Hop-by-hop Multicast

Luís Henrique M. K. Costa[1,2] Serge Fdida[1] Otto Carlos M. B. Duarte[2]
Luis.Costa@lip6.fr Serge.Fdida@lip6.fr otto@gta.ufrj.br

[1] LIP6/UPMC – 4, place Jussieu - 75252 - Paris Cedex 05 - France
[2] GTA/UFRJ – P.O. Box 68504 - 21945-970 - Rio de Janeiro - RJ - Brasil *

Résumé

L'architecture IP Multicast n'a pas été pleinement déployée au sein de l'Internet, malgré une dizaine d'années de recherche dans le domaine. Le routage multicast est difficile à réaliser, l'allocation d'adresses de groupe uniques est problématique et les gros fournisseurs d'infrastructure réseau résistent à l'implantation du service multicast. Dans ce papier nous proposons Hop-By-Hop multicast (HBH), un nouveau protocole de routage multicast. HBH s'appuie sur l'infrastructure de routage unicast déjà existante pour réaliser le relayage de paquets et utilise des tables de routage plus petites que les protocoles de routage traditionnels. HBH résout le problème d'allocation d'adresses en réduisant le modèle de groupe à une distribution de 1 vers N. HBH a aussi l'avantage de construire des arbres SPT (Shortest-Path Tree), et pas Reverse SPT, comme la plupart des protocoles actuels. Cette caractéristique garantit un chemin de meilleure qualité en présence de liens asymétriques et rend le protocole plus adapté à une éventuelle implémentation du routage à Qualité de Service (QoS) garantie.

1 Introduction

L'architecture IP Multicast, proposée originellement par Deering [1] n'a pas été pleinement déployée au sein de l'Internet, malgré une dizaine d'années de recherche dans le domaine [2, 3, 4, 5, 6, 7]. Ce fait est dû en grande partie à des caractéristiques propres à l'architecture elle même. Le modèle de gestion de groupe est complètement ouvert : n'importe qui peut envoyer à un groupe et n'importe qui peut avoir accès à l'information transmise. Une adresse multicast n'a pas de sens physique, n'étant pas allouée à une organisation par un organisme régulateur comme une adresse unicast. De ce fait, le routage multicast est difficile à réaliser - un protocole de routage résistant au facteur d'échelle n'a pas encore vu le jour. L'allocation d'adresses de groupe uniques est problématique : un mécanisme additionnel est nécessaire pour empêcher que la même adresse IP classe D soit utilisée par des applications différentes depuis des machines différentes. Les gros fournisseurs d'infrastructure réseau montrent une tendance à déployer des réseaux dorsaux de plus en plus performants en bande passante, mais aussi le plus simple possible au niveau routage. Il s'avère donc indispensable de définir une architecture permettant le déploiement progressif du multicast, à travers des protocoles capables de bien traiter des nuages unicast.

La recherche dans le domaine s'est donc tournée vers de nouvelles propositions [7] qui rivalisent à différents niveaux avec le modèle originel de Deering. Après l'étude des différentes approches trouvées dans la littérature, nous avons été poussés à proposer le protocole Hop-By-Hop multicast (HBH). HBH s'appuie sur l'infrastructure de routage unicast déjà existante pour réaliser le relayage de paquets et utilise des tables de routage plus petites que les protocoles de routage traditionnels car il sépare l'information de routage entre un plan de contrôle et un plan de données (qui realise le relayage des paquets). HBH résout le problème d'allocation d'adresses en réduisant le modèle de groupe à une distribution de 1 vers N. HBH a aussi l'avantage de construire des arbres SPT (Shortest-Path Tree), et pas Reverse SPT, comme la plupart des protocoles actuels. Cette caractéristique garantit un chemin de meilleure qualité pour les récepteurs dans un environnement où les liens ne sont pas symétriques (ce qui est le cas pour

*Nous remercions le CAPES/COFECUB pour le financement de Luís Costa.

Internet), et rend le protocole plus adapté à une éventuelle implémentation du routage à Qualité de Service (QoS) garantie.

Le reste du papier est organisé ainsi : la Section 2 présente les motivations et idées de base soutenant le protocole HBH, la Section 3 donne la description détaillée du fonctionnement du protocole, ensuite la Section 4 présente une comparaison de la performance de HBH avec d'autres protocoles de routage à l'aide de l'outil de simulation NS [8]. La Section 5 conclut le papier.

2 Motivations

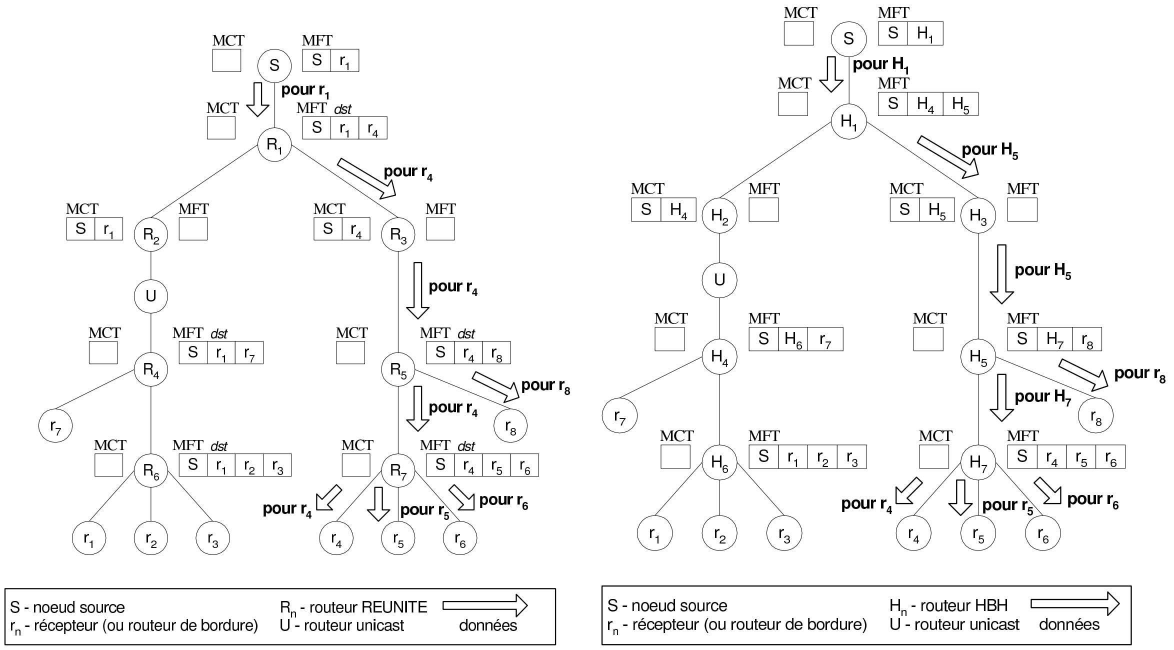
Deux des protocoles de routage récemment proposés méritent une attention particulière : EXPRESS et REUNITE. EXPRESS [9] introduit la notion de canal : un récepteur s'abonne au canal $< S, G >$, où S est l'adresse unicast de la source et G est une adresse de groupe IP classe D. L'arbre multicast et le relayage de paquets sont basés sur $< S, G >$, la distribution étant restreinte à 1 vers N. Cette restriction simplifie le problème. Puisque l'adresse unicast est par définition unique, il reste simplement à la source de choisir si nécessaire des adresses de groupe différentes pour ses différentes applications. L'allocation d'adresses est donc locale à la source. Les idées de base de EXPRESS ont été reprises dans le protocole PIM-SSM (PIM - *Source-Specific Multicast*)[10], une version modifiée de PIM-SM (*Protocol Independent Multicast* - *Sparse Mode*)[5] qui implémente le service source spécifique. Néanmoins, EXPRESS n'apporte pas de solution pour le problème de la coexistence avec des routeurs purement unicast.

