Transports nouvelle génération dans les réseaux à très haut débit

Pawel Hadam

INSTITUT NATIONAL POLYTECHNIQUE DE GRENOBLE

THÈSE

pour obtenir le grade de

DOCTEUR DE L'INPG

Spécialité : « Informatique : Systèmes et Communications »

préparée au laboratoire LSR – IMAG
dans le cadre de l'École Doctorale
« Mathématiques, Sciences et Technologies de l'Information »

présentée et soutenue publiquement par

Paweł HADAM

Le 29 Juin 2005

Transports nouvelle génération dans les réseaux à très haut débit

---

Directeur de thèse : M. Andrzej DUDA

Co-directeur de thèse : M. Jean-Luc RICHIER

JURY :

M. Jacques MOSSIÈRE,	Président
M. Éric FLEURY,	Rapporteur
M. Laurent TOUTAIN,	Rapporteur
M. Andrzej DUDA,	Directeur de thèse
M. Jean-Luc RICHIER,	Co-directeur de thèse
M. Franck ROUSSEAU,	Examinateur

Remerciements

Je tiens tout d'abord à remercier particulièrement M. Andrzej Duda pour m'avoir accueilli au sein de l'équipe Drakkar au laboratoire LSR et pour avoir encadré ma thèse pendant ces trois années. Je tiens également à remercier M. Jean-Luc Richier, co-directeur de cette thèse, pour toute sa coopération et toutes les discussions qui ont beaucoup contribué à la forme finale de mon travail.

Je remercie les deux rapporteurs M. Éric Fleury et M. Laurent Toutain d'avoir accepté de lire et évaluer ma thèse. Je voudrais aussi remercier M. Jacques Mossière d'avoir accepté d'être le président du jury.

Je remercie toutes les personnes que j'ai rencontrées au cours du projet VTHD++, surtout l'équipe de l'ENST Bretagne pour leur support et leur coopération pendant les expériences menées sur le réseau VTHD.

Je voudrais remercier aussi l'ensemble des collègues du laboratoire LSR pour leur amitié et leur sympathie. Mes remerciements vont surtout aux membres de l'équipe Drakkar, Gilles Berger-Sabbatel, Jacques Chassin de Kergommeaux, Dominique Decouchant, Martin Heusse, Pascal Sicard, ainsi qu'aux thésards qui ont partagé avec moi la vie de jeune chercheur, notamment Laurentiu-Sorin Paun, Cristina Pop, Vincent Untz, Justinian Oprescu, Fabrice Salpétrier, Phuong-Hoang Nguyen, Hoa-Binh Nguyen, Leyla Toumi et Stephan Lo-Presti.

Je remercie plus particulièrement Franck Rousseau pour tous ses commentaires et ses remarques qui ont guidé et bien amélioré ma recherche. Je remercie aussi Paul Starzetz qui m'a beaucoup aidé pendant le travail avec le noyau Linux.

Je voudrais remercier M. Roland Groz avec qui j'ai pu gagner mes premières expériences pédagogiques. Son professionnalisme et ses remarques précieuses ont beaucoup apporté à mon travail pédagogique avec les étudiants du Département Télécommunication de l'INP Grenoble.

Je voudrais adresser aussi un grand merci à tous mes amis polonais, français et ceux venant du différents pays du monde que j'ai rencontré à Grenoble : Marta, Magda, Wik-

tor, Marlena et Stanisław, Honorata, Ela et Jacek, Kasia et Adam, Agnieszka et Marek, Joanna et Jacek, Ewa et Gabriel, Stephanie et beaucoup d'autres. Votre présence a rendu mon séjour à Grenoble inoubliable.

Finalement, je souhaite remercier ma femme Monika pour son soutien constant dans mon travail et pour avoir été toujours à mes côtés tout au long de cette thèse.

A toutes ces personnes je dis un grand merci.

Table des matières

Table des figures

Liste des tableaux

Chapitre 1

Introduction

1.1 Motivation

Aujourd'hui nous pouvons observer une évolution très importante des réseaux. Cette évolution prend deux formes : le nombre et la diversité de technologies et le débit offert par les nouvelles technologies. Les ordinateurs personnels commercialisés actuellement sont équipés d'office de plusieurs interfaces réseau. Il s'agit, par exemple, de cartes Ethernet 1Gb, de cartes sans fils 802.11g ou de modems ADSL. La téléphonie mobile arrive avec UMTS. Le débit antérieurement mesuré en Kb/s se mesure maintenant en Gb/s et le progrès n'a pas l'air de pouvoir s'arrêter à ce point. Ce progrès peut continuer et le débit peut plus que doubler chaque année pendant au moins 30 ans qui viennent [1]. Il est clair que le très haut débit devient aujourd'hui la réalité et qu'il est disponible pour tout le monde sous des formes diverses et économiques.

D'un autre côté, l'Internet d'aujourd'hui est toujours principalement basé sur le protocole IPv4 (*Internet Protocol, version 4*)[2] dans la couche réseau et sur les protocoles TCP (*Transmission Control Protocol*)[3] et UDP (*User Datagram Protocol*)[4] dans la couche transport. Ces protocoles ont été proposés et développés dans les années 70 pour des technologies qui peuvent être aujourd'hui considérées comme des technologies anciennes. À cette époque, qui est considérée comme l'époque de la naissance de l'Internet dans sa forme moderne, le réseau était une innovation et a vite évolué vers un réseau global comprenant une centaine de machines. Sa bande passante était fortement limitée et très chère. Les interfaces réseau n'étaient pas aussi populaires qu'elles le sont aujourd'hui. Tout cela a provoqué la naissance de protocoles adaptés au contexte existant. Les protocoles étaient optimisés pour ne pas gaspiller des ressources précieuses. Alors qu'aujourd'hui l'Internet global est passé à des millions d'utilisateurs à l'échelle globale et que les technologies sont plus avancées, les protocoles TCP/IP (*Internet Protocol*) fonctionnent toujours et assurent la connectivité pour ses utilisateurs. Cela prouve que les concepts de bases étaient solides et ont bien résisté au temps et à l'évolution constante. Néanmoins, l'héritage historique de

TCP/IP peut provoquer certaines contraintes. Il est difficile d'étendre les fonctionnalités ou d'en ajouter de nouvelles, par exemple le multi-accès, la communication multipoint et les mécanismes de qualité de service (QoS). Même si certains mécanismes et protocoles one été proposés, leur déploiement à l'échelle globale est fortement limité. Le protocole IPv6 (*Internet Protocol, version 6*)[5] a apporté un espoir en proposant un nouvel adressage. Malgré cela, une adresse IP garde toujours deux fonctions différentes, la localisation et l'identification, qui ne coopèrent pas bien dans toutes les situations.

Cette thèse a été motivée par les questions qu'on peut se poser en observant ce progrès technologique. Nous avons voulu savoir comment on peut apporter de nouvelles fonctionnalités dans le contexte de réseaux à très haut débit. Nous pensons qu'avec l'apparition des réseaux à très haut débit, les paradigmes de transmission de données peuvent évoluer. Auparavant, quand le débit était une ressource rare est chère, on évitait de transférer inutilement les données. On préférait les traiter et transférer uniquement les résultats ou les données explicitement demandées et nécessaires. Maintenant les moyens de transmission sont aussi abondants que les moyens de traitement de données. Cette situation permet de transférer plus de données avant de les avoir traitées, dans leur forme brute. Nous laissons le traitement et la sélection des données aux récepteurs qui s'en occupent, une fois les données entrées en leur possession.

Dans cette thèse nous cherchons des propositions qui peuvent apporter deux fonctionnalités :

Le multi-accès - Du point de vue de la multitude de technologies, une machine possédant plusieurs interfaces réseau n'est plus identifiée par une adresse unique. Elle peut profiter du multi-accès[1]. Le multi-accès donne la possibilité d'utiliser plusieurs interfaces réseau d'une machine terminale pour profiter de plusieurs chemins de transmission indépendants entre deux extrémités.

La distribution du contenu - En considérant la bande-passante offerte par les réseaux à très haut débit, nous pensons tout de suite à en profiter pour la distribution de données multimédia. Cela implique l'utilisation de protocoles de communication multipoint qui permettent une transmission de données entre plusieurs utilisateurs à chaque extrémité (plusieurs émetteurs et/ou plusieurs destinataires).

Nous cherchons des protocoles qui peuvent fournir ces deux types de fonctionnalités. Dans cette thèse, nous voulons explorer des méthodes différentes de transport réseau[2].

[1]Nous utilisons ce terme pour le terme anglais *multihoming*. Dans la littérature francaise on peut également trouver le terme *multi-domiciliation*.

[2]Dans ce contexte la notion de transport englobe les couches réseau et transport. Nous considérons le transport comme l'ensemble de mécanismes qui assurent l'adressage, le routage et la gestion de la transmission de données entre les émetteurs et les récepteurs des données. Le transport offre sa fonctionnalité aux applications de manière transparente.

Pour les deux problèmes présentés, il existe plusieurs propositions, surtout dans la couche application. Nous pensons ici aux réseaux CDN (*Content Distribution Network*)[6] ou *peer-to-peer*. Dans les deux cas, les solutions permettent d'établir plusieurs connexions indépendantes pour une application, ce qui peut offrir une solution aux problèmes du multi-accès et de la communication multipoint. Les propositions qui utilisent des caches (des copies de données réparties géographiquement) facilitent la distribution de contenu multimédia, en le rapprochant des utilisateurs potentiellement intéressés.

Néanmoins, les solutions implémentées dans la couche d'application présentent des inconvénients. Elles ne coopèrent pas correctement entre elles et ne sont pas facilement réutilisables dans d'autres applications. Cela nous a motivé à chercher des solutions dans les couches réseau et transport. De cette façon, nous gagnons l'universalité d'une solution qui peut servir comme une base solide pour des applications diverses, par exemple pour les CDN ou *peer-to-peer*.

1.2 Contexte et objectif de cette thèse

Le travail de la recherche de cette thèse a été effectué au sein de l'équipe Drakkar du laboratoire LSR-IMAG et financée par le projet VTHD++ (*Vraiment Très Haut Débit*). VTHD++ [7] est un projet scientifique soutenu par le Réseau National de Recherche en Télécommunications et mené entre les différents organismes de recherche français sur une plate-forme d'expérimentation IP/WDM. Les organismes de recherche suivants ont participé à ce projet : France Télécom R&D, l'ENST, l'ENST Bretagne, l'Institut EURECOM, l'INRIA et l'IMAG. Le projet a eu pour ambition de développer des techniques, des applications et des services pour l'Internet nouvelle génération. Il s'agit, entre autres, du multicast, du multi-accès (multihoming), de la qualité de service, de l'exploitation des outils d'ingénierie de trafic ou déploiement d'IPv6. Le projet se déroulait sur une plate-forme VTHD qui a été créée pour donner une possibilité d'expérimentation à très haut débit. Le réseau dorsal de la plate-forme VTHD couvrait plusieurs centres de recherche en France et comportait des routeurs interconnectés par des liens à très haut débit, des liens optiques longue distance à 1 ou 2,5 Gbit/s. L'infrastructure dorsale de la plate-forme a été empruntée à l'infrastructure de transport de France Télécom. Pendants nos travaux, nous avons expérimenté avec un lien à très haut débit entre Grenoble et Rennes.

L'objectif de cette thèse était d'étudier le changement des concepts réseau apparaissant avec le progrès technologique observé. Les concepts existant jusqu'à maintenant ne sont plus suffisants, étant donné qu'ils ont été conçus dans un autre contexte et pour une réalité technologique différente. C'est pourquoi, les protocoles existants peuvent freiner le développement de nouveaux services réseau et des possibilités ouvertes par une nouvelle technologie. Nous nous sommes donc orientés vers de nouvelles fonctionnalités au niveau réseau et transport : le multi-accès et la diffusion.

1.3 Contribution

Le travail réalisé pendant cette thèse s'insère dans le domaine de l'ingénierie de protocoles réseau. Notre objectif est de proposer aux utilisateurs des solutions qui leur permettent de profiter pleinement, facilement et efficacement de nouvelles technologies. Plus particulièrement, nous pensons ici aux réseaux à très haut débit offerts aux utilisateurs. Nous partons du constat que cet avènement de nouvelles technologies nécessite un changement de certains concepts du réseau. Dans cette thèse, nous traitons deux aspects liés à la redéfinition de l'adressage et de la notion de connexion. Tout d'abord, nous nous efforçons de trouver une solution qui permet à l'utilisateur d'utiliser plusieurs interfaces réseau pour pouvoir augmenter la capacité et la fiabilité de ses connexions. Ensuite, nous cherchons à proposer un protocole qui, grâce à un réseau à très haut débit, facilite la distribution du contenu multimédia. Nous essayons d'apporter nos solutions dans les couches réseau et transport pour qu'elles soient universelles pour les applications.

Le multi-accès donne la possibilité d'utiliser plusieurs interfaces réseau dans une machine terminale où l'utilisateur est capable de changer les interfaces sans interrompre les transmissions de données en cours. On définit également le multi-accès simultané. C'est la possibilité d'utiliser plusieurs interfaces réseau pour acheminer différents flots de données par des interfaces différentes selon le choix de l'utilisateur. Le multi-accès est introduit pour atteindre une meilleure qualité de service, une meilleure fiabilité et une transmission plus optimale en profitant de plusieurs chemins dans le réseau. Dans cette situation, une connexion entre deux utilisateurs ne peut plus être définie par une adresse source et une adresse destination ; la définition devrait changer. Les extrémités d'une connexion sont des groupes d'adresses fonctionnant également pour permettre la transmission de données. Plusieurs chemins indépendants sont possibles entre deux extrémités de la connexion. Dans cette partie de la thèse nous pouvons résumer les principales contributions de manière suivante :

- Nous avons étudié et défini les exigences qui doivent être satisfaites par un protocole de multi-accès. Après un état de l'art, nous avons constaté que le protocole SCTP (*Stream Control Transmission Protocol*)[8] propose des bases propres et solides pour le multi-accès au niveau transport. Des études plus approfondies de SCTP et des expériences avec son support pour le multi-accès nous ont amenés à constater que SCTP souffre d'un problème dans son comportement. Nous avons identifié ce problème et proposé une modification du protocole qui le résout. Un prototype a été élaboré et évalué. Nous l'avons testé en effectuant une expérience qui n'était pas possible auparavant avec le SCTP classique.

- Nous avons conçu et élaboré un outil de mesure de performance de SCTP. Cet outil à été utilisé pour mener une série de mesures de performance de SCTP sur des réseaux à très haut débit de types différents. Dans certains mesures, SCTP a été confronté à

TCP.

– Nous avons proposé une extension du protocole SCTP qui permet d'effectuer la transmission simultanée et multichemin dans une seule connexion en profitant du multi-accès. Un prototype a été réalisé et une expérience a montré ces avantages. Les résultats sont encourageants et ouvrent des perspectives intéressantes.

La bande-passante offerte par des réseaux à très haut débit permet d'aborder la distribution de données multimédia, par exemple vidéo et audio. Cela implique l'utilisation des protocoles de communication multipoint qui permettent une transmission de données entre plusieurs utilisateurs à chaque extrémité (plusieurs émetteurs et/ou plusieurs destinataires). Dans les approches actuelles, un adressage spécifique est utilisé : une extrémité est définie par une adresse qui désigne un groupe d'utilisateurs. Si nous disposons d'une très grande bande-passante, nous pouvons étudier d'autres modèles de distribution de contenu multimédia. Au lieu d'avoir une gestion explicite de groupes où chaque utilisateur demande l'adhésion au groupe, nous préférons des groupes implicites où l'émetteur crée un groupe en décidant d'envoyer un contenu. Le contenu est proposé par défaut à tous les utilisateurs susceptibles d'être intéressés par ce contenu, qui n'ont pas besoin d'adhérer explicitement au groupe. Cette approche implique certainement une plus grande utilisation de la bande-passante. Cependant, en profitant du très haut débit, nous pouvons consacrer une partie de la bande-passante à la recherche de la simplicité et de l'universalité de la solution. La contribution de cette partie de la thèse se résume de manière suivante :

– Une architecture de distribution du contenu multimédia a été définie. Cette architecture est orientée contenu et propose un nouveau concept d'une session multimédia. La notion de session intègre le contenu véhiculé ainsi que l'adressage et le routage orientée contenu. La distribution de session se fait en *mode "forcé"*[3], dans lequel l'émetteur envoie le contenu aux utilisateurs sans demande préalable du côté utilisateur. Un prototype simple a été élaboré et des expériences ont évalué le coût de son fonctionnement.

Les contributions de cette thèse montrent une évolution des services offerts par des protocoles réseau et transport. Notre approche s'appuie sur de nouvelles définitions de l'adressage, de routage et de la notion de connexion.

1.4 Organisation du manuscrit

Cette thèse comprend deux parties principales.

La première partie, qui aborde la problèmatique du transport multichemin et concurrent, comprend 8 chapitres et est organisée de la façon suivante. Après une courte intro-

[3]Nous utilisons ce terme pour le terme anglais *push mode*.

duction dans le chapitre 2, le chapitre 3 présente la problèmatique du multi-accès. Les problèmes apparaissant dans le multi-accès y sont décrits et discutés. Pour situer notre étude, nous proposons d'abord une revue des solutions les plus intéressantes dans le domaine du multi-accès (chapitre 4). Ensuite, nous étudions en détails le protocole SCTP (chapitre 5). Dans le chapitre 6, nous présentons l'outil de mesure de performance ainsi que les mesures de performance effectuées sur des réseaux à haut débit et leurs résultats. La partie qui suit (chapitre 7) porte sur les expériences avec le multi-accès dans SCTP. Nous y montrons deux scénarios différents, un où le multi-accès de SCTP s'avère bénéfique, et l'autre où il n'apporte pas d'améliorations. Le chapitre 8 est consacré à nos deux extensions proposées pour le multi-accès dans SCTP. Ce chapitre décrit les extensions et les prototypes réalisés, des expériences menées et les résultats obtenus. Cela nous amène à formuler des conclusions et à réfléchir sur les perspectives dans le chapitre 9.

La deuxième partie de notre thèse est consacrée au transport de contenu. Elle comprend 7 chapitres et est organisée comme suit. L'introduction est suivie de l'exposition de la problèmatique (chapitre 11). Notre proposition étant ancrée dans les travaux existants, nous présentons l'architecture actuelle des protocoles multicast existants (chapitre 12). Nous consacrons la suite du travail à la description de notre architecture orientée contenu (chapitre 13). Pour mieux illustrer nos propos, nous présentons un scénario d'application pour notre architecture dans le chapitre 14. L'évaluation du prototype et l'analyse des expérimentations sont exposées dans le chapitre 15. La conclusion et la présentation des perspectives clôturent cette deuxième partie de la thèse.

Enfin, une conclusion générale termine la thèse.

Les annexes contiennent la bibliographie, la liste des abréviations utilisées et les algorithmes du routage orienté contenu.

Première partie

Transport concurrent et multichemin

Chapitre 2

Introduction

La première contribution de cette thèse concerne le problème du transport concurrent et multichemin basé sur le multi-accès. Comme nous pouvons distinguer deux types de multi-accès, une explication est nécessaire pour situer nos travaux.

Nous distinguons le multi-accès pour un site et le multi-accès pour les stations. Le premier reflète le fait qu'un réseau local se connecte à un réseau externe, public (par example l'Internet) par deux (ou plus) fournisseurs d'accès. Par conséquent, ce réseau possède plusieurs routeurs de sortie[1]. De cette façon, plusieurs chemins vers des machines externes sont possibles pour des machines locales. Le deuxième cas du multi-accès, celui qui nous intéresse, est celui des stations. Il concerne une situation particulière où une machine possédant plusieurs interfaces réseau les utilise pour avoir plusieurs connexions simultanées au réseau. Elles restent pourtant indépendantes et différentes.

Pour commencer, nous discutons la problématique du multi-accès pour les stations. Nous formulons et discutons les caractéristiques qu'une solution doit avoir pour ce type de multi-accès. Ensuite, un chapitre présente des solutions existantes. Nous ne prétendons pas présenter une liste exhaustive, nous discutons seulement des solutions qui, à notre avis, sont intéressantes et qui nous ont inspirées dans nos travaux. Certaines d'entre elles viennent plutôt du domaine de mobilité - étant donné qu'elles traitent d'aspects similaires à celles du multi-accès, nous les prenons en compte dans notre recherche. Les autres peuvent être appliquées aussi bien au multi-accès pour un site qu'au multi-accès pour des machines car elles présentent une approche plus générale.

Dans notre recherche, nous nous concentrons sur une proposition du multi-accès dans la couche transport. D'un côté, cette orientation permet à notre proposition de rester invisible pour les applications : elles peuvent établir des connexions qui profitent du multi-accès de

[1]Nous pouvons imaginer une situation où un réseau local est connecté à plusieurs réseaux externes par un seul routeur. Nous considérons aussi ce cas comme multi-accès pour un site, même s'il y a certaines différences entre eux. Nous ne présentons pas ici ces différences, parce que dans cette thèse nous n'abordons pas ce type de multi-accès.

manière transparente. D'un autre côté, cette orientation est aussi invisible pour la couche réseau qui reste inchangée. C'est la couche transport qui gère plusieurs connexions au niveau réseau[2] et donne à l'application la vision d'une connexion unique. Tout cela nous amène à proposer une solution sur la base d'une extension du protocole SCTP [8].

Étant donné que les réseaux à très haut débit sont au centre de notre intérêt, nous consacrons une partie de travaux à des expériences avec SCTP sur des réseaux de ce type. Dans ce but, un logiciel de mesure de performance a été élaboré pour mener des expériences diverses. Les résultats des mesures obtenus sur un réseau à très haut débit (local et aussi global) sont présentés dans la suite. Des expériences sur le multi-accès dans SCTP sont aussi présentées et discutées. Dans une de ces expériences nous observons un problème que nous analysons et montrons comment le résoudre. Notre première proposition permet d'éviter ce problème alors que le multi-accès original ne donne pas de résultats satisfaisants. Notre deuxième proposition permet d'augmenter la performance d'une connexion SCTP en profitant de transmissions simultanées sur plusieurs chemins indépendants entre deux machines. Nous décrivons ensuite deux prototypes ainsi que des résultats d'expériences. Nos résultats sont encourageants et ouvrent des perspectives intéressantes dans le domaine du multi-accès. Ces perspectives, discutées dans la conclusion de cette partie, sont proposées pour le protocole SCTP, mais elles peuvent également être appliquées dans d'autres solutions de multi-accès au niveau transport.

[2]Nous utilisons ici le terme *connexions au niveau réseau* pour un *chemin* dans le réseau qui interconnecte une adresse IP locale avec une adresse IP distante.

Chapitre 3

Problématique du multi-accès

La connexion d'une station équipée de plusieurs interfaces réseau pose plusieurs problèmes. Certains d'entre eux sont évidents, d'autres ne peuvent être aperçus qu'après une étude plus approfondie.

En général, les problèmes apparaissent quand une machine a plusieurs adresses IP obtenues de plusieurs opérateurs réseau différents. Toutes ces adresses sont opérationnelles et peuvent être utilisées pour transmettre et recevoir des données. Néanmoins, ces connexions peuvent avoir des caractéristiques différentes.

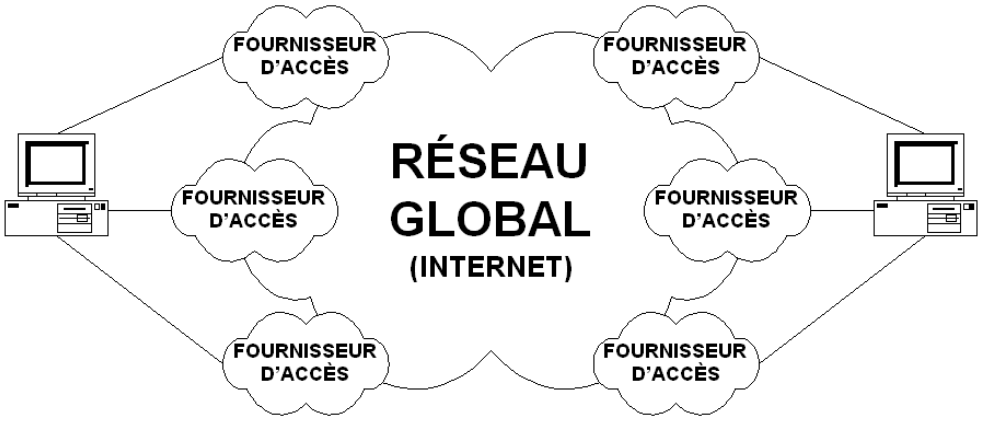

Fig. 3.1 – Deux machines bénéficiant du multi-accès

L'autre extrémité de la connexion peut aussi bénéficier du multi-accès : il peut avoir plusieurs interfaces réseau et plusieurs adresses IP. Sur la figure 3.1 on voit une situation classique où deux machines sont connectées directement à l'Internet grâce à plusieurs opérateurs réseau. Cela veut dire qu'une carte réseau dans une machine est directement reliée à un routeur d'opérateur sans nœuds intermédiaires. Il peut y avoir alors entre les

deux machines plusieurs chemins dans le réseau qui interconnectent des adresses de chaque côté (chaque chemin interconnecte une adresses locale avec une adresse distante).

Parmi les problèmes qui se posent dans cette situation, nous considérons :

- le choix de l'interface de sortie et de l'adresse source pour des paquets d'une connexion,
- le choix de la meilleure adresse destination pour des paquets d'une connexion,
- le passage progressif d'une adresse à l'autre (source ou destination),
- la sécurité pendant le changement d'adresses,
- l'ajout et la suppresion dynamique des adresses sur une machine - ce dernier problème touche à la mobilité. Dans nos travaux nous le considérons comme problème secondaire et nous n'essayons pas de le résoudre.

D'autres problèmes non étudiés dans cette thèse sont par exemple : la mise en relation de plusieurs adresses sur une machine, le maintien de cohérence de cette mise en relation ou l'identification unique d'une machine possédant plusieurs adresses pouvant changer.

Dans la suite, nous discutons les problèmes annoncés.

3.1 Adresse source

Une machine qui bénéficie du multi-accès possède plusieurs interfaces réseau connectées à des opérateurs réseau différents. Cela donne à la machine plusieurs connexions indépendantes à l'Internet global. Chaque connexion peut avoir des caractéristiques différentes (prix, débit, charge). Pour choisir la meilleure connexion pour un paquet, la machine doit choisir l'interface de sortie qui offre les meilleures conditions d'envoi pour ce paquet. On doit utiliser une adresse de l'interface choisie. Si on choisit une adresse qui ne lui appartient pas, on risque le filtrage des paquets avec des adresses source étrangères (*ingress filtering*[1] [9]).

3.2 Adresse destination

Chaque fois que nous envoyons un paquet, nous avons le choix entre plusieurs adresses du destinataire sur la base de l'accessibilité, de la bande passante disponible ou d'un autre paramètre ou critère. Notre machine peut régulièrement vérifier et évaluer toutes les adresses du destinataire pour mieux savoir quelle adresse utiliser.

[1] *L'ingress filtering* est une méthode pour réduire le nombre de paquets avec de fausses adresses source. Un opérateur réseau filtre des paquets sortant de son réseau en vérifiant leurs adresses source. Seuls les paquets portant des adresses source appartenant à l'opérateur sont autorisés à sortir du réseau de l'opérateur. Un paquet portant une adresse source étrangère est rejeté.

3.3 Changement d'une adresse

Si les paramètres du réseau peuvent changer, la configuration d'une machine doit suivre les changements : par exemple changer aussi bien l'adresse source que l'adresse destination pour atteindre une meilleure bande passante ou une meilleure performance de la connexion. Ces changements doivent être effectués d'une manière invisible pour une connexion déjà établie. Des applications doivent être inconscientes des changements d'adresse (ou des adresses) et doivent fonctionner sans perturbation.

3.4 Sécurité

Le problème de sécurité du multi-accès se pose quand on veut changer d'adresse pendant une connexion déjà établie (par exemple dans le cas de la mobilité). Nous devons être sûrs que la machine qui demande le changement d'adresse est vraiment celle de notre interlocuteur et non pas une fausse machine d'un tiers. En même temps, nous devons savoir si toutes les adresses annoncées par notre interlocuteur appartiennent vraiment à sa machine.

Le mécanisme de changement d'adresses doit être sécurisé et insensible aux attaques et aux perturbations externes.

3.5 Gestion d'adresses

Si une nouvelle adresse apparaît sur une machine (la machine se connecte à un nouveau opérateur par une autre interface réseau) ou si elle disparaît, la configuration de la machine doit être adaptée de manière adéquate. Après avoir ajouté une adresse, l'évaluation de toutes les chemins rendus possibles par cette adresse doit être effectuée. Une fois l'adresse supprimée, toutes les connexions ayant utilisé cette adresse doivent être transférées à une autre adresse d'une manière transparente. Le routage peut aussi être modifié après l'ajout ou la suppression d'une adresse.

Chapitre 4

Solutions existantes au problème du multi-accès

Le problème d'améliorer TCP pour prendre en compte du multi-accès (et de la mobilité) a été beaucoup étudié ces dernières années. Plusieurs solutions ont été publiées. La plupart sont expérimentales et n'ont pas dépassé le stade de la proposition. Dans ce chapitre, nous présentons les approches qui proposent une solution pour le multi-accès. Comme ce problème est assez proche du problème de la mobilité, ils sont souvent discutés ensemble. Certaines solutions sont spécialement conçues pour la mobilité, néanmoins, elles peuvent aussi servir pour le multi-accès. Chaque approche sera brièvement discutée en précisant les caractéristiques les plus importantes.

4.1 Homeless Mobile IP

Cette solution a été proposée par Pekka Nikander [10]. L'idée principale est d'attacher une socket à une nouvelle structure cache d'hôte (*host cache*) qui contient plusieurs adresses IP au lieu d'une adresse dans la socket classique. Sur la figure 4.1 on voit une socket qui est attachée localement à un cache d'hôte et connectée à un cache d'hôte distant.

Les adresses dans les caches d'hôte peuvent être ajoutées et supprimées à l'aide de messages de mise à jour (*binding update*) similaires à ceux de Mobile IPv6 [11] [12]. Cela permet aussi un support pour la mobilité, car une machine peut se déplacer d'un réseau à un autre, donc changer son adresse IP.

Comme cette proposition a été plutôt conçue pour la mobilité, elle ne donne pas de support suffisant pour le multi-accès. Elle n'offre pas ni de choix, ni d'évaluation d'adresses.

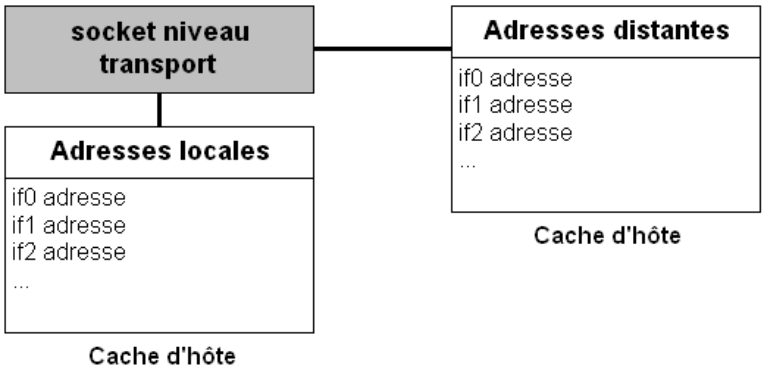

FIG. 4.1 – Une socket et les structures Cache d'Hôte

4.2 Host Identity Payload

La proposition suivante est HIP (*Host Identity Protocol*). Elle a été proposée dans trois documents décrivant l'architecture [13], le protocole [14] et l'implémentation [15]. Cette proposition introduit une nouvelle couche HIP à la pile réseau classique, placée au-dessous de la couche du transport et au-dessus de la couche du réseau. C'est pour cela qu'elle est appelée "la couche 3,5". Elle est responsable de la gestion des identificateurs des machines. Les identificateurs sont uniques et ne changent jamais pour une machine.

Sur la figure 4.2 on voit un schéma des couches qui montre le principe de fonctionnement. Dans la couche transport une socket est attachée à l'adresse IH (*Identificateur d'Hôte*). Ensuite, la couche HIP traduit cette adresse à une adresse IP. L'adresse IP est transmise à la couche du réseau et le paquet est envoyé directement vers cette adresse, comme dans la pile réseau classique.

Pour initier une connexion, HIP nécessite des services supplémentaires, notamment un répertoire (de type DNS (*Domain Name System*) [16][17]) qui permet de trouver un serveur rendez-vous qui transmet ensuite le paquet initial à la destination. Le serveur rendez-vous transmet aussi les paquets dans le cas où les deux machines changent leur localisation en même temps.

Même si les machines peuvent facilement changer leurs adresses IP, à un moment donné une seule adresse IP est en relation avec l'identificateur IH de la machine. Cela n'est pas une méthode de multi-accès où une machine bénéficie de plusieurs adresses IP à tout moment.

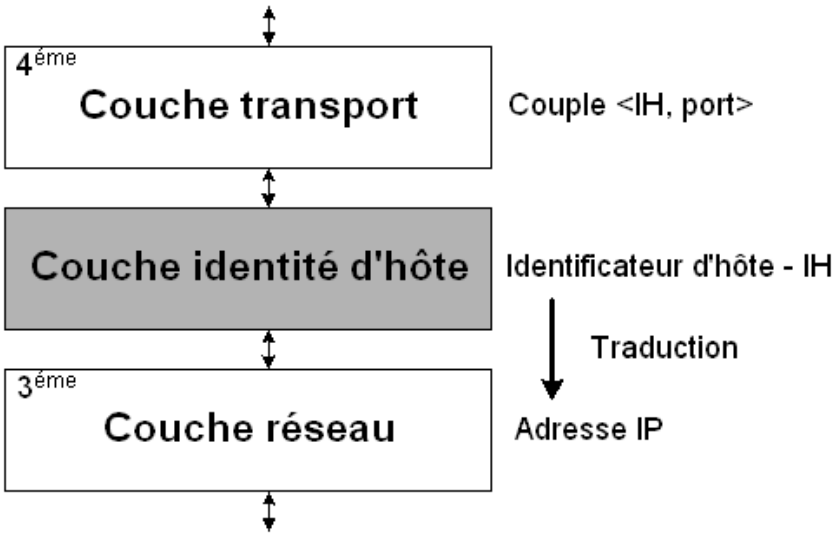

FIG. 4.2 – "La couche 3,5" dans HIP

4.3 LIN6

Cette solution propose de diviser la couche réseau en deux sous-couches : la couche livraison (*Delivery Sublayer*) et la couche identification (*Identification Sublayer*) [18]. Comme cette solution est basée sur IPv6 [5], elle exploite l'adresse IPv6 sur 128 bits en la divisant en deux parties. La première partie de l'adresse contient l'identificateur de nœud (*Node Identifier - LIN6 ID*) et la deuxième contient une localisation d'interface (*Interface Locator*). Les deux parties ont chacune 64 bits de longueur (cf. figure 4.3).

Adresse LIN6 128 bits	
Couche livraison	Couche identification
Préfixe réseau	LIN6 ID (globalement unique)
Localisation d'interface 64 bits	L'identificateur du nœud 64 bits

FIG. 4.3 – La répartition d'une adresse IPv6 dans LIN6

L'identificateur de nœud fait partie de l'adresse définie dans la couche identification.

Cette partie stable et inchangeable identifie la machine d'une manière unique. L'identificateur du nœud est attribué globalement et de cette façon il est globalement unique.

Le préfixe réseau est défini dans la couche livraison pour la localisation d'une machine. Cette partie de l'adresse identifie le réseau auquel la machine est actuellement attachée. Chaque fois que la machine se déplace, le préfixe réseau change.

Pour permettre aux autres machines de trouver une machine, un service spécial appelé agent mère (MA - *Mapping Agent*) est introduit pour maintenir la correspondance entre les identificateurs de nœud (globalement connus) et les préfixes réseau actuels des machines. Chaque machine a un MA où il est engistré. Chaque MA peut servir plusieurs machines. Le système DNS est utilisé pour trouver un MA pour un identificateur de nœud donné. Donc, c'est le DNS qui fait la correspondance entre l'identificateur de nœud et son MA, et c'est le MA qui fait ensuite la correspondance entre l'identificateur de nœud et le préfixe réseau actuel d'une machine. Une fois qu'on a l'identificateur de nœud et le préfixe réseau, on peut construire une adresse IP valide pour y envoyer directement des paquets.

Comme cette proposition a été conçue plutôt pour la mobilité, elle ne permet pas d'avoir plus d'une adresse IP en même temps. Bien sûr, on peut avoir deux interfaces réseau, mais comme deux identificateurs de nœud différents seront attribués aux interfaces, ils ne seront pas reconnus comme une seule machine. Cette proposition ne permet donc non plus d'avoir ni un identificateur de nœud pour plusieurs interfaces ni plusieurs préfixes réseau pour un identificateur nœud.

En fait, cette proposition montre une approche intéressante basée sur la division d'une adresse en deux parties. La première, responsable de l'identification d'une machine et la deuxième, responsable de la localisation d'une machine dans le réseau. Comme l'identification et la localisation sont deux fonctions complètement séparées, cette idée peut être aussi intéressante pour le développement du multi-accès.

4.4 Multi Homing Aliasing/Translation Protocol

Cette proposition n'est pas vraiment une solution pour le multi-accès [19] [20]. Les machines peuvent avoir plusieurs adresses (interfaces réseau), mais à côté de chaque extrémité (l'émetteur et le destinataire), on trouve un routeur (passerelle) spécialisé qui transforme le cas de multi-accès en une connexion standard. Entre les deux passerelles le trafic est envoyé d'une manière classique en utilisant une seule adresse de chaque passerelle.

Cela permet de bénéficier de plusieurs interfaces sur une machine, mais ne donne pas plusieurs chemins indépendants entre l'émetteur et le destinataire.

4.5 Stream Control Transmission Protocol

SCTP est un protocole de transport récemment proposé [8]. Il a été conçu pour offrir de nouvelles caractéristiques par rapport aux protocoles de transport existants TCP et UDP. En même temps, il a gardé certaines caractéristiques de TCP, par exemple la fiabilité et le contrôle de congestion. Comme de nouvelles fonctionnalités, le SCTP introduit le multi-accès, le multistreaming et renforce la sécurité des connexions. Nous décrivons ces caractéristiques rapidement dans la suite :

Multi-accès : une connexion SCTP peut utiliser plusieurs adresses de la couche réseau. En cas de rupture de connectivité à travers l'adresse utilisée, elle peut être rétablie avec une autre.

