Architecture de QoS en environnement IPv6 à services différenciés

[*]C Chassot, [*]F Garcia, [*]A Lozes, [°]P Anelli, [**]T Bonald

[*]*LAAS/CNRS*
7 av du Colonel Roche
31077 Toulouse cedex 04, France
chassot, alozes, fgarcia@laas.fr

[°]*LIP6*
8, rue du capitaine Scott
75015, Paris, France
pascal.anelli@lip6.fr

[**]*France Télécom R&D*
38-40 rue du Général Leclerc
92794 Issy les Moulineaux cedex, France
thomas.bonald@.francetelecom.fr

Résumé

Face aux besoins en communication des nouvelles applications distribuées (multimédias et coopératives, simulation interactive, etc.), la communauté Internet a initié un certain nombre de travaux ayant pour objectif le développement d'une nouvelle génération de protocoles amenés à être mis en œuvre dans le cadre d'une nouvelle architecture de l'Internet. Les travaux présentés dans cet article se situent dans ce double contexte applicatif et réseau et portent sur la conception d'une architecture de communication à qualité de service garantie en environnement IPv6 à services différenciés. L'approche proposée repose sur la mise à disposition des applications, via une interface de programmation générique, d'une qualité de service par flux de donnée applicatif, tout en gérant une qualité de service par classe de trafic au niveau IP.

Mots clés

Architecture de communication, qualité de service, Internet à service différenciés, API

Abstract

Advanced distributed applications (multimedia and cooperative applications, distributed interactive simulation, etc.) have both new features and new constraints in terms of communication requirements. To tackle these needs, several projects have been initiated within the Internet community to design a new generation of protocols for the future Internet's architecture. The research reported here deals with the design of a new communication architecture with guaranteed end to end quality of service (QoS) in an IPv6 environment providing differentiated services. The proposed approach uses a generic programming interface (API) allowing the application to specify a QoS per application flow, while a QoS per class of traffic is managed at the IP level.

Keywords

Communication architecture, quality of service (QoS), Differentiated services, Internet, API

1. Introduction

Ces dernières années, les évolutions technologiques conjointes de l'Informatique et des Télécommunications ont conduit au développement de nouveaux types d'applications distribuées : applications multimédias et coopératives, simulation interactive, etc., présentant des caractéristiques et des contraintes nouvelles : hauts débits, délai de transit borné, etc. Face à ces nouveaux besoins, la communauté Internet (par le biais initialement des groupes IntServ [1] et DiffServ [2] de l'IETF[1]) a initié un certain nombre de travaux ayant pour objectif le développement d'une nouvelle génération de protocoles amenés à être mis en œuvre dans le cadre d'une nouvelle architecture de l'Internet TCP/IP. L'un des points clefs de cette architecture concerne la définition de nouveaux types de services (« intégrés » ou « différenciés » selon l'approche) destinés à répondre à l'ensemble des besoins requis par les applications multimédias.

Les travaux présentés dans cet article se situent dans ce double contexte applicatif et réseau et portent plus précisément sur la conception d'une architecture de communication à qualité de service (QoS : Quality of Service) garantie en environnement IP à services différenciés.

Ces travaux ont été réalisés dans le cadre du projet @IRS[2] dont l'objectif est de développer et d'expérimenter les nouveaux protocoles de l'Internet ainsi que les fonctions associées qui permettront, d'une part, d'offrir aux usagers des services adaptés à leurs besoins quel que soit le point d'accès (fixe ou mobile) et, d'autre part, de tirer partie d'une infrastructure de télécommunication de plus en plus performante et hétérogène (ATM, satellite, réseaux sans fil, réseaux Locaux, etc.). Initié en décembre 1998, ce projet s'inscrit sur une période de vingt sept mois au terme de laquelle les solutions étudiées et développées seront démontrées sur une plate-forme ATM nationale (RENATER 2), à l'aide d'applications innovantes (simulation interactive distribuée et contrôle/commande notamment) représentatives des besoins technologiques futurs.

Le projet @IRS fait suite à un autre projet national (le projet DIS/ATM [3]), dont l'objectif était de concevoir et d'expérimenter une architecture de communication, apte à supporter les besoins des applications de simulation interactive distribuées (DIS : Distributed Interactive Simulation) en environnement de réseaux hétérogènes : réseaux locaux et plate-forme nationale ATM (SAFIR, aujourd'hui RENATER2) interconnectés en IPv6 à services intégrés (IntServ/RSVP). Afin de comprendre les choix d'architecture présentés dans cet article, nous résumons très brièvement certains des travaux réalisés dans le projet DIS/ATM en soulignant leurs limites vis à vis du contexte ciblé dans le projet @IRS.

1.1. Travaux réalisé dans le projet DIS/ATM

L'une des tâches du projet DIS/ATM consistait d'une part à étudier les besoins des applications de DIS, d'autre part à définir l'architecture de bout en bout permettant de prendre en compte les besoins en communication des applications DIS et de déployer ces mêmes applications dans un environnement réseau de type IPv6 IntServ/RSVP. Nous ne résumons ici que la seconde partie de ces travaux. Précisons seulement que l'étude des besoins des applications

[1] IETF : Internet Engineering Task Force

[2] @IRS : « Architecture Intégrée de Réseaux et Services ». Projet du RNRT (Réseau National de la Recherche en Télécommunications).

DIS nous a conduit à proposer une répartition des PDU DIS en classes de PDU nécessitant chacune une certaine QoS en terme de fiabilité et/ou de délai de transit, pour que l'application soit exécutée correctement [4,5].

1.1.1. L'architecture DIS/ATM

L'architecture de bout en bout du projet DIS/ATM, que nous appellerons dorénavant « architecture DIS/ATM » (cf. Figure 1), a été définie de façon :

– à supporter les besoins des applications DIS du projet ;

– à être mise en œuvre dans un environnement Transport/Réseau correspondant à UDP et IPv6 incluant les services intégrés définis par l'IntServ [6,7,8,9].