REUNITE [11] abandonne presque complètement le modèle de Deering. Les adresses classe D ne sont pas utilisées - un groupe est identifié par l'ensemble $< S, P >$, S étant l'adresse unicast de la source et P un numéro de port alloué par la source.[1] La distribution multicast dans REUNITE est implémentée à travers un arbre unicast récursif. REUNITE se base sur le fait que dans les arbres multicast typiques de l'Internet, une majorité de routeurs va simplement relayer les paquets d'un lien d'entrée vers un seul lien de sortie, sans avoir à les dupliquer sur plusieurs routes. Une minorité de routeurs est responsable pour les branchements de l'arbre. Pourtant, dans les protocoles de routage classiques tous les routeurs maintiennent l'état dans leurs tables de routage pour tous les groupes qui les traversent. L'idée est donc de séparer l'information de routage dans un plan de contrôle et un plan de données. Un routeur appartenant à l'arbre $< S, P >$ ne gardera de l'information dans son plan de données que si il réalise un branchement pour l'arbre $< S, P >$.

Chaque routeur REUNITE possède une table de contrôle de routage (MCT - *Multicast Control Table*) et une table de relayage de paquets (MFT - *Multicast Forwarding Table*). Seuls les routeurs qui réalisent la duplication de paquets pour le groupe $< S_1, P_1 >$ ont des entrées dans leur MFT pour $< S_1, P_1 >$. La table de contrôle, MCT, n'est pas utilisée pour le relayage de données. Les routeurs de relais simple dans l'arbre de $< S_1, P_1 >$ ne gardent qu'une entrée dans leur MCT pour $< S_1, P_1 >$, et pas d'entrée dans leur MFT.

Dans REUNITE, les paquets de routage sont relayés à travers de l'unicast récursif. La source envoie les données en unicast vers le premier récepteur à s'abonner au groupe. Dans un routeur de branchement, R_B, les paquets de données reçus arrivent adressés au premier récepteur, r_i, à s'abonner au groupe dans le sous arbre défini à partir de R_B. r_i est maintenu dans l'entrée *dst* de la MFT de R_B. Dans R_B, une copie du paquet destiné à r_i est créée et envoyée vers chaque récepteur présent dans la MFT. Le paquet originel est lui aussi relayé. Les copies ont comme adresse destination l'adresse des récepteurs dans la MFT. La Figure 1(a) illustre le fonctionnement de l'arbre multicast REUNITE. Une partie du chemin des données y est représentée : S envoie en unicast vers r_1 (ces paquets arriveront à r_1 en unicast). R_1 crée une copie des données et les envoie vers r_4. R_3 relaie tout simplement, en unicast, les données, n'ayant pas à garder d'état dans sa MFT. R_5 crée une copie pour r_8, et finalement R_7 crée une copie des données pour r_5 et r_6.

[1] Nous adoptons la nomenclature canal pour $< S, P >$ dans REUNITE. REUNITE permet aussi un modèle multi source où un groupe est identifié par l'adresse d'un nœud *root* et un numéro de port ($< R, P >$). Toutes les sources envoient les données vers R, à partir duquel l'arbre de distribution est construit. Sans perte de généralité, nous nous restreignons ici au modèle mono source, le modèle multi source pouvant être considéré comme un mono source où R serait la source.



(a) Arbre REUNITE.

(b) Arbre HBH.

FIG. 1 – L'envoi des données à travers l'unicast récursif.

Grâce à ce schéma d'adressage et relayage de paquets, REUNITE permet le déploiement progressif du service multicast. Cette utilisation des adresses unicast permet à REUNITE de supporter la présence de routeurs purement unicast dans l'arbre de distribution. Ces routeurs sont incapables de faire la duplication de paquets vers deux branches différentes de l'arbre, mais peuvent toutefois relayer les paquets sans aucun problème, puisque l'adresse destination est unicast.

REUNITE présente néanmoins quelques problèmes relatifs à la construction et maintenance de l'arbre de distribution. Le mécanisme adopté fait que les abonnements au groupe peuvent se réaliser selon des routes non optimales pour certains récepteurs, c'est à dire, le chemin le plus court n'est pas toujours garanti. Un deuxième problème est que les routes pour certains récepteurs peuvent changer lors du départ d'un autre membre du groupe, un comportement non souhaitable du point de vue de la QoS. L'occurrence de ces phénomènes est détaillée dans la section suivante, où le mécanisme de construction de l'arbre dans REUNITE est brièvement présenté.

2.1 L'arbre REUNITE et les problèmes liés au routage asymétrique

Par routage asymétrique nous entendons que le chemin unicast pour aller de A vers B n'est pas nécessairement le chemin emprunté pour aller de B vers A. Le routage asymétrique dans l'Internet peut être dû à différents facteurs. Le cas le plus simple est l'occurrence de liens par nature asymétriques ou unidirectionnels. Il y a pourtant beaucoup d'asymétries dues à des erreurs de configuration, d'une part, et d'autre part à des routes configurées asymétriquement de façon intentionnelle. Ce genre de configuration vise à éviter qu'un réseau soit utilisé comme réseau de transit pour le trafic entre deux réseaux extérieurs.

Dans le routage multicast, cette caractéristique peut avoir des effets indésirables puisque la plupart des protocoles de routage construisent un arbre de type Reverse Shortest-Path Tree. Cela veut dire que les données envoyées en multicast d'une source S vers un récepteur R suivent le chemin utilisé pour aller de R vers S en mode unicast. Si les chemins ont des caractéristiques différentes, par exemple de délai, l'utilisation du Reverse SPT pourrait être mauvaise pour la QoS. La capacité de bien traiter les réseaux asymétriques s'avère donc un avantage dans l'algorithme de construction de l'arbre multicast.

Une des caractéristiques qui différencient REUNITE de ses prédécesseurs est son aptitude à construire des arbres SPT (MOSPF - *Multicast Open Shortest Path First* [3] est le seul des protocoles proposés pour l'Internet à construire des arbres SPT). Ceci est possible car ce sont les messages *tree* envoyés par la source vers les récepteurs qui construisent les branches de l'arbre et pas les messages *join* qui suivent le chemin inverse. Néanmoins, dans certains cas d'asymétrie, il reste toujours possible que REUNITE construise des chemins non optimaux vers certains récepteurs. La Figure 2 en donne un exemple.

