of multimedia traffic*, ICC '96 Dallas, vol. 3, 1996, pp. 1612–1618.

[GT00] Ramesh Govindan and Hongsuda Tangmunarunkit, *Heuristics for internet map discovery*, IEEE INFOCOM 2000 (Tel Aviv, Israel), IEEE, March 2000, pp. 1371–1380.

[Hed88] C. Hedrick, *RFC-1058: Routing Information Protocol*, Tech. report, IETF, june 1988.

[Jac98] Philippe Jacquet, *Long term dependences and heavy tails in traffics and queues generated by memoriless on/off sources in series*, Tech. Report 3516, INRIA, 1998.

[Jac00] Philippe Jacquet, *Traffic and queueing from an unbounded set of independent memoriless on/off sources*, ch. 11, pp 269–283, in Self similar traffics, Park and Willinger editor, Willey, 2000.

[JCJ00] C. Jin, Q. Chen, and S. Jamin, *Inet: Internet topology generator*, Tech. Report CSE-TR443-00, Department of EECS, University of Michigan, 2000.

[KTA+98] Satish Kumar, David Thaler, Cengiz Alaettinoglu, Deborah Estrin, and Mark Handley, *The masc/bgmp architecture for inter-domain multicast routing*, ACM SIGCOMM, September 1998.

[LTWW93] W. E. Leland, M. S. Taqqu, W Willinger, and D. V. Wilson, *On the Self-Similar Nature of Ethernet Traffic*, SIGCOMM, ACM, 1993, San Francisco, USA.

[MMB00] A. Medina, I. Matta, and J. Byers, *On the origin of power laws in internet topologies*, Computer Communications Review **30** (2000), no. 2, 18–28.

[MN68] B. Mandelbrot and J. Van Ness, *Fractional Brownian motion, fractional noises and applications*, SIAM Review **10** (1968), 422–437.

[Moy97a] J. Moy, *RFC-1585: MOSPF, Analysis and Experience*, Tech. report, IETF, March 1997.

[Moy97b] J. Moy, *RFC-2178: OSPF Version 2*, Tech. report, IETF, July 1997.

[MP01] D. Magoni and J.-J. Pansiot, *Analysis of the autonomous system network topology*, ACM SIGCOMM Computer Communication Review **31** (2001), no. 3, 26 – 37.

[MP02a] Damien Magoni and Jean-Jacques Pansiot, *Analysis and comparison of internet topology generators*, Networking - 2nd IFIP Networking Conference, LNCS 2345, May 2002, pp. 364–375.

[MP02b] Damien Magoni and Jean-Jacques Pansiot, *Internet topology modeler based on map sampling*, ISCC'02 - 7th IEEE Symposium on Computers and Communications, July 2002, pp. 1021–1027.

[MV97] S. Molnár and A. Vidács, *On modeling and shaping self-similar ATM traffic*, 15th Int. Teletraffic Congress (ITC '97), 1997.

[NMW99] I. Norros, P. Mannersalo, and J. Wang, *Simulation of fractional Brownian motion with conditionalized random midpoint displacement*, Adv. Perf. Anal. **2** (1999), no. 1, 77–101.

[Nor94] I. Norros, *A storage model with self-similar input*, Queueing Systems **16** (1994), 387–396.

[Nor95] I. Norros, *On the Use of Fractional Brownian Motion in the Theory of Connectionless Networks*, IEEE Journal of Selected Areas in Communications **13** (1995), no. 6, 953–962.

[Nor99] I. Norros, *Busy periods of fractional Brownian storage: a large deviations approach*, Adv. Perf. Anal. **2** (1999), no. 1, 1–19.

[ÖSH00] S.A.M. Östring, H. Sirisena, and I. Hudson, *Designing and Managing Networks for Self-Similarity*, EMAC, 2000, pp. 235–238.

[PBC+99] R. Perlman, A. Ballardie, J. Crowcroft, Z. Wang, T. Maufer, C. Diot, J. Thoo, and M. Green, *Simple multicast: A design for simple, low-overhead multicast Internet Draft draft-perlman-simple-multicast-03.txt*, Tech. report, IETF, October 1999.

[PF95] V. Paxon and S. Floyd, *Wide-Area Traffic: The Failure of Poisson Modeling*, IEEE/ACM Transaction on Networking **3** (1995), no. 3, 226–244.

[PG98] J.-J. Pansiot and D. Grad, *On routes and multicast trees in the internet*, ACM SIGCOMM Computer Communication Review **28** (1998), no. 1, 41–50.

[PGLA97] M. Parsa and J. J. Garcia-Luna-Aceves, *A Protocol for Scalable Loop-Free Multicast Routing*, IEEE Journal on Selected Areas in Communications **15** (1997), no. 3, 316–331.

[PKC96] K. Park, G. Kim, and M. Crovella, *On the relationship between file sizes, transport protocols, and self-similar network traffic*, 4th Int. Conference on Network Protocols (ICNP '96), 1996, pp. 171–180.