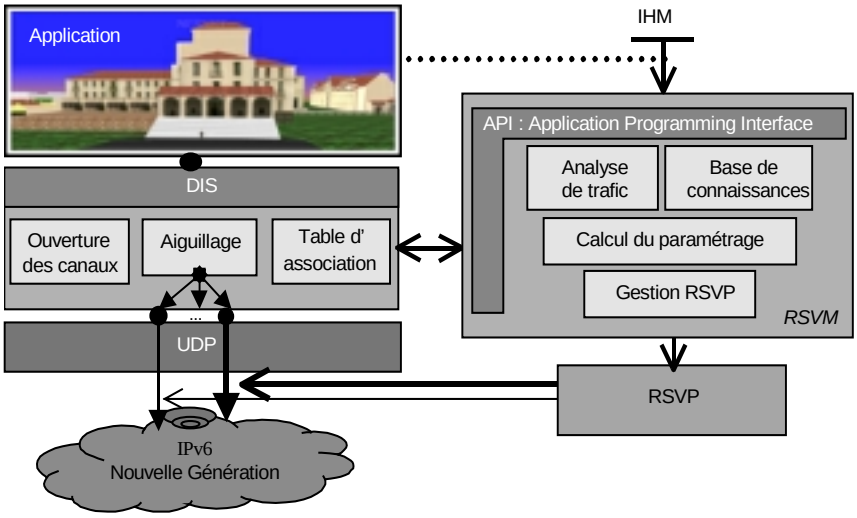

Figure 1 : *Architecture de bout en bout issue du projet DIS/ATM*

Les principes de cette architecture sont :

– de doter les applications DIS d'un support de communication de bout en bout avec une gestion de la QoS s'appuyant sur les mécanismes IntServ et le protocole RSVP ;

– de gérer plusieurs canaux de communication de bout en bout *multicast*, destiné chacun à diffuser une classe de PDU réclamant une même QoS ;

– de rendre aussi transparente que possible la réservation de ressources pour chaque canal en dérivant la caractérisation du trafic requise par RSVP (*TSPEC*) à partir d'informations de niveau applicatif. Ceci est réalisé par le RSVM (Resource reSerVation Manager) ;

– d'aiguiller chaque PDU vers le canal adéquat à partir de la seule information que contient le paquet. Cette fonction est réalisée à l'intérieur du module DIS modifié.

1.1.2. Limites de ces travaux

Telle qu'elle a été définie, l'architecture DIS/ATM présente plusieurs limites vis à vis de la problématique générale posée dans le projet @IRS, parmi lesquelles les deux suivantes :

– l'architecture DIS/ATM est fortement orientée DIS ; en d'autres termes, elle ne permet pas de prendre directement en compte les autres applications ciblées par le projet @IRS ;

– l'étude initiale conduit à un choix relativement figé des services de Transport (réduit à UDP) et des services de gestion de la QoS au travers du réseau (réduits au service garanti de l'IntServ) ; l'un des objectifs majeurs du projet @IRS est d'étudier la possibilité d'utiliser les services différenciés (DiffServ).

1.2. Contenu et plan de l'article

Les contributions présentées dans cet article ont été réalisées dans l'objectif de répondre aux deux limites précédentes.

L'objectif général sous-jacent à la démarche adoptée est de définir une architecture de bout en bout qui soit à la fois apte à prendre en compte les besoins en QoS des applications du projet (DIS et contrôle/commande notamment), mais aussi générique, c'est à dire indépendante des applications, en offrant une seule et même interface de programmation (API : *Application Programing Interface*) de niveau Transport. Ce second objectif vise à offrir au programmeur la plus grande transparence possible dans l'accès au système de communication et à assurer aux applications une portabilité immédiate vis à vis des différents mécanismes de gestion de la QoS, de niveau Réseau ou Transport.

A ces fins, plusieurs études sont ou ont été menées dans le projet, parmi lesquelles :
– les modifications à apporter aux applications existantes ;
– la définition de l'architecture de QoS de bout en bout ;
– la définition de l'architecture de QoS de niveau IP ;
– le lien entre les architectures de QoS de bout en bout et de niveau IP ;

Cet article décrit les premiers résultats de ces études. La suite de l'article est structurée de la façon suivante : la section 2 expose l'architecture de bout en bout définie pour le projet @IRS ; la section 3 expose l'architecture de QoS de niveau IP ainsi que le lien entre QoS de bout en bout et QoS de niveau IP ; les conclusions et perspectives de ces travaux sont enfin exposées en section 4.

2. Architecture de QoS de bout en bout

Le principe de base qui sous-tend la proposition d'architecture de QoS de bout en bout rejoint celui de plusieurs autres, dédiées au transport de flux multimédias [10,11,12,13]. L'idée est que le trafic échangé dans le cadre d'une application distribuée peut être éventuellement décomposé, au niveau applicatif, en plusieurs flux de données réclamant chacun des besoins spécifiques en communication (délai, fiabilité, ordre, …). En d'autres termes, il s'agit pour l'application de pouvoir réclamer (et se voir offrir), par flux applicatif, une QoS spécifique, et ce, par le biais d'une API offrant les paramètres et primitives de service nécessaires.

Avant de décrire cette architecture, nous présentons brièvement les modifications minimales à apporter aux applications pour s'y intégrer.

2.1. Modifications des applications existantes.

Deux types de modifications ont été définis (cf. Figure 3) : (1) la traduction des appels au service de Transport utilisé (actuellement UDP) en invocation des primitives de service de la nouvelle API ; (2) l'insertion d'une couche d'adaptation réclamant l'établissement du(des) canal(aux) de communication de bout en bout ainsi que la QoS associée, et traduisant les besoins exprimés en termes applicatifs en des paramètres génériques reconnus par l'API.

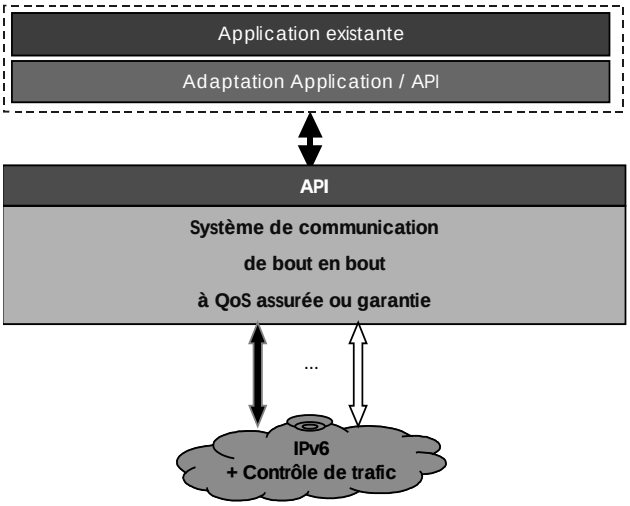

Figure 2 : *Modifications au niveau applicatif*

2.2. Description générale de l'architecture

L'architecture de QoS de bout en bout a été définie de façon à répondre à deux objectifs : (1) satisfaire les besoins des applications étudiées dans le projet, et de façon générale, de tous les types d'applications présentant des exigences de QoS, en particulier temporelles ; (2) banaliser l'accès aux services de niveau Transport et de niveau Réseau, ce qui se traduit par les trois points suivants :

– choix d'un modèle de trafic unique pouvant être traduit en différents groupes de paramètres selon la description du trafic imposée par le contrôle d'admission associé aux services de niveau IP ;

– définition des paramètres de QoS permettant de choisir un service ou une classe de service de niveau IP ;

– mise à disposition des applications d'une API unique.

L'architecture proposée (Figure 3) permet aux applications (via l'API) d'établir des sessions entre deux ou plusieurs participants.