Supposons les routes unicast suivantes : $r_1 > R_2 > R_1 > S$; $S > R_1 > R_3 > r_1$; $r_2 > R_3 > R_1 > S$; $S > R_4 > r_2$. Supposons maintenant que r_1 s'abonne au groupe, r_2 s'abonne ensuite et finalement que r_1 quitte le groupe. r_1 s'abonne au canal en envoyant un message $join(S, r_1)$ vers S [2]. Ce message arrive à S qui ajoute r_1 à sa MFT. Nous disons que r_1 s'est abonné au canal dans S. S commence à envoyer des messages $tree(S, r_1)$ sur l'arbre. Ces messages créent des entrées $[S, r_1]$ dans la MCT de R_1 et de R_3. Les données suivent le même chemin vers r_1. Ensuite, quand r_2 s'abonne en envoyant un $join(S, r_2)$ vers S, ce message est intercepté par R_3 car ce routeur a déjà installé l'état pour ce canal. R_3 crée une MFT avec r_1 comme *dst* et rajoute r_2 comme récepteur. Quand des données sont reçues de S vers r_1, une copie est créée et envoyée à r_2. Nous disons que r_2 s'est abonné au canal dans R_3.

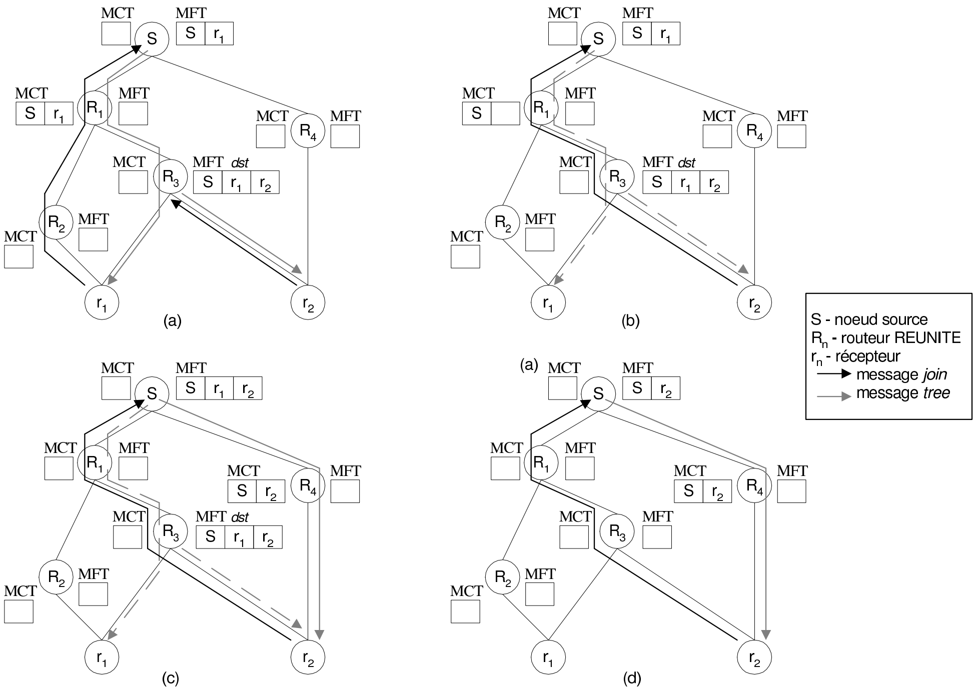


FIG. 2 – Le mécanisme de gestion de l'arbre de REUNITE.

Dans cette configuration, r_1 reçoit les données émises par S à travers le chemin le plus court. Ce n'est pourtant pas le cas pour r_2. Puisque la route de r_2 à S diffère de celle entre S et r_2 et que le message *join* n'a pas remonté jusqu'à la source, r_2 reçoit les données par le chemin $S > R_1 > R_3 > r_2$ (le même emprunté par le message *tree* entre S et r_2 dans la Figure 2(a).

Les états de contrôle (MCT) et de *forwarding* (MFT) dans REUNITE sont relâchés. Les récepteurs produisent périodiquement des $join(S, r_i)$ et la source périodiquement envoie en multicast des messages $tree(S, r_i)$. Un récepteur souhaitant quitter le canal arrête simplement l'envoi des $join$. Quand la structure de l'arbre est stabilisée, les messages $tree(S, r_i)$ envoyés depuis la source et produits au fur et à mesure par les nœuds de branchement rafraîchissent les entrées r_i dans les MCT ainsi que les entrées MFT.$dst = r_i$ tout au long de l'arbre. Les messages *join* envoyés par les récepteurs rafraîchissent l'entrée correspondante dans la MFT du nœud où le récepteur s'est abonné (dans l'exemple de la Figure 2, $join(S, r_1)$ rafraîchit l'entrée r_1 dans la MFT de S ainsi que $join(S, r_2)$ l'entrée r_2 dans la MFT de R_3).

[2] Dans le reste du papier, nous utilisons éventuellement S au lieu de $< S, P >$ ou de $< S, G >$ pour simplifier la notation.

Supposons maintenant que r_1 quitte le groupe. r_1 arrête donc d'envoyer des $join(S, r_1)$. Après l'échéance d'un timer *t1* l'entrée r_1 dans la MFT de S devient *stale* et un deuxième timer, *t2*, est créé et va éventuellement détruire cette entrée. S commence donc à envoyer de messages $tree(S, r_1)$ marqués *stale*, ce qui veut dire que bientôt l'envoi des données vers r_1 s'arrêtera (et que la partie de l'arbre basé sur r_1 doit être reconfigurée). La réception du message $tree(S, r_1, stale)$ entraîne la destruction des entrées r_1 dans les MCT tout au long de l'arbre, ainsi que le marquage en *stale* des MFT qui ont MFT.$dst = r_1$. Par conséquent, les *join* émis par r_2 iront à présent atteindre S (puisque la MFT de R_3 est *stale*), provoquant le branchement de r_2 à l'arbre dans S. La route suivie par les données entre S et r_2 devient $S > R_2 > r_2$ (dans ce cas, la nouvelle route est le chemin le plus court).

Dans un deuxième type de pathologie occasionnée par les asymétries du réseau, l'arbre construit par REUNITE peut conduire à la duplication de paquets sur certains liens.[3] La Figure 3 illustre cette situation. Le premier récepteur, r_1, envoie un $join(S, r_1)$ qui suit le chemin $r_1 > R_4 > R_2 > R_1 > S$. Les messages $tree(S, r_1)$ suivent la route $S > R_1 > R_6 > R_4 > r_1$. Supposons maintenant que r_2 se branche à l'arbre ($join(S, r_2)$ suit $r_2 > R_5 > R_3 > R_1$). Les messages $tree(S, r_2)$ et $tree(S, r_1)$ partageront le lien $R_1 - R_6$, car r_1 et r_2 se sont branchés sur l'arbre avant R_6. L'algorithme utilisé par REUNITE n'est pas capable d'identifier R_6 comme nœud de branchement, car R_6 ne reçoit pas de message *join*, ni de r_1 ni de r_2. Les données seront donc envoyées en double sur le lien $R_1 - R_6$. Par conséquent, le coût (en nombre de copies du même paquet sur les liens du réseau) d'un arbre REUNITE peut être plus grand que le coût de l'arbre construit par un protocole de routage classique, comme PIM-SM. Même si l'arbre construit par PIM-SM est un Reverse SPT, l'algorithme RPF (*Reverse Path Forwarding*) utilisé par PIM-SM garantit qu'une seule copie du paquet transite sur chaque lien de l'arbre.

Dans la prochaine section, nous décrivons le protocole proposé et la manière dont son mécanisme de gestion de l'arbre gère les problèmes liés au routage unicast asymétrique.