Multistreaming : SCTP permet d'envoyer plusieurs flots de données indépendants sur une connexion au niveau transport. La fiabilité et le contrôle de congestion sont gérés indépendamment pour chaque flot. S'il y a des problèmes qui apparaissent dans un flot, ils ne dérangent pas les autres.

Sécurité : SCTP propose des procédures sécurisées d'initiation et de fermeture d'une connexion. Ces procédures permettent d'éviter les problèmes qui peuvent se produire dans TCP (attaques de déni du service, connexions semi-ouvertes).

Interface de programmation : SCTP utilise des sockets de manière similaire à TCP et UDP. De cette façon, il est très facile de récrire des applications existantes (utilisant TCP ou UDP) pour qu'elles puissent utiliser SCTP.

Les caractéristiques citées ci-dessus montrent que SCTP est un protocole qui garde les fonctionnalités de TCP, renforce la sécurité des connexions et ajoute de nouvelles fonctions, notamment le multi-accès et multistreaming. En se situant dans la couche de transport, le protocole reste indépendant de la couche réseau et de l'adressage IP. Du côté des applications, SCTP est facile à utiliser grâce à l'interface socket. Pour toutes ces raisons, SCTP peut être un concurrent réel de TCP en tant que protocole du transport d'utilisation générale. Il est bien possible que SCTP puisse remplacer TCP au moins dans certains contextes grâce à ses nouvelles fonctionnalités (le multi-accès, le multistreaming et la sécurité améliorée). Ces constatations nous ont amenés à nous intéresser plus particulièrement à ce protocole dans cette partie de notre travail.

Comme dans la suite de notre travail nous nous occupons plus particulièrement de SCTP, nous décrivons dans le chapitre 5 les détails de ce protocole.

4.6 End-Point Control Protocol

Finalement, nous présentons EPCP (*End-Point Control Protocol*) [21]. EPCP est principalement basé sur le protocole SCTP. Il propose un nouveau protocole de signalisation placé entre la couche transport et la couche réseau. En principe, il est indépendant du niveau transport et du réseau, il peut donc travailler avec des protocoles différents. Ceci est son avantage le plus important par rapport à SCTP.

Les machines utilisant EPCP établissent une association pour échanger les adresses IP et pour les mettre à jour pendant les connexions en cas de besoin. Trois types d'adresses sont possibles :

- **L'adresse principale (main address)** - chaque machine possède une seule adresse de ce type. Cette adresse est statique et ne change jamais pour une machine. Chaque machine est responsable pour annoncer cette adresse aux autres. La couche transport utilise cette adresse pour identifier une association et pour établir une connexion.

- **L'adresse primaire (primary address)** - chaque machine possède une seule adresse de ce type. Cette adresse peut être changée dynamiquement à tout moment. C'est cette adresse qui est utilisée par la couche réseau pour envoyer des paquets. Au début d'une association l'adresse primaire est égale à l'adresse principale. Chaque machine doit annoncer son adresse primaire.

- **L'adresse alternative (alternative address)** - chaque machine peut posséder zéro ou plusieurs adresses de ce type. Ces adresses peuvent être dynamiquement ajoutées ou supprimées à tout moment. Toutes les adresses ajoutées à une association sont d'abord classées comme adresses alternatives. Plus tard, le statut d'une adresse alternative peut être changé en adresse primaire.

Il n'y a pas de répertoire pour stocker les adresses, les machines doivent annoncer leurs adresses directement à leurs interlocuteurs.

4.7 Multi6 shim

La solution "multi6 shim" a été récemment proposée pour le multi-accès des sites dans le groupe de travail IETF (*Internet Engineering Task Force*) multi6 [22][23]. Mais comme son approche est aussi intéressante pour le multi-accès de machines, nous la présentons ici.

Cette proposition est basée sur une couche introduite au milieu de la couche réseau (figure 4.4). Plus précisément, multi6 shim se place entre la sous-couche d'extrémité IP (*endpoint sublayer*), qui est responsable de la fragmentation, rassemblage et IPsec, et la

sous-couche de routage (*routing sublayer*), qui est responsable du choix des chemins pour les paquets. Ce placement permet d'utiliser un identificateur unique de la machine (ULID - *upper layer identifier*) dans la sous-couche d'extrémité, et plusieurs adresses différentes (*locateurs*) dans la sous-couche de routage. Multi6 shim s'occupe de la traduction de l'identificateur en différents locateurs. De cette façon, du point du vue du transport et des applications, un paquet possède toujours un identificateur statique (ULID), même si finalement, pour transmettre le paquet, il est traduit en un locateur. Les paquets sont transmis dans le réseau en utilisant des locateurs. À l'arrivée à l'autre extrémité, le locateur du paquet est traduit en identificateur, et le paquet est livré à la couche supérieure.

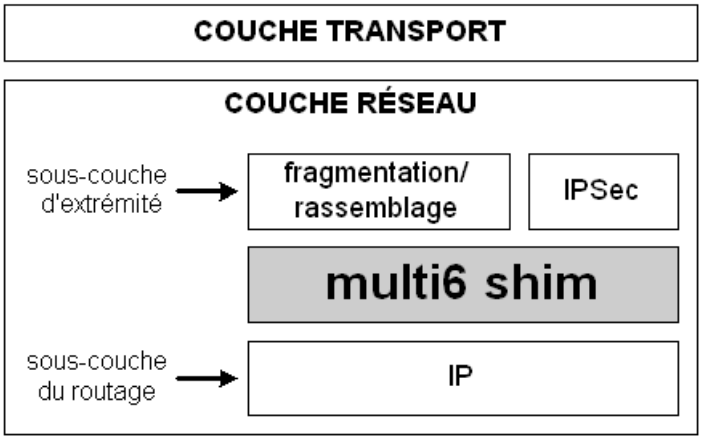

FIG. 4.4 – La sous-couche multi6 shim dans la couche réseau

Ce protocole permet de gérer des pannes dans le réseau en adaptant la traduction entre l'identificateur et le locateur. Quand une machine détecte qu'il n'y a plus de communication (à cause d'une panne), le protocole peut changer le locateur et essayer de rétablir la communication en utilisant un autre locateur mais en gardant toujours le même identificateur. Il est proposé que le protocole vérifie et évalue le chemin avant de changer de locateur.

Cette solution est indépendant de la couche transport et ne traite pas des interactions avec celle ci. Par exemple, pour le protocole TCP, il va exister de manière invisible plusieurs chemins et l'effet sur le traitement des paquets (réordonnancement, délais de réémission, calcul du RTT) n'est pas clair. Cela est acceptable dans le cas de pannes franches, mais peut poser problème si on étendait cette solution à l'optimisation des chemins (QoS, partage de charge, ...).

Chapitre 5

SCTP

SCTP (*Stream Control Transmission Protocol*) a été initialement conçu pour transporter des messages de signalisation dans des réseaux de télécommunication. Les premières exigences demandées pour ce nouveau protocole ont été les suivantes :

- la transmission des données en messages,
- la possibilité d'assembler plusieurs messages dans un seul trame,
- la livraison de paquets en séquence ou sans,
- la fiabilité de la transmission,
- plusieurs adresses IP pour une association[1],
- plusieurs flots de messages dans une association.

Avec le temps, SCTP est devenu un protocole d'utilisation générale du niveau transport, comme TCP et UDP, et a été spécifié dans RFC 2960 [8]. Les principales caractéristiques de SCTP sont maintenant les suivantes :

- initiation d'une association sécurisée - utilisation du mécanisme COOKIE pour la protection contre les attaques de déni du service du type *SYN-flood*,
- fermeture d'une association sécurisée - la procédure évite des connexions semi-ouvertes,
- multi-accès - possibilité d'utiliser plusieurs adresses IPv4 ou/et IPv6 pendant une association,
- multistreaming - pour la transmission de données, SCTP utilise des flots de messages ; le protocole permet d'avoir plusieurs flots indépendants dans une association.

Les différences de SCTP par rapport à TCP sont les suivantes :

- *code de détection d'erreurs* CRC sur 32 bits au lieu de 16 bits en TCP,
- flot de messages (appelés portions ou *chunks*) au lieu d'un flot d'octets en TCP. Chaque message de données porte un numéro de séquence - TSN (*Transmission*

[1]La notion d'association dans SCTP correspond à la notion de connexion dans TCP, néanmoins la notion d'association est plus large, par exemple elle peut comprendre plusieurs adresses IP.

Sequence Number). TSN est un identificateur unique de message dans une association. Le contrôle de la congestion est effectué sur la base du TSN.

Les principales caractéristiques spécifiques au SCTP sont décrites plus précisément dans les sections suivantes.

5.1 Format de l'unité de protocole

La structure d'une unité de protocole SCTP (dans la suite nous allons utiliser la notion de *message SCTP*) est modulaire. Il contient une en-tête de format commun pour tous les types possibles. Il comprend aussi une ou plusieurs portions. Deux types de portions existent : la portion de données et les portions de contrôle. Il y a un seul type de portion pour contenir des données. Pour transporter les différents messages de contrôle utilisés pendant de différentes étapes d'une association, il y a plusieurs types de portions de contrôle. La spécification initiale [8] définit 14 types des portions de contrôle, mais avec des extensions [24] [25], il y en a actuellement 17. Le tableau 5.1 présente tous les types de portions.

VALEUR	NOM	DESCRIPTION
0	DATA	Les données d'utilisateur
1	INIT	Initiation d'une association
2	INIT ACK	Acquittement de la portion INIT
3	SACK	Acquittement de la réception de données
4	HEARTBEAT	Vérification d'accessibilité d'une adresse IP de l'autre extrémité
5	HEARTBEAT ACK	Acquittement de la portion HEARTBEAT
6	ABORT	Fermeture immédiate d'une association
7	SHUTDOWN	Début de la fermeture normale d'une association
8	SHUTDOWN ACK	Acquittement de la portion SHUTDOWN
9	ERROR	Notification d'une erreur
10	COOKIE ECHO	Porte le COOKIE pendant l'initialisation d'une association
11	COOKIE ACK	Acquittement de la portion COOKIE ECHO

12	ECNE	Réservé pour utilisation dans Explicit Congestion Notification
13	CWR	Réservé pour utilisation dans Explicit Congestion Notification
14	SHUTDOWN COMPLETE	Acquittement de la portion SHUTDOWN ACK
15-62		Non attribués dans RFC2960
63		Extension réservée par l'IETF
64-126		Non attribués dans RFC2960
127		Extension réservée par l'IETF
128-190		Non attribués dans RFC2960
128	ASCONF-ACK	Acquittement de la portion ASCONF
191		Extension réservée par l'IETF
192-254		Non attribués dans RFC2960
192	FORWARD TSN	Gestion avancée des acquittements
193	ASCONF	Configuration des adresses
255		Extension réservée par l'IETF

Tab. 5.1: Les types des portions de SCTP

Les portions de données peuvent être mises dans un message avec des portions de contrôle. Il y a pourtant certaines contraintes :

- les portions du type INIT, INIT ACK et SHUTDOWN COMPLETE doivent être seules dans un paquet,
- la portion ABORT ne peut pas être mise dans le même message que les portions des données (du type DATA),
- la taille totale du message ne peut pas excéder le MTU (la taille maximale de trame - *Maximal Transmission Unit*).

La figure 5.1 présent le format général d'un message SCTP qui montre l'en-tête commune et plusieurs portions différentes.

5.2 Initialisation sécurisée

Un des points faibles de TCP est la possibilité d'effectuer une attaque de déni du service de type *SYN-flood*. Il s'agit d'une attaque où un attaquant commence plusieures connexions

avec de fausses adresses source. Comme ces adresses sont fausses, voire n'existent pas, le serveur TCP ne peut pas répondre et donc établir la connexion complète. Dans cette situation, le serveur consomme des ressources (mémoire) pour maintenir les connexions semi-ouvertes. Cela provoque un épuisement de la mémoire et finalement un blocage du serveur.

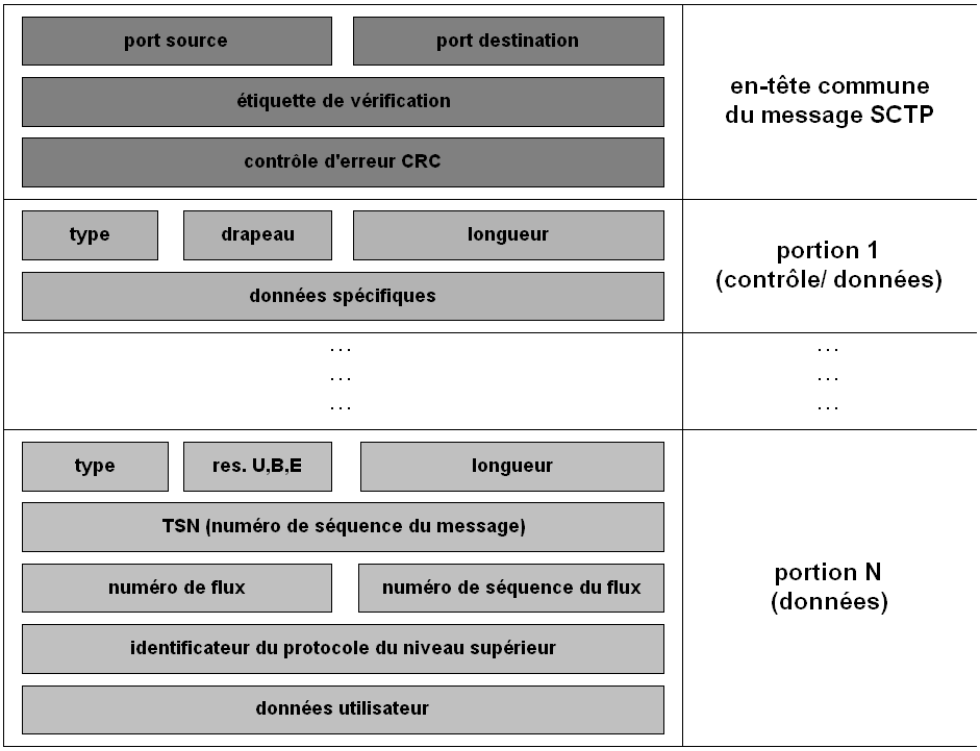

FIG. 5.1 – Format général d'un message SCTP

Pour se protéger contre ce type d'attaques, le protocole SCTP prévoit un mécanisme d'initialisation sécurisée basé sur un COOKIE. C'est une valeur calculée par le serveur pour vérifier le client qui a demandé l'initialisation d'une association. Le schéma d'initialisation avec un COOKIE est présenté dans la figure 5.2, suivi d'une explication détaillée du mécanisme.

INIT

Le client qui veut établir une association envoie vers le serveur un message INIT qui contient obligatoirement les informations suivantes :

– **Initiate Tag** - la valeur qui sera utilisée par le client comme étiquette de vérification (*Verification Tag*) dans tous les messages émis pendant l'association.

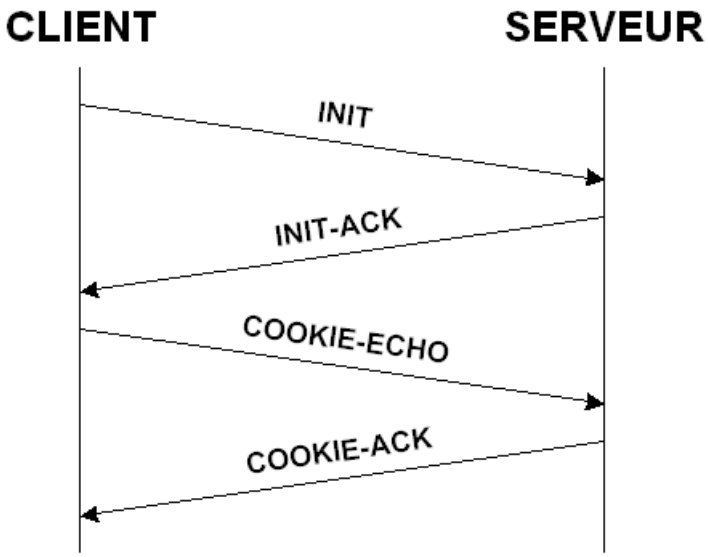

FIG. 5.2 – Initialisation de l'association SCTP

- **Advertised Receiver Window Credit** - le nombre d'octets que le client peut recevoir du serveur.
- **Number of Outbound Streams** - le nombre de flots sortants proposés par le client pour cette association.
- **Number of Inbound Streams** - le nombre de flots entrants proposés par le client pour cette association.
- **Initial TSN** - la valeur initiale du TSN pour le client. La valeur du *Initiate Tag* peut être utilisée comme le TSN initial.

Dans le message INIT le client peut aussi mettre optionnellement :

- **IPv4/IPv6 Address Parameter** - si le client veut utiliser plusieurs adresses pendant une association, il peut inclure des adresses supplémentaires IPv4 ou/et IPv6. Les deux types d'adresses peuvent être utilisés en même temps dans une association.
- **Host Name Address** - le client peut mettre son nom au lieu de son adresse IP. L'autre extrémité doit convertir ce nom en adresse IP pour établir une association. On peut utiliser cette option pour établir une association en présence d'un serveur NAT (*Network Address Translation*).
- **Supported Address Types** - le client peut spécifier les types d'adresse supportés : IPv4, IPv6 ou Hostname.
- **Cookie Preservative** - la durée de validité des COOKIES. Par défaut un COOKIE est valable pendant 60 secondes.

INIT-ACK

Un serveur qui reçoit un message INIT ne crée aucune structure de données dans sa mémoire et ne stocke aucune information concernant cette association. En se basant sur les données reçues dans le message INIT, il calcule un COOKIE. La longueur et la méthode de la calcul du COOKIE ne sont pas précisées dans la spécification et elles peuvent être différentes pour chaque implémentation de SCTP. En fait, le COOKIE est important seulement pour le serveur, car c'est lui qui doit le reconnaître dans l'étape suivante. Le client ne traite pas de COOKIE, mais le renvoie simplement au serveur. Cela ne dépend pas du format de COOKIE.

Dans le message INIT-ACK le serveur met aussi obligatoirement les informations suivantes :

- **Initiate Tag** - la valeur qui sera utilisée par le serveur comme étiquette de vérification (*Verification Tag*) dans tous les messages émis pendant l'association.
- **Advertised Receiver Window Credit** - le nombre d'octets que le client peut recevoir du serveur.
- **Number of Outbound Streams** - le nombre de flots sortants proposés par le serveur pour cette association.
- **Number of Inbound Streams** - le nombre de flots entrants proposés par le serveur pour cette association.
- **Initial TSN** - la valeur initiale du TSN pour le serveur. La valeur du *Initiate Tag* peut être utilisée comme le TSN initial.

Le serveur peut ajouter des paramètres optionnels :

- **IPv4/IPv6 Address Parameter** - si le serveur veut utiliser plusieurs adresses pendant une association, il peut inclure dles adresses supplémentaires IPv4 ou/et IPv6 comme un paramètre optionnel.
- **Host Name Address** - le serveur peut mettre son nom au lieu d'adresses IP. Le client doit convertir ce nom en adresse IP pour terminer l'initialisation.
- **Error** - si le serveur n'a pas reconnu un des paramètres dans le message INIT, il envoie un message d'erreur.

COOKIE-ECHO

En troisième phase d'initialisation le client renvoie au serveur le COOKIE trouvé dans le message INIT-ACK. Ce message ne contient rien de plus, mais le client peut mettre une portion des données (DATA *chunk*) dans le même paquet avec le message COOKIE-ECHO.

Après la réception du message COOKIE-ECHO, le serveur calcule encore une fois le COOKIE de la même façon qu'après la réception du message INIT. Si le COOKIE calculé est identique avec le COOKIE reçu dans le message COOKIE-ECHO, le serveur peut être

sûr que l'adresse du client est réelle et qu'il peut établir une association avec ce client. Dans le cas contraire, le serveur supprime silencieusement le message COOKIE-ECHO et termine l'initialisation sans établir une association.

COOKIE-ACK

Dès que le serveur reçoit un message COOKIE-ECHO il crée toutes les structures nécessaires pour servir cette association et termine l'initialisation par un message COOKIE-ACK. Dans le même message le serveur peut mettre aussi une portion de données.

5.3 Fermeture d'une association

La procédure de fermeture d'une association peut être initiée par les deux côtés d'une association (dans notre description nous supposons que c'est le client qui commence la procédure de la fermeture). Elle est sécurisée en utilisant trois messages comme montré sur la figure 5.3 et décrit ci-dessous.

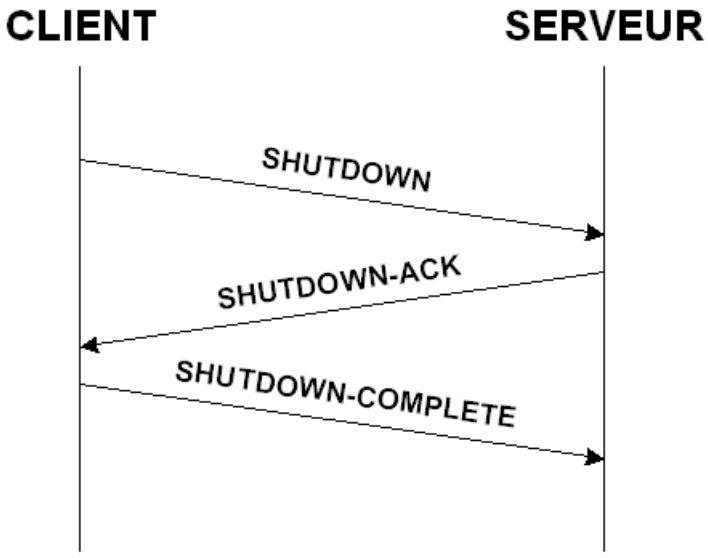

Fig. 5.3 – Fermeture d'une association SCTP

SHUTDOWN

Le client envoie un message SHUTDOWN au serveur quand il n'a plus de données à envoyer et quand toutes les données envoyées ont été déjà acquittées par le serveur. Après ce message le client n'envoie plus des données au serveur.

SHUTDOWN-ACK

En réponse d'un message SHUTDOWN, le serveur envoie un message SHUTDOWN-ACK. Le serveur peut envoyer ce message quand il a terminé d'envoyer toutes ses données au client. Après ce message, le serveur n'envoie plus de données au client.

Le message SHUTDOWN-ACK n'a aucune option.

SHUTDOWN-COMPLETE

Le client termine la procédure de fermeture en envoyant un message SHUTDOWN-COMPLETE. Cette méthode de fermeture d'une association permet d'éviter des associations semi-ouvertes. Malgré cela, il est toujours possible qu'une extrémité ne réponde pas avec un message SHUTDOWN-ACK ou SHUTDOWN-COMPLETE et continue d'envoyer des données. Dans ce cas, l'autre extrémité peut utiliser un message ABORT pour terminer l'association immédiatement.

5.4 Multistreaming

La possibilité d'avoir plusieurs flots indépendants dans une association est une caractéristique qui rend SCTP différent des protocoles de transport existants.

Contrairement à TCP où les données sont transmises en flot d'octets, dans SCTP les données sont envoyées en flot de messages. En plus, SCTP permet d'avoir plusieurs flots de données dans une association. Cela est possible grâce à l'utilisation d'un drapeau présent dans chaque portion de données (les messages de contrôle ne sont attribués à aucun flot), indiquant le numéro de flot approprié. D'après ce drapeau, l'émetteur décide dans quel flot le message est transmis et le récepteur peut reconnaître de quel flot le message vient. L'identificateur de flot est de 16 bits ce qui permet d'avoir 65536 flots séparés dans une association.

Comme la gestion d'acquittement est basée sur des flots, la transmission de chaque flot est indépendante. En cas de blocage ou de perte d'un message d'un flot, les autres flots peuvent toujours être délivrés sans contrainte. Des problèmes de transmission sont résolus au niveau du flot considéré.

Parmi des applications qui peuvent bénéficier du multistreaming, on peut mentionner des protocoles de signalisation dans des réseaux téléphoniques PSTN (*Public Switched Telephone Network*) ou des navigateurs web [26]. Dans le premier cas, ce qui est essentiel, c'est l'indépendance des flots dans une association et la possibilité de livraison de messages en dehors de séquence. Si un message est bloqué ou retardé, les autres peuvent être délivrés au destinataire. Dans le deuxième cas, le navigateur peut télécharger une page entière dans une association. Tous les objets de la page (fichiers html, images, sons, styles, etc.) peuvent être téléchargés en même temps dans des flots indépendents. Cela permet d'économiser des ressources, du côté du client aussi bien que du côté du serveur.

5.5 Multi-accès

Une connexion de TCP est définie par 4 paramètres : une adresse IP d'émetteur, un port source, une adresse IP du récepteur et un port destination. Cette façon de définir une connexion détermine deux adresses stables et inchangables pendant la connexion. Une fois les adresses définies au début d'une connexion, elles sont gardées jusqu'à sa terminaison. Dans cette logique, une connexion TCP est établie entre deux adresses IP. SCTP propose une logique différente : une association est établie entre deux structures appelées *extrémités* (*endpoint*). Une extrémité est un ensemble d'adresses de destination auxquelles on peut également envoyer des paquets ou un ensemble d'adresses source desquelles on peut également accepter des paquets. Une extrémité comprend aussi un numéro du port, qui doit être unique et ne peut pas être utilisé par une autre extrémité. Pour ce qui est des adresses, une extrémité SCTP peut utiliser une liste d'adresses IPv4, IPv6 ou des noms de machine.

La notion de *chemin*[2] est introduite pour gérer des extrémités SCTP. Elle fait appel à une structure de données créée dans une association SCTP. Un chemin est créé pour chaque adresse IP annoncée par le destinataire pendant la phase d'initialisation. Il contient une adresse IP destination et une adresse IP source qui est déterminée par les règles du routage local pour l'adresse IP de la destination donnée. Le chemin contient également des paramètres additionnels (des paramètres de contrôle de congestion, des compteurs de retransmission, le MTU etc.).

Quant à l'envoi d'un message, SCTP ne choisit pas d'adresse destination, mais un chemin. Une fois chemin choisi, le protocole met les adresses destination et source de ce chemin dans le message et l'envoie. A ce stade, le protocole n'a aucune influence ni sur l'adresse source ni sur l'adresse destination du message. Les deux adresses sont celles du chemin choisi.

[2]Le nom anglais *path* est aussi utilisé [8]. Certaines implémentations, comme lkSCTP [27], utilisent le nom *transport*.

Le multi-accès dans SCTP a été proposé dans le but d'augmenter la fiabilité du protocole. En cas de panne impliquant une adresse (interface, câble, réseau etc.), SCTP permet d'utiliser un autre chemin. De cette façon, la communication dans une association peut être maintenue même en cas de problèmes avec la connectivité impliquant une adresse.

Pour l'instant SCTP ne permet pas d'appliquer la répartition de charge ni le choix dynamique d'adresses pendant une association.

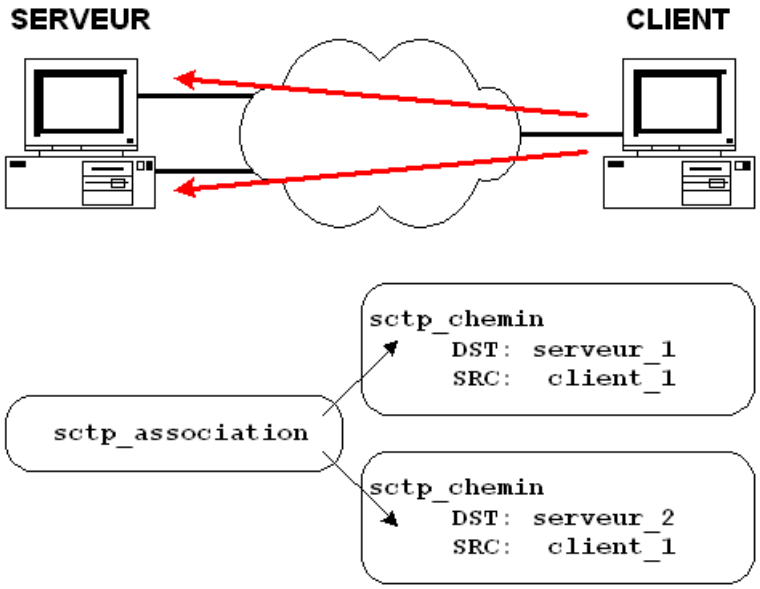

FIG. 5.4 – Une association SCTP avec deux chemins

Dans les figures 5.4 et 5.5 nous présentons deux scénarios différents. Dans la figure 5.4 nous avons un client équipé d'une carte réseau et un serveur avec deux cartes réseau. Quand le client établit une association, il crée deux chemins : un pour l'adresse `server_1` et le deuxième pour l'adresse `server_2`. Dans les deux chemins il met son adresse locale `client_1` comme l'adresse source. La deuxième figure (5.5) montre la situation inverse. Le client possède deux cartes réseau et le serveur n'en possède qu'une. Dans ce cas, le client ne crée qu'un chemin pour l'adresse `server_1`. Selon sa table du routage, il met dans ce chemin son adresse `client_1` comme l'adresse source. La deuxième adresse du client n'est pas utilisée dans ce cas. Ainsi, le client ne possède qu'un seul chemin.

Dans le chapitre 7 nous présentons des expériences effectuées dans les deux configurations.

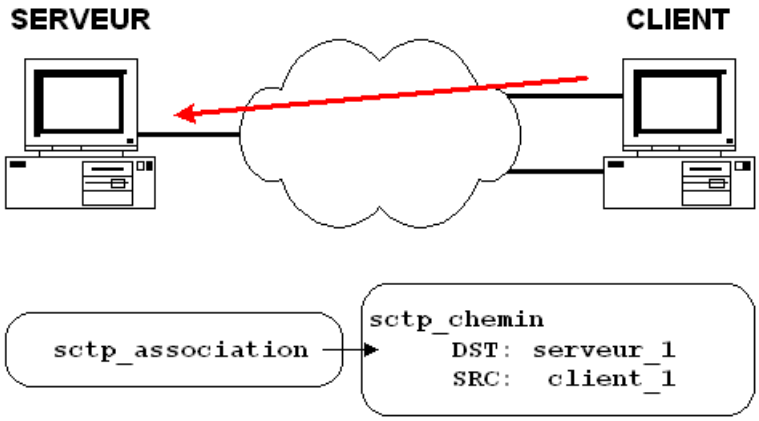

FIG. 5.5 – Une association SCTP avec un seul chemin

5.6 Multi-accès dynamique

Pendant la phase d'initialisation au début d'une association, les deux extrémités échangent des informations sur des adresses disponibles pour cette association. Les adresses établies pendant l'initialisation sont gardées jusqu'à la fin de l'association. Néanmoins, il existe une extension de SCTP permettant de reconfigurer les adresses dynamiquement lors d'une association [24]. Pour cela, deux nouvelles portions de contrôle ont été proposées. Le message du type ASCONF est utilisé pour communiquer à l'autre extrémité l'adresse qui doit être ajoutée à l'association ou supprimée. L'adresse concernée est indiquée dans le corps du message. L'autre extrémité de l'association doit confirmer la modification de l'adresse en utilisant le message du type ASCONF-ACK. Le message ASCONF sert à l'ajout et à la suppression d'adresse avec des drapeaux différents.

La fonctionalité de reconfiguration dynamique peut être particulièrement utile dans les deux cas suivants :

- **Les machines de haute disponibilité** - il s'agit de machines qui travaillent en permanence sans arrêt pour fournir des services critiques. Au niveau physique, elles permettent d'ajouter et d'enlever des elements du matériel sans perturber le travail. Le multi-accès dynamique permet de modifier la configuration des interfaces réseaux sans déranger des connexions réseau établies.

- **Le support de mobilité** - pour des machines possédant plusieurs interfaces réseau, un cas specifique de mobilité peut être supporté à l'aide du multi-accès dynamique. Sur une machine, une interface réseau est utilisée pour assurer la connectivité d'une association. En même temps, l'autre interface se connecte à un autre réseau, et une fois connecté, elle est ajoutée à l'association. La transmission de données au cours

de l'association peut être maintenant assurée par la deuxième interface. La première peut se reconnecter alors à un autre reseau.

5.7 Interface de programmation

Pour faciliter le développement d'applications, les constructeurs de SCTP ont proposé une interface de programmation (API) *socket* similaire à celle utilisée pour TCP/UDP/IPv4/IPv6. Il existe deux types d'interface de programmation dans SCTP :

- **UDP-style** - ce style est similaire au style *sans connexion* de UDP. Il offre l'accès à toutes les options spécifiques de SCTP.

- **TCP-style** - ce style offre une sémantique de type *mode connecté* de TCP. Il ne donne pas l'accès à toutes les options spécifiques de SCTP. Il est proposé pour faciliter le portage d'applications utilisant TCP à SCTP. Dans le cas le plus simple, cela se traduit par une seule modification du type de *socket* dans le code d'une application.

Toutes les fonctions et leurs options dans les deux interfaces de programmation sont compatibles avec l'interface de programmation classique des *sockets*, sauf pour les options spécifiques à SCTP.

Chapitre 6

Mesures de performance

Le multi-accès offert par SCTP permet de bénéficier de plusieurs adresses dans une association. Grâce à cela l'utilisateur peut avoir plusieurs interfaces réseau et utiliser, en même temps, plusieurs connexions indépendantes à différents réseaux. Chaque connexion peut être établie par un autre opérateur réseau en utilisant une autre interface.

Dans notre contexte, des opérateurs à très haut débit deviennent une réalité et des ordinateurs peuvent se connecter en même temps à plusieurs opérateurs offrant des caractéristiques différentes pour une connexion (le débit, la QoS offerte ou le prix). Cela donne la possibilité de choisir à tout moment la meilleure connexion pour une transmission. SCTP semble être un candidat de choix pour un protocole qui peut bénéficier de plusieurs connexions en même temps grâce à son support du multi-accès. Cet aspect nous a motivé à évaluer la performance de SCTP par une série de mesures menées sur des réseaux à très haut débit. Nous les décrivons dans ce chapitre.

Dans la première partie du chapitre nous présentons deux environnements à très haut débit établis pour la réalisation des tests. Le premier environnement s'appuie sur un réseau local 1 Gb/s et le deuxième utilise le réseau VTHD++ 2,5 Gb/s entre Grenoble et Rennes (800 km environ). Nous décrivons aussi des outils de mesure de performances élaborés par nous-mêmes dans le but d'effectuer ces tests. Dans la suite du chapitre nous présentons le surcoût des entêtes protocolaires pour les différents protocoles du transport ainsi que les différents scénarios de mesures réalisés et les résultats obtenus.

6.1 Description des environnements de test

Dans cette section nous présentons deux configurations utilisées pour les tests.

6.1.1 Environnement local

Pour les tests sur un réseau local, nous avons mis en place la configuration présentée sur la figure 6.1.

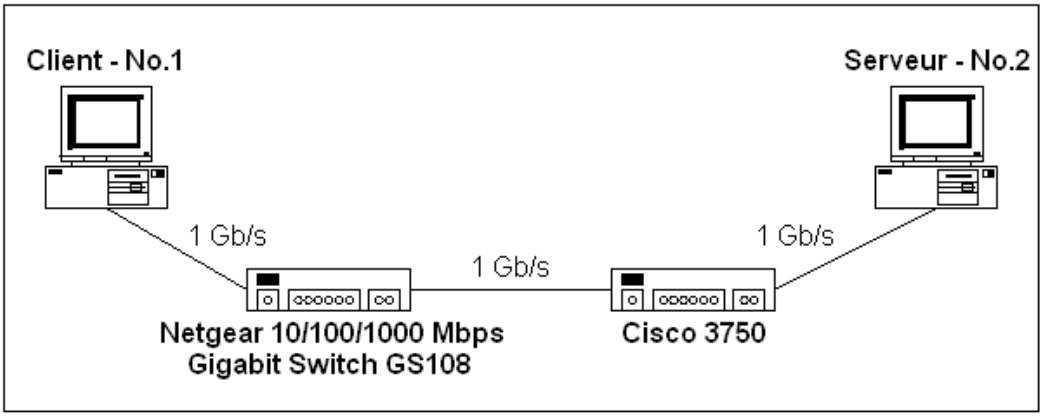

FIG. 6.1 – L'environnement pour des tests locaux

Cet environnement contenait deux machines PC connectées à l'aide d'un commutateur 1 Gb/s (Netgear 10/100/1000 Mbps Gigabit Switch GS108) et un commutateur 1 Gb/s (Cisco 3750). Les deux machines fonctionnaient sous système Linux avec le noyau 2.6.7 [28]. Cette version du noyau intègre, par défaut, une implémentation du protocole SCTP, lkSCTP en version 1.0.1 [27]. Les machines utilisées pour les tests avaient les configurations présentées dans les tableaux 6.1 et 6.2. Elles ont été équipées des mêmes cartes réseau 1 Gb/s, Broadcom Tigon3. La machine no.1 était utilisée comme le client et la machine no.2 comme le serveur pendant tous ces tests.

Pendant les tests aucune autre machine n'utilisait le commutateur Gigabit. Dans cet environnement, le délai aller-retour était de l'ordre de 0,13 millisecondes (130 microsecondes). Ce délai à été obtenu comme une moyenne de 50 paquets ICMP de la taille 10 octets envoyés par *ping*. Avec le débit nominal disponible de 1 Gb/s sur ce lien, le produit délai-bande passante est de 16,25 Koctets.

