Architecture réseau pour le projet DIS/ATM : **RSVP Switching**, une solution pour un service multipoint avec QoS en environnement IPv6 sur ATM

Olivier Fourmaux   Vincent Roca   Christophe Deleuze
Serge Fdida

Laboratoire d'Informatique de Paris 6 (LIP6)
Université Pierre et Marie Curie / CNRS
4, place Jussieu 75252 PARIS Cedex 05 FRANCE
Tél : (+33) 1 44 27 71 26
Fax : (+33) 1 44 27 62 86
Mél : `Olivier.Fourmaux@lip6.fr`
WWW : `http://www.lip6.fr/rp/~fourmaux/dis`

Septembre 1998

Table des matières

1 Introduction **5**
1.1 Situation dans DIS/ATM 5
1.1.1 Les applications DIS 5
1.1.2 Intégration du DIS avec RSVP 5
1.2 Solution IP sur ATM avec de la QoS 6
1.3 Plan du document 6

2 Commutation IP et QoS **7**
2.1 Introduction à la Commutation de Labels 7
2.2 Choix d'une solution 7
2.3 Les solutions RSVP avec la Commutation IP 8
2.3.1 IP Switching et QoS 8
2.3.2 IS-IPSO et QoS 9
2.3.3 MPLS et QoS 9
2.3.4 QSR et QoS 10

3 RSVP Switching **11**
3.1 La proposition RSVP Switching 11
3.2 L'intégration du service multipoint de RSVP 13
3.2.1 Cut-through 13
3.2.2 Filtrage 16
3.2.3 Agrégation 19
3.3 Spécialisation du modèle théorique RSVP Switching 21
3.3.1 IPv6 et RSVP Switching 21
3.3.2 ATM et RSVP Switching 23

4 Classes de services **26**
4.1 L'architecture IntServ/RSVP 26
4.2 Caractérisation des flots 27
4.3 Mécanismes 28
4.4 Le cas du *Guaranteed Service* 29
4.5 Une implémentation du *Controlled Load* 31
4.5.1 Motivation 31

4.5.2 Réalisation 32
4.6 L'ordonnanceur utilisé 34
4.7 Conclusions sur les classes de services 35

5 Implémentation **36**
5.1 Contexte d'implémentation et conséquences : émulation du RSVP switching 36
5.1.1 Le système Linux 36
5.1.2 Émulation du RSVP Switching 38
5.2 Modifications de RSVP 38
5.2.1 La souche RSVP et le module de gestion de ressources 38
5.2.2 Ajout de la gestion des labels dans RSVP 39
5.2.3 Modification de l'interface avec les services noyaux . . 41
5.3 La souche IPV6 de la DRET 42
5.4 Le gestionnaire Linux-ATM de l'EPFL 42
5.5 La couche émulation de la commutation 43
5.5.1 Organisation générale 43
5.5.2 Labels utilisés 43
5.5.3 Table de commutation 44
5.5.4 Mécanismes d'ordonnancement 44
5.5.5 Création d'une nouvelle famille d'adresses et ioctls . . 44
5.6 Configuration de la plate-forme ATM 45

6 Conclusion **47**

Bibliographie **47**

Table des figures

1.1 Architecture DIS avec QoS 6

3.1 Chemin de routage initial 14
3.2 Création du chemin avec label 14
3.3 Création d'un *cut-through* unicast 15
3.4 Création d'un cut-through multicast 16
3.5 Chemin initial routé pour le filtrage 17
3.6 Association des labels pour le filtrage 17
3.7 Création du *cut-through* pour le filtrage 18
3.8 Chemin initial routé pour l'agrégation 20
3.9 Association des labels pour l'agrégation 20
3.10 Création du *cut-through* pour l'agrégation 21
3.11 Entête Ipv6 22
3.12 Architecture d'un RSVP Switch 24

4.1 Les 3 états de l'architecture IntServ 27
4.2 Caractérisation d'un flot 28
4.3 Enveloppes du flot et du service fourni 31
4.4 Flot vidéo MPEG : R en fonction de M 32
4.5 Flot vidéo MPEG : R en fonction de D 33
4.6 Bande passante utilisée et réservée par les flots GS et BR . . 34
4.7 Mécanisme d'ordonnancement 35

5.1 Vue d'ensemble 37
5.2 Objet `LABEL` 39
5.3 Émulation de la commutation, vue générale 43
5.4 Configuration de la plate-forme ATM 46

Liste des tableaux

3.1 Comparaison des niveau routage et commutation 12

4.1 Les classes de service Internet 26
4.2 Les classes de service Internet 27

Chapitre 1

Introduction

Ce rapport décrit une proposition d'architecture réseau fournissant un service multipoint-à-multipoint avec QoS (*Quality of Service*). L'environnement réseau dans lequel on se place est IP version 6 avec le protocole de signalisation RSVP (*Ressource ReSerVation Protocol*) et les services de l'IntServ (*Integrated Services*). L'infrastructure support est basée sur la technologie ATM (*Asynchronous Transfer Mode*) qui nous permet d'utiliser des liens à longue distance et d'atteindre des performances élevées.

1.1 Situation dans DIS/ATM

Ce rapport présente la contribution du LIP6 au projet DIS/ATM.

1.1.1 Les applications DIS

Le projet DIS/ATM consiste en l'expérimentation d'une série d'applications, utilisant le protocole DIS (*Distributed Interactive Simulation*) fournies par Dassault Électronique [Dul97] dans un environnement IPv6 sur réseau ATM.

1.1.2 Intégration du DIS avec RSVP

Les applications DIS ne sont pas prévues initialement pour spécifier leurs besoins en terme de QoS. Des travaux ont été réalisés au sein du LAAS pour caractériser le trafic DIS et proposer une architecture étendue avec gestion de la QoS. La QoS requise est indiquée par le protocole RSVP [BZB+97] avec des paramètres spécifiés par le groupe de travail IETF IntServ [Wro97b, SPG97, Wro97a].

L'architecture proposée par le LAAS intègre un module intermédiaire permettant d'interfacer les applications DIS avec RSVP [CDD+98]. La Figure 1.1 présente ce module, appelé RSVM (*Ressource reSerVation Manager*). Celui-ci communique directement avec les applications DIS (DISA), le

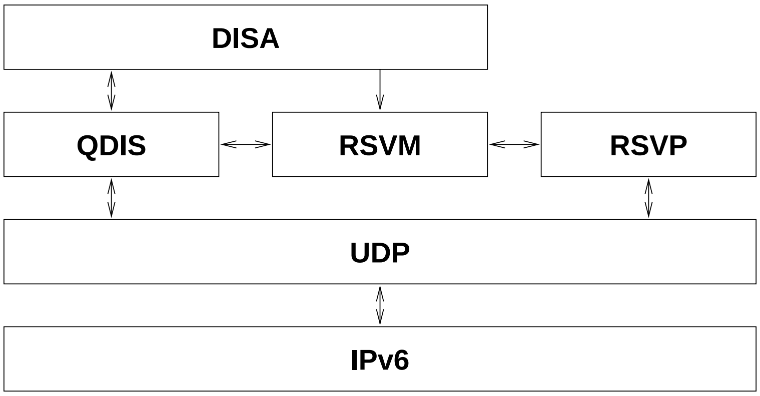


FIG. 1.1: Architecture DIS avec QoS

protocole DIS agrémenté de la QoS (QDIS) et RSVP. Le tout repose sur le service UDP/IPv6.

Notre travail supporte donc l'architecture précédente en fournissant un service IPv6 avec RSVP dans un environnement ATM.

1.2 Solution IP sur ATM avec de la QoS

Les solutions usuelles utilisées pour supporter IP dans un environnement ATM sont CLIP (*Classical IP over ATM)* [LH98] et LANE (*LAN Emulation*) [LAN95]. Toutes deux reposent sur le mode orienté connexion de la signalisation ATM Forum et sont peu adaptées au support de RSVP. Le modèle utilisé par IP/RSVP met en jeux des connexions multipoint-à-multipoint avec QoS renégociable et hétérogène sur chaque branche de l'arbre de distribution alors que le modèle RSVP, avec CLIP ou LANE, met en ouvre des connections limitées aux capacités de l'ATM, point-à-multipoint avec QoS homogène et non renégociable (voir [FD97]).

Les solutions CLIP et LANE éliminées, une solution récente s'est offerte à nous : la Commutation de Labels (ou **Commutation IP sur ATM** dans notre cas). Cette solution possède les propriétés nécessaires pour fournir le service requis par l'architecture du LAAS de manière efficace.

1.3 Plan du document

Dans un premier temps, nous présenterons la technologie que nous avons choisie pour utiliser IP sur ATM, puis nous détaillerons l'adaptation de la QoS (RSVP et service IntServ) dans celle-ci. Nous développons ensuite les classes de services gérées. Nous terminons par la description de l'implémentation réalisée et les expérimentations envisagées.

Chapitre 2

Commutation IP et QoS

Une nouvelle approche d'IP sur ATM permet d'éviter le mode connecté d'ATM employé par CLIP et LANE. Cette approche, appelée “Commutation IP”, a pour but de simplifier l'intégration des deux technologies tout en profitant au mieux des performances de la commutation ATM et des qualités de la signalisation IP.

2.1 Introduction à la Commutation de Labels

La Commutation IP recouvre de nombreuses solutions qui peuvent ne pas être spécifiques à IP et ATM, mais qui sont basées sur les mêmes principes : **échange de label** (*label swapping*) dans les nœuds intermédiaires du réseau et **commutation de paquets** (*paquet switching*) basé sur l'association rapide de labels de manière à éviter le chemin de routage habituel (utilisation d'un chemin direct au niveau 2, ou *cut-through*, dans les nœuds). Le tout forme l'approche générique appelée **“Commutation par Label”** (*Label Switching*).

2.2 Choix d'une solution

Les solutions basées sur la Commutation par Label se répartissent dans deux catégories principales : la première est dite “orientée flot” (*flow driven*) et comprend les solutions qui se basent sur le trafic de données local pour décider de l'utilisation de labels ; la seconde est dite “orientée topologie” (*topology driven*) et regroupe les solutions basées sur une distribution systématique des labels selon la signalisation de routage. Pour fournir des garanties de QoS, un état global du réseau doit être maintenu. RSVP est adapté pour cette tâche en environnement non connecté. En conséquence, nous l'introduisons dans l'approche de la Commutation par Label, créant ainsi une troisième catégorie de solutions basées sur l'association des labels avec les flots et les sessions RSVP. Cette approche est dite “orientée session” (*session driven*) et permet la gestion de la QoS.

RSVP peut être introduit dans les deux premières catégories (orientées flot ou topologie). Il produit les solutions “orientées session” suivantes :

- L’introduction de RSVP dans l’approche “orientée flot” est naturelle, car cette approche est conçue dans l’optique d’identifier les flots localement après observation du trafic. Avec RSVP, l’approche utilise aussi les flots mais ceux-ci sont identifiés à partir de la signalisation RSVP.
- L’approche “orientée topologie”, initialement présentée pour le trafic unicast et basée sur les informations de routage, à été étendue pour tenir compte de tout type de signalisation, ce qui en fait une approche “orientée signalisation” (*control driven*). Dans ce cadre, RSVP est l’un des protocoles de signalisation pour le transport des labels, au même titre que les signalisations de routage et du multicast.

Même si ces deux approches “orientées session” se comportent sensiblement de la même manière, toutes deux utilisant la signalisation RSVP pour transporter des informations sur les labels associés au flots, on peut noter que la gestion de RSVP s’intègre naturellement dans l’approche “orientée flot”. Avec l’approche “orientée signalisation”, chaque protocole est intégré séparément ce qui produit un modèle plus disparate.

Des différences importantes existent aussi au niveau de la gestion du multicast, les approches “orientées flot” l’intégrant simplement au même titre que l’unicast.

C’est l’approche **“orientée flot”** dans laquelle nous avons décidé d’intégrer le multicast IP avec QoS, en y adjoignant une architecture RSVP. Nous l’avons choisie pour sa plus grande cohérence par rapport aux autres solutions. Nous nous sommes basés pour cela sur la solution **IP Switching** pour laquelle ce genre d’extensions n’existe pas et avons appelé notre proposition **“RSVP Switching”**.

2.3 Les solutions RSVP avec la Commutation IP

Dans cette section nous présentons les différentes solutions intégrant de la QoS dans le contexte de la Commutation IP, ou plus généralement, de la Commutation par Label.

2.3.1 IP Switching et QoS

La solution IP Switching, basée sur l’approche “orientée flot”, repose sur le principe de localité pour respecter le principe d’absence d’état de l’Internet. Dans les protocoles de gestion du commutateur [NEH$^+$98] proposés, l’introduction de niveaux de services différents est prévue, allant des débits fixes au WRR (*Weighted Round Robin*) en passant par les niveaux de priorité, mais uniquement pour une utilisation statique. Les performances annoncées (gestion de plus de 100.000 files d'attente au niveau du commutateur) indiquent que le matériel est à la hauteur d'une intégration de la QoS. L'introduction de RSVP dans IP Switching permettrait de combler l'absence d'architecture pour réaliser une gestion de la QoS dynamique. C'est une étude de ce type d'intégration que nous proposons de faire dans la suite du chapitre.