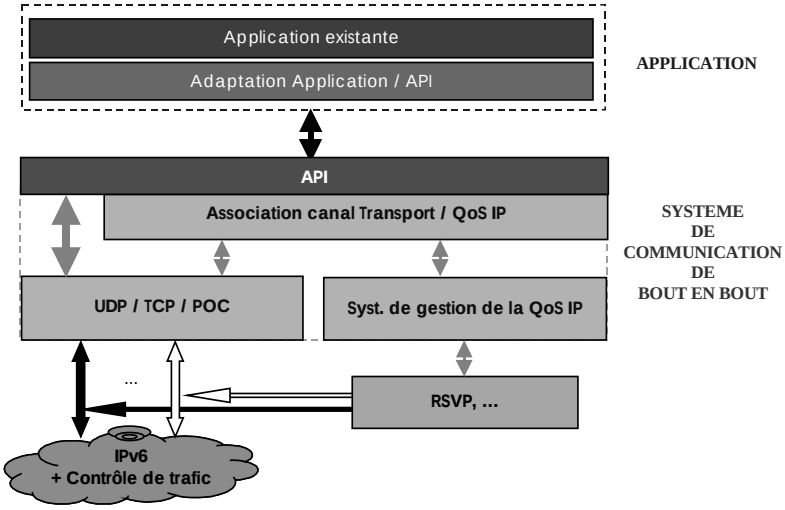

Figure 3 : *Architecture du système de communication de bout en bout*

Au travers d'une session, le logiciel applicatif peut réclamer l'établissement d'un ou plusieurs canaux de communication de bout en bout :

– *unicast* ou *multicast* ;

– dédié chacun au transfert d'un flux de données applicatif ;

– offrant une QoS spécifique au flux véhiculé.

Outre l'API, trois modules sont alors conceptuellement identifiables au travers de l'architecture proposée :

– un module chargé de la mise en œuvre des fonctionnalités de niveau Transport : UDP, TCP ou protocole à ordre et fiabilité partiels (POC : *Partial Order Connection* [10]) par exemple ;

– un module chargé de la mise en œuvre des mécanismes liés à l'utilisation des services de QoS de niveau IP (pilotage de RSVP par exemple) ;

– un module chargé de l'établissement des canaux de bout en bout et de leur mise en correspondance avec le système de QoS au niveau IP.

2.3. Service

Tel qu'introduite précédemment, l'architecture proposée permet aux applications d'établir des sessions entre deux ou plusieurs participants, au travers desquelles sont établis un ou plusieurs canaux de communication de bout en bout, *unicast* ou *multicast*, chaque canal étant dédié au transfert d'un flux de données applicatif et offrant une QoS spécifique au flux véhiculé. Le service proposé offre un mode d'accès « avec connexion » et le mode de transfert est le mode « message », le système de communication de bout en bout préservant en réception, pour chaque flux, la frontière des données.

2.3.1. QoS : paramètres et sémantiques de garantie

La QoS d'un canal est négociée, indépendamment pour chaque sens du canal, lors de la création de ce dernier. Elle comporte comme paramètres principaux :

– une fiabilité partielle, exprimée par exemple en nombre maximum de pertes consécutives ou en pourcentage de perte ; implicitement, toute donnée remise à une application réceptrice est garantie sans erreur bit ;

– un délai de transit maximal ;

– un ordre partiel, pouvant être exprimé à la fois intra et inter flux selon que l'utilisateur souhaite disposer d'un service de synchronisation logique intra canal ou inter canal (synchronisation logique de la voix et de la vidéo par exemple si les deux médias sont véhiculés sur des canaux distincts).

A ces paramètres sont associées deux sémantiques de garantie :

– la garantie « dure », imposant au système de communication de respecter la valeur nominale des paramètres de QoS ;

– la garantie « statistique »[3], permettant au système de communication d'avoir une certaine marge d'erreur dans la satisfaction de la QoS.

2.3.2. Autres paramètres de service

Outre les paramètres de QoS, l'application doit spécifier un certain nombre de paramètres nécessaires à la mise en œuvre du service de communication de bout en bout.

– Le premier paramètre permet de caractériser le trafic que va générer l'application émettrice : le modèle retenu est celui du saut à jetons (ou « *Token Bucket »).*

– Le second paramètre désigne le protocole de Transport que l'application souhaite voir activer. En effet, si l'une des perspectives de nos travaux concerne la sélection automatique du service de Transport par le système de communication, ce choix est pour l'instant laissé à l'application.

– Le troisième paramètre désigne le système de gestion de la QoS de niveau IP que l'application souhaite voir activer. Là encore, la sélection automatique de ce service (combinée à celle du service de Transport) constitue une perspective de nos travaux, hors du cadre du projet @IRS.

– Le dernier paramètre identifie l'adresse (*unicast* ou *multicast*) du ou des logiciels applicatifs destinataires.

2.3.3. L'interface de programmation de niveau applicatif (API)

Pour accéder au service du système de communication, l'application dispose d'une API spécifique, composée d'un ensemble de primitives destinées à la gestion :

– des sessions et des flux ;

– du transfert des données ;

– de la QoS.

[3] Une expression possible de cette sémantique est par exemple que D % des données soumises par l'application satisfassent la QoS nominale requise.

Ces primitives comportent un certain nombre de paramètres de service que nous présentons brièvement ci-après. Les primitives de l'API sont exposées dans un second temps.

a) Paramètres de service

Les paramètres de service peuvent être répartis en 4 familles :
– les paramètres de désignation de session et de flux ;
– les paramètres de description de la QoS ;
– les paramètres de description du trafic ;
– les paramètres de choix explicite du protocole de Transport et du système de gestion de la QoS de niveau IP.

Paramètres de désignation des sessions et des flux

Pour référencer une session sur un hôte ou un flux dans une session, les applications utilisent des identificateurs entiers. Un identificateur est unique dans son contexte (i.e. hôte pour une session, session pour un flux).

Paramètres de QoS

Pour chaque flux (et pour chaque sens d'un flux), l'application peut réclamer une certaine QoS. Cette QoS est définie par :
– une fiabilité partielle ;
– un ordre partiel sur les objets (i.e. les paquets) du flux considéré ;
– un délai de bout en bout.

De plus, une application peut réclamer un ordre partiel entre les flux d'une session plutôt que sur les objets d'un seul flux.

Expression de la fiabilité partielle. Il existe plusieurs représentations possibles de la fiabilité. Parmi celles ci, les plus simples à mettre en œuvre (que nous avons retenues dans le projet @IRS) sont :
– une fiabilité exprimée en nombre maximum de pertes d'objets consécutives ;
– une fiabilité exprimée en pourcentage minimal de paquets à recevoir.

Expression de l'ordre partiel. De même que dans [14], l'ordre partiel (supposé périodique) est décrit par un tableau de taille égale à une période. Pour chaque objet, on range dans ce tableau la liste de ses prédécesseurs comme indiqué par l'exemple de la figure 4.