Processeur	AMD Athlon(TM) XP 1800+ 1533 MHz
Carte réseau 1 Gb	Broadcom Corporation NetXtreme BCM5701 Gigabit Ethernet (rev 15)
Autres cartes réseau	3Com Corporation 3c905C-TX/TX-M [Tornado] (rev 78)
Mémoire RAM	512 Moctets

TAB. 6.1 – Configuration de la machine no.1 - client

Processeur	AMD Athlon(TM) XP 1800+ 1533 MHz
Carte réseau 1 Gb	Broadcom Corporation NetXtreme BCM5701 Gigabit Ethernet (rev 15)
Autres cartes réseau	3Com Corporation 3c905C-TX/TX-M [Tornado] (rev 78) 3Com Corporation 3c905C-TX/TX-M [Tornado] (rev 74)
Mémoire RAM	512 Moctets

TAB. 6.2 – Configuration de la machine no.2 - serveur

Pour les protocoles en mode connecté (TCP et SCTP) qui utilisent un mécanisme du démarrage (*slow start*) on peut calculer la durée de la phase du démarrage selon la formule suivante [29] :

$$DD = RTT * log_2 \left(\frac{RTT * D_n}{MSS} \right) \tag{6.1}$$

Dans cette formule, DD est la durée du démarrage, RTT (*Round-Trip Time*) est le délai aller-retour entre deux machines, D_n est le débit nominal disponible et MSS (*Maximum Segment Size*) est la taille maximale de données qu'un protocole peut envoyer dans un paquet (1452 octets pour les messages SCTP envoyés dans l'Ethernet). Avec les paramètres de notre réseau local, nous avons la DD = 0,45 ms. Dans nos mesures, nous considérons que cette valeur est négligeable et que la procédure du démarrage n'influence pas de débits mesurés.

6.1.2 Environnement VTHD++

Pour effectuer les tests sur le réseau VTHD++ nous avons également utilisé deux machines (fig. 6.2). La machine locale était installée sur le site de l'IMAG à Grenoble. Sa configuration est présentée dans le tableau 6.2. Elle était connectée au commutateur 1 Gb et puis au réseau VTHD++ en passant par un routeur Juniper de l'IMAG. Cette machine était utilisée dans les tests en tant que serveur. La deuxième machine, distante, était installée sur le site de l'ENST Bretagne à Rennes. Sa configuration est présentée dans le tableau 6.3. Elle était aussi équipée d'une carte réseau 1 Gb (Intel Corp. 82544EI Gigabit Ethernet Controller). Cette machine était utilisée comme client pendant tous les tests décrits ci-dessous. Les deux machines travaillaient sous système Linux avec le noyau 2.6.7, intégrant lkSCTP en version 1.0.1 [27].

Entre les machines no.3 (à Rennes) et no.2 (à Grenoble) il y avait plusieurs nœuds du réseau VTHD++. Tous ces nœuds supportaient un débit de 2,5 Gb/s. On peut voir

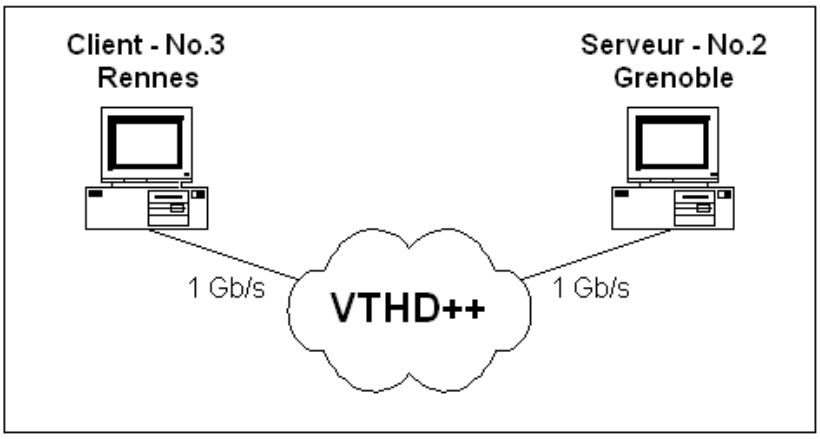

FIG. 6.2 – L'environnement VTHD++

Processeur	Pentium III (Coppermine) 933 MHz
Carte réseau 1 Gb	Intel Corp. 82544EI Gigabit Ethernet Controller (Copper) (rev 02)
Mémoire RAM	512 Moctets

TAB. 6.3 – Configuration de la machine no.3 - client à Rennes

ces nœuds intermédiaires sur les résultats de *traceroute* lancé sur la machine no.2 vers la machine no.3 :

```
traceroute to 192.108.119.6 (192.108.119.6) 30 hops max, 38 byte packets

 1  juniper (195.221.231.254)  0.316 ms  0.239 ms  0.220 ms
 2  193.252.113.189 (193.252.113.189)  0.240 ms  0.168 ms  0.162 ms
 3  193.252.113.33 (193.252.113.33)  1.457 ms  1.380 ms  1.384 ms
 4  193.252.113.25 (193.252.113.25)  6.743 ms  6.680 ms  6.668 ms
 5  193.252.113.21 (193.252.113.21)  6.842 ms  6.760 ms  6.742 ms
 6  193.252.226.229 (193.252.226.229)  12.358 ms  12.330 ms  12.292 ms
 7  193.252.113.50 (193.252.113.50)  12.473 ms  12.433 ms  12.428 ms
 8  vthd.ipv6.rennes.enst-bretagne.fr (192.108.119.94)  12.630 ms  12.655 ms  12.621 ms
 9  thieste (192.108.119.6)  13.261 ms  12.284 ms  12.315 ms
```

Pour mesurer le délai aller-retour entre les deux machines, nous avons lancé 50 fois la commande *ping* en utilisant des paquets de la taille 10 octets. Le délai moyen d'aller-retour entre les deux machines était de l'ordre de 12,8 millisecondes. Avec le débit nominal disponible de 1 Gb/s entre nos deux machines, nous obtenons le produit délai-bande passante de 1600 Koctets. D'après ces paramètres et la formule 6.1 nous pouvons calculer la durée du démarrage qui est égale à 129,35 ms. Comme cet intervalle est très petit par rapport à la durée d'une session de mesures (au moins 5 seconds), nous avons négligé son influence.

6.1.3 Logiciels de mesures - sctpperf

Pour effectuer les tests, nous avons élaboré notre propre logiciel *sctpperf* [30] qui sert à mesurer le débit de SCTP. *sctpperf* a été écrit en langage C en utilisant l'interface de *socket* [31]. Il est constitué de deux applications indépendantes : le client et le serveur. Les deux applications sont décrites dans les paragraphes suivants.

Client

Le client est une application qui permet d'envoyer des flots de données en utilisant le protocole SCTP. Il compte et affiche la bande passante calculée sur la base de la quantité de données envoyées pendant le temps indiqué dans les options. Plusieurs options permettent de modifier les conditions de travail du client. Lancé sans aucune option, le client affiche toutes les options accessibles :

```
[root@my_host]# ./clnt_bdw


Usage: ./clnt_bdw -P local_port -H local_host -B local_host -h remote_host
 -p remote_port -l message_size -t time_to_run -x period -m streams -f size_kB -v


 -P local_port       - sending port on local host
 -H local_host       - local host IPv4 address
 -B local_host1...   - local host IPv4 addresses (when > 1)
 -h remote_host      - remote host IPv4 address
 -p remote_port      - listening port on the remote host
 -l message_size     - sent message size
 -t time_to_run      - total run time
 -x period           - counting period
 -m streams          - number of streams in mutlistream mode
 -f size_kB          - send and receive buffer size in kB
 -v                  - verbose mode
```

Une fois le client lancé, il envoie le premier message marquant le début d'une transmission. Puis, en boucle, il envoie les données vers le serveur. A la fin, le client envoie un message marquant la fin de la transmission.

Sauf pour les messages marquant le début et la fin de la transmission, les données sont envoyées au serveur en messages SCTP de la taille indiquée en paramètre *-l*. La transmission dure *-t* secondes. Les résultats sont affichés chaque *-x* secondes. Les tampons d'envoi et de réception sont réglés par le paramètre *-f* (en utilisant la fonction *setsockopt()*, le client modifie les paramètres SO_RCVBUF et SO_SNDBUF de la socket qui sont responsables des tailles des tampons d'envoi et de réception). Par défaut le client utilise un seul flot de données, mais cela peut être modifié en utilisant le paramètre *-m*. L'option *-v* affiche plus d'informations.

Après la période de *-t* secondes, le client arrête d'envoyer les données, envoie un message terminant la transmission, termine son exécution et affiche les résultats.

En utilisant l'option *-B*, on peut attacher la *socket* du client à plusieurs adresses IPv4 de la machine. Cela permet de bénéficier de multi-accès de SCTP. Pour pouvoir bénéficier du multi-accès du côté serveur, il suffit de se connecter à une adresse du serveur. Si le serveur possède plusieurs adresses IP et est lancé avec l'option *-B*), il annonce ses autres adresses pendant la phase d'initialisation.

Serveur

Le serveur lancé écoute sur le port donné par le paramètre *-P*. Dès qu'il reçoit un message du début de la transmission, il commence à compter les données reçues. Toutes les secondes, le serveur affiche la bande passante intermédiaire sur la dernière seconde. Après avoir reçu le message terminant la transmission du côté client, il affiche la bande passante moyenne calculée sur la totalité du temps entre les messages du début et de la fin. Ensuite, le serveur est prêt pour une nouvelle transmission du côté client.

Les options du serveur sont montrées ci-dessous. Elles ont la même signification que les options du client :

```
[root@my_host]# ./srv_bdw

Usage: ./srv_bdw -P local_port -H local_host -B local_host1 -B local_host2 -f size_kB -v

 -P local_port       - listening port on local host
 -H local_host       - local host IPv4 address
 -B local_host1...   - local host IPv4 addresses (when > 1)
 -f size_kB          - send and receive buffer size in kB
 -v                  - verbose mode
```

De la même façon que le client, le serveur peut être lancé en mode multi-accès en utilisant l'option *-B*.

Le logiciel *sctpperf* peut être téléchargé depuis sa page web [30].

6.2 Le surcoût des entêtes protocolaires

Avant de passer aux tests, nous voulons comparer le surcoût des entêtes protocolaires de SCTP par rapport à TCP et à UDP. Dans le tableau 6.4, nous présentons les tailles des entêtes pour ces protocoles, en considérant que les protocoles du transport sont véhiculés dans les paquets IP et ensuite dans les trames Ethernet.

Ethernet	entête Ethernet	14	octets
	FCS	4	octets
	préambule	7	octets
	SFD	1	octets
	IFG	12	octets
IP	IPv4	20	octets
	IPv6	40	octets
TCP	entête TCP (sans options)	20	octets
UDP	entête UDP	8	octets
SCTP	entête SCTP	12	octets
	entête d'une portion de données	16	octets

TAB. 6.4 – Les tailles des entêtes

Dans la figure 6.3, nous pouvons voir le pourcentage effectif du débit nominal qui peut être utilisé pour véhiculer des données d'utilisateur par des protocoles différents. Ce graphe a été préparé en assumant que :

- Dans la couche liaison nous utilisons Ethernet avec son MTU (*Maximal Transmission Unit*) égal à 1500 octets, et ses entêtes qui occupent 38 octets.

- La taille minimale de données dans une trame Ethernet est de 46 octets. Les données véhiculées dans une trame Ethernet sont donc susceptibles au bourrage. Cela veut dire que les données plus petites que 46 octets sont remplies jusqu'aux 46 octets.

- Dans la couche réseau nous utilisons IPv4, dont l'entête occupe 20 octets.

- Si la taille de données est petite, SCTP peut mettre plusieurs portions de données dans un messages. Chaque portion de données comprend son propre l'entête de 16 octets.

- La taille totale d'une portion de données dans SCTP doit être une multiplication de 4. Si c'est ne pas le cas, la portion de données est remplie jusqu'à une taille la plus proche exigée.

- TCP émet des données en utilisant un paquet pour véhiculer un message. Il n'attend pas pour regrouper des petits messages pour les mettre dans un paquet. Dans le cas contraire, le TCP envoie des paquets de la taille égale à MSS et atteint le niveau de 94,92% du débit nominal.

Le graphe comprend 3 courbes qui correspondent aux TCP, UDP et SCTP respectivement. Pour les petits messages, SCTP peut en mettre plusieurs dans un paquet. De cette

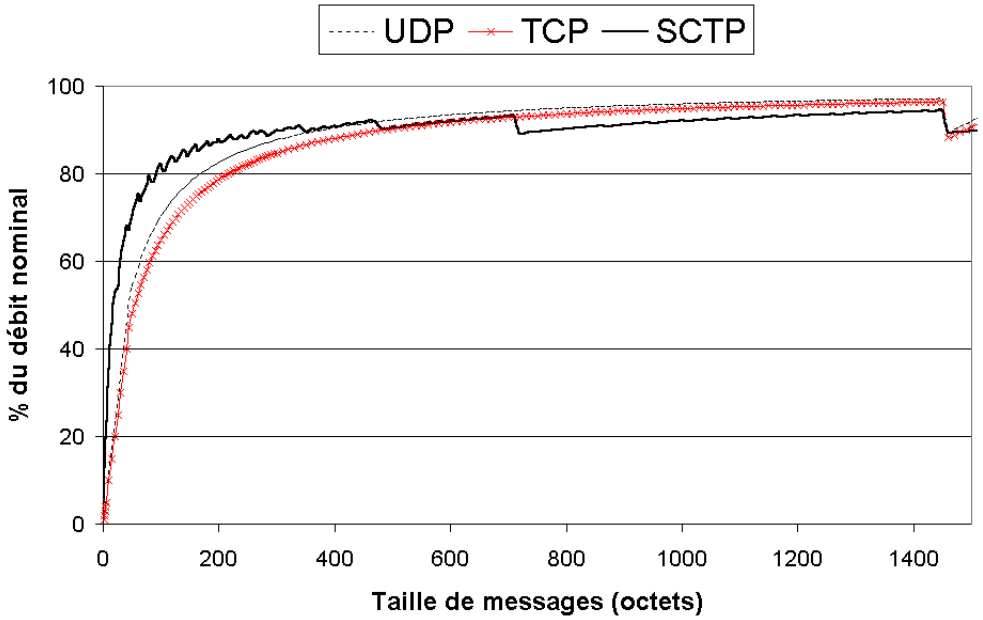

FIG. 6.3 – Le pourcentage du débit nominal disponible pour des données

façon, les paquets sont mieux remplis de données et SCTP gagne de l'efficacité par rapport aux TCP et UDP. Sur le graphe, nous pouvons observer les points ou le nombre de portions de données change avec la taille d'une portion qui augmente (cela est surtout bien visible pour les tailles entre 0 et 400 octets). Pour les grands messages (la taille > MSS/2), tous les protocoles se comportent de la même manière et véhiculent un message dans un paquet. Pour cette raison, le débit effectif dépend uniquement de la taille d'entête et est de 94,93% pour TCP, de 95,70% pour UDP et de 94,40% pour SCTP pour les messages de la taille du MSS. Comme l'entête de SCTP est le plus grand, son efficacité est la plus basse dans ce cas.

6.3 Tests effectués

Cette partie du travail concerne les tests effectués sur les configurations présentées précédemment.

6.3.1 Taille de message

Dans ces expériences, nous avons évalué l'influence de la taille du message sur le débit obtenu. Nous avons effectué des tests avec les protocoles UDP, TCP et SCTP sur le réseau

local pour pouvoir les comparer. Ensuite, nous avons évalué l'influence de la taille du message SCTP sur le débit sur le réseau VTHD++.

Pour chaque taille du message nous avons lancé une association qui envoyait pendant environ 5 secondes les messages de la taille précisée. Le nombre de messages envoyés pendant cette période de mesure oscillait de 60000 (pour les grands messages SCTP) jusqu'au 4000000 (pour les petits messages TCP). L'association SCTP utilisait un seul flot de données, les tampons d'envoi et de réception ont été fixés à 512 Ko et le MTU à 1500 octets. La taille du message variait de 10 octets jusqu'à 1450 octets pour tous les 3 protocoles. Les résultats obtenus de cette façon sont présentés sur la figure 6.4.

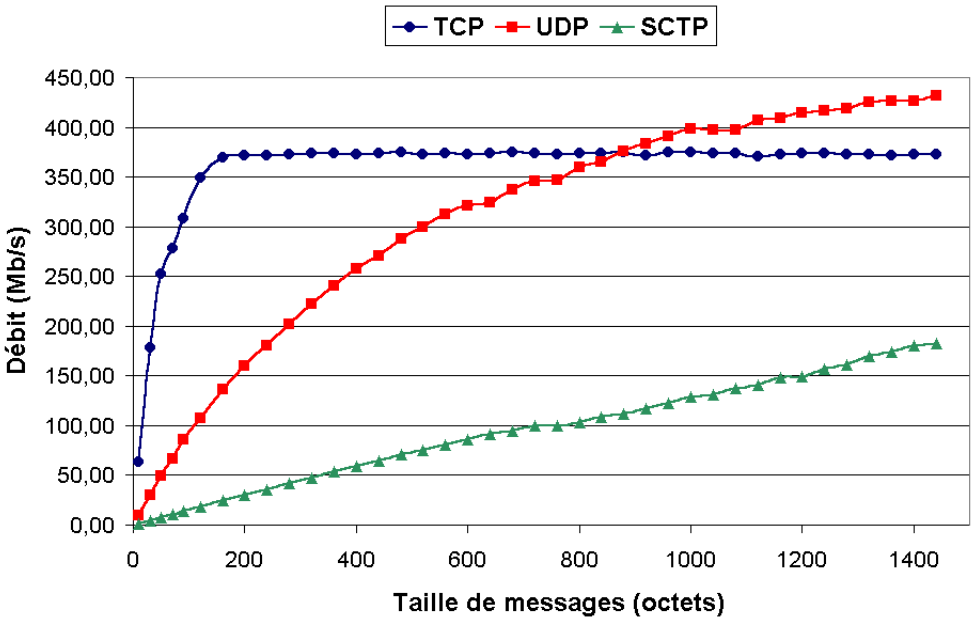

FIG. 6.4 – Effet de la taille de message sur le réseau local

Sur le graphe, on peut voir que la performance du SCTP est la plus basse entre les 3 protocoles et très en dessous du débit nominal (c.f. figure 6.3). Comme tous ces 3 protocoles ont été évalués dans les mêmes conditions, la faible performance du SCTP est probablement due à l'implémentation qui est assez récente et peut souffrir du manque de l'optimisation (ce qui n'est pas le cas pour TCP et UDP). C'est pour cela qu'elle ne permet pas d'atteindre la même performance que TCP ou UDP. On peut également observer que pour de petites tailles de messages (inférieures à 800 octets), les performances de TCP sont meilleurs que UDP. Ceci vient du fonctionnement de TCP orienté "flot d'octets" : même si l'application écrit des messages de petite taille, TCP les regroupe dans un paquet ce qui évite le surcoût de l'entête protocolaire.

Cette explication peut être confirmée par l'autre test effectué sur le réseau VTHD++ (figure 6.5). Dans ce graphe nous voyons bien les points de fragmentations de paquets. Nous

voulons surtout attirer l'attention sur le premier point de la fragmentation qui est au niveau du premier MTU (1500 octets). Après ce point nous observons une dégradation du débit très importante. Cela montre, que le coût de la création et de l'envoi d'un nouveau paquet est très grand et influence beaucoup le débit. Une optimisation éventuelle de l'implémentation du STCP reste alors à étudier.

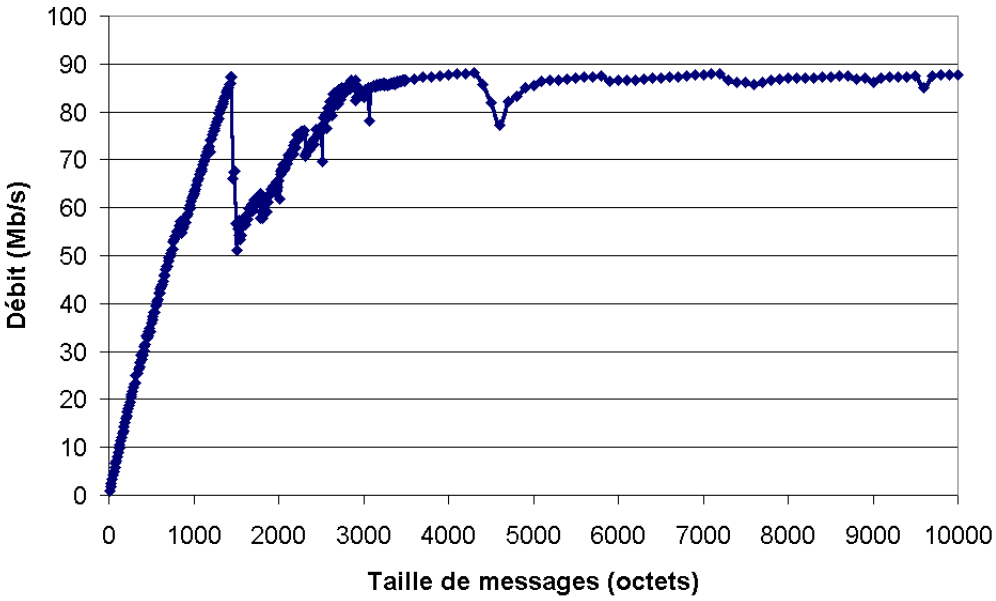

FIG. 6.5 – Effet de la taille de message sur le réseau VTHD++

6.3.2 Nombre de flots

Une nouvelle fonctionnalité offerte par SCTP est une possibilité d'envoyer plusieurs flots de données indépendants dans une association. Dans cette expérience, nous avons mesuré l'influence de plusieurs flots dans une association sur le débit obtenu.

Pour gérer les flots multiples, SCTP attribue chaque portion de données à un flot. Même quand il y a un seul flot, la portion de données porte l'identificateur de ce flot (qui est égale à 0 dans ce cas). S'il y a plusieurs flots dans une association, chaque portion porte l'identificateur de son propre flot. Comme on peut voir dans la figure 6.6, le champ d'identificateur de flot a 16 bits de longueur, ce qui permet d'avoir jusqu'à 65536 flots indépendants dans une association. L'identificateur de flot est obligatoire, donc il est présent dans chaque portion de données.

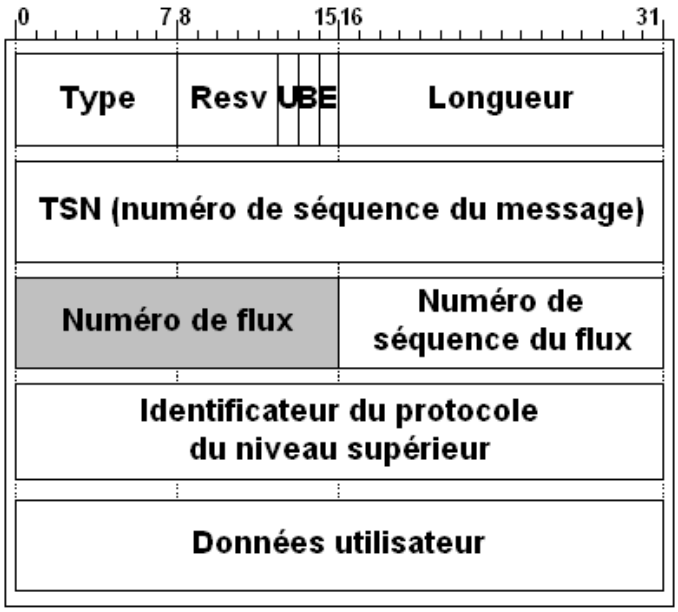

FIG. 6.6 – L'identificateur de flot dans une portion de données

Dans l'expérience, nous avons comparé le débit d'associations ayant un nombre différent de flots. La taille des messages était de 1000 octets et les tampons d'envoi et de réception étaient de 512 Ko pendant tous les tests présentés ici.

La figure 6.7 présente deux graphes avec les débits dans les deux environnements. Comme on peut voir, en aucun cas le débit ne dépend pas du nombre de flots qui sont servis par une association. Au niveau SCTP la bande passante est stable par rapport au nombre de flots. Elle atteint 62 Mb/s pour VTHD++ et 122 Mb/s pour le réseau local.

Cela s'explique par la façon dont la gestion de flots est faite au sein du protocole SCTP. Pour attribuer un message à un flot, le protocole met son identificateur dans l'entête. Cela est fait pour chaque message qui véhicule les données, peu importe le nombre de flots dans une association. Cette opération n'est pas complexe au niveau de calcul, donc elle n'a pas d'influence sur les performances.

6.3.3 Taille des tampons d'envoi et de réception

Pour évaluer l'influence des tampons d'envoi et de réception sur le débit, nous avons utilisé une association SCTP avec un seul flot de données. Pour chaque taille des tampons d'envoi et de la réception (8Ko, 16Ko, 32Ko, 64Ko, 128Ko, 256Ko, 512Ko), nous avons lancé une nouvelle association pour une période de 5 secondes (la mesure a été prise entre le temps t_0=0s et t_1=5s). La taille des tampons a été réglée en utilisant l'option *-f* dans

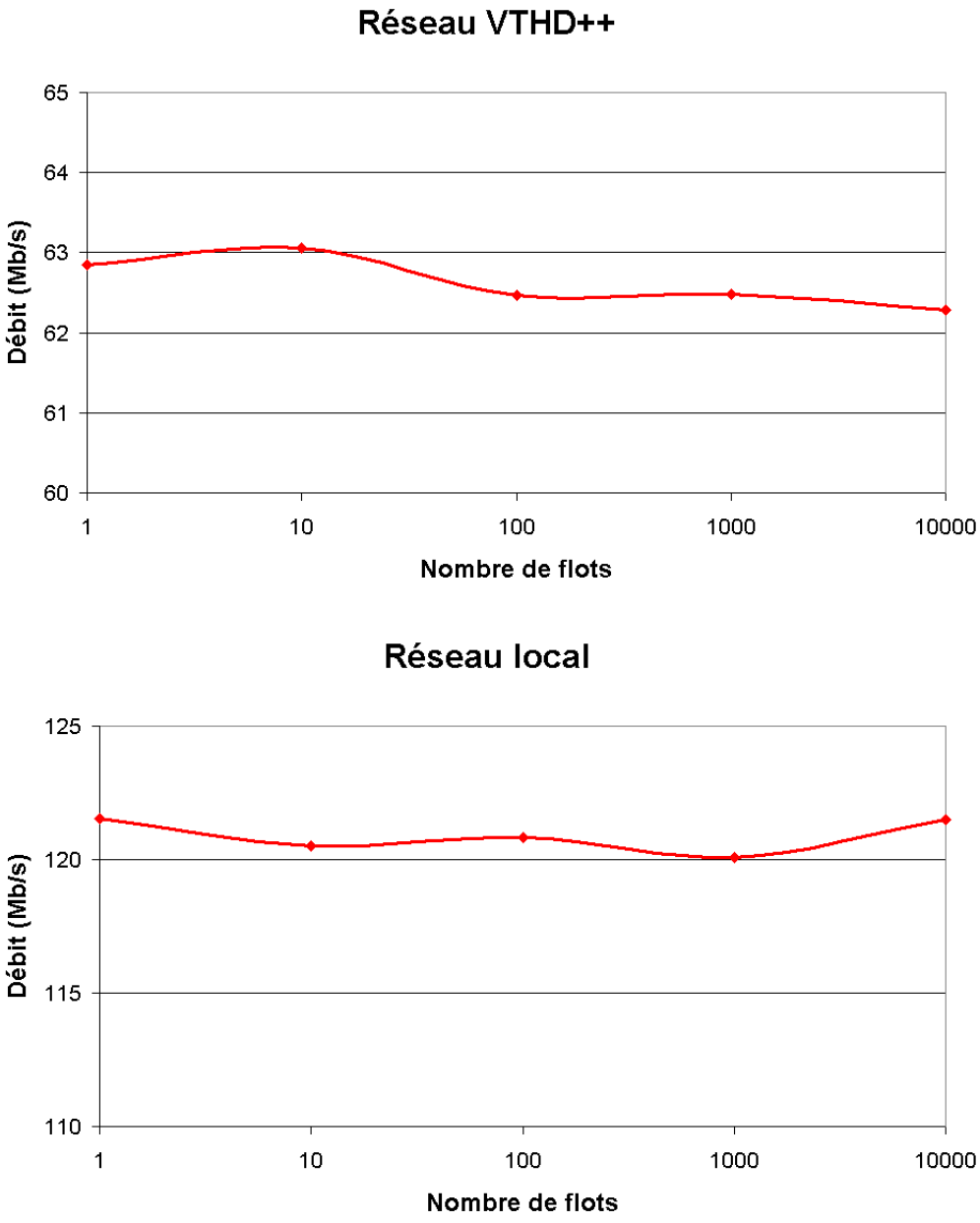

FIG. 6.7 – Débit de plusieurs flots par association

l'application du client et du serveur. La taille de données SCTP était de 1000 octets pour toutes les mesures.

Dans la figure 6.8 on peut voir les résultats pour le réseau VTHD++ et pour le réseau local. Sur VTHD++ on voit mieux la différence entre les tampons de petite et de grande taille. Pour les tampons de 8 Ko, on peut observer un débit de 8 Mb/s et pour les tampons de 512 Koctets un débit de 63 Mb/s. Sur le réseau local le débit monte légèrement avec la taille de tampons d'envoi et de la réception. La courbe de débit commence au niveau de 117 Mb/s pour les tampons de 8 Koctets et atteint 122 Mb/s pour les tampons de 512 Koctets.

Cela montre qu'un petit tampon peut être un facteur limitant le débit atteint. En augmentant la taille des tampons, nous pouvons gagner du débit jusqu'au certain niveau. À partir de ce moment, la taille des tampons n'influence plus le débit, parce que les autres facteurs deviennent plus importants (par exemple le délai aller-retour). Sur le réseau VTHD++, nous pouvons observer qu'au-dessus de 64 Koctets le débit est stable. Dans le réseau local, cette influence est moins visible. Comme le délai aller-retour est plus petit dans le réseau local, la taille des tampons est moins importante.

6.3.4 Associations SCTP simultanées

Pour évaluer le comportement des plusieurs associations SCTP simultanées, nous avons lancé n serveurs sur une machine et n clients sur une autre. Cela nous a donné n associations indépendantes sur le même réseau. Dans ce test, toutes les n associations ont fonctionné en même temps pendant 25 seconds. Toutes les n associations ont été identiques (la même taille de messages, la même taille de tampons, un seul flot des données). Dans la figure 6.9 on peut voir les résultats de ce test effectué sur le réseau VTHD++. Le débit cumulé de toutes les associations est la somme de tous les débits individuels observés sur toutes les n associations.

Le débit maximal observé est de 153 Mb/s. Nous voyons que le débit cumulé augmente avec le nombre d'associations. Le maximum a été obtenu pour neuf associations concurrentes. Nous pouvons expliquer cet événement de manière suivante. Une association possède toujours des périodes inactives qui viennent du fonctionnement du système d'exploitation, des interruptions hardware, etc. Pendant ces périodes l'association n'envoie pas de données. Néanmoins, s'il y a une autre association qui fonctionne en même temps, elle peut envoyer ses données dans ces périodes. De cette façon, plusieurs associations simultanées peuvent mieux utiliser le débit disponible. Le débit a été également partagé par toutes les associations.

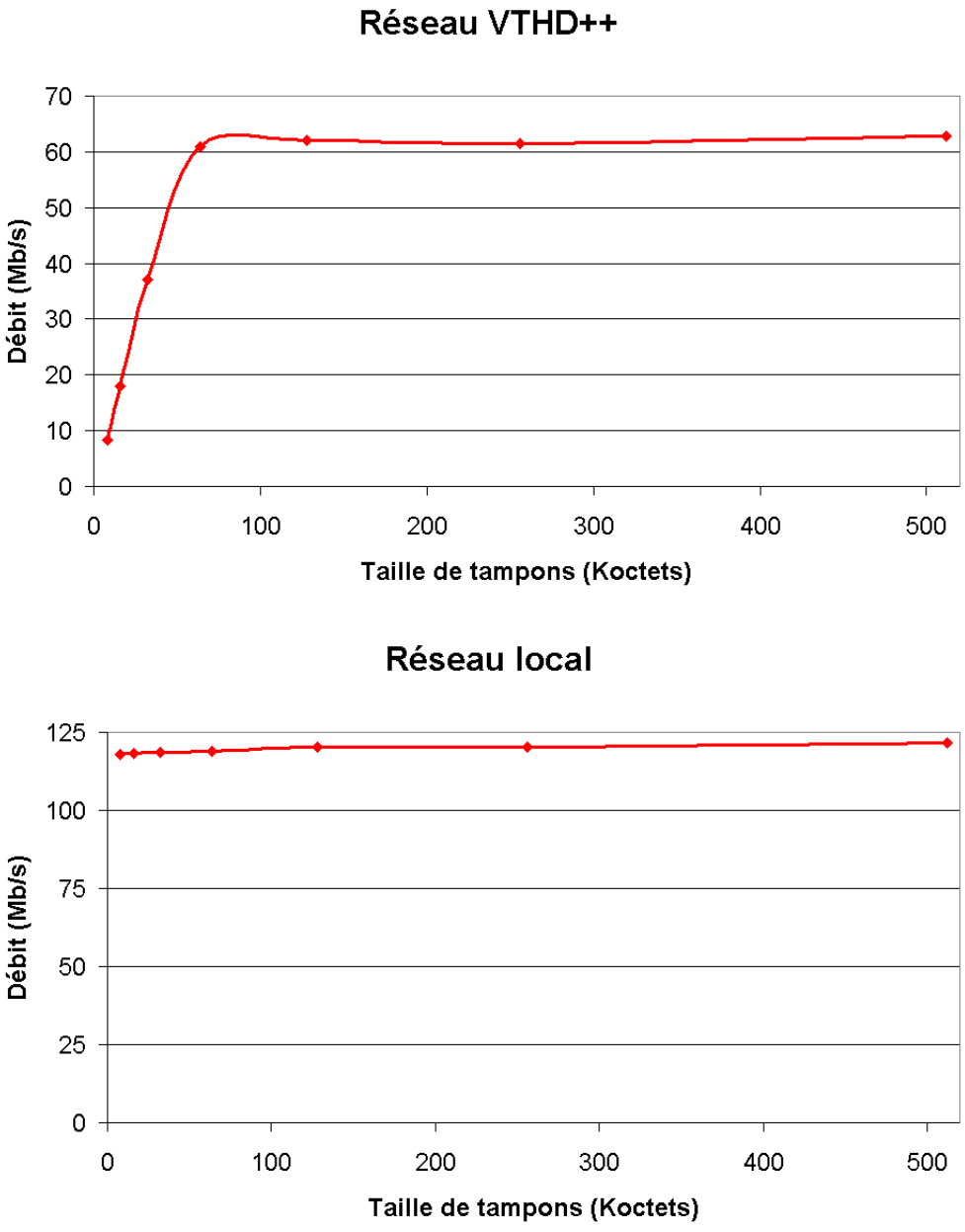

FIG. 6.8 – Débit par rapport aux tampons d'envoi et de la réception

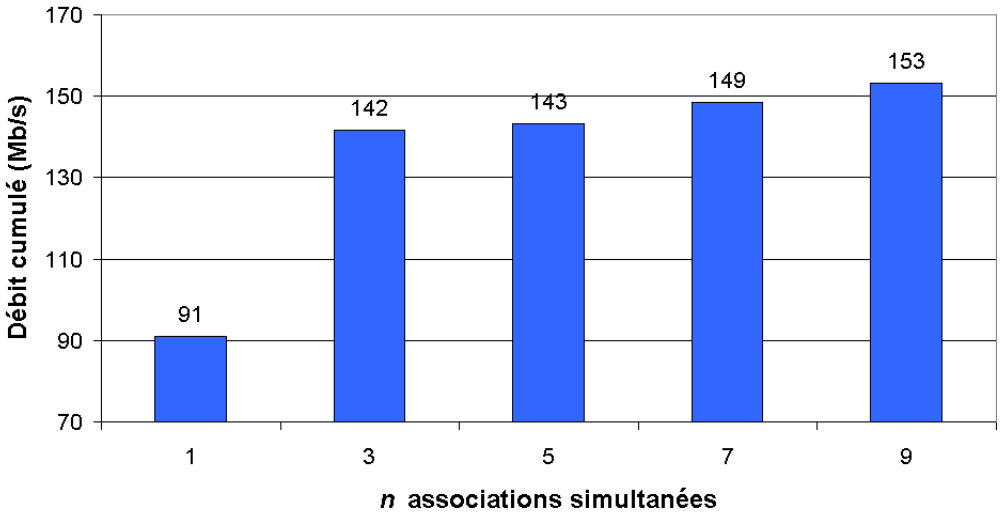

FIG. 6.9 – Le débit cumulé des associations simultanées sur le réseau VTHD++

6.3.5 Associations concurrentes

Cette section décrit des expériences dans lesquelles deux associations rivalisent pour le débit disponible sur le même réseau. Tout d'abord nous présentons les expériences avec deux associations SCTP et ensuite les expériences avec une association SCTP rivalisant avec une connexion TCP. Ces expériences ont été menées sur le réseau local 1 Gb/s.

SCTP en concurrence avec SCTP

Dans ces expériences, nous avons observé deux associations SCTP concurrentes rivalisant pour le débit sur le même réseau. Pour cela, nous avons lancé une association SCTP qui utilisait une taille de message de 1000 octets. Après quelques secondes, nous avons lancé une deuxième association avec des messages de 100 octets (dans la deuxième expérience, la deuxième association utilisait des messages de 1000 octets). La deuxième association a fonctionné pendant 20 secondes. Après la fin de la deuxième association, la première a encore fonctionné pendant quelques secondes.

Les résultats sont présentés sur la figure 6.10. La courbe rouge (les losanges) représente la première association, la courbe bleue (les carrés) représente la deuxième association et la courbe verte (les triangles) le débit cumulé de deux associations.

Dans la première expérience, on voit le débit de la première association de 120 Mb/s. Après le lancement de la deuxième association, ce débit a baissé jusqu'à 56 Mb/s. Avec 7 Mb/s pour la deuxième association cela a donné 63 Mb/s, ce qui n'a pas atteint le

débit du début de l'expérience. Cette mesure montre qu'une association SCTP qui utilise de petits paquets est moins performante qu'une autre utilisant de grands paquets. La première ne peut pas rivaliser au même niveau avec la deuxième et donc ne peut pas obtenir le même débit. La deuxième association, qui génère de petits paquets, introduit une charge importante sur la machine émettrice, c'est qui est responsable de la dégradation du débit cumulé.