2.3.2 IS-IPSO et QoS

IS-ISPSO [AAOS98] est aussi une solution basée sur l'approche "orientée flot". Elle s'appuie sur la proposition IPSOFACTO et propose l'intégration de RSVP et des IIS (*Internet Integrated Services*) pour fournir le service IP multicast avec QoS. Ce modèle suit une approche très similaire à la notre, mais se restreint aux limitations de la technologie ATM, alors que nous étudions les problèmes liés à ces limitations et à l'utilisation des fonctionnalités RSVP au niveau 2 en général. Nous proposons dans la suite une méthode pour dépasser le modèle multipoint ATM et gérer les communications multipoint-à-multipoint avec QoS proposées par RSVP, avec intégration des mécanismes d'agrégation de celui-ci.

2.3.3 MPLS et QoS

La proposition MPLS est issue de deux solutions, qui ont chacune envisagé l'intégration de RSVP. Pour le Tag Switching, le document *"Tag Switching with RSVP"* [BR96] présente la manière d'associer les tags aux flots RSVP et explique comment distribuer ces tags dans la signalisation RSVP. C'est l'une des premières propositions à suggérer une extension des messages RSVP pour transporter des labels. Le groupe ARIS a proposé le document *"Soft State Switching"* [VSB97]. La réunion des travaux des deux groupes nous fournit la proposition générique présentée par MPLS : *"Use of Label Switching With RSVP"* [DRV+98].

Les propositions précédentes reposent sur l'approche "orientée signalisation" et intègrent le trafic RSVP et sa signalisation, de la même manière que les trafics IP unicast et multicast. L'attribution des labels se fait sur la base d'un label par flot RSVP[1]. Le transport des labels utilise la possibilité de *piggybacking* sur la signalisation RSVP, voire de l'intégration directe au protocole RSVP des labels, car un nouvel objet est défini dans RSVP pour leur transport. Cet objet peut se trouver dans les messages PATH ou RESV, car MPLS supporte les diffusions de labels dans les deux sens. Une utilisation de l'agrégation est proposée selon les capacités du commutateur.

[1]Un flot RSVP est identifié par un émetteur (une adresse source et optionnellement un port) et une session (un protocole, une adresse destination et optionnellement un port).

2.3.4 QSR et QoS

Notre énumération ne saurait être complète sans la proposition QSR (*QoS capable Switch-Router*) [BBD+97]. Celle-ci propose la définition et l'implémentation d'un modèle focalisé sur l'intégration des flots RSVP dans un environnement IP sur ATM. La base de l'approche QSR est de commuter les flots avec QoS, le reste du trafic étant routé de manière standard. Les concepts d'extension du protocole RSVP pour le transport des labels et l'utilisation de celui-ci pour le maintient des chemins commutés est déjà présent. La solution proposée repose sur une implémentation détaillée et développe particulièrement l'architecture matérielle et la gestion de ressource.

Chapitre 3

RSVP Switching

Nous adoptons dans la suite une approche de type "Commutation par Label" dans laquelle nous adaptons le protocole RSVP. Cette approche nous permet, outre un gain en performance significatif, une intégration simplifiée d'IP sur ATM évitant le modèle avec serveur. La solution que nous proposons s'appelle **"RSVP Switching"** et met en œuvre les mécanismes nécessaires au fonctionnement du modèle multipoint-à-multipoint hétérogène et dynamique du service IP multicast avec QoS dans un environnement commuté. Nous spécialisons ensuite notre approche pour un usage avec IP sur ATM.

Ce chapitre débute par une description des solutions ayant adopté une approche similaire. Nous présentons ensuite de manière générale notre solution "RSVP Switching" et proposons une approche générique pour gérer les fonctionnalités du niveau de la couche réseau dans un environnement commuté. Nous développons particulièrement l'adaptation des mécanismes d'agrégation et de filtrage de RSVP. Nous terminons en proposant une spécialisation de notre solution basée sur IPv6 en environnement ATM.

3.1 La proposition RSVP Switching

Notre but est de fournir le service IP multicast avec QoS en utilisant RSVP au sein d'une architecture basée sur la Commutation par Label. Nous proposons donc une solution générique que nous spécialiserons à IP et à ATM dans la suite. La base de notre réflexion est la solution "orientée flot" **IP Switching** où les décisions locales de commutation sont guidées par la signalisation RSVP plutôt que par le trafic observé. Nous utilisons donc une approche "orientée session" où RSVP est utilisé pour la gestion des réservations qui sont associées à la création des chemins commutés.

Nous appelons notre solution **RSVP Switching** car nous ne commutons que les flots associés à la QoS. Le trafic *Best Effort* est simplement routé, mais des mécanismes identiques à ceux de IP Switching peuvent aussi être mis en œuvre. Une limitation des solutions basées sur l'approche "orientée

	Niveau routage	**Niveau commutation**
État	Dynamique (Soft State)	Dynamique (Soft State)
QoS Multipoint	Hétérogène (filtrage)	Homogène (matrice de commutation)
Modèle Multipoint	N-vers-M (agrégation)	1-vers-M (matrice de commutation)

TAB. 3.1: Comparaison des niveau routage et commutation

flot" est que l'efficacité dépend de la proportion de données commutées. Avec une approche basée sur RSVP, ce problème disparaît car tout le trafic RSVP est commuté.

Nous pensons que le choix de l'approche "orientée flot" est plus efficace pour son intégration naturelle des trafics unicast et multicast. De même, conserver les données avec la signalisation au sein du parcours routé usuel avant de procéder à la commutation permet une utilisation transparente de RSVP. L'utilisation du multicast au niveau commutation est prévue avec les flots RSVP. Les problèmes qui lui sont liés seront résolus ensembles. Le trafic IP multicast *Best Effort* est supposé routé. Si une commutation de type IP Switching est envisagée pour ce trafic, les problèmes non résolus tels que l'agrégation sont à prendre en compte.

Au niveau de la commutation, la matrice nous fournit localement des connexions point-à-multipoint avec une QoS homogène alors qu'au niveau du routage, le modèle multipoint de RSVP supporte des connections multipoint-à-multipoint hétérogènes. La Table 3.1 présente une comparaison des services fournis par les deux modèles au niveau d'un nœud. Nous supposons que la technologie de commutation a les mêmes limitations que celle d'ATM.

La ligne "*État*" de la Table 3.1 décrit la manière dont les informations sont conservées dans un élément du réseau. Les deux modèles s'appuient sur un état dynamique. Au niveau routage, le *Soft State* de RSVP est utilisé pour maintenir les réservations dans le réseau. Au niveau commutation, avec IP Switching, un mécanisme équivalent est mis en œuvre pour la distribution des labels à travers IFMP. Dans notre solution, nous utilisons directement RSVP à la place d'un protocole de distribution de label spécifique. Nous réutilisons ainsi le *Soft State* de RSVP et lui assignons la double fonction de gestion des réservations et des informations de commutation.

La ligne "*QoS Multipoint*" de la Table 3.1 précise si chaque port de sortie peut avoir une QoS différente. Au niveau routage, chaque destination peut choisir une QoS indépendamment. Cela provient du modèle orienté récepteur de RSVP où chaque récepteur choisit sa réservation indépendamment. Au niveau commutation, une simple duplication est effectuée, ne permettant qu'une QoS homogène sur les branches de l'arbre de distribution. La fonctionnalité de filtrage utilisée au niveau routage pour fournir l'hétérogénéité n'est pas disponible au niveau commutation, créant ainsi un premier problème d'adaptation de RSVP.

La ligne "*Modèle Multipoint*" de la Table 3.1 présente le type de connexions multipoints disponible. Au niveau routage, IP multicast permet un modèle *N-vers-M*. RSVP utilise ce modèle et propose d'y ajouter différents types d'agrégation pour la gestion de la QoS. Cette fonctionnalité est, de même que le filtrage, non disponible au niveau de la commutation. Si rien n'est prévu au niveau commutation, l'agrégation de flots résulte en un multiplexage des données, qui dans le cas d'une technologie basée sur des cellules, entraîne la perte des informations.

La mise en œuvre des différents mécanismes nécessaires à la réalisation des fonctionnalités RSVP au niveau commutation dans **RSVP Switching** est présentée dans la suite.

3.2 L'intégration du service multipoint de RSVP

Dans cette section, nous présentons les mécanismes de commutation mis en œuvre pour les flots RSVP, puis nous étudions particulièrement l'intégration de deux fonctionnalités qui posent des difficultés : le filtrage et l'agrégation.

3.2.1 Cut-through

Le principe de base de la commutation par label est la création de *cut-through*, c'est-à-dire la commutation au lieu du routage usuel au niveau d'un nœud. Nous décrivons dans cette section la mise en place d'un *cut-through* au niveau d'un élément mettant en œuvre la solution RSVP Switching appelé un **RSVP Switch**. La Figure 3.1 présente un chemin de routage (*Routed path*) entre trois RSVP Switches pour la session **S** (le trafic remonte dans la couche 3 pour les trois éléments). Avant et après la zone définie par ces trois éléments, il peut y avoir d'autres RSVP Switches, des réseaux IP avec RSVP ou des hôtes directement connectés. Sur la figure, nous avons décomposé dans chaque RSVP Switch la couche réseau où se situent les fonctionnalités de routage avec RSVP (*Routing*) et la couche liaison où se trouve le moteur de commutation (*Switching*). Le trafic de signalisation est représenté par des flèches bidirectionelles blanches (*Signaling traffic*) et celui des données par des flèches unidirectionelles noires (*Data traffic*).

Après le passage du message PATH, les réservations se mettent en place. Les données sont séparées de la signalisation. Elles quittent le chemin routé par défaut pour suivre un chemin avec label représenté au niveau commutation. Les informations relatives aux labels sont transmisent à travers les messages RESV, dans un nouvel objet spécifique ajouté à la signalisation RSVP

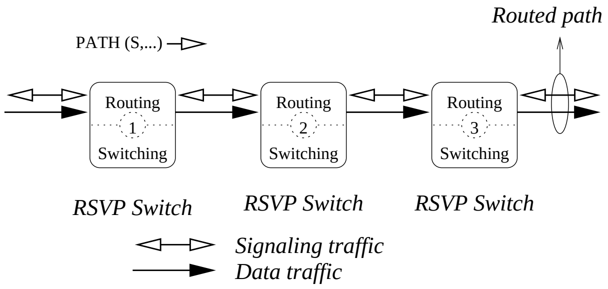
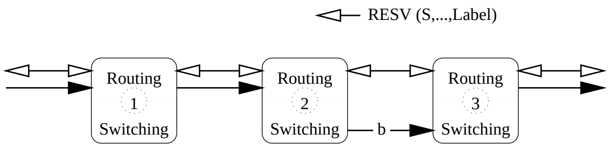


FIG. 3.1: Chemin de routage initial



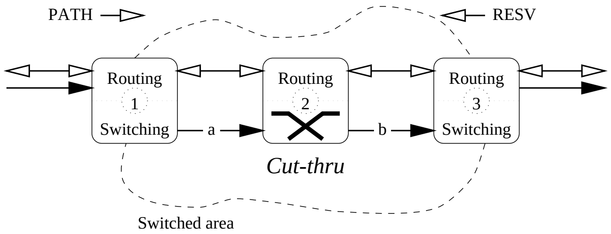
Informations dans le nœud 2					
Table des labels			*Table des* cut-through		
Session	Label	Port	Session	De	Vers
S	“b”	(3)out			

FIG. 3.2: Création du chemin avec label

dans ce but. Nous appelons cet objet “`LABEL`”. Après avoir échangé des messages avec l’information Label (contenue dans l’objet `LABEL`), nous pouvons créer un lien virtuel spécifique étiqueté grâce au label. Des réservations distinctes correspondent à des liens différents.

Dans la Figure 3.2, un message RESV est propagé du récepteur vers l’émetteur. Le message RESV contient l’objet `LABEL` pour étiqueter le lien spécifique. Dans la partie gauche du dessin, entre les RSVP Switch 1 et 2, les données sont encore routées. Dans la partie droite, une réservation a été réalisée. La liaison correspondante est établie et les données, même si elles restent routées, appartiennent à un lien virtuel différent de celui du trafic routé. Les flots avec labels attendent pour l’établissement d’un *cut-through*. Le nouveau lien est représenté par une flèche au niveau de la couche liaison avec le label “b”. Le label est local au lien considéré et change de liens en liens.

Dans la figure 3.3, un label “a” pour la session **S** a été attribué entre les RSVP Switches 1 et 2. Un *cut-through* peut être réalisé. Dans les tableaux associés aux figures, nous supposons qu’un lien ou un port du nœud N est référencé par la notation (N). Dans la table de commutation (ou “*Table des* cut-through”) du RSVP Switch 2, les données arrivent du port d’entrée (1) avec le label “a” et sont commutées vers le port de sortie (3) avec le label



Informations dans le nœud 2					
Table des labels			*Table des* cut-through		
Session	Label	Port	Session	De	Vers
S	"b"	(3)out	**S**	"a" (1)	"b" (3)
S	"a"	(1)in			

FIG. 3.3: Création d'un *cut-through* unicast

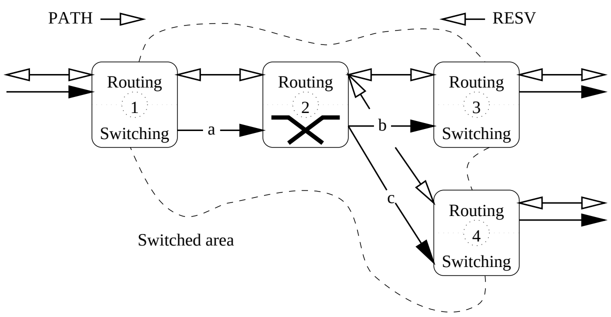
"b" correspondant. A ce niveau nous ne considérons pas les problèmes liés au filtrage ou à l'agrégation.