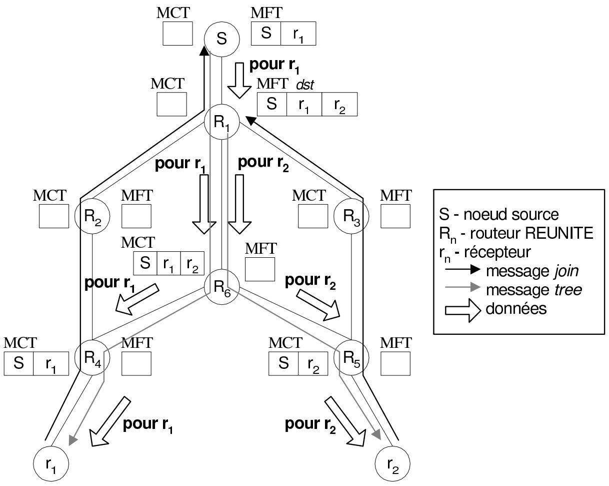


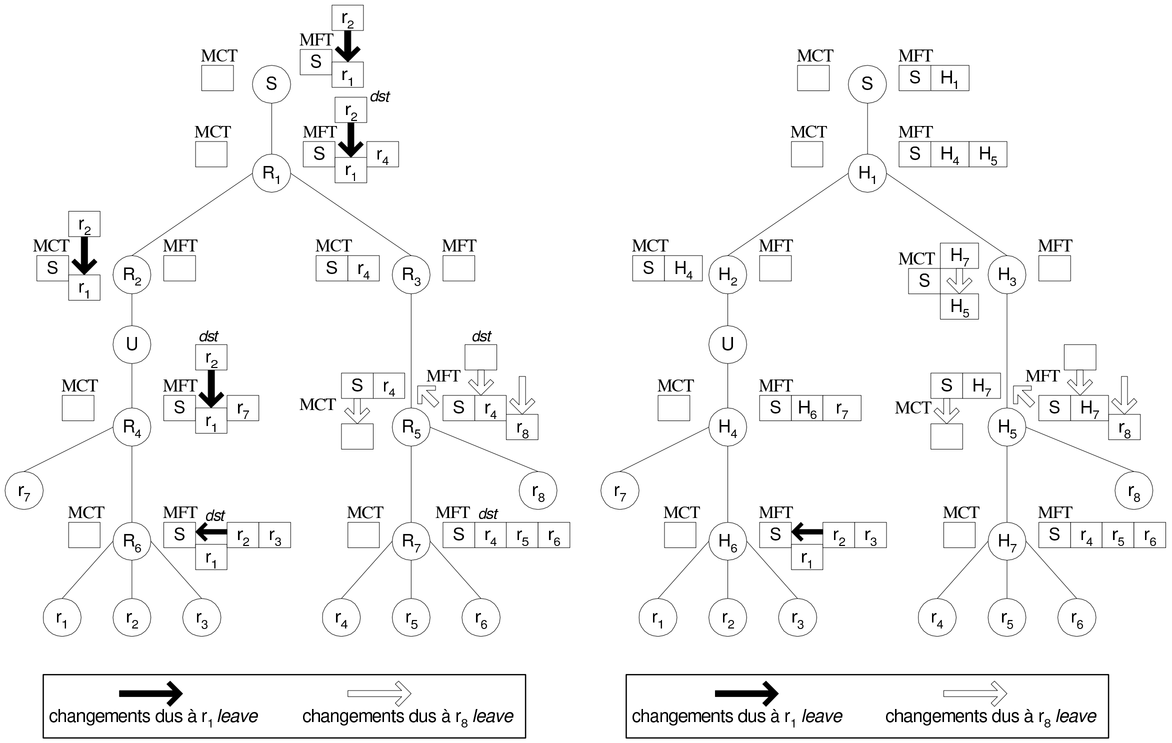
FIG. 3 – Exemple de duplication de données dans REUNITE.

3 Le Protocole Hop-By-Hop Multicast (HBH)

HBH implémente le service multicast à travers des arbres unicast récursifs, comme REUNITE, ce qui donne au protocole la capacité à traverser des nuages unicast. HBH utilise pourtant un mécanisme de construction de l'arbre qui est capable de bien traiter les cas pathologiques d'asymétrie dans le routage unicast. HBH utilise deux tables de routage distinctes, une MCT et une MFT qui ont à peu près la même fonction que dans REUNITE. Néanmoins, dans HBH les tables de relayage (MFT) contiennent des adresses des *prochains nœuds de branchement*, au lieu des adresses des *récepteurs* (sauf pour le

[3] Cette possibilité existe déjà quand le réseau contient des routeurs purement unicast ou quand les routeurs REUNITE sont surchargés. Le nœud de branchement doit migrer vers un autre routeur, ce qui peut entraîner la duplication de paquets sur un même lien. Pour une description plus détaillée, consulter [11].

routeur de branchement le plus proche du récepteur). La MFT ne contient pas d'entrée *dst*. Les données reçues par un routeur de branchement HBH lui sont adressées en unicast (dans REUNITE elles sont adressées à MFT.<*dst*>). Ce choix rend la structure de l'arbre plus stable. HBH utilise comme définition de canal un ensemble $< S, G >$, où S est l'adresse unicast de la source et G est une adresse de groupe IP classe D. Cette définition de canal de 1 vers N, comme dans EXPRESS, résout le problème d'allocation d'adresses tout en gardant la compatibilité avec le modèle de Deering - HBH pourrait ainsi supporter de manière souple l'existence des nuages IP Multicast dans les feuilles de l'arbre. Ceci est un des aspects à développer dans un travail futur.



(a) Reconfiguration de l'arbre dans REUNITE.

(b) Reconfiguration de l'arbre dans HBH.

FIG. 4 – Reconfiguration de l'arbre lors du départ d'un récepteur.

La Figure 1(b) montre un arbre HBH. Le relayage des données se passe ainsi : S envoie les données adressées en unicast vers H_1. H_1 crée deux copies des données, qui sont envoyées respectivement à H_4 et H_5 (les prochains nœuds de branchement). H_3 relaie les paquets en unicast, sans avoir à consulter sa MFT. H_5 reçoit les données et envoie une copie à H_7 et r_8. Finalement, H_7 envoie une copie des données à r_4, r_5 et r_6. La distribution se passe symétriquement de l'autre côté de l'arbre.

La structure de l'arbre HBH a comme avantage une plus grande stabilité des entrées dans les tables de routage que dans REUNITE, ayant comme coût additionnel le fait que le paquet entrant dans un routeur de branchement produira $N + 1$ copies modifiées dans HBH, tandis que dans REUNITE il génère N copies modifiées. Le gestion de l'arbre de HBH essaie de minimiser l'impact du départ d'un récepteur sur la structure de l'arbre. Ceci est possible car dans HBH les entrées contenant des adresses des récepteurs sont repoussées aux routeurs de branchement les plus proches des récepteurs. Par exemple, le départ de r_1 provoquera plus de changements dans l'arbre REUNITE que dans l'arbre HBH (Figures 4(a) et 4(b)). Dans le pire des cas (quand un nœud de branchement devient un nœud de simple relayage), HBH peut utiliser un changement de plus dans les tables de routage que REUNITE (le cas du départ de r_8 dans l'exemple présenté). Dans cet exemple, les liens sont symétriques, n'imposant pas la modification des routes pour les autres membres du groupe, ce qui peut arriver dans certains cas dans REUNITE mais pas dans HBH.