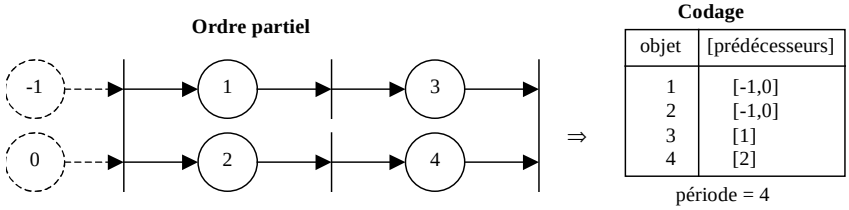

Figure 4 : *Exemple de codage d'ordre partiel*

Expression du délai de bout en bout. Le délai de bout en bout est exprimé en terme de délai applicatif (délai entre l'émission et la délivrance des données à l'application distante, à

l'asynchronisme du système près). On peut aussi exprimer un délai non borné si l'application ne présente pas d'exigence temporelle particulière.

Paramètres de description du trafic

Pour chaque flux, l'application doit décrire le trafic qu'elle compte générer ; pour cela, elle utilise un TSPEC (*Traffic Specification*) de façon analogue à celui de RSVP. Ce TSPEC est composé :

– d'un débit crête ;
– d'un *Token Bucket* comprenant un débit moyen et une taille maximale de rafale.

Choix explicite du Transport et du système de QoS Réseau

L'un des objectifs de l'architecture de bout en bout est de masquer le choix du Transport et du système de QoS de niveau IP à l'application. Cependant, on laisse la possibilité au programmeur d'une application de demander explicitement[4] l'utilisation d'un protocole de Transport ou d'un système de gestion de la QoS IP.

b) Primitives de service

Dans cette section, nous exposons brièvement les principales primitives qui sont offertes à l'application pour accéder aux système de communication.

Gestion des sessions et des flux

Cet ensemble de primitives permet de créer ou de détruire une session ou un flux. En outre, il est possible d'obtenir des informations sur un flux (QoS associée par exemple).

Ouverture de la session. La primitive ***open_session()*** permet de créer une session. C'est la première primitive que doit utiliser le programmeur d'une application.

Fermeture de la session. La primitive ***close_session()*** permet de fermer une session, en libérant toutes les ressources qui lui sont associées.

Ajout d'un flux dans une session. La primitive ***add_flow()*** permet de créer un flux et de l'ajouter à la liste des flux d'une session. L'appel à cette primitive déclenche l'établissement de la connexion et la mise en place du système de QoS.

Un flux étant défini comme une association de niveau Transport à laquelle on attache un profil de trafic, une QoS en émission et une QoS en réception, il est nécessaire, lors de la création du flux, de fournir au système de communication les informations suivantes :

– ***ses_id*** : identifiant de la session à laquelle le flux doit être ajouté ;
– ***local_addr*** : adresse IPv6 du hôte local (structure **sockaddr_in6**) ;
– ***dist_addr*** : adresse IPv6 du hôte distant (structure **sockaddr_in6**) ;
– ***Tspec*** : pointeur vers un profil de trafic de type **t_Tspec** ;

[4] Ce choix doit être fait en tenant compte des contraintes existant entre les différents paramètres de QoS. Ainsi un programmeur ne peut pas forcément requérir une fiabilité totale en même temps qu'un délai borné, ou demander l'utilisation du protocole UDP si l'application requiert une fiabilité absolue . A terme, cette vérification sera faite par le système de communication qui devra rejeter toute demande dont il saura qu'elle ne peut être satisfaite.

– ***QoS_E*** : QoS requise par l'application pour le flux en émission ;
– ***QoS_R*** : QoS requise par l'application pour le flux en réception.

Les autres paramètres de cette primitive servent à nommer le flux (***label***) et à expliciter le choix du protocole de Transport (***Transport***) et du système de fourniture de QoS Réseau (***QoS_sys***).

En retour, la primitive fournit à l'application l'identificateur du flux créé.

L'appel à la primitive ***add_flow()*** peut échouer pour plusieurs raisons :
– session inexistante ;
– destinataire inconnu ou injoignable ;
– ressources réseau insuffisantes pour satisfaire la demande en QoS ;
– inconsistance de la demande de QoS.

Destruction d'un flux. Quand un flux n'est plus utilisé par une application, il faut libérer les ressources qui lui sont associées. C'est le rôle de la primitive ***del_flow()***.

Gestion de la QoS

Les primitives ***add_flow()*** et ***del_flow()*** font aussi parties de ce groupe de primitives car elles mettent en place et libèrent les ressources réseau nécessaires à la satisfaction des besoins de l'application

Déclaration de l'ordre partiel entre les flux. Dans l'architecture de communication, il est prévu de pouvoir définir un ordre partiel sur les objets d'un flux (ordre « intra flux »), ainsi que sur plusieurs flux d'une session (ordre « inter flux »). La primitive ***set_poc_on_flows()*** permet de définir l'ordre partiel inter flux.

Gestion du transfert de données

Ces primitives sont similaires aux primitives de transfert de données disponibles dans les API Winsock et socket.

Envoi de données. La primitive ***send_data()*** permet l'envoi de données sur un flux d'une session.

Réception non bloquante de données. La primitive ***recv_data()*** permet de lire les données présentes sur un flux d'une session.

Réception bloquante de données. La primitive ***wait_data()*** permet d'attendre « un certain temps » des données sur un ou plusieurs flux d'une même session.

3. Architecture de QoS au niveau IP

3.1. Services

Ces dernières années, une réflexion sur le nombre et sur les caractéristiques des services à offrir au niveau IP a été (en particulier) menée par le groupe de travail Intserv de l'IETF. Cette étude a conduit à la définition de trois modèles de service, qui restent pertinents de nos jours. L'architecture proposée dans [15,16] s'appuie également sur un découpage des services en trois catégories, que nous avons repris pour @IRS.

Les services ainsi proposés au niveau IP sont :

– un service garanti (*GS : Garanteed Service*) pour les flux de données ayant des contraintes fortes en termes de délai et de fiabilité et ne supportant pas de variation de ces paramètres (flux non réactifs). Notons que ce service présente une propriété particulièrement intéressante dans le contexte du *multicast* dans la mesure où il garantit un très faible taux de pertes de paquets ;

– un service assuré (*AS : Assured Service*) destiné aux flux éventuellement réactifs n'ayant pas de contraintes fortes en termes de délai de transit, mais requérant une « garantie » de débit moyen minimum (*d*). Ce débit se décompose en deux parties :

– une partie fixe, qui correspond au débit minimum assuré d_a. En cas de congestion conjoncturelle dans le réseau, les paquets de cette partie sont marqués comme inadéquats pour la perte ;

– une partie « élastique », égale à (d - d_a) et constituée de paquets dits « opportunistes », dont l'application est censée diminuer le débit en cas de congestion dans le réseau. En cas de congestion, ces paquets opportunistes sont éliminés en premier. Notons que pour un flux non réactif, le débit se limite à la partie fixe.

– un service « au mieux » (*BE : Best Effort*) n'offrant aucune garantie de QoS.