Dans la figure 3.4, un exemple de flot multicast est établi en adjoignant une nouvelle entrée dans la table de commutation. Cela est réalisé en ajoutant un message RESV qui produit l'association du label "c" pour ce flot en direction de (4).

Les frontières de la zone de commutation (*Switched area*) sont définies par les RSVP switches entrants (*"ingress" node*) et sortants (*"egress" node*). Ces éléments entrants ou sortants sont les points par lesquels les données commencent ou terminent leur passage dans la zone commutée. Dans la Figure 3.4, le nœud 1 est un élément entrant, les nœuds 3 et 4 sont des éléments sortants.

L'utilisation de la commutation a un impact sur l'entête des données. Par exemple le TTL n'est plus modifié tant que le paquet progresse dans la couche liaison sans remonter au niveau du routage. Nous devons donc le recalculer à partir des paquets de signalisation qui sont toujours routés. En conservant la signalisation au niveau routage, on évite les boucles de manière habituelle.

Les problèmes que nous présentons dans la suite sont identiques à certains que nous avons déjà rencontrés dans d'autres approches de l'intégration d'IP sur ATM. Il sont liés au modèle restrictif de la couche liaison qui ne peut être étendu sans remonter dans une couche supérieure. Pour mettre en œuvre la solution RSVP Switching, nous devons déplacer les fonctionnalités qui devaient prendre place au niveau routage hors de la zone commutée. Ainsi le filtrage et l'agrégation, qui ne peuvent avoir lieu dans la couche liaison, sont réalisés en dehors de cette zone. C'est ce que nous présentons dans les deux prochaines sections.



Informations dans le nœud 2					
Table des labels			*Table des* cut-through		
Session	Label	Port	Session	De	Vers
S	"b"	(3)out	**S**	"a" (1)	"b" (3)
S	"c"	(4)out		"a" (1)	"c" (4)
S	"a"	(1)in			

FIG. 3.4: Création d'un cut-through multicast

3.2.2 Filtrage

Dans RSVP, la fonctionnalité de filtrage permet de sélectionner des sous-ensembles de paquets dans les flots d'une session. Ces sous-flots sont définis par n'importe quel champ des entêtes d'IP et des couches supérieures. Par exemple, on utilise habituellement un filtrage par émetteur pour sélectionner les flots dans une session après une agrégation. D'autres types de filtrage permettent d'extraire différents sous-flots dans un flot utilisant un codage hiérarchique. Dans la version actuelle des spécifications de RSVP, seule l'adresse des émetteurs est utilisée pour des raisons de simplicité, mais l'extension peut être réalisée de manière aisée.

Dans la suite, nous présentons comment nous gérons le filtrage pour une session RSVP avec sous-flots. Nous proposons d'utiliser un "pré-filtrage" avant la zone commutée à l'aide de la signalisation de réservation. La signalisation continue d'emprunter le chemin routé et les messages PATH ne sont pas modifiés. L'arbre de retour est construit par ces messages de manière habituelle. Les messages RESV sont étendus pour transporter l'objet `LABEL`. Pour pouvoir fournir la QoS hétérogène orientée récepteur, plusieurs arbres multicast sont créés au niveau de la couche liaison pour recevoir le trafic de données commuté de chaque sous-flot. Ces arbres sont construits pendant la phase de réservation. Chaque sous-flot est caractérisé par une QoS homogène ainsi que le requiert la couche de commutation.

Dans le but d'éviter le problème du filtrage au niveau de la couche liaison, celui-ci est géré au début de la zone commutée. L'élément entrant doit assurer la fonctionnalité de filtrage en sélectionnant et transmettant les paquets de données sur l'arbre multicast correspondant. A la sortie de la zone

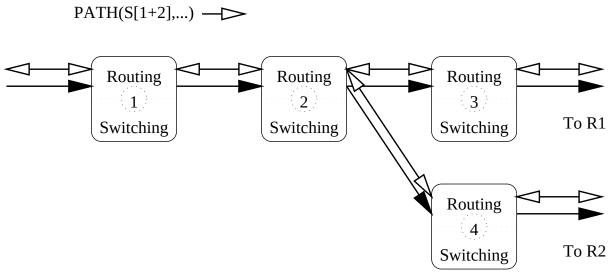
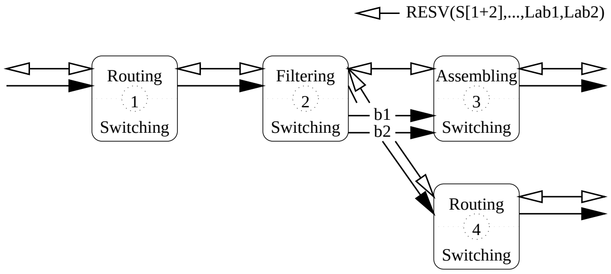


FIG. 3.5: Chemin initial routé pour le filtrage



Informations dans le nœud 2					
Table des labels			*Table des* cut-through		
Session	Label	Port	Session	De	Vers
S sub 1	"b1"	(3)out			
S sub 2	"c2"	(3)out			

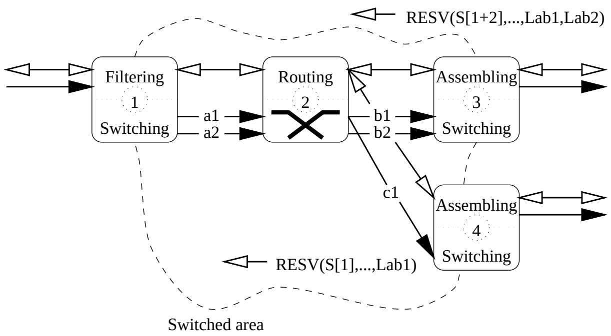
FIG. 3.6: Association des labels pour le filtrage

commutée, un élément sortant peut avoir besoin de plusieurs sous-flots pour recréer le flot avec la QoS demandée (le filtrage maximum est toujours réalisé en entrée de la zone commutée pour préserver la dynamique du modèle).

Nous présentons dans la Figure 3.5, un émetteur générant un flot de données dans la session RSVP **S**. Ce flot est composé de deux sous-flots (1 et 2). Les récepteurs peuvent demander le flot complet ou un sous-ensemble à travers l'un des sous-flot. Les messages PATH transportent l'information vers les récepteurs pour leur permettre de choisir les sous-flots adéquats à demander lors de la phase de réservation.

Dans la Figure 3.6, le récepteur R1 demande le flot complet et effectue une réservation de ressources pour les sous-flots 1 et 2. Les RSVP Switches gèrent les liens virtuels en associant chaque sous-flot aux labels respectifs "b1" et "b2".

Dans la Figure 3.7, le récepteur R2 réclame une QoS plus faible en ne réservant des ressources que pour le sous-flot 1. Ce sous-flot est associé au label "c1". La réservation effectuée entre les nœuds 1 et 2 est le "maximum"



Informations dans le nœud 2					
Table des labels			*Table des* cut-through		
Session	Label	Port	Session	De	Vers
S sub 1	"b1"	(3)out	**S sub 1**	"a1" (1)	"b1" (3)
S sub 2	"b2"	(3)out		"a1" (1)	"c1" (4)
S sub 1	"c1"	(4)out	**S sub 2**	"a2" (1)	"b2" (3)
S sub 1	"a1"	(1)in			
S sub 2	"a2"	(1)in			

FIG. 3.7: Création du *cut-through* pour le filtrage

des deux réservations précédentes (agrégation des messages RESV), ce qui correspond aux réservations pour les sous-flots 1 et 2. Les labels associés aux liens virtuels respectifs sont "a1" et "b1".

Les flots entrants et sortants du RSVP Switch 2 ont un label ; les *cut-through* peuvent être mis en place. Pour le sous-flot 1 de la session **S**, un *cut-through* multicast de (1) avec le label "a1" vers (3) et (4) avec les labels respectifs "b1" and "c1" est construit. Pour le sous-flot 2 de la session **S**, un *cut-through* unicast est créé de (1) avec le label "a2" vers (3) avec le label "b2".

La fonctionnalité de filtrage est assurée dans le nœud 1. Celui-ci est l'élément entrant de la zone commutée. Ce décalage du filtrage évite de le réaliser au sein du nœud 2 où il aurait eu lieu avec le routage. De même ce mécanisme permet d'utiliser le modèle multicast homogène dans la zone commutée en conservant globalement les possibilités de RSVP.

Dans le nœud sortant 3, les deux sous-flots sont nécessaires pour reconstruire le flot initial réclamé. Dans le nœud sortant 4, seul le sous-flot 1 est nécessaire. Les éléments sortants de la zone commutées sont responsables du calcul du nouvel entête des paquets après que celui-ci est été laissé inchangé après son passage continu dans la couche liaison.

3.2.3 Agrégation

L'agrégation est une fonctionnalité nécessaire au modèle RSVP pour permettre l'utilisation efficace du service multipoint-à-multipoint. Cela permet de réduire la consommation de ressources par rapport au service point-à-multipoint de la même manière que le multicast le fait par rapport à l'unicast. Associée avec le filtrage, l'agrégation permet de réunir des flots d'émetteurs différents au sein d'une même session pour les séparer ensuite.

Le problème posé par l'agrégation est difficile à résoudre en environnement commuté à cause de l'absence de fonctionnalité multipoint-à-point au niveau de la couche liaison. La question à résoudre est comment, au niveau commutation, utiliser plusieurs labels vers un seul en sortie sans entrelacement. Nous utilisons des circuits séparés dans tout le parcours commuté. Nous utilisons le même principe que pour le filtrage, nous déplaçons la fonctionnalité hors de la zone commutée. L'agrégation a lieu après les *shortcuts* pour permettre à tout flot de rejoindre une session, et ce dans n'importe quel nœud. A la sortie de la zone commutée, dans l'élément sortant, les différents flots remontent au niveau routage où ils sont agrégés.

La séparation des flots ne signifie pas que nous séparons les réservations, et effectuons ainsi de la sur-réservation. Nous conservons les identifiants de la couche liaison, les labels, séparés tout en groupant les ressources qui leur sont attribuées. Nous associons une réservation commune à plusieurs liens virtuels. La section 3.3.2 présente la manière de la réaliser avec la technologie ATM. Dans la couche réseau, nous conservons la signalisation RSVP qui nous indique une réservation pour la session, mais plusieurs objets `LABEL` sont transportés au sein des messages RESV pour identifier chaque sous-flot. Pour les styles RSVP, on obtient :

WF et SE (*Wildcard Filter* et *Shared Explicit*) Les styles de réservations partagées utilisent un label par émetteur et une seule réservation correspondant au "maximum" des ressources demandées. Le nombre d'émetteurs avec le style WF est potentiellement élevé, mais dans la pratique, il est limité par l'objet RSVP `SCOPE`.

FF (*Fixed Filter*) Le style de réservations séparées utilise un label et une réservation par émetteur. La quantité de ressources demandées correspond à la "somme" des ressources réservées pour chaque flot.

Dans la Figure 3.8, deux émetteurs (E1 et E2) envoient des messages PATH dans la session **S**. Leurs trafics de données vers le récepteur R sont agrégés. Les messages PATH transportent l'information de leurs sources pour permettre le filtrage.

La réponse aux messages PATH est un message RESV avec deux labels "b1" et "b2", correspondant aux deux sources et aux flots associés dans la session **S** (Figure 3.9). Seule une réservation est gérée, mais elle est associée aux deux flots. Au niveau commutation, deux liens virtuels sont crées.

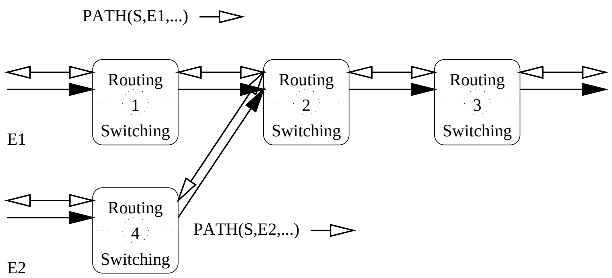
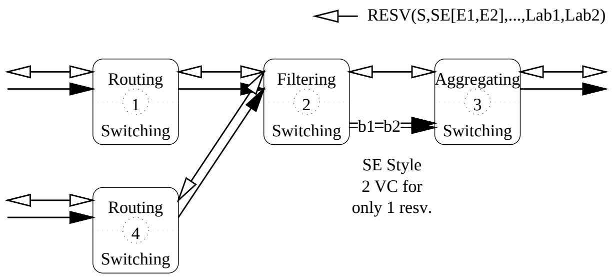


FIG. 3.8: Chemin initial routé pour l'agrégation



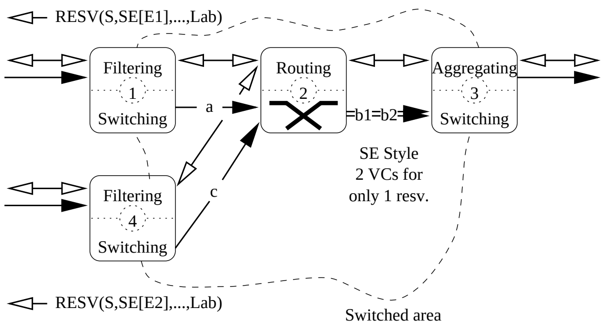
Informations dans le nœud 2					
Table des labels			*Table des* cut-through		
Session	Label	Port	Session	De	Vers
S(E1)	"b1"	(3)out			
S(E2)	"b2"	(3)out			

FIG. 3.9: Association des labels pour l'agrégation

Lorsque l'affection des labels est terminée de chaque coté du nœud 2 (Figure 3.10), le *cut-through* est créé. Le trafic de données de chaque émetteur utilise des labels différents dans toute la zone commutée, restant ainsi dans des circuits virtuels distincts et évitant le multiplexage au niveau de la couche liaison. Chaque flot est associé à un *cut-through* dans le nœud 2. Les flots sont agrégés à la sortie de la zone commutée.