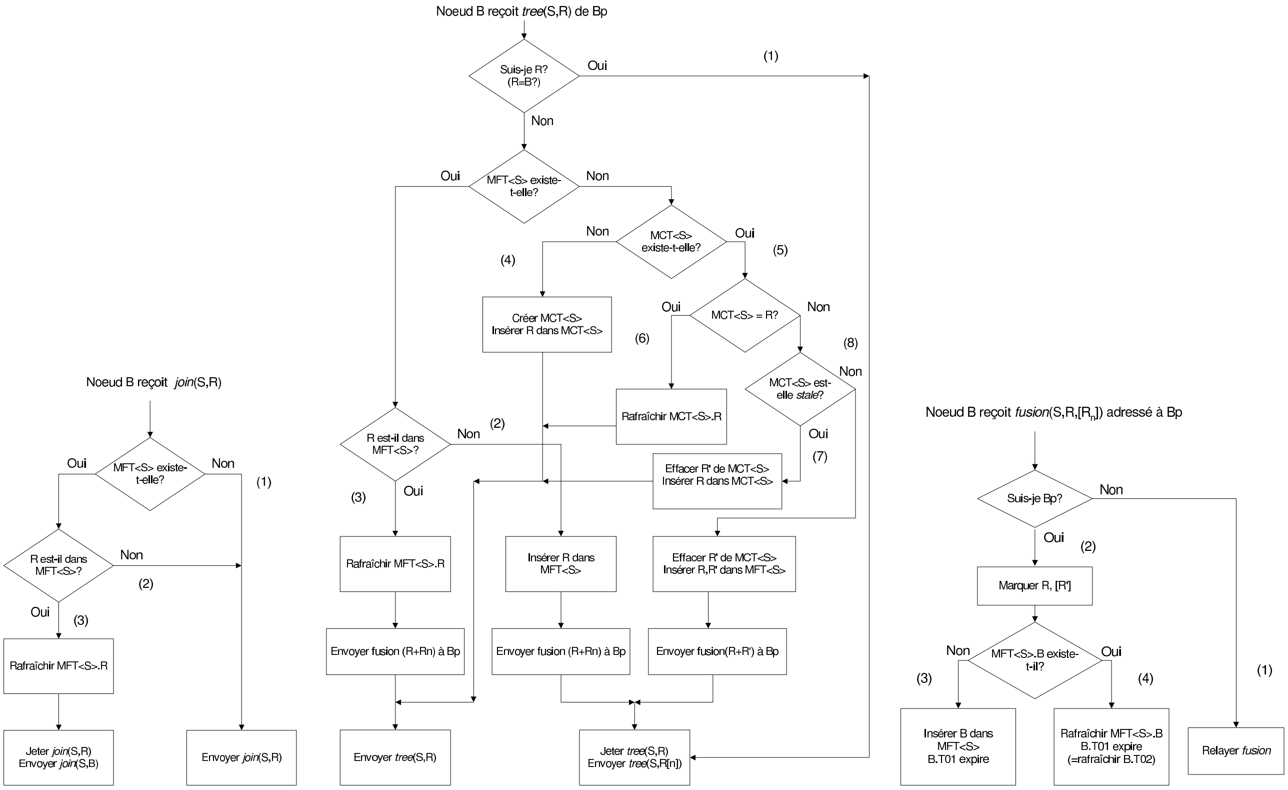
3.1 Construction et maintenance de l'arbre HBH

HBH utilise trois messages dans la construction de l'arbre multicast : *join*, *tree* et *fusion*. Les messages *join* sont périodiquement envoyés en unicast par les récepteurs vers la source, et servent à rafraîchir l'état de *forwarding* (l'entrée dans la MFT) dans le nœud auquel le récepteur s'est branché. Les messages *join* peuvent être modifiés en cours de route vers la source, car dans HBH un routeur de branchement s'abonne au prochain routeur de branchement en amont dans l'arbre. La source périodiquement envoie en multicast un message *tree* qui sert à rafraîchir la structure de l'arbre. Des messages *fusion* sont envoyés en unicast par des routeurs de branchement potentiels afin de construire la structure de distribution de l'arbre, à l'aide des messages *tree* reçus depuis la source. Dans la suite nous reprendrons l'exemple présenté dans la Section 2.1 pour décrire le fonctionnement de HBH.

Chaque routeur HBH dans l'arbre de distribution de S a soit une MCT soit une MFT pour S. La MCT$<S>$ ne peut avoir qu'une entrée, à laquelle sont associés deux timers, *t1* et *t2*. Quand *t1* expire, l'entrée devient *stale*. À l'échéance de *t2*, l'entrée (la MCT par conséquent) est détruite. Un nœud qui est sur l'arbre de S mais qui ne fait pas de branchement aura une MCT pour S.

Un routeur de branchement pour le canal S a une MFT pour S. À chaque entrée dans la MFT sont associés deux timers, *t1* et *t2*. À l'échéance de *t1* l'entrée devient *stale* et à l'échéance de *t2* elle est détruite. Une entrée *stale* est utilisée pour relayer les données en multicast sur l'arbre, mais ne produit pas de message *tree* en aval de l'arbre. L'entrée dans la MFT peut aussi être marquée. Une entrée marquée produira un message *tree* en sortie mais pas de données. Ces principes sont argumentés ci-dessous.

La Figure 5 présente les schémas avec les règles de traitement des trois types de message utilisés dans HBH. Un récepteur R envoie périodiquement un message $join(S, R)$ en unicast vers la source. La source envoie périodiquement un message *tree* en multicast sur chaque canal $< S, G >$.



(a) Message *join*. (b) Message *tree*. (c) Message *fusion*.

FIG. 5 – Le traitement des messages dans HBH.

Message *join* (Figure 5(a)) - Quand un routeur B reçoit un message $join(S, R)$, il le retransmet inchangé si il n'a pas de MFT (1) ou si R n'est pas présent dans sa MFT (2). Seulement dans le cas où R est branché à l'arbre dans B (le cas où R est dans la MFT) le routeur intercepte le $join(S, R)$ et va envoyer par la suite un $join(S, B)$ puisque B est un routeur de branchement pour le canal $< S, G >$ (3).

Message *tree* (Figure 5(b)) - Quand un routeur B reçoit un $tree(S, R)$, il doit le traiter et le relayer en multicast sur l'arbre. Si B est un nœud de branchement, il peut recevoir un $tree$ adressé à lui. Dans ce cas il le jette et envoie un $tree$ pour chaque routeur présent dans sa MFT (1). Si B est un nœud de branchement et $tree(S, R)$ n'est pas adressé à B, soit R est un nouveau récepteur (dans le cas où R est inséré dans la MFT et un $fusion$ est envoyé en amont) (2), soit R est présent dans la MFT - mais B ne reçoit pas de $join(S, R)$ - et dans ce cas B n'a qu'à rafraîchir l'entrée R dans sa MFT et envoyer un $fusion$ en amont (3). Dans le cas où B n'est pas un nœud de branchement, soit B n'était pas dans l'arbre de distribution pour S (dans le cas où il crée une MCT contenant R) (4), soit B était déjà dans l'arbre comme nœud de simple relayage (dans ce cas, il possède une MCT pour S) (5). Si R était dans la MCT de B, rien n'a changé et B doit simplement rafraîchir R dans sa MCT (6). Si R n'est pas le nœud présent dans la MCT, soit la table n'est pas à jour, au cas où B remplace l'entrée dans la MCT par R, soit la MCT est à jour et cela signifie qu'un deuxième nœud va recevoir les données à travers B, et B doit donc devenir un nœud de branchement - ce qui entraîne la création d'une MFT, la destruction de la MCT pour S et l'envoi d'un $fusion$ en amont (8). Les messages $fusion$ produits par B contiennent tous les nœuds que B garde dans sa MFT - les nœuds pour lesquels B fonctionne comme nœud de branchement.