Dans la deuxième expérience les deux associations envoyaient des messages de même taille, 1000 octets. Comme auparavant, la première association a atteint le débit de 120 Mb/s pour baisser jusqu'à 56 Mb/s après lancement de la deuxième. La deuxième association s'est stabilisé au même niveau de 56 Mb/s, ce qui donne 122 Mb/s comme débit cumulé de deux associations. Dans ce cas, nous voyons que le deux associations identiques (utilisant la même taille des paquets et chargeant la machine émettrice de la même façon) ont partagé également les ressources disponibles. La première association a détecté correctement l'apparition d'une deuxième et a ajusté son débit pour lui permette d'atteindre le même niveau. Ce mécanisme est connu de TCP et permet aux associations de rivaliser pour gagner le débit dans les réseaux ou nombreuse associations coexistent.

SCTP en concurrence avec TCP

Les expériences suivantes montrent une association SCTP qui partage les ressources réseau avec une connexion TCP. Une fois que l'association SCTP a marché pendant 20 secondes, la connexion TCP a été lancé pour une période de 20 seconds. Nous avons effectué deux tests analogiques en changeant la taille des messages envoyés. Nous avons choisi de petits messages (de 10 octets) et de grands messages (de 1450 octets). Aussi bien SCTP que TCP envoyaient un message dans un paquet. Les résultats des deux tests sont présentés sur la figure 6.11.

En utilisant de grands messages (de 1450 octets) une association SCTP a pu arriver au niveau de 147 Mb/s au début de l'expérience et a baissé à 67 Mb/s après le lancement de la connexion TCP. La connexion TCP a atteint un débit de 270 Mb/s. Alors, le débit disponible n'est pas partagé de façon égale entre SCTP et TCP. Dans le deuxième graphe, nous pouvons voir le même phénomène pour l'expérience avec de petits messages (de 10 octets). L'association SCTP n'a pas pu arriver au même débit que TCP. Cela peut être dû à l'implémentation qui n'est pas, pour l'instant, si optimisé et performante que celle du TCP.

6.3.6 Calcule du code de détection d'erreurs

Pour des raisons de sécurité présentées par Stone [32], l'algorithme original d'Adler-32 [33] a été remplacé par CRC-32c [34] pour calculer le code de détection d'erreurs dans les messages SCTP [35]. Comme l'algorithme CRC-32 est plus complexe qu'Adler-32, nous avons

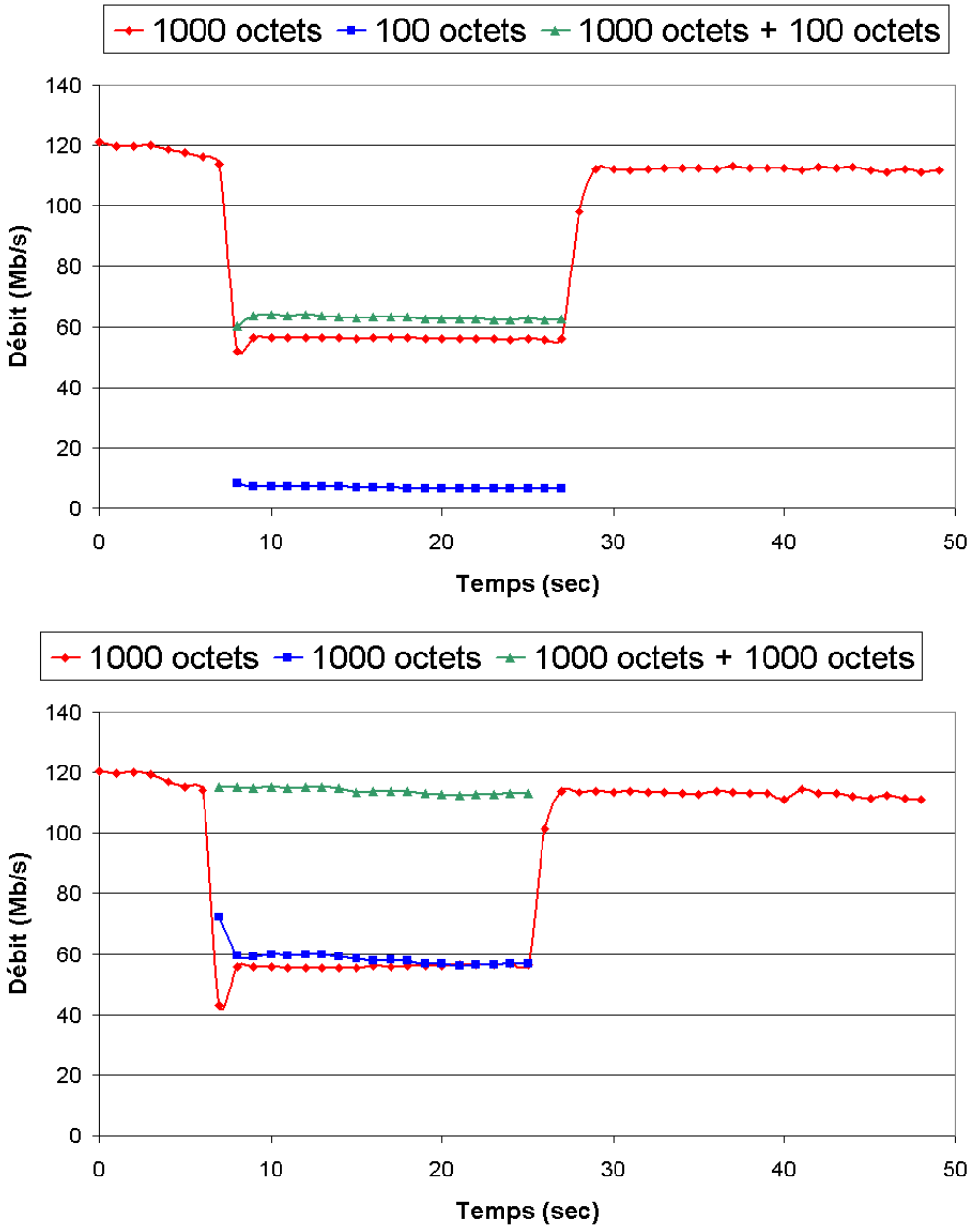

FIG. 6.10 – Deux associations SCTP concurrentes

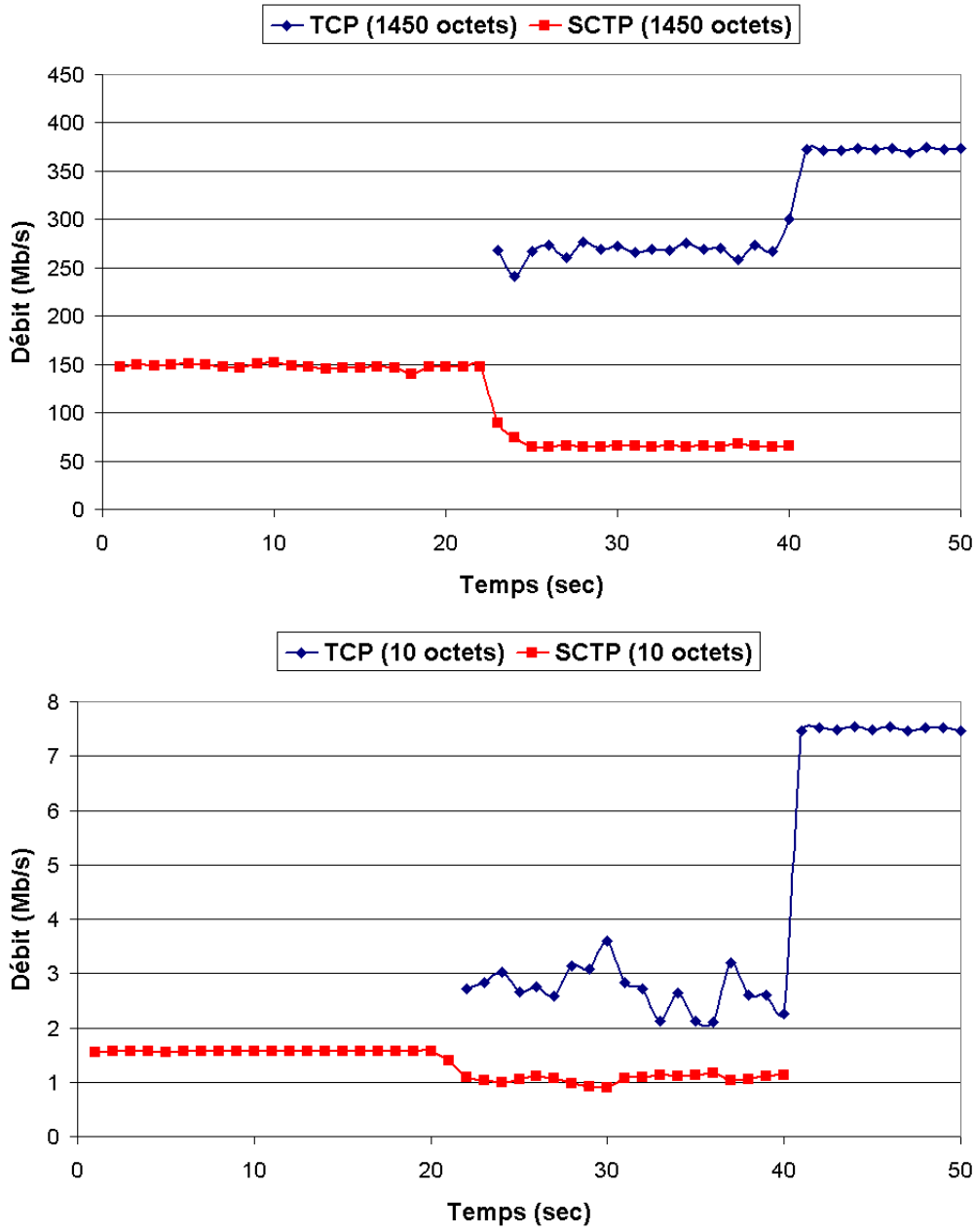

FIG. 6.11 – Une association TCP concurrente avec SCTP (messages de 10 et 1450 octets)

décidé d'évaluer l'influence de cet algorithme sur la performance du SCTP. En utilisant l'implémentation lkSCTP, qui donne une possibilité de choisir l'algorithme pour calculer le code de détection d'erreurs, nous avons préparé trois versions SCTP différentes :

Adler-32 - une version qui utilisait l'algorithme Adler-32,

CRC-32 - une version qui utilisait l'algorithme CRC-32c,

Aucun code de détection d'erreurs - dans la troisième version nous avons introduit notre propre algorithme pour calculer le code de détection d'erreurs. Cet algorithme donne toujours zéro à la sortie et ne fait aucun calcul.

Avec ces trois versions, nous avons effectué des mesures de performance. Les mesures ont été faites sur une seule machine en utilisant une interface loopback avec MTU de 16436 octets. Cela nous a permis d'éviter l'influence du réseau sur les résultats. Pour chaque version, nous avons mesuré le débit pour trois tailles de paquets différentes, 10, 1000 et 10000 octets respectivement. Chaque mesure durait 30 seconds et a été répété 10 fois. La table 6.5 présente les moyennes et les écart-types sur 10 mesures pour chaque taille de paquets et chaque algorithme.

Taille des paquets (octets)	Adler-32		CRC-32		Aucun	
	débit (Mb/s)	l'écart-type (Mb/s)	débit (Mb/s)	l'écart-type (Mb/s)	débit (Mb/s)	l'écart-type (Mb/s)
10	1,26	0,0073	1,21	0,0174	1,29	0,0079
1000	106,27	0,6252	98,84	16,4168	125,44	0,6711
10000	425,97	6,2317	458,39	10,6403	1036,60	7,3192

TAB. 6.5 – L'influence du code de détection d'erreurs sur le débit

D'après ces résultats, nous pouvons constater que le changement de l'algorithme n'influence pas les performances du SCTP. Vu les écart-types obtenus, les différences sont négligeables pour des paquets de la taille petite et moyenne. Nous pouvons observer une différence plus importante pour les grands paquets de la taille de 10000 octets. Néanmoins, les résultats pour les paquets de cette taille restent théoriques, car cette taille dépasse largement 1500 octets du MTU d'Ethernet. Même si l'algorithme CRC-32 est plus complexe qu'Adler-32, son implémentation, bien optimisée et performante, ne dégrade pas la performance du SCTP.

Dans la dernière colonne, nous voyons la performance du SCTP quand aucun code de détection d'erreurs n'est calculé. Tandis que pour les petits paquets la différence est négligeable, elle commence à être visible pour les paquets de la taille moyenne et est encore plus importante pour les grands paquets. Cela montre que le calcul du code de détection d'erreurs coûte cher, surtout quand la taille de paquets augmente, et qu'on peut réfléchir comment l'améliorer. Sachant que l'implémentation d'algorithme CRC-32 est déjà optimisée et performante on peut, par exemple, chercher à le réaliser en matériel.

6.4 Conclusion

Dans les expériences menées avec le protocole SCTP, nous avons pu évaluer ses performances. En confrontation avec TCP et UDP, nous pouvons constater que la performance de SCTP testé n'est pas encore comparable parce que son implémentation n'est pas encore si performante que celle de TCP ou d'UDP. Par rapport à TCP et UDP, SCTP offre de nouvelles fonctionnalités, notamment le multistreaming dont les performances on a pu évaluer. Dans nos mesures, nous avons observé que l'introduction de plusieurs flots dans une association est bien gérée et n'influence pas les performances de la transmission.

En ce qui concerne la coexistance de plusieurs transmissions sur un réseau, nous avons évalué des associations simultanées et concurrentes. Plusieurs associations sur le même réseau ont pu atteindre le débit cumulé plus grand qu'une seule association. Le débit a été également partagé par des associations qui ont été identiques (la même taille des messages, la même taille des tampons) ce qui montre que le contrôle de congestion fonctionne et détecte correctement d'autres associations. Pour les associations différentes, une association favorisée (qui a utilisé des messages plus grands) a pu atteindre le débit plus important qu'une association défavorisé (qui a utilisé des messages plus petits). En partagant les ressources réseau avec TCP, une association SCTP n'a pas pu avoir le même débit a cause de son implémentation moins performante.

Nous avons également évalué l'influence du changement de l'algorithme pour calculer le code de détection d'erreurs. L'introduction de l'algorithme CRC-32, même que plus complexe que l'algorithme ancien d'Adler-32, ne dégrade pas les performances du SCTP. Néanmoins, le coût de calcul du code de détection d'erreurs est remarquable et on peut réfléchir comment le baisser.

Nos mesures montrent que SCTP est un protocole qui offre de nouvelles fonctionnalités intéressantes mais sa performance générale reste encore à améliorer.

Chapitre 7

Multi-accès dans SCTP

SCTP propose une nouvelle méthode pour gérer l'adressage dans la couche transport. Une machine d'un utilisateur peut être identifié et accessible par plusieurs adresses IP. Cela se traduit par une association SCTP dans la couche transport qui est attachée à plusieurs adresses de la couche réseau (dans ce cas, plusieurs adresses IPv4 ou IPv6). Toutes ces adresses sont également valables pour la transmission de données. En cas de panne concernant une adresse, l'association peut rétablir la transmission en utilisant une autre adresse de manière transparente pour une application. Cela permet d'augmenter la fiabilité de la connexion et d'éviter les problèmes qui peuvent apparaître dans le réseau.

Pendant nos études sur le protocole SCTP, nous avons pu observer deux situations opposées dans le fonctionnement du multi-accès. Dans la première, le multi-accès de SCTP fonctionne correctement et permet de récupérer la transmission en cas de panne d'une adresse en utilisant une autre. L'autre situation est défavorable. Dans ce cas, le multi-accès du SCTP ne peut pas aider même si l'association possède plusieurs adresses. Nous identifions ce cas comme un problème de gestion d'adressage dans SCTP.

Dans ce chapitre, nous présentons deux expériences qui mettent en évidence les deux situations mentionnées. Nous commençons par la description de la configuration préparée (section 7.1), puis nous présentons les tests effectués dans cette configuration et leurs résultats (sections 7.2 et 7.3).

Dans le chapitre 8 nous proposons une solution pour résoudre le problème identifié ici.

7.1 Configuration de tests

La configuration utilisée dans les tests est présentée sur la figure 7.1. Nous avons utilisé deux machines connectées au réseau local. Le client a été équipé d'une carte réseau tandis que le serveur avait deux cartes réseau. Les conditions de travail et de charge de deux

réseaux pendant les tests ont été différentes. Cela a créé un débit différent sur les deux chemins existants entre le client et le serveur.

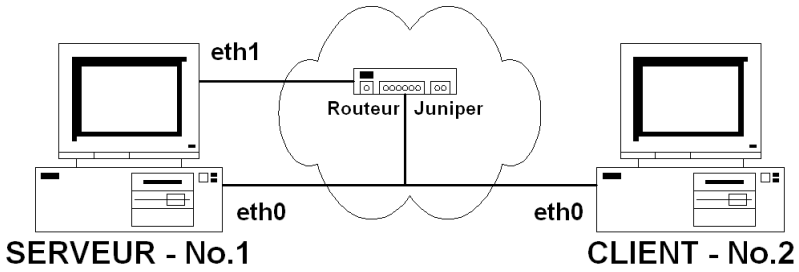

FIG. 7.1 – La configuration utilisée pour les tests

Comme pour les mesures présentées dans le chapitre 6, nous avons utilisé le noyau Linux en version 2.6.7 avec lkSCTP en version 1.0.1.

Pour le serveur nous avons configuré la table de routage suivante :

```
DESTINATION      INTERFACE

client_eth0      server_eth0
```

Selon cette table du routage, tout le trafic sortant du serveur vers le client partait vers l'interface `server_eth0`.

Sur la machine client, nous avions la table du routage suivante :

```
DESTINATION      INTERFACE

server_eth0      client_eth0
server_eth1      client_eth0
```

Comme le client ne possédait qu'une seule carte de réseau, tout le trafic sortait par cette interface (`client_eth0`).

7.2 Exemple de bon fonctionnement

Pour la première expérience, nous avons lancé le serveur SCTP sur la machine serveur. La socket SCTP a été attachée à deux adresses IP du serveur. Sur la machine client nous

avons lancé SCTP avec l'adresse distante du serveur `server_eth1`. De cette façon, une association SCTP a été établie entre `client_eth0` et `server_eth1`.

Selon les tables du routage, le client envoyait les données vers le serveur par l'interface `client_eth0` et le serveur envoyait ses acquittements par l'interface `server_eth0`. La route n'était pas symétrique.

Après quelques secondes nous avons interrompu la connexion entre le `client_eth0` et `server_eth1`. Pour simuler cette panne, nous avons débranché le câble du commutateur du côté `server_eth1`. Bien que l'interface `server_eth1` ait été toujours opérationelle, cela a interrompu la transmission des données.

SCTP a commencé les retransmissions pour rétablir la transmission. Le paramètre *'Path.Max.Retrans'* indique le nombre des retransmissions avant le changement du chemin [8]. La valeur de ce paramètre par défaut est cinq (modifiable dans lkSCTP par un variable `/proc/sys/net/SCTP/path_max_retrans`). Cela veut dire que les cinq premières retransmissions sont envoyées sur le même chemin et la sixième sur un autre chemin. Cela permet de rétablir la transmission sur un autre chemin au bout de 63 secondes.

Comme on peut voir dans la figure 7.2, dans ce test la transmission a été rétablie après 63 secondes.

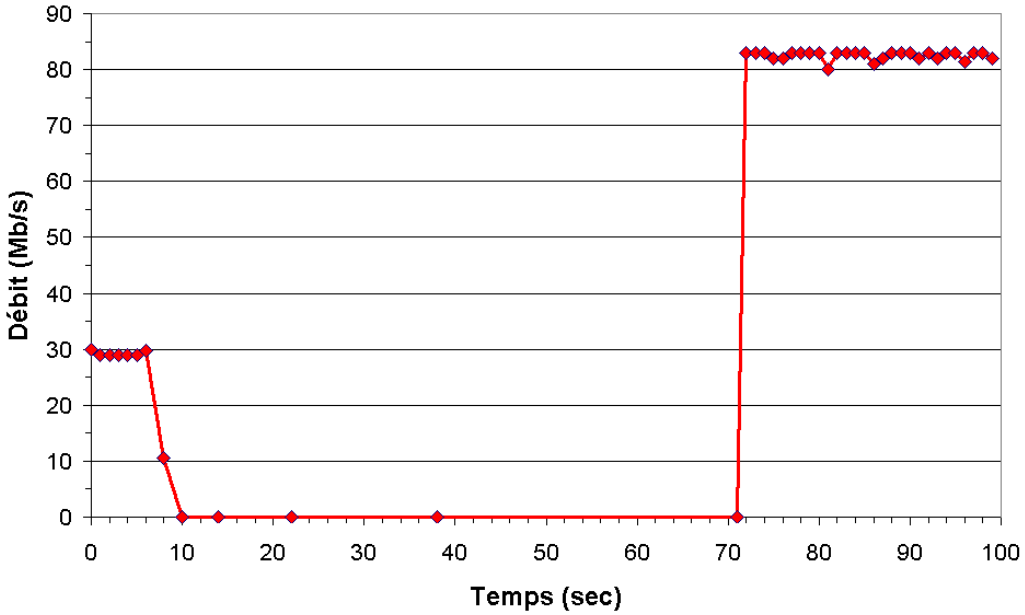

FIG. 7.2 – Le cas de bon fonctionnement du multi-accès

Après le rétablissement de la transmission, le client envoie les données sur l'interface `client_eth0`. Le serveur envoie ses messages d'acquittement par l'interface `server_eth0`.

La transmission est donc symétrique. Sur la figure, on voit une autre valeur de le débit, qui vient des conditions réseau sur le nouveau chemin.

7.3 Exemple d'un fonctionnement incorrect

La deuxième expérience a été menée dans la même configuration. Les tables de routage étaient les mêmes. La seule différence était la suivante : nous avons lancé le client avec l'adresse distante du `server_eth0`. Dans ce cas-là, le client envoyait toutes ses données vers l'interface `client_eth0` à l'adresse `server_eth0` et le serveur envoyait tous les messages d'acquittement vers l'interface `server_eth0`. La transmission était donc symétrique. L'interface `server_eth1` du serveur n'était pas utilisée dans cette transmission.

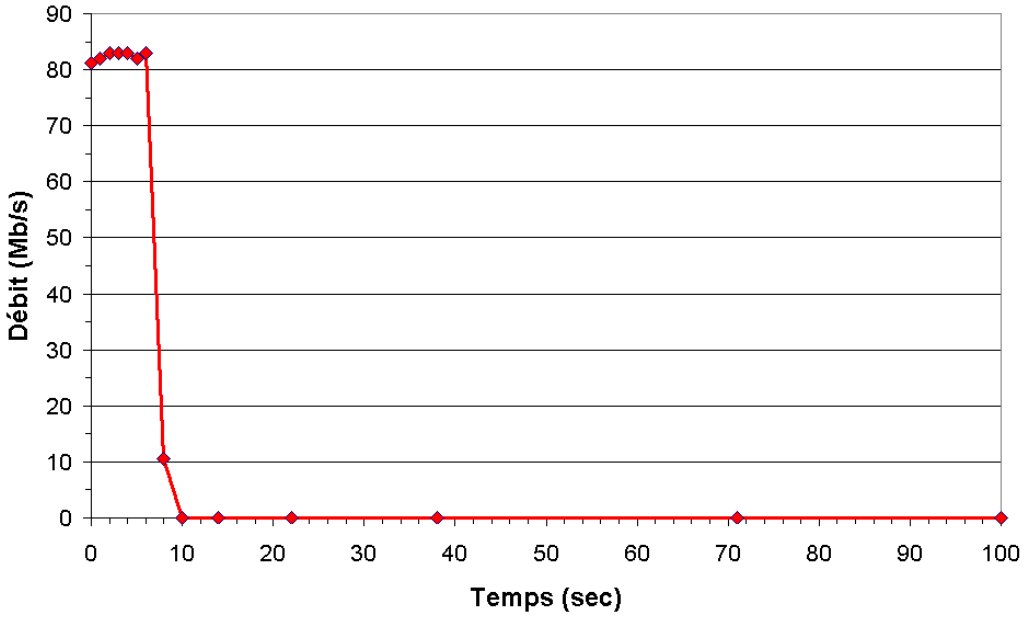

FIG. 7.3 – Le cas d'un fonctionnement incorrect du multi-accès

De la même façon qu'auparavant, nous avons débranché un câble du commutateur pour simuler une panne dans le réseau : le câble du côté `server_eth0`. Cela a interrompu la transmission des données entre `client_eth0` et `server_eth0`. Toute de suite après que la panne soit apparue, le protocole SCTP a commencé les retransmissions. Il a envoyé 5 retransmissions sur le même chemin (vers `server_eth0`), changé de chemin et envoyé les retransmissions suivantes sur un autre chemin, vers `server_eth1`.

Malheureusement, comme `server_eth0` et `client_eth0` sont sur le même réseau, la coupure du câble de `server_eth0` n'a pas suffit à changer les routes entre `server_eth0`

et `client_eth0`, si bien qu'aucune retransmission n'aie pu être acquittée par le serveur. Ce problème apparaît si la panne se manifeste dans le même sous-reseau que l'interface `server_eth0`. Dans d'autres cas le problème doit être résolu par la gestion dynamique du routage sur les routeurs du réseau.

La figure 7.3 montre les résultats obtenus pendant cette expérience. Au début on voit la transmission sur le premier chemin (`client_eth0`, `server_eth0`) et ensuite on voit la panne qui bloque la transmission jusqu'à la fin de l'expérience.

Chapitre 8

Extensions du multi-accès dans SCTP

Le protocole SCTP propose des bases solides pour le multi-accès dans la couche du transport. Néanmoins, à notre avis, la solution n'est pas complète et nécessite des extensions. Dans ce chapitre, nous proposons deux prototypes développant la fonctionnalité du multi-accès dans SCTP. Le premier aborde le problème présenté dans le chapitre 7.3 et l'autre permet d'établir les transmissions concurrentes sur plusieurs chemins entre deux machines.

Nous présentons dans ce chapitre les deux prototypes, ainsi que des expériences et leurs résultats.

8.1 Utilisation de toutes les adresses accessibles

Cette proposition est destinée à éviter la situation défavorable présentée dans le chapitre 7.3 en modifiant le protocole SCTP. Notre prototype permet de bénéficier de toutes les adresses IP sur les deux côtés d'une association. De cette façon, en cas de panne sur une adresse (interface ou liens réseau), la transmission peut être rétablie sur une autre. Dans le SCTP classique cela est possible uniquement sous certaines conditions. Notre prototype offre cette possibilité pour chaque machine possédant plusieurs adresses. Une solution similaire est proposée par Kubo et al. [36]. Mais elle est incomplète, car elle n'adresse pas le problème de routage dans la couche réseau. Notre solution est plus complète.

8.1.1 Idée du prototype

Notre proposition consiste à créer tous les chemins possibles entre le serveur et le client. Pour n adresses du serveur et m adresses du client, cela donne $n*m$ chemins utilisables pour une association. La création de transports sur la machine du client se déroule de la manière suivante :

```
for each dest_adr in server_addresses {
     for each src_adr in client_addresses {

          new_transport = create_transport()

          new_transport.destination_address  =  dest_adr
          new_transport.source_address       =  src_adr

     }
}
```

Dans la configuration sur la figure 8.1 on voit deux machines équipées de deux cartes réseau chacune. Dans ce cas, le protocole SCTP non modifié sur la machine client ne crée que deux chemins, un chemin pour chaque adresse IP de l'autre extrémité.

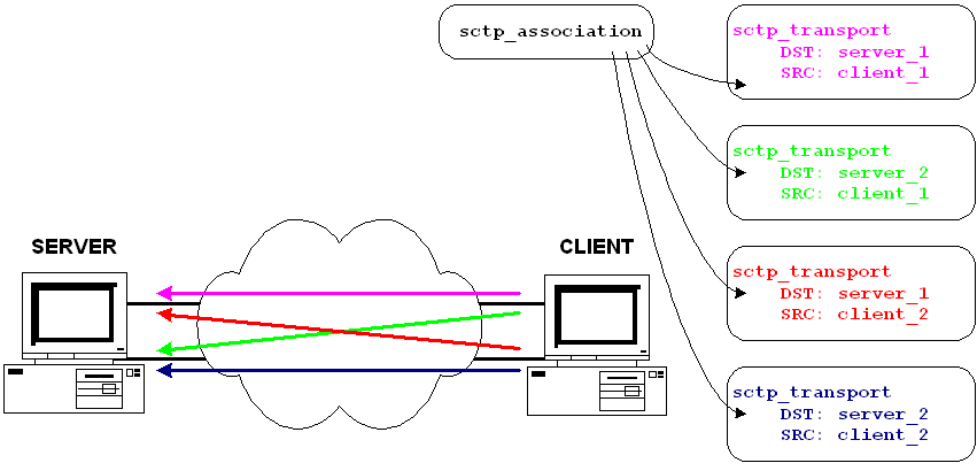

FIG. 8.1 – Les quatre chemins possibles entre le client et le serveur

La figure montre les résultats après notre modification. On y voit un chemin pour chaque paire d'adresses accessibles. Cela nous donne finalement quatre chemins différents à utiliser pendant une association. Toutes les adresses du client et celles du serveur sont utilisées.

Nous avons implementé notre modification sur le noyau Linux en version 2.4.23 avec lkSCTP en version 0.6.9.

8.1.2 Gestion du routage

Le cas décrit dans le chapitre précédent présente une difficulté au niveau de routage. Dans notre situation, nous avons deux chemins avec la même adresse du destinataire, mais avec des adresses source différentes (par exemple le premier et le troisième chemin sur la figure 8.1). Pourtant, selon le routage, nous avons toujours une interface de sortie pour une adresse IP du destinataire donnée. De cette façon, les paquets issus de deux chemins ayant la même adresse du destinataire sont relayés vers la même interface, bien qu'ils aient les adresses source différentes. Cela ne permet pas de bénéficier de la deuxième carte réseau et de plus les paquets peuvent être bloqués dans le réseau d'opérateur par le filtrage des paquets avec des adresses source étrangères (*ingress filtering* [9]).

On peut aussi imaginer une situation où le routage donne la même interface de sortie pour toutes les adresses du destinataire. Dans cette situation également, la deuxième carte réseau ne présente aucun intérêt pour une association SCTP.

Pour résoudre ce problème, nous avons dû appliquer un routage par l'adresse source, qui permet d'envoyer les paquets selon leur adresse source. Le logiciel Netfilter [37] est un ensemble de modules qui permettent de faire le routage spécifique dans le noyau du Linux. Le module ROUTE [38] permet de choisir l'interface de sortie du paquet selon les critères précisés.

target	prot	opt	source	destination	
ROUTE	132	--	client_1	0.0.0.0/0	ROUTE oif:eth0
ROUTE	132	--	client_2	0.0.0.0/0	ROUTE oif:eth1

FIG. 8.2 – La table de routage spécifique

Dans la figure 8.2 on voit la table du routage que nous avons configuré. Selon cette table, les paquets issus du protocole SCTP sont envoyés vers l'interface `eth_0` s'ils portent l'adresse source `client_1`. S'ils ont comme l'adresse source `client_2`, ils sont envoyés vers l'interface `eth_1`. L'adresse de destination n'est pas prise en compte pour la décision de routage dans ce cas.

8.1.3 Expériences avec le prototype

Pour évaluer notre proposition nous avons mis en place une configuration présentée sur la figure 8.3. Le client a été équipé de deux cartes réseau, donc il avait deux adresses IP. Le serveur a été équipé d'une seule carte réseau. La table du routage du client a été configurée comme sur la figure 8.2. étant donné que le serveur ne possédait qu'une seule carte, sa table du routage n'a pas nécessité de changement.

Profitant de notre modification, le client a créé deux chemins suivants :

1. (DST : `serveur_1`, SRC : `client_1`)
2. (DST : `serveur_1`, SRC : `client_2`)

Pendant l'expérience, nous avons commencé la transmission sur le premier chemin. Après quelques secondes nous avons simulé un problème sur le réseau en débranchant le câble du commutateur. Cela a provoqué les retransmissions du côté client. Avec la sixième retransmission, le client a rétabli la transmission sur le deuxième chemin en utilisant l'adresse source `client_2`.

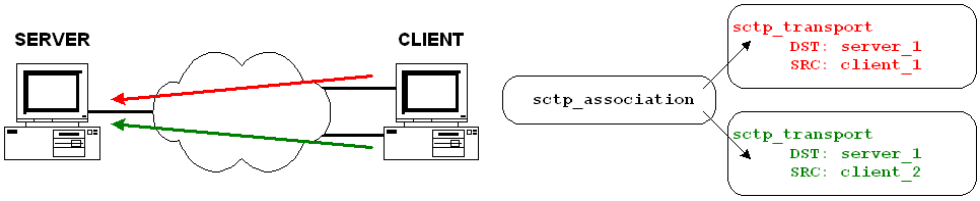

FIG. 8.3 – Configuration de l'expérience

Sur la figure 8.4, on peut voir le niveau du débit pendant cette expérience. En première phase, nous voyons le débit sur le premier chemin. Après la coupure du réseau, on voit que la transmission s'arrête. Au bout de 63 secondes (comme expliqué dans le chapitre 7.2) la transmission est rétablie sur le deuxième chemin (src : `client_2`, dst : `serveur_2`), ce qu'on peut voir par le changement du débit.

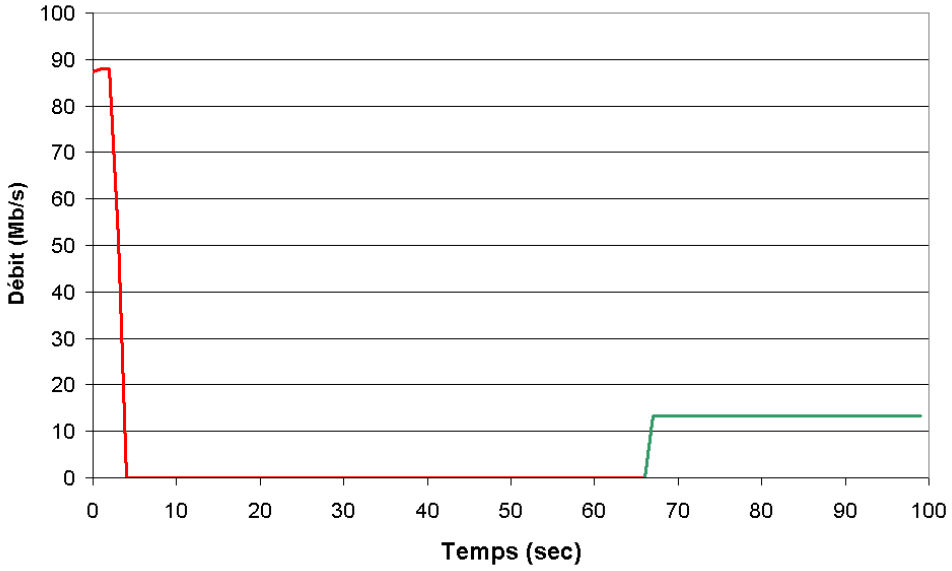

FIG. 8.4 – Le changement de chemin pendant l'association

8.2 Transmissions concurrentes

Dans ce chapitre nous présentons un prototype qui permet d'utiliser plusieurs chemins pour envoyer les données en même temps. Cela permet d'avoir des transmissions concurrentes sur plusieurs chemins dans une association SCTP.

8.2.1 Prototype

SCTP permet d'avoir plusieurs chemins dans une association. Cela peut signifier qu'on peut avoir plusieurs chemins entre le client et le serveur qui sont complètement indépendants. Malgré cela, le protocole SCTP utilise seulement un chemin pour la transmission des données. Les autres chemins sont gardés pour le cas d'une panne.

Nous proposons d'utiliser plusieurs chemins pour transmettre des données dans une association. Cela peut permettre d'obtenir un débit plus élevé en utilisant le débit cumulé sur plusieurs chemins liant les machines communicantes.

Le prototype proposé est en fait une extension du prototype décrit dans le chapitre 8.1. De la même façon nous créons tous les chemins possibles entre le client et le serveur et nous utilisons quelques uns d'entre eux pour transmettre des données.

Dans notre prototype, la socket envoie des messages de façon classique, il envoie un seul message à la fois. Une fois le message envoyé par la socket, il est traité par le protocole SCTP, et c'est là où le chemin approprié est choisi. Après le choix du chemin, le protocole copie les adresses de la destination et de la source dans le paquet et l'envoie. De cette façon, on peut choisir individuellement un chemin en réseau pour chaque message envoyé.

Le choix de chemin n'est fait que pour les portions de données. Tous les messages de contrôle sont traités de manière classique par SCTP.

Comme précédemment, notre prototype a été implementé dans le noyau Linux en version 2.4.23 avec lkSCTP en version 0.6.9.

Ce prototype exige aussi une gestion du routage spécifique. Pour cette raison, nous avons utilisé la même méthode et la même table du routage que celles présentées dans le chapitre 8.1.2.

8.2.2 Expérience avec le prototype

Pour évaluer notre prototype, nous avons utilisé la configuration présentée sur la figure 8.1. Les deux machines ont été connectées à deux réseaux locaux différents. Cela nous a donné des conditions différentes de transmission de données en passant par des

réseaux différents. Selon notre modification décrite en chapitre 8.1, quatre chemins ont été créés sur deux machines.

Pour cette expérience, nous avons utilisé une association SCTP avec deux flots de données. Nous avons configuré notre prototype pour qu'il envoie le premier flot sur le premier chemin (src : `client_1`, dst : `serveur_1`) et le deuxième flot sur le dernier chemin (src : `client_2`, dst : `serveur_2`). De cette façon, l'association utilisait deux chemins réseau complètement indépendants pour envoyer les données.

Les deux flots de données ont été composés de paquets de même taille. Pendant l'expérience, l'association envoyait la même quantité de paquets (donc d'octets) par chaque chemin.