La fonctionnalité d'agrégation est déplacée du nœud 2, où elle a lieu au niveau RSVP, vers les nœuds sortants 3 et 4. Ce mécanisme autorise l'utilisation de RSVP dans un environnement commuté. Comme pour le filtrage, à la sortie de la zone commutée, il est nécessaire de recalculer l'entête des paquets.

L'agrégation et le filtrage sont réalisés dans chaque nœud intermédiaire de la zone commutée pour fournir le service dynamique associé à RSVP. La zone commutée peut être variable, donc tous les RSVP Switches doivent pouvoir assurer les fonctionnalités des nœuds entrants et sortants. Un nœud en-



Informations dans le nœud 2					
Table des labels			*Table des* cut-through		
Session	Label	Port	Session	De	Vers
S(E)1	"b1"	(3)out	**S (E1)**	"a" (1)	"b1" (3)
S(E)2	"b2"	(3)out			
S(E1)	"a"	(1)in	**S (E2)**	"c" (4)	"b2" (3)
S(E2)	"c"	(4)in			

FIG. 3.10: Création du *cut-through* pour l'agrégation

trant gère la séparation en sous-flots et un nœud sortant assure le réassemblage partiel de ces sous-flots ainsi que l'agrégation. La gestion des sous-flots est identique à la gestion du style SE dans notre modèle. Les RSVP Switches sont aussi tenus d'assurer les fonctionnalités de base associées au protocole de routage, à l'architecture RSVP et aux mécanismes de commutation.

3.3 Spécialisation du modèle théorique RSVP Switching

La solution RSVP Switching utilise une approche générique, en ne spécifiant pas de protocole de réseau particulier, ni de technologie de commutation, même si les limitations envisagées de cette dernière sont fortement typées commutation de cellules. Nous présentons en conséquence une spécialisation de notre proposition, qui est adaptée ici au protocole IPv6 et à la technologie ATM.

3.3.1 IPv6 et RSVP Switching

Nous proposons d'appliquer l'approche RSVP Switching dans un environnement IPv6. Cette nouvelle version du protocole IP intègre le multicast en natif ; seule l'adjonction des éléments RSVP est nécessaire pour obtenir un service multipoint avec QoS. La nouvelle version intègre un entête de taille fixe avec des extensions chaînées, permettant un traitement efficace du paquet. Par contre, l'accès aux champs des couches supérieures est difficile à

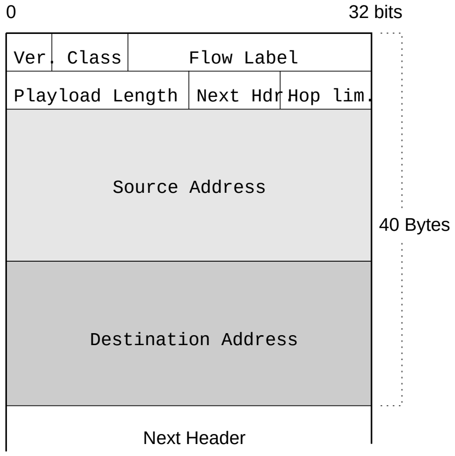


FIG. 3.11: Entête Ipv6

cause de ce chaînage et d'autres obstacles tels que le cryptage des paquets. Nous présentons l'utilisation des champs IPv6 telle que nous l'envisageons dans RSVP Switching.

Un nouveau champ a été défini dans l'entête IPv6 : le `Flow label` (voir Figure 3.11). Celui-ci nous permet d'identifier des sous-flots sans recourir à la violation de la structure en couche comme cela est nécessaire avec IPv4. La description suivante présente l'utilisation des champs IPv6 :

- L'identification de la session RSVP est réalisée avec les champs `Destination address` et `Flow label`.
- L'identification de la source pour le filtrage est réalisée avec le champ `Source address`.
- L'identification des sous-flots[1] peut être faite avec une partie du champ `Flow label` que nous appelons "sub" (réduisant ainsi la taille de `Flow label` pour l'identification de la session).

Une possibilité d'utilisation des champs IPv6 pour RSVP Switching est la suivante :

Filtrage (Table des labels dans le nœud 2) :

- "b1" port(3)out $\mapsto$ @dest(S), Flow(S), sub1
- "b2" port(3)out $\mapsto$ @dest(S), Flow(S), sub2

Agrégation (Table des labels dans le nœud 2) :

- "b1" port(3)out $\mapsto$ @dest(S), Flow(S), @src(E1)

[1]Précédemment, l'identification des sous-flots pouvait être réalisée par le champ `Priority` (priorités relatives à la source). Ce champ de l'entête ipv6 est maintenant remplacé par le champ `Class`, dans le but d'assurer une différenciation de service (priorités relatives au réseau) et ne peut donc plus être utilisé pour la mise en œuvre de nos mécanismes.

– “b2” port(3)out $\mapsto$ @dest(S), Flow(S), @src(E2)

Cette mise en correspondance associe les labels de chaque port d’un commutateur avec les champs précités. “@dest” est l’adresse destination, “Flow” est le *flow label* associé à la session RSVP, “sub1” et “sub2” sont les champs utilisés pour distinguer les sous-flots et “@src” est l’adresse source.

La table des labels doit contenir les quatre champs suivants pour permettre la classification complète des paquets avec RSVP Switching : adresse destination, *flow label*, adresse source et identificateur de sous-flots.

3.3.2 ATM et RSVP Switching

Dans notre proposition, nous utilisons la technologie ATM au niveau commutation. Nous nous intéressons à la partie matérielle des commutateurs ATM, qui comprend principalement la matrice de commutation et des mécanismes d’ordonnancement. Le choix d’ATM repose sur le fait que c’est la principale technologie de commutation disponible. De nombreuses autres solutions proposant de la commutation (réseaux locaux commutés) intègre aussi de la technologie ATM.

Avec ATM, les labels prennent naturellement place dans les champs VPI/VCI de l’entête des cellules. La table d’échange des labels au niveau de la couche de liaison correspond à la table de commutation et l’utilisation des *shortcuts* est similaire à celles des circuits virtuels ATM.

De la même manière qu’IP Switching, nous gérons directement le matériel ATM, mais sans avoir besoin d’un protocole spécifique, tel que IFMP, pour assurer l’échange des VPI/VCI. La diffusion des VPI/VCI est réalisée à travers le nouvel objet `LABEL`.

La Figure 3.12 présente l’architecture d’un RSVP Switch. C’est l’intégration des architectures RSVP et IP Switching. Au niveau de la couche réseau, nous trouvons les éléments classiques liés à RSVP avec une extension pour la gestion des labels dans le module RSVP. Celle-ci utilise l’objet `LABEL` et communique avec le commutateur ATM à travers le protocole GSMP. Au niveau de la couche liaison se trouve la partie commutation ATM standard, où les flots routés utilisent des VC dédiés vers la partie routage du RSVP Switch, et où les flots commutés traversent seulement la matrice de commutation.

La partie ATM nécessite des capacités d’ordonnancement, avec un flot par file d’attente (*per flow queuing*) ou avec un groupe de flots par file d’attente (*per class queuing*). La première technique permet de gérer simplement la QoS sur la base d’un flot (une file d’attente par VPI/VCI). La seconde est utilisée pour gérer la QoS des agrégations de flots (où plusieurs flots sont associés à la même réservation et plusieurs VPI/VCI à une file d’attente). Dans ce cas, on parle d’ordonnancement par classes ; des ensembles de labels sont associés à ces classes qui représentent la réservation de la session. Cette fonctionnalité apparaît dans les commutateurs récents. Une présentation de

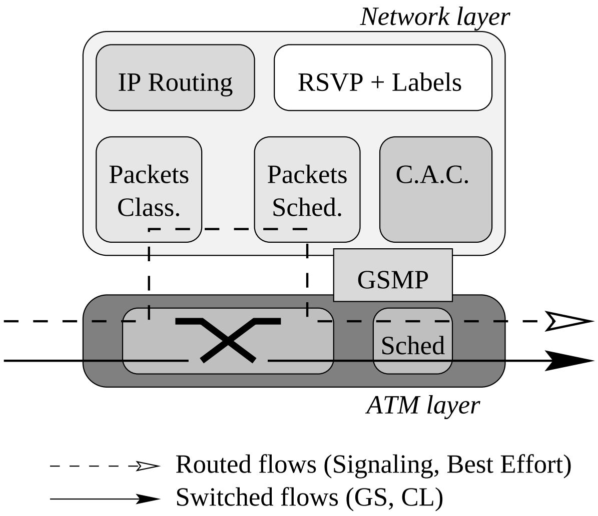


FIG. 3.12: Architecture d'un RSVP Switch

l'ordonnancement par classes se trouve dans [FJ95]. Cette technique permet d'éviter l'utilisation de commutateurs avec la fonctionnalité de *VC Merge* (commutateurs travaillant au niveau des trames pour réaliser le service N-vers-1)[2].

Notre proposition n'est pas destinée au support de tous les types d'applications. Nous visons les applications avec des contraintes temporelles fortes ; ce qui demande le maintien d'un état par flot dans le réseau, que nous réalisons avec le *Soft State* de RSVP. Un des problèmes majeurs rencontrés avec ATM est la limitation de l'espace des VPI/VCI, car nous associons un identifiant VPI/VCI par flot ou sous-flot. Cela ne pose pas de problèmes au niveau d'un hôte, généralement limité à quelques centaines d'identifiants, ou à un commutateur de réseau local qui adressera la totalité des quelques 65.000 VPI/VCI. Par contre on ne peut envisager d'utiliser RSVP Switching au sein d'un réseau fédérateur, où le nombre d'identifiants serait trop faible et où la charge d'un traitement de la QoS par flots beaucoup trop élevée.

D'autres solutions, telles que celles misent en œuvre dans le groupe de l'IETF DiffServ (*Differentiated Services*) proposent une approche sans état bien plus légère. Mais ces solutions ne peuvent procurer que des classes de services et, en aucun cas, assurer des contraintes temporelles dures. L'approche RSVP Switching n'est pas adaptée à un environnement global, mais s'intègre plutôt à un environnement de réseaux locaux, voire de fédération

[2]Le *VC Merge* est la solution proposée par plusieurs adaptations de RSVP à la commutation. Cette solution n'est qu'une adaptation directe des mécanismes de la couche réseau. On y perd les avantages liés à la commutation de cellules (temps de latence si on rassemble les trames) et on ne peut ensuite identifier les flots multiplexés.

de réseaux locaux à étendue limitée (réseau de campus) et aux VPN *Virtual Private Network*. L'utilisation combinée de réseaux locaux avec RSVP et de réseaux étendus est un problème important qui suscite de nombreuses recherches actuellement.

Chapitre 4

Classes de services

4.1 L'architecture IntServ/RSVP

Le groupe de travail IETF **IntServ** a défini une architecture pour l'intégration de services dans l'Internet, bâtie autour du protocole RSVP. Ce protocole est orienté récepteur. Il propose un modèle multicast dit "étendu" (multipoint à multipoint), dynamique, hétérogène et permettant l'agrégation (*merging*) des réservations de différents émetteurs. Restant conforme à la philosophie "non connecté" de IP, les sessions RSVP sont gérées par des *Soft States*, états périssables qui doivent être rafraîchis périodiquement sous peine de disparaître. Ceci permet l'effacement automatique d'états "orphelins" ainsi que la gestion transparente des changements de route.

Deux classes de service sont actuellement définies et sont présentées dans le Tableau 4.1 :

Classe	*Description*
Guaranteed Services (GS)	Garantie "dure" sur le délai de chaque paquet [SPG97].
Controlled Load (CL)	Service semblable à celui fourni par un réseau non congestionné [Wro97a].

TAB. 4.1: Les classes de service Internet

Pour chaque flot auquel est garantie une qualité de service (QoS) trois états doivent être maintenus dans chaque nœud traversé par ce flot (voir la Figure 4.1) :

Classification (identification des paquets du flot),

Ordonnancement (traitement approprié des paquets),

Signalisation (informations de contrôle).