Message *fusion* (Figure 5(c)) - Quand un routeur B reçoit un $fusion(S, R, ...R_n)$, si le message ne lui est pas adressé il le relaie simplement en amont (1). Si le message lui est adressé, B doit marquer les entrées des récepteurs indiqués dans le $fusion$ dans sa MFT (2). Une entrée marquée dans la MFT est utilisée pour le relayage des messages $tree$ mais pas pour la distribution des données. Ensuite, si B_p n'est pas dans la MFT, il y est ajouté. De plus, le timer *t1* de B_p expire - B_p devient donc *stale* (3). Par conséquent B_p sera utilisé pour le relayage des données sur l'arbre, mais pas pour le relayage des messages *tree*. Si B_p était déjà dans la MFT de B, son timer *t2* est rafraîchi (ce qui empêche que cette entrée soit détruite), mais pas son timer *t1* (qui reste expiré) (4). Si par la suite B_p, qui a envoyé le $fusion$ pour $R, ...R_n$ (et qui servira donc de point d'attachement à l'arbre pour $R, ...R_n$), reçoit les $join$ issus de ces récepteurs, il va les intercepter et envoyer un $join(S, B_p)$ signé par lui même. Dans ce cas les entrées pour $R, ...R_n$ dans B seront éventuellement périmés et l'entrée B_p sera rafraîchie par les $join(S, B_p)$. Dans le cas où B_p ne reçoit pas les $join$ issus de $R, ...R_n$, l'envoi des $fusion$ va continuer de façon définitive - un nœud B en amont recevra les $join$ et produira périodiquement des $tree$ pour $R, ...R_n$ mais enverra les données vers B_p puisque les entrées des récepteurs $R, ...R_n$ sont marquées. De cette façon HBH est capable de bien traiter le deuxième cas d'asymétrie présenté dans la Section 2.1.

La Figure 6 reprend le scénario de la Figure 2. r_1 s'abonne au groupe dans S, qui commence à envoyer des messages $tree(S, r_1)$. Ces messages créent une MCT pour S (contenant r_1) dans R_1 et R_3. Quand r_2 commence à envoyer des $join(S, r_2)$ en s'abonnant au groupe, ces messages ne sont pas interceptés et atteignent la source qui commence à envoyer des $tree(S, r_2)$. Le premier $join$ émis par le récepteur n'est jamais intercepté et arrive toujours à la source. Les messages $tree(S, r_2)$ émis par la source créent l'état pour le groupe S dans la MCT de R_4 (Figure 6(b)).

Considérons maintenant que r_3 (routes unicast : $S > R_1 > R_3 > r_3$ et $r_3 > R_3 > R_1 > S$) veut s'abonner au canal. Il envoie un $join(S, r_3)$ vers S qui commence à envoyer des $tree(S, r_3)$. Dès que R_1 commence à recevoir deux messages $tree$ différents, il envoie un message $fusion(S, r_1, r_3)$ vers la source. La réception du $fusion$ par S entraîne l'ajout de R_1 dans la table MFT de S, et le marquage de r_1 *et* r_3. Rappelons qu'une entrée r marquée dans la MFT de HBH implique l'envoi de messages $tree(S, r)$ mais inhibe l'envoi de données vers r. De la même façon que R_1, R_3 reçoit des messages $tree(S, r_1)$ et $tree(S, r_3)$ et envoie par conséquent un $fusion(S, r_1, r_3)$ vers la source. La MFT de R_3 contient maintenant r_1 et r_3. Les $join(S, r_1)$ subséquents seront interceptés par R_1 (et rafraîchiront

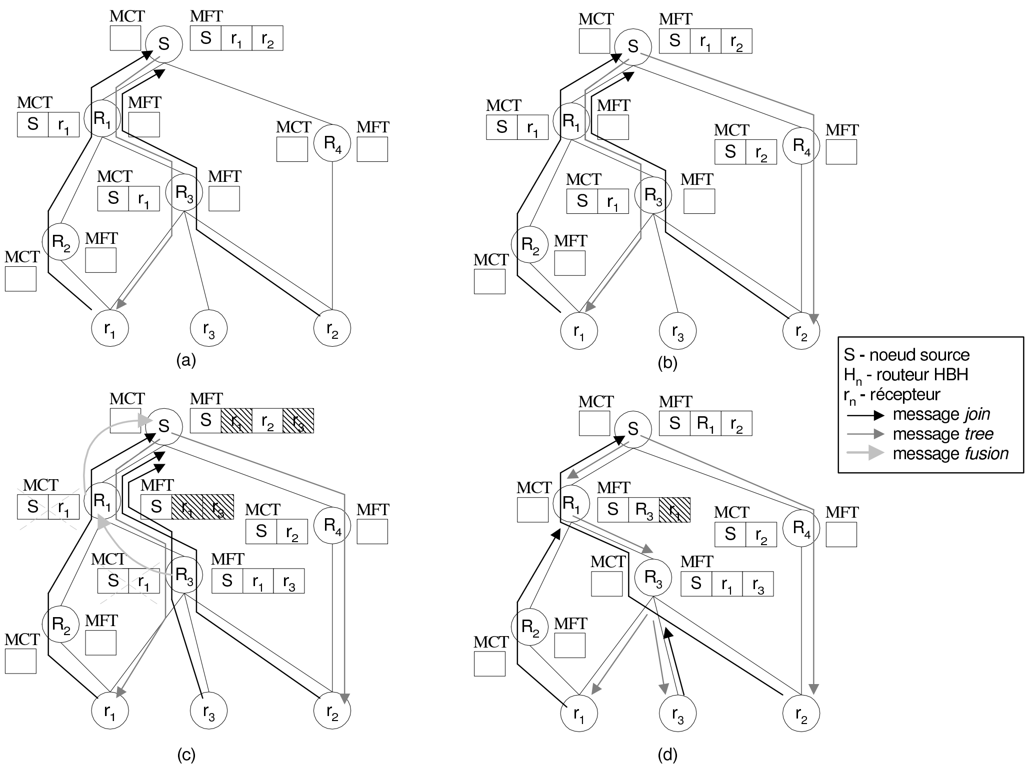


FIG. 6 – Le mécanisme de gestion de l'arbre de HBH.

l'entrée, marquée, r_1 dans sa MFT). Les $join(S, r_3)$ rafraîchiront l'entrée r_3 dans la MFT de R_3. Les données adressées par S à R_1 sont envoyées (adressées) à R_3, qui les relaye à r_1 et envoie une copie à r_3. De cette façon le point correct de branchement des paquets est utilisé par HBH. Comme S ne recevra plus de $join$ issus de r_1 et r_3, ces entrées disparaissent à l'échéance des temporisateurs correspondants. La structure finale est celle de la Figure 6(d).