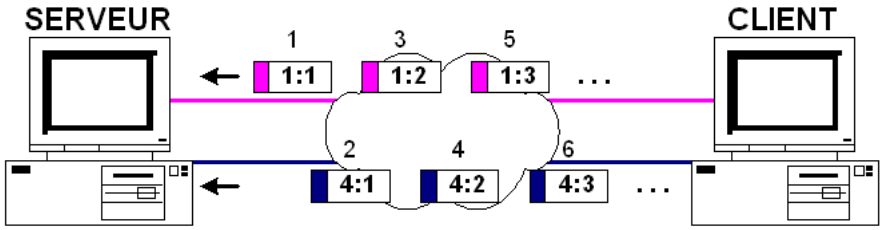

FIG. 8.5 – La transmission de données pendant l'expérience

La figure 8.5 présente la transmission de données pendant l'expérience. On voit deux flots de données qui sont envoyés sur les deux chemins différents. Les rectangles symbolisent les paquets. Les numéros dans les rectangles indiquent le numéro du flot et le numéro du paquet dans le flot. Les numéros sur des paquets indiquent l'ordre de l'envoi des paquets par l'association. On peut observer que les paquets sont envoyés tour à tour sur les deux chemins.

	no.1	no.4	mix (no.1 + no.4)
Mb/s	12,79	76,77	21,78
s/Mb	0,078	0,013	0,045

TAB. 8.1 – Les débits sur plusieurs chemins

Le tableau 8.1 montre les résultats. La deuxième colonne contient le débit obtenu par l'utilisation d'un seul chemin no.1 (src : `client_1`, dst : `serveur_1`). La troisième colonne contient la bande passante obtenue par l'utilisation d'un seul chemin no.4 (src : `client_2`, dst : `serveur_2`). La dernière colonne montre le débit obtenu suite à l'utilisation des deux deux chemins (no.1 et no.4). Comme les paquets n'ont pas été envoyés parallèlement mais tour à tour, le débit observé n'égale pas la somme de deux débits individuels.

Notre prototype n'est pas optimisé pour de transmissions concurrentes. Par suite, il ne permet pas de profiter d'un débit plus élevé en utilisant plusieurs chemins. Pour

obtenir un débit plus élevé, il faut étudier d'autres problèmes qui se posent dans le cas des transmissions concurrentes dans une association. L'un d'eux est l'envoi et la réception parallèles de paquets par une assocation. Un autre est le problème de l'ordre des paquets. Si les paquets empruntent des chemins différents pour arriver à la destination, l'ordre d'arrivée peut être différent de l'ordre d'envoi. Dans certains cas le coût de réordonner des paquets pour une connexion peut être grand et annuler les gains. Il faudrait donc étudier des mécanismes de gestion de paquets pour remédier à ce problème - par example s'il y a plusieurs flots on peut répartir les flots sur les différents chemins.

Chapitre 9

Conclusion sur le transport concurrent et multichemin

Dans cette partie de la thèse, nous nous sommes occupés du problème de multi-accès qui permet de profiter, dans une connexion de niveau transport, de plusieurs adresses au niveau réseau. Après une étude des solutions existantes, nous avons constaté que le nouveau protocole SCTP propose une approche intéressante du multi-accès dans la couche transport. Afin de pouvoir mesurer les performances du protocole SCTP, nous avons élaboré un outil approprié. Les mesures de performance ont été effectuées dans deux environnements de très haut débit, sur le réseau VTHD++ et sur un réseau local de 1 Gb/s. En même temps, nous avons constaté que SCTP présente certains inconvénients dans son fonctionnement. Nous avons donc proposé deux modifications qui peuvent être appliquées à SCTP pour l'améliorer. Deux prototypes ont été élaborés et des expériences appropriées ont montré leur bon fonctionnement.

Deuxième partie

Transport de distribution de contenu

Chapitre 10

Introduction

Les avancées récentes dans le domaine de réseaux optiques nous ouvrent la voie vers un débit de plus en plus élevé. Des utilisateurs domestiques peuvent aujourd'hui profiter des réseaux d'accès xDSL dont le débit est mesuré en dizaines de Mb/s.

Cette abondance de ressources permet de changer les paradigmes sur lesquels sont bâtis les réseaux actuels. Les protocoles de pile réseau TCP/IP, conçus dans une autre réalité technologique, ont été suffisants dans leur période de naissance, autour des années 70. Pendant toutes ces années et jusqu'à aujourd'hui, le modèle TCP/IP a vieilli parce que les caractéristiques des réseaux ont évolué. Maintenant, il doit être retravaillé pour pouvoir proposer de nouvelles solutions conformes à la réalité moderne.

Nous cherchons ici une autre définition de la notion de connexion. Actuellement, le contenu est explicitement demandé par un utilisateur et lui est personnellement acheminé par l'émetteur par une transmission point-point établie à la demande. C'est l'utilisateur qui demande explicitement à l'émetteur d'établir une connexion pour lui transmettre les données demandées. Dans ce travail, nous voulons proposer une autre définition de la connexion qui se fonde sur la notion de transmission multipoint et sur le mode "forcé". Par transmission multipoint, nous comprenons un mode où une seule transmission est envoyée à plusieurs destinataires et par mode "forcé", nous comprenons des transmissions à l'initiative de l'émetteur et non du destinataire des données. Grâce au très haut débit, l'émetteur peut envoyer des données vers des utilisateurs potentiels sans se demander s'ils veulent vraiment les recevoir.

Dans cette logique, la deuxième partie de cette thèse aborde le problème de la distribution du contenu dans des réseaux à très haut débit. Nous commençons cette partie par la description précise de la problèmatique qui nous intéresse. Après un état de l'art nous formulons une proposition qui répond au problème présenté.

Chapitre 11

Problématique

Pour étudier le problème de la distribution de contenu nous devons préciser certains termes :

> **Contenu** - toute forme de données électroniques qui sont identiques pour tous les utilisateurs qui les reçoivent. Ces données ne sont pas personnalisées et chaque utilisateur reçoit exactement les mêmes données. Au moment de la création du contenu, le créateur peut ne pas connaître les identités des utilisateurs qui les recoivent.

> **Distribution** - l'ensemble des activités qui sont nécessaires pour déplacer des données du contenu depuis sa source jusqu'à toutes les destinations (une ou plusieurs) qui désirent le recevoir.

Selon les définitions ci-dessus, la distribution de contenu s'appuie sur une multitude de destinations auxquelles les données doivent être livrées. Cela peut être assuré de deux façons. Le première consiste en plusieurs connexions individuelles établies entre la source de données et chaque destinataire. Cette méthode nécessite des ressources abondantes aussi bien chez l'émetteur que dans le réseau. L'autre approche se fonde sur des transmissions multipoint. Une transmission multipoint est une transmission qui possède plusieurs émetteurs et/ou plusieurs destinataires. Dans notre recherche, nous nous limitons au cas avec un émetteur et plusieurs destinataires. C'est le cas où l'émetteur sert tous ses utilisateurs en effectuant une seule transmission de données. Au premier abord, les transmissions multipoint offrent des avantages significatifs. Elles permettent d'économiser de ressources en effectuant une seule transmission au lieu d'en faire plusieurs. Mais en même temps elles posent des problèmes qui doivent être résolus pour assurer la simplicité et l'efficacité d'une solution proposée.

Dans cette thèse nous visons à proposer une solution qui répond aux problèmes suivants :

La transmission multipoint à plusieurs destinations : une seule transmission de l'émetteur doit assurer la livraison des données à tous les utilisateurs désirant les recevoir. Pour cela, les données doivent être correctement acheminées à tous les utilisateurs en passant par des réseaux divers et indépendants.

La localisation d'utilisateurs inconnue : l'émetteur qui envoie des données peut ne pas savoir où sont localisés les utilisateurs. De plus, d'habitude, il n'a pas la possibilité de connaître ni leur identité, ni leur localisation. L'émission de données ne doit pas être destinée à une (ou plusieurs) localisations dans le réseau mais aux utilisateurs qui désirent recevoir des données concernées.

Exploiter des réseaux à très haut débit : dans notre contexte nous bénéficions des réseaux à très haut débit. Nous considérons que le débit n'est plus une ressource rare dans le réseau. Cela nous amène à modifier le concept classique de connexion où un utilisateur demande explicitement son établissement. Nous voulons pousser le contenu vers les utilisateurs et le leur offrir dans un environnement local. Grâce au très haut débit nous pensons que le contenu peut être poussé à proximité des utilisateurs sans avoir une connaissance préalable de leurs besoins.

L'adressage et le routage orientés contenu : le contenu n'est pas envoyé à une localisation dans le réseau mais aux utilisateurs qui désirent le recevoir. Pour cela, les paquets portant des données doivent être identifiés par le contenu transporté. Le routage utilisé doit prendre en compte les identificateurs de contenu porté dans les paquets et pas les adresses topologiques. Pour résumer, le protocole doit assurer la livraison des données selon un adressage orienté contenu et non selon adressage topologique des utilisateurs.

Dans cette partie de la thèse nous essayons de répondre à ces questions. Dans ce but, nous analysons d'abord les solutions existantes et ensuite nous proposons la nôtre.

Chapitre 12

Le multicast IP

Pour économiser des ressources réseau, l'architecture du multicast propose la livraison de données à plusieurs utilisateurs en même temps. Dans ce chapitre nous présentons une vue générale sur la thématique du multicast et sur cette architecture. Nous montrons les principes du multicast/diffusion. Ensuite les techniques et les protocoles de création de graphes de distribution multicast sont discutés. Finalement, nous présentons des protocoles associés : la gestion de groupes multicast et de session multicast.

12.1 Les principes du multicast IP

Il existe une solution pour la communication multipoint : le multicast IP tel quel défini par S.E. Deering [39] [40] [41] [42]. D'autres documents, plus anciens [43] [44], présentent certaines idées sur la communication multipoint, notamment sur le multicast et la diffusion.

Nous n'allons pas faire ici un resumé de ces travaux, mais nous allons énoncer les notions les plus importantes de l'idée de multicast. Le multicast introduit la notion de groupe de machines destinatrices d'un paquet multicast défini par une adresse commune (*group address* ou *multicast address*). Pour envoyer un paquet à tous les membres du groupe, la source utilise l'adresse du groupe, au lieu de l'adresse individuelle (*unicast*), comme adresse destination du paquet. Cette approche présente une définition logique, plutôt que topologique, d'un groupe. Pour atteindre certaines machines, il faut connaître le groupe dont elles sont les membres, plutôt que de savoir où elles se trouvent. Parmi les propriétés des groupes multicast, les plus importantes sont les suivantes :

– **L'émetteur peut ne pas être membre du groupe** : n'importe quelle machine peut envoyer les données aux membres d'un groupe. Il faut joindre un groupe uniquement pour recevoir des données. Les groupes de ce type s'appellent *ouverts*.

– **Le nombre de membres récepteurs n'est pas limité** : par défaut, un groupe peut être constitué par *zero ou plus* de machines.

– **Il n'y a pas de contraintes topologiques** : la localisation de machines n'est pas limitée, et les machines peuvent se trouver dans des réseaux différents.

– **La participation dans les groupes est dynamique** : chaque machine est libre de joindre ou de quitter un groupe à n'importe quel moment. Une machine n'est pas obligée de communiquer avec les autres membres du groupe concerné pour le faire.

– **Les groupes sont permanents ou temporaires** : les groupes permanents sont définis par des adresses bien-connues (*well known addresses).* Ils peuvent être utilisés comme des identificateurs logiques pour des services populaires (par exemple le service de résolution de noms) quand la localisation exacte du service n'est pas connue de l'utilisateur. Les groupes temporaires existent pendant une durée précise, par exemple pour la durée d'exécution d'une application.

12.2 Techniques de construction de l'arbre de distribution

La transmission de données dans des protocoles du multicast est basée sur l'idée *d'arbre de distribution*, un graphe qui couvre tous les nœuds du réseau désirant participer au trafic multicast. Il contient la source du trafic dans sa racine et tous les récepteurs du trafic dans ses feuilles. Tous les autres nœuds du graphe sont des routeurs intermédiaires. Comme ce graphe doit former un arbre, il ne peut pas comprendre de boucles.

Dans la section suivante, nous présentons un algorithme pour éviter les boucles, puis deux techniques pour créer l'arbre de distribution. Tous les algorithmes existants sont basés sur une de ces deux techniques.

12.2.1 Éviter les boucles

Afin d'éviter l'inondation du réseau par le trafic multicast, le graphe de distribution ne peut pas contenir de boucles, il doit former un arbre. L'algorithme RPF (*Reverse Path Forwarding*) a été proposé dans ce but [44]. Cet algorithme analyse et sélectionne des paquets avant de les transférer pour élaguer, éventuellement, les branches pouvant former une boucle dans le graphe. Quand un paquet arrive à un routeur, le routeur vérifie le chemin le plus court vers la source du paquet en utilisant le protocole du routage *unicast.* Selon ce protocole, le routeur détermine l'interface de sortie vers la source du paquet qui est arrivé. Si c'est une interface d'arrivée de paquet, celui-ci est transféré à toutes les autres

interfaces. Sinon, le paquet est rejeté. Dans cet algorithme, pour chaque adresse source, il y a seulement une interface qui mène vers lui. De cette façon, pour chaque adresse source il y a une seule interface sur laquelle les paquets sont acceptés. Cela évite la formation le boucles dans le graphe créé.

Dans le chapitre 13.3.3, nous présentons un algorithme du type similaire que RPF. Notre proposition est basée surtout sur le chemin le plus rapide, et non sur celui le plus court.

12.2.2 Inondation-et-élagage

Les protocoles basés sur la technique *inondation-et-élagage*[1] commencent la création de l'arbre de distribution à partir de la source. Le premier paquet du flot multicast est envoyé par la source en aval vers toutes les extrémités du réseau. Dès que le paquet arrive à un routeur, ce dernier a deux possibilités. Il peut accepter le trafic multicast et transférer le paquet aux interfaces appropriées, ou, au contraire, il peut refuser ce trafic et envoyer un message *prune* au routeur en amont qui a lui envoyé ce paquet. Dans le deuxième cas, le trafic multicast concerné n'est plus envoyé à ce routeur. De cette façon, le trafic est proposé à tous les routeurs du réseau. Les routeurs ne voulant pas le recevoir le refusent et ne font pas partie du groupe multicast. Une fois l'arbre de distribution créé, tous les paquets de ce flot multicast suivent cet arbre. Pour chaque couple {*source, groupe de la destination*}, un arbre est créé.

Pour former un arbre de distribution, le protocole doit éviter de produire des boucles. Dans ce but, les protocoles de ce type utilisent la technique RPF [44].

Les protocoles qui utilisent cette méthode sont aussi appelés *"protocoles de jointure implicite"*. Ils considèrent, dès le départ, que tous les routeurs dans le réseau souhaitent participer au trafic multicast. Si un routeur ne souhaite pas recevoir le trafic, il doit s'élaguer explicitement de l'arbre de distribution.

12.2.3 L'arbre partagé

La technique alternative, par rapport à celle presentée auparavant, utilise les *arbres partagés*. Les protocoles qui s'en servent sont également appelés *"protocoles de jointure explicite"* du fait que, pour envoyer et recevoir le trafic multicast, une machine doit explicitement s'abonner à un groupe multicast. Aucune machine, ni aucun routeur n'est membre d'aucun groupe par défaut.

Cette technique est basée sur un *point central* auquel tous les membres se connectent. C'est le point central qui crée, gère et maintient l'arbre de distribution. Il y a un seul point

[1] Nous utilisons ce terme pour le terme anglais *broadcast and prune*.

central pour chaque groupe multicast. La source envoie les données au point central, qui s'occupe de la livraison de données à tous les membres du groupe. Le trafic d'un groupe multicast est diffusé seulement aux routeurs qui ont explicitement demandé de le recevoir. Dans ce type de distribution, on n'a pas besoin de définir la source, car chaque machine peut également fonctionner comme une source. Elle doit tout simplement envoyer les données au point central pour qu'elles soient diffusées.

12.3 Les protocoles de routage multicast

Il existe plusieurs protocoles pour créer et gérer des arbres de distribution. Nous pouvons distinguer deux catégories de protocoles qui sont basées sur les deux techniques presentées dans le chapitre 12.2 : les protocoles de type *dense* qui utilisent la technique *d'inondation-et-élagage* et les protocoles de type *dispersé* qui utilisent la technique de *l'arbre partagé*. Dans les sections suivantes, nous présentons les deux catégories et les protocoles qui les représentent.

12.3.1 Protocoles du type *dense*

Les protocoles de cette famille supposent que :

- la plupart des machines dans le réseau veulent recevoir le trafic,
- les machines sont regroupées d'une manière dense dans le réseau (relativement un grand nombre de machines sur chaque réseau local),
- le débit est facilement disponible ou n'est pas cher.

Pour accomplir les exigences citées ci-dessus, les protocoles de cette famille appliquent la technique *d'inondation-et-élagage* pour construire l'arbre de distribution. Au début, on suppose que tous les routeurs et toutes les machines participent à la distribution, donc on leur diffuse le trafic. Dans le processus de diffusion, les machines qui ne veulent pas participer à la distribution refusent le trafic et ne le reçoivent plus. Si un routeur (ou une machine) souhaite joindre un arbre de distribution, il (ou elle) doit explicitement le demander en contactant le routeur qui est déjà connecté à l'arbre desiré.

Dans les paragraphes suivants, nous présentons trois protocoles de cette famille : DVMRP (*Distance Vector Multicast Routing Protocol*), MOSPF (*Multicast Open Shortest Path First*) et PIM-DM (*Protocol-Independent Multicast - Dense Mode*).

Distance Vector Multicast Routing Protocol

DVMRP est le plus ancien protocole de multicast [45]. Il construit l'arbre de distribution à la demande, en utilisant la technique *d'inondation-et-élagage* de manière presentée

ci-dessus. La source de données envoie le premier paquet en aval vers tous les routeurs multicast adjacents. Chaque routeur transmet sélectivement le paquet aux routeurs en aval. Pour transmettre le paquet de manière sélective, le routeur vérifie le chemin le plus court vers la source, en utilisant son propre protocole du routage unicast. Si l'interface par laquelle le paquet est arrivé est celle qui mène vers la source (selon le protocole du routage unicast), le paquet est transmis sur toutes les autres interfaces. Dans l'autre cas, le paquet est silencieusement supprimé. Le paquet est transféré vers tous les routeurs adjacents, sauf vers le routeur par lequel le paquet est arrivé.

Un routeur multicast peut quitter un groupe multicast en envoyant un message *prune* vers son routeur en amont. Cela peut arriver dans les deux cas suivants :

- quand le routeur ne possède plus de stations connectées directement à lui qui veulent recevoir le trafic de ce groupe multicast,
- quand tous les routeurs adjacents en aval lui ont envoyé un message *prune*.

Pour être encore plus sélectif et pour éviter la duplication de paquets, DVMRP utilise la technique suivante : chaque routeur maintient les informations concernant tous ses routeurs adjacents. Ces informations lui permettent de déterminer si un routeur adjacent est capable de le reconnaître comme le routeur qui se trouve sur le chemin le plus court allant à la source multicast. Si cela n'est pas le cas, le routeur supprime ce routeur de la liste et ne lui transmet pas de trafic multicast.

DVMRP ne permet pas passer à grande échelle en raison de la quantité de données échangée entre les routeurs. Les tables de routage souffrent du même problème : elles croissent rapidement parce qu'elles comprennent des informations concernant l'état de routeurs voisins.

Multicast Open Shortest Path First

Le protocole MOSPF [46] a été proposé comme une extension multicast pour le protocole OSPF version 2 [47].

Dans MOSPF, chaque routeur maintient l'état topologique du réseau entier. Tous les routeurs collectent également les informations concernant la participation dans les groupes multicast. Dans ce but, ils utilisent le protocole IGMP (*Internet Group Management Protocol*) (présenté dans la section 12.5). Toutes ces informations sont échangées périodiquement entre les routeurs pour être sûr qu'ils possèdent toutes les informations courantes sur la topologie du réseau et sur la participation dans tous les groupes multicast. Possédant les informations nécessaires, chaque routeur calcule le même arbre de distribution couvrant le réseau depuis la source (dans la racine) jusqu'aux clients (dans les feuilles). L'arbre est calculé en utilisant l'algorithme de Dijkstra [48] et puis il est utilisé pour livrer le trafic de la source à tous les clients.

A cause des échanges périodiques de grandes quantités de données, ce protocole ne passe pas à grande échelle. Il nécessite aussi beaucoup de ressources, notamment la mémoire des routeurs pour maintenir toutes les informations nécessaires.

Protocol-Independent Multicast - Dense Mode

Le protocole PIM (*Protocol-Independent Multicast*) [49] existe en deux versions : la version dense [50] et la version dispersée [51]. Nous présentons d'abord la version dense, la version dispersée est présentée dans la section 12.3.2.

L'idée de PIM-DM est similaire à celle de DVMRP. Mais PIM-DM ne dépend pas du protocole unicast utilisé dans la couche réseau et il peut utiliser n'importe quel protocole de routage unicast implémenté sur un routeur.

L'arbre de distribution est construit par le premier paquet du flot multicast grâce au processus RPF. Le routeur vérifie le chemin le plus court allant à la source du paquet, pour chaque paquet qui arrive à un routeur. Si la vérification est positive (c'est-à-dire que l'interface par laquelle le paquet est venu est celle qui mène à la source du paquet selon le protocole unicast), le paquet est transmis. Si la vérification s'avère négative, le paquet est silencieusement supprimé. La transmission du paquet est effectué vers tous les routeurs en aval, jusqu'à ce que le paquet arrive à un utilisateur ou à un routeur refusant le trafic de ce groupe multicast. Dans ce dernier cas, le paquet n'est pas transmis et le routeur qui a refusé de recevoir le trafic est élagué de l'arbre de distribution.

En comparaison avec DVMRP, PIM-DM provoque un trafic additionel dû à la duplication de certains paquets. En même temps, PIM-DM présente aussi d'autres défauts. Le premier consiste dans le fait d'inonder périodiquement le réseau. Cela est utilisé pour rafraîchir les arbres existants et pour permettre aux routeurs n'appartenant pas aux groupes de les rejoindre. Comme l'inondation est périodique, cela provoque un trafic qui consomme beaucoup de ressources réseau. L'autre défaut consiste dans le fait que tous les routeurs PIM-DM gardent les informations sur les groupes auxquels ils ne participent pas. Cela permet à un routeur de rejoindre facilement et rapidement un groupe, sans attendre la prochaine inondation d'informations concernant ce groupe. Malheureusement, cela nécessite aussi des ressources sur les routeurs, notamment de la mémoire.

12.3.2 Protocoles du type *dispersé*

Dans cette catégorie, les protocoles supposent que :

- les récepteurs sont très dispersés dans le réseau,
- personne ne veut recevoir du trafic par défaut,
- le débit est fortement limité ou très cher.

Les protocoles dans cette catégorie sont basés sur la technique *"d'arbre partagé"* et sur l'idée de *"point central"*. Un groupe multicast n'a pas de membres par défaut. Une machine qui veut recevoir du trafic, doit explicitement joindre le groupe en le demandant à un routeur qui reçoit déjà le trafic multicast. La source envoie le trafic au point central, et c'est le point central qui le diffuse à tous les membres du groupe. Pour la diffusion du trafic, on se sert de *l'arbre partagé*.

Les paragraphes qui suivent présentent les protocoles suivents : CBT (*Core Based Trees*) [52] [53] et PIM-SM (*Protocol-Independent Multicast - Sparse Mode*) [51].

Core Based Trees

Dans ce protocole, le point central du groupe multicast est un routeur "cœur" (*core router*) qui crée l'arbre de distribution. Les autres routeurs joignent le groupe en envoyant un message *join* soit au routeur cœur soit à un routeur qui fait déja partie du groupe.

Il y a un seul arbre de distribution pour tout le trafic dans un groupe multicast. Chaque source peut utiliser le même arbre pour diffuser son trafic. Cela permet à chaque membre de diffuser son trafic à tous les autres membres du groupe. L'utilisation d'un seul arbre facilite sa création et son maintien.

Le protocole CBT a été conçu pour les applications qui nécessitent un trafic plusieurs--vers-plusieurs (*many-to-many*), comme par exemple les jeux du réseau ou les conférences multi-utilisateurs.

Protocol-Independent Multicast - Sparse Mode

Tournons nous maintenant vers une autre version du protocole PIM, celle de type dispersée, PIM-SM. Par rapport à la version dense, elle est plus sélective et restreinte dans la distribution du trafic multicast que la version dense.

Le cœur de l'arbre de distribution est localisé dans un point de rendez-vous (RP - *Rendezvous Point*), dont l'adresse est publiquement connue par tous les membres potentiels d'un groupe multicast. L'arbre de distribution est créé à partir du RP grâce à RPF [44] (pour éviter de créer des boucles). Chaque nœud ou routeur qui veut recevoir le trafic multicast doit tout d'abord contacter le RP pour connaître la localisation de la source. Un fois qu'il connaît la source, le récepteur peut choisir la méthode avec laquelle le trafic multicast lui sera livré. Il y a deux possibilités :

L'arbre partagé par groupe : l'arbre créé par le RP est partagé par tous les membres du groupe. Sa construction est simple et économise le débit. Pourtant, dans l'arbre partagé, le trafic est concentré autour du RP ce qui peut dégrader les performances si l'on rencontre un volume important de trafic multicast. Il est possible

d'avoir plusieurs RP pour éliminer les problèmes de congestion excessive dans un point du réseau.

L'arbre de chemin le plus court : au début de la diffusion du trafic, l'arbe partagé est créé et utilisé. Une fois l'arbre partagé construit, le récepteur peut demander la création d'un arbre où le chemin vers la source est le plus court. Pour cela, il envoie un message *join* à la source. Après la création du nouvel arbre, toutes les branches superflues sont élaguées et le trafic commence à être diffusé par le nouvel arbre optimisé.

12.4 Allocation d'adresses de groupes

Dans l'approche basique du multicast, un groupe multicast est défini par une adresse spécifique. Cette approche est simple à implémenter et à gérer dans les domaines sous une authorité unique (*single administrative domain*). Dans des réseaux à échelle globale, cela pose des problèmes d'attribution d'adresses. Une fois une adresse allouée à un groupe, elle ne peut pas être utilisée par un autre groupe. L'attribution d'adresses doit être unique au niveau global pour éviter des colisions. Dans ce chapitre nous présentons des exemples de solutions existantes pour résoudre ce problème. Nous ne visons pas ici à les présenter toutes, mais simplement à décrire les deux approches générales à ce problème d'allocation d'adresses.

Plusieurs propositions existent pour attribuer des adresses multicast au niveau global. On peut citer, par exemple, l'architecture MALLOC (*Multicast Address Allocation Architecture*) [54]. C'est une architecture modulaire qui comprend trois couches : (i) un mécanisme de host-serveur responsable de l'allocation et de la gestion d'adresses ; (ii) un mécanisme d'intra-domaine pour assurer que les adresses allouées ne sont pas dupliquées ; (iii) un mécanisme d'inter-domaine pour permettre d'annoncer des adresses allouées aux autres machines dans le réseau (surtout aux routeurs). Dans cette logique, pour la couche (i), le protocole MADCAP (*Multicast Address Dynamic Client Allocation Protocol*) est proposé [55]. Il permet aux clients de demander une adresse multicast à un serveur d'allocation qui possède une certaine plage d'adresses. Pour la couche (iii) on utilise le protocole MASC (*Multicast Address-Set Claim*) [56]. Dans ce protocole, une machine (un routeur) annonce une plage d'adresses qui seront réservées à son domaine. La plage est annoncée dans le protocole du routage inter-domaine, par exemple BGP4+ [57]. Les clients qui demandent une adresse multicast reçoivent une adresse de cette plage annoncée.

Le protocole *Administratively Scoped IP Multicast* [58] est une autre proposition qui se concentre sur la limitation de l'étendue de l'utilisation d'adresses multicast allouées. Les paquets multicast ne peuvent pas dépasser les limites administratives d'un domaine et des adresses multicast sont allouées localement dans un domain administratif. De cette façon, le mécanisme assure que des adresses dans un domaine donné sont uniques. Cela limite

fortement l'utilisation du multicast et est en contradiction avec les principes de base du multicast global car il ne devrait pas être limité topologiquement.

Les deux approches présentent une tendance générale dans les techniques d'allocation d'adresses. Soit une architecture complexe s'avère nécessaire, soit l'application de la solution est topologiquement limitée.

12.4.1 Multicast spécifique à la source

Pour attaquer le problème de l'attribution d'adresses, le protocole *EXPlicitly REquested Single-Source (EXPRESS)* [59] a changé la définition du groupe multicast. La notion de groupe multicast a été remplacée par un *canal multicast*. Celui-ci est un service défini par un couple (*S, E*) où *S* est la source qui émet des données et *E* est une adresse de destination du canal. Il n'y a que la source *S* qui peut envoyer des données aux machines définies par *E*. Pour qu'elle puisse envoyer des données aux membres du groupe désigné par *E*, la source *S'* doit former un autre canal défini par (*S', E*). Afin de recevoir des données, tous les récepteurs sont obligés de s'abonner au canal en précisant *S* et *E* du canal choisi.

Cette définition évite des problèmes avec l'attribution d'adresses. Comme l'identificateur du canal contient l'adresse de la source, chaque source peut gérer indépendamment les adresses pour ses canaux. Chaque source possède une liberté complète pour disposer d'adresses multicast. Une adresse multicast E peut être utilisée en même temps par plusieurs sources pour former plusieurs canaux différents.

Cette idée a été développée et renommée SSM (*Source-Specific Multicast*) [60].

12.5 Gestion de groupes

La gestion des groupes multicast comporte deux fonctions de base :

- **Abonner un utilisateur à un groupe multicast** : un utilisateur qui veut recevoir le trafic désiré doit se manifester et joindre le propre groupe multicast. Pour cela il rejoint l'arbre de distribution et accepte le trafic.

- **Désabonner un utilisateur d'un groupe multicast** : un utilisateur qui ne veut plus recevoir de trafic doit quitter le groupe en partant de l'arbre de distribution.

Le protocole IGMP est utilisé pour accomplir ces deux fonctions. Dans son évolution, au cours de trois versions, nous voyons comment la gestion de groupes change. Afin de pouvoir discuter cette évolution nous présentons d'abord brièvement les trois versions.

IGMPv1

IGMPv1 (*Internet Group Management Protocol, version 1*) [41] est la première version du protocole présentée. Elle précise les exigences de base pour la participation de membres dans les groupes multicast.

La participation dans un groupe est dynamique. Cela veut dire que toutes les machines sont libres de le joindre ou de le quitter à n'importe quel moment, selon leurs décisions personnelles. Il n'y a pas de restriction ni sur la localisation de machines, ni sur le nombre de machines dans un groupe. Une machine peut être en même temps membre d'un ou de plusieurs groupes divers. Pour envoyer les données à un groupe, la machine n'est pas obligée d'être membre de ce groupe.

Le protocol IGMPv1 définit deux messages de contrôle pour gérer la participation dans des groupes multicast.

- *Membership report* : des messages de ce type sont envoyés par les machines qui souhaitent recevoir le trafic d'un groupe multicast.

- *Membership query* : ce sont des messages envoyés par des routeurs sur des réseaux locaux pour savoir s'il y a des machines interessées par le trafic multicast.

Pour s'abonner à un groupe, une machine doit envoyer un message *Membership report* à un routeur. Elle peut le faire dans les deux cas suivants :

- elle souhaite explicitement recevoir le trafic multicast du groupe défini dans le message,
- en réponse à un message *membership query* reçu de la part d'un routeur local.

Dans IGMPv1, il n'y a pas de possibilité de quitter un groupe explicitement. Le routeur envoie périodiquement les messages *membership query* pour chaque groupe multicast desservi. Toutes les machines intéressées par le trafic multicast doivent à chaque fois répondre par un *membership report* pour groupes concernés. Une machine qui ne désire plus recevoir de trafic ne répond pas aux messages *membership query*. Si le routeur ne reçoit plus de messages *membership report* pour un groupe, pendant temps prédéfini, il arrête la diffusion de ce groupe.

IGMPv2

Pour assurer plus d'efficacité, la version IGMPv2 (*Internet Group Management Protocol, version 2*) a été proposé [61]. Dans cette version, les machines peuvent explicitement quitter un groupe si elles ne désirent plus recevoir de trafic de ce groupe. Dans ce but, le protocole introduit un message *leave group* qui est envoyé à un routeur par une machine

qui veut se désabonner. Cela limite le trafic provoqué par la diffusion des groupes qui n'ont plus d'abonnés.

IGMPv3

La troisième version du protocole a été proposée [62] pour permettre le support de SSM (décrit dans la section 12.4.1). Elle ajoute le support du filtrage selon l'adresse source. Cela veut dire qu'un utilisateur peut préciser aussi la source dont il souhaite recevoir des paquets.

Chacune parmi ces trois versions du IGMP est compatible avec les versions précédentes. Les trois protocoles IGMP ont été proposés pour IPv4. Pour IPv6, il existe deux protocoles, MLD (*Multicast Listener Discovery*) [63] équivalent à IGMPv2 et MLDv2 (*Multicast Listener Discovery, version 2*) [64] équivalent à IGMPv3.

12.6 Gestion de sessions

La notion de session multimédia est directement liée au domaine de la distribution de contenu multimédia. Une session multimédia est un ensemble d'émetteurs, de récepteurs et de flots de données transmises depuis des émetteurs à des récepteurs. Des conférences multimédia multiutilisateurs, des appels téléphoniques, l'apprentissage à distance, la diffusion de contenu multimédia sont des exemples de session multimédia.

Dans les paragraphes suivants, nous présentons les bases de la gestion de sessions. Nous expliquons comment elles peuvent être annoncées, initiées et décrites.

12.6.1 Annoncer des sessions

La procédure d'annoncer de sessions est un ensemble de mécanismes par lesquels, la description d'une session est transmise aux utilisateurs de façon proactive (la description de la session n'est pas explicitement demandée par les utilisateurs) [65].

Dans le but d'annoncer des sessions multimédia aux utilisateurs potentiels, un répertoire de sessions a été proposé pour stocker des descriptions de sessions en cours et les diffuser périodiquement en utilisant un protocole multicast. Les utilisateurs écoutent sur une adresse multicast bien connue et reçoivent des descriptions de sessions annoncées. Cela permet aux utilisateurs de savoir où et quelles sessions sont disponibles. Une fois qu'un utilisateur reçoit une description de session, il peut joindre la session de manière autonome. Le protocole pour annoncer des sessions s'appelle SAP (*Session Announcement Protocol*) [66].

12.6.2 Initier des sessions

SIP (*Session Initiation Protocol*) [67] [68] est un protocole applicatif de signalisation qui permet d'établir, de modifier et de terminer des sessions multimédia. Il est utilisé pour annoncer des sessions aux utilisateurs ou pour inviter des utilisateurs aux sessions déjà établies par un autre moyen. Les messages SIP portent les descriptions de sessions pour permettre aux utilisateurs de connaître les paramètres des sessions. La fonction de SIP est purement informative. Il ne réserve pas de ressources réseau. Il n'alloue pas non plus d'adresses multicast pour des sessions.

12.6.3 Description de sessions

La description d'une session est faite selon un format précis pour livrer aux utilisateurs suffisamment d'informations pour découvrir et pour participer à une session multimédia [65].

Pour cette raison, au moment de l'initiation d'une session multimédia, on doit informer tous les utilisateurs potentiels des détails de la session, des adresses de transport, des détails du contenu multimédia ou d'autres attributs ou informations optionnelles. Pour préparer des descriptions de sessions utilisées par SAP, un protocole de description SDP (*Session Description Protocol*) a été proposé [65]. SDP a pour but de décrire les sessions afin de les annoncer, initier ou inviter les utilisateurs. Ces descriptions ont un format textuel qui contient des informations détaillées sur la session (notamment, celles mentionnées au début du paragraphe). Elles sont stockées dans un répertoire de sessions.

L'idée de SDP est générale et ne dépend pas de la méthode de transport des descriptions, on peut utiliser, par exemple SAP, SIP, RTSP (*Real Time Streaming Protocol*) [69], HTTP (*HyperText Transfer Protocol*) [70], LDAP (*Lightweight Directory Access Protocol*) [71] ou le courrier électronique. Cela rend le protocole universel et lui permet d'être appliqué dans un vaste choix d'environnements réseaux et par des applications diverses.

Les protocoles cités ci-dessus ont été conçus en tant que parties d'une architecture complexe proposée par l'IETF pour organiser la communication multimédia. Dans cette architecture, on trouve des protocoles comme RSVP [72] (pour réserver de ressources réseau), RTP [73] (pour transporter des données en temps réel), RTSP [69] (pour contrôler la livraison d'un flot multimédia), SAP [66] (pour annoncer des sessions multimédia en utilisant le multicast), SIP [67] [68] (pour initier des sessions), SDP [65] [74] (pour des descriptions de sessions multimédia) et MSDP (*Multicast Source Discovery Protocol*) [75] (pour échanger des informations entre des domaines indépendants). Le fonctionnement de chaque protocole est indépendant des autres.

12.7 Conclusion

Dans ce chapitre nous avons présenté l'architecture actuelle de protocole multicast IP. Les protocoles de multicast ainsi que les techniques de gestion de groupes et de sessions ont été décrites. Nous pouvons constater que l'architecture est très complexe et difficile à déployer [76]. Elle est composée de plusieurs protocoles qui doivent coopérer à des niveaux différents. Leur coopération exige l'existence de services supplémentaires. Les opérateurs qui supportent le multicast doivent échanger de nombreuses informations. Les plus grands problèmes de cette architecture sont :

- la gestion de groupes,
- la distribution d'adresses multicast de manière distribuée,
- le passage difficile à grande échelle.

Tout cela nous amène à dire que cette approche n'est pas bien adaptée à notre contexte de réseaux à très haut débit et à nos besoins actuels de distribution de contenu. Pourtant, elle nous fournit des idées intéressantes que nous reprendrons dans notre approche présentée dans la suite de cette thèse.

Chapitre 13

L'architecture orientée contenu

Après la présentation générale de l'environnement des protocoles multicast actuels dans le chapitre 12, nous pouvons constater qu'il n'est pas bien adapté à nos besoins actuels. Surtout, il ne passe pas à l'échelle globale où un grand nombre d'opérateurs réseau indépendants coexiste. La coopération au niveau global exige de nombreux échanges de données pour supporter le service du multicast. De plus, cet environnement est très complexe. Pour assurer le fonctionnement de base, l'utilisation de plusieurs protocoles dans les différentes couches réseau est nécessaire [76][77].