Les deux premiers sont situés dans le chemin de données, on les appelle aussi "états rapides" (*fast states*). Le troisième est situé dans le chemin de contrôle, c'est un "état lent" (*slow state*).

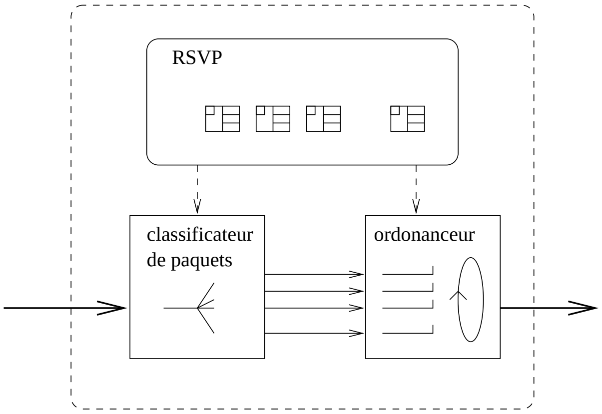


FIG. 4.1: Les 3 états de l'architecture IntServ

L'état de signalisation est utilisé beaucoup moins intensivement que les deux autres, ce qui peut amener à penser qu'il ne pose pas réellement de problème d'extensibilité. Il est cependant beaucoup plus lourd à gérer, aussi bien en place mémoire qu'en puissance de calcul (structures de données complexes, gestion de temporisations...).

4.2 Caractérisation des flots

Dans l'architecture IntServ, chaque flot est caractérisé par un TSPEC (*traffic specification*) qui comprend les cinq paramètres décrit dans le Tableau 4.2.

Paramètre	*Description*
b	Taille maximum de rafale.
r	Débit moyen.
p	Débit crête.
L (ou M)	Taille maximum de paquet.
m	Plus petite unité de policage.

TAB. 4.2: Les classes de service Internet

Les paramètres les plus importants sont les trois premiers, ils permettent de caractériser le flot selon un modèle constitué d'un **seau à jetons** (*token bucket*) qui contrôle le débit moyen et la taille de rafale, suivi d'un **seau percé** (*leaky bucket*) qui contrôle le débit crête. Ceci est illustré par la figure 4.2.

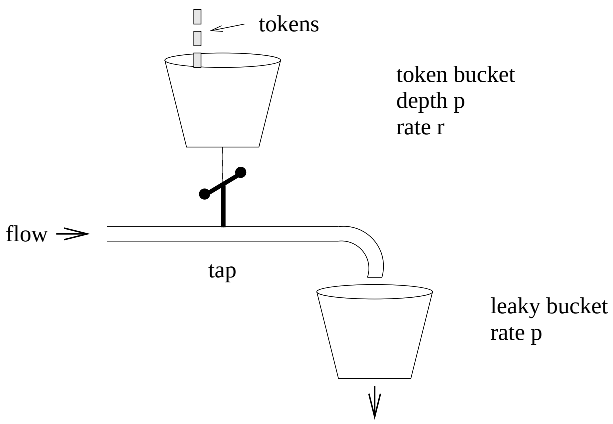


FIG. 4.2: Caractérisation d'un flot

Les deux autres paramètres sont secondaires, dans le sens où ils ne caractérisent le flot que par rapport à l'implémentation des mécanismes de qualité de service. Les ressources réservées dans le réseau dépendent bien entendu, pour une qualité de service donnée, de cette caractérisation.

Dans le cas du *Controlled Load*, le service fourni peut s'apparenter en première approximation à une garantie de bande passante moyenne. Par contre, le *Guaranteed Service* permet de réaliser des garanties dures et est basé sur des outils mathématiques relativement complexes [Cru96, Bou96]. Nous décrivons les bases théoriques du service GS dans la section 4.4.

4.3 Mécanismes

Une session RSVP est identifiée par une adresse IP destination (éventuellement multicast) et un numéro de session. Une application désirant participer en tant qu'émetteur dans cette session envoie des messages PATH contenant le TSPEC du flot qui sont routés jusqu'au(x) destinataire(s). Dans chacun des routeurs qu'ils traversent, ces messages PATH créent un *Path State* contenant essentiellement l'adresse du routeur précédent (*Previous Hop*), créant ainsi un **arbre de parcours inverse**. Chaque routeur ajoute également au message PATH des informations, contenues dans les objets ADSPEC, sur les qualités de service qu'il est capable de fournir. Un appel à la fonction de routage permet ensuite de déterminer le lien de sortie à utiliser pour envoyer le message vers le nœud suivant.

Quand une application réceptrice reçoit le message PATH, elle connaît alors les caractéristiques du flot de données proposé et celles du chemin parcouru. Elle spécifie alors la qualité de service qu'elle désire en envoyant un message RESV qui va "remonter" vers l'émetteur grâce aux *Path States* installés par le message PATH. À chaque nœud, un contrôle d'admission est réalisé pour vérifier que suffisamment de ressources sont disponibles pour fournir le service demandé. Si c'est le cas, un *Resv State* est créé, contenant les informations relatives à la réservation à effectuer et le message est envoyé au nœud amont. Les *Resv States* créés par différents récepteurs (de la même session) et associés au même lien de sortie sont accumulés dans un *Traffic Control State* contenant la réservation globale sur ce lien pour cette session.

Au niveau implémentation, un *Path State* est réalisé par un *Path State Block* (ou PSB), un *Resv State* par un *Resv State Block* (ou RSB) et un *Traffic Control State* par un *Traffic Control State Block* (ou TCSB) [BZ97].

4.4 Le cas du *Guaranteed Service*

La classe de service *Guaranteed Service* est conçue pour fournir aux applications une garantie stricte sur le délai subi par chaque paquet d'un flot pendant qu'il traverse les nœuds du réseau. Pour fournir ce type de garantie, il faut isoler complètement le flot des autres flots de données en lui fournissant un **canal de communication dédié** à l'intérieur de chaque lien qu'il traverse.

L'émulation parfaite de ce canal, dans laquelle le flot se voit attribuer une part de bande passante est appelée le **modèle fluide**. Comme l'on se situe dans un réseau à commutation de paquets les nœuds ne peuvent fournir qu'une approximation du service fourni par le canal dédié. Dans chaque nœud, la déviation entre le modèle fluide et cette approximation peut se caractériser par deux termes d'erreur que l'on définit de la manière suivante :

C est le terme d'erreur dépendant du débit. Il représente le délai qu'un paquet peut subir à cause des paramètres de caractérisation du flot.

D est le terme d'erreur indépendant du débit. Il représente la plus grande variation de délai (indépendante du débit du flot) qu'un paquet peut subir dans le nœud.

Les valeurs de ces termes d'erreur dépendent de la façon dont est réalisée l'approximation du canal du modèle fluide, c'est à dire de l'ordonnanceur mis en œuvre dans le nœud.

L'enveloppe fluide (i.e. dans un modèle abstrait non paquétisé) d'un flot caractérisé par le TSPEC (b, r, p) et ayant L comme taille maximum de paquet est [Ga96] :

$$A(\tau) = min\{p.\tau, b + r.\tau\}$$

ce qui s'explique de la manière suivante. À long terme (τ grand) le service maximum est obtenu quand le flot a émis une rafale de taille maximum (b) et continue d'émettre à son débit moyen (r). Ceci est exprimé par le terme $b + r.\tau$. Cependant, la rafale ne peut pas être émise instantanément, le débit étant limité par le débit crête (p). Ainsi, si à partir de $\tau = 0$ la source émet une rafale, elle pourra émettre à son débit crête jusqu'à ce que la taille maximum de rafale ait été atteinte, après quoi elle devra se limiter à son débit moyen. Donc, pour τ petit, l'enveloppe serait $p.\tau$.

Il faut cependant tenir compte du fait que le flot n'est pas fluide mais constitué de paquets. Quel que soit le lissage mis en œuvre, un paquet est toujours émis au débit du lien, qui peut être beaucoup plus important que le débit crête du flot et que l'on considérera donc comme infini. Ainsi, l'enveloppe du flot pour des valeurs de τ petites devient $L+p.\tau$, ce qui correspond à une augmentation de l'effet des rafales.

$$A(\tau) = min\{L + p.\tau, b + r.\tau\}$$

Voyons maintenant l'enveloppe du service fourni par le réseau. Dans le pire des cas, le réseau ne fournit aucun service avant qu'un délai $\sum(\frac{C_i}{R} + D_i)$ ne soit écoulé. Ensuite le service fourni est constant et égal à la réservation effectuée R.

La Figure 4.3 montre comment ces deux enveloppes peuvent être utilisées pour calculer le délai maximum subi par un paquet (délai d'attente dans les queues, n'incluant pas le délai de propagation) et la taille maximum de tampons nécessaires dans un nœud du réseau.

Le segment IJ représente le délai maximum, le segment EF la quantité maximum de données en attente de traitement, donc la taille maximum des tampons nécessaires pour éviter toute perte de paquet. Le calcul de la longueur du segment IJ nous amène à la formule suivante (n étant le nombre de nœuds) :

$$D = \frac{(b-L)}{R}\frac{(p-R)}{(p-r)} + \frac{L}{R} + \sum_{i=1}^{n}\left[\frac{C_i}{R} + D_i\right] \tag{4.1}$$

Bien entendu, cette équation n'est valable que si la réservation R est supérieure au débit moyen du flot. On peut exprimer cette équation différemment pour obtenir la réservation nécessaire pour un délai donné :

$$R = \frac{p.x + \sum_{i=1}^{n} C_i}{x + D - \sum_{i=1}^{n} D_j} \qquad avec \qquad x = \frac{b-L}{p-r} \tag{4.2}$$

Par exemple, considérons un lien OC-3 (155 Mb/s) en sortie d'une station ATM. Prenons le cas de l'émission d'un flot vidéo MPEG stocké sur disque, avec un débit moyen d'environ 3 Mb/s et une taille maximum de rafale de 100 Ko. Le débit crête n'est limité que par la vitesse du lien auquel est connectée la source, par exemple 25 Mb/s. Nous considérons que le flot traverse 5 nœuds avec un délai de propagation total de 20 ms. En soustrayant

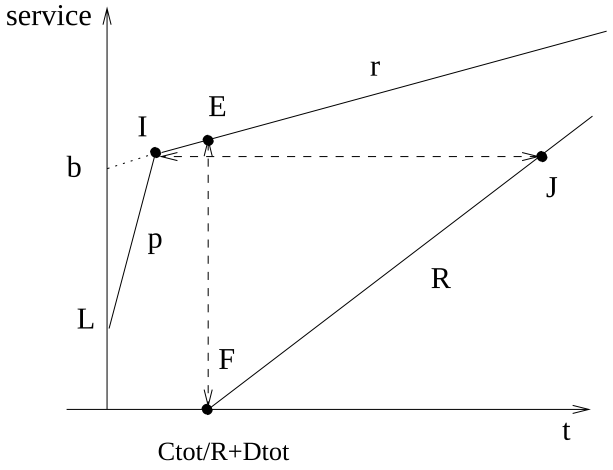


FIG. 4.3: Enveloppes du flot et du service fourni

le délai de propagation au délai de bout en bout désiré nous obtenons le délai dû aux files d'attente maximum admissible de bout en bout. Tous ces paramètres doivent être ajustés pour prendre en compte l'*overhead* ajouté par les couches ATM (encapsulation LLC/SNAP et en-tête de cellule). En utilisant la formule 4.2 nous pouvons déterminer la réservation à affecter au flot.

Les Figures 4.4 et 4.5 montrent la réservation nécessaire en fonction de la taille maximum de paquet (M) et du délai de bout en bout désiré (D). Dans la Figure 4.4, le délai de bout en bout requis est de 75 ms.

4.5 Une implémentation du *Controlled Load*

4.5.1 Motivation

L'implémentation du *Guaranteed Service* résulte en une faible utilisation de la bande passante du lien, puisque la réservation R est toujours plus grande que le débit moyen du flot. En fait elle peut même être beaucoup plus importante comme le montrent les courbes de notre exemple sur le flot vidéo. Bien sur cette bande passante n'est pas "perdue" dans le sens où elle reste disponible pour le trafic *Best Effort* si on utilise un ordonnanceur qui conserve le travail, par exemple un *Weighted Fair Queuing* (WFQ) [DKS89]. Cependant, dans la mesure où elle a été réservée, elle devrait pouvoir être utilisée pour fournir des garanties de services à d'autres flots. Évidement, le type de service qu'il est possible de garantir est bien plus "faible" que le GS puisque cette bande passante n'est disponible qu'en **moyenne**.