Les classes de service GS et AS permettent donc d'effectuer une différenciation qualitative entre les flux en fonction de leurs exigences de QoS en terme de délai de transit ou de débit moyen minimum.

3.2. Lien entre QoS de bout en bout et QoS de niveau IP

L'architecture décrite en section (2.2) permet de réclamer la mise en œuvre d'une QoS « par flux », chaque canal de bout en bout devant fournir une QoS spécifique au flux véhiculé. Ceci étant, le contrôle de trafic effectué au niveau IP pour offrir les trois services précédemment décrits repose sur la manipulation de paquets IP marqués par leur appartenance à une « classe » de service au sens DiffServ (champ DSCP des paquets IPv6 [17]) : la question se pose alors de savoir comment réaliser le lien entre QoS de bout en bout « par flux » et QoS de niveau IP « par classe ».

Dans cette section, nous décrivons la solution proposée dans le cadre du projet pour établir ce lien, ainsi que les mécanismes mis en œuvre pour offrir les services définis dans la section précédente. Dans un premier temps, nous décrivons l'une des plates-formes d'expérimentation définies dans le projet @IRS. Le principe de la solution proposée est décrit dans un second temps. Son application et la mise en œuvre des différents services de niveau IP sont explicités en dernier lieu.

3.2.1. Plate-forme d'expérimentation

La plate-forme d'expérimentation sur laquelle seront déployées les applications DIS du projet est illustrée figure 5 :

– les plates-formes locales sont raccordées à l'ISP (Internet Service Provider) que représente la plate-forme ATM RENATER2 par le biais de routeurs de bordure (R_b) ;

– quatre routeurs de cœur (R_c) sont introduits dans l'ISP. Physiquement, ces routeurs sont situés sur les plates-formes locales, mais ils appartiennent logiquement à l'ISP. Les

commutateurs ATM raccordant les routeurs de cœur à la plate-forme RENATER2 ne sont pas représentés sur la figure.

Afin de simplifier les explications fournies ci-après, les plates-formes locales y sont volontairement définies de façon minimale et quatre sites uniquement y sont représentés.

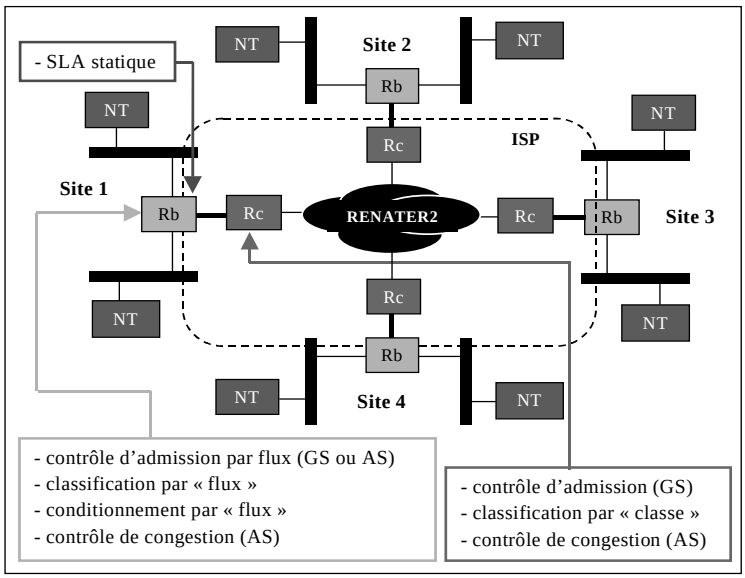

Figure 5 : *Plate-forme d'expérimentation*

3.2.2. Principe de la proposition

Chaque site (cf. Figure 5) dispose par le biais de son routeur de bordure d'un point d'entrée à l'ISP, caractérisé par un certain contrat de trafic (appelé « SLA : Service Level Agreement » dans [16]) établi statiquement, et constitué pour chaque classe de service :

– des règles de classification et de re-marquage éventuel des paquets ;

– du profil de trafic à respecter en émission (appelé « TCA : Traffic Conditioning Agreement » dans [16]) ; le modèle ici retenu étant celui *Token Bucket* ;

– des actions entreprises en cas de non respect du profil annoncé.

C'est au routeur de bordure de mettre en œuvre ce SLA lorsqu'il a à introduire des flux applicatifs dans l'ISP.

a) Rôle des routeur de cœur.

Indépendamment du service (GS ou AS) à mettre en œuvre, le contrôle de trafic appliqué au sein de l'ISP (i.e. dans les routeurs de cœur) est effectué par classe (le marquage, ayant eu lieu au niveau des routeurs de bordure), que ces paquets appartiennent ou non à un même flux. Trois marques ont été définies (EF : Expedited Forwarding, AF : Assured Forwarding et BE : Best Effort, en référence à [18] et [19]), chacune correspondant à l'un des services supportés.

Plus précisément, le contrôle de trafic mis en œuvre au niveau des routeurs de cœur comportent les fonctions principales suivantes (cf. Figure 6) :

– un contrôle d'admission (pour le service GS uniquement) ;

– une classification de type BA (*behavior aggregate*) : tous les paquets marqués identiquement (même valeur du champ DSCP) sont rangés dans une même file d'attente (trois files d'attente sont donc définies) ;

– un ordonnancement des trois files précédentes couplant mécanismes de priorité (PQ : *priority queuing*) et « *weighted fair queuing* » (WFQ)

– un contrôle de congestion (pour le service AS uniquement), destiné à faire diminuer le débit des paquets opportunistes en cas de saturation d'un ou plusieurs routeur de cœur dans l'ISP ; ce contrôle inclut en particulier un mécanisme de rejet sélectif à seuil de type « *partial buffer sharing* » (PBS) [20].

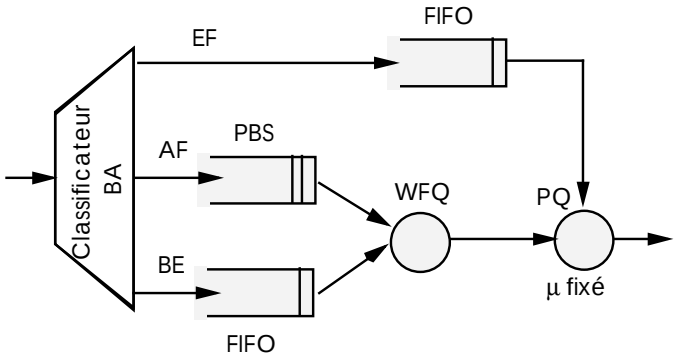

Figure 6 : *Eléments fonctionnels d'un routeur de cœur*

b) Rôle des routeur de bordure.

Au sein des sites, les trois services proposés sont également accessibles par les hôtes, qui doivent pour cela (excepté pour le BE) s'adresser à leur routeur de bordure, faisant ici office de « *Bandwidth Broker* » (tel que défini dans [15]).