[PKC97] K. Park, G. Kim, and M. Crovella, *On the Effect and Control of Self-Similar Network Performance*, SPIE Conf. on Performance and Control of Network Systems, 1997.

[PST99] Graham Phillips, Scott Shenker, and Hongsuda Tangmunarunkit, *Scaling of Multicast Trees: Comments on the Chuang-Sirbu Scaling Law*, SIGCOMM, 1999, pp. 41–52.

[PT00] Kihong Park and Tsunyi Tuan, *Performance evaluation of multiple time scale TCP under self-similar traffic conditions*, Modeling and Computer Simulation **10** (2000), no. 2, 152–177.

[RBR98] I. Rhee, N. Balaguru, and G. N. Roukuskas, *MTCP: Scalable TCP-like Congestion Control for Reliable Multicast*, Tech. Report TR-98-01, Department of Computer Science, North Carolina State Uiversity, 1998.

[RC98] O. Richard and F. Cappello, *Sur la nature auto-similaire de l'activité de stations de travail et de serveurs HTTP*, Technique et Science informatiques **17** (1998), 635–658.

[Rek95] Y. Rekhter, *RFC-1771: A Border Gateway Protocol 4 (BGP-4)*, Tech. report, IETF, March 1995.

[RVA00] Matthew Roughan, Darryl Veitch, and Patrice Abry, *Real-time estimation of the parameters of long-range dependence*, IEEE/ACM Transactions on Networking **8** (2000), no. 4, 467–478.

[SCH81] P. J. Slater, E. J. Cockayne, and S. T. Hedetniemi, *Information Dissemination in Trees*, Society for Industrial and Applied Mathematics **10** (1981), no. 4, 692–701.

[Ste94] W. Richard Stevens, *TCP/IP Illustrated*, vol. 1, Addison-Wesley, 1994.

[TG97] B. Tsybakov and N. Georganas, *On self-similar traffic in ATM queues: definitions, overflow probability bound, and cell decay distribution*, IEEE/ACM Transaction on Networking **5** (1997), no. 3, 397–409.

[TWS97] Murad S. Taqqu, Walter Willinger, and Robert Sherman, *Proof of a fundamental result in self-similar traffic modeling*, ACMCCR: Computer Communication Review **27** (1997), 5–23.

[VB00] A. Veres and M. Boda, *The chaotic nature of TCP congestion control*, IEEE INFOCOM, 2000, pp. 1715–1723.

[VKMV00] A. Veres, Z. Kenesi, S. Molnár, and G. Vattay, *On the propagation of Long-Range Dependence in the Internet*, ACM SIGCOMM, 2000, pp. 243–254.

[VMGC98] A. Vidács, S. Molnár, G. Gordos, and I. Cselényi, *The impact of long range dependence on cell loss in an ATM wide area network*, IEEE GLOBECOM'98, 1998.

[Vui78] Jean Vuillemin, *A data structure for manipulating priority queues*, Communications of the ACM **21** (1978), no. 4, 309–315.

[Wax88] Bernard M. Waxman, *Routing of Multipoint Connections*, IEEE Journal on Selected Areas in Communications **6** (1988), no. 9, 1617–1622.

[WE94] L. Wei and D. Estrin, *The trade-offs of multicast trees and algorithms*, ICCCN '94 Int. Conf on Computer Communications and Networks, 1994.

[WJ02] Jared Winick and Sugih Jamin, *Inet-3.0: Internet topology generator*, Tech. Report UM-CSE-TR-456-02, Department of EECS, University of Michigan, 2002.

[WTSW97] Walter Willinger, Murad S. Taqqu, Robert Sherman, and Daniel V. Wilson, *Self-similarity through high-variability: statistical analysis of Ethernet LAN traffic at the source level*, IEEE/ACM Transactions on Networking **5** (1997), no. 1, 71–86.

[wwwa] `http://people.ee.ethz.ch/~oetiker/webtools/mrtg`, MRTG.

[wwwb] `http://www.caida.org/projects/internetatlas/`, CAIDA.

[wwwc] `http://www.internet2.edu/abilene`, `http://www.abilene.iu.edu`, `http://monon.uits.iupui.edu/abilene`, Abilene Network.

[ZBHJ97] L. Zhang, S. Berson, S. Herzog, and S. Jamin, *RFC-2205: Resource ReSerVation Protocol (RSVP)*, Tech. report, IETF, September 1997.

[ZCD97] Ellen W. Zegura, Kenneth L. Calvert, and Michael J. Donahoo, *A quantitative comparison of graph-based models for Internet topology*, IEEE/ACM Transactions on Networking **5** (1997), no. 6, 770–783.

[ZGLA91] W.T. Zaumen and J.J. Garcia-Luna-Aceves, *Dynamics of shortest-path routing algorithms*, ACM SIGCOMM, September 1991, pp. 31–42.