4 Analyse de Performance de HBH

HBH a été simulé à l'aide du simulateur NS (Network Simulator)[8]. Les simulations avaient pour but de valider le mécanisme de gestion de l'arbre et de distribution des données de HBH, ainsi que de le comparer avec REUNITE en analysant les caractéristiques des arbres construits. Nous avons analysé le délai moyen éprouvé par les récepteurs du groupe et la quantité de copies du même paquet transmises pour couvrir tous les récepteurs.

4.1 Scénario de simulation

Pour les simulations nous avons utilisé la topologie présentée dans la Figure 7. Cette topologie est typique d'un grand réseau d'ISP [12]. Sans perte de généralité, un seul récepteur est connecté à chaque nœud de la topologie. De cette façon, nous ne considérons pas l'agrégation dû au service multicast dans les réseaux locaux. La présence de un ou plusieurs récepteurs attachés à un même nœud du réseau ne change pas le coût de l'arbre multicast. Les nœuds de 0 à 17 sont des routeurs tandis que les nœuds de 18 à 35 sont des membres potentiels du canal multicast.

Au lien $<n_1, n_2>$ qui connecte les nœuds n_1 et n_2 sont associés deux coûts, $n_1 > n_2$ et $n_2 > n_1$ choisis aléatoirement dans l'intervalle $[1, 10]$. Les simulations considèrent un groupe multicast de 1 vers N. Le nœud 18 est fixé comme source. Un nombre variable de récepteurs est choisi aléatoirement et attaché aux autres nœuds du réseau. Pour chaque nombre de récepteurs, nous avons réalisé 500 simulations par algorithme.

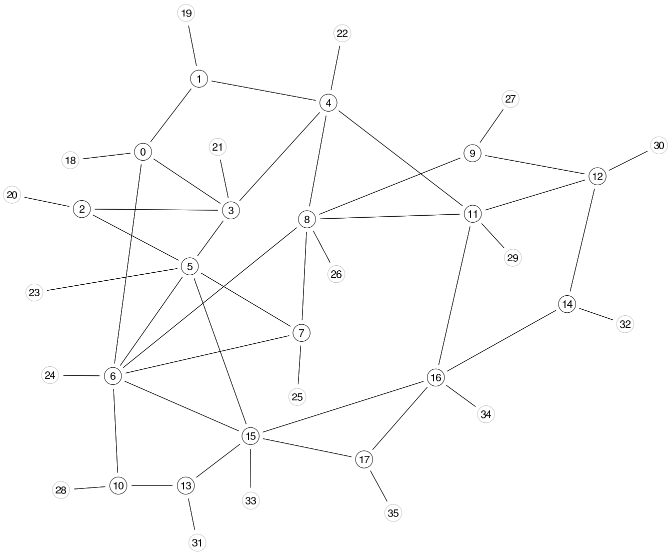
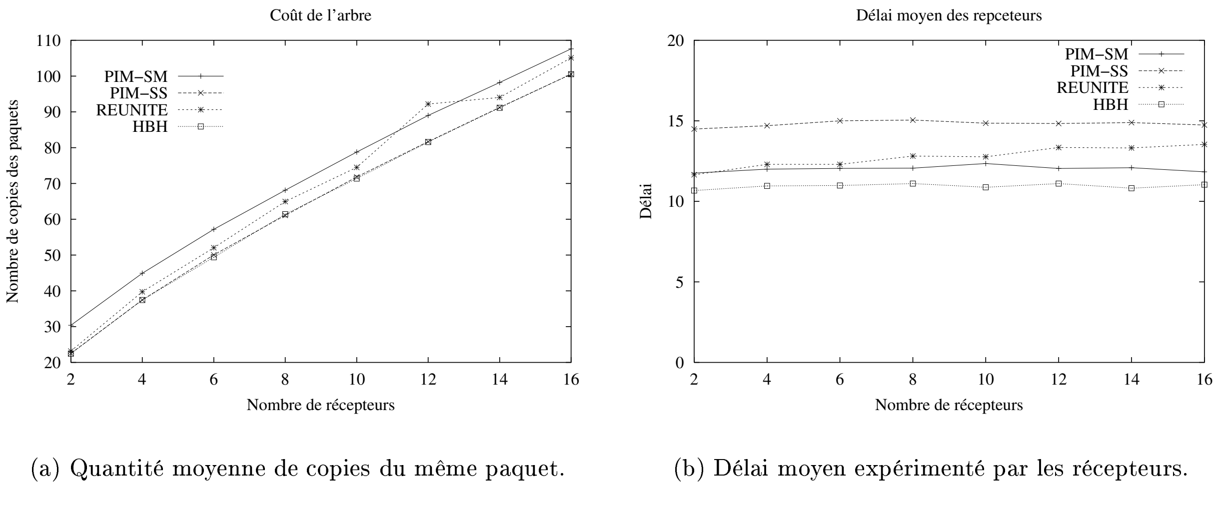


Fig. 7 – La topologie ISP.

4.2 Résultats obtenus

Afin de comparer HBH aussi avec le multicast classique, nous avons aussi simulé deux autres styles de routage qui sont disponibles dans NS. Le simulateur possède un protocole de routage qui est capable de construire des arbres partagés et aussi des arbres par source de manière centralisée. Le changement vers l'arbre par source se fait à partir d'une commande explicite et non pas à partir d'un seuil de trafic comme dans le PIM-SM originel [5]. PIM-SM dans les graphes à suivre est donc un protocole qui construit uniquement des arbres partagés. PIM-SS est un protocole qui construit uniquement des arbres par source. La structure de l'arbre construit par PIM-SS est équivalente à celle du protocole PIM-SSM qui est en phase de normalisation [10]. REUNITE représente l'implémentation que nous avons réalisé dans NS à partir de [11] et HBH l'implémentation de notre proposition. Dans les expérimentations réalisées tous les routeurs implémentent le service multicast.



(a) Quantité moyenne de copies du même paquet.

(b) Délai moyen expérimenté par les récepteurs.

Fig. 8 – Comparaison des différents protocoles multicast.

4.2.1 Coût de l'arbre

Dans un premier temps, nous avons évalué le coût des arbres produits par les divers protocoles multicast. Dans cette analyse, le coût est défini comme le nombre de fois qu'un même paquet est transmis sur les liens du réseau. Le coût n'est donc pas simplement le nombre de liens utilisés par l'arbre de distribution, puisque dans la technique d'unicast récursif utilisée par REUNITE et HBH, un même paquet peut être transmis sur un lien plus qu'une fois. Ceci peut être dû aux asymétries du réseau, comme la Section 2.1 l'a présenté, mais aussi à cause de routeurs purement unicast qui ne peuvent pas constituer un nœud de branchement de l'arbre.

La Figure 8(a) présente le coût moyen des arbres construits par les divers protocoles pour un nombre variable de récepteurs. L'approche PIM-SM est celle qui produit les arbres les plus coûteux dans la plupart des cas. Ceci est un résultat attendu, puisque PIM-SM construit des arbres partagés. Comme dans les simulations il n'y a qu'une seule source, l'utilisation d'un arbre partagé est désavantageux puisque cet arbre est centré sur un point de rendez-vous (RP). Cet arbre sera avec une forte probabilité plus coûteux qu'un arbre construit à partir de la source (considérant le même groupe). PIM-SS construit les arbres les moins coûteux, en égalité avec HBH. Ceci est dû au fait que PIM-SS construit un arbre par source basé sur un algorithme de type RPF. Ces deux approches nous donnent une base pour comparer HBH à REUNITE.