Pour pouvoir fournir un contenu multimédia d'une façon simple et efficace à une multitude d'utilisateurs répartis géographiquement, nous présentons dans ce chapitre une architecture basée et orientée contenu. Du point de vue topologique, cette architecture est basée sur un réseau à très haut-débit. Il permet de lier des opérateurs répartis géographiquement et de leur offrir rapidement des contenus émis par d'autres participants. Nous ne nous limitons pas à fournir le contenu demandé. Profitant du très haut-débit, nous poussons tous les contenus vers toutes les extrémités du réseau. Cette façon de rapprocher le contenu permet aux utilisateurs d'en profiter comme s'il était disponible localement, de manière simple et rapide. Cette architecture est aussi une base pour notre approche à la transmission multipoint orienté contenu. Notamment, nous proposons de redéfinir la notion de session que nous appelons dans la suite le *canal*. Notre définition s'appuie sur le format de données qui intègrent un contenu avec sa description et fournit un support spécifique pour l'adressage et pour le routage orienté contenu. L'accès au contenu pour les utilisateurs est facilité pour cette définition orientée contenu. Dans l'adressage unicast, une adresse indique une machine, dans le multicast, une adresse signifie un groupe d'utilisateurs ; dans notre approche, une adresse décrit le contenu transmis. Les utilisateurs s'abonnent au contenu et le protocole du routage le transmet selon l'adresse attribuée. Cette adresse est directement liée au contenu.

Afin de construire notre architecture orientée contenu, nous nous sommes inspirés de certaines idées de la littérature. Notamment, nous utilisons des techniques de construction

de l'arbre de distribution semblables à celles présentées dans le chapitre 12.2. Notre mécanisme pour éviter les boucles dans le graphe de distribution est similaire à celui du chapitre 12.2.1. L'idée d'adressage unique géré individuellement par chaque émetteur a été inspirée de SSM [60]. Les idées mentionnées n'ont pas été reprises dans leurs formes originales. Nous les avons adaptées et réunies pour offrir une approche nouvelle qui peut assurer la distribution du contenu de manière simple et efficace dans notre contexte.

Les sections suivantes présentent les composants de notre proposition qui conduisent vers une nouvelle définition de session. Nous commençons par la définition d'un format de contenu qui comprend le support pour l'adressage et le routage orienté contenu. Ensuite, nous présentons l'architecture. Celle-ci est composée de deux éléments essentiels : un cœur à très haut-débit et des réseaux locaux d'accès. Puis nous décrivons le routage basé sur l'adressage orienté contenu et le protocole du type multicast/diffusion. Nous étudions et discutons également les trois services nécessaires pour le fonctionnement complet du système. Finalement, nous signalons deux aspects qui ne sont qu'abordés dans cette thèse, mais qui peuvent être nécessaires dans certaines situations. Il s'agit de la sécurité (notamment de l'authentification des émetteurs de contenu et de l'autorisation des utilisateurs à le recevoir) et de la fiabilité de transmission.

13.1 Le *canal* - format de contenu

Pour définir le format du contenu, nous introduisons la notion de *canal de contenu*. Dans notre approche, nous considérons le canal comme la plus petite unité indépendante de contenu qui peut être traitée individuellement. Le canal est spécialement construit pour être distribué de manière simple et efficace. Il est composé de trois parties essentielles : le corps, l'étiquette et l'identificateur de contenu. Cette composition est complétée par un identificateur de trafic identique pour tous les canaux. Ces quatre composants sont décrits dans les sections suivantes. Sur la figure 13.1, on peut voir la composition schématique d'un canal.

13.1.1 Le corps

Le corps du canal comprend le contenu lui-même. Cela peut se traduire par un ou plusieurs fichiers et/ou un ou plusieurs flots de données (audio, vidéo, messages textuels, etc.). Dans le cas le plus simple, le corps contient un fichier ou un flot de données. Comme canal composé, nous prenons l'exemple d'un fichier MP3 qui contient une chanson avec un fichier texte avec des paroles de la chanson et un fichier JPEG avec une photo de l'artiste. Nous pouvons imaginer également un cas où plusieurs canaux partagant le même contenu. Par exemple, deux parties d'un film peuvent être diffusées séparément comme deux canaux. Dans ce cas, les deux canaux sont considérés comme indépendants.

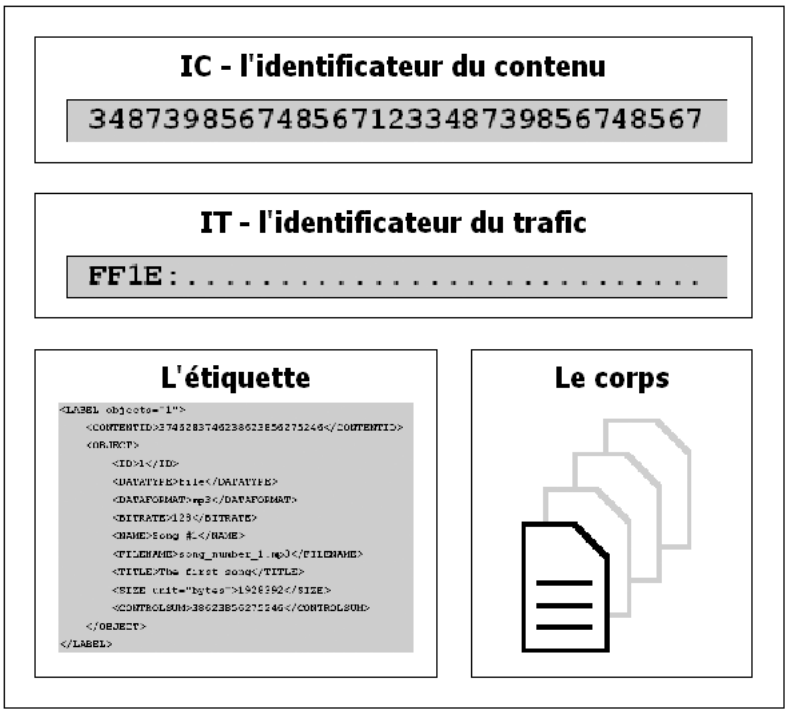

FIG. 13.1 – Composition schématique d'un canal

13.1.2 L'étiquette

L'étiquette contient une description textuelle d'un canal en format lisible par les humains. Les informations de l'étiquette servent aux utilisateurs à chercher, découvrir et demander un contenu afin de pouvoir participer à une session qui diffuse le contenu concerné. Ce sont des informations sur le contenu porté dans le canal : mots clés, type de contenu, format de données, format de compression, taille, émetteur, code de détection d'erreurs etc. Les informations contenues dans l'étiquette peuvent varier selon le type de contenu ou selon les décisions individuelles de l'émetteur.

Pour décrire un contenu, SDP conseille le format BNF/ABNF [78] [79]. Nous proposons d'utiliser XML [80]. XML (*eXtensible Markup Language*) dérivé du SGML [81], est un format ouvert qui permet de créer de manière facile et flexible des documents adaptés aux besoins. Dans notre cas, XML permet de modifier le format de l'étiquette selon les besoins de l'émetteur ou selon le modèle de diffusion. Cela permet aussi d'utiliser facilement des étiquettes ailleurs, par exemple, dans des pages web. De plus, de nombreuses API (*Application Program Interface*) existent pour manipuler facilement des documents XML.

Nous présentons un exemple d'étiquette dans la figure 13.2. Cette étiquette contient la description d'un canal composé de deux objets : un fichier MP3 avec de la musique et

```
<LABEL objects="2">
    <CONTENTID>91923374628374623862385627</CONTENTID>
    <SENDER>MTV</SENDER>
    <SENDERIP>2002:766b:213c:1:d0:8f06:f010:8</SENDERIP>
    <OBJECT>
        <ID>1</ID>
        <DATATYPE>file</DATATYPE>
        <DATAFORMAT>mp3</DATAFORMAT>
        <BITRATE>320</BITRATE>
        <NAME>Song #1</NAME>
        <FILENAME>song_number_1.mp3</FILENAME>
        <TITLE>The first song</TITLE >
        <SIZE unit="bytes">4928392</SIZE>
        <ARTIST>Michael Jackson</ARTIST>
        <CONTROLSUM>6001274868120152 46</CONTROLSUM>
    </OBJECT>
    <OBJECT>
        <ID>2</ID>
        <DATATYPE>file</DATATYPE>
        <DATAFORMAT>jpg</DATAFORMAT>
        <WIDTH>800</WIDTH>
        <HEIGHT>600</HEIGHT>
        <FILENAME>cover_number_1.jpg</FILENAME>
        <DESCRIPTION>The cover of the CD album</DESCRIPTION>
        <SIZE unit="bytes">81839</SIZE>
        <AUTHOR>Michael Jackson</AUTHOR>
        <CONTROLSUM>12228720102812016</CONTROLSUM>
   </OBJECT>
</LABEL>
```

FIG. 13.2 – L'exemple d'une étiquette pour un canal qui comprend deux objets

un fichier JPEG avec l'image d'une couverture de CD. Les deux objets possèdent des attributs différents, appropriés aux types des objets. Dans ce contexte, nous pouvons imaginer d'autres objets portés par le même canal, par exemple, un fichier texte avec les paroles de la chanson ou une photo du musicien. Les descriptions de ces objets doivent aussi être présentées dans l'étiquette.

13.1.3 Les identificateurs

Dans cette section nous présentons les deux identificateurs qui sont asscoeiés à un canal : l'IC (*Identificateur de Contenu*) et l'IT (*Identificateur de Trafic*).

L'idée principale de l'IC est d'identifier le contenu véhiculé dans un canal de manière unique. En même temps il sert aussi à acheminer des paquets qui forment un canal et pour cela il doit être suffisement petit pour être présent dans chaque paquet. De cette façon les routeurs peuvent prendre leurs décisions de routage en se basant sur le contenu véhiculé dans un paquet.

Nous proposons l'IC comme une valeur unique calculée individuellement pour chaque canal. Pour calculer l'IC, nous proposons une fonction de hashage qui, pour des données différentes, donne une valeur unique à la sortie. Le calcul de l'IC se fonde sur le corps et/ou l'étiquette du canal plus l'identificateur de l'émetteur. L'identificateur de l'émetteur est introduit dans la procédure du calcul du IC pour que le même contenu provenant de deux émetteurs différents obtienne deux IC différents. Nous proposons d'utiliser l'adresse IP réelle de l'émetteur comme son identificateur dans ce cas.

En ce qui concerne l'IT, il sert à supporter des décisions du routage. Tous les paquets qui appartiennent à notre protocole de diffusion portent un IT specifique (dans notre architecture, nous utilisons trois ITs différents : un pour les paquets qui véhiculent des données, le deuxième pour les messages de contrôle (voir le chapitre 13.3.2) et le troisième pour les messages portant des requêtes d'utilisateurs (voir le chapitre 13.3.4)). Pour cela la taille de l'IT, comme pour l'IC, doit être assez limitée pour le mettre facilement dans chaque paquet avec des données. Quand un paquet arrive à un routeur, celui-ci vérifie son IT. Si le paquet est reconnu comme un paquet du notre protocole de diffusion de contenu, il est redirigé vers un module qui s'occupe du routage orienté contenu. Dans le cas contraire, le paquet est traité normalement. Cela nous donne un routage par contenu qui est effectué en deux pas : tout d'abord un paquet est reconnu par son IT et ensuite il est traité et acheminé selon son IC.

Comme pour acheminer un paquet un routeur doit traiter ses deux identificateurs nous proposons les mettre dans l'en-tête du paquet IP. Nous avons à notre diposition les champs suivants :

– le type de protocole (ou type du prochain en-tête dans IPv6) - c'est champs est très

court (8 bits) et il est déja trés utilisé, alors il est peu intéressant,

– les champs extensions et options - comme ces champs ont une position variable, le coût de traitement peut être important,

– l'identificateur de trafic - comme sa sémantique est actuellement floue, l'utiliser peut être en conflit avec d'autres protocoles.

Cette analyse justifie le choix de redéfinir les champs adresses IPv6 pour les raisons suivantes :

– Dans le cas classique l'adresse source sert à (i) répondre à l'émetteur et (ii) identifier l'émetteur. Dans notre cas la diffusion de contenu est unidirectionnelle et va depuis un émetteur vers des utilisateurs. Normalement, un utilisateur n'a pas besoin de contacter l'émetteur, donc il ne doit pas obligatoirement connaître son adresse IP réelle. Si l'émetteur trouve cela utile, il peut mettre son adresse IP réelle dans une étiquette du canal pour permettre aux utilisateurs lui repondre. En ce qui concerne l'identificateur de l'émetteur, il peut être aussi placé dans une étiquette du canal. De cette façon l'émetteur peut fournir aux utilisateurs la fonctionnalité classique de l'adresse source sans utiliser le champ source dans le paquet IP.

– L'adresse destination est plus difficile à redéfinir car elle est partagée par les différents modes de transport (unicast, anycast, multicast). Dans la communication multipoint (multicast), l'adresse destination définit un groupe d'utilisateurs qui veulent recevoir les mêmes données. La plage des adresses réservées pour la communication multipoint est limitée à 112 bits [82] et nous proposons de les utiliser pour notre but.

Après cette analyse nous pouvons constater que nous disposons de 230 bits dans les champs source et destination. Regardons maintenant comment on peut les utiliser pour véhiculer l'IC et l'IT. Comme l'IC a besoin d'être assez grand nous proposons prendre 128 bits et de mettre l'IC dans adresse source IPv6. Par la suite, la table du routage est basée sur 128 bits de l'en-tête à offset fixe. Cela n'augmente pas le coût hardware si on veut réaliser le routage en matériel. En ce qui concerne l'IT, il a peu de valeurs. Pour l'instant nous pensons à 3 valeurs et proposons de choisir 3 adresses multicast pour le champ destination IPv6.

Au niveau de l'implémentation de l'IC, nous proposons d'utiliser l'algorithme de hashage MD5 [83]. MD5 est un algorithme qui, pour des données entrantes différentes, donne à la sortie une valeur unique de longueur 128 bits. Dans notre cas, MD5 prend comme entrée l'étiquette et/ou le corps d'un canal et l'identificateur de son émetteur et produit un IC du canal concerné. De manière syntéthique, nous pouvons présenter le calcul d'IC comme ci-dessous :

```
IC = MD5 ( ( corps ET/OU l'étiquette ) ET l'identificateur de l'émetteur )
```

Le même IC, une fois calculé pour un canal avant de commencer la distribution, est utilisé dans le champ source de tous les paquets portant les données de ce canal.

Pour l'IT, selon les indications données dans [84] et [85], nous avons choisi les adresses suivantes :

– `FF1E::1111:1111` - pour les paquets de données,
– `FF1E::1111:1112` - pour les requêtes d'utilisateurs (voir le chapitre 13.3.4),
– `FF1E::1111:1113` - pour les messages de contrôle (voir le chapitre 13.3.2).

Ces adresses sont les adresses IPv6 de type multicast global et possèdent un caractère temporaire. De cette façon, les routeurs qui ne supportent pas notre protocole ignorons les paquets avec ces destinations. Les routeurs reconnaissant notre protocole vont les traiter selon leurs tables de routage spécifiques par contenu.

Pour conclure, l'IC est une valeur qui identifie le canal de manière unique. Il est calculé à partir du contenu porté dans le canal et ensuite mis dans le champ source de chaque paquet IPv6 qui véhicule des données de ce canal. L'IT est présent dans le champ destination des paquets et sert à reconnaître les paquets qui appartiennent à notre protocole de diffusion. Les routeurs reconnaissent les paquets selons leurs ITs et ensuite les traitent et acheminent selons leurs ICs. Cela veut dire, que tout d'abord l'adresse destination est analysée (comme dans un routage classique) et le routage selon l'adresse source suit.

13.1.4 Le transport du canal

Pour transporter les canaux nous proposons d'utiliser les paquets UDP/IPv6. Nous fragmentons, selon le MTU de l'émetteur, les données du canal (l'étiquette et le corps) pour les mettre dans plusieurs paquets IPv6. Chaque paquet IPv6 utilise l'identificateur de trafic comme adresse destination, et l'identificateur de contenu comme adresse source. Tous les paquets d'un canal possèdent les mêmes adresses source et destination - c'est à dire, les mêmes identificateurs de contenu et les mêmes identificateurs du trafic. Chaque paquet contient un fragment de données. De manière schématique, cette opération est présentée sur la figure 13.3.

Comme le protocole UDP ne peut pas garantir la fiabilité de transmission, nous discutons le problème de la fiabilité dans chapitre 13.6.

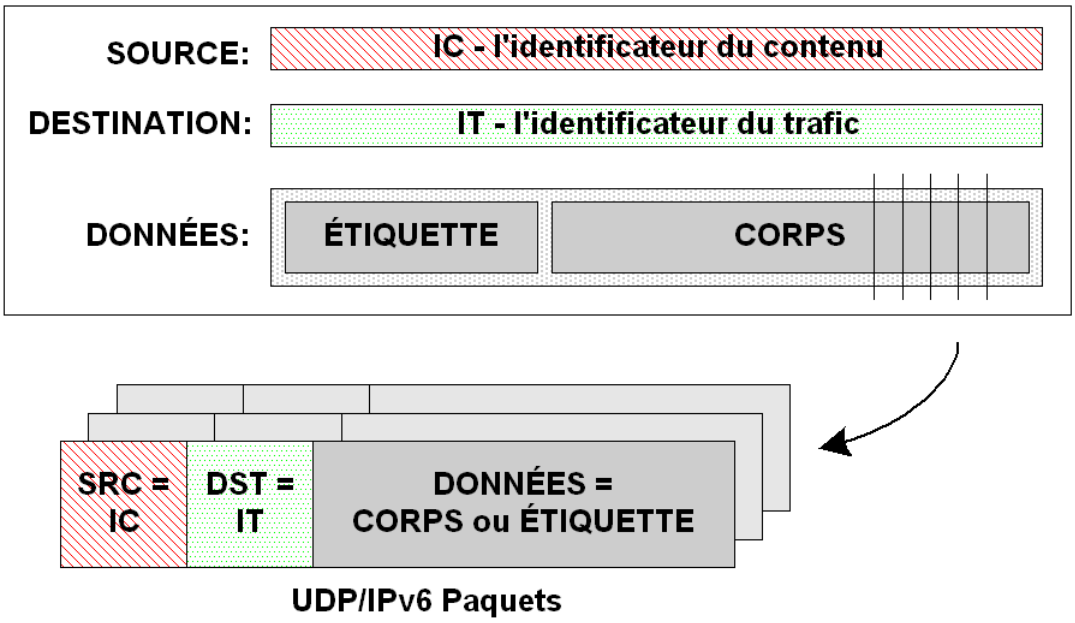

FIG. 13.3 – Des paquets UDP portant des données d'un canal

13.2 L'architecture de réseau

Dans cette section, nous décrivons l'architecture proposée qui est composé de deux parties : le réseau du cœur et des réseaux d'accès.

L'architecture que nous proposons est basée sur des réseaux à très haut-débit. C'est un élément important de notre architecture car, dans notre approche, nous insistons sur le fait que le débit est considérée comme virtuellement illimitée. Nous proposons de traiter le réseau comme un médium de transmission pour porter le contenu aux consommateurs. Cela se compare à l'air utilisé pour porter des ondes radioélectriques de la télévision ou de la radio. Le médium n'est utilisé que pour la livraison du contenu. Il n'est pas de la compétence du médium (le réseau dans notre cas) de savoir d'où vient le contenu, ni à qui il est destiné. Dans cette logique, le réseau ne tient pas compte ni des adresses source ni des adresses destination. Quand le contenu apparaît dans le réseau, il est transporté vers tous les utilisateurs potentiellement ou explicitement intéressés par ce contenu.

Dans notre architecture, nous pouvons distinguer deux types de réseau. Le réseau de cœur qui profite du très haut-débit et des réseaux d'accès de débits différents (parfois fortement limités). Dans les sections suivantes, nous présentons les détails des deux parties.

13.2.1 Réseau du cœur

Cette section présente le réseau de cœur basé sur des connexions à très haut-débit. Les opérateurs réseau qui veulent diffuser et recevoir le contenu construisent un réseau de cœur. Ils possèdent des routeurs très puissants inter-connectés entre eux par des liens à très haut-débit. Nous appelons ces routeurs les CCR (le routeur de contenu de cœur - *Core Content Router*). Dans ce réseau, nous considérons que le débit est théoriquement illimité. Chaque routeur, dans le réseau du cœur, participe par défaut à la distribution du trafic. Cela veut dire que chaque routeur reçoit tous les contenus présents dans notre système. Néanmoins, les routeurs peuvent refuser de recevoir le trafic (ou une partie du trafic) à tout moment. Ils sont aussi libres par la suite de rejoindre la distribution du trafic.

Un opérateur (un routeur) qui veut s'attacher au réseau du cœur doit établir un lien à très haut-débit avec un routeur (chez un autre opérateur) faisant déjà partie du réseau. Un routeur peut posséder plusieurs liens à très haut-débit le connectant à plusieurs routeurs différents. A partir de ce moment, le nouveau routeur peut participer activement à la distribution du contenu. Cela veut dire qu'il peut diffuser et recevoir tout le trafic et peut fournir le contenu disponible dans le réseau à ses utilisateurs locaux. Comme cette partie est un réseau inter-opérateurs, nous la considérons comme assez statique. Les décisions des opérateurs concernant la participation à la distribution du contenu sont prises à long terme et elles ne changent pas souvent. Pour cette raison, les opérations consistant à attacher et à détacher les routeurs peuvent être effectuées manuellement. Pour cela, l'opérateur doit configurer les interfaces sur ses CCR en précisant les liens à très haut-débit qui mènent vers d'autres CCR.

Pour conclure, le réseau de cœur sert à transférer le contenu sur des liens inter-opérateurs de longue distance à très haut-débit. Ni des émetteurs ni des destinataires du contenu ne font partie de ce réseau. Ils sont connectés aux réseaux d'accès d'opérateurs.

13.2.2 Réseau local d'accès

Des réseaux d'accès regroupent des utilisateurs et leur permettent d'accéder au contenu. Ils sont gérés individuellement par des opérateurs locaux. Du point de vue de l'architecture, les réseaux locaux sont composés des routeurs locaux de contenu LCR (*Local Content Routers*) et d'utilisateurs desservis par l'opérateur. Ils sont connectés au réseau de cœur par un routeur intermédiaire de contenu ICR (*Intermediate Content Router*) qui est lui aussi géré par l'opérateur. ICR est un routeur à la fois CCR et LCR. Sur la figure 13.4, nous voyons le schéma de la connexion d'un réseau local au réseau de cœur par un ICR. Les points noirs représentent les CCR. Le grand point noir est un ICR. Les points gris sont les LCR. Les utilisateurs sont connectés aux LCR.

Nous expliquons la fonctionnalité du LCR et du ICR dans les paragraphes suivants.

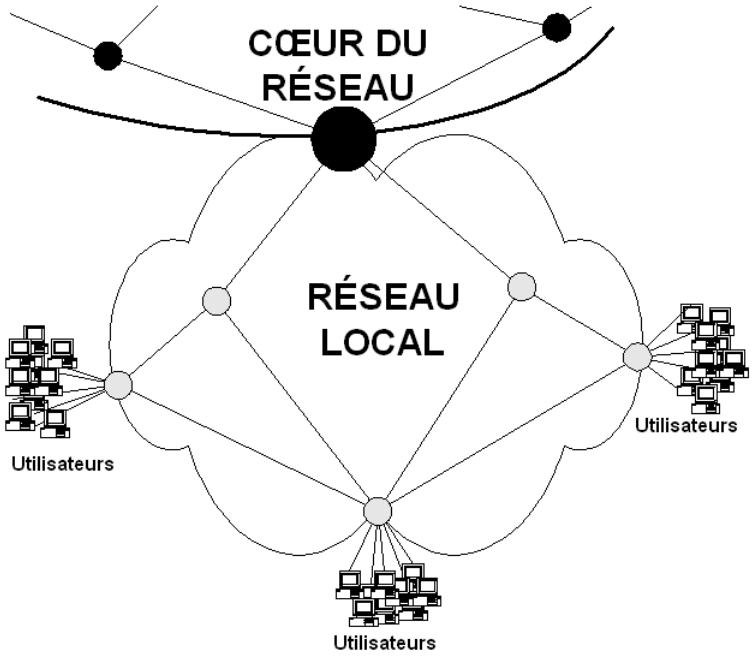

FIG. 13.4 – La partie locale de l'architecture

Les LCR participent à la distribution locale du contenu. Comme les réseaux locaux présentent des conditions (technologies, débit, etc.) hétérogènes, la distribution doit être plus sélective que dans le réseau de cœur. Les LCR ne distribuent que le contenu explicitement demandé par les utilisateurs locaux. Pour cela, ils transfèrent des requêtes d'utilisateurs à l'ICR et ensuite, ils distribuent aux utilisateurs les contenus fournis par l'ICR.

L'ICR accomplit en même temps deux fonctions, celle du CCR et celle du LCR. Comme il est un CCR, il est connecté aux autres CCR dans le réseau du cœur par des liens à très haut-débit. De cette façon, il participe à la diffusion du contenu entre les CCR. De l'autre côte, il est aussi un LCR. Cela veut dire qu'il est connecté par un autre lien au réseau local d'opérateur. Ayant accès à tous les canaux diffusés, l'ICR prépare un index de canaux et l'annonce aux utilisateurs pour les informer des canaux actuellement disponibles dans le réseau. C'est aussi l'ICR qui reçoit des requêtes d'utilisateurs. Il leur répond en leur fournissant le contenu.

En résumé, les réseaux locaux servent à distribuer aux utilisateurs locaux le contenu présent dans le réseau du cœur. Les réseaux locaux sont composés d'un ICR et de LCR.

13.3 Routage orienté contenu

Le format du canal proposé intègre une information importante du point de vue du routage des paquets portant du contenu. Il s'agit de l'identificateur de contenu (IC). La fonction principale de l'IC consiste à permettre le routage (une autre sert à identifier le canal car l'IC est unique pour chaque canal). Cette section présente le routage basé sur les IC - le routage orienté contenu.

Dans notre architecture, nous proposons de baser le routage sur les IC et non sur les adresses destinations réelles. Pour nous, ce n'est pas la localisation des utilisateurs qui est importante mais le contenu qu'ils désirent recevoir. Pour chaque canal nous avons toujours un grand nombre d'utilisateurs (même si seulement potentiels). Nous proposons d'utiliser un arbre de distribution pour acheminer le trafic. L'arbre de distribution, proposé dans les protocoles multicast/diffusion présentés en détail dans le chapitre 12, nous permet d'économiser des ressources réseau dans le cas où les mêmes données sont destinées à une multitude des utilisateurs. Même si le principe de routage est similaire, les mécanismes de création et de gestion de l'arbre de distribution sont différents dans le réseau de cœur et dans les réseaux d'accès.

Dans les sections suivantes, nous décrivons la table de routage et expliquons son fonctionnement. Ensuite, nous définissons les types de messages utilisés pour gérer des arbres de distribution. Finalement, nous présentons deux mécanismes de la création et de la gestion d'arbre de distribution : un pour le réseau de cœur et l'autre pour des réseaux d'accès. La création d'arbres de distribution consiste à configurer des tables de routage sur les routeurs. Nous décrivons aussi la procédure de reconfiguration des arbres qui consiste à modifier les tables du routage selon les besoins qui peuvent apparaître.

13.3.1 Table de routage

La table de routage par contenu sert à acheminer des paquets selon le contenu porté. Sur la figure 13.5 nous en voyons un exemple de table de routage avec des routes définies pour deux IC différents. Le couple {*IC, interface*} est une clé principale (PK - *primary key*) de la table et ne peut pas, alors, se répéter dans la même table. Pour chaque couple {*IC, interface*}, il existe une entrée dans la table du routage. Sur les CCR, les interfaces définies comme très haut-débit sont seules considérées. Pour chaque PK il y a un état définie. Cet état détermine ce qu'un CCR (*Core Content Router*) doit faire sur l'interface concernée avec un paquet portant l'IC. Nous distinguons trois états suivantes :

INCOMING : une interface est marquée INCOMING pour un IC pour dire que les paquets portant l'IC peuvent arriver au routeur uniquement par cette interface. Si un paquet arrive par une interface qui n'est pas marquée INCOMING pour son IC, le paquet doit être silencieusement rejeté. Les paquets arrivant sur l'interface INCO-

IC	INTERFACE	STATE
IC_1	eth0	INCOMING
IC_1	eth1	ACTIVE
IC_1	eth2	ACTIVE
IC_2	eth0	INCOMING
IC_2	eth1	ACTIVE
IC_2	eth2	PRUNED

FIG. 13.5 – La table de routage

MING sont destinés à être retransmis sur d'autres interfaces définies.

ACTIVE : une interface marquée ACTIVE est destinée aux retransmissions des paquets en aval de l'arbre de distribution. Pour un paquet qui arrive au routeur par l'interface INCOMING, le routeur cherche dans sa table du routage les interfaces ACTIVE et transmet le paquet sur ces interfaces.

PRUNED : l'interface qui ne participe pas à la diffusion d'un canal est marquée PRUNED pour l'IC concerné. Toutes les interfaces qui sont exclues de la diffusion du contenu doivent être marquées PRUNED.

13.3.2 Les messages de contrôle

Dans notre approche, nous cherchons la simplicité de la solution proposée. Au niveau de routage, nous essayons de limiter le nombre de messages échangés entre des routeurs pour gérer les arbres de distribution et pour effectuer le routage. Nous tenons particulièrement à ce que la configuration et la reconfiguration des arbres soient provoquées par des données transmises et que le nombre de messages de contrôle soit marginal. Néanmoins, nous devons introduire quelques messages de contrôle indispensables pour gérer certaines situations. Nous les décrivons rapidement ici. Une description plus complète est présentée dans les sections suivantes concernant le routage. Les algorithmes complets de traitement de tous les messages sont présentés dans l'annexe B. Tous ces messages (sauf un) sont accompagnés par un IC et s'appliquent uniquement à lui.

CORE_TRAFFIC n'est pas un pur message du contrôle, mais un paquet portant des données. Ce type de paquet peut provoquer la création d'un arbre de distribution, s'il n'existe pas encore pour l'IC concerné.

CORE_PRUNE est le message utilisé par les CCR pour refuser la réception de trafic (d'un canal) dans le but d'éviter de créer des boucles dans les arbres de distribution.

CORE_RENEW est le seul message qui n'est pas accompagné d'un IC spécifique, parce qu'il s'applique au trafic entier (tous les canaux). Ce message est utilisé pour rejoindre des arbres de distribution précédemment quittés.

CORE_FAILURE est le message envoyé après la détection d'une panne. Les arbres de distribution doivent être recréés suite à la réception de ce message.

LOCAL_ANNOUNCE est un message utilisé par un ICR pour manifester sa présence dans un réseau local.

LOCAL_TRAFFIC correspond au CORE_TRAFFIC dans un réseau d'accès. C'est un paquet portant des données. Dans le cas d'un réseau d'accès, ce paquet ne provoque pas la création de l'arbre de distribution car l'arbre est créé à priori, avant que la distribution ne commence.

LOCAL_REQUEST est le message envoyé par un utilisateur pour demander un canal.

LOCAL_STOP est un message destiné à arrêter le canal quand l'utilisateur ne veut plus le recevoir. L'utilisateur envoie un message de ce type à son LCR par défaut. Après ce message, il est débranché de l'arbre de distribution local pour l'IC concerné et ne reçoit plus ce canal.

En ce qui concerne les IT, les messages décrits ci-dessus ne sont pas traités uniformément. Les messages CORE_TRAFFIC et LOCAL_TRAFFIC sont considérés comme des données et contiennent l'IT présenté dans le chapitre 13.1.3. Les messages LOCAL_REQUEST et LOCAL_STOP sont considérés comme des requêtes et contiennent l'IT présenté dans le chapitre 13.3.4. Tous les autres types de message sont considérés comme des messages de contrôle. Pour les paquets portant ces messages, nous avons choisi l'adresse `FF1E::1111:1113` comme l'IT. Avec l'IT de ce type, les messages sont reconnus par des routeurs et traités proprement comme des messages de contrôle.

13.3.3 L'arbre du diffusion dans le réseau du cœur

Dans le réseau de cœur tous les CCR participent à la diffusion du trafic suite à l'application d'un mécanisme du diffusion. La diffusion, comme défini dans [44], propose un arbre de distribution qui prend sa racine dans la source émettant les données et qui s'étend pour couvrir tous les nœuds dans le réseau. Nous reprenons cette idée pour créer l'arbre de distribution dans le réseau de cœur. Nous réutilisons aussi le mécanisme pour

éviter la création de boucles dans les graphes de distribution, et nous proposons de l'activer sur le chemin le plus rapide vers la source. L'utilisation du chemin le plus rapide au lieu du chemin le plus court vers la source simplifie l'algorithme de la création de l'arbre car cette approche nous permet d'être complètement indépendant du protocole de routage unicast de la couche réseau. Dans ce cas, le protocole unicast n'est pas nécessaire.

La création de l'arbre de distribution

De manière plus détaillée, la création de l'arbre de distribution se déroule de manière suivante. Tout d'abord précisons que la création de l'arbre est provoquée par des données (*data driven*). Cela veut dire que ce sont les données elles-mêmes qui construisent un arbre et qu'il n'y a aucun message supplémentaire proposé dans ce but.

Des données sont émises par un émetteur (une source de données). Le premier paquet avec des données trace le chemin et provoque la création d'un arbre. Le routeur (CCR) qui reçoit un paquet vérifie son IC. S'il ne trouve pas cet IC dans sa table du routage, il marque l'interface d'arrivée comme INCOMING pour cet IC. Ensuite il marque toutes les autres interfaces de très haut-débit comme des interfaces ACTIVE pour cet IC. Finalement, il transfère le paquet sur toutes les interfaces ACTIVE. Si le CCR trouve l'IC dans sa table du routage, il vérifie l'interface par laquelle le paquet est arrivé. Si l'interface d'arrivée est marquée INCOMING dans la table du routage, le paquet est transféré sur toutes les interfaces qui sont marquées ACTIVE pour l'IC concerné. Sinon, le paquet est rejeté et le routeur manifeste sa volonté de ne plus recevoir ce canal par un message CORE_PRUNE.

Éviter des boucles

Un routeur qui ne souhaite pas recevoir de canal (pour éviter la création des boucles ou pour une raison administrative) envoie un message de type CORE_PRUNE au routeur CCR voisin dont il a reçu un paquet de ce canal. Puis, il marque cette interface comme PRUNED pour cet IC. Dès qu'il reçoit le message CORE_PRUNE, le routeur voisin CCR marque aussi son interface comme PRUNED pour cet IC et ne transfère plus ce canal sur cette interface. Comme les canaux de contenu sont gérés indépendamment, le CCR envoie un message du type CORE_PRUNE pour chaque canal dont il veut refuser la réception. Les messages du type CORE_PRUNE sont envoyés pour refuser les canaux qui arrivent déjà à un CCR par une autre interface ; c'est-à-dire quand l'interface d'arrivée est différente de celle INCOMING pour l'IC. Cette procédure évite la création de boucles dans les graphes de distribution.

La détection de pannes

Un CCR qui détecte une panne (nous n'entrons pas dans les détails pour déterminer comment un CCR détecte une panne) sur une interface (un lien) marquée INCOMING envoie un message du type CORE_FAILURE en aval de l'arbre d'IC concerné (sur toutes les interfaces marquées ACTIVE pour ce IC). Chaque CCR qui reçoit ce message envoie le message CORE_RENEW sur toutes les interfaces marquées PRUNED pour l'IC concerné. Cela sert à rejoindre l'arbre de distribution pour les interfaces précédemment débranchées. Ensuite le CCR transmet le message CORE_FAILURE en aval et efface toutes les entrées concernant l'IC de sa table de routage. Le routeur qui a détecté la panne est celui qui commence cette procédure. Il efface aussi toutes les entrées concernant l'IC. Un CCR qui reçoit un message CORE_RENEW change le statut de l'interface de PRUNED en ACTIVE et commence à diffuser le canal concerné sur cette interface. De cette façon, il commence la procédure de recréation de l'arbre de distribution qui se déroule en aval, comme décrit au début de cette section.

Cette procédure de dépannage est initiée séparément par le CCR concerné pour chaque canal dont l'interface INCOMING est tombée en panne.

13.3.4 Le multicast sélectif dans des réseaux locaux d'accès

Les réseaux locaux d'accès possèdent des caractéristiques différentes. Par conséquent, les utilisateurs peuvent bénéficier de ressources différentes. Pour cette raison, nous ne pouvons pas inonder des réseaux d'accès et nous devons proposer un mécanisme plus sélectif dans la diffusion de contenu. L'arbre de distribution ne couvre que les utilisateurs ayant explicitement manifesté leur volonté de recevoir un certain contenu. Aucun utilisateur ne reçoit de canal de contenu par défaut, sans une demande explicite de sa part.

Pour cette partie de l'architecture, nous proposons de baser l'arbre de distribution sur une idée de RP (*rendezvous point*) repris du PIM-SM [51]. Le RP est un routeur central, bien connu par tous les utilisateurs, qui gère la diffusion du contenu pour un réseau local. Pour le rôle du RP, nous utilisons un ICR qui interconnecte le réseau local au réseau du cœur. La fonction de cet ICR est de recevoir le contenu du réseau du cœur et de le fournir aux utilisateurs selon leurs demandes. Plus exactement, elle consiste en deux fonctions essentielles qui sont décrites dans les paragraphes suivants.

Annoncer l'ICR dans le réseau local

Pour être connu dans le réseau local, l'ICR doit manifester sa présence à tous les utilisateurs de ce réseau. Dans ce but, l'ICR utilise le message du type LOCAL_ANNOUNCE. Tous les routeurs LCR voisins de l'ICR sont inondés par ce message. Ensuite chaque LCR qui le reçoit marque l'interface d'arrivée comme une interface qui mène à l'ICR et transfère

le message à toutes les autres interfaces. Un mécanisme similaire au protocole RPF [44], mais basé sur le chemin le plus rapide, est appliqué pour éviter la création de boucles. Une fois le message parvenu aux utilisateurs terminaux, un arbre est établi (les utilisateurs terminaux ne traitent pas ce message et ne font pas partie de l'arbre). Cet arbre est ensuite utilisé pour transmettre les requêtes des utilisateurs (les messages de type LOCAL_REQUEST) à l'ICR. Chaque utilisateur participe à un seul arbre de ce type, cela veut dire qu'il a un seul ICR à qui il destine ses requêtes.

S'abonner et se désabonner d'un canal

Les utilisateurs utilisent le message LOCAL_REQUEST pour s'abonner à un canal et le message LOCAL_STOP pour s'en désabonner. Ces deux types de message sont envoyés à leur ICR.