Beaucoup d'applications n'ont pas besoin de garanties strictes en terme de délai, mais ont plutôt besoin un certain niveau de garantie en bande passante. Nous pensons en particulier à toutes les applications qui ont besoin

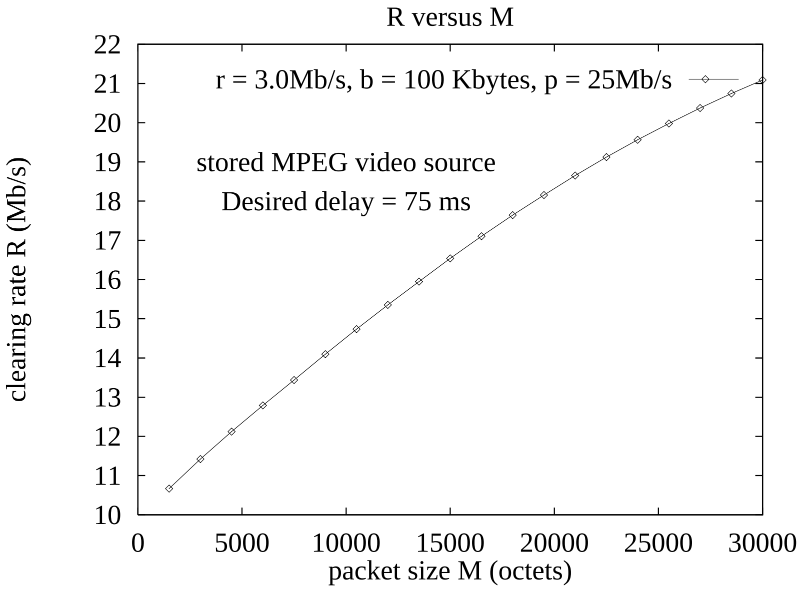


FIG. 4.4: Flot vidéo MPEG : R en fonction de M

d'une "interactivité à échelle humaine", c'est à dire qui ont besoin d'effectuer leurs transferts de données en un temps "raisonnable" (tel que perçu par l'utilisateur). Les butineurs web ou les applications de transfert de fichier en sont un bon exemple. En fait, la plupart des applications peuvent être satisfaites par un service à débit minimum plus une part équitable de la bande passante résiduelle. Pour le service *Best Effort*, le débit minimum garanti est égal à zéro.

Aussi nous proposons une classe de service permettant d'utiliser ces réservations pour fournir une garantie de bande passante "à long terme". L'idée de ce service est similaire à ce qui est présenté dans [Ga96]. La différence principale réside dans la façon dont il est implémenté et dont il interagit avec le trafic *Best Effort*.

Le type de QoS fournie par cette classe de service correspond tout à fait à ce qui est requis pour la classe de service *Controlled Load*. Il s'agit donc en fait d'une implémentation possible du CL, conçue pour être implémentée conjointement avec la classe GS de façon à optimiser l'utilisation des ressources.

4.5.2 Réalisation

Pour implémenter cette classe de service, que nous appellerons BR (pour *Bandwidth Recovery*), nous utilisons un ordonnanceur de type *Weighted Fair Queuing* avec des files séparées pour chaque flot GS ou BR. À chaque flot GS est affecté un poids qui détermine la quantité de bande passante R_i allouée à ce flot. Supposons qu'il y ait N flots GS chacun caractérisé par

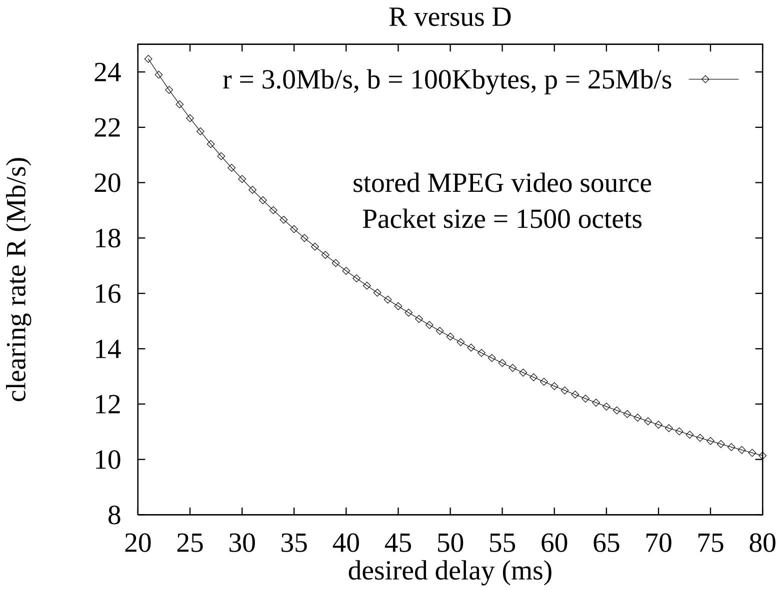


FIG. 4.5: Flot vidéo MPEG : R en fonction de D

une enveloppe (b, r, p) et un lien de capacité C_l. Pour garantir le service aux flots GS, l'inégalité suivante doit être respectée :

$$\sum_{i=1}^{N} R_i \leq C_l - \Delta_{\text{BR}} - \Delta_{\text{SIG}} \tag{4.3}$$

Cette inégalité est illustrée sur la partie gauche de la Figure 4.6. Dans l'équation précédente, Δ_{BR} et Δ_{SIG} représentent les portions de bande passante réservées respectivement aux flots BR et à la signalisation. Le poids associé au “flot signalisation” est basé sur une estimation de la bande passante nécessaire au trafic de signalisation.

Le poids associé à chaque flot BR, et donc la bande passante réservée, est choisie volontairement très petite. Elle permet juste aux flots BR d'accéder à la bande passante réservée mais pas utilisée par les flots GS. Dans un ordonnanceur de type WFQ, une portion de bande passante est garantie à chaque flot, mais si certains n'utilisent pas la totalité de ce qui leur a été réservé, d'autres flots tentant d'utiliser plus que leur propre part se partageront alors cette bande passante. Ainsi dans notre cas les flots GS n'utiliseront pas la totalité de leur part, et les flots BR utiliseront plus que la leur en récupérant la bande passante “gaspillée” par les flots GS. Ceci est illustré sur la Figure 4.6.

Considérons N' flots BR avec un débit moyen égal à r', leur garantie de débit minimum à long terme est possible si :

$\sum R_i$ $\sum r_i$

$\sum R_i - \sum r_i$ $\sum r'_i$

Δ_{BR}

Δ_{SIG} Δ_{SIG}

FIG. 4.6: Bande passante utilisée et réservée par les flots GS et BR

$$\sum_{i=1}^{N} r_i + \sum_{i=1}^{N'} r'_i \leq C_l - \Delta_{\text{SIG}} \tag{4.4}$$

L'équation (4.4) assure que si les flots GS respectent leur contrat de trafic (ou qu'un mécanisme de polissage — par exemple un algorithme GCRA au niveau de l'UPC — est utilisé), les flots BR recevront chacun un débit minimum garanti r'_i sur une échelle de temps raisonnable.

4.6 L'ordonnanceur utilisé

Dans un RSVP Switch, les flots peuvent être routés au niveau IP ou commutés au niveau ATM. Pour des raisons présentés précédemment, nous commutons les flots demandant une qualité de service. L'ordonnanceur mettant en œuvre les mécanismes décrits ci-dessus pour l'implémentation des services GS et BR est donc situé au niveau ATM.

Au niveau routage, on fait la distinction entre deux types de trafic :

- le trafic de signalisation,
- le trafic *Best Effort*.

Le trafic de signalisation est considéré vital et est donc prioritaire sur le trafic *Best Effort*, qui est traité comme un seul flot. Ainsi, l'ordonnanceur au niveau routage est un FIFO à priorités avec deux files, une pour la signalisation et une pour le trafic *Best Effort*.

Au niveau ATM, le trafic *Best Effort* en provenance du niveau IP est placé dans une file de priorité inférieure de façon à ne pas perturber les flots demandant une QoS. Il faut noter que dans nos calculs précédents concernant les flots GS et BR nous avons considéré qu'il leur était alloué toute la bande passante (mis à part la portion réservée à la signalisation). Nous pourrions très bien réserver une part de la bande passante pour le trafic *Best Effort*, en utilisant un terme Δ_{BE} de la même façon que nous l'avons fait avec Δ_{SIG}.

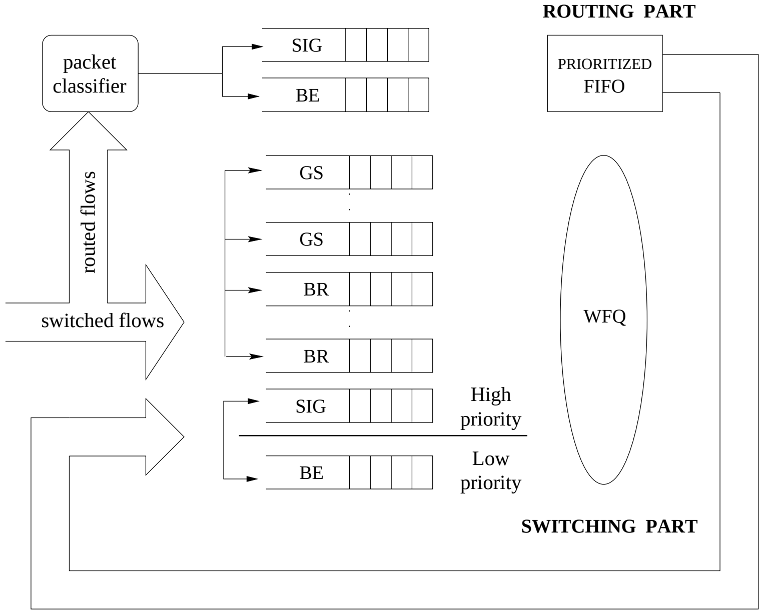


FIG. 4.7: Mécanisme d'ordonnancement

La Figure 4.7 décrit le mécanisme d'ordonnancement global que nous utilisons. Y sont représentés les flots GS, BR et BE et les files d'attente correspondantes.

4.7 Conclusions sur les classes de services

Nous proposons donc une implémentation des classes de service de l'Int-Serv avec les particularités suivantes :

- adapté à la technologie "commutation IP",
- prenant en compte le trafic de signalisation,
- utilisant une implémentation "intelligente" du service CL.

Chapitre 5

Implémentation

La proposition RSVP Switching a été implémentée puis intégrée et testée au sein de la plate-forme SAPHIR.

La Figure 5.1 offre une vue d'ensemble de l'implémentation qui a été faite de RSVP Switching.

Nous distinguons de haut en bas :

- le démon RSVP, lequel a été modifié afin de pouvoir gérer les labels,
- la pile IPv4/IPv6, utilisée dans le cas où aucune réservation n'a été établie,
- la couche d'émulation de la commutation RSVP Switching,
- et le gestionnaire ATM.

Nous décrivons au sein de ce chapitre ces différents niveaux ainsi que les interfaces entre eux.

5.1 Contexte d'implémentation et conséquences : émulation du RSVP switching

5.1.1 Le système Linux

La base de notre implémentation repose principalement sur trois composants issus de la communauté de recherche : une implémentation du démon RSVP, une souche IPv6 et un gestionnaire de carte ATM. Les machines de développement sont des PC sous Linux. Le système d'exploitation Linux est un clone d'Unix gratuit supporté par une large communauté de développeurs. Le noyau ainsi que la plupart des outils associés au système sont disponibles gratuitement et avec les sources sous licence de type GPL (*GNU General Public Licence*) [1].

[1]La GPL est une sorte d'anti-licence qui assure la gratuité d'un logiciel et de ceux qui en découlent. Elle peut être caractérisée par la phrase : *"This program is free soft-*

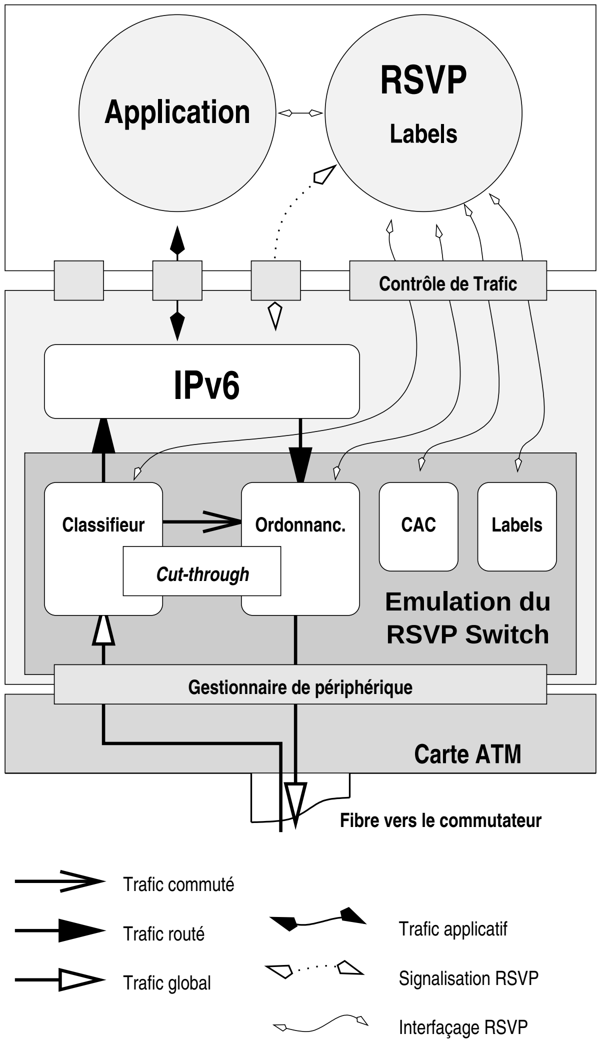


FIG. 5.1: Vue d'ensemble

5.1.2 Émulation du RSVP Switching

Notre plate-forme repose sur du matériel ATM standard (une switch Fore ASX200BX et un switch Centillon100 de Bay Networks), sur lequel nous ne pouvons accéder directement à la partie commutation. Celle-ci est en effêt gérée intégralement par le switch sans aucun moyen pour un hôte extérieur d'agir sur la matrice de commutation.