Du point de vue d'un routeur de bordure (cf. Figure 7), la disponibilité des services AS ou GS est fonction :

– du contrat de trafic (TCA) négocié de façon statique avec le routeur de cœur pour le service considéré ;

– de l'état courant d'utilisation du service considéré par le site.

Ceci étant et indépendamment du service choisi, le contrôle de trafic appliqué par les routeur de bordure est effectué par flux. En conséquence et de façon à respecter le TCA local : il est nécessaire de mettre en œuvre un contrôle d'admission, que le service requis soit de type AS ou GS.

Les autres fonctions des routeurs de bordure sont les suivantes :

– un conditionnement du trafic par flux, c'est à dire :

– un marquage Diffserv des paquets fonction de la QoS requise pour le flux (EF, AF et BE) ;

– une vérification de la conformité du trafic à la caractérisation annoncée par l'application (le modèle état celui du *Token Bucket*) ;

– un marquage « hors profil » des paquets « opportunistes » (dans le cas du service AS), ou le rejet de ces paquets (dans le cas du service GS) en cas de non conformité du trafic ;

– une classification par flux (ou de type MF : *multi field*) à partir de plusieurs champs des paquets IPv6, comme par exemple l'identificateur de flot et l'adresse source (trois files d'attente sont définies) ;

– un ordonnancement couplant là encore mécanismes de priorité et de WFQ.

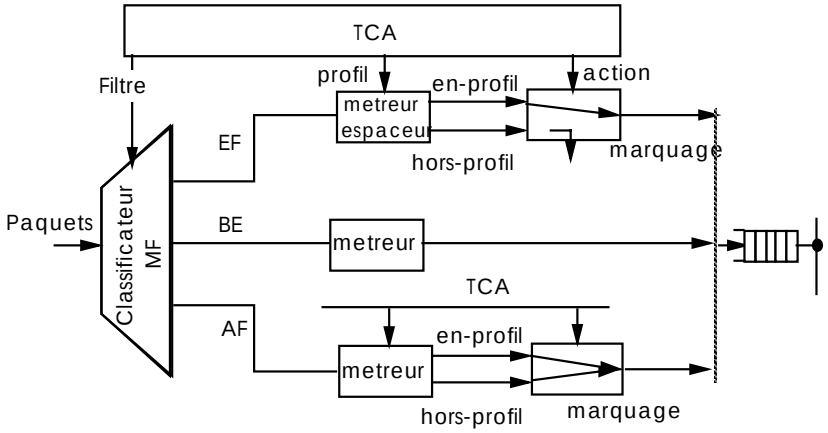

Figure 7 : *Eléments fonctionnels d'un routeur de bordure*

Etudions ces mécanismes plus en détails dans le cadre de chacun des trois services offerts au niveau IP.

3.2.3. Cas du service AS

Tel qu'introduit dans la section précédente, les principaux mécanismes mis en œuvre au niveau des routeurs de bordure et/ou de cœur pour offrir un service AS sont les suivants (se référer à la figure 5 pour illustration graphique).

a) Contrôle d'admission

Dans le cas du service assuré, le contrôle d'admission n'est envisagé qu'en bordure, c'est à dire au niveau des sites émetteur et récepteur(s), via hôte et routeur de bordure « émetteurs » d'une part, hôte(s) et routeur de bordure « récepteurs » de l'autre[5].

[5] Il n'est pas exclu (hors implémentation) de définir une signalisation plus complexe visant à affiner les caractéristiques du service fourni (QoS définie de façon plus précise qu'elle ne l'est actuellement).

La signalisation envisagée pour la mise en œuvre de ce contrôle repose sur l'utilisation d'une version allégée du protocole RSVP. Les messages RSVP échangés entre hôtes émetteur et récepteur(s) ne sont interprétés que par les routeurs de bordure de façon à pouvoir accepter ou refuser une requête applicative (portant donc sur un flux) en fonction :

– pour les flux sortants, de la connaissance de la bande passante encore disponible au regard du TCA local ;

– pour les flux entrants, de la connaissance de la bande passante encore disponible au sein du site. L'hypothèse faite sur ce point dans le cadre du projet est le sur provisionnement des sites en bande passante.

Conformément à l'approche DiffServ, l'hypothèse sous-jacente est que le réseau (l'ISP) est suffisamment bien provisionné pour satisfaire un service assuré sous la seule condition que les différents TCA soient chacun respectés par les routeurs de bordure. Toutefois, le service AS ayant été défini de façon à absorber un débit de paquets « opportunistes » (c'est à dire excédant le contrat initial tant qu'il n'y a pas de congestions dans le réseau), un contrôle de congestion s'avère nécessaire. Etudions tout d'abord le conditionnement du trafic aux bornes du réseau.

b) Conditionnement

Le conditionnement du trafic intervient au niveau des routeurs de bordure. Ceux ci sont en particulier chargés de marquer les paquets relatifs à des flux applicatifs sortants, en tenant compte du contrat de trafic établi pour chaque flux lors du contrôle d'admission (plus précisément, la partie débit moyen du *Token Bucket* annoncé par l'application pour le flux). Les paquets conformes au *Token Bucket* sont marqués AF « dans le profil » (IN), les paquets non conformes sont marqués AF « hors profil » (OUT) par le biais du champs DSCP des paquets IPv6. Contrairement à la classe GS (et en l'absence de congestion), tous les paquets émis par l'application sont injectés dans l'ISP par le routeur de bordure.

Le conditionnement pour le service AS consiste alors à :

– identifier et classer les flux à servir en AS :

– marquer les paquets et indiquer leur priorité à l'écartement (IN ou OUT);

– faire respecter le caractère réactif des flux, c'est à dire vérifier qu'il y a bien baisse du débit du trafic OUT lorsque la connaissance de l'existence d'une congestion sur la route du trafic a été signalée à l'application. En l'absence de réaction de la source, le conditionneur limite le trafic émis au débit minimum garanti et l'excès de trafic est supprimé.

c) Contrôle de congestion

En l'absence de contrôle d'admission à l'intérieur de l'ISP, la réactivité des flux assurée par le contrôle de congestion est fondamentale car garante d'une bonne exploitation du réseau en terme de trafic utile écoulé [21]. Ce contrôle concerne les paquets opportunistes.

Rappelons que pour un transfert avec le service AS, le débit minimum doit être garanti. Par conséquent, il ne doit pas y avoir de congestion suffisamment importante pour perturber les trafics dans leur débit minimum. Cette condition doit rester vraie si le provisionnement du service est suffisant par rapport à sa souscription.