Pour s'abonner à un canal, l'utilisateur envoie un message LOCAL_REQUEST (qui contient l'IC du canal demandé) à son LCR le plus proche (son routeur par défaut). Le LCR vérifie l'interface d'arrivée. Si l'interface d'arrivée est déjà marquée ACTIVE, il incrémente la valeur du compteur pour cette interface (il y a un compteur pour chaque couple {*IC, interface*}) et supprime silencieusement le message. Sinon, il marque l'interface d'arrivée comme ACTIVE, met 1 comme la valeur de son compteur et transfère le message en amont de l'arbre créé grâce au message LOCAL_ANNOUNCE d'ICR (décrit dans le paragraphe précédent). Une fois la première requête arrivée à l'ICR, il commence la distribution du canal en utilisant l'arbre de distribution créé par les requêtes.

De la même façon, pour se désabonner d'un canal, l'utilisateur envoie un message LOCAL_STOP à son LCR le plus proche. Le LCR décrémente la valeur du compteur pour l'interface d'arrivée. Si la valeur atteint zéro, cela veut dire qu'il n'y a plus de consommateurs pour ce canal sur cette interface. Il change le statut de l'interface à PRUNED. Si le LCR n'a plus d'interfaces marquées ACTIVE pour l'IC, il transfère le message LOCAL_STOP en amont et nettoie la table du routage pour l'IC. Sinon, il supprime silencieusement le message. L'utilisateur doit envoyer un message LOCAL_REQUEST/LOCAL_STOP à chaque canal auquel il veut s'abonner ou se désabonner.

Pour que des paquets portant des requêtes puissent être reconnus correctement par des routeurs (par les LCR (*Local Content Router*) dans ce cas) ils portent un IT spécifique. De la même façon que pour des paquets avec des données (voir chapitre 13.1.3), nous avons choisi pour des paquets avec des requêtes l'adresse destination `FF1E::1111:1112`. Celle-ci possède les mêmes caractéristiques que les autres IT. Si un LCR reçoit un paquet avec cet IT, il le traite comme une requête.

Sur la figure 13.6, nous voyons la procédure d'abonnement des utilisateurs. Les petits points noirs sont les LCR et le grand point noir l'ICR. Les flèches pointillées montrent le chemin des messages LOCAL_REQUEST/LOCAL_STOP et les flèches continues le flot du trafic dans le sens inverse.

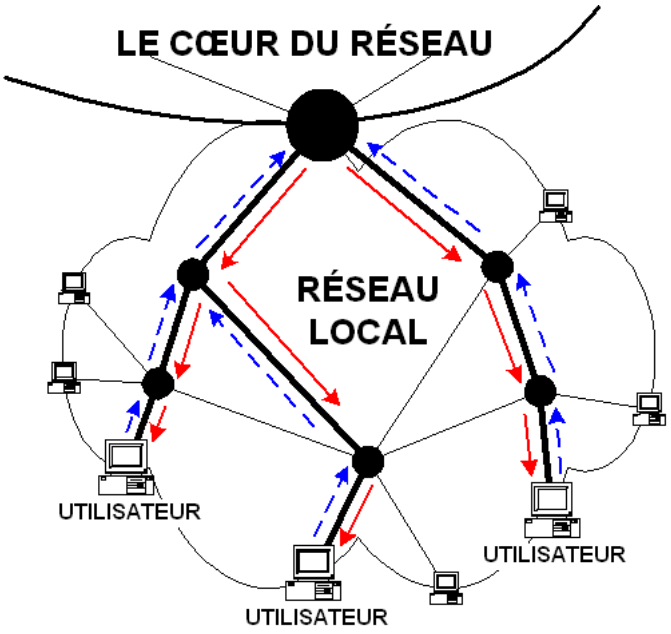

FIG. 13.6 – L'abonnement d'utilisateurs

13.4 Les services - la mise à la disposition du contenu

Pour compléter les fonctionnalités de notre architecture, nous devons définir des services essentiels du point de vue des utilisateurs. Ils servent à mettre à la disposition des utilisateurs (se trouvant dans des réseaux locaux d'accès) le contenu diffusé par les émetteurs.

Annonceur : pour annoncer aux utilisateurs les contenus (les canaux) disponibles.

Navigateur : pour chercher et demander le canal désiré.

Player : pour recevoir et consommer le contenu (le canal choisi).

Leurs descriptions sont présentées dans les sections suivantes.

13.4.1 Annonceur - annoncer le contenu

Le service *annonceur* sert à regrouper, indexer et annoncer les canaux présents dans le réseau de cœur aux utilisateurs dans les réseaux locaux d'accès. Ce service est installé sur chaque ICR et fonctionne de façon à desservir son réseau local. En tournant sur l'ICR, l'*annonceur* a l'accès à tous les canaux diffusés dans le réseau du cœur. A partir des

étiquettes de tous ces canaux, l'*annonceur* crée un canal spécial que nous appelons *index de canaux* (index par la suite).

L'index est un canal qui est construit de la même manière qu'un canal usuel mais il possède certaines caractéristiques qui le rendent unique. Tous les index créés sur tous les ICR possèdent toujours le même l'IC. De plus, pour que cet identificateur soit bien connu des utilisateurs, il ne se compose que de zéros. L'étiquette d'index contient sa description : le temps de création, le nombre de positions (canaux) dans les index, les informations sur l'ICR (le créateur d'index) et éventuellement les informations optionnelles. Dans son corps, l'index comprend les étiquettes de tous les canaux disponibles et indexés par l'ICR au moment de la création de l'index. L'identificateur du trafic (IT) pour l'index est le même que pour les canaux usuels. L'index de canaux peut être signé (de manière facultative) par l'ICR. Le contenu du canal d'index est régulièrement rafraîchi (le canal est recréé) pour être constamment à jour et pour pouvoir toujours annoncer les canaux qui sont réellement disponibles aux utilisateurs.

Pour conclure, le rôle d'*annonceur* se compose de deux fonctions essentielles : créer l'index à partir des canaux disponibles et l'annoncer aux utilisateurs dans le réseau local.

13.4.2 Navigateur - choisir et demander le contenu

Le service *navigateur* sert à présenter le contenu disponible à un utilisateur pour lui permettre d'y souscrire. Ce service est installé sur les machines d'utilisateurs qui veulent recevoir le contenu. Il accomplit quatre fonctions de base :

Demander l'index de canaux : le *navigateur* demande la réception de l'index de canaux à l'ICR approprié. Comme l'IC d'index est bien connu et toujours le même, cette tâche est entièrement automatique et ne nécessite pas l'intervention de l'utilisateur. La demande est transférée à l'ICR sur l'arbre de distribution qui existe grâce à la construction présentée dans la section 13.3.4.

Afficher l'index à l'utilisateur : une fois l'index demandé et reçu, le *navigateur* affiche son contenu à l'utilisateur. Cela se traduit par la présentation à l'utilisateur de toutes les étiquettes de canaux disponibles.

Permettre à l'utilisateur de choisir le canal : suite à la fonction précédente, l'utilisateur reçoit les descriptions et les attributs de tous les canaux disponibles. A partir de ces descriptions, l'utilisateur choisit un canal (ou plusieurs) et prend la décision d'y souscrire. Pour choisir le canal, l'utilisateur peut le chercher dans l'index selon les attributs.

Souscrire à un canal choisi par l'utilisateur : une fois le canal choisi, l'utilisateur peut y souscrire en utilisant son IC. L'utilisateur peut commander plusieurs canaux

en même temps en précisant plusieurs IC ensemble. Les requêtes d'utilisateur sont transférées à l'ICR sur l'arbre de distribution présentée dans la section 13.3.4.

En bref, le *navigateur* permet à l'utilisateur de choisir et commander les canaux désirées.

13.4.3 Player - recevoir et consommer le contenu

Le *player* permet aux utilisateurs de recevoir les canaux et de consommer le contenu reçu. Étant donné que dans notre architecture nous ne limitons pas les formats du contenu, le *player* doit savoir gérer les types et les formats de données différents. Pour cela, il a la possibilité d'utiliser les applications extérieures, spécifiques au format de données reçues, installées sur la machine d'utilisateur. De cette façon, il lance le lecteur de la musique pour un fichier MP3 ou pour un flot audio, un éditeur texte pour un message textuel et un navigateur web pour un document HTML (*HyperText Markup Language*) ou XML. Le choix d'application est fait sur la base de description du canal reçu. Dans son étiquette chaque canal possède les informations nécessaires pour choisir une application approprié. Le *player* peut également intégrer la fonctionnalité d'afficher lui-même certains types de données. Cette fonctionnalité est similaire à celle de navigateurs web modernes. Ils peuvent recevoir tous les types de données, mais pour les consommer (afficher) ils doivent lancer des applications extérieures spécifiques au type des données comprises dans le fichier (ou flot) reçu.

En résumant, les trois services proposés permettent aux utilisateurs de savoir quel contenu est disponible dans le réseau, le commander, le recevoir et le consommer.

13.5 Sécurité

Même si la sécurité des systèmes informatiques est toujours une question essentielle, dans cette thèse nous n'abordons pas profondément cette problèmatique. Néanmoins, nous voyons deux aspects de sécurité que nous devons signaler ici du fait qu'ils sont particulièrement liés à l'approche présentée dans notre travail.

Comme notre protocole de diffusion n'utilise pas d'adresses source pour la transmission de paquets, il est impossible de savoir d'où vient un paquet. Par conséquent, il est impossible de savoir d'où vient le contenu (un canal) porté par les paquets. Donc, nous devons proposer un mécanisme qui permet d'identifier, de manière sure, l'émetteur de contenu. Il s'agit de l'authentification de l'émetteur du contenu du point de vue des utilisateurs.

L'autre aspect qui nous mène à discuter la sécurité est le contenu même auquel nous destinons notre architecture. Nous pensons ici plus particulièrement au contenu vidéo ou audio offert par des stations TV ou radio. Dans ce cas les émetteurs ne veulent pas toujours offrir leur contenu gratuitement à tous les utilisateurs. Pour certains contenus, ils veulent

limiter les utilisateurs à ceux qui en ont payé. C'est pour cela que nous proposons un mécanisme qui permet d'autoriser certains utilisateurs à recevoir un contenu.

Les deux aspects de la sécurité qui nous intéressent sont présentés dans les sections suivantes.

13.5.1 Authentification

L'authentification est une procédure qui permet de connaître l'identité d'un utilisateur avant d'entreprendre une action. Dans notre architecture, il s'agit de connaître l'identité de l'émetteur du contenu livré à l'utilisateur. Cette procédure doit permettre aux utilisateurs de savoir qui émet le contenu reçu.

Dans ce contexte, nous proposons d'enrichir la composition du canal par une signature électronique basée sur des certificats numériques. Des certificats peuvent être obtenus d'une AC (*Autorité de Certification*) de confiance. Chaque émetteur doit en obtenir un et l'utiliser avec tous ses canaux. L'utilisation de signatures se déroule de la même manière que pour signer les messages du courrier électronique. Une fois le canal signé (l'étiquette et/ou le corps) la signature est attachée au canal et est transmise après l'étiquette et avant le corps du canal. Comme dans le courrier électronique, les utilisateurs connaissent les autorités de certification parce qu'ils possèdent une liste des AC de confiance. Une fois la signature reçue, l'utilisateur est capable de la vérifier grâce à cette liste.

Cette proposition est correcte pour des données de taille déterminée (les fichiers et les flots des données existants), mais elle pose problème pour des flots des données du temps réel. Si on ne connaît pas l'ensemble de données à priori, il est impossible de les signer de cette manière. Dans ce cas nous devons nous limiter à signer uniquement l'étiquette. Cela est suffisant pour authentifier l'émetteur, mais n'est pas suffisant pour déterminer l'intégralité des données reçues. Il existe certains propositions pour signer les flots de donnée ([86][87][88]) que nous ne décrivons pas ici.

13.5.2 Autorisation

Avant de discuter du problème de l'autorisation, nous présentons rapidement les bases de deux méthodes cryptographiques. La première méthode est la cryptographie symétrique dans laquelle la même clé est utilisée pour chiffrer et déchiffrer les données. Cette méthode présente le problème de distribution des clés car la même clé doit être en possession de l'émetteur et du récepteur de messages. Malgré cela, son utilisation est très répandue parce qu'elle est rapide et efficace au niveau de calcul. Elle peut être facilement appliquée sur les outils de puissance limitée ou dans les situations où les grandes quantités des données doivent être rapidement chiffrées. Cette cryptographie est utilisée par le protocoles DES et 3DES [89][90][91].

L'autre méthode est la cryptographie asymétrique. Dans ce cas, il existe une paire de clés asymétriques, la clé publique et la clé privée correspondante, pour chiffrer et déchiffrer les données. La clé privée est gardée en secret par le propriétaire et utilisée pour déchiffrer les messages reçus. La clé publique est distribuée à tous les utilisateurs qui veulent envoyer les messages au propriétaire de la clé. Quand un utilisateur chiffre un message avec la clé publique, seul le destinataire peut le déchiffrer en utilisant sa clé privée. Cette méthode ne présente pas le problème de la distribution de clés parce que la clé publique peut être distribuée publiquement sans aucun risque. Elle est aussi moins susceptible aux attaques, car le chiffrement est plus fort. Néanmoins, cette méthode est assez complexe au niveau de calcul et ne peut pas être utilisée dans tous les contextes. La cryptographie asymétrique permet aussi de créer les signatures numériques pour identifier les émetteurs des messages. Cette cryptographie a été proposée par Whitfield Diffie et Martin Hellman en 1976 et est utilisée maintenant dans le protocole RSA [92].

Dans la pratique, nous rencontrons des solutions qui utilisent les deux méthodes en même temps. Cela pour gagner la sécurité de la cryptographie asymétrique d'un côté et l'efficacité de la cryptographie symétrique de l'autre. Comme exemple nous pouvons mentionner OpenSSH [93]. Cette application utilise des clés asymétriques au début d'une connexion pour échanger une clé symétrique qui est ensuite utilisée pour chiffrer la communication. La clé symétrique n'est crée que pour la durée d'une session.

Après cette introduction, nous pouvons passer à la discussion de l'autorisation dans notre architecture. L'autorisation est un ensemble d'actions entreprises pour déterminer ce qui est permis à l'utilisateur. Dans notre cas cela se traduit par déterminer si un utilisateur est autorisé à recevoir un certain contenu ou pas. La procédure d'autorisation se déroule de manière suivante.

Pour mieux comprendre notre approche de l'autorisation, nous commençons par présenter le rôle des clés cryptographiques dans notre architecture. Chaque émetteur crée une paire de clés asymétriques pour chaque canal diffusé. À l'inverse de l'approche classique, l'émetteur garde les clés publiques et distribue les clés privées. Les clés privées ne sont distribuées qu'aux utilisateurs concernés, cela veut dire à ceux qui peuvent recevoir le contenu (par exemple ceux qui ont payé pour le contenu), en dehors de notre architecture. Cela peut s'effectuer par courrier électronique, par une page web ou par un autre moyen. Cela peut être aussi le *navigateur* qui gère la distribution de clés du côté utilisateur en se communiquant directement avec un émeteur. Les clés privées sont accompagnées des IC des canaux pour lesquelles ils ont été crées. Les deux clés ont la duré de vie déterminée (un mois, un an, ...).

Pour chaque canal dont l'autorisation est exigée, l'émetteur crée une clé symétrique. Il chiffre cette clé avec la clé publique du canal et l'attache au canal entre l'étiquette et corps. Ensuite, il chiffre le corps avec la clé symétrique et diffuse le canal. L'utilisateur qui reçoit l'étiquette cherche dans son répertoire la clé privée correspondant à ce canal et essaye de déchiffrer la clé symétrique. S'il arrive à déchiffrer la clé symétrique, il peut continuer

en déchiffrant le corps du canal. Les utilisateurs possèdent les clés privées uniquement pour les canaux qu'ils sont autorisés à recevoir. Les clés symétriques ne sont valables que pour une émission d'un canal. Chaque nouvelle émission du même canal exige la création d'une nouvelle clé symétrique. Cela permet d'introduire des limites temporelles à l'accès aux canaux. Un utilisateur qui possède la clé symétrique pour un canal ne peut l'utiliser qu'une seule fois pour l'émission courante.

En résumant, nous devons insister sur le fait que les deux aspects présentés sont facultatifs dans notre approche. Ils ne constituent pas l'essence de l'architecture qui est entièrement fonctionnelle au niveau de distribution du contenu sans eux. Chacun des deux aspects peut être appliqué seulement à certains canaux ou par certains émetteurs, sans aucune influence sur les autres.

13.6 Fiabilité de transmission

Dans ce chapitre nous discutons la question de la fiabilité de transmission. Nous soulignons deux cas où la fiabilité de transmission pose des problèmes différents et indiquons des techniques qui peuvent aider à les résoudre.

L'architecture proposée est basée sur la diffusion/multicast. Comme d'autres protocoles de ce type, le nôtre utilise le protocole UDP dans la couche transport. UDP est un protocole sans connexion entre l'émetteur et le récepteur. Il ne garantit donc pas la fiabilité de transmission des données parce qu'il ne donne pas la possibilité de détecter des paquets perdus en cours de la transmission. Cela peut provoquer des problèmes dans certaines situations. De ce point de vue, nous pouvons distinguer deux types de données :

Données peu sensibles aux pertes de paquets : comme exemple, nous pouvons citer les flots audio et vidéo. Dans ce cas, une perte de paquets (un ou plusieurs) ne bloque pas complètement la réception de données mais dégrade la qualité de données reçues. La dégradation de la qualité dépend de la fiabilité de transmission. Plus il y a de paquets perdus, plus la qualité est dégradée. Le codage des informations prévoit les pertes et les compense.

Données sensibles aux pertes de paquets : les données où la perte d'un paquet détruit l'intégralité des données et ne permet pas de les utiliser. Les fichiers en sont un exemple. Si un paquet est perdu pendant la transmission, il est impossible de lire le fichier et la partie reçue correctement devient inutile. Dans notre composition du canal, l'étiquette est une partie sensible aux pertes de paquets. Pour être lisible par les utilisateurs, elle doit être reçue correctement en intégralité. Si le corps d'un canal comprend des fichiers, cela est aussi un cas de données sensible aux pertes de paquets.

Dans la littérature, on trouve deux approches de base pour fiabiliser la transmission. ARQ (*Automatic Repeat reQuest*) [94] est un mécanisme qui gère la fiabilité de transmission

par des retransmission de paquets perdus. L'autre approche est FEC (*Forward Error Correction*) [95] [96] [97] qui propose d'ajouter des données redondantes pour couvrir les pertes éventuelles qui peuvent se produire pendant une transmission.

Dans le domaine du multicast fiable [98] des recherches ont permis de proposer des méthodes qui peuvent assurer la fiabilité de transmission. Certains méthodes s'appliquent au niveau applicatif pour des types de données spécifiques, par exemple pour les flots vidéo ou audio [99] [100] [101] [102]. Il existe aussi des méthodes plus générales pour assurer la fiabilité de transmission [103] [104].

Pour résumer, notre approche est basée sur le protocole non fiable UDP. La couche applicative doit donc compenser les pertes de données pendant les transmissions. Les parties de données les plus sensibles dans notre architecture sont les étiquettes de canaux et les fichiers transférés dans les corps de canaux. Les flots audio et vidéo sont moins sensibles grâce aux méthodes de codage en vigeur.

Le problème est semblable au cas du multicast IP, et les solutions proposées (au niveau du codage de l'informations ou au niveau applicatif) devraient pouvoir s'appliquer de manière analogue.

13.7 Conclusion

L'architecture présentée s'appuie sur une nouvelle définition de la session de distribution. La notion de canal est proposée dans ce but. Le canal intègre le contenu, sa description et supporte le routage orienté contenu. De cette façon, le routage par contenu est possible ce qui introduit un nouveau adressage. Dans notre cas, l'adressage et les décisions du routage sont basés sur le contenu. Pour permettre aux utilisateurs d'accéder au contenu, trois services nécessaires ont été définis et présentés. Certains aspects, comme la sécurité et la fiabilité de la transmission, ont été mentionnés. Néanmoins, ils nécessiteraient une étude supplémentaire pour arriver à des solutions précises, concrètes et appropriées à notre contexte.

Chapitre 14

Scénario d'application

Pour mieux comprendre le fonctionnement de notre architecture orientée contenu, nous présentons dans ce chapitre un scénario type d'application. Nous spécifions les acteurs majeurs et leurs fonctions dans l'architecture : les opérateurs de liaisons à très haut-débit, les opérateurs de réseaux d'accès, les émetteurs du contenu et les utilisateurs. Dans la suite de ce chapitre nous donnons des exemples et des descriptions de chaque acteur.

Les opérateurs de liaisons à très haut-débit

Ces opérateurs forment le réseau de cœur. Ils possèdent et gèrent les CCR interconnectés entre eux par des liens à très haut-débit. Leur rôle est d'assurer le transfert de données à toutes les extrémités du réseau de cœur. Pour cela, chaque CCR possède plusieurs liens à très haut-débit le connectant à d'autres CCR (d'un même opérateur ou d'un autre). Pour ce rôle nous voyons des opérateurs internationaux et nationaux possédant un réseau à très haut-débit qui fournit un réseau de cœur à étendue globale ou au moins internationale. Les opérateurs de ce type ne desservent pas nécessairement des utilisateurs finaux voulant recevoir le contenu. Si un opérateur veut en même temps desservir des utilisateurs, il joue aussi le rôle décrit dans le paragraphe suivant.

Les opérateurs de réseaux d'accès

Ils bénéficient, eux aussi, du très haut-débit, mais ils ne utilisent qu'un seul lien à très haut-débit qui les connecte à un autre opérateur (du type précédent). Ainsi, ils sont uniquement des destinataires de contenu et ils ne transmettent pas de contenus aux autres opérateurs. Dans le réseau de cœur, ils sont placés aux feuilles des arbres de distribution. La fonction principale de ces opérateurs est de fournir le contenu à leurs utilisateurs. Pour cela, ils fournissent le réseau local d'accès qui regroupe les utilisateurs. Ce réseau local est composé de LCR et est relié au réseau de cœur par un routeur ICR qui est aussi géré

par cet opérateur. L'ICR annonce le contenu aux utilisateurs de l'opérateur, reçoit leurs requêtes et leur répond en leur fournissant le contenu demandé.

Nous pouvons imaginer une situation où plusieurs réseaux mondiaux de cœur existent et fonctionnent indépendamment pour de raisons diverses (économiques, technologiques, etc.). Dans ce cas, un opérateur local peut être en même temps connecté à plusieurs réseaux de cœur et recevoir le contenu de tous ces réseaux.

Pour ce rôle, nous voyons les fournisseurs d'accès Internet au niveau national ou régional qui desservent des particuliers. Comme exemple on peut citer des opérateurs d'ADSL, de télévision par câble ou de téléphonie fixe et mobile.

Les émetteurs de contenu

Ces sont des entreprises qui veulent diffuser son contenu à un grand nombre d'utilisateurs répartis géographiquement au niveau national ou international. Leurs clients potentiels sonr rattachés à divers réseaux d'accès gérés par des opérateurs différents.

Chaque émetteur prépare lui-même ses canaux. Il compose toutes les parties du canal : il calcule l'IC, forme l'étiquette et fournit le contenu. Finalement, il diffuse le canal en l'injectant dans le réseau de cœur. Il injecte le canal soit directement dans le réseau de cœur, soit par l'intermédiaire d'un réseau d'accès. Pour cela, l'émetteur ne doit pas nécessairement bénéficier du très haut-débit. Une fois le canal injecté dans le réseau de cœur, il est diffusé à toutes ses extrémités et puis à tous les utilisateurs qui le demandent. Quand l'émission d'un canal est terminée, l'émetteur peut recommencer. Le canal peut être diffusé en boucle en gardant le même IC. Un émetteur peut diffuser plusieurs canaux différents en même temps.

Des exemples des émetteurs dans ce scénario sont :

stations TV et radio : qui diffusent des émissions audio et vidéo, des films, de la musique, des retransmissions d'événements,

agences de presse : qui émettent des flots de messages textuels, des nouvelles, des informations de bourse, la météo.

Les utilisateurs

Il s'agit ici de consommateurs de contenu. Ce sont, en général, des utilisateurs domestiques qui veulent recevoir un contenu vidéo (télévision, films, émissions télé), audio (radio, musique) ou des messages textuels (nouvelles des agences de presse, la bourse, la météo). De point de vue topologique, des utilisateurs se trouvent dans notre architecture dans les réseaux locaux d'accès des opérateurs. Ils peuvent utiliser des technologies d'accès

différentes, l'ADSL, un modem classique, un téléphone portable ou autres. Les utilisateurs savent contacter leur ICR parce qu'il annonce sa présence dans tout le réseau d'accès. En utilisant l'application navigateur, les utilisateurs demandent à leur ICR l'index de canaux. Dans la liste de canaux actuellement disponibles, ils cherchent un canal selon leurs critères individuels, le demandent et le reçoivent. Un utilisateur peut demander et recevoir chaque canal disponible (un ou plusieurs). Si un canal est chiffré, l'utilisateur doit posséder une clé pour le canal. Les clés sont distribuées aux utilisateurs autorisés à la réception d'un canal par son émetteur. Si un utilisateur ne possède pas de clé, il peut recevoir le canal, mais il ne peut pas consommer le contenu.

Ce scénario montre un seul exemple, mais notre approche ne se limite pas à celui présenté. D'autres scénarios sont possibles.

Chapitre 15

Prototype

Pour évaluer le fonctionnement du routage par contenu nous avons implémenté un prototype simple du routeur CCR. Nous le présentons ici. Puis nous présentons des expériences menées avec ce prototype. Ils montrent le délai du traitement de paquets sur le CCR, le débit atteint et le fonctionnement des tables de routage, notamment leur création.

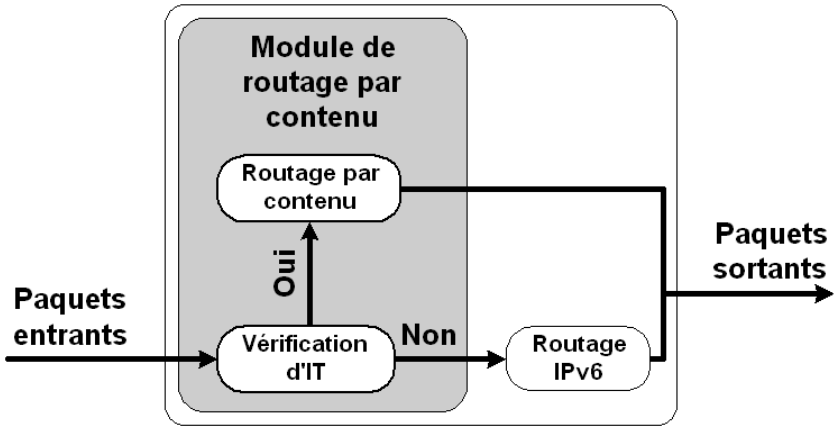

FIG. 15.1 – Le routage par contenu dans le noyau Linux

L'implémentation de ce prototype à été effectuée sur le système d'exploitation Linux. Nous l'avons réalisé comme un module externe au noyau. Des paquets qui arrivent au noyau sont examinés selon leurs IT. Dans notre cas, c'est l'adresse destination du paquet qui est examinée. Si on trouve qu'un paquet porte l'IT qui appartient à notre protocole (préfixe `FF1E::`), le paquet est dirigé vers notre module. Dans le cas contraire, le routage classique est effectué. Le module possède la table du routage par contenu qui comprend tous les IC connus avec les actions appropriées pour chaque interface. Sur la figure 15.1 nous voyons le chemin d'un paquet dans le noyau, présenté de manière schématique.

La configuration des interfaces qui participent à la distribution de contenu est implémentée dans le système de fichier `/proc`. Dans le répertoire `/proc/sys/net/ipv6/croute/dev` le module crée un fichier pour chaque interface réseau trouvée sur la machine. Chaque fichier comprend une seule valeur. Si l'interface est destinée à la distribution du contenu, la valeur dans son fichier est 1. Dans le cas contraire, la valeur est égale 0 et l'interface ne participe pas à la distribution. Cette valeur peut être facilement modifiée en utilisant une commande `echo`.

Pour simuler l'envoi de données, nous avons utilisé le logiciel **sendip** [105] qui permet d'envoyer de paquets paramétrés selon les besoins de l'utilisateur. Nous l'avons modifié pour qu'il puisse envoyer des flots de paquets avec l'adresse source et l'adresse destination manuellement configurées. Comme l'adresse destination, nous utilisions l'IT pour les données, notamment l'adresse `FF1E::1111:1111`. Comme l'adresse source, nous avons utilisé des valeurs différentes selon le besoin du test effectué.

15.1 Environnements de test

Pour le test décrit dans le chapitre 15.2.1 nous avons utilisé deux machines connectées au même sous-réseau (SR1) (figure 15.2). Une machine possédait trois interfaces réseau (`eth0`, `eth1` et `eth2`) et travaillait comme routeur CCR. L'autre, travaillait comme émetteur de flots de données et ne possédait qu'une seule interface réseau `eth0`.

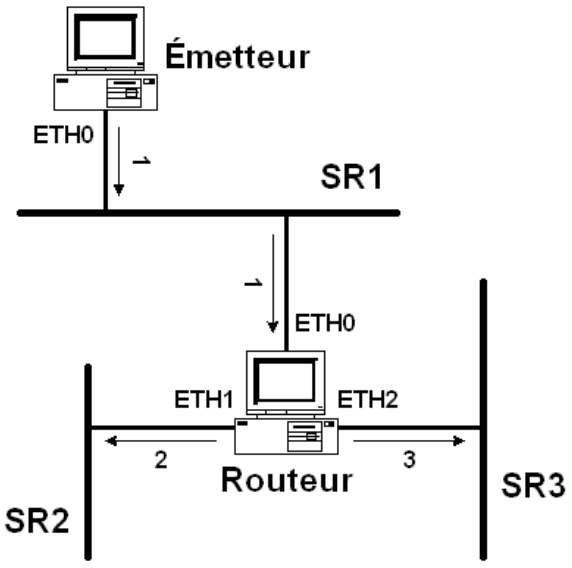

FIG. 15.2 – L'environnement pour les mésures du délai de traitement de paquets

Pour les tests décrits dans les chapitres 15.2.2 et 15.2.3 nous avons préparé un environnement simple comme montré sur la figure 15.3. Nous avons configuré trois sous-réseaux, SR1, SR2 et SR3 respectivement, de débit maximal 100Mb/s chacun. Dans le sous-réseau SR1 nous avons installé un émetteur de données E1 et dans les sous-réseaux SR2 et SR3 nous avons installé les récepteurs de données, R1 et R2 respectivement. Le routage par contenu à été assuré par deux routeurs, CCR1 (entre SR1 et SR2) et CCR2 (entre SR2 et SR3).

Sur toutes les machines dans cet environnement, nous avons installé Linux avec la version 2.4.23 du noyau. Le logiciel **sendip** à été installé sur la machine E1 pour envoyer des flots de paquets UDP/IPv6. Sur CCR1 et CCR2 nous avons installé notre module du routage par contenu. Sur les machines R1 et R2 nous avons installé un logiciel pour compter les données reçues. Avant chaque test, tous les logiciels ont été réinitialisés, et les tables du routage par contenu ont été vidées. Aucun logiciel dans cet environnement n'a été optimisé pour la performance.

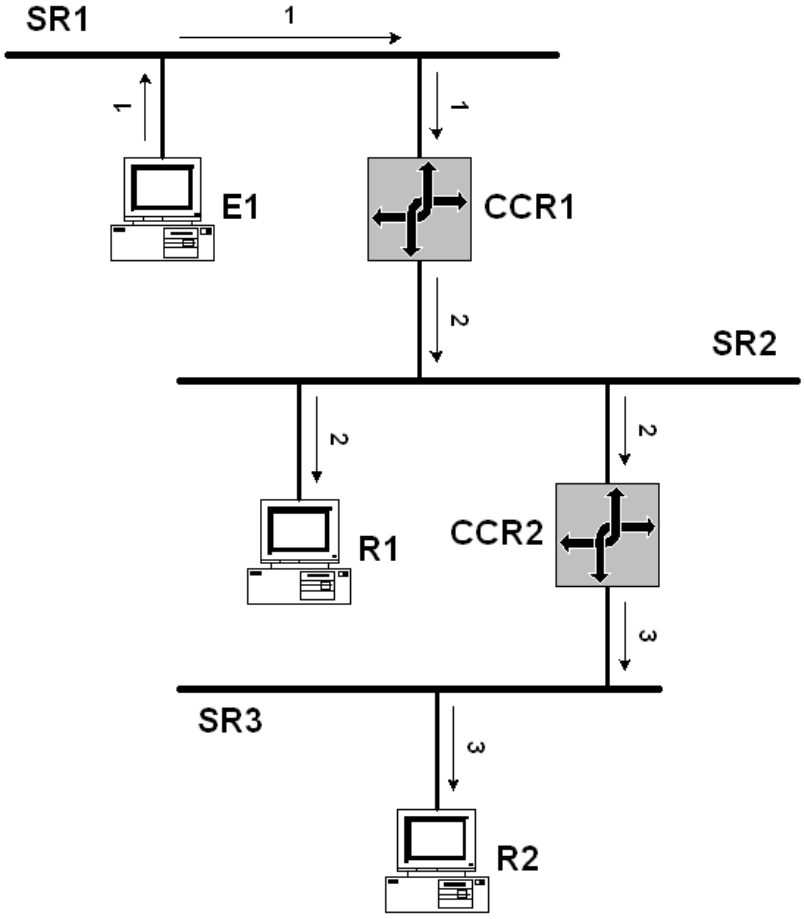

FIG. 15.3 – L'environnement pour les tests du routage par contenu

15.2 Expériences et résultats

15.2.1 Délai de traitement

Dans ce test nous avons comparé le délai de traitement de paquets dans notre module du routage par contenu avec le délai introduit par le routage unicast classique. Pour cela nous avons mesuré le temps entre l'entrée d'un paquet dans le routeur et sa sortie par une autre interface. Cela a été fait pour de paquets de tailles différentes. Pour la mesure d'unicast nous avons envoyé des paquets vers l'interface `eth0` du CCR (flèches 1 sur la figure 15.2). Le routage a été configuré pour retransmetre des paquets sur l'interface `eth1`. Pour la mesure du routage par contenu, des paquets ont été envoyés à l'interface `eth0` du CCR (flèche 1) et retransmis deux fois, tout d'abord sur l'interface `eth1` (flèche 2) et puis sur `eth2` (flèche 3).

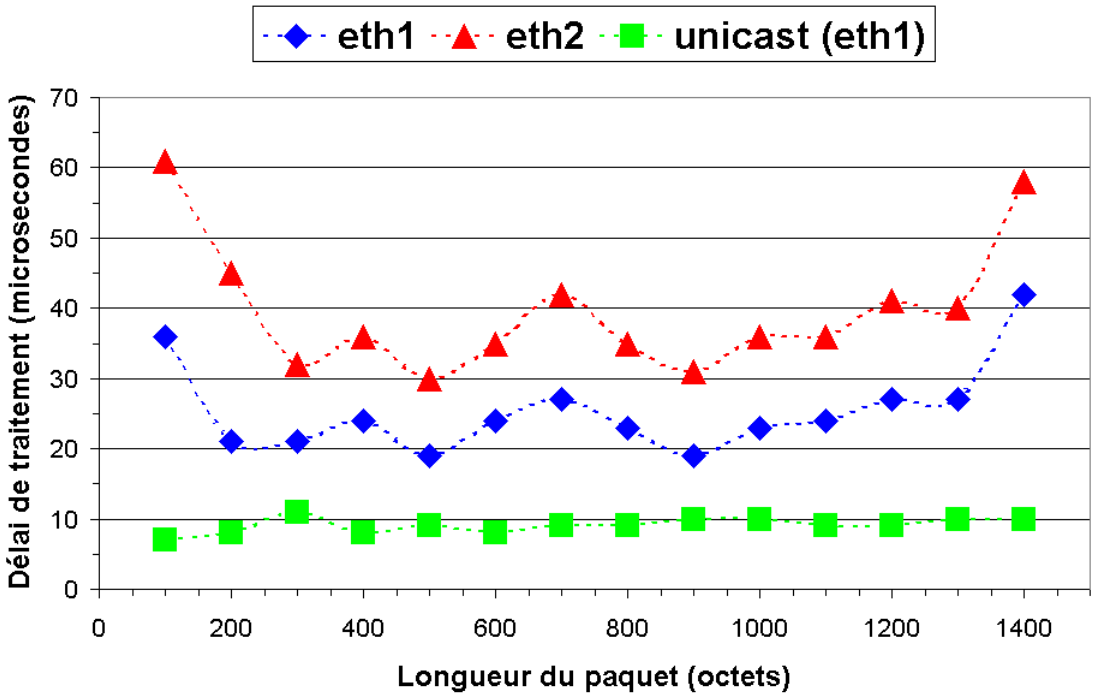

FIG. 15.4 – Délai de traitement introduit par le module du routage par contenu

Sur la figure 15.4 nous voyons les délais de transmission des paquets mesurés sur le routeur. La courbe en bas, en carrés, montre le délai observé en routage unicast. Les deux courbes en haut montrent le délai observé sur les deux interfaces de sortie (respectivement, les losanges pour `eth1` et les triangles pour `eth2`). Le retard observé pour le routage par contenu vient du fait que le paquet reçu doit être copié dans le module et traité pour être

retransmis. Comme il y a deux interfaces sortantes, le module doit faire deux copies du paquet ce qui justifie le retard supplémentaire pour les paquets sortant par l'interface `eth2`.

15.2.2 Débit

Dans ce test, nous avons lancé un flot de données depuis la machine E1. Les CCR1 et CCR2 ont bien configuré leurs tables de routage et ont livré le flot aux machines R1 et R2. Le débit mesuré sur les récepteurs était au niveau de 12 Mb/s. Sur les deux récepteurs nous avons observé le même débit. La dégradation de débit vient de deux faits. Le premier est que, comme dans la section précédente, le module de routage ajoute un retard causé par le traitement du paquet. L'autre, c'est que notre implémentation du prototype n'est pas optimale. L'amélioration de l'implémentation peut niveler cet effet à un certain dégré.