Il nous faut donc mettre en œuvre l'émulation d'un accès direct à cette matrice. Cette émulation va être effectuée au sein de l'hôte, dans une couche intermédiaire entre le gestionnaire de la carte ATM et les modules de la couche réseau (IPv6 et RSVP). L'émulation consiste à déplacer les fonctionnalités de routage ET de commutation au sein de l'hôte. Ainsi la présence de réservations au sein d'un flux permet aux datagrammes associés d'être commutés dans la couche d'émulation, sans avoir besoin de remonter au niveau trois (IPv4/6) [2]. Ceci est illustré Figure 5.1.

Si cette solution est moins performante (traitement logiciel des datagrammes après reassemblage des cellules, indirections supplémentaires), elle offre cependant l'avantage d'être opérationelle quelle que soit la nature du switch, et éventuellement quelle que soit la nature du support de communication, ce dernier servant dans notre cas de simple tuyau d'interconnexion.

5.2 Modifications de RSVP

Pour éviter la redondance de fonctionnalités nous avons intégré les protocoles de réservation et de distribution des labels. Cela se traduit par la création d'un nouvel objet RSVP pour le support de ces labels. La partie contrôle de trafic et le traitement des messages ont eux aussi été étendus ainsi qu'il est montré dans la section 5.2.2).

Cependant les mécanismes RSVP sont conservés de même que la compatibilité avec la signalisation RSVP standard. Ceci permettra une totale interopérabilité avec une implémentation RSVP classique, en particulier à l'interfaçage entre les zones ATM et Ethernet.

5.2.1 La souche RSVP et le module de gestion de ressources

Pour notre implémentation, nous utilisons la version 4.2a3 de la distribution des sources RSVP de l'ISI (*USC Information Sciences Institute*) intégrant IPv6. Ce démon assure la partie signalisation RSVP.

---

ware; you can redistribute it and/or modify it under term of the GNU General Public License as published by Free Software Foundation". La GPL peut être trouvée à l'URL : `http ://www.fsf.org/copyleft/copyleft.html`

[2] Seuls les datagrammes pour lesquels aucune réservation n'a été établie sont traités ainsi qu'il se doit au niveau 3. Cela inclut en particulier les messages de signalisation RSVP.

Length (= 8 octets) sur 16 bits	**Class** (= 40) 8 bits	**C-type** (= 1) 8 bits
Label sur 32 bits		

FIG. 5.2: Objet `LABEL`

A celui-ci nous avons associé la partie noyau de réservation et gestion de ressources issue de la souche IPv6. C'est ce module qui contrôle effectivement l'allocation des ressource via un ordonnancement adéquat des paquets à transmettre.

Ces deux composants : démon et module de gestion de ressource, sont essentiels à l'obtention des services désirés et communiquent via une interface définie à la section 5.2.3.

5.2.2 Ajout de la gestion des labels dans RSVP

Dans un premier temps, les concepts de sous-flots vus dans le modèle théorique ne sont pas introduit. Nous allons donc associer nos labels à des flots RSVP, identifiés par leur émetteur au sein d'une session.

L'objet introduit respecte le format défini par la spécification de RSVP, minimisant ainsi les modifications à réaliser au traitement des messages. L'objet `LABEL` est présenté dans la Figure 5.2. Des valeurs libres ont été choisies pour l'identifier (**Class** = 40). Nous définissons initialement le type label seul sur 32 bits (**C-type** = 1). L'objet ainsi défini fait 8 octets de long. Il peut contenir un label ATM basé sur le VPI/VCI ou celui d'une autre technologie. L'intégration de l'objet est réalisée en le liant à un objet qui identifie le flot de données. Les labels étant nécessaires seulement en cas de réservation, nous les insérons dans la structure du message RESV qui est défini ainsi :

```
<Resv Message> ::=
                <Common Header> [ <INTEGRITY> ]
                <SESSION> <RSVP_HOP> <TIME_VALUES>
                [ <RESV_CONFIRM> ] [ <SCOPE> ]
                [ <POLICY_DATA> ... ]
                <STYLE> <flow descriptor list>

<flow descriptor list> ::=
                <empty> |
                <flow descriptor list> <flow descriptor>
```

La définition des `<flow descriptor list>` est spécialisée selon les différents styles disponibles :

WF Style :

```
<flow descriptor list> ::=
                <WF flow descriptor>
```

```
<WF flow descriptor> ::=
                <FLOWSPEC>
```

FF style :

```
<flow descriptor list> ::=
                <FLOWSPEC> <FILTER_SPEC>  |
                <flow descriptor list> <FF flow descriptor>

<FF flow descriptor> ::=
                [ <FLOWSPEC> ] <FILTER_SPEC>
```

SE style :

```
<flow descriptor list> ::=
                <SE flow descriptor>

<SE flow descriptor> ::=
                <FLOWSPEC> <filter spec list>

<filter spec list> ::=
                <FILTER_SPEC> |
                <filter spec list> <FILTER_SPEC>
```

Les descripteurs de flots des styles contiennent l'information sur l'émetteur du flot dans l'objet `FILTER_SPEC`. C'est associé à celui-ci que nous localisons l'objet `LABEL`. Le style WF ne contient pas de `FILTER_SPEC`, nous ne l'utiliserons pas dans un premier temps. L'alternative aurait été d'associer le label à l'objet `FLOWSPEC` qui est présent dans tous les styles, mais il aurait été nécessaire d'agréger les labels dans le style SE et de trouver un mécanisme pour identifier les émetteurs avec le style WF. Les styles supportés sont définis ainsi :

FF style :

```
<flow descriptor list> ::=
                <FLOWSPEC> <FILTER_SPEC> [ <LABEL> ] |
                <flow descriptor list> <FF flow descriptor>

<FF flow descriptor> ::=
                [ <FLOWSPEC> ] <FILTER_SPEC> [ <LABEL> ]
```

SE style :

```
<flow descriptor list> ::=
                <SE flow descriptor>

<SE flow descriptor> ::=
                <FLOWSPEC> <filter spec list>

<filter spec list> ::=
                <FILTER_SPEC> [ <LABEL> ] |
                <filter spec list> <FILTER_SPEC> [ <LABEL> ]
```

Les objets `LABEL` peuvent être omis pour rester compatibles avec le trafic RSVP standard. Une présentation complète des travaux de modification de RSVP se trouve dans [Die98].

5.2.3 Modification de l'interface avec les services noyaux

Plusieurs fonctions permettent au démon RSVP, chargé de la signalisation, de dialoguer avec le module de gestion de ressources du noyau, implémenté à part. Ce sont :

```
rhandle  TC_AddFlowspec(OIf, FLOWSPEC, SENDER_TSPEC,
                        ADSPEC, flags, FLOWSPEC);
FLOWSPEC TC_ModFlowspec(OIf, rhandle, FLOWSPEC, SENDER_TSPEC,
                        ADSPEC, flags, FLOWSPEC);
         TC_DelFlowspec(OIf, rhandle);

fhandle  TC_AddFilter(OIf, rhandle, Session, FILTER_SPEC);
         TC_DelFilter(OIf, fhandle);
```

Les fonctions `TC_AddFlowspec`, `TC_ModFlowspec` et `TC_DelFlowspec` sont utilisées pour initier et modifier les paramètres de l'ordonnanceur de paquets, et les fonctions `TC_AddFilter` et `TC_DelFilter`, pour les paramètres du classifieur de paquets. Les *Flowspecs* et *Filterspecs* existant peuvent être référencées par des *handles* retourné par les par fonctions. Les détails de ces fonctions peuvent être trouvés dans [BZB+97].

Modification de l'interface

Afin de prendre en compte la gestion des labels, nous avons modifié cette interface. La liste des nouvelles fonctions relatives au contrôle de trafic est la suivante :

```
rhandle  TCrsw_AddFlowspec(OIf, FLOWSPEC, SENDER_TSPEC,
                           ADSPEC, flags, FLOWSPEC);
FLOWSPEC TCrsw_ModFlowspec(OIf, rhandle, FLOWSPEC, SENDER_TSPEC,
                           ADSPEC, flags, FLOWSPEC);
         TCrsw_DelFlowspec(OIf, rhandle);

fhandle  TCrsw_AddFilter(OIf, rhandle, Session, FILTER_SPEC);
fhandle  TCrsw_AddFilter(OIf, rhandle, Session, FILTER_SPEC,
                         LabelSwap);
         TCrsw_DelFilter(OIf, fhandle);

LABEL    TCrsw_GetLabel(IIf);
         TCrsw_FreeLabel(LABEL);
```

Les labels à échanger sont passés en paramètres par la fonction de contrôle de trafic `TCrsw_AddFilter`. Ces labels sont fournis à RSVP par `TCrsw_GetLabel` et rendus par `TCrsw_FreeLabel`.

La description de l'implémentation des modifications de RSVP ISI se trouve dans [Die98].

5.3 La souche IPV6 de la DRET

La souche IPv6 utilisée est la souche DRET développée conjointement par le LIP6 et l'INRIA. Cette souche implémente un jeu réduit de fonctionnalités et a pour but de mener des expérimentations pour la gestion de la QoS. Elle repose sur un noyaux Linux 2.1.23 modifié spécifiquement. Des informations sur cette souche sont disponibles à :

`http ://www-rp.lip6.fr/IPv6`

5.4 Le gestionnaire Linux-ATM de l'EPFL

Nous utilisons les gestionnaires de carte ATM de l'EPFL. Depuis 1995, l'EPFL (École Polytechnique Fédérale de Lausane) développe le support d'une plate-forme ATM [Alm96] sur Linux pour les communautés de la recherche et de l'enseignement. Des gestionnaires de carte ATM et des implémentations de protocoles ont été réalisés et sont disponibles au sein de la distribution Linux-ATM.

Les cartes ATM que nous utilisons font partie du matériel supporté. Ce sont des cartes **Efficient Networks ENI155** pour port PCI. Ces cartes supportent des débits de 155 Mbps sur fibre optique multimode.

Les types de connexions supportées sont les suivants :

- PVC point-à-point avec les classes de services CBR et UBR,
- SVC point-à-point avec les classes de services CBR et UBR et la signalisation UNI 3.0 ou UNI 3.1.

Les éléments de signalisation suivants ne sont pas supportés :

- Les classes de service VBR et ABR pour tous types de connexions,
- Les **connexions point-à-multipoint**,
- La signalisation UNI 4.0.

Les services et protocoles suivants sont fournis avec la distribution Linux-ATM de l'EPFL :

- Accès direct à la commutation ATM à travers l'AAL0 (absence d'AAL),
- Accès direct au service de transport de l'AAL5,
- *Classical IP* en encapsulation NULL ou LLC/SNAP.
- *LAN Emulation* client et serveur (LEC, LECS, LES et BUS),
- Arequipa (*Application REQuested IP over ATM*), développé par l'EPFL pour établir des VC avec QoS entre deux hôtes IP membres du nuage ATM.

Le gestionnaire a été modifié par l'INRIA afin de permettre un fonctionnement avec IPv6 en mode Classical IP. Les connections ATM établies sont de type PVC point-à-point, une adresse IPv6 est établie à chaque extrémité, ainsi qu'une interface logique. La table de routage IPv6 est alors mise à jour et une entrée ajoutée pour chaque PVC pour atteindre la station en regard.

5.5 La couche émulation de la commutation

5.5.1 Organisation générale

La couche d'émulation remplace la commutation qui aurait due être effectuée dans le commutateur et le protocole de communication GSMP qui aurait dû être utilisé. En effet, la communication s'établissant directement dans l'hôte, nous n'avons plus besoin de GSMP. Les échanges entre RSVP et la nouvelle couche sont gérés principalement à travers les fonctions liée au contrôle de trafic ainsi qu'il est expliqué plus loin. La Figure 5.3 décrit les fonctions à mettre en œuvre pour l'émulation de la commutation.

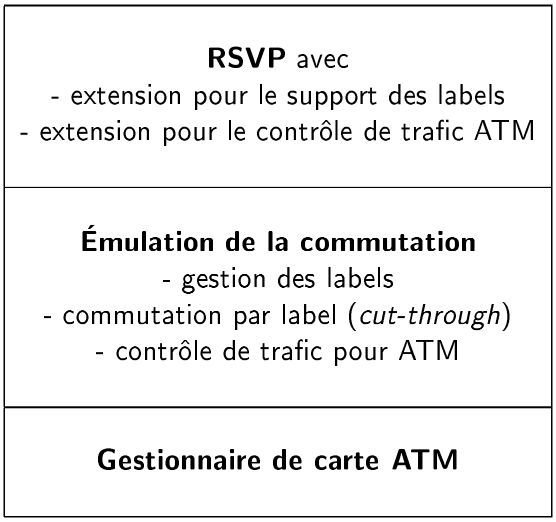


FIG. 5.3: Émulation de la commutation, vue générale

5.5.2 Labels utilisés

Les labels manipulés par le protocole RSVP sont des entiers 32 bits n'ayant pas à ce niveau de signification propre. En revanche les labels correspondent, au sein de la couche d'émulation de la commutation, au couple VPI/VCI associé au PVC. En effet, cette information, associée à l'identité de l'interface ATM de réception, permet d'identifier totalement la qualité de service associé au flux le cas échéant. De surcroit, il n'est nullement nécessaire d'inclure l'information dans les entêtes protocolaires (IPv4/6) puisque celle-ci est associée au datagramme par le gestionnaire ATM (`struct sk_buff` dans `include/linux/skbuff.h`).