Le principe du contrôle est le suivant. Sur occurrence d'une congestion dans un routeur de cœur, celui entame deux actions :

– il applique un mécanisme de gestion des pertes permettant d'éliminer prioritairement les paquets opportunistes. Le document [22] recommande d'utiliser des mécanismes à partage probabiliste tel que RED (*Random Early Detection*) pour gérer les pertes. Cependant des travaux récents [23] ont montré les défauts introduits par ce type de mécanisme, comme par exemple les oscillations du niveau de la file d'attente. Les mécanismes à base de seuil statique tel que PBS (*Partial Buffer Sharing*) [20] pourraient servir de base à des politiques de discrimination efficaces. C'est celle que nous avons retenue dans @IRS ;

– il en informe l'application « propriétaire » du flux hors profil afin que celle-ci adapte son débit en conséquence, faute de quoi les paquets hors profil (OUT) seront rejetés par le routeur de bordure et non transmis dans l'ISP. La notification est également interprétée par le routeur de bordure par lequel proviennent les paquets qui subissent une congestion afin qu'il puisse effectuer le contrôle de réactivité de l'application. Ce mécanisme, basé sur une notification explicite des congestions, est actuellement en cours d'étude dans le cadre du projet @IRS.

3.2.4. Cas du service garanti

Dans le cas du service garanti, la signalisation implique les hôtes et les routeurs de bordure émetteurs et récepteurs, mais également les routeurs de cœur. En d'autres termes, les messages RSVP échangés lors du contrôle d'admission entre hôtes émetteur et récepteur sont interprétés par les routeurs de bordure et par les routeurs de cœur.

Le service GS est défini pour garantir un délai le plus faible possible (proche du temps de propagation) et donc une gigue limitée. Cette garantie est obtenue par mise en œuvre :

– d'un mécanisme de priorité stricte des paquets de la classe GS sur les paquets des classes AS et BE ;

– d'un contrôle d'admission impliquant les routeurs de bordure et les routeurs de cœur, permettant d'assurer que la probabilité d'attente d'un paquet de la classe GS dans une file soit négligeable.

Comme précédemment, se référer à la figure 5 pour illustration graphique.

a) Contrôle d'admission

Le contrôle d'admission a pour but d'assurer qu'à tout instant, le débit total généré par l'ensemble des flux de la classe GS traversant un lien n'excède pas la capacité de ce lien. C'est ce qu'on appelle couramment le « multiplexage à enveloppe de débit » [24]. Dans l'architecture IntServ par exemple, ce multiplexage est réalisé par une réservation explicite des ressources au moyen du protocole RSVP.

Dans l'architecture @IRS, nous utilisons une version allégée du protocole RSVP pour véhiculer les messages échangés entre les routeurs de bordure et les routeurs de cœur lors du traitement d'une requête. Notons qu'aucune réservation de ressource n'est effectuée à l'intérieur de l'ISP.

Le contrôle d'admission se fait de manière distribuée, chaque routeur prenant la décision, lors de la réception d'une requête, d'accepter ou de rejeter cette requête en fonction de deux paramètres :

– le débit crête annoncé par l'application pour le flux ;

– une valeur maximale du débit de paquets de la classe GS traversant le lien (cette valeur étant strictement inférieure à la capacité du lien par l'administrateur du réseau).

Recevant une requête via un message RSVP :

– un routeur de bordure (comme dans le cas du service AS) accepte ou refuse une requête applicative en fonction :

– pour les flux sortants, de la connaissance de la bande passante encore disponible au regard du TCA local ;

– pour les flux entrants, de la connaissance de la bande passante encore disponible au sein du site (l'hypothèse faite sur ce point dans le cadre du projet étant le sur provisionnement des sites en bande passante).

– un routeur de cœur effectue une mesure du débit « courant » en trafic GS (sur la minute précédente par exemple) ; si le débit crête requis additionné au débit courant n'excède pas la valeur maximale fixée par l'administrateur, la requête est acceptée, sinon elle est rejetée. La requête est acceptée si aucun des routeurs (bordure et cœur) sur la route empruntée par le flux ne la rejette.

Le premier avantage de ce type de contrôle d'admission est qu'il ne nécessite pas le maintien d'une table d'état des flux dans les routeurs de cœur :

– seuls les routeurs de bordure maintiennent une table d'état des flux en cours permettant ensuite (et en particulier) de vérifier la conformité du trafic sortant ;

– cette table est rafraîchie périodiquement pour effacer les entrées correspondant à des flux inactifs. Il n'y a donc pas de message explicite de fin de session RSVP.

Comparativement à l'approche IntServ, les autres avantages de cette architecture sont :

– une signalisation allégée : les messages RSVP sont à la fois plus simples à traiter et moins nombreux que dans le cas de l'architecture IntServ. En particulier, la signalisation RSVP est suspendue une fois le contrôle d'admission effectué (les messages RSVP ne sont pas générés périodiquement), de sorte que les messages de contrôle ne représentent qu'une faible proportion de la bande passante ;

– une signalisation unidirectionnelle : le contrôle d'admission se fait à la demande du routeur amont vers le routeur aval par le biais d'un message RSVP « PATH ». La décision d'acceptation ou de rejet est directement renvoyée au routeur amont (via un message « RESV »), et donc ne passe pas nécessairement par la même route que le message PATH initial.

Cette simplification présente a priori un coût en terme de garantie de QoS : les garanties fournies par un tel mécanisme de contrôle d'admission semblent moins strictes que celles fournies par un mécanisme de réservation de type IntServ. En réalité, ce coût concerne davantage l'administrateur du réseau qui a pour charge supplémentaire de définir un jeu de paramètres (période de rafraîchissement des tables, débit crête maximal des flux, lissage des flux, etc.) permettant d'obtenir une QoS équivalente à celle fournie par un mécanisme à réservation.

b) Conditionnement

En phase de transfert de données, le routeur de bordure doit vérifier que le flux entrant respecte les caractéristiques annoncées lors du contrôle d'admission, c'est à dire que son débit crête reste inférieur au débit crête annoncé lors de l'appel, et que son profil de trafic (*Token Bucket*) respecte celui annoncé par l'application lors de l'invocation du service (cf. section 2.3.3 sur l'API).

Ceci peut se faire au moyen d'un *espaceur* de paquets IP par exemple, qui n'est autre qu'une file d'attente de taille finie, la vitesse du serveur correspondant au débit crête souhaité. Les paquets acceptés dans la file sont marqués EF. Les paquets refusés sont rejetés, et donc non transmis.

c) Fixation de la route

Des changements de route en cours de session peuvent engendrer une dégradation de la QoS sur la nouvelle route car le contrôle d'admission n'est effectué que sur la route initiale. Face à ce problème, une solution consiste à figer la route sur laquelle le contrôle d'admission a été effectué.

La commutation par label est un moyen simple de stabiliser une route. Dans [25], il est proposé d'utiliser la signalisation RSVP pour transporter les informations sur les labels associés aux flux, d'où la notion de *RSVP switching*. Le label d'un flux trouve naturellement sa place dans le champ *flow label* de l'en-tête IPv6. L'établissement du chemin commuté (*cut-through*) a lieu en même temps que le contrôle d'admission, chaque routeur indiquant au routeur qui le précède le label assigné au flux, au moyen d'un message qui peut être un message RSVP RESV par exemple.