La courbe de REUNITE dans la Figure 8(a) démontre que son algorithme de construction de l'arbre multicast peut effectivement souffrir des cas pathologiques du routage unicast, comme nous l'avions évoqué dans la Section 2.1. Ce phénomène arrive avec une probabilité plus faible s'il y a très peu de récepteurs (la possibilité que deux récepteurs se partagent des liens de l'arbre est plus petite). REUNITE souffre moins aussi lorsque le nombre de récepteurs occupe un pourcentage important des nœuds du réseau, puisque dans ce cas l'arbre de distribution occupera un grand pourcentage des liens du réseau (ce pourcentage sera d'autant plus grand que la connectivité du réseau est importante).

L'analyse de la courbe obtenue par HBH démontre l'efficacité de l'algorithme de construction de l'arbre proposé. L'avantage de HBH par rapport à REUNITE en termes de coût atteint 11% et est en moyenne de 5%. Cet avantage représente une économie de la ressource bande passante du réseau.

4.2.2 Délai

La Figure 8(b) présente le délai moyen expérimenté par les récepteurs du groupe pour le même ensemble d'expérimentations. Les courbes montrent que HBH est effectivement capable de générer des routes de meilleure qualité que REUNITE grâce à son mécanisme de construction de l'arbre.

Deux résultats sont pourtant inattendus dans ce graphe. En premier, PIM-SM a une performance en termes de délai meuilleure que PIM-SS (autrement dit, les arbres partagés ont une performance en délai supérieure aux arbres par source). Ce fait s'explique parce que, d'une part, l'arbre par source construit est un *Reverse* SPT, et pas un SPT. Le délai n'est donc pas minimisé. D'autre part, le délai n'est pas minimisé non plus dans l'arbre partagé construit par PIM-SM. Il faut pourtant observer que les chemins entre la source et les récepteurs dans l'arbre partagé ont en commun la partie entre la source et le RP. Rappelons que dans l'arbre partagé, les données sont envoyées encapsulées en unicast entre la source et le RP. Ces données suivent donc une route unicast qui minimise le délai entre source et RP. Par conséquent, les chemins entre source et récepteurs ont une partie où le délai est minimisé et une deuxième partie construite en *reverse shortest-path* où le délai n'est pas minimisé. Cela explique la meilleure performance en délai de l'arbre partagé par rapport à l'arbre par source.

La deuxième observation importante dans le graphe est que l'effet des asymétries du réseau sur la qualité des arbres construits par REUNITE peut effectivement être très important. Ceci est démontré par la performance plus faible que REUNITE présente par rapport à PIM-SM notamment quand le nombre de récepteurs est plus grand.

L'avantage de HBH par rapport à REUNITE est plus importante quand le nombre de récepteurs est plus grand, s'élevant à environ 18% avec 14% de moyenne. En considérant la taille du réseau (18 nœuds avec connectivité de 3,3), l'avantage de HBH par rapport à REUNITE (5% en coût et 14% en délai) peut être considéré comme un résultat prometteur, puisque pour des réseaux plus grands cet avantage devrait croître. Cet analyse sera faite dans un travail futur.

5 Conclusions

Dans HBH, tous les récepteurs sont assurés de recevoir les données par le chemin le plus court depuis la source, même en présence de routes asymétriques. Le mécanisme de gestion de l'arbre de HBH est également capable de mieux gérer d'autres cas pathologiques dûs à des asymétries du routage unicast, de façon à minimiser la duplication de paquets dans le réseau (inévitable lors de la présence de routeurs unicast). De cette façon, HBH a le potentiel de produire des arbres multicast moins coûteux (en termes de bande passante) que REUNITE. Le service multicast devient donc plus attrayant pour les ISPs qui n'ont pas les mêmes capacités en bande passante que les gros fournisseurs d'infrastructure. Par ailleurs, la meilleure performance de HBH en termes de délai experimenté par les récepteurs en fait un protocole mieux adapté aux applications qui ne supportent pas de gros délais, notamment celles qui incluent de l'interactivité.

Les résultats obtenus à partir des simulations ont montré que l'approche HBH est prometteuse. Dans un réseau de taille moynenne, HBH a montré un avantage moyen d'environ 5% en termes de coût de l'arbre et de 14% en termes de délai par rapport à REUNITE.

Nos travaux futurs consistent à réaliser des expérimentations avec des réseaux de taille plus importante et à analyser le comportement du protocole en présence de routeurs unicast. Parallèlement, l'interface entre HBH et IP Multicast classique doit être specifiée. Le pas suivant sera l'étude pour inclure des paramètres de QoS dans la construction de l'arbre.

Références

[1] S. Deering, *Host Extensions for IP Multicasting.* RFC 1112, Aug. 1989.

[2] D. Waitzman, C. Partridge, and S. Deering, *Distance Vector Multicast Routing Protocol.* RFC 1075, Nov. 1988.

[3] J. Moy, *Multicast Extensions to OSPF.* RFC 1584, Mar. 1994.

[4] S. Deering, D. L. Estrin, D. Farinacci, V. Jacobson, C.-G. Liu, and L. Mei, "The PIM architecture for wide-area multicast routing," *IEEE/ACM Transactions on Networking*, vol. 4, no. 2, pp. 153–162, Apr. 1996.

[5] D. Estrin, D. Farinacci, A. Helmy, D. Thaler, S. Deering, M. Handley, V. Jacobson, C. Liu, P. Sharma, and L. Wei, *Protocol Independent Multicast-Sparse Mode (PIM-SM) : Protocol Specification.* RFC 2362, June 1998.

[6] C. Diot, W. Dabbous, and J. Crowcroft, "Multipoint communication : A survey of protocols, functions and mechanisms," *IEEE Journal on Selected Areas in Communications*, vol. 15, no. 3, pp. 277–290, Apr. 1997.

[7] C. Diot, B. N. Levine, B. Liles, H. Kassem, and D. Balensiefen, "Deployment issues for the IP multicast service and architecture," *IEEE Network*, pp. 78–88, Jan. 2000.

[8] K. Fall and K. Varadhan, *NS notes and documentation.* University of California at Berkeley/Lawrence Berkeley National Laboratory, Feb. 2000. Available at http ://www-mash.cs.berkeley.edu/ns.

[9] H. W. Holbrook and D. R. Cheriton, "IP multicast channels : EXPRESS support for large-scale single-source applications," in *ACM SIGCOMM'99*, Sept. 1999.

[10] S. Bhattacharyya, C. Diot, L. Giuliano, R. Rockell, J. Meylor, D. Meyer, and G. Shepherd, *A Framework for Source-Specific IP Multicast Deployment*, July 2000. Internet-draft : draft-bhattach-pim-ssm-00.txt.

[11] I. Stoica, T. S. E. Ng, and H. Zhang, "REUNITE : A recursive unicast approach to multicast," in *IEEE INFOCOM'2000*, Mar. 2000.

[12] G. Apostolopoulos, R. Guerin, S. Kamat, and S. K. Tripathi, "Quality of service based routing : A performance perspective," in *ACM SIGCOMM'98*, pp. 17–28, Sept. 1998.