15.2.3 Fonctionnement

Pour observer la création de tables du routage, nous avons simulé la distribution de plusieurs canaux différents avec les IC différents. Dans ce but, la machine E1 envoyait plusieurs paquets (5000), chacun avec un IC différent dans l'adresse source du paquet. Cela simulait la distribution de 5000 canaux différents. Sur les deux CCR, nous avons observé la création des tables du routage. Pour chaque interface configurée comme très haut-débit, il y avait une entrée correctement crée et configurée. Dans le cas d'une inondation excessive du côté E1, la perte de certains paquets UDP a été observée. Comme des paquets n'ont pas été reçus par les CCR, les entrées dans les tables de routage n'ont pas été créées. Néanmoins, pour tous les IC correctement reçus, toutes les interfaces ont été correctement marquées comme INCOMING ou ACTIVE. Dans un cas réel, la perte n'est pas gênante, parce que les paquets suivants d'un canal peuvent aussi provoquer la création d'entrées dans la table du routage.

15.3 Conclusion

Dans ce chapitre, nous avons présenté un premier prototype réalisé ainsi que des expériences dans un environnement de test. Les expériences montrent que l'implémentation offre la fonctionnalité demandée. Sa performance peut être encore améliorer. Des expériences dans des environnements plus grands et plus complexes pourraient apporter une évaluation plus exhaustive.

Chapitre 16

Conclusion sur le transport de distribution de contenu

Dans la deuxième partie de cette thèse nous avons proposé une architecture pour la distribution de contenu. Le contexte de réseaux à très haut débit qui apparaissent dans de nombreuses technologies nous a amené à nous intéresser aux transmissions multipoint en mode "forcé". Nous avons proposé une nouvelle définition de la session multimédia, qui est une version élargie de la transmission de type multicast/diffusion. Notre session intègre le contenu transporté avec un adressage basé sur ce contenu et sa distribution s'effectue grâce à une architecture conçue pour un routage orienté contenu. Notre proposition présente des avantages suivants :

Simplicité : grâce à l'absence du protocole de gestion de l'arbre et de groupes, l'architecture proposée est facile à déployer. Il suffit qu'un opérateur réseau connecte son routeur (CCR ou ICR) à un autre routeur qui participe déjà à la distribution. Des arbres de distribution appropriés seront automatiquement créés.

Passage à l'échelle : le nombre de messages échangés pour gérer les arbres de distribution ne dépend ni du nombre d'opérateurs (et leurs routeurs), ni du nombre d'utilisateurs. Les ressources nécessaires pour gérer les tables de routage dépendent uniquement du nombre de canaux desservis, et non du nombre de machines participant à la distribution.

Nous pouvons regarder quelles sont des limites réelles de notre architecture. Imaginons un réseau du cœur à 2.5 Gb/s, comme par exemple VTHD++. En supposant remplir 10% du débit disponible nous pouvons diffuser 50 canaux avec des flots vidéo MPEG (5 Mb/s par un canal). En imaginant des réseaux futurs à plusieurs dizaines Gb/s nous pouvons desservir des centaines de flots. Nous devons aussi remarquer que dans notre environnement

de test le routage de paquets est moins performant qu'avec l'unicast. Son influence sur les performances globales de l'architecture reste encore à étudier.

Chapitre 17

Conclusion de la thèse et perspectives

Dans cette thèse, nous sommes partis du constat que le progrès technologique dans le domaine des réseaux permet une évolution des protocoles utilisés. Une réflexion sur des concepts de base a été déjà provoquée par la prise en compte de la mobilité il y a quelques années. Aujourd'hui nous montrons que l'arrivée du très haut débit peut nous amèner à changer de nouveau de paradigmes. Nous avons montré qu'il est possible de modifier des hypothèses pour apporter de nouvelles fonctionnalités. Dans ce travail, les concepts d'adressage et de connexion ont été étudiés dans le contexte des réseaux à très haut débit. Les autres concepts du réseau doivent certainement constituer une activité de la recherche future.

Dans la première partie de la thèse nous avons exploré le problème du multi-accès. Nous avons étudié le protocole SCTP et proposé des extensions qui permettent de :

- Bénéficier de toutes les adresses IP sur les deux côtés d'une association. Cela est possible dans le SCTP classique uniquement sous certaines conditions. Dans notre prototype cette possibilité est offerte à chaque machine possédant plusieurs adresses IP.

- Utiliser plusieurs chemins dans le réseau pour transmettre des données entre deux machines. Cela peut offrir un débit plus élevé dans une association en utilisant le débit cumulé sur plusieurs chemins disponibles.

Pour les deux extensions proposées, nous avons élaboré des prototypes appropriés. Ils ont été évaluées par des expériences. Suite aux résultats de ces expériences, nous pensons

que nos travaux ouvrent les perspectives suivantes :

Gestion dynamique d'adresses et de chemins : nous pensons ici au choix dynamique d'adresses pendant une connexion pour choisir le meilleur chemin entre deux extrémités. Le choix dynamique, cela veut dire que pour chaque paquet envoyé, une extrémité peut choisir une nouvelle adresse source (et en même temps une autre interface de sortie) et aussi une nouvelle adresse destination. Le choix peut être basé sur des caractéristiques de chemin entre les deux adresses : la qualité de service (QoS), le débit nominal, le délai aller-retour observé, le nombre de retransmissions, etc. Pour cela, il est nécessaire de proposer un mécanisme d'observation continue de tous les chemins possibles entre les deux extrémités. Une technique d'évaluation d'adresses et de chemins est aussi nécessaire pour pouvoir constater à tout moment quel choix est le meilleur.

Échange de préférences entre les deux extrémités d'une connexion : a priori une extrémité peut recevoir des données sur toutes les adresses disponibles. Elle peut, malgré tout, préférer les recevoir sur une adresse choisie (pour des raisons diverses) si possible. Pour cela, elle doit communiquer cette volonté à l'autre extrémité pour qu'elle utilise l'adresse choisie comme adresse destination de ses paquets. Le choix de l'adresse peut être impératif ou indicatif pour l'autre extrémité et peut varier dans le temps. Cette fonctionnalité exige l'existence d'un protocole d'échange des préférences entre les deux extrémités d'une connexion.

Répartition de charge entre plusieurs chemins : s'il existe plusieurs chemins entre les deux extrémités, nous pouvons les utiliser tous en même temps pour obtenir des conditions de transmission plus favorables. Cela permettrait d'atteindre un débit cumulé plus élevé qu'en utilisant un seul chemin. Pour cela des études sur les techniques de répartition de charge sont nécessaires. Comme chaque chemin offre une caractéristique individuelle et différente, les études doivent couvrir la problèmatique de l'ordre d'arrivée de paquets, la gestion de retransmissions et le contrôle de congestion.

La deuxième partie de cette thèse a été consacrée à la distribution de contenu. Nous avons proposé une architecture de diffusion de contenu qui permet de bénéficier de réseaux à très haut débit. Elle aporte :

– Facilité de deployement : un opérateur qui veut participer à la distribution doit connecter son routeur à un routeur qui fait déjà partie du réseau de cœur. Il doit aussi configurer le lien à très haut débit par lequel il est connecté au réseau de cœur. Les arbres de distribution se configurent automatiquement par l'arrivée des données.

– Passage facile à grande échelle : notre architecture passe très bien à grande échelle

parce que la complexité de gestion de groupes ne change pas avec leur taille. Le nombre de messages échangés entre les machines ne dépend pas du nombre de machines participant à la distribution (routeurs et utilisateurs). Il est limité et aucun message n'est pas diffusé à toutes les machines.

Avec ces caractéristiques, notre proposition dépasse le modèle existant du multicast/diffusion, très complexe et difficile à déployer. Un prototype de routeur réalisé a permis d'évaluer par des expériences le fonctionnement et les performances du routage orienté contenu.

Au-delà de nos travaux, nous voyons plusieurs voies pour continuer. En partant des perspectives à court terme et en allant vers des perspectives à longue terme, nous pouvons les résumer de manière suivante :

Une évaluation exhaustive du prototype : ces travaux ont pour but d'évaluer de manière plus exhaustive le prototype réalisé. Nous pensons ici à des expériences dans un scénario plus large et plus complexe pour comparer pratiquement notre approche avec d'autres approches et montrer ses avantages de passage à grande échelle. Pour évaluer des scénarios plus complexes nous envisageons des expériences de tests réels du type EmuLab [106] [107] ou PlanetLab [108] [109].

Sécurité et fiabilité : ces deux aspects qui ont été mentionnés auparavant devraient être étudiés plus profondément pour offrir l'authentification et l'autorisation des utilisateurs. Des mécanismes de sécurité existants pourraient être adaptés dans notre architecture. En ce qui concerne la fiabilité, des mécanismes appropriés doivent être étudiés pour protéger le contenu contre les pertes des paquets.

Adaptation des applications existantes : notre approche vise des applications qui diffusent du contenu multimédia. Une adaptation des applications existantes, par exemple du type CDN ou *peer-to-peer* permettrait de bénéficier des avantages de notre architecture. Quelques applications devraient être étudiées de ce point de vue et la possibilité de leur adaptation devrait être explorée. Des applications d'autres types peuvent également être considérées, comme par exemple les jeux en réseaux ou les conférences multi-utilisateurs.

Agrégation d'adresses : le modèle d'adressage proposé ne permet pas l'agrégation. Cela peut produire des tables de routage inutilement grandes car pour chaque groupe un routeur crée de nouvelles entrées. Une méthode d'agrégation pourrait permettre d'économiser sur la mémoire de routeurs ainsi qu'améliorer la performance en réduisant la taille des tables de routage.

Agrégation de contenu : dans notre architecture chaque canal est contenu dans une

suite de paquets. Même s'il y a très peu de données à transmettre (moins que le MTU disponible), un paquet est créé. Pour mieux utiliser la bande passante disponible, des petits paquets de canaux différents qui prennent le même trajet peuvent être agrégés et livrés comme un seul paquet. A la fin du trajet commun ils devraient être répartis et continuer leurs trajets individuels. Ce problème nécessite une modification du routage proposé.

Mobilité des utilisateurs : dans des réseaux locaux d'accès, nous pouvons avoir des utilisateurs mobiles qui, en cours de la transmission, se déplacent et de cette façon changent de point d'attachement. Chaque fois qu'un utilisateur se déplace, la méthode de distribution doit garantir la livraison de contenu pour tous les utilisateurs abonnés. Ce type de gestion de distribution reste à étudier.

Troisième partie

Annexes

Annexe A

Abréviations

AC	*Autorité de Certification*
ABNF	*Augmented Backus-Naur Form*
ADSL	*Asynchronous Digital Subscriber Line*
API	*Application Program Interface*
ARQ	*Automatic Repeat reQuest*
BNF	*Backus-Naur Form*
CBT	*Core Based Trees*
CCR	*Core Content Router*
CDN	*Content Distribution Network*
DNS	*Domain Name System*
DVMRP	*Distance Vector Multicast Routing Protocol*
EPCP	*End-Point Control Protocol*
FCS	*Frame Check Sequence*
FEC	*Forward Error Correction*
HIP	*Host Identity Protocol*
HTML	*HyperText Markup Language*
HTTP	*HyperText Transfer Protocol*
IC	*Identificateur de Contenu*
IH	*Identificateur d'Hôte*
ICR	*Intermediate Content Router*
IETF	*Internet Engineering Task Force*
IFG	*InterFrame Gap*
IGMP	*Internet Group Management Protocol*

IGMPv1 *Internet Group Management Protocol, version 1*
IGMPv2 *Internet Group Management Protocol, version 2*
IGMPv3 *Internet Group Management Protocol, version 3*
IP *Internet Protocol*
IPv4 *Internet Protocol, version 4*
IPv6 *Internet Protocol, version 6*
IT *Identificateur de Trafic*
LDAP *Lightweight Directory Access Protocol*
LCR *Local Content Router*
lkSCTP *Linux Kernel Stream Control Transmission Protocol*
NAT *Network Address Translation*
MADCAP *Multicast Address Dynamic Client Allocation Protocol*
MALLOC *Multicast Address Allocation Architecture*
MASC *Multicast Address-Set Claim*
MLD *Multicast Listener Discovery*
MLDv2 *Multicast Listener Discovery, version 2*
MOSPF *Multicast Open Shortest Path First*
MP3 *MPEG-1 Audio Layer III*
MTU *Maximal Transmission Unit*
MSDP *Multicast Source Discovery Protocol*
MSS *Maximum Segment Size*
PSTN *Public Switched Telephone Network*
PIM *Protocol-Independent Multicast*
PIM-DM *Protocol-Independent Multicast - Dense Mode*
PIM-SM *Protocol-Independent Multicast - Sparse Mode*
RPF *Reverse Path Forwarding*
RTSP *Real Time Streaming Protocol*
RTT *Round-Trip Time*
SAP *Session Announcement Protocol*
SDP *Session Description Protocol*
SFD *Start Frame Delimeter*
SIP *Session Initiation Protocol*
SGML *Standard Generalized Markup Language*
SSM *Source-Specific Multicast*

SCTP	*Stream Control Transmission Protocol*
TCP	*Transmission Control Protocol*
TSN	*Transmission Sequence Number*
UDP	*User Datagram Protocol*
UMTS	*Universal Mobile Telecommunications System*
WDM	*Wave Division Multiplexing*
XML	*eXtensible Markup Language*

Annexe B

Le routage orienté contenu

Cette annexe comprend les algorithmes du traitement de messages de contrôle. Chaque fois qu'un message est reçu par un routeur, l'algorithme approprié est réalisé.

B.1 LOCAL_ANNOUNCE

```
packet_incoming_interface = packet.incoming_interface

if ICR_found_in_ROUTING_TABLE()
then
    icr_interface = get_icr_interface()

    if packet_incoming_interface == icr_interface
    then
        forward_to_all_other_interfaces(packet)
        drop(packet)
        finish_processing()
    else
        send_PRUNE_upstream(packet, packet_incoming_interface)
        mark_interface_PRUNEd_for_icr(packet_incoming_interface)
        drop(packet)
        finish_processing()
else
    mark_interface_as_ICR(packet_incoming_interface)
    forward_to_all_other_interfaces(packet)
    finish_processing()
```

B.2 LOCAL_TRAFFIC

```
IC = get_ic(packet)
packet_incoming_interface = packet.incoming_interface

if ACTIVE_routes_exist(IC)
then
    forward_to_all_ACTIVE_interfaces(packet)
    finish_processing()

if no_ACTIVE_routes(IC)
then
    send_LOCALSTOP_packet_upstream(packet_incoming_interface, IC)
    drop(packet)
    finish_processing()
```

B.3 LOCAL_REQUEST

```
IC = get_ic(packet)
packet_incoming_interface = packet.incoming_interface

if packet_incoming_interface == ACTIVE_interface_for_ic(IC)
then
    increase_counter(packet_incoming_interface, IC)
    drop(packet)
    finish_processing()
else
    set_interface_ACTIVE(packet_incoming_interface, IC)
    set_counter_to_one(packet_incoming_interface, IC)

    if any_other_interface_already_ACTIVE(IC)
        drop(packet)
        finish_processing()
    else
        forward_to_upstream_router(packet)
        finish_processing()
```

B.4 LOCAL_STOP

```
IC = get_ic(packet)
packet_incoming_interface = packet.incoming_interface
decrease_counter(packet_incoming_interface, IC)

if get_counter(packet_incoming_interface, IC) == 0
then
    mark_interface_PRUNEd_for_this_IC(packet_incoming_interface)

if ACTIVE_routes_exist(IC)
then
    drop(packet)
    finish_processing()

if no_ACTIVE_routes(IC)
then
    forward_to_upstream_router(packet)
    delete_entries_from_routing_table(IC)
    finish_processing()
```

B.5 CORE_TRAFFIC

```
IC = get_ic(packet)
packet_incoming_interface = packet.incoming_interface

if IC_found_in_ROUTING_TABLE()
then
    ic_INCOMING_interface = get_INCOMING_interface_for_ic(IC)

    if packet_incoming_interface == ic_INCOMING_interface
    then
        if ACTIVE_routes_exist(IC)
            forward_to_all_ACTIVE_interfaces(packet)
            finish_processing()
        if no_ACTIVE_routes(IC)
            send_PRUNE_upstream(packet, packet_incoming_interface)
            mark_interface_PRUNEd_for_this_IC(packet_incoming_interface, IC)
            drop(packet)
            finish_processing()
    else
        send_PRUNE_upstream(packet_incoming_interface)
        mark_interface_PRUNEd_for_this_IC(packet_incoming_interface, IC)
        drop(packet)
        finish_processing()
else
    mark_INCOMING(packet_incoming_interface, IC)
    mark_ACTIVE(all interfaces except the INCOMING and PRUNED, IC)

    if ACTIVE_routes_exist(IC)
    then
        forward_to_all_ACTIVE_interfaces(packet)
        finish_processing()

    if no_ACTIVE_routes(IC)
    then
        send_PRUNE_upstream_for_this_IC(packet_incoming_interface, IC)
        mark_interface_PRUNEd_for_this_IC(packet_incoming_interface, IC)
        drop(packet)
        finish_processing()
```

B.6 CORE_PRUNE

```
IC = get_ic(packet)
packet_incoming_interface = packet.incoming_interface

mark_interface_PRUNEd_for_this_IC(packet_incoming_interface, IC)

if no_ACTIVE_routes(IC)
then
    send_PRUNE_upstream_for_this_IC(packet_incoming_interface, IC)
    delete_entries_from_routing_table(IC)
    drop(packet)
    finish_processing()
```

B.7 CORE_RENEW

```
IC = get_ic(packet)
packet_incoming_interface = packet.incoming_interface

change_PRUNED_to_ACTIVE(packet_incoming_interface, IC)
```

B.8 CORE_FAILURE

```
send_CORERENEW_to_all_PRUNED_interfaces(IC)
forward_to_all_ACTIVE_interfaces(packet, IC)
delete_entries_from_routing_table(IC)
```

Annexe C

Bibliographie

[1] George Gilder. On the Bandwidth of Plenty (Interview). *IEEE Internet Computing*, 1(1) :9–18, 1997.

[2] J. Postel. Internet Protocol. RFC 791 (Standard), September 1981. Updated by RFC 1349.

[3] J. Postel. Transmission Control Protocol. RFC 793 (Standard), September 1981. Updated by RFC 3168.

[4] J. Postel. User Datagram Protocol. RFC 768 (Standard), August 1980.

[5] S. Deering and R. Hinden. Internet Protocol, Version 6 (IPv6) Specification. RFC 2460 (Draft Standard), December 1998.

[6] Content Delivery and Distribution Services. `http://www.web-caching.com/cdns.html`.

[7] Page web du projet VTHD++. `http://www.vthd.org`.

[8] R. Stewart, Q. Xie, K. Morneault, C. Sharp, H. Schwarzbauer, T. Taylor, I. Rytina, M. Kalla, L. Zhang, and V. Paxson. Stream Control Transmission Protocol. RFC 2960 (Proposed Standard), October 2000. Updated by RFC 3309.

[9] P. Ferguson and D. Senie. Network Ingress Filtering : Defeating Denial of Service Attacks which employ IP Source Address Spoofing. RFC 2827 (Best Current Practice), May 2000. Updated by RFC 3704.

[10] J. Lundberg P. Nikander. Homeless Mobile IPv6. draft-nikander-mobileip-homelessv6-01.txt, February 2001. Internet Draft.

[11] Charles E. Perkins and David B. Johnson. Mobility Support in IPv6. In *Mobile Computing and Networking*, pages 27–37, 1996.

[12] D. Johnson, C. Perkins, and J. Arkko. Mobility Support in IPv6. RFC 3775 (Proposed Standard), June 2004.

[13] R. Moskowitz. Host identity payload architecture. draft-moskowitz-hip-arch-02.txt, February 2001. Internet-Draft.

[14] R. Moskowitz. Host identity payload and protocol. draft-ietf-moskowitz-hip-05.txt, November 2001. Internet-Draft.

[15] R. Moskowitz. Host identity payload implementation. draft-moskowitz-hip-impl-01.txt, February 2001. Internet-Draft.

[16] P.V. Mockapetris. Domain names - concepts and facilities. RFC 1034 (Standard), November 1987. Updated by RFCs 1101, 1183, 1348, 1876, 1982, 2065, 2181, 2308, 2535.

[17] P.V. Mockapetris. Domain names - implementation and specification. RFC 1035 (Standard), November 1987. Updated by RFCs 1101, 1183, 1348, 1876, 1982, 1995, 1996, 2065, 2136, 2181, 2137, 2308, 2535, 2845, 3425, 3658.

[18] M. Ishiyama F. Teraoka. LIN6 : A Solution to Mobility and Multi-Homing in IPv6. draft-teraoka-ipng-lin6-01.txt, August 2001. Internet Draft.

[19] M. Py. Multi Homing Aliasing Protocol (MHAP). draft-py-mhap-01a.txt, April 2002. Internet-Draft.

[20] M. Py. Multi Homing Translation Protocol (MHTP). draft-py-multi6-mhtp-00.txt, August 2001. Internet-Draft.

[21] M. Urueña. End-Point Control Protocol. draft-muruenya-epcp-00.txt, November 2002. Internet-Draft.

[22] E. Nordmark and M. Bagnulo. Multihoming L3 Shim Approach. draft-ietf-multi6-l3shim-00.txt, January 2005. Internet-Draft.

[23] G. Huston. Architectural Commentary on Site Multi-homing using Level 3 Shim. draft-huston-l3shim-arch-00.txt, February 2005. Internet-Draft.

[24] R. Stewart, M. Ramalho, and Q. Xie. Stream Control Transmission Protocol (SCTP) Dynamic Address Reconfiguration. draft-ietf-tsvwg-addip-sctp-11.txt, February 2005. Internet-Draft.

[25] R. Stewart, M. Ramalho, Q. Xie, M. Tuexen, and P. Conrad. Stream Control Transmission Protocol (SCTP) Partial Reliability Extension. RFC 3758 (Proposed Standard), May 2004.

[26] Serveur web Apache2 et navigateur web Mozilla avec support pour protocole SCTP. `http://www.sctp.org/download.html`.

[27] Page web du lkSCTP - une implémentation de SCTP pour le noyau Linux. `http://lksctp.sourceforge.net`.

[28] Page web du noyau Linux. `http://www.kernel.org`.

[29] Van Jacobson. Congestion Avoidance and Control. In *ACM SIGCOMM '88*, pages 314–329, Stanford, CA, August 1988.

[30] Page web du *sctpperf* - un outil pour mesurer la performance de SCTP. `http://www-lsr.imag.fr/Les.Personnes/Pawel.Hadam/work/sctpperf/index.html`.

[31] A. Jungmaier and M. Schopp. Sockets API Extensions for Stream Control Transmission Protocol (SCTP).

[32] J. Stone. *Checksums and the Internet*. PhD thesis, Stanford University, August 2001.

[33] P. Deutsch and J-L. Gailly. ZLIB Compressed Data Format Specification version 3.3. RFC 1950 (Informational), May 1996.

[34] G. Castagnoli, S. Braeuer, and M. Herrman. Optimization of cyclic redundancy-check codes with 24 and 32 parity bits. *IEEE Transactions on Communications*, 41(6) :883, 1993. IEEE Transactions on Communications.

[35] J. Stone, R. Stewart, and D. Otis. Stream Control Transmission Protocol (SCTP) Checksum Change. RFC 3309 (Proposed Standard), September 2002.

[36] T. Kubo, S. Kashihara, K. Iida, Y. Kadobayashi, and S. Yamaguchi. Path Management of SCTP to Eliminate Single Point of Failure in Multihoming. Proc. IEEE 5th International Conference on Advanced Communication Technology (ICACT2003), Korea, January 2003.

[37] Page web de NetFilter - un module pour filtrer les paquets dans le noyau Linux. `http://www.netfilter.org`.

[38] Le module ROUTE pour NetFilter. `http://www.netfilter.org/patch-o-matic/pom-extra.html#pom-extra-ROUTE`.

[39] S. Deering and David R. Cheriton. Multicast Routing in Datagram Internetworks and Extended LANs. *ACM Transactions on Computer Systems*, 8(2) :85–111, May 1990.

[40] David R. Cheriton and Stephen E. Deering. Host groups : a multicast extension for datagram internetworks. In *SIGCOMM '85 : Proceedings of the ninth symposium on Data communications*, pages 172–179. ACM Press, 1985.

[41] S.E. Deering. Host extensions for IP multicasting. RFC 1112 (Standard), August 1989. Updated by RFC 2236.

[42] S. E. Deering. *Multicast routing in a datagram internetwork*. PhD thesis, Stanford University, 1992.

[43] Lorenzo Aguilar. Datagram routing for internet multicasting. *Computer Communication Review*, 14(2) :58–63, 1984.

[44] Y. K. Dalal and R. M. Metcalfe. Reverse Path Forwarding of Broadcast Packets. *Communications of the ACM*, 21(12) :1040–1048, December 1978.

[45] D. Waitzman, C. Partridge, and S.E. Deering. Distance Vector Multicast Routing Protocol. RFC 1075 (Experimental), November 1988.

[46] J. Moy. Multicast Extensions to OSPF. RFC 1584 (Proposed Standard), March 1994.

[47] J. Moy. OSPF Version 2. RFC 2328 (Standard), April 1998.

[48] E. W. Dijkstra. A note on two problems in connexion with graphs. In *Numerische Mathematik (1)*, pages 269–271, 1959.

[49] Stephen Deering, Deborah L. Estrin, Dino Farinacci, Van Jacobson, Ching-Gung Liu, and Liming Wei. The PIM architecture for wide-area multicast routing. *IEEE/ACM Transactions on Networking*, 4(2) :153–162, 1996.

[50] A. Adams, J. Nicholas, and W. Siadak. Protocol Independent Multicast - Dense Mode (PIM-DM) : Protocol Specification (Revised). RFC 3973 (Experimental), January 2005.

[51] 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 (Experimental), June 1998.

[52] A. Ballardie. Core Based Trees (CBT version 2) Multicast Routing – Protocol Specification. RFC 2189 (Experimental), September 1997.

[53] A. Ballardie. Core Based Trees (CBT) Multicast Routing Architecture. RFC 2201 (Experimental), September 1997.

[54] D. Thaler, M. Handley, and D. Estrin. The Internet Multicast Address Allocation Architecture. RFC 2908 (Informational), September 2000.

[55] S. Hanna, B. Patel, and M. Shah. Multicast Address Dynamic Client Allocation Protocol (MADCAP). RFC 2730 (Proposed Standard), December 1999.

[56] P. Radoslavov, D. Estrin, R. Govindan, M. Handley, S. Kumar, and D. Thaler. The Multicast Address-Set Claim (MASC) Protocol. RFC 2909 (Experimental), September 2000.

[57] T. Bates, Y. Rekhter, R. Chandra, and D. Katz. Multiprotocol Extensions for BGP-4. RFC 2858 (Proposed Standard), June 2000.

[58] D. Meyer. Administratively Scoped IP Multicast. RFC 2365 (Best Current Practice), July 1998.

[59] Hugh W. Holbrook and David R. Cheriton. IP Multicast Channels : EXPRESS Support for Large-scale Single-source Applications. In *Proceedings of SIGCOMM '99*, pages 65–78, 1999.

[60] H. Holbrook and B. Cain. Source-Specific Multicast for IP. draft-ietf-ssm-arch-06.txt, September 2004. Internet-Draft.

[61] W. Fenner. Internet Group Management Protocol, Version 2. RFC 2236 (Proposed Standard), November 1997. Obsoleted by RFC 3376.

[62] B. Cain, S. Deering, I. Kouvelas, B. Fenner, and A. Thyagarajan. Internet Group Management Protocol, Version 3. RFC 3376 (Proposed Standard), October 2002.

[63] S. Deering, W. Fenner, and B. Haberman. Multicast Listener Discovery (MLD) for IPv6. RFC 2710 (Proposed Standard), October 1999. Updated by RFCs 3590, 3810.

[64] R. Vida and L. Costa. Multicast Listener Discovery Version 2 (MLDv2) for IPv6. RFC 3810 (Proposed Standard), June 2004.

[65] M. Handley and V. Jacobson. SDP : Session Description Protocol. RFC 2327 (Proposed Standard), April 1998. Updated by RFC 3266.

[66] M. Handley, C. Perkins, and E. Whelan. Session Announcement Protocol. RFC 2974 (Experimental), October 2000.

[67] M. Handley, H. Schulzrinne, E. Schooler, and J. Rosenberg. SIP : Session Initiation Protocol. RFC 2543 (Proposed Standard), March 1999. Obsoleted by RFCs 3261, 3262, 3263, 3264, 3265.

[68] A. B. Roach. Session Initiation Protocol (SIP)-Specific Event Notification. RFC 3265 (Proposed Standard), June 2002.

[69] H. Schulzrinne, A. Rao, and R. Lanphier. Real Time Streaming Protocol (RTSP). RFC 2326 (Proposed Standard), April 1998.

[70] R. Fielding, J. Gettys, J. Mogul, H. Frystyk, L. Masinter, P. Leach, and T. Berners-Lee. Hypertext Transfer Protocol – HTTP/1.1. RFC 2616 (Draft Standard), June 1999. Updated by RFC 2817.

[71] K. Zeilenga. Lightweight Directory Access Protocol version 2 (LDAPv2) to Historic Status. RFC 3494 (Informational), March 2003.

[72] R. Braden, L. Zhang, S. Berson, S. Herzog, and S. Jamin. Resource ReSerVation Protocol (RSVP) – Version 1 Functional Specification. RFC 2205 (Proposed Standard), September 1997. Updated by RFCs 2750, 3936.

[73] H. Schulzrinne, S. Casner, R. Frederick, and V. Jacobson. RTP : A Transport Protocol for Real-Time Applications. RFC 3550 (Standard), July 2003.

[74] M. Handley, V. Jacobson, and C. Perkins. SDP : Session Description Protocol. draft-ietf-mmusic-sdp-new-24.txt, February 2005. Internet-Draft.

[75] B. Fenner and D. Meyer. Multicast Source Discovery Protocol (MSDP). RFC 3618 (Experimental), October 2003.

[76] C. Diot, B. Levine, B. Lyles, H. Kassem, and D. Balensiefen. Deployment Issues for the IP Multicast Service and Architecture. *IEEE Network magazine special issue on Multicasting*, February 2000.

[77] Kevin C. Almeroth. The evolution of multicast : From the MBone to inter-domain multicast to Internet2 deployment. *IEEE Network*, 14 :10–20, feb 2000.

[78] D. Crocker. Standard for the format of ARPA Internet text messages. RFC 822 (Standard), August 1982. Obsoleted by RFC 2822, updated by RFCs 1123, 1138, 1148, 1327, 2156.

[79] D. Crocker and P. Overell. Augmented BNF for Syntax Specifications : ABNF. RFC 2234 (Proposed Standard), November 1997.

[80] La spécification du langage XML. `http://www.w3.org/TR/2004/REC-xml-20040204/`.

[81] International Organization for Standardization. *ISO 8879 :1986 : Information processing — Text and office systems — Standard Generalized Markup Language (SGML).* International Organization for Standardization, Geneva, Switzerland, August 1986.

[82] R. Hinden and S. Deering. IP Version 6 Addressing Architecture. RFC 2373 (Proposed Standard), July 1998. Obsoleted by RFC 3513.

[83] R. Rivest. The MD5 Message-Digest Algorithm . RFC 1321 (Informational), April 1992.

[84] R. Hinden and S. Deering. Internet Protocol Version 6 (IPv6) Addressing Architecture. RFC 3513 (Proposed Standard), April 2003.

[85] B. Haberman. Allocation Guidelines for IPv6 Multicast Addresses. RFC 3307 (Proposed Standard), August 2002.

[86] Tommaso Cucinotta, Gabriele Cecchetti, and Gianluca Ferraro. Adopting Redundancy Techniques for Multicast Stream Authentication. In *9th IEEE International Workshop on Future Trends of Distributed Computing Systems*, pages 183–189, San Juan, Puerto Rico, May 2003.

[87] Jung Min Park, Edwin K. P. Chong, and Howard Jay Siegel. Efficient Multicast Packet Authentication Using Signature Amortization. In *SP '02 : Proceedings of the 2002 IEEE Symposium on Security and Privacy*, page 227, Oakland, CA, May 2002. IEEE Computer Society.

[88] Philippe Golle and Nagendra Modadugu. Authenticating Streamed Data in the Presence of Random Packet Loss. In *NDSS '01 : Proceedings of the eighth Annual Symposium on Network and Distributed System Security*, San Diego, CA, February 2001.

[89] Data Encryption Standard. Federal Information Processing Standards Publication (FIPS PUB) 46-1, NIST, 1977-1978.

[90] Data Encryption Standard (DES). Federal Information Processing Standards Publication (FIPS PUB) 46-3, NIST, 1999.

[91] Data Encryption Algorithm. Standard ANSI X3.92-1981, American National Standards Institute, 1980.

[92] R.L. Rivest, A. Shamir, and L. Adleman. A method for obtaining digital signatures and public-key cryptosystems. Communications of the ACM, February 1978.

[93] Page web de OpenSSH. `http://www.openssh.org`.

[94] G. Fairhurst and L. Wood. Advice to link designers on link Automatic Repeat reQuest (ARQ). RFC 3366 (Best Current Practice), August 2002.

[95] M. Luby, L. Vicisano, J. Gemmell, L. Rizzo, M. Handley, and J. Crowcroft. The Use of Forward Error Correction (FEC) in Reliable Multicast. RFC 3453 (Informational), December 2002.

[96] M. Luby and L. Vicisano. Compact Forward Error Correction (FEC) Schemes. RFC 3695 (Experimental), February 2004.

[97] M. Luby, L. Vicisano, J. Gemmell, L. Rizzo, M. Handley, and J. Crowcroft. Forward Error Correction (FEC) Building Block. RFC 3452 (Experimental), December 2002.

[98] Page web du groupe IETF Reliable Multicast Transport (rmt). `http://www.ietf.org/html.charters/rmt-charter.html`.

[99] Jianfei Cai and Chang Wen Chen. FEC-Based Video Streaming over Packet Loss Networks with Pre-Interleaving. In *International Symposium on Information Technology*, pages 10–14, Las Vegas, NV, April 2001.

[100] Shirish Karande and Hayder Radha. Rate-Constrained Adaptive FEC for Video over Erasure Channels with Memory. In *IEEE International Conference on Image Processing (ICIP)*, Singapore, October 2004.

[101] Jean-Chrysostom Bolot and Thierry Turletti. Adaptive error control for packet video in the Internet. In *IEEE International Conference on Image Processing 1996*, Lausanne, Switzerland, September 1996.

[102] Jean-Chrysostom Bolot, Sacha Fosse-Parisis, and Don Towsley. Adaptive FEC-Based error control for Internet Telephony. In *Proceedings of IEEE INFOCOM 1999*, New York, NY, March 1999.

[103] J. Peltotalo, S. Peltotalo, and V. Roca. Simple XOR, Reed-Solomon, and Parity Check Matrix-based FEC Schemes. draft-peltotalo-rmt-bb-fec-supp-xor-pcm-rs-00.txt, June 2004. Internet-Draft.

[104] V. Roca and C. Neumann. Design, Evaluation and Comparison of Four Large Block FEC Codecs, LDPC, LDGM, LDGM Staircase and LDGM Triangle, plus a Reed-Solomon Small Block FEC Codec. Technical Report RR-5225, INRIA, June 2004.

[105] Page web de SENDIP. `http://www.earth.li/projectpurple/progs/sendip.html`.

[106] Brian White, Jay Lepreau, Leigh Stoller, Robert Ricci, Shashi Guruprasad, Mac Newbold, Mike Hibler, Chad Barb, and Abhijeet Joglekar. An Integrated Experimental Environment for Distributed Systems and Networks. In *Proceedings of 5th Symposium on Operating Systems Design and Implementation (OSDI '02)*, pages 255–270, Boston, MA, December 2002. USENIXASSOC.

[107] Page web du EmuLab - un environnement d'émulation réseau. `http://www.emulab.com`.

[108] A. Bavier, M. Bowman, B. Chun, D. Culler, S. Karlin, S. Muir, L. Peterson, T. Roscoe, T. Spalink, and M. Wawrzoniak. Operating System Support for Planetary-Scale Network Services. In *Proceedings of the 1st Symposium on Network System Design and Implementation (NSDI '04)*, pages 253–266, San Francisco, CA, March 2004.

[109] Page web du PlanetLab - un environnement de tests réseau à l'échelle globale. `http://www.planetlab.com`.

Transports nouvelle génération dans les réseaux à très haut débit

Résumé

Cette thèse a été motivée par le développement des réseaux à très haut débit (au-delà de 1 Gb/s). Nous avons étudié comment ce type de réseaux peut changer les concepts et les protocoles utilisés actuellement. Nous avons considéré deux problèmes : le premier est le multihoming : la possibilité pour un hôte de bénéficier de plusieurs connexions simultanées aux fournisseurs d'accès, et le deuxième la distribution de contenu. Nous avons étudié le nouveau protocole de niveau transport SCTP et proposé des extensions qui permettent d'augmenter les performances et la fiabilité de communication grâce au multi-accès. Le protocole SCTP et les extensions proposées ont été testés et validés sur le réseau à très haut débit VTHD++. Pour le deuxième problème, nous avons conçu et prototypé un protocole de diffusion de contenu basé sur la notion d'inondation. Grâce au routage par contenu proposé pour le protocole, le contenu peut être livré aux consommateurs sans connaître leur localisation.

Mots clés : transport, réseau à très haut débit, multi-accès, SCTP, diffusion de contenu.

New generation transport in high-speed networks

Abstract

This thesis was motivated by the development of high speed networks (above 1 Gb/s). We have studied how this kind of networks may change the current network concepts and protocols. We have considered two aspects: the first : multihoming - a host with several connections to network access providers, and the second : content distribution. We have studied the SCTP protocol and proposed extensions of multihoming for increased performance and reliability. The SCTP protocol and the proposed extensions were tested and validated on the VTHD++ high speed network. For the second aspect, we have designed and implemented a content distribution protocol based on the concept of flooding and content routing. In the proposed protocol the content is delivered to the users without knowing their localisation.

Keywords : transport, high speed networks, multihoming, SCTP, content distribution.

Discipline : Informatique, Systèmes et Communications
Laboratoire : LSR-IMAG - équipe Drakkar
BP 72, 38402 Saint Martin d'Hères Cedex, France