La commutation par label est effectuée par les routeurs de cœur. Ceux-ci doivent maintenir une table d'état des labels, qui est rafraîchie périodiquement comme la table d'état des flux des routeurs de bordure. Ceci impose qu'une activité soit maintenue par le flux pour conserver sa commutation. Ce mode de gestion des états « mous » (*soft state*) est décrit dans le document [26]. Notons que l'état du flux associé à la commutation est indépendant des fonctions de QoS. Son rafraîchissement ne met pas en œuvre de messages explicites.

3.2.5. Cas du service best effort

Le service BE utilise la bande passante disponible, non utilisée par les classes AS et GS. Notons qu'une fraction fixe de la bande passante non utilisée par la classe GS est réservée à la classe BE par l'ordonnancement WFQ, ce qui assure que ce service est toujours disponible. La valeur précise de cette fraction de bande passante réservée au service BE est laissée à la charge de l'administrateur.

4. Conclusions et perspectives

Les travaux présentés dans cet article ont porté sur la conception d'une architecture de communication à QoS garanties en environnement IP à service différenciés. L'objectif de l'étude était d'envisager la mise en œuvre d'applications distribuées présentant des contraintes (en particulier temporelles) en terme de communication.

L'architecture proposée vise à offrir une QoS de bout en bout « par flux » applicatif en s'appuyant sur une QoS de niveau IP « par classe » de trafic. La motivation sous-jacente à cette proposition est double :

– une application (multimédia par exemple) manipule plusieurs types de flux (audio et vidéo par exemple) présentant chacun des besoins spécifiques en terme de communication : le choix a donc été fait de permettre aux applications de disposer, via une API générique, d'un canal de bout en bout à QoS spécifique au flux véhiculé ;

– à des fins de mise à l'échelle et de simplicité de conception, la solution qui tend à se dégager à l'heure actuelle dans l'Internet (vis à vis du problème de la QoS) est de traiter non pas des flux mais des classes de trafic, dans le cadre d'un nombre limité de services « différenciés ».

Dans cet article, nous avons proposé les QoS et sémantiques de garantie associées, à la fois de bout en bout et de niveau IP. Nous avons également proposé une façon de procéder pour établir le lien entre QoS de bout en bout et QoS de niveau IP dans le cadre d'une plate-forme expérimentale représentative (celle retenue pour le projet @IRS). Actuellement en cours d'implémentation, les différents éléments de l'architecture seront testés et évalués dans les prochains mois, les applications ciblées étant la simulation interactive distribuée (DIS) d'une part et une application de contrôle/commande de l'autre. Certains points (traités dans le cadre du projet) n'ont volontairement pas été abordés dans cet article, faute de place. Au niveau IP, nous n'avons par exemple pas explicité la gestion du *multicast* : précisons simplement que la solution retenue impose que tous les récepteurs d'une même communication *multicast* reçoivent la même QoS IP. De même, nous n'avons pas explicité la plage des services effectivement accessible (c'est à dire dont la mise en œuvre est possible) par l'application. Ces différents points constituent les perspectives immédiates des travaux ici présentés.

5. Références

[1] IETF : Integrated Service Working Group: http://www.ietf.org/html.charters/intserv-charter.html.

[2] IETF : Differentiated Service Working Group: http://www.ietf.org/html.charters/diffserv-charter.html.

[3] Contrôle de ressources en environnement hétérogène. Rapport final du projet DIS/ATM. (conv. n°97/73-291). Dassault Electronique Ed. Décembre 1998

[4] Chassot C, Lozes A, Garcia F, Dairaine L, Rojas L. Specification and realization of the QoS required by a DIS application in a new generation environment. Interactive Distributed Multimedia Systems and Telecommunication Services, IDMS'99, LNCS 1718, 6th Int. Workshop, Toulouse, France, October 1999.

[5] Chassot C, Lozes A, Diaz M, Dairaine L, Rojas L. QoS requise par une application de DIS distribuée dans un environnement réseau grande distance. 7e Colloque Francophone sur l'Ingénierie des Protocoles (CFIP'99) Nancy, Avril 1999.

[6] Shenker S, Partridge C, Guerin, R. Specification of Guaranteed Quality of Service, RFC 2212, IETF Network WG (1997).

[7] Wroclawski J. Specification of the Controlled Load Network Element Service, RFC 2211, IETF Network WG (1997).

[8] Braden R, Zhang L., Berson S, Herzog S, Jamin S. Resource reSerVation Protocol : Functional Specification, RFC 2205, IETF Network WG (1997).

[9] Wroclawski J. The use of RSVP with Integrated Services, RFC 2210, IETF Network WG (1997).

[10] Chassot, C, Diaz, M, Lozes A. From the partial order concept to partial order multimedia connection, Journal for High Speed Networks, Vol. 5, n°2 (1996).

[11] Campbell A., Coulson G., Hutchinson D. A quality of service architecture, ACM Computer Communication Review (1994).

[12] Nahrstedt K., Smith J. Design, Implementation and experiences of the OMEGA end-point architecture, IEEE Journal on Selected Areas in Communications, Vol.14 (1996).

[13] Gopalakrishna G., Parulkar G. A framework for QoS guarantees for multimedia applications within end system, GI Jahrestagung, Zurich, Switzerland (1995).

[14] Fournier M. Réalisation et évaluation de performances d'un protocole de transport multimédia à ordre partiel. Thèse de Doctorat de l'Université Paul Sabatier. Mars 1997.

[15] Nichols K, Jacobson V, Zhang L. A Two-bit Differentiated Services Architecture for the Internet, November 1997.

[16] Blake S; Black D, Carlson M. RFC: 2475. An Architecture for Differentiated Services, 1998.

[17] Deering S, Hinden R. Internet Protocol Version 6 (IPv6) : Specification. RFC 2460. December 1998.

[18] Jacobson V, Nichols K, Poduri K. An Expedited Forwarding PHB. RFC 2598. June 1999.

[19] Heinanen J; Baker F, Weiss W, Wroclawski, J. Assured Forwarding PHB Group. RFC 2597. 1999.

[20] Bonald T, May M, Bolot J. Analytic Evaluation of RED Performance. Infocom'2000.

[21] Floyd S, Fall K. Promoting the Use of End-to-End Congestion Control in the Internet. IEEE/ACM Transactions on Networking, Vol. 7, No. 4, pp. 458-472, August 1999.

[22] Braden B and al. RFC 2309. Recommandations on Queue Management and Congestion Avoidance in the Internet.

[23] Roberts J.W. Performance evaluation and design of multiservice networks. Coll.: Information technologies and sciences. Cost 224, pp. 139-140. Performance evaluation and design of multiservice.

[24] Roberts J.; Mocci U. and Vitramo J. (1996). Broadband Network Teletraffic. Rapport final de l'action COST 242. Springer

[25] Fourmaux O. Les communications Multipoints dans les Réseaux Haut Débit Multimédia : Le Multicast en Environnement IP sur ATM. Thèse de Doctorat, UPMC, Décembre 1998.

[26] Partridge C. Using the Flow Label Field in IPv6. RFC 1809. June 1995.