5.5.3 Table de commutation

A la structure device associée à chaque interface logique (en l'occurence il existe une interface par PVC créé) nous avons associé une table de commutation. Cette table est indexée sur le label d'entrée et fait correspondre une interface de sortie et un label de sortie. Nous avons modifié la structure device afin d'ajouter un pointeur sur l'entrée correspondante dans la table de commutation. La table de commutation est définie ainsi :

```
/*
 * Switching table structure.
 */
typedef struct {
        struct device   *dev;           /* outgoing interface */
        label_t         label;          /* outgoing label */
} swtl_t;                               /* switching table line */

/*
 * Array of switching tables; one switching table per interface.
 * (eth0, atm0, atm1, atm2, etc.)
 */
#define MAX_SWTAB       10

extern  swtl_t  swtabs[MAX_SWTAB][MAX_LABELS];
```

5.5.4 Mécanismes d'ordonnancement

La QoS est fournie à l'aide des mécanismes d'ordonnancement situés dans la couche d'émulation. Les datagrammes sont ensuite envoyés au sein d'un PVC UBR standard. Il n'y a pas besoin de garanties de QoS ATM au sein de ce PVC puisque ces garanties sont fournies par l'ordonnancement effectué au sein de chaque hôte (voir la section 4.6).

5.5.5 Création d'une nouvelle famille d'adresses et ioctls

Les communications entre composants de niveau applicatif (démon RSVP et outils de configuration/débuggage) et composants noyau sont effectuées par le biais d'une nouvelle famille de protocoles (`AF_RSVPSW`) créée pour la circonstance. Une application se contente alors d'ouvrir une socket en spécifiant cette famille, puis une fois le descripteur de fichier récupéré, de générer des ioctls. Ces derniers sont de plusieurs types et permettent d'implémenter les fonctions de contrôle de trafic `TCrsw_*` mentionnées à la section 5.2.3. La structure de données utilisée pour les ioctls ainsi que les types d'ioctls sont les suivants :

```
/*
 * struct for user/kernel communication
 * According to the ioctl type, certain fields are the parameters provided
 * to the ioctl request, while other fields are completed at the return of
 * the ioctl system call.
 */
typedef struct {
        char    iname[IFNAMSIZ];        /* input interface name */
        char    oname[IFNAMSIZ];        /* output... */
        label_t ilabel;                 /* input label on the input if */
        label_t olabel;                 /* output... */
        int     status;                 /* error code returned */
} rswreq_t;

/*
 * ioctls for rsvp_sw driver
 */
#define RSWIOC_GETSTATS         _IOR('R', 4, int)
#define RSWIOC_GETSTATE         _IOR('R', 5, int)
#define RSWIOC_ADDSHORTCUT      _IOWR('R', 6, rswreq_t)
#define RSWIOC_RMSHORTCUT       _IOWR('R', 7, rswreq_t)
#define RSWIOC_ADDSESSION       _IOWR('R', 8, int)
#define RSWIOC_RMSESSION        _IOWR('R', 9, int)
#define RSWIOC_CHANGESESSION    _IOWR('R', 10, int)
#define RSWIOC_GETLABEL         _IOWR('R', 11, rswreq_t)
#define RSWIOC_FREELABEL        _IOWR('R', 12, rswreq_t)
```

5.6 Configuration de la plate-forme ATM

La configuration de la plate-forme ATM pour les expérimentations envisagées est présentée sur la Figure 5.4. Sur ce schéma seules les connexions ATM sont représentées pour les stations disposant d'une interface ATM directe. Toutes les stations sont aussi rattachées à travers des réseaux locaux classiques (*Ethernet*) à l'Internet. Nous pouvons ainsi réaliser un grand nombre de configurations protocolaire afin de comparer les différentes solutions envisagées :

- Topologie locale ou longue distance.
- Comparaison de RSVP Switching par rapport à RSVP + IP/Ethernet.
- Comparaison de RSVP Switching par rapport a RSVP + IP/LANE.

Des stations communiquent indirectement avec la plateforme en utilisant RSVP non modifié sur réseau classique afin de démontrer la compatibilité et l'interfonctionnement avec les protocoles existants.

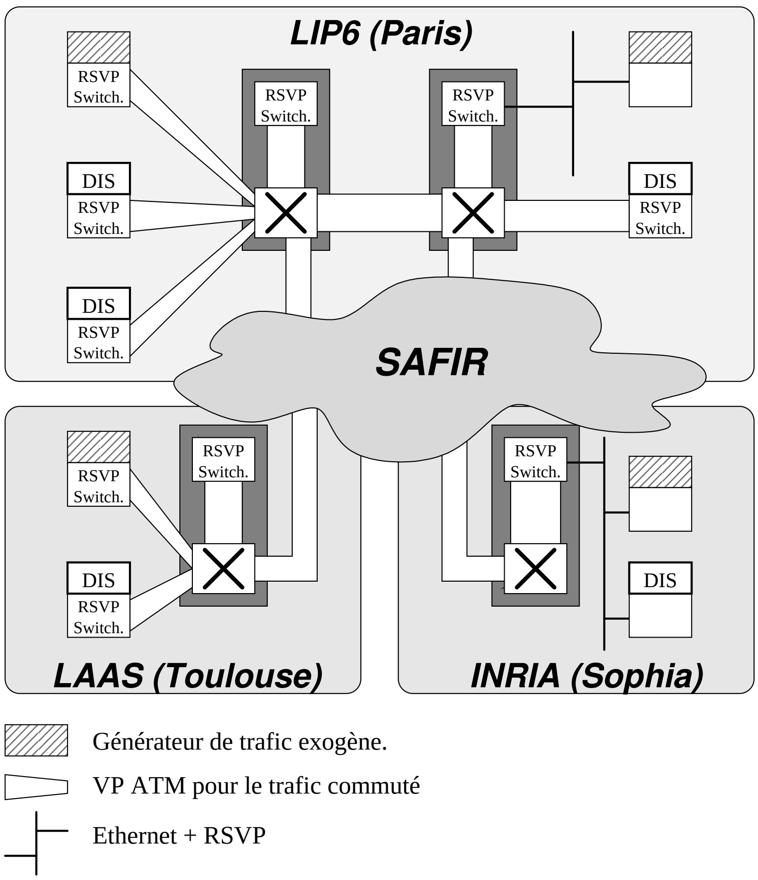


FIG. 5.4: Configuration de la plate-forme ATM

Chapitre 6

Conclusion

L'intérêt de l'approche RSVP Switching réside dans la possibilité d'utiliser les performances de la commutation ATM sans être limité par le mode connecté usuel de l'ATM. La possibilité d'obtenir le faible temps de latence de la commutation de cellules et des débits élevés, tout en utilisant directement la signalisation IP, nous laisse penser qu'une intégration efficace du multicast avec QoS est possible avec cette nouvelle approche de l'intégration d'IP sur ATM.

Nous proposons, à travers RSVP Switching, d'utiliser directement la commutation ATM au niveau de la couche liaison avec IPv6 au niveau de la couche réseau. Cela nous permet de gérer efficacement le modèle de communication multipoint-à-multipoint avec QoS utilisé par RSVP et nécessaire aux applications DIS.

Les limitations de notre architecture vont plutôt se trouver au niveau de la partie contrôle où la gestion d'un nombre important d'états est difficile. La solution RSVP Switching est plus particulièrement adaptée aux environnements limités en terme de taille du groupe, tels que les VPN, mais non limité en terme de distance grâce à ATM. L'approche RSVP Switching est dédiée aux environnements où les garanties de QoS sont plus importantes que le nombre de flots.

Pour valider expérimentalement notre solution nous implémentons un prototype pour le projet DIS/ATM, sur notre plate-forme ATM locale et sur la plateforme ATM nationale SAFIR. Les limitations d'accès à la commutation ATM nous amènent à proposer un modèle émulant la commutation mais supportant les applications DIS avec de fortes contraintes en terme de QoS.

Bibliographie

[AAOS98] Arup Acharya, Furquan Ansari, Max Ott, and Henning Sanneck. Dynamic QoS for IP switching using RSVP over IPSOFACTO. In *Proceedings of SYBEN '98*, Zurich, May 1998.

[Alm96] Werner Almesberger. ATM on Linux. Technical report, Laboratoire de Réseaux de Communication (LRC), École Polytechnique Fédérale de Lausanne (EPFL), CH-1015 Lausanne, Switzerland, March 1996.

[BBD+97] E. Basturk, A. Birman, G. Delp, R. Guérin, et al. Design and Implementation of QoS Capable Switch-Router. Technical Report RC 20848 (01/31/97) Computer Science, IBM Research Division, January 1997.

[Bou96] Jean-Yves Le Boudec. Network Calculus Made Easy. Technical Report EPFL-DI 96/128, Laboratoire de Réseaux de Communication (LRC), École Polytechnique Fédérale de Lausanne (EPFL), CH-1015 Lausanne, Switzerland, December 1996.

[BR96] F. Baker and Y. Rekhter. Tag Switching with RSVP. Internet Draft `<draft-baker-tag-rsvp-00.txt>`, work in progress, IETF, December 1996.

[BZ97] R. Braden and L. Zhang. Resource ReSerVation Protocol (RSVP) – Version 1 Message Processing Rules. Request For Comment 2209, IETF, September 1997.

[BZB+97] R. Braden, L. Zhang, S. Berson, S. Herzog, and S. Jasmin. Resource ReSerVation Protocol (RSVP) – Version 1 Functional Specification. Request For Comment 2205, IETF, September 1997.

[CDD+98] C. Chassot, L. Dairaine, M. Diaz, A. Lozes, and L. Rojas. Réservation de ressources pour applications DIS en environnement Internet nouvelle génération. Technical report, LAAS-CNRS, 1998.

[Cru96] R. Cruz. Quality of Service Guarantees in Virtual Circuit Switched Networks. *IEEE Journal on Selected Areas in Communications*, 13(6):1048–1056, August 1996.

[Die98] Tomas Dietz. Label Swapping with RSVP in an IP over ATM Environnement. Diplomarbeit am Institut für Telematik, Fakultät für Informatik - Universität Karlsruhe, May 1998.

[DKS89] Alan Demers, Srinivasan Keshav, and Scott Shenker. Analysis and Simulation of a Fair Queueing Algorithm. In *Proceedings of ACM SIGCOMM '89*, volume 19, Austin, Texas, USA, September 1989. ACM Computer Communication Review.

[DRV+98] B. Davie, Y. Rekhter, A. Viswanathan, S. Blake., V. Srinivasan, and E. Rosen. Use of Label Switching With RSVP. Internet Draft `<draft-ietf-mpls-rsvp-00.txt>`, work in progress, IETF, March 1998.

[Dul97] Didier Dulac. Spécification de l'application DIS. Rapport provisoir de projet, Dassault Electronique, May 1997.

[FD97] Olivier Fourmaux and Christophe Deleuze. Qualité de service hétérogène pour liaisons ATM point-à-multipoint. In *Actes de CFIP'97*, pages 383–398, Liège, Belgium, September 1997. Université de Liège, Hermes.

[FJ95] Sally Floyd and Van Jacobson. Link-Sharing and Ressource Management Models for Packet Networks. *IEEE/ACM Transactions on Networking*, 3(4):365–386, August 1995.

[Ga96] L. Georgiadis and al. Efficient support of delay and rate guarantees in an internet. In *Proceedings of ACM SIGCOMM '96*, volume 26, Palo Alto, California, USA, September 1996. ACM Computer Communication Review.

[LAN95] Lan emulation over atm – version 1.0 specification. Technical Report af-lane-00021.000, ATM Forum, January 1995.

[LH98] M. Laubach and J. Halpern. Classical IP and ARP over ATM. IETF Request For Comment 2225, Com21, Inc. and Newbridge Networks, Inc., April 1998. *Remplace les RFC 1577 et 1626.*

[NEH+98] P. Newman, W. Edwards, R. Hinden, E. Hoffman, F. Ching Liaw, T. Lyon, and G. Minshall. Ipsilon's General Switch Management Protocol Specification Version 2.0. IETF Informational Request For Comment 2297, Nokia, Sprint, Fiberlane, March 1998.

[SPG97] S. Shenker, C. Partridge, and R. Guerin. Specification of Guaranteed Quality of Service. Request For Comment 2212, IETF, September 1997.

[VSB97] A. Viswanathan, V. Srinivasan, and S. Blake. Soft State Switching – A Proposal to Extend RSVP for Switching RSVP Flows. Internet Draft `<draft-viswanathan-mpls-rsvp-00.txt>`, work in progress, IBM Corporation, August 1997.

[Wro97a] J. Wroclawski. Specification of the Controlled-Load Network Element Service. IETF Request For Comment 2211, MIT LCS, September 1997.

[Wro97b] J. Wroclawski. The Use of RSVP with IETF Integrated Services. IETF Request For Comment 2210, MIT LCS, September